Project Number :	**IST-1999-10561-FAIN**
Project Title :	**Future Active IP Networks**

Requirements Analysis & Overall AN Architecture

CEC Deliverable Nr :	**WP2-DT-004-D01-Int**
Deliverable Type :	**PU**
Dissemination:	**Pub**
Deliverable Nature :	**R**
Contractual date :	**December 31, 2000**
Actual date :	**May 30, 2001**

Editor :	**Christian Garbrecht**
Workpackage(s) :	**WP2**

Abstract : This document present the results of the FAIN workpackage WP2 ("Requirements Analysis and Overall AN Architecture"). It is centered on an Enterprise Model for Active Networks, which defines the involved actors, the roles these actors play and the relationships between these roles.

Operators' expectations on Active Networks are included in order to guide all work in the workpackage.

Requirements for the FAIN system and architecture are derived by following an application-oriented approach. For this purpose, three applications, which benefit from active network are modelled by use cases and mapped onto the enterprise model: Web Service Distribution, Reliable Multicasting and Virtual Private Networks.

These applications, the operators' expectations as well as results from related projects are collected into a list of requirements. Finally, taking into consideration the list of requirements the various reference points of the EM are described.

Keyword List : Active Network, AN Requirements, AN Architecture, Enterprise Model, Business Model, Reference Point, Active Application, Multicasting, VPN, CNM, Active Web Service, Content Distribution, Use Cases

Project Number :	**IST-1999-10561-FAIN**
Project Title :	**Future Active IP Networks**

Requirements Analysis & Overall AN Architecture

Editor :	**Christian Garbrecht**
Document No:	**WP2-DT-004-D01-Int**
File Name	**WP2-DT-004-D01-Int.doc**
Contributors :	
Version :	**1.0**
Date :	**Wednesday, 30 May 2001**
Distribution :	**WP2**

The FAIN Consortium consists of:

Partner	Status	Country
UCL	Partner	United Kingdom
JSIS	Associate Partner to UCL	Slovenia
NTUA	Associate Partner to UCL	Greece
UPC	Associate Partner to UCL	Spain
DT	Partner	Germany
FT	Partner	France
KPN	Partner	Netherlands
HEL	Partner	United Kingdom
HIT	Partner	Japan
SAG	Partner	Germany
ETH	Partner	Switzerland
GMD	Partner	Germany
IKV	Associate Partner to GMD	Germany
INT	Associate Partner to GMD	Spain
UPEN	Partner	USA

The FAIN Consortium

University College London (UCL)

Josef Stefan Institute	(JSIS)
National Technical University of Athens	(NTUA)
Universitat Politecnica De Catalunya	(UPC)
T-Nova Deutsche Telekom Innovationsgesellschaft mbH	(DT)
France Télécom / R&D	(FT)
Koninklijke KPN NV, KPN Research	(KPN)
Hitachi Europe Ltd.	(HEL)
Hitachi Ltd.	(HIT)
Siemens AG	(SAG)
Eidgenössische Technische Hochschule Zürich	(ETH)
GMD Forschungszentrum Informationstechnik GmbH	(GMD)
IKV++ GmbH Informations- und Kommunikationstechnologie	(IKV)
Integracion Y Sistemas De Medida, SA	(INT)
University of Pennsylvania	(UPEN)

Project Management

Alex Galis
University College London
Department of Electronic and Electrical Engineering,
Torrington Place
London WC1E 7JE
United Kingdom
Tel +44 (0) 207 458 4463
Fax +44 (0) 207 388 9325
E-mail: a.galis@ee.ucl.ac.uk

Authors

Christian Garbrecht (DT) – Editor
Cornel Klein (SAG) – Workpackage Leader
Alex Galis (UCL) – Project Coordinator
Arso Savanovic (JSIS)
Chiho Kitahara (HIT)
Drissa Houatra (FT)
Ermolaos Zimboulakis (NTUA)
Evelyn Pfeuffer (SAG)
Fawzi Daoud (GMD)
Hui Guo (GMD)
Jens Meinköhn (DT)
Jan Laarhuis (KPN)
Juan Luis Mañas (INT)
Jürgen Dittrich (IKV++)
Kiminori Sugauchi (HIT)
Matthias Bossardt (ETH)
Mercedes Urios (INT)
Peter Graubmann (SAG)
Spyros Denazis (HEL)
Wilson Lim (UCL)
Yiorgos Bouloudis (NTUA)
Yiannis Nikolakis (NTUA)

Executive Summary

This document is the deliverable D1, which has been produced within workpackage 2 („Requirements Analysis and overall AN Architecture“) of the FAIN project, and which summarises the results of this workpackage.

The work in FAIN WP2 and hence this document is centred on an enterprise model (EM) for active networks. This EM defines the involved actors, the roles these actors play and the relationships between these roles. Of particular importance in FAIN are the „Network Infrastructure Provider“, the „Active Network Service Provider“ and the „Service Provider“ who provides active services to its customers, the „End Users“, based on service components (i.e. code), which he obtains from a role called the „Service Component provider. Although similar approaches are used in related work (e.g. in TINA), the FAIN EM is the first one designed explicitly for active networks.

The document is structured as follows: after an introduction to the deliverable, operators' expectations on active networks are explicitly stated. These expectations are intended to guide all work in the remaining parts of the deliverable. After that, in chapter 3, the enterprise model is motivated, introduced and demonstrated by a simple example.

Requirements for the FAIN system and architecture are derived by following an application-oriented approach. For this purpose, three applications, which benefit from active network are modelled by use cases and mapped onto the enterprise model: Web Service Distribution, Reliable Multicasting and Virtual Private Networks. Besides being convincing applications for the usefulness of active networking technology, which will be implemented in demonstration scenarios later on, it is expected that by following this application-centred approach the resulting architecture will support a large variety of applications.

These applications, the operators' expectations as well as activities in fora like the MSF are collected into a list of requirements. Finally, taking into consideration the list of requirements the various reference points of the EM are described. The level of completeness and detail in this descriptions was dependent not only on the resources spent for the workpackage, but also on the results achieved so far and on the level of consciousness.

The deliverable ends with a conclusion, which outlines continuation of the workpackage 2 activities in other workpackages in FAIN.

The following internal reports were used in the production of the deliverable: R0-Glossary , R1 – Enterprise Model (for chapter 2, 3, 6), R2-Use Cases (for chapter 4), R34- Active applications requirements (for chapter 5).

Change History

1. Initial Draft May 21, 2001. Version 0.1
2. Integration of all section in a Word central document
3. Integration of updates for section 3, section 4.1, section 6.1
   Integration of section 7
   Change text format of all bullet lists
   Add authors to the list of authors
4. Integration of all sub-documents
5. Executive Summary, Conclusion; Comparision of AN and other BMs; reformatting IDL-descriptions
6. Final Versoin May 29, 2001.

Table of Contents

Table of Figures

1 INTRODUCTION

The objectives and an initial topic list of the FAIN deliverable D1, which is the result of FAIN Workpackage 2 (Requirements Analysis and Overall AN Architecture), have already been given in the Technical Annex (TA) of the FAIN project [FAIN-TA].

In this introduction, we will interpret the WP2 objectives in the light of technical progress and the gained knowledge since the preparation of the TA. The introduction is structured as follows: In section 1.1, we will give a motivation for requirements engineering. In section 1.2, we outline how this generic process and deliverable structure map into the corresponding entities in FAIN, and we will give an overview of the deliverable D1. In section 1.3, we will restate the objectives of WP2. For the purpose and scope of the specified system, we refer to the technical annex [FAIN-TA].

1.1 Requirements Engineering

It is generally agreed that any successful system development process should begin with a detailed phase of requirements engineering (RE). RE aims at defining the purpose of a proposed system and specifying its external behaviour.

1.1.1 Results of Requirements Engineering

The final product of requirements engineering is normally a document called Requirements Specification (RS), i.e. a document, which describes *what* a system is intended to do and *not how* it should do it. A requirements specification must be internally consistent, correct and complete in relation to the needs of the different stakeholders. It must be clear and understandable both to users, customers, designers, and testers and capable of serving as a basis for both design and testing procedures.

Requirements Engineering is concerned with what needs to be designed rather than how it is to be designed. Requirements Engineering is also concerned with some future situation. Traditionally, requirements engineering has been seen as the first phase of the software life cycle in which a specification is produced from informal ideas. During RE, the functional and non-functional requirements to be met by the system, as well as the criteria for measuring the degree of their satisfaction, must be elicited and documented in a *requirements specificatio*n.

The requirements analysis activity is aimed at understanding the problem and how the proposed solution will address it. There are a number of definitions of the term ‘requirements’, the most notable being IEEE Standard 610 [IEEE]:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
3. A documented representation of a condition or capability as in 1 or 2.

1.1.2 Process of Requirements Engineering

The activities conducted during requirements engineering can be divided into five categories:

- *requirements elicitation* which is the process of exploring, acquiring, and reifying user requirements through discussion with the different stakeholders.
- *requirements modelling* where a model for the target system is produced, as well as a conceptual model of the system context as seen by the different stakeholders. This model is meant to capture as much of the semantics of the real world as possible and is used as the foundation for an abstract description of the requirements.

- *requirements specification* where the various components of the models are precisely described and possibly formalised to act as a basis for contractual purposes between the problem owners and the developers.
- *requirements validation* where the specifications are evaluated and analysed against correctness properties (such as completeness and consistency), and feasibility properties (such as cost and resources needed).
- *requirements management* refers to the set of procedures that assists in maintaining the evolution of requirements throughout the development process. These include planning, traceability, impact assessment of changing requirements and so on.

Errors in requirements specifications have a large effect on the software costs. It is evident that early detection and correction of potential problems during requirement analysis may alleviate much larger problems later on during testing and maintenance. Already two decades ago, Boehm [Boe81] claimed that by investing more up-front effort in verifying and validating the software requirements and design specifications, software projects may harvest the benefits of reduced integration and test costs, higher software reliability and maintainability.

1.2 Requirements Engineering within FAIN

In this section, we interpret the generic description of RE process and RE result, as it has presented in the previous section, onto the FAIN project.

1.2.1 Process in FAIN

The generic activities of requirements engineering can be mapped into the activities within FAIN as follows:

- *Requirements elicitation:* Requirements elicitation is done by presentations of the expectations of the different stakeholders (e.g. Operators' expectations from AN) and discussions of applications/use cases.
- *Requirements modelling*: Requirements modelling is done by building a conceptual model of the system context in the form of a Glossary, as well as building an architectural model of FAIN.
- *Requirements specification:* Requirements specification is done by investigating some selected use cases/applications in more detail and deriving from them the interfaces of the involved system components. In addition, requirements for the FAIN system are derived from the operators' expectations and the application.
- *Requirements validation:* Before the final deliverable D1 is produced, the requirements specification will be reviewed both internally as well as externally (e.g. in co-operation with WP3 in order to discuss costs/feasibility issues)
- *Requirements management:* WP2 will end after this deliverable has been completed. If necessary, FAIN may be redefined in order to renter RE activities in a later stage.

In particular, FAIN takes a use-case oriented approach to requirements elicitation and documentation, as it is described for instance in [SW99]. With respect to the process, this is reflected in the following approach:

- The network operators, which are working in FAIN, have been asked to state their high-level expectations on active networks. Such operators may act in the roles of network infrastructure provider, network service provider or service provider (see Chapter 3 for a definition of these roles in the FAIN enterprise model). These "operators expectations" yield both requirements as well as that they give guidance when some design decisions have to be made.

- In addition, applications for active networks have investigated and modelled by use cases. These use cases show typical interaction scenarios between the FAIN system, its subsystems and its users. The users in such scenarios correspond mainly to the roles of the End User and the Service Provider in the FAIN enterprise model. From the interaction between these roles and the FAIN system, requirements for the different reference points are derived.

1.2.2 Results in FAIN

Requirements Engineering in FAIN WP2 should produce the following documents.

- A business model, capturing the involved players (actors, roles) and the relationships between them.
- Functional and non-functional requirements on the system.
- In the case that the system is required to be structured into subsystems, reference point specifications between the different subsystems.
- Known platform or architectural constraints: Machines, operating systems, distribution, middleware, legacy systems, and interoperability requirements, all captured by a package structure and collaborations. In FAIN, these are to be investigated within WP3/WP4.
- Project and planning constraints: Budgets, schedules, staff, and user involvement. In FAIN, these are already included in the technical annex [FAIN-TA].
- For FAIN WP2, all these documents are aggregated within the deliverable D1.

The document is structured as follows:

Chapter 1: Introduction

The introduction covers the following topics:

- Introduction to Requirements Engineering: Motivation for Requirements Engineering and an overview over the processes for requirements engineering and the produced results.
- Requirements Engineering in FAIN (this section): Relates the general statements about requirements specification to FAIN WP2.
- Objectives of Workpackage 2: Restate the objectives of the workpackage

Chapter 2: Operator's Expectations on Active Networks– part of the internal report R1

Active networks are operated by operators, which want to make money out of them by exploiting their potential flexibility for the benefit of both the operator and the end-user. In this chapter, keys expectations, which are relevant for that purpose, are addressed, in the ability for

- rapid deployment of new services;
- customisation of existing service features;
- scalability and cost reduction in network and service management;
- independence of network equipment manufacturer;
- information network and service integration;
- diversification of services and business opportunities.

Chapter 3: Active Networks Enterprise Modelling– part of the internal report R1

Chapter 3 introduces the FAIN Enterprise Model (EM) for Active Networks, while the detailed description of its reference points is given in Chapter 6. It is structured as follows:

- Introduction to Business Models: Gives a motivation for introducing the FAIN Enterprise Model

- FAIN Enterprise Model: An Introduction to the FAIN Enterprise Model (EM)
- Conclusions: An outlook and a comparison of the FAIN EM with related approaches.

Chapter 4: Applications and Use Cases- – part of the internal report R2

Chapter 4 contains several applications, which benefit from the features offered by active networks. These applications are modelled by use-cases, which are modelled down to the level of subsystem interactions. Based on these interactions, requirements for the different reference points (RPs) are derived. The applications we consider are:

- Active Web Service
- Reliable Multicasting
- Virtual Private Networks

A short description of these applications is contained in the introduction to Chapter 4.

Chapter 5: Requirements- – part of the internal report R34

In Chapter 5, the requirements of the proposed system are systematically written down. These requirements are set into relation to the different applications they are derived from and to the operator's expectations.

Chapter 6: Reference Point Specifications – part of the internal report R1

In this chapter, the different reference points, i.e. the interfaces between the subsystems of FAIN network, are specified in various degrees of detail.

Chapter 7: Conclusion

In the conclusion, we summarise our achievements and give an outlook on future continuation activities within FAIN WP3 and WP4.

Appendix

The Appendix contains an overview over existing enterprise models in the telecom domain. In addition, it contains a Glossary defining the WP2 core terms. This glossary is an excerpt of the FAIN Glossary, which is submitted as a separate document as internal report R0.

1.3 Objectives of the WP

In the last decade the business world witnessed the rapid development of Internet and IP networks in private and corporate areas. The wide acceptance of IP originated from its unparalleled ability to provide ubiquitous access and low prices regardless of underlying networking technologies. Moreover, based on the existing best-effort IP transport service, new *application services* can be offered on a global scale by almost everyone, simply by connecting a new web server to the Internet. Today IP is considered as unique glue to bridge diverse application/user requirements with broadband transfer capability. Various research initiatives such as NGI (Next Generation Internet), CANARIE, Internet2, etc., are progressing to provide unlimited bandwidth for Internet users. In parallel, based on the conventional Internet architecture, IETF (Internet Engineering Task Force) is performing a "bottom-up" development of Internet protocols and techniques, to fulfil upcoming requirements from applications, users, and providers.

However developing and deploying new network services, i.e. services which operate on the IP layer, through best practice and standardisation is too slow, and cannot match the steps in which requirements of various applications, e.g. multimedia multiparty communication, are growing. Examples of such services are signalling for quality of service (QoS), reliable multicast or Web Proxies/Caches/Switches/Filters. Similar to the intelligent network (IN) architecture in the PSTN world, the current Internet architecture needs to be enhanced in order to allow for a more rapid introduction of such services.

Active Networks (AN) have been proposed as a solution for the fast and flexible deployment of new network services. The basic idea of active networks is to enable third parties (end users, operators, and service providers) to inject application-specific services (in the form of code) into the network. Applications are thus able to utilise these services to obtain required network support in terms of, e.g. performance, that is now becoming network-aware. As such, active networks allow *dynamic injection of code* as a promising way to realise application-specific service logic, or perform dynamic service provision on demand. But the dynamic injection of code can only be acceptable by network providers if it does not compromise the integrity, the performance and /or the security of networks.

Therefore viable architectures for active networks have to be carefully engineered to achieve suitable trade-offs among flexibility, performance, security and manageability. Besides these technological challenges, which have to be solved, such architectures have to be carefully defined in order to be able to fulfil requirements as they stem from a business-oriented view on active networks. To our knowledge, this aspect has been neglected in existing work in AN, which has mainly been technology driven in a bottom-up way. It is the objective of WP2 to elicit such requirements. In particular, requirements are derived from two sources:

- Business-oriented expectations from operators and manufacturers, yielding to an enterprise model and in particular to a high-level system architecture,
- Requirements, which are derived from, concrete applications and services and their realisation on top of the system architecture.

Please note that besides these formal activities and the formal, measurable results they produce (the "Requirements Specification" and the "Glossary"), and important role of RE is of in the social aspect of team building. This is of crucial importance in particular in a collaborative research project like FAIN, where many stakeholders are involved with different background, expectations, terminology and objectives. These different views have to be consolidated in order to achieve not only a formal result, but also to develop a common language to communicate and "system vision" in the minds of the involved persons.

1.4 References

[Boe81] Barry Boehm: Software Engineering Economics Prentice Hall, 1981.

[FAIN-TA] Technical Annex of IST project IST-1999-10561 "FAIN – Future Active IP Networks", FAIN consortium, March 2000.

[IEEE] IEEE Software Engineering Standards Collection. Institute of Electrical and Electronic Engineers, 1993.

[SW99] Desmond Francis D'Souza and Alan Cameron Wills. Objects, Components, and Frameworks with UML: The Catalysis Approach. Prentice Hall, 1999.

2 OPERATOR'S EXPECTATIONS ON ACTIVE NETWORKS

2.1 Introduction

Active Networks are networks in which the functionality of (some of) their network elements are dynamically programmable. With the latter it is meant that executable code is injected into that network element (for which several not to be mentioned methods exist) thereby creating the new functionality at run-time. Allowing this, active networks have the potential of unprecedented flexibility in telecommunications.

The key question from a (public network) operator's point of view is: how can we exploit this potential flexibility for the benefit of both the operator and the end-user? The answer lies in the promising potential that emerge with the advent of Active Networks of the following aspects:

- rapid deployment of new services;
- customisation of existing service features;
- scalability and cost reduction in network and service management;
- independence of network equipment manufacturer;
- information network and service integration;
- diversification of services and business opportunities.

The benefit of Active Networks on these aspects will be explained below. For each aspect, one or more real-life example is described to illustrate the potential benefit.

2.2 Speeding Service Deployment and Service Customisation

Currently, the deployment of new services and service features by the network operator in its network can only be carried out with consent of the equipment manufacturer due to the equipment's closed nature. Current network elements are closed in the sense that their functionality is 'baked in' during manufacture by the vendor. Any change in the functionality of a network element requires in general a lengthy standardisation-phase and technology diffusion phase before being operational.

The above situation results in undesirable delays when planning the deployment of new services/features. The Active Network approach, however, with its open nature based on standardised interfaces, will allow the network operator to freely select different service developers for implementing and deploying its new services/features according to market-driven requirements. Depending on the service design and implementation time, this can be done nearly on-demand. Interestingly, vendors could use the Active Network properties to upgrade the network elements with their latest network software release.

Besides the introduction of *new* services, Active Networks can advantageously be used to *modify* an existing service at run-time, too. This property is extremely useful for an operator to offer personalised services to a large extent to their customers. Notice that service customisation is not about configuring parameters of a service, but about modifying the behaviour of a basic resident service according to the requirements of the end-user. There are various ways to pass the user requirements and the modifying code to the appropriate nodes and resident services, e.g., by means of (router) plug-ins. An interesting example of service customisation is multicast. At active nodes there is a resident, basic multicast forwarding service. This service could be enhanced with for instance reliability, security, real-time.

2.3 Leveraged Network and Service Management

In this section we argue that Active Networks enable more feasible network and service management. Key mechanisms to achieve this are out-sourcing and distribution of management tasks (decisions). This directly results in better scalable solutions and leads to cost reductions. These aspects will be illustrated using the deregulated wholesale telecommunications market as an example. The example is therefore of special interest for incumbent operators.

From a regulator's point of view, ex-PNOs that now dominate the network provider market are forced to open their network infrastructure to third-party service providers on the basis of the EU ONP directive. This directive mainly addresses the traditional telephony networks (POTS) but its application to IT communications infrastructure in general is to be anticipated. In order to comply with the ONP directive a network operator will allow third-party service providers to use its network infrastructure under service provider-specific circumstances regarding e.g. accounting and billing. Due to the proprietary nature of current network infrastructure (hardware, protocols) the management of each third party service provider's specific requirements must remain with the network operator (the third party service provider has no access to network element management functions), resulting in a complex and costly overhead of operation and maintenance as there will be very large numbers of third party service providers whose network usage requirements must be individually supported by the network operator. These additional costs cannot be pushed to the third party service providers because of the ONP directive; also, the additional costs cannot be reflected in the network operator's pricing structure as this would lead to reduced competitiveness. The Active Network concept allows the network operator to delegate full management responsibility to the third party service providers, thereby complying with regulatory demands and simultaneously avoiding the management overhead mentioned above.

The existence of third party service providers (in addition to ex-PNO network operators) is a cost-effective differentiation of the standard value chain in providing IT communications and services. The cost argument outlined above is generally applicable to a network operator with proprietary network infrastructure in the face of potentially very large numbers of third party service providers. By means of an Active Network approach based on open, standardised protocols/interfaces to network management the network operator may achieve substantial cost reductions through delegation of management responsibility to the third party service providers.

2.4 Decreased Vendor Dependency

An important aspect of the Active Network concept is the fact that it will employ open, standardised interfaces. There is a fundamental difference of the Active Network concept with the current telecommunications paradigm. In Active Networks, the way functionality is to be implemented is prescribed (API); functionality itself is not prescribed. In the current paradigm the functionality itself is prescribed (protocols) while their implementation in network elements is left for vendor to choose. The important consequence of this paradigm shift is that any communicating entities don't have to adhere to the rigid communication patterns imposed by protocols. Instead, they have the ability to dynamically introduce the required functionality at network nodes by means of injecting executable code into the node.

Due to the Active Network concept, the network platform provisioning is more and more decoupled from the network software provisioning, ultimately leading to corresponding two separate network vendor types. On the other hand, but related to the previous, the Active Networks concept will make operators less dependent on a particular vendor. This is due to the increased competition of the vendor market where network software vendor A is developing innovative services/features for vendor B's network platform.

The operator's independence of a single network equipment manufacturer, as currently is the opposite case with proprietary network infrastructure, is very illustrative in the case of network (element) management. The network operator will be able to select different manufacturers of switches, routers etc. according to market-driven requirements, e.g. price, availability, functionality. Managing them collectively takes place via a standardised management interface that is downloaded on each network element.

2.5 Information network and service integration

Since about five years ago, a new paradigm is being adopted for the design, the deployment and the operations of telecommunication networks. Precisely, the paradigm shift concerns the use of distributed object-oriented platforms (CORBA, DCOM, JVM/RMI) and service frameworks as a new software basis for networks and information services. One of the major consequences of this new paradigm is to enable the integration of networks and information services within the same systems software platform. Until recently, networks and information services have almost always been designed, implemented as separate system, that are then integrated during their operations. For example, the existing (intelligent network) IN service control functions (SCF) and service management functions (SCF) are designed without a common conceptual framework, and they are implemented using different system supports (service nodes), both in terms of hardware and system service components. Usually, there is no service framework to facilitate the interactions between services running on SCF and SMF supporting systems. Therefore, the design and implementation of services that require interactions between SCF and SMF supporting systems is sometimes very difficult. With the demand on new services, requirements of this kind are becoming more important. Another example is the provision of services, that will allow a customer to access and manage its account from an Internet access: typically, customers would like to access their account from the web whenever they want, in order to acknowledge the cost of their communication, change some parameters and service options, and so on. The provision of this kind of services require interactions between IN service nodes and a web based information service provided by operators, a requirement that is sometimes hard to meet due to the structure of the existing networks and information services.

We expect Active Networks to facilitate the realisation of service platforms based on the new paradigm. The integration of information and network services in particular can be facilitated by Active Networks, by the definition, the implementation and use of standard open application programmable network interfaces. Beyond the needs of interactions between service components of the same operator, such interfaces can also be used for the interactions between service components that belong to different operators, and even between network operators and some of their clients, i.e. service providers. This second point is more related to the point described in 3, concerning the leverage of networks and the management of services. It is also related to, e.g. the provision of global and/or ubiquitous services. The potential impact of open and programmable interfaces provided by an active network on telecommunication networks and information services will be similar to that of socket interfaces and other Internet programming standards on the recent development of the Internet and the web.

2.6 Diversification of services and business opportunities

Due to the new regulatory environments and also to market demands, networks operators need to invent new services and push forward their traditional limits it terms of the types of services they can provide. The existing networks appear to have some limitations for new types of services. It is sometimes impossible in practice (usually very difficult) to provide some new services because of the structure of networks, or sometimes possible but wasteful in terms of network resources. For example, an operator can provide video-on-demand services with an acceptable QoS based on resource reservation, but problems will arise if the operator wants to extend the capability of its network to support more users or customers. Also, it will be very hard for a network operator to provide VPN services to client companies with varying QoS requirements in terms of reliability, dependability, timely delivery, scalability or security, a service profile that can be seen as a very important business opportunity by operators. This is also the case for policy-based network management, which stands for the management of networks taking into account QoS requirements specified by client organisations. Other examples such as the provision of e-commerce platforms, remote sensing and control operations for industrial plants are also business opportunities for network operators, and most of the case such services require variability and dynamics in QoS requirements.

The ability of Active Networks to adapt network services to the needs of target applications (by allowing application programs to use their own communication services instead, that can control and manage network resources adequately), and the use of application programming interfaces to access network elements, can be seen as key networking features, that will facilitate the diversification of services and enable more business opportunities.

3 ACTIVE NETWORKS ENTERPRISE MODEL

3.1 Introducing business models

Business models have been around since business exists! It is only recently that they have emerged as a favourite method to explain the success or the failure of a business. The transformation of the traditional client-vendor relationship to networked business models forces companies to look for new and effective business models to build or rejuvenate their profit and value generation mechanisms.

The evolving market for information services with its increased competitive pressure will enforce a further specialisation of the participating parties. Enterprises will seek advantage by concentration on their own core business sector leading to a further differentiation of roles and a formation of elaborate supplier chains. Typically some enterprises may direct business more towards end customer needs, while others will concentrate on the operation of large-scale high bandwidth networks.

Managing by business models is a very effective way to develop clear communications within a company as well as with its economic partners. Developing or analysing business models helps a company isolate where and how in its value chain it creates profit and value. With such understanding, a company can better discern how to exploit the potential of its markets, increase customer satisfaction, and raise barriers to entry from competitors. Using such an approach, a company has the tools to relentlessly improve upon its business practices to build sustainable long-term competitive advantage.

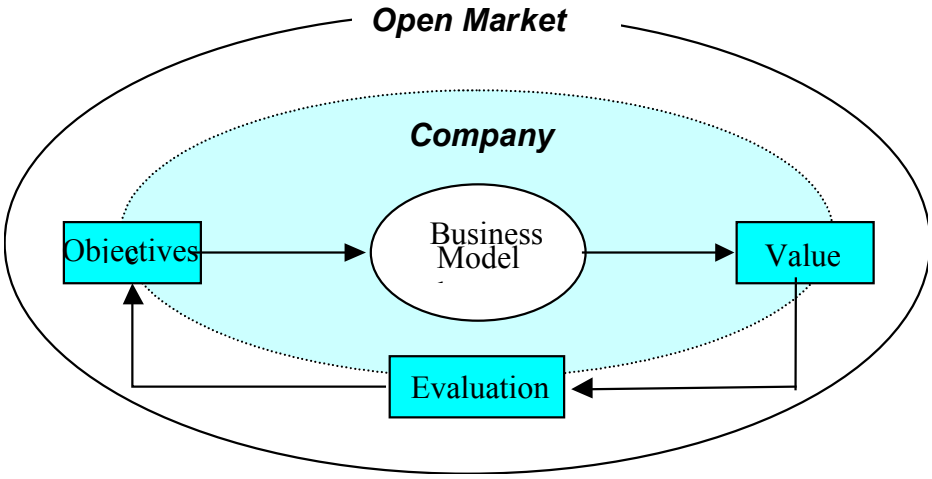


Figure 3-1 Business Modeling

The Business Model defines the actors, which roles these actors undertake, the relationship between the roles in terms of responsibilities and how to fulfil these responsibilities. Furthermore, the Business Model determines the rights pertaining to the different roles and the pre-requisites needed to execute these rights. The processes, roles, and responsibilities described in a business model could be defined in a business use-case model and a business object model (according to Rational Process Modelling).

The TINA reference Business Model is a specific example that aims at giving a structure which eases the application of TINA in a multi-stakeholder, multi-vendor environment. It does so by defining the interaction between business administrative do-mains in terms of contracts, business roles, business administrative domains and reference points. It also provides a framework for the definition of these reference points and a set of high level descriptions of the initial reference points.

3.2 FAIN Enterprise Model

3.2.1 Introduction

From an abstract point of view, an enterprise model consists of (1) actors, i.e., persons, companies, or organisations of any kind that own, use, support and/or administrate parts of an (active) networking system[1], (2) the various (business) roles these actors play, and (3) the contractual relationships between these roles, i.e., the reference points. The term "business role" refers to the specific behaviour, properties, and responsibilities assigned to an actor with respect to an identifiable part of the networking system and the related activities that are necessary to keep this part of the system operational. The set of (FAIN) actors is mapped onto the set of business roles occurring in the FAIN context, i.e., each role is understood to be played by at least one actor which may, however, perform more than one role. The relationships between roles are expressed as reference points which determine the interactions (interfaces and action sequences, together with related properties and constraints, granted rights, etc.) between the actors in the respective roles.

The FAIN actors are the "usual suspects": (a) the providers, i.e., companies operating networks (by managing the respective hardware and software resources) and offering services over the network (employing their own resources or those of the – traditional – network operators), etc.; (b) the users, i.e., individuals or companies which use the proffered connectivity and services; and (c) the manufacturing companies which support the providers by developing (and vending) the necessary hardware and software components. The manufacturers are not in the main focus of the FAIN context, even if their role is an important one and, as such, briefly reflected in the enterprise model, in order to make the picture complete.

The FAIN enterprise model only describes the business roles which may be taken by the above listed actor companies (and obviously, each actor company may act in more than one role if this suits its business). The purpose of defining business roles is to partition an active networking system in a way that responsibilities are identified, such that they can be clearly separated and assigned to different actors. Consequently, the enterprise model has to determine the interactions that are expected to take place between the actors acting in the respective business roles, in order to allow for independence and "compositionallity" of the partial systems provided by the actors. The reference points are the means to describe these interactions which comprise the administrative relations, i.e., how to establish/handle the involved business relations and sometimes even legal issues (the "business model" in the terminology of the Network Management Forum NMF) and the technically necessary interaction schemes that have to be supported by the respective interacting components, their functionality and their capabilities, the qualities they have to guarantee, etc., in short, the necessary technical (interface related) contracts that have to be established between the components in the realm of the various actors in their respective roles (this makes the "business process model" in the NMF terminology). Since the contents of reference points in the "business model" actually are of no decisive relevance for the proper development of an active networking system (even if they are important for the decisions which services are to be developed and how), it is only presented as far as it is helpful to define the context. The focus of this document, however, is the presentation of the technical reference points and the thereby determined partition of the parts of an active networking system.

With respect to service access we align with the architecture of the Telecommunication Service Access and Subscription (TSAS) specification, adopted recently by OMG [OMG-TSAS]. The set of interfaces contained within this specification provides the domain facilities through which network operators can offer third-party enterprises secure access to the capabilities of the network. The TSAS specification is a domain facility enabling end-users to flexibly access communication services.

[1] The active networking system is the "FAIN universe of discourse", i.e., the (active) network (router and their software) together with all the services provided in relation with it.

3.2.2 Overview of the FAIN Enterprise Model

The business model for active network/service provisioning is depicted in Figure 3-2. It contains four groups of business roles:

- the role of the service and network users ("Consumer"),
- the roles of the service and network providers ("Service Provider", "Active Network Service Provider", and "Network Infrastructure Provider"),
- the supporting roles of the software and hardware component manufacturers ("Service Component Manufacturer", "Active Middleware Manufacturer", and "Hardware Manufacturer"),
- the auxiliary role of a software component distribution mechanisms, i.e., a kind of service component "yellow pages" ("Broker").

The business roles as understood here are strictly involved in the active networking business, i.e., they do reflect the additional activities/properties which are necessary in this case. They, however, share some commonalties (common activities and properties) with their non-active counterparts (i.e., the providers of not-active services, connectivity providers, etc.). We see these roles mostly in the light of the TINA business model. The respective differences will be discussed later in the document.

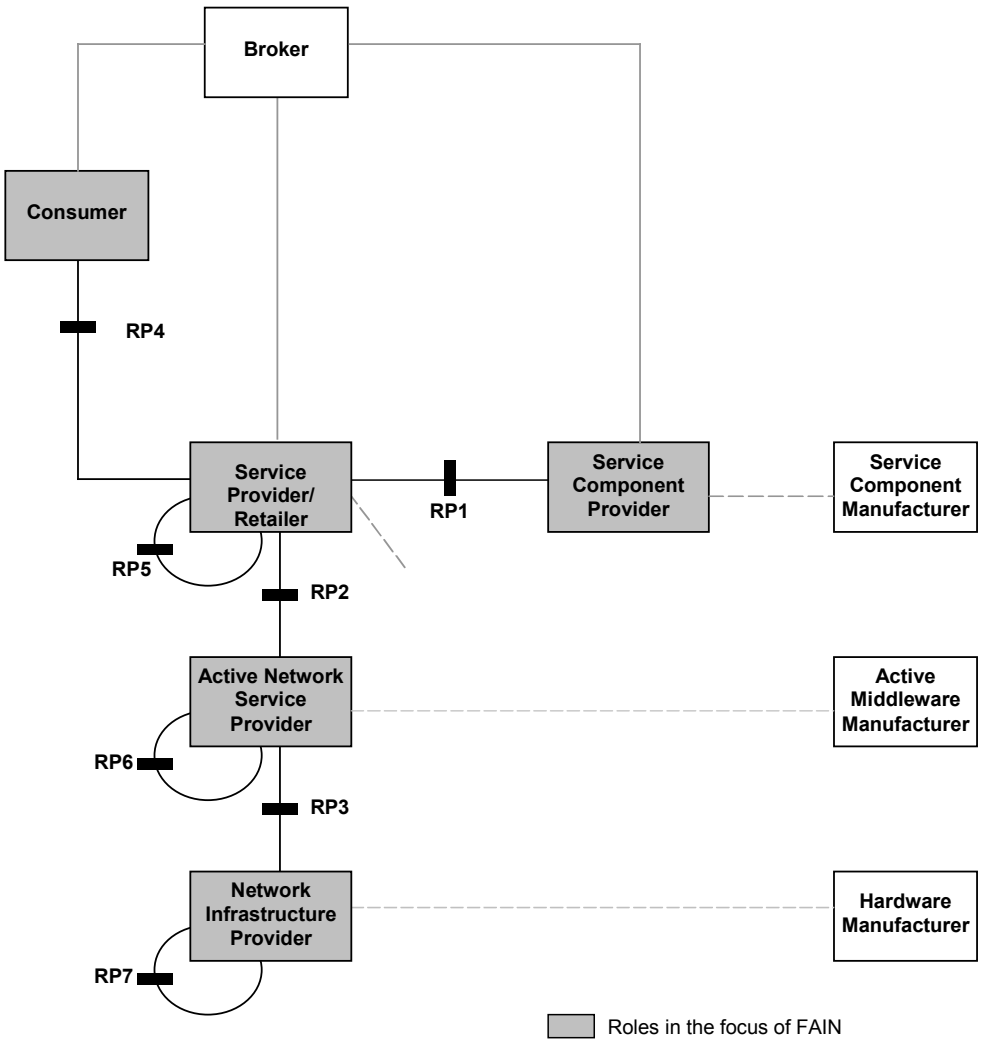


Figure 3-2 The FAIN Business Model

The highlighted/grey-shaded business roles are considered to be in the focus of the FAIN project, the other roles in the figure are not further detailed beyond the brief description in the following section.

3.2.3 The FAIN Business Roles

Within the business model depicted in Figure 3-2 the following FAIN business roles are identified:

Consumer (C)

The Consumer is the end user of the active services offered by a Service Provider. In FAIN, a Consumer may be located at the edge of the information service infrastructure (i.e., be a classical end user) or it may be an internet application, a connection management system, etc. In order to find the required service among those proffered by the Service Providers the Consumer may contact a Broker and, e.g., use its service discovery features.

Service Provider (SP)

A Service Provider composes services, including active components delivered by a Service Component Provider, deploys these components in the network via the ANSP, and offers the resulting service to the Consumers. The services provided by a Service Provider may be end user services or communication services. A Service Provider may federate with other Service Providers in order to build more complex services. Descriptions of the offered services are published via the Broker service.

Active Network Service Provider (ANSP)

An Active Network Service Provider provides facilities for the deployment and operation of active components into the network. It provides Virtual Environment facilities, (one or more) for active service components to Service Providers . Together with the Network Infrastructure Provider, it forms the communication infrastructure

Network Infrastructure Provider (NIP)

A Network Infrastructure Provider provides managed network resources (bandwidth, memory and processing power) to Active Network Service Providers. It offers a network platform including AN components to the Active Network Service Providers who can build their own Execution Environments, and proffers basic IP Connectivity[2] which may be based upon traditional transmission technology as well as emerging ones (both wired and wireless)[3].

Service Component Provider (SCP)

A Service Component Provider is a repository of active code that may be used by the Consumer and the Service Provider. A Service Component Provider may publish its components via a broker.

Service Component Manufacturer (SCM)

A Service Component Manufacturer builds service components and active code for active applications and offers them to Service Providers, Consumers and Service Component Providers in appropriate form (e.g., as binary or as source code).The Service Component Provider is not a genuine FAIN business role since it is not directly involved in the main FAIN business process.

Middleware Manufacturer (MM)

An Active Middleware Manufacturer provides platforms for active services (including the Execution Environments) to the Active Network Service Provider. The Active Middleware Manufacturer is not a genuine FAIN business role since it is not directly involved in the main FAIN business process.

---

[2] Since FAIN focusses on Internet technology, a basic network infrastructure consists on the following characteristics: presence of a physical network; IPv4 and/or IPv6 forwarding mechanism; default IP routing protocol; if RP 7 is implemented: peering service to other NIPs or ISPs; element management interface.

[3] The Service offered by the NIP may also be used by current Internet Providers.

Hardware Manufacturer (HM)

A Hardware Manufacturer produces programmable networking hardware for the Network Infrastructure Provider. The Hardware Manufacturer is not a genuine FAIN business role and will not be detailed further in the FAIN Enterprise Model.

Retailer (RET)

The Retailer (according to the TSAS model) acts as "access point" to the services provided by Service Providers to Customers; it has to be capable of negotiating and managing the end user contract (handling authentication and security issues as well as accounting and billing, etc.) and it has to allow the service life cycle management by the user (subscription and customisation of a service, administrative interactions, etc.).

Broker (BR)

The Broker collects, stores and distributes information about (1) available services, (i.e., those offered by the Service Providers), (2) the administrative domains in the active networking system, and (3) service components (i.e., code offered by the Service Component Providers). The FAIN broker basically resembles the "Broker" role in the TINA business model (which is characterised as to "allow all Consumers to have fair and equal access to information about how and where to find services and administrative domains" [TinaBook]). In addition to these TINA related broker properties, the FAIN broker is expected to publish service components, in particular active code components, which may be injected into the active network. The Broker may show a variety of auxiliary features, such as service property descriptions, service version management, service discovery, etc. The broker is no genuine FAIN business role and will not be detailed further in the FAIN Enterprise Model.

3.2.4 The FAIN Reference Points

The relations which have to be established among the FAIN business roles are described by the FAIN reference points. This section gives a short overview of the reference points. More detailed specifications are to be found in the following sections.

RP1 (SCP-SP)

- Contracting, subscription and transfer of components in a client server situation
- Information about Execution Environments should be exchanged to determine which components to transfer

RP2 (ANSP-SP)

- SP delivers components to be injected in the network by the ANSP
- SP requests generic network services, processing resources and code injection from the ANSP
- ANSP allocates resources depending on availability; if the allocation of resources fails, a roll-back scenario has to be executed
- ANSP provides information about capabilities of network services (SP can request this information)
- SP is responsible for service session management, according to the agreed service level (ANSP-SP)
- ANSP is responsible for network service management, according to the agreed service level (ANSP-SP)

RP3 (NIP-ANSP)

- NIP provides physical active networking resources

- ANSP requests resources (network, processing, transmission, etc.) from the NIP in order to provide network services
- NIP allocates resources depending on availability; if the resource allocation fails, a roll-back scenario has to be executed
- ANSP is responsible for injection of active code: for this, the ANSP requests from NIP the actual injection of the active code.
- ANSP can request a dedicated environment from the NIP (isolated from the other environments), depending on the agreed service level.

RP4 (SP-C)

- The Consumer requests and uses services offered by the Service Provider. The SP provides the service with its functionality. It also subsumes the role of the Retailer as service access point (i.e., it determines the service contract).In a composed service, it also acts as an intermediary between the Customer and the other Service Providers which provide the additional service features. The contract defined between the Consumer and the (initial) Service Provider defines access control and management services (e.g., resource usage and the related costs). An analogous contractual situation is established between co-operating Service Providers. Within FAIN, we assume that the federating relationship among Service Providers is hidden: the respective service facilities are directly offered to the Consumer.
- The possibility of injecting code by the customer has to be reflected in the reference point.
- The reference points described next are all references between different instances of the same role, here for short called self-references.

RP5 (SP-SP)

- A Service Provider may provide features of the services of other co-operating Service Providers and it may use these Service Providers to offer facilities of different Service Component Providers it is not associated with directly. The co-operation among Service Providers is an contractual issue and, depending on the chosen agreements, various service level may be supported.

RP6 (ANSP-ANSP)

- ANSP need to federate with other ANSP to provide integrated network services. This federation may be based on functional as well as geographical considerations.

RP7 (NIP-NIP)

- Most NIP are not able to provide global transport capabilities, they need to federate with other NIP to provide related infrastructure. This federation is mainly based on geographical considerations.

3.2.5 Comparison between the business models of FAIN and TINA

The TINA service architecture consists of a set of concepts, principles, rules and guidelines for constructing, deploying and operating TINA services. A TINA service is defined as a meaningful set of capabilities provided by an existing or intended system to all business roles that utilise it. TINA has identified business roles and reference points that help designing and reusing software telecommunication service elements. While TINA service description is relevant to some aspects addressed in FAIN project, it does not cover all requirements set by the technical annex of the FAIN project.

The TINA business model is to some extent “Retailer”-centred and much of the detailed specifications relate to it. In the FAIN project, the focus is on active networking issues which seem to be not crucial with respect to the “Retailer” functionality. Thus, we exploit the “Retailer” aspects within the Service Provider unless further consideration show that a separation of the two roles becomes necessary.

The FAIN business model includes additional roles that are relevant to active networking (i.e., the Service Component Manufacturer and the Active Network Service Provider) which result from the new business opportunities enabled by active networks. Thus, reference points have to be added or modified, respectively.

Active networking probably increases the number of actors in the service provisioning chain and may offers new business opportunities for value added service providers as well as for network operators.

An important new aspect in the FAIN enterprise model is the possibility to delegate management facilities along the service provisioning chain: from the Network Infrastructure Providers toward the Active Network Service Providers, the Service Provider, and the Consumer. Management delegation results from a genuine property of active networking, namely the feature that actors may inject/execute their own code on a provider level. The *session* concept is one of the fundamental paradigms of the TINA service architecture. A TINA session is the representation of all the intelligence required to enable access and usage of a TINA service. This intelligence is distributed over a number of service components, which themselves are deployed over a number of stakeholder domains. The TINA session concept forces a strict separation between *access session*, *service session* and *communication session.* While in FAIN we adopt the session concept, we do not concentrate TINA related details, but extend/adapt the TSAS framework to the details of IP networks and supporting active capabilities at the communication session.

The TINA "Connectivity Provider" (CP) as such does not match the needs of active IP networking since the CP focused on connection-oriented networks and does not address control, management and transport of connection-less IP networks. Also, within a connectivity provider domain, it is difficult to represent IP connectivity resources, generic and technology independently.

There is no framework that describes how the connectivity services, provided by a CP, are derived and built from generic and reusable connectivity components, and how these connectivity components are provided between different layer network domains.

In FAIN, we investigate how different active components are dynamically acquired and plugged into other service components to provide active services.

TINA is considered heavy-weight architecture, that is difficult to implement. In FAIN, we've adopted two frameworks to simplify implementations. The first framework is based on the separation between business and process models as proposed by TMF. However, we would not adopt fully TMF model for business process, but a light weight-model focusing on Reference Points specifications. The second framework is based on the segmentation facilities as proposed by OMG-TSAS, at least for the access &subscription part, while trying to apply segmentation for the communication part. Segmentation is thus used to reduce implementation complexity and to in-crease scalability.

3.2.6 Video-Conferencing: A simple Use Case

In this section an example video-conferencing session is elaborated to illustrate the different roles and interactions between the roles. This Section is meant as illustration only and will not consist of a complete use case.

Via a Retailer/Broker a Consumer finds out if a videoconferencing service with the required functionalities is available from a Service Provider;

- the Consumer subscribes to the video-conferencing service of a Service Provider. The Service Provider authenticates and authorises the Consumer (RP4);
- the Service Provider composes his video-conferencing service by using service components provided by the Service Component Provider, e.g. an IP multicast component and a transcoder for converting between different video- or audio formats etc. (RP1);
- the Service Provider is responsible for the session management of the videoconferencing session, for example to add or remove participants.

- the Service Provider installs the components in the relevant active nodes. This can be done both semi-permanent (for more sessions) or on demand (just for one session). For example if multiple video-conferencing sessions are running in parallel it may be easier to install the multicast component on a semi-permanent base (for each session multicast is needed) and to install the transcoder on demand. The Service Provider requests from the Active Network Service Providerto deliver basic network services between the participants (e.g. a 2 Mbps connection between node A and node B with a maximum delay of X). The Service Provider can either request bandwidth on demand (for each session) or semi-permanent bandwidth from the Active Network Service Provider(RP2);
- if a new participant enters the video-conferencing session (RP4), the Service Provider will install additional components , e.g. an additional transcoder, on some active nodes and will request the Active Network Service Providerto deliver additional bandwidth (RP2). If a participant leaves a video-conferencing session or the video-conferencing session is ended (RP4) the Service Provider will ask the Active Network Service Providerto remove the bandwidth resources (RP2) and the Service Provider will remove the components in the active nodes (RP2);
- the Active Network Service Provider delivers to the Service Provider an execution environment or active middleware environment (e.g. Java Virtual machine) in which the Service Provider can install the components for the service (RP2);
- the Active Network Service Provideris responsible for providing and maintaining the execution environment and the basic network resources;
- the Active Network Service Providerrequests from the Network Infrastructure Provider the necessary physical resources, like CPU time, transmission and memory (RP3);
- the Network Infrastructure Provider periodically informs the Active Network Service Providerabout the use of the physical resources. The Network Infrastructure Provider informs the Active Network Service Provider about faults or errors on physical resources (line outage, memory overflow etc.) (RP3);
- the Active Network Service Provider periodically informs the Service Provider about the use of the network services and middleware platform (RP2).

3.3 Conclusions

In the open service environment there will be a variety of ways in which organisations can co-operate in providing active services. As services could be composed of service components, service provisioning could be based on vertical relationships where services from one provider are used by other providers as components in their services (SP, SCM, ANSP, NIP), resulting in differentiated service provisioning value chains. Horizontal relationships exist also between the various providers of similar services in cases where service providers cannot offer complete global coverage on their own and enter into co-operative agreements with other service providers in order to deliver a service to users, i.e., the primary service provider is sub-contracting with other providers to fulfil a customer's needs.

In order to develop the business model for FAIN, we analysed different available business models (TINA, ETSI, ITU, ...), we adopted the separation between business model and process model as proposed by NMF.

3.4 References

[OMG-TSAS] Telecom Service Access & Subscription Specification V1.0, OMG, October 2000, http://www.omg.org/techprocess/meetings/schedule/TSAS_FTF.html

[NMF BMP] Network Management Forum: *A Service Management Business Process Model*, NMF, NJ, 1996.

[TINA-BM] TINA Consortium: *Reference Points and Business Model*, Version 3, June 1996

[TINA-SA] TINA Consortium: *TINA-C Service Architecture*, Version 5.0, 1997.

[TINA-CMC] TINA Consortium: *Computational Modelling Concepts*, Version 3.2, 1997.

[Y.110] ITU-T Recommendation Y.110: *Global Information Infrastructure principles and framework architecture*. 06/98.

[Y.120] ITU-T Recommendation Y.120: *Global Information Infrastructure scenario methodology*. 06/98.

[TinaBook] Yuji Inoue, Martine Lapierre, Cesare Mossotto (eds.): "The TINA Book. A Co-operative Solution for a Competitive World", Prentice Hall Europe, 1999

[TinaBusiness] Harm Mulder (ed.): TINA Business Model and Reference Points, V4.0. Tina Consortium, 1997.

4 APPLICATIONS FOR ACTIVE NETWORKS

4.1 Introduction

In this chapter an application-oriented approach is used to derive requirements for FAIN. The idea is to select applications which we expect to benefit from the distinguishing property of active networks, i.e. the possibility to download code into the network and to distribute intelligence flexibly this way. Among the multitude of applications which have been presented in the literature and which have been discussed in FAIN, we have selected the following ones:

- *Virtual private networks:* Virtual private networks (VPNs) are private networks built on top of a public network like the Internet. Due to the diversity of requirements for virtual private networks, no uniform "standard" for VPNs has been established until now. For this reason, the IETF has stopped standardisation activities in this area. Nevertheless, several "building blocks" required for active networks are standardised and can be used: Techniques for tunnelling, support for Quality of Service (QoS), Encryption, Routing...

  From Active Networks, we expect to be able to flexibly compose these basic-building blocks in order to implement the requirements of a particular customer, and to deploy the resulting VPN service logic into the network.

- *Reliable Multicasting*: Protocols for unreliable best-effort multicasting are available both in standards and within products since more than 10 years. Intensive research and development into various extensions (multicast using the properties of the underlying link-layers such as ATM, reliable multicasting, multicasting with QoS) has been conducted. Nevertheless, none of the proposals, not even unreliable best-effort multicasting have been deployed on a larger scale.

  In our opinion, this is also due to the diversity of application requirements, which will make the development, deployment and operation of a uniform, one-fits-all multicasting protocol hard or even impossible.

  Active Networks may help in developing and deploying application-specific multicast protocols with characteristics tailored to the requirements for the application. A prominent example is networked games, which pose hard requirements, e.g. w.r.t. QoS, reliability and group management.

- *Web Service Distribution:* The World Wide Web is certainly the most popular and important application of the Internet. The underlying technologies like HTML, XML, HTTP, ... are not only important for "internet surfing", but for mission critical business applications as well. However, protocols like HTTP have originally been invented as a pure client-/server approach, making the scalable, reliable and performing implementation of web-services difficult. Current approaches address particular issues in an ad-hoc way (e.g. distribution of content via content distribution networks), but without addressing the issue on an architectural basis.

  We expect to benefit from activeness in the network in two variants. First, activeness can be used to distribute code in order to filter out web traffic. The downloaded code constantly monitors network-internal conditions (bandwidth, availability of servers, ...), and distributes the filtered traffic in an adequate way to implement objectives like high-performance or availability. Second, activeness can be beneficial for distributing not only content, but also service logic of web-services, which is clearly desirable for scalable large-scale web services with multimedia content.

In the following sections, these applications are elaborated in more detail, using a use-case oriented approach. Each of these sections is structured as follows:

- *Introduction:* The introduction briefly describes the applications, justifies the use of active networking technology for implementing the applications, and gives references to related work

- *Embedding of the application:* In this section, the relevant actors are described and are mapped onto the actors of the FAIN enterprise model. In addition, the necessary networking resources for running the application are stated.
- *Use-Cases:* This section contains several use-cases for modelling the application. For each use case, we start with a summary, a description of the problem to be solved by the use cases, a description of the resources and of the preconditions required to run the use cases. In addition, one or several execution scenarios are given in the form of a sequence of execution steps, which are mapped onto the respective reference points of the enterprise model. If considered as necessary, these may also contain alternative scenarios or exceptional situations. Also, an assessment of the relevance of the different use cases is given.

Emphasis in the modelling of the applications is to work out the AN-specific aspects.

Besides being a means to derive requirements for the system as a whole within FAIN workpackage 2 (see Chapter 5), these applications are also candidates for the detailed design & architectural investigations within WP3 and WP4 as well as for the implementation in Demos.

4.2 Active Web Service

4.2.1 Introduction

In the application "Active Web Services" (AWS), active network technology is used to provide scalable, reliable large-scale web services. In the following subsections, we describe and motivate this application, which will be modelled by use cases later on. The application is a merge of two applications which have been discussed in the FAIN project before, namely Web Service Distribution [WSD] and Active Web Gateway [AWG].

4.2.1.1 Application Description

Web technology has been the single most important technology responsible for the explosive growth of the Internet. It is not only the uniform technology that accommodates Internet surfing by end users, but it is as well mission critical for business applications both between and within enterprises. In this context, a number of requirements need to be addressed:

- To accommodate increasing numbers of customers,
- To provide scalable, reliable and efficient web services,
- To be able to quickly add new services and modify existing services,
- To reduce end-to-end traffic and server load.

To meet this diversity of requirements coming from customers and service providers we propose to design and implement an environment for web services based on Active Networks. In order to present our ideas, we first revisit the state of the art in web technology. Web technology (e.g., HTTP, HTML, ...) has been developed originally as a pure client-/server architecture, where End Users are connected with the Web Service Provider site via an application-unaware IP network. Usually, such an IP network is composed of an access network for both the End User and the Web Service Provider site and an IP core network (Figure 4-1).

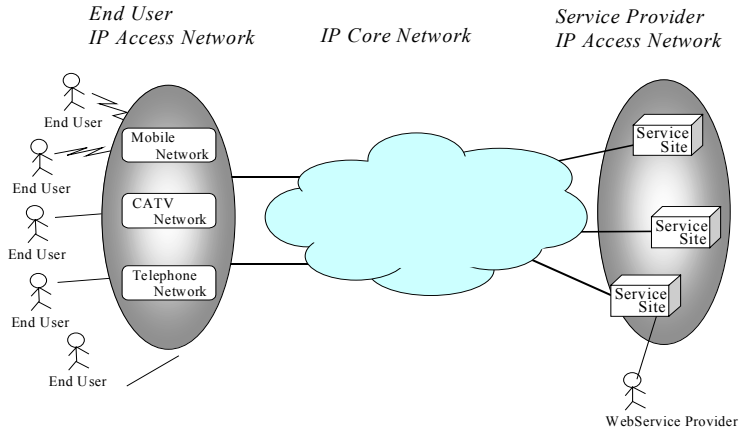


Figure 4-1: Physical Network Architecture of Web Services

This simple approach, where the „intelligence“ mainly is located in the Client and the Server, was one of the main reasons for the enormous success of the internet. However, it is also well-known that over the public internet, this pure client-/server model on top of an “dumb” IP network has shown to lack important characteristics, which are required for today’s and future Internet applications. These include reliability, performance and scalability as well as the possibility to take network internal conditions (e.g. the available bandwidth for a particular end-users) to take into account into the service provisioning.

In order to overcome these deficiencies, the network infrastructures have already been extended in various ways. Some solutions which have been proposed and deployed in response to the growing demands of existing and new applications are: (see also Figure 4-2)

- Web caching [WREC], where static content is stored within caches within the network to reduce the response time for subsequent requests.
- load balancing/layer 7 switching [RAD], where requests are distributed to several servers in a server farm. To clients, this server farm appears as one “virtual web server”, which is reachable by one IP address, which in reality is served by multiple servers.
- content distribution networks [CD, DIG], where large volume content such as images or video is pushed onto dedicated content distribution servers, which are distributed worldwide. These servers all have an own IP address and DNS name, and the traffic is redirected baseded on features included in the HTTP protocol.

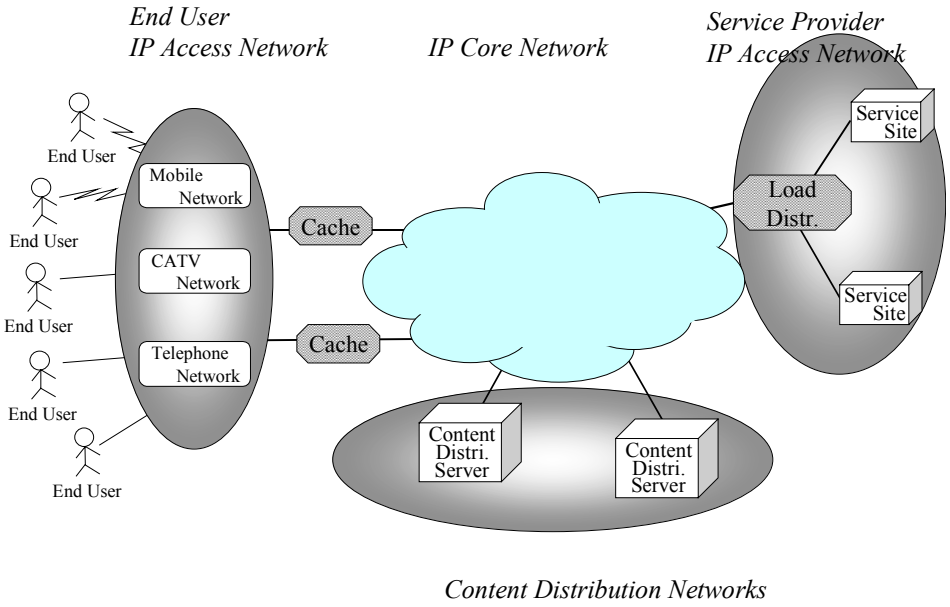


Figure 4-2: Web caches, content distribution servers and load distribution servers

However, these can only be considered as ad-hoc solutions to specific problems. For instance, caches only deal with static content. Important features of today's web services, such as e.g. the observation of page hits of a particular page by the service provider or a third party, can still only be implemented by relying on centralised servers. We therefore believe that also today's web services already can benefit from an architectural framework that can support an easy, rapid, and uniform way of deployment of new solutions that need "intelligence" within the network.

In addition, more network services are requested by customers and designed by service creators which clearly show that there is a demand for extensibility and flexibility on the ISPs and operator's infrastructure. [LWG98] gives some examples of web-services which would clearly benefit from such a mechanism. A good example is a personalised *stock quote* service. Stock quotes are frequently changing data which can hardly be cached. If in addition a service for distributing stock quotes should be personalised (e.g. with respect to the portfolio of a certain user), it becomes inevitable that with current approaches a high network- and computing load is generated on a centralised server. A distributed architecture would permit personalisation and information distribution within the network, being clearly more scalable.

The basic idea of our active web service infrastructure is to exploit the capabilities of activeness in the following two ways:

- On the one hand, so called "service nodes" within the network are implemented using AN technology. These service nodes are "Active Web Servers", which can be programmed by the web service provider. They provide for instance persistent storage in order to allow the service provider to store content locally and an environment to execute web service logic (e.g. JavaBeans). The service provider downloads content and service logic onto them and by this way can implement features such as content distribution and personalisation, aggregation of user replies or fast response times.

- On the other hand, so called "redirection nodes" are also implemented using AN technology. They provide features which allow to filter out HTTP traffic, to build service sessions (i.e. to deal with per-user state) and to forward the traffic of a particular session to a service node. Onto this nodes, code is downloaded which observes the network load, observes the load and availability of servers and based on this information determines a strategy for redirecting the traffic to the service node which is most suitable for a given web service/user.

Of course, both kinds of functionality may be combined within one physical node, and there may also be nodes which provide a functionality which is a mixture of both. However, we usually expect them to be separate, both physically and logically. On the one hand, programming a network (e.g. load balancing algorithms, routing algorithms,...) requires different skills than programming web services and hence will be carried out by different actors. Second, both kinds of nodes will usually be implemented with different design objectives: While redirection nodes will be some kinds of routers with an emphasis on network throughput, service nodes will require at least some part of functionality of web servers (i.e. huge computing power, persistent storage...).

Also note that both kinds of activeness are complementary and can be implemented independently: The redirection of traffic by redirections can also be beneficial if a distributed set of web servers is implemented and operated completely independent of AN technology. Vice versa, a distributed, AN-based implementation of a web service infrastructure can also be used with conventional techniques for load balancing. In the sequel, we call any web service which is implemented using some kind of activeness an "Active Web Service".

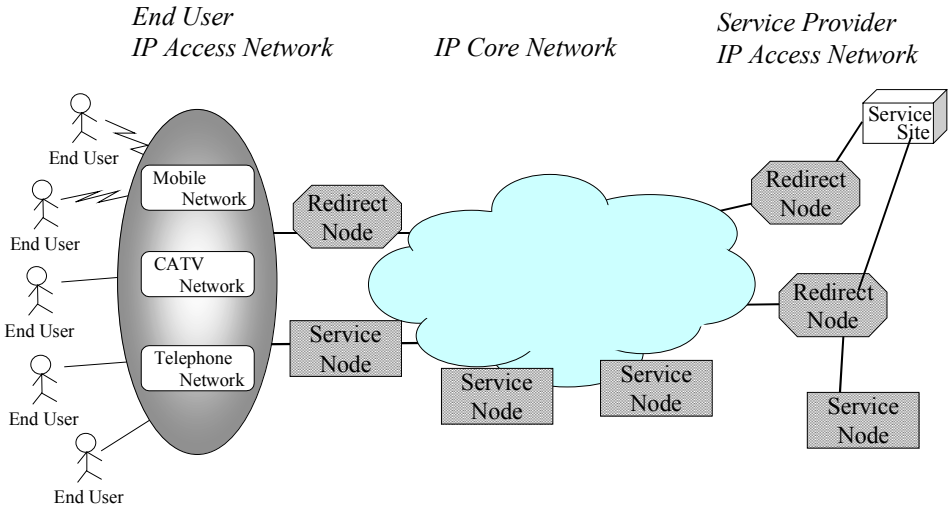


Figure 4-3: Service nodes and redirect servers implement active web services

The overall scenario is depicted in Figure 4-3. Both redirection nodes and service nodes will usually physically located within the access network of the end user, the access network of the web service provider, or connected via a separated access network. For this reason, we may also call both of them "Active Web Gateways". Note that we assume that the core network is non-active and only provides basic IP connectivity.

One design goal is that the overall setting is fully transparent to the end user, i.e., the end user uses the web service via an ordinary web browser, using standard protocols such as IP or HTTP. This assumption takes into account that updating several hundreds of millions of web clients is not only infeasible, but also unnecessary.

To sum up, the proposed application can be viewed as a continual evolution of existing web infrastructure:

- Traditionally, the web has been based on a "dumb" IP network, offering only pure best-effort IP routing and forwarding capabilities.
- Recently, the web has been enhanced by installing caches, switches and content distribution networks. However, the "intelligence" (i.e. the service logic) still resides on centralised servers operated by the web service providers.
- In the future, the web might by a fully distributed computing infrastructure, with service intelligence distributed on the IP and/or on the application layer.

4.2.1.2 Justification of the Use of Active Network Concepts

With Active Network technology web traffic can be handled flexibly within the network. Potential benefits include among others:

- *Network aware web services:* Contrary to existing web services, they can operate based on information such as available bandwidth on access and network links, network topology, and load of servers. Because the information is not available at the client- or server side, it should be implemented with AN technology within the network.
- *Ease of service programming and management*: Active networks eliminate the need for cumbersome ad-hoc solutions to specific problems which require separate management; active networks provide common ground for deployment of new services and mechanism inside the network and unifies the management of these mechanism and services.
- *Distribution of service logic*: Service logic is executed at several locations, including specific points inside the active network, which is potentially advantageous for large volume services, fine (per user) granularity of services, new service features, …. With existing solutions such as caches or content distribution networks, only content, but not service logic can be located within the network..
- *Dynamic, autonomous adaptation*: The task of operators on service site (service provider, network provider) is getting bigger as the number of network customer/consumer is growing. So the task of provisioning should be dynamic. Besides active nodes may cooperate with each other without operators' (humans) intervention.

4.2.1.3 Relevant Documents

[AWG] Chiho Kitahara et al.: Application Active Web Gateway, FAIN project, Internal WP2 Report WP2-HIT-004-R32-Int.doc

[BAEN] Baentsch et. Al. "World Wide Web Caching: The Application-Level View of the Internet", IEEE Communications Mag., June 1997

[CD] cdcenter.com – content delivery and distribution resource centre. http://www.cddcenter.com/

[DIG] Digital Island – http://www.digisle.com/

[HTTP] Fielding et al.: "Hypertext Transfer Protocol – HTTP/1.1". RFC 2616, June 1999, http://www.ietf.org/rfc/2616.txt/

[JEFF1] Jeffrey C. Mogul, et al.. "Squeezing More Bits Out of HTTP Caches"

IEEE Network May/June 2000 Vol.14 No.3

[JEFF2] Jeffrey C. Mogul."Hinted caching in the Web", SIGOPS European Workshop 1996.

[LWG98] U. Legedza, D. Whetherall and John Guttag: Improving the Performance of Distributed Applications using Active Networks. IEEE Infocom, San Francisco. Proceedings, 1998.

[RAD] Building Bullet Proof Internet/intranet sites with IP Load Balancing - Scalability, Optimisation, Fault Tolerance and Availability with RADWARE. White Paper, Radware.com, http://www.radware.com/support/papers/bullet.pdf.

[WREC] IETF Web Replication and Caching (wrec) Working Group. http://www.ietf.org/html.charters/wrec-charter.html

[WSD] Cornel Klein et al.: Application: Web Service Distribution., FAIN project, Internal WP2 Report WP2-SAG-002-I01-Int.doc

4.2.2 Embedding of the Application

4.2.2.1 Overview of Actors and their Interrelations

4.2.2.1.1 Overview of Actors

The actors involved in the application are:

End User(End User):

This actor uses the Web Service provided by the Web Service Provider (WSP). Alternatively, the End User may have subscribed to the service, after which a contract with the Web Service Provider has been established, which contains the preferable service level. This contract is usually called "SLA". As a result, the End User can consume the AWS with the chosen service level for which he would pay.

Web Service Provider(WSP):

This actor provides a web service for End Users (e.g. as yahoo.com). He designs and implements service logic, asks WSDP to distribute it to web servers, and offers the service to End Users.

Communication Infrastructure Provider (CIP):

This actor owns and operates physical network infrastructure, consisting of active routers.

Server Infrastructure Provider (SIP):

This actor owns and operates a physical active web server infrastructure.

Active Network Operator (ANO):

This actor provides managed resources on active routers which are owned by Communication Infrastructure Provider. Managed resources are e.g. CPU, memory, bandwidth... IP connectivities are supported by NIP with assigned QoS. The ANO provides an environment which allows to filter out HTTP traffic, deal with service sessions (i.e. with user-specific state) and to redirect HTTP traffic to service nodes.

Web Service Distribution Provider(WSDP):

This actor provides the Web Service Distribution Service to the Web Service Provider with high reliability and performance. He also programs and implements active codes that are installed in active routers and active access gateways. A web service distributed by the WSDP will be called "Active Web Service".

Web Service Infrastructure Operator(WSIO):

This actor operates and manages the distributed web hosting infrastructure consisting of the service nodes owned by the SIP. On these service nodes, the WSIO builds an environment where web service logic can be installed and executed.

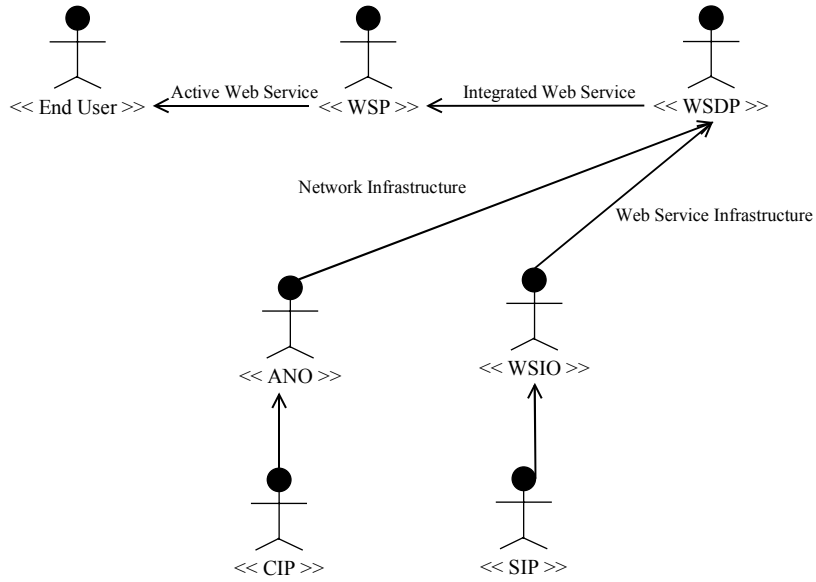


Figure 4-4: Actors in AWS and their relationships

These actors are mapped on FAIN Enterprise Model roles as follows:

AWS Actor	**EM Role**
End User	Consumer
Web Service Provider	Service Provider
Web Service Distribution Provider	Service Provider
Active Network Operator	Active Network Service Provider
Web Service Infrastructure Operator	Active Network Service Provider
Communication Infrastructure Provider	Network Infrastructure Provider
Server Infrastructure Provider	Network Infrastructure Provider

4.2.2.1.2 Architectural Assumptions

Usually, web applications are built around a multi tier architecture, as depicted in the following picture:

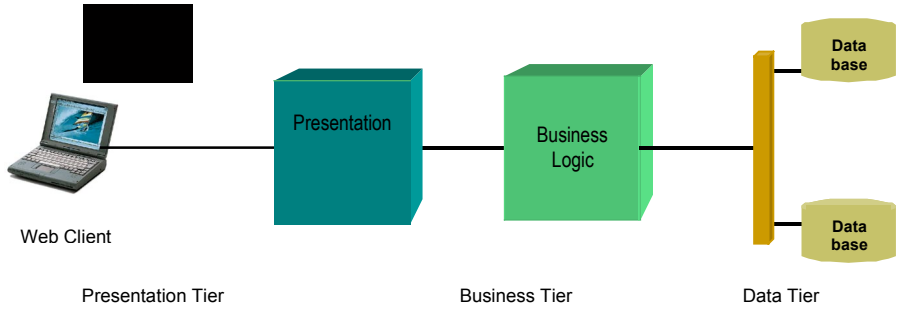


Figure 4-5: Current Web Application Architecture

- The Web Client is usually a PC, where the web browser runs and which access the web via IP. The client part of the presentation tier is located here.
- On the Server side, the presentation tier is built up by numerous web servers (HTTP Servers), often operated by Service provider. Often they are physically integrated with an execution environment for Servlets/JSPs which compute dynamic content of pages.
- The business Logic is responsible for the presentation-independent part of the application. In modern systems it is often implemented using JavaBeans and integrated with legacy business components.
- The data tier, which provide integrated access to one or more databases, which holds user specific data.

We are not discussing issues of distributing a database, but assume that a centralised database is operated by the web service provider. We are discussing the distribution of HTTP Servers (Presentation tier, business logic tier) on service nodes, and the re-routing of IP traffic to these service nodes via an active IP network in contrast to a dumb IP network with best-effort forwarding.

4.2.2.2 Actor Description

4.2.2.2.1 End User

An End User uses the service provided by the Web Service Provider (WSP) via an off-the shelf web browser such as Netscape Navigator or Internet Explorer, because upgrading several millions of client sites will be infeasible. Moreover, we assume that the End User is connected with the AN via the public Internet or another IP network (Figure 3-2) and hence that the AN infrastructure is fully transparent to the End User . In particular, we assume that the End User does not program the network in any way.

4.2.2.2.2 Web Service Provider

This actor uses the Web Service Distribution Provider (WSDP) to provide highly reliable, scalable web service(s) to End Users. The WSP may also operate an own web server, but in general the operation of a web server may be completely outsourced. The business of the WSP is the development of web services for end-user. For this reason, minimal additional skills should be required to use the service provided by the WSDP. We therefore assume that the underlying AN infrastructure is hidden as far as possible to the WSP by the WSDP.

4.2.2.2.3 Web Service Distribution Provider

The Web Service Distribution Provider (WSDP) offers a web service distribution service to its customers. To provide this service, this actor uses the distributed web hosting infrastructure provided by the web server infrastructure operator to operate the service nodes, and the active communication infrastructure provided by the AN Operator in order to distribute traffic the among the service nodes. The WSDP offer to its customers (i.e., the web service providers) the operation highly reliable, highly scalable web service with high performance. It uses the AN infrastructure directly in the sense that it programs active nodes. Since these nodes distribute traffic among the web servers, we call them "redirection nodes".

4.2.2.2.4 Active Network Operator

This actor provides managed resources on the AN infrastructure to its customers, i.e., to Service Providers like the Distributed Web Service Provider. It has complete control over the AN infrastructure and is responsible for configuring, managing, and operating its AN domain. There may be many Active Network Operators, each responsible for its own domain.

4.2.2.2.5 Web Service Infrastructure Operator

The Web Server Infrastructure Operator (WSIO) operates a distributed web hosting infrastructure. This consists of so called "service nodes", which run active web servers. They are implemented on the Servers provided by the Server Infrastructure Provider (SIP).

4.2.2.2.6 Server Infrastructure Provider

The Server Infrastructure Provide owns and operates servers, which he provides to the WSIO.

4.2.2.2.7 Communication Infrastructure Provider

The Communication Infrastructure Provider provides active routers, which are used to implement the redirection facilities.

4.2.2.3 Resources assumed for the Application

- The End-user operates an ordinary web browser on a personal computer, a workstation, or mobile devices in order to use AWS.
- The WSP operates an ordinary web server.
- The WSIO operates a management system for web servers which are implemented on top of the infrastructure provided by the SIP.
- The SIP provides a set of server nodes, which provide disk space, CPU, computational resources etc.
- The CIP operates active routers and provides them to the ANO.
- The WSDP operates a server for operating his service.
- The ANO manages an infrastructure of programmable routers and access gateways, which are capable of recognising & analysing HTTP packets (i.e. ordinary IP packets) and executing management function and service functions.

Basic connectivity among these actors is provided by an IP network, like e.g. the public Internet.

4.2.3 Overview of the Use Cases and their Interrelations

The AWS application is modelled by the following use cases with interaction of above actors as shown in Figure 3-2. We only model and describe the most important use cases, while for a full specification of the application – which is not indended here – a more comprehensive set of use cases would be needed, covering also the exceptions and alternative cases:

Configure Distribution Infrastructure:

WSDP sets up AWS as a new active service with purchasing and configuring the necessary resources. After the execution of this use case, a WSP can use the WSDP to distribute their services.

Distribute WS:

WSP designs and implements service logic and asks WSDP to distributes service logic onto active web servers.

Modify AWS:

WSP has implemented a new version of the service logic and asks the WSDP to distribute the service logic onto the web servers, while maintaining a continuous service operation.

Subscribe AWS:

WSP subscribes AWS from WSDP, altenatively an End User may subscribe AWS from WSP. When subscribtion is done, it is based on service level agreement (SLA).

Use AWS:

An End User uses an Active Web Service.

Reconfigure Distribution Scheme:

During AWS service running, some resources may need to be reconfigured automatically depending on status of network or active nodes.

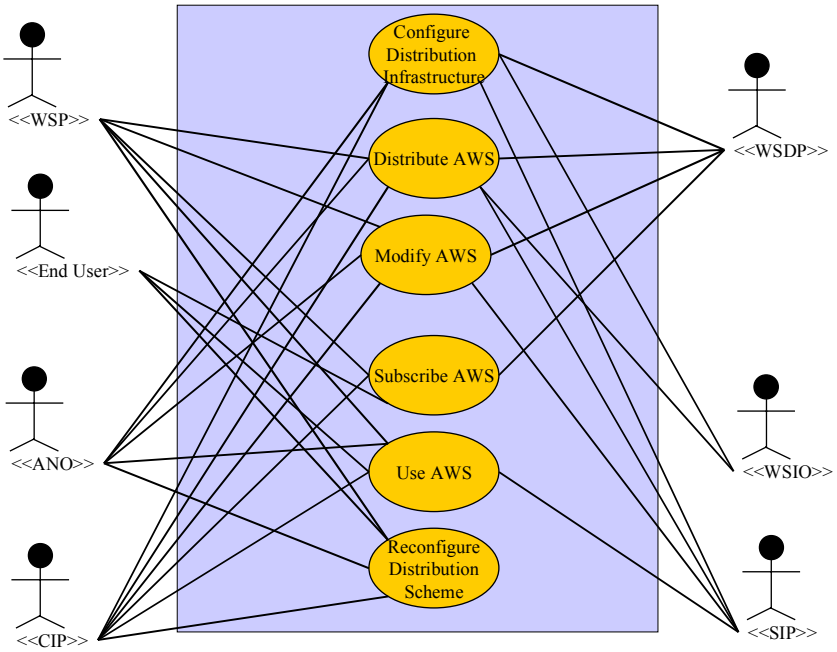


Figure 4-6: Use Case Diagram for AWS

4.2.3.1 Use Case "Configure Distribution Infrastructure"

4.2.3.1.1 Use Case Summary

This use case is initiated by the WSDP. The WSDP purchasesresources from ANO and WSIO and configures them for its services.

4.2.3.1.2 Actors Involved in the Use Case

- Web Service Distribution Provider (WSDP)
- Active Network Operator (ANO)
- Web Service Infrastructure Operator (WSIO)

4.2.3.1.3 Problem Statement

Before the WSDP can serve any WSP or End User, he has to initialise his infrastructure. This involves initialising the Active Web Server Infrastructure, and distributing and initialising the redirect logic.

4.2.3.1.4 Resources assumed for the Use Case

Some resources, especially storage, memory, may be used to store configuration data in management node and some of active nodes.

4.2.3.1.5 Pre-Conditions

Active nodes are managed by ANO and basic function is running on web servers of WSIO.

WSDP has designed and estimated required resources, e.g. cache size, caching interval, redirect schema, QoS, and this may be done for some amounts of End Users (not only for an individual End User) and WSPs.WSDP has designed and implemented the redirect logic to be deployed onto the active routers.

4.2.3.1.6 Use Case Description (Main Flow)

1. WSDP purchase active web server resources from WSIO, who operates the active web server infrastructure. The requirements from the WSDP are expressed as an SLA, which contains statements about the topological location of the active nodes and about the resource required on these nodes. Typical resources include the size of caqche space, processing power, bandwith etc. **(RP2)**

2. Likewise, the WSDP purchases resources on redirection nodes, based on an SLA, from the ANO who owns managed resources on active nodes. The SLA will be computed based on the servers which have been negotiated in step 1, their location and their capabilities. (**RP2**)

3. Both SLAs are translated by the WSIO and ANO into policies, which are in turn distributed to the active nodes/active servers. **(RP3)**

4. The redirect logic will be installed into appropriate active nodes and executed with policies information. **(RP2)**

5. The active web servers will be initialised. **(RP2)**

6. WSDP may ask WSPs to promote AWS as a new service, which may be portal web access point of End Users. (**RP5**) (not shown in the figure)

7. WSP advertises AWS on its web site. (**RP4**) (not shown in the figure)

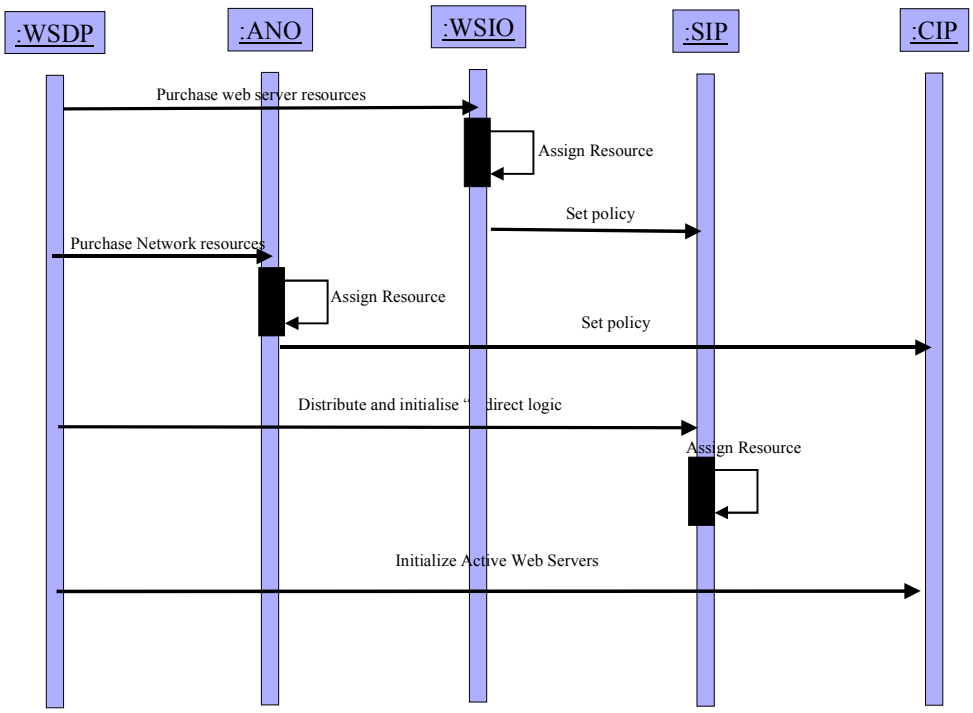


Figure 4-7: Sequence Diagram of "Configure Distribution Infrastructure"

4.2.3.1.7 Exceptions and Alternative Flows

We consider the following exceptions and alternative flows:

- If WSDP can not obtain the required resource that means ANO or WSIO is not able to provide the required resource to WSDP, the WSDP may re-estimate the necessary resources and negotiate with the ANO or WSIO again or look for other ANOs/WSIOs.
- Several ANOs/WSIO might be need for global coverage.
- An ANO shall negotiate with other ANO to provide IP connectivity through multiple core network domains or core plus access network. **(RP6)**

4.2.3.1.8 Post-Conditions

A basic installation of AWS to relevant active nodes has been done. Initial active codes are ready to execute and policies (these may be service specific policies) are installed.

4.2.3.2 Use Case "Distribute AWS"

4.2.3.2.1 Use Case Summary

This use case starts when WSP has designed and implemented a new AWS and uses WSDP to distribute the service on active nodes.

4.2.3.2.2 Problem Statement

Distribution of the service logic in the network and configuration of the nodes.

4.2.3.2.3 Actors Involved in the Use Case

- Web Service Distribution Provider (WSDP)

- Active Network Operator (ANO)
- Web Service Provider (WSP)
- Web Service Infrastructure Operator (WSIO)
- Communication Infrastructure Provider (CIP)

4.2.3.2.4 Resources assumed for the Use Case

4.2.3.2.5 Pre-Conditions

The use case "Configure Distribution Infrastructure" has been successfully completed.

WSP has designed and implemented service logic on top of an open API on the active web servers. The "service logic" may be composed of static content (e.g., represented as a directory tree), configuration information for a web server (e.g., an Apache Config file), and program code (e.g. cgi-bin scripts, Java Servlets, Java Beans, JSPs, etc.).

4.2.3.2.6 Use Case Description (Main Flow)

1. WSP contacts WSDP to distribute service logic. **(RP5)**
2. WSDP contacts one more WSIOs to distribute the service logic onto the web servers managed by these WSIOs. **(RP2)**
3. Each WSIO in turn distributes and initalises the service logic onto his web servers.
4. The redirect logic is informed that the service logic is available on some servers, and may that it should forward subsequent traffic to these servers. (**RP2)** (delegated access to CIP)
5. The WSDP is notified about the successful installation of the service logic, and in turn configures the redirect logic appropriately. **(RP2)**
6. The WSP is informed about the successfull completion of his request. **(RP5)**

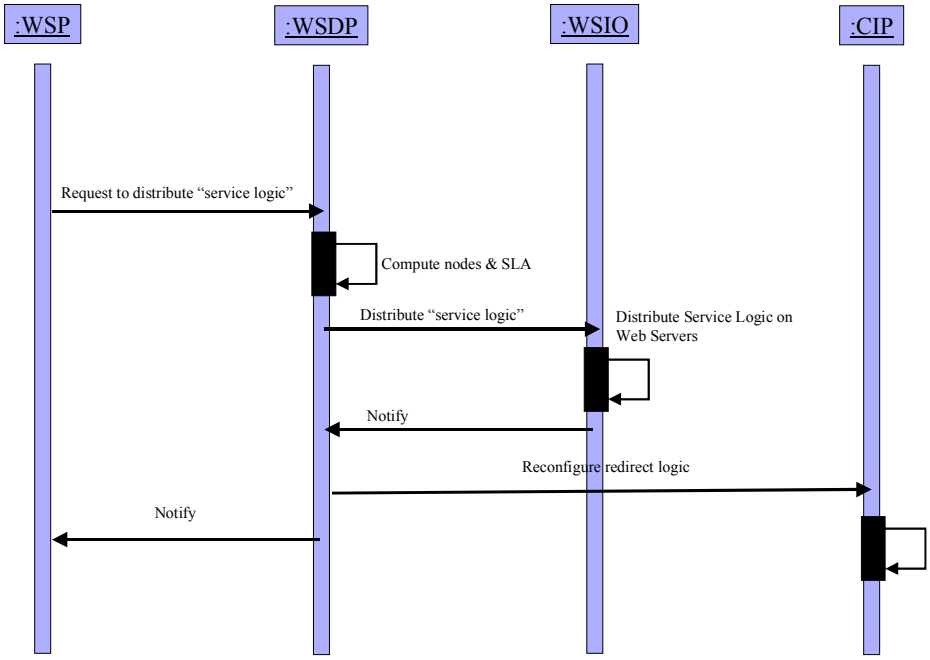


Figure 4-8: Sequence Diagram "Distribute AWS"

4.2.3.2.7 Exceptions and Alternative Flows

4.2.3.2.8 Post-Conditions

The web service is distributed in the network and can now be subscribed and used by the End User.

4.2.3.3 Use Case "Modify AWS"

4.2.3.3.1 Use Case Summary

Replace an already deployed service logic with a new version of the service logic.

4.2.3.3.2 Problem Statement

End users should experience a non-stop operation of the Web Services and not being concerned with an upgrade of the service logic.

4.2.3.3.3 Actors Involved in the Use Case

- Web Service Distribution Provider (WSDP)
- Active Network Operator (ANO)
- Web Service Provider (WSP)
- Web Service Infrastructure Operator (WSIO)
- Communication Infrastructure Provider (CIP)

4.2.3.3.4 Resources assumed for the Use Case

Sufficient resources are available on the web servers to run both version of the service logic simultatenously (in particular memory).

4.2.3.3.5 Pre-Conditions

The web service has already been distributed by "Distribute AWS".

The WSP has implemented a new version of the service logic.

4.2.3.3.6 Use Case Description (Main Flow)

1. WSP contacts WSDP to update the service logic of an already distributed WS. (**RP5)**
2. WSDP contacts the involved WSIOs to distribute the service logic onto the web servers managed by these WSIOs. **(RP2)**
3. Each WSIO in turn distributes the service logic onto their web servers, while keeping the old service logic up and running. **(RP3)**
4. After the WSIO has completed the installation and start of the service logic, the WSDP configures redirect logic to forward traffic for new sessions to the new service logic. (RP2) (delegated)
5. The WSP will be informed that the new service logic has been distributed and that new traffic is forwarded to this service logic. **(RP5)**
6. WSDP notifies asks WSIO to terminate old service logic. **(RP2)**
7. WSIO forwards the request to terminate the old service logic to his web servers. **(RP3)**
8. The service logic will run until the last user has terminated his service session.
9. The WSP will be installed about all terminations of the old service logic on the different servers. **(RP2)**

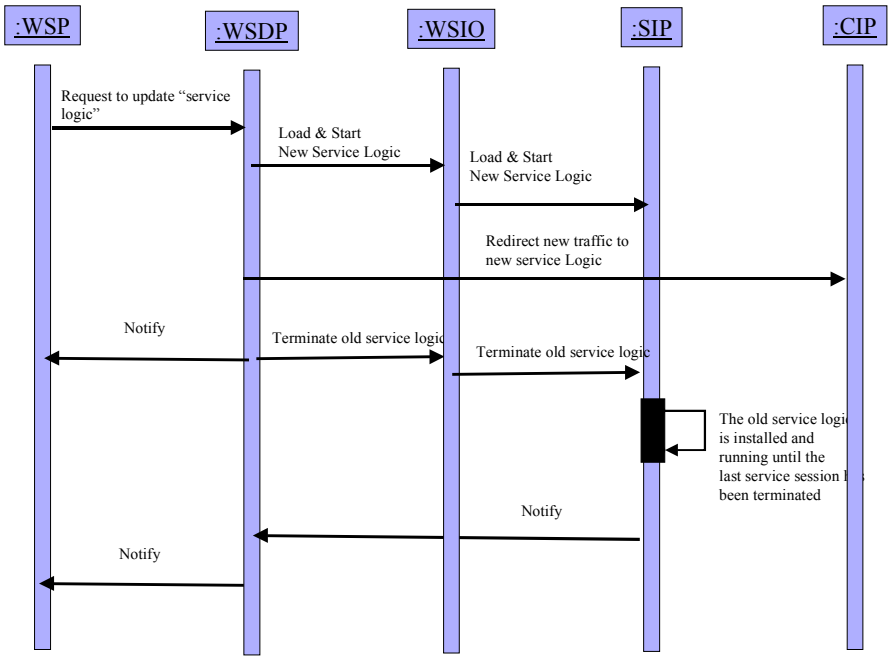


Figure 4-9: Sequence Diagram "Modify AWS"

4.2.3.3.7 Exceptions and Alternative Flows

We consider the following exceptions and alternative flows to be important:

- For resource reasons or some other reason (e.g. incompatibility of new and old service logic, explicit request by WSP...) the running service sessions will be terminated by the Web Server.

4.2.3.3.8 Post-Conditions

The new version of the web service is distributed in the network and can now be subscribed and used by the End User by invoking new service sessions.

4.2.3.4 Use Case "Subscribe AWS"

4.2.3.4.1 Use Case Summary

In this use case, an End User subscribes to a particular Web Service. Using the facilities provided by the WSDP, the network is configured as such that the End User get service quality as specified in a SLA.

4.2.3.4.2 Problem Statement

In many cases, the WSP will negotiate with the WSDP to handle all requests destined for that WSP, irrespective of the End Use. An alternative scenario might be where the End User has first subscribed to this feature, maybe after having negotiated a particular service level. After the subscription, the active nodes are configured appropriately by the WSDP.

4.2.3.4.3 Actors Involved in the Use Case

- Web Service Distribution Provider (WSDP)
- Active Network Operator (ANO)
- Web Service Provider (WSP)

- Web Service Infrastructure Operator (WSIO)
- Communication Infrastructure Provider (CIP)
- End User

4.2.3.4.4 Resources assumed for the Use Case

End User uses its own terminal, e.g. personal computer, mobile device, in order to subscribe new service via internet. WSDP modifies its resource configuration .

4.2.3.4.5 Pre-Conditions

Basic installation of AWS has been done by executing "Configure Distribution Infrastructure" and "Distribute AWS".

4.2.3.4.6 Use Case Description (Main Flow)

1. End User subscribes AWS from WSP and negotiates service type (level) and SLA with WSP . For instance, SLA may be shown as follows: **(RP4)**

   Gold Service; Response time is within 50ms, availability more than 99.9%.

   Silver Service; Response time is with in 500ms, availability more than 90%

   Bronze Service; Response time is not guaranteed, availability is not guaranteed.
2. The information about the new End User and the SLA are forwarded to the WSDP. **(RP5)**
3. WSDP accepts the registration of the End User and in turn computes the necessary modifications for the resource allocations of the resources he has purchased from ANO and WSIO.
4. The WSIO is asked to make the modifications on the web servers. **(RP2)**
5. The redirect logic is informed about the new resource requirements. **(RP2)** (delegated)

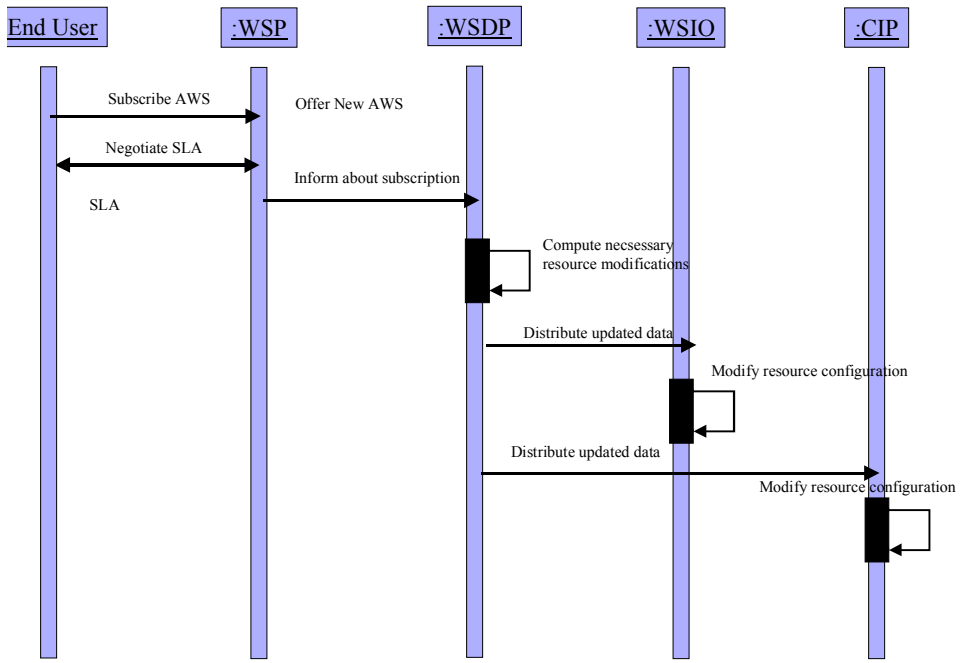


Figure 4-10: Sequence Diagram "Subscribe AWS"

4.2.3.4.7 Exceptions and Alternative Flows

The following alternative flow is considered to be important:

- In order to serve the End User, new web servers and/or redirections have to be configured. The respective interactions are the same as described in "Distribute WS".
- Also, new negotiations with the ANO may be necessary.

4.2.3.4.8 Post-Conditions

End User has negotiated an SLA with a WSP , the necessary resource reservations have been computed and distributed in the network, and in the the End User is ready to use this service under the terms of the SLA.

4.2.3.5 Use Case "Use AWS"

4.2.3.5.1 Use Case Summary

AWS is used by the End User via an ordinary web browser over the public, "non active" Internet.

4.2.3.5.2 Problem Statement

A Web service which has been provided by the Web Service Provider is used by an End User. We assume that the standard HTTP protocol is used for that purpose.

AWS may make End User possible to use web service with high availability.

4.2.3.5.3 Actors Involved in the Use Case

- Web Service Distribution Provider (WSDP)
- Active Network Service Provider(ANSP)
- Web Service Provider (WSP)
- Web Service Infrastructure Operator (WSIO)

4.2.3.5.4 Resources assumed for the Use Case

An End User accesses web pages using web browser on his personal computer or mobile device, then sends request and receive response on HTTP. Storage spaces and computational resources are used for caching web data and for execution the service logic on the service nodes.

4.2.3.5.5 Pre-Conditions

The use cases "Configure Distribution Infrastructure" , "Distribute AWS"and "Subscribe AWS "have been successfully completed. The End User is connected to the Internet.

4.2.3.5.6 Use Case Description of (Main Flow)

1. An End User accesses the web page of the WSP by sending an HTTP request into the network to that web service provider..
2. Eventually, the request passes a redirection node, which will usually be located in the access part of the network (customer side, service provider side). The redirection detects that the request should receive special treatment, either because the End User has subscribed to the WSDP, or because the WSP has asked the WSDP to distribute his services. The request is forwarded to the active code running in the redirection node, which creates a new web service session. **(RP3)**

3. The active code will usually lookup a table containing the user/service provider profile, as well as a dynamically updated table on suitable servers for handling the request (containing their availability, distance, load, capabilities etc.). Based on this information, the redirection will choose an appropriate service node and forward the request to it. Also, the service node which has been choosen for that service session will be stored in a separate table. All subsequent request for that web service session are forwarded to that node. **(RP6)**
4. The request will usually pass one or several active nodes. It will be sent within active packets, which configure these nodes in order to reserve sufficient resources (i.e. a path) to send the replies back and to accommodate further requests. **(RP7)**
5. The reply to the request is computed in the service node and sent back over the reserved path. The computation of the request may require further interaction with the service provider**. (RP 2).**
6. AWS in order to set resource table according to user level or content level. Also QoS level will be mapped on this request. Then packet will be forwarded to next active node, i.e. active router in core network. **(RP2)**

4.2.3.5.7 Exceptions and Alternative Flows

We consider the following exceptions and alternative flows to be important:

- The request does not pass an active node, but eventually reaches the service node owned and operated by the WSP, i.e. it is a normal web service without any involvement of active components.
- The web service session no longer exists due to an exceptional result of "Update AWS".
- The request can not be handled due to insufficient resources.

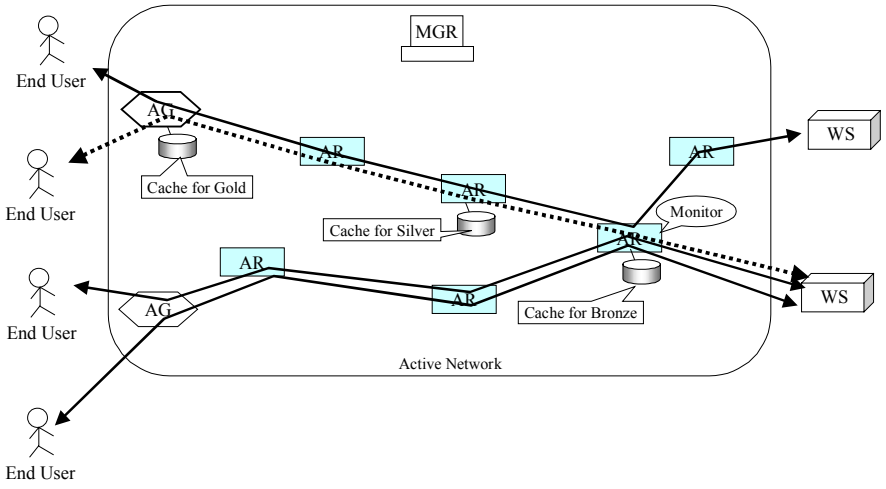
4.2.3.5.8 Post-Conditions

End User has received the reply to his HTTP request.

4.2.3.6 Use Case "Reconfigure Distribution Scheme"

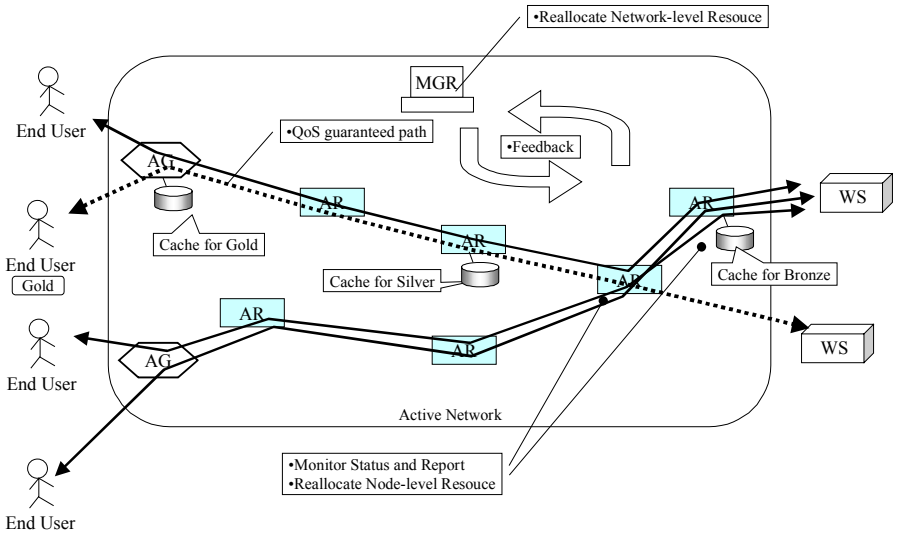
4.2.3.6.1 Use Case Summary

While End Users are using AWS, resource allocations are changed according to status of active nodes or traffic. An example of redirection is shown in Figure 4-11 and Figure 4-12. Active nodes monitor usage of cache space and traffic status.



AG: Access Gateway, AR: Active Router, WS: Web Server, MGR: Management Node

Figure 4-11: Reconfigure Distribution Scheme 1



AG: Access Gateway, AR: Active Router, WS: Web Server, MGR: Management Node

Figure 4-12: Reconfigure Distribution Scheme 2

4.2.3.6.2 Problem Statement

Adapt the traffic distribution strategy to changing network situation and requirements.

4.2.3.6.3 Actors Involved in the Use Case

- Web Service Distribution Provider (WSDP)

- Active Network Operator (ANO)
- Web Service Provider (WSP)
- Web Server Infrastructure Operator (WSIO)

4.2.3.6.4 Resources assumed for the Use Case

The action of this use case is executed without operators intervention. This action is triggered by policy data which are installed each active nodes.

4.2.3.6.5 Pre-Conditions

All installation of AWS has been done by "Configure Distribution Infrastructure" and „Distribute AWS".

4.2.3.6.6 Use Case Description of (Main Flow)

1. Active nodes periodically monitor cache status and traffic status, then report to Network Management System. **(RP3)**
2. Network Managing System analyses status and if necessary, this reallocates cache location. Cache location is considered to move when some unused cache is found or traffic congestion happened.
3. Service Nodes periodically monitors server load status and also reports to Network Management System. Network Management System decides to redirect and notify relevant active nodes, e.g. active access gateways and active routers. **(RP3)**

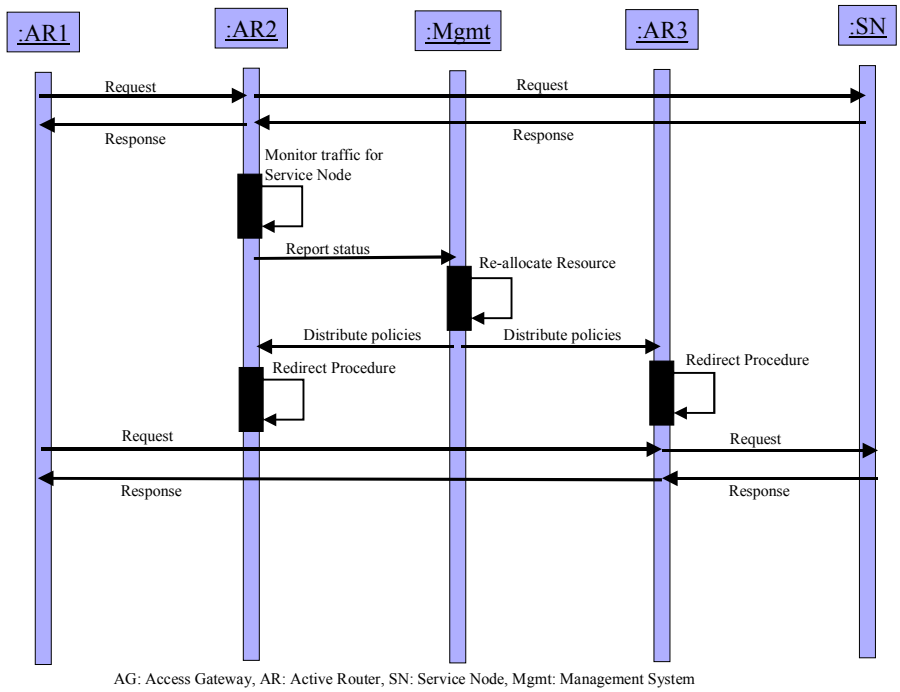


Figure 4-13: "Reconfigure Distribution Scheme 1"

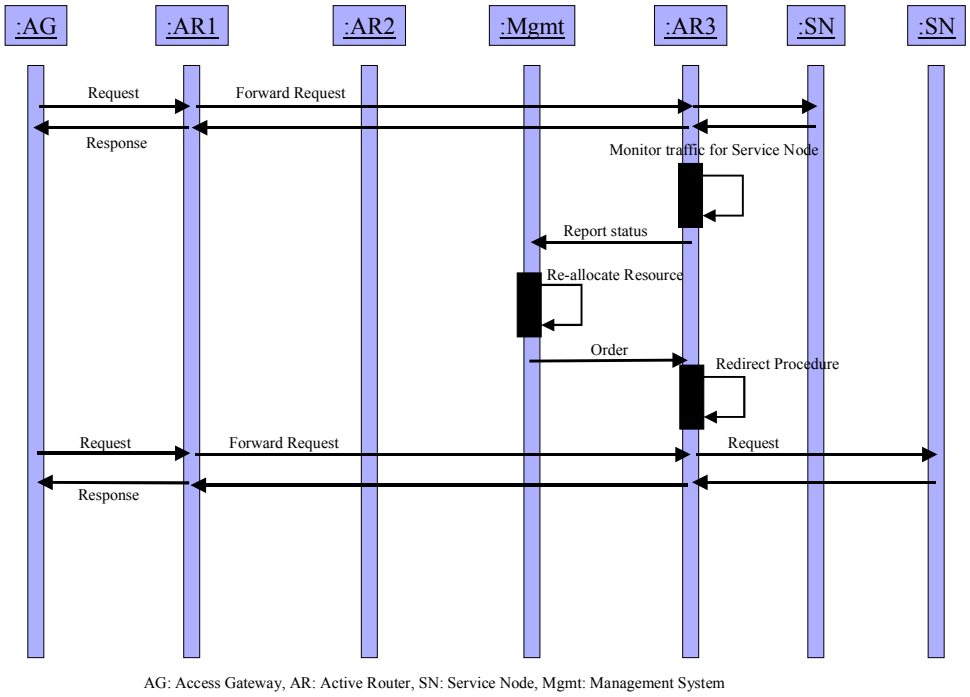


Figure 4-14: "Reconfigure Distribution Scheme (2)"

4.2.3.6.7 Exceptions and Alternative Flows

4.2.3.6.8 Post-Conditions

Resource allocation on some active nodes are updated.

4.2.4 Assessment of the Use Cases

An AWS is to provide differentiated web distribution service among end users. With this concept consumers of web service needs to pay depending on service level that they could accept. Generally web service is provided through internet, so there are some problems, e.g. response delay, waste of bandwidth, packet loss. AWS could give a solution using dynamic control of cacheing and redirection.

In addition, AWS might serve as basis of higher service based on web distribution, such as network auction, network voting system, etc..

4.2.5 FAIN Reference Points Involved in the Application

4.2.5.1 FAIN Reference Points involved in the Use Case "Configure Distribution Infrastructure"

RP2: The reference point between Service Provider(SP) and Active Network Service Provider(ANSP) is mapped by the relation of WSDP and ANO and relation of WSDP and WSIO in an AWS scenario.

RP3: The reference point between Active Network Service Provider(ANSP) and Network Infrastructure Provider(NIP) is mapped by the relation of ANO and CIP and relation of WSIO and SIP.

RP4: The reference point between Consumer(C) and Service Provider(SP) is mapped by the relation of End User and WSP in an AWS scenario.

RP5: The reference point between multiple (SP)s is mapped by the relation of WSP and WSDP in an AWS scenario.

RP6: The reference point between multiple ANSPs is mapped to the relation of different ANOs.

4.2.5.2 FAIN Reference Points involved in the Use Case „Distribute AWS“

RP2: The reference point between Service Provider(SP) and Active Network Service Provider (ANSP) is mapped by the relation of WSDP and ANO and relation of WSDP and WSIO in an AWS scenario.

RP5: The reference point between multiple (SP)s is mapped by the relation of WSP and WSDP in an AWS scenario.

4.2.5.3 FAIN Reference Points involved in the Use Case „Modify AWS“

RP2: The reference point between Service Provider(SP) and Active Network Service Provider(ANSP) is mapped by the relation of WSDP and ANO and relation of WSDP and WSIO in an AWS scenario.

RP3: The reference point between Active Network Service Provider(ANSP) and Network Infrastructure Provider(NIP) is mapped by the relation of ANO and CIP and relation of WSIO and SIP.

RP5: The reference point between multiple (SP)s is mapped by the relation of WSP and WSDP in an AWS scenario.

4.2.5.4 FAIN Reference Points involved in the Use Case „Subscribe AWS“

RP2: The reference point between Service Provider(SP) and Active Network Service Provider(ANSP) is mapped by the relation of WSDP and ANO and relation of WSDP and WSIO in an AWS scenario.

RP4: The reference point between Consumer(C) and Service Provider(SP) is mapped by the relation of End User and WSP in an AWS scenario.

RP5: The reference point between multiple (SP)s is mapped by the relation of WSP and WSDP in an AWS scenario.

4.2.5.5 FAIN Reference Points involved in the Use Case „Use AWS“

RP2: The reference point between Service Provider(SP) and Active Network Service Provider(ANSP) is mapped by the relation of WSDP and ANO and relation of WSDP and WSIO in an AWS scenario.

RP3: The reference point between Active Network Service Provider (ANSP) and Network Infrastructure Provider (NIP) is mapped by the relation of ANO and CIP and relation of WSIO and SIP.

RP6: The reference point between multiple ANSPs is mapped by the relation of different ANOs.

RP7: The reference point between multiple NIPs is mapped by the relation of different CIMs.

4.2.5.6 FAIN Reference Points involved in the Use Case „Reconfigure Distribution Scheme“

RP3: The reference point between Active Network Service Provider (ANSP) and Network Infrastructure Provider (NIP) is mapped by the relation of ANO and CIP and relation of WSIO and SIP.

4.3 Reliable Multicasting

4.3.1 Application Description

4.3.1.1 Overview

The FAIN multicast application use case specifies requirements related to network infrastructure services and the business roles for the design, the implementation and use of an active IP network that will support various kinds of multicast communications. We intend to pave a way for both the re-engineering of state-of-art multicast IP technologies, and the development of new multicast communication services on active IP networks. We also describe some of the technical capabilities of next generations of multicast communications, in light of requirements expressed by networked game platforms and applications. Active IP networks are expected to enable the design and implementation of new forms of multicast protocols, services and applications, that could not (or hardly) be implemented using the existing - traditional - network technologies, especially when properties such as the flexibility of communications service supports (e.g. through modification of associated computations), control of communications through external agents (outside the network), or the provision of multiple and variable QoS properties are required. Also, it is important for network operators to provide a smooth transition of their infrastructures to meet the requirements of new networking capabilities. In that context, it is important to take into account the possibility to re-engineer existing services using new networking technologies. The use case specification in this document also deals with the necessity or integrating existing and next generation services within the same active network platform, by describing the way state-of-art multicast communication services can be implemented using active IP networks.

4.3.1.2 State-of-Art Multicasting

It is now well accepted that scalable and flexible multicasting support is increasingly needed for emerging communication systems. The demand is driven by multimedia applications, especially, by collaborative applications. The same holds for applications, such as scientific computing and distributed simulation. For many of these applications group communication is a natural paradigm. Therefore, proper support within the communication subsystem is required. This demand is acknowledged by various recent approaches that address transport protocols for reliable multicast. The focus of most current proposals is on the support of reliability. Mostly, all group members experience the same level of service, independent of their network attachment and end system equipment. Thus, all participants in the group are provided with a homogeneous quality of service (QoS).

Furthermore the last years the communication environments are becoming increasingly heterogeneous. This imposes new challenges on communication support for multimedia and collaborative applications. Because of that we need active multicast protocols to provide multi-point communication support for large-scale groups with heterogeneous receivers. Active Multicasting nodes inside the network include so-called QoS filters that remove information from continuous media streams in order to reduce data rate for low-end receivers without affecting high-end receivers.

4.3.1.3 Next Generation Multicasting and QoS

4.3.1.3.1 The needs of Active Multicast Services with QoS

Early Internet-based multicast service specifications were limited to the ability to deliver the same data packets and messages to a receivers at a multicast group, with very few guarantee regarding the properties of message delivery. With the development of the Internet, reliable data communications (multicast file transfer), real-time multimedia communications, and (increasingly) distributed applications over the Internet, the needs of multicast services with various guarantees is becoming one of the principal requirements for future Internet services. QoS requirements related to logical time, the ordering of messages, real-time timeliness, reliability or fault-tolerance are some of the most important properties, that the future generations of multicast services must support. Multicast services with real-time delivery, reliable delivery or flexible group control and management mechanisms have been studied largely in the context of traditional IP networks, with a limited success. It seems that some of the difficulties in the design of multicast protocols with such properties rely in the very basics of the design of these traditional IP networks, which do not allow manipulations of network resources, nor modification | superposition of computations. Active IP networks should ease the enhancement of multicast protocols with these properties (or even more complex ones) and enable new applications, provided that multicast issues are dealt with appropriately at the initial design stage.

Concerning the next generation multicast services, the studies presented in this document deals with the provision of QoS related to real-time delivery and reliability, and the ordering of active packets. These topics have been chosen for their popularity in the multicast research community and their use in networked games in particular. Other topics such as security or flexible group management may also be chosen to illustrate the use of active network technologies for next generation multicast services.

4.3.1.3.2 QoS related to Reliable Delivery

Reliable multicast protocols are often approached in networking research as protocols that take into account reliability in message delivery. Usually reliable multicast protocols can be distinguished by the way they tackle reliable delivery and their performance in term of the degree (or the ratio) of reliable delivery they can offer. The reliability ratio can range from 0% (totally unreliable protocols) to 100% (guarantied or reliability). Protocols that can be included strictly within this interval are sometimes called semi-reliable protocols. In the over hand, almost all of the protocols developed in research fields such as computer networking or fault-tolerant distributed computing do not only provide guarantied reliability but also some of them impose additional reliable delivery semantics to make the protocols more suitable for fault-tolerant distributed computing services. Reliable multicast protocol specifications the most usually used in this case imposes the following conditions :

validity: a message sent to a multicast group is eventually received by all the receivers of this group without alteration,

agreement: if a receiver of a multicast group receives a message, then the same message is eventually received by all the receivers of this group,

integrity: a message sent to a multicast group is received at most once by any receiver of this group, and only if it has been previously sent to this group.

Networked game applications seem to need all of these classes of reliable multicast protocols, a requirement that can cause, e.g. performance problems, or is sometimes impossible due to the underlying network. We believe that active network services can be used to overcome these limitations.

4.3.1.3.3 QoS related to Timely Delivery

Timeliness constraints appear in multicast protocols for audio and video streaming, and data communications between entities that interact in a networked game. Like the case of reliable delivery, protocols in this case are required to provide a variable level of real-time guaranty, ranging from 0% (communications without timed constraints) to 100% communications that guaranty that all the timed constraints will be meet. Several multicast protocols have been developed for audio/video streaming over the Internet, that provide a varying level of real-time guarantees, depending on their performance and the supporting network. However, very few of them can provide stringent real-time guarantees, due in part on the impossibility to have an external control over network resources. Active and programmable network services will allow to implement multicast protocols with a better control over real-time requirements.

4.3.1.3.4 QoS related to Ordering

Beyond the re-ordering of data packets received at the transport level according to their sequence number, some applications need more complex ordering mechanisms, that apply to messages sent or received by applications. The following ordering semantics are usually required by distributed applications in general, and networked games in particular:

First In First Out (FIFO) multicast: A reliable multicast service that satisfies a FIFO order, defined among the flow of messages sent and received in a multicast service session. For example, the FIFO order may impose that messages are delivered to receivers of a multicast group according to the order in which they are sent to that group.

Causal multicast: A reliable multicast service that satisfies the causal order, i.e., the partial order that results from the causality relation between message emission and reception events introduced by Lamport [Lam78].

Atomic multicast: A reliable multicast service that satisfies the total order defined as follows : if two receivers *p* and *q* receive both messages *m* and *m'*, then *p* receives *m* before *m'* if and only if *q* receives *m* before *m'*. The atomic order ensures the same view of the sequence of messages received in the system.

Variants of these ordering semantics can also be defined, as well as new ones. Combinations of ordering semantics can also me defined. Several attempts to implement multicast protocols with such ordering semantics leaded to frustrations in the past (performance problem, non practicability, …), due in part to the lack of an adequate network infrastructure. The use of active network services can lead to a better ability to deal with these problems.

4.3.1.3.5 Networked Game Application Framework

Networked gamed that allow multiple players to interact in real-time over the Internet will emerge in the near future. An important challenge with these games is their ability to scale to supports thousands of end-users spread around the world, while maintaining an acceptable QoS. The QoS requirements themselves can vary a lot, depending on the semantics of games, and processing and communications required by parts of games. A new trend is to build a distributed communication and processing platform that will allow to implement virtual environment supports for these games. Networked games may also be used as application frameworks to do research in several distributed simulation systems, e.g., military simulation applications, concurrent engineering, virtual marketplaces, etc.

The availability of multicast protocols with various technical properties is one of the key requirements for networked games, and beyond the networked game application framework, such protocols will allow the implementation of applications with more stringent communication needs than those that can be satisfied by today's Internet. During the past decades, multicast research and the software industry produced various protocol specifications, standards, prototypes and commercial products. However, almost all the resulting products have been tailored for specific application domain and/or they satisfy a relatively limited set of technical properties. For example, the multicast IP protocol allows a host station to send the same packet to a set of receiving hosts over the Internet, with no guaranty in terms of reliable delivery, delay of reception or order of reception of different packets. Other research and products provided multicast protocols with a reliable delivery and/or a particular order of delivery, using a classical point to point transport protocol, but most of such protocols cannot scale to support thousands of receivers, and they do not respect timely delivery. Here, we are concerned with very general multicast protocol frameworks, that are open with regards to the QoS properties that can be ensured, and that can be tailored or adapted to meet the needs of specific applications.

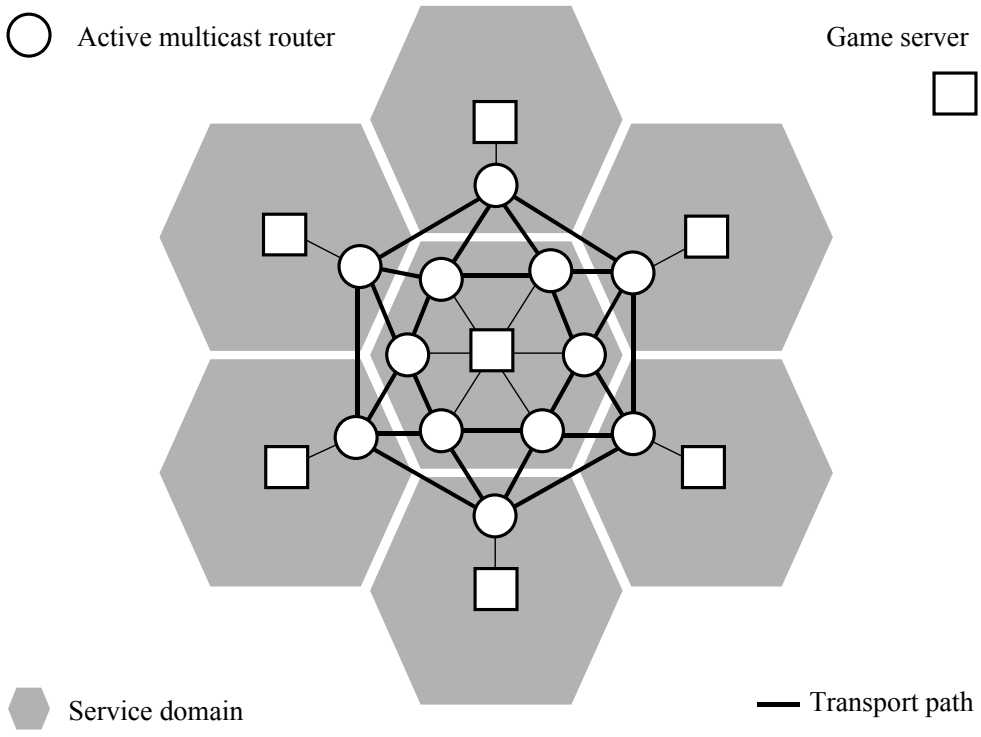


Figure 4-15 networked game infrastructure

4.3.1.4 Justification of the Use of Active Network Concepts

Past experiences with multicast services show that it is very difficult to implement protocols that satisfy many of these features or technical requirements at the same time. For example, there seems to be a trade-off between reliable delivery and real-time delivery, or between ordering mechanisms and scalability techniques, etc. It is very hard to implement protocols that can satisfy all these requirements at the same time using the existing network architecture, and sometimes it is impossible due to initial architectural constructs. For example, it is impossible to guarantee hard real-time delivery constraints with the existing packet-switching networks, simply because, by design, the network has no means to provide such guarantees.

At the over hand applications such as distributed game systems that need protocols with all these requirements are emerging. A solution is to provide multiple-protocol frameworks, i.e., networks and support services that will allow to use several protocols within the same multicast service session. Other solutions will be to develop protocols (or protocol frameworks) that allow to manipulate messages carrying individual technical properties. The later approach seems to be particularly interesting. It as been investigated in the application level framing (ALF)[CT90, FJL97] approach since a certain time, but it seems that there is a lack of an adequate network support for the full implementation of ALF protocol. We expect active networks to provide the adequate architecture and support services for doing so.

Multicast protocols that use active network services could be developed later in WP4, both as a support for policy based network management (specific case) and as an example application of dynamic protocol provision (more general multicast protocols). The later application case can be oriented towards networked game applications, e.g., examples realisations of DIS-like communication services[DIS95]. France Telecom possesses an important background in the study of networked games and their multicast communication requirements[DGS99, Hou00]. We expect the FAIN project to provide a more adequate network infrastructure to experiment these systems.

4.3.1.5 Relevant Documents

The documents referenced at the end of the multicast use case are the basis upon which the use case description is built. These papers will provide the readers an initial understanding of the technical problems posed by networked game platforms and their multicast requirements, multicasting over the internet, and initial ideas concerning the use of active networks to solve these problems. The multicast use case also relies substantially on the FAIN enterprise model, the technical annex, and advances on active networking research during the last five years.

4.3.2 Embedding of the Application

4.3.2.1 Overview of Actors and their Interrelations

The multicast application use case involves the following actors:

End-users(EU): These are owners and/or controllers of end-systems. An end-system is an autonomous, external system or an human (player) driven device that interacts with other end-systems and game service components using the computing and communication infrastructure. End-users are the equivalent of ***consumers(C)*** of the FAIN enterprise model.

Application providers(AP): These are the actors in charge of the provision of applications using active multicast services. These actors are the equivalent of ***services providers (SP)*** of the FAIN enterprise model.

Active network providers(ANP): These are the actors in charge of the provision and the operation of active networks. Active network providers are the equivalent of ***active network service providers (ANSP)*** of the FAIN enterprise model.

Multicast protocol providers(MPP): These are actor that realise and provide an online access to multicast protocols. Multicast protocol providers are instances of ***service component providers (SCP)*** of the FAIN enterprise model.

Game network providers(GNP): Institutions that provide a network of game servers with protocols, hardware and operating systems better suited for the implementation of distributed control, co-ordination and management functions for networked games. Game network providers are instances of ***services providers (SP)*** of the FAIN enterprise model. The services provided by game network provider belong to the same category as, e.g. network management services.

Media network providers(MNP): Institutions that provide an active and internet-based transport network with protocols better suited for the streaming of multimedia information between end-systems and game servers. Media network providers are instances of ***active network service providers (ANSP)*** of the FAIN enterprise model.

Game platform providers(GPP): Institutions that provide a distributed processing infrastructure with control, co-ordination and management functions to support the execution of game environments. Game platform providers are instances of ***services providers (SP)*** of the FAIN enterprise model. The services provided by game platform providers are distributed object services that will support the execution of game services.

Game service and environment providers(GSEP): Institutions that will provide game services and environments themselves, and deploy their game services within game platforms. Game services and environment providers are instances of ***service providers (SP)***, that target end-users.

The figure below describes the relationships between actors involved in the multicast application use cases and their relationships.

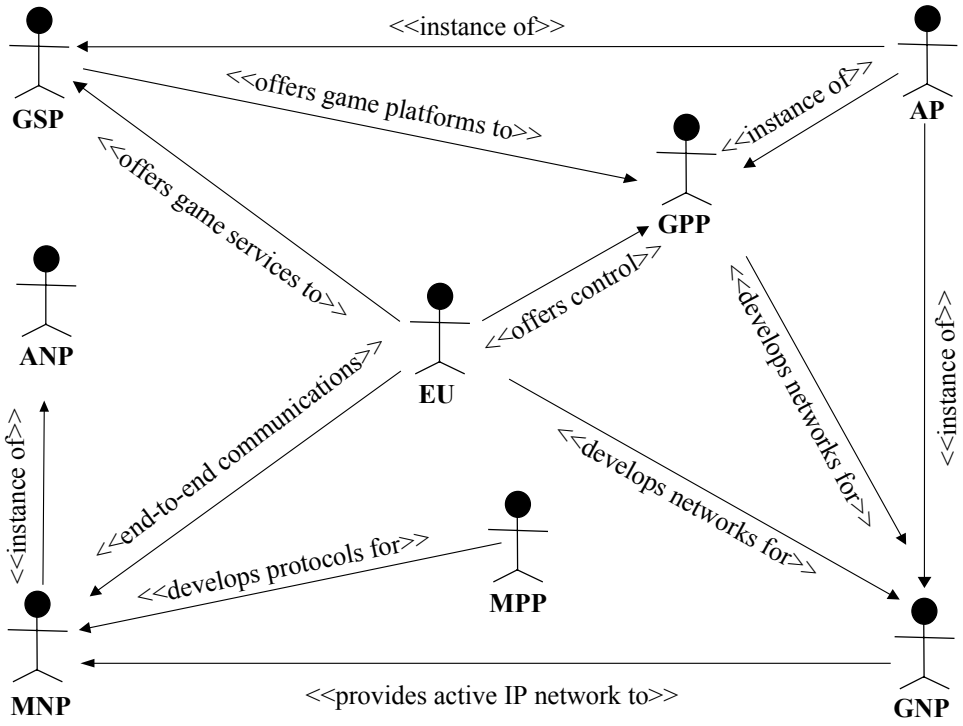


Figure 4-16 actors and their relationships

The table below summarises the relationships between actors in the multicast application use cases and business roles described in the FAIN enterprise model.

Multicast application business role	***FAIN business role***
End-user	Consumer (C)
Application provider	Service provider (SP)
Active network provider	Active network service provider (ANSP)
Multicast protocol provider	Service component provider (SCP)

Game network provider	Service provider (SP)
Media network provider	Active network service provider (ANSP)
Game platform provider	Service provider (SP)
Game service and environment provider	Service provider (SP)

Figure 4-17 multicast application actors and the FAIN business roles

4.3.2.2 Actor Description

4.3.2.2.1 Actors in Multicast Networks for Game Applications

4.3.2.2.1.1 End-User

An end-user uses end-systems to access networked game environments and interact with its homologues. To do so, an end-user needs to access a game server and the active IP network infrastructure. The two accesses are done by means of some access technology (ADSL, UMTS, LAN, …) that is not presented in this multicast application description. End-users establish a commercial relationship with game service and environment providers in which they pay for using game services. End-users need to access game networks for several reason : initial access to the directory of services and software available in the network, subscription procedures, authentication for usage, and so on. End-users also use services provided by game platform providers, for example to download software components needed to participate in a game adequately. Finally, an end-user might need to establish a commercial relationship with a media network provider, in which the end-user pays for media communications needed by games.

4.3.2.2.1.2 Game Network Provider

A game network provider establishes commercial relationships with game platform providers, in which the game network provider supplies equipment (especially game servers) and software that satisfy some technical requirements specified by the game platform providers. Game network providers also uses media networks as transport supports for communications between their systems. Game network providers might also provide various information services to end-users, as it is explained above.

4.3.2.2.1.3 Media Network Provider

A media network provider might rely one infrastructures provided by traditional network operators, or build its own entire network infrastructure. The media network is intended to be an active IP network with protocols better suited for multimedia communications involved by networked games. Media networks will be used as the basic infrastructure that will support game networks. Game network providers are important potential clients of media network providers, since they will pay for the control and media communications that take place between game servers.

4.3.2.2.1.4 Game Platform Providers

Game platform providers establish commercial relationships with game network providers in which they use game networks to support distributed control, co-ordination and management functions needed by game environments. Game platforms are distributed software systems that run on game servers and databases (for persistency properties) and possibly on end-systems. Game platforms offer various services (bearing, security and privacy, persistency, isolation, directory services, binding of game services and end-users) to game service providers, e.g., support the execution of several game services. Game service and environment providers will pay game platform providers for the use of software infrastructure services they need to implement networked games.

4.3.2.2.1.5 Game Service and Environment Providers

Game service and environment providers establish commercial relationships with end-users in which they provide an access to games with tariffs, security, privacy, registration and account management functions to end-users. Game service providers also have commercial relationships with game platform providers as it is explained above. Networked games are designed as distributed and/or replicated services in order to take into account scalability and performance requirements involved by geographical dispersion of players. Typically, game environments are replicated on game servers, each of the servers serving a geographical area, as it is represented in the first figure. A important role of a game platform is to maintain a consistent view of replicated game environments, using distributed control mechanisms.

4.3.2.2.2 Actors in State-of-Art Multicasting over Active Equipment

4.3.2.2.2.1 End-user

An end-user (an individual person or a company) is the actor that requests an application or a service. As a result of such a request, network resources will be used for the execution of the application. In order to request an application, an end-user addresses his request to the application provider, according to already established agreement between these two parties. The agreements could also be re-negotiated dynamically in case the application requires changes on the initial contract.

4.3.2.2.2.2 Application Provider

The role of this actor is to provide specific applications and services, making use of the network infrastructure of active network providers. He must have commercial agreements with multicast protocol providers, in order to be able to use the multicast protocol. The service and application providers also interacts with the Active Network Service Providers, to obtain the network infrastructure for the multicast service and to reserve the necessary network resources for the multicast application.

4.3.2.2.2.3 Active Network Provider

This actor has the responsibility to provide the fundamental infrastructure for the implementation of an active multicast network. He provides the basic facilities to Service Providers, enabling them to build their services.

This actor is responsible for the good operation of the network, that is, the structure of the network as well as the respect of certain commercial agreements and policies in order to keep the information about how many and what kind of users make use of the network at any time.

He is also responsible for the management of active nodes. For example, when multicast packets are about to travel among different domains, he must be able to manage the connection of the nodes, providing (for example) a tunnelling mechanism among the active nodes of the domains.

4.3.2.2.2.4 Multicast Protocol Provider

This actor has to provide the appropriate multicast protocols. The multicast protocol providers develop multicast protocols and make them available to the service providers. The protocol code could be stored on a code server, from where it could be downloaded on demand by a service provider. These two actors should make an agreement so that the service provider is authorised to use the protocols provided by the multicast protocol provider.

4.3.2.3 Resources Assumed for Multicast Applications

4.3.2.3.1 Resource Consumption in Next Generation Multicast Networks

4.3.2.3.1.1 Communication Resources

The game applications assumes the existence of an Internet-based transport network, that can handle broadband multimedia communications with a wide range of constraints, especially in terms of timeliness, ordering of communications and reliable delivery. It also assumes the existence of network connections between game servers and the media network (through one or many routers), so as to realise a game network with connection between any couple of servers. The application also assumes a network connection between any end-system and the server in its geographical area, and a network connection between any end-system and the media network (possibly through the server in its geographical area). Point-to-point and multicast protocols must be available for communications trough the media network, the game network, and communications between a game server and end-systems it is in charge of. Both the media network and the game network are intended to use active network technology to satisfy the diversity of requirements involved by the application. The network connections must have sufficient bandwidth and provide QoS guarantees to allow the transmission of game environment objects in real-time, the transmission of game control events, and so on.

4.3.2.3.1.2 Computing Resources

The end-systems used to access the game applications are personal computers, workstations or wireless terminals with multimedia capabilities. The end-systems must be equipped with a graphical unit that has enough power display portions of game environments in real-time, processors that will support the execution of access software and local processes associated to the execution of game platforms and applications.

The game servers are intended to support the execution of multiple game environments and applications, that belong to several game platforms and applications. The game servers are also intended to run e.g. access control, management or billing related services, processes in which servers are required to maintain information about end-users, game service providers and game platform providers. Thus, the game servers must be equipped with very powerful computing resources so as to satisfy real-time requirements.

The media network nodes (active IP routers) must be equipped with computing power and software to implement the active IP network service mechanisms. These are (according to the initial FAIN active node architecture) an active network processor, a node operating system and platform environment, a service execution environment and network services used by the active IP network protocols.

4.3.2.3.1.3 Storage Resources

Some game applications may require disk space at the end-systems to store information related to game access procedures, and also information related to games, game services/environments and platform providers. Game servers require an important amount of persistent storage resources to support programs used by game platform and service components, and game environment objects. The storage of game environments is particularly important for the provision of persistent game services.

4.3.2.3.2 Resource Consumption for State-of-Art Multicasting

In this paragraph, we examine communication and computational resource consumption. These resources consumption will be explained in the viewpoints of data transfer, data control and data management planes whenever it is possible.

4.3.2.3.2.1 Computational Resources & Data Transfer

In terms of computational resources and data transfer, the node should estimate the amount of bandwidth that it is necessary for the secure data transmission. The node should take under consideration a variety of factors during the computation procedure. For example, an active node should have knowledge of the consumer's required QoS in order to allocate bandwidth in such a way so that these requirements will be met. So, in cases that precision is essential (for example videoconference) the maximum available bandwidth should be allocated. On the other hand (for example packets containing texts) where it is tolerable for some packets to be lost, the node has the ability to allocate the minimum bandwidth so that the session won't end.

4.3.2.3.2.2 Computational Resources & Data Control

By this term we mean the computation that the node should perform to estimate time critical procedures. For example, as we mentioned above in the use case 'Path Reconfiguration', the active node may come to a point where it discovers that the path that has already been configured should alter due to a change to the network (congestion, an already used active node is down etc.). The active node, should have the appropriate processing capacity, that will allow it to perform the reconfiguration as soon as possible and transparent to the end user, so that the packet recovering will be accomplished with the less possible resource consumption.

Data arriving into a node might come with a variant priority (from low to high). The node should be able to calculate what packets are to leave the node first and route them accordingly. Mechanisms for the data managing such as queues handling should be provisioned.

Furthermore, another factor that must be taken under consideration is the number of consumers a node can serve. The more the consumers are the more data amount arrives to an intermediate node. In case that the node can not serve all the session at once, delays or congestion might occur.

Process power can also be consumed in cases that the node has to perform compressing or decompressing of data.

A node will also consume computational power in order to add or delete members in a multicast tree on demand.

4.3.2.3.2.3 Computational Resources & Data Management

The factor that plays an important role in data managing is the storage power that a node may have. A great amount of storage power will benefit the multicasting procedure, as it can lead to a more effective network. For example, as we mentioned in the use case 'Packets' Recovering' the active node can maintain in its cache the packets that have been transmitted towards a consumer. Great storage along with high processing capacity will work to the benefit of the procedure in the sense that the bigger the storage the more transferred packets the active node can store. So, when the active node will be asked to retransmit lost packets it can check its cache so that it will have to retransmit only the necessary packets.

A node will also consume computational power in order to make the initialisation of the multicast tree.

4.3.2.3.2.4 Communication Resources & Data Transfer

During the multicast procedure, while data packets are transferred, the main resource that is consumed is bandwidth. According to the packets that are delivered or the size of the application, the whole or a portion of bandwidth is used. The estimation of the necessary bandwidth that is required for the secure completion of a session is a computational resource consumed by the node. In this case we are interested in the portion of the bandwidth required for the sessions establishment.

The sessions that a node can serve are also limited since although the necessary bandwidth for the transmission might exist, the node can not establish any other sessions.

4.3.2.3.2.5 Communication Resources & Data Control

In terms of communication resource and data control we mean the resource that are consumed to control the data. For example, we can say that in this consumption we can apply the acknowledgements that are sent towards or from the active node and they indicate a change to the network. So, a portion of the bandwidth should not be consumed and it must always be available for such kind of communication.

In addition, mechanisms that will establish appropriate multicast protocols on demand should also exist.

4.3.2.3.2.6 Communication Resources & Data Management

A node should have predefined ways of how to mange data getting into the node before the multicast procedure. By the later we mean the data that arrives to the node during the initial multicast tree construction. By this function, the node should acquaint knowledge of the consumers that it will serve, as well as knowledge of the other nodes with which it will co-operate complete the multicast task. So, the node should manage the data arriving to it, and to distinguish whether this data are packets that need retransmission towards a final end user, or these data are acknowledgements that request for a multicast protocol's set up.

4.3.3 Use Cases and their Interrelations for State-of-Art Multicasting

4.3.3.1 Overview

The multicast application use case specification is intended to show different facets of the use of active network infrastructure to implement state-of-art multicast services, next-generation multicast services better suited for the provision of online networked games that can scale to support thousands of users dispersed on a wide geographical area. We describe 11 multicast application use cases. Use case 1 to 6 describe the use of active network an infrastructure to set-up, use, control and manage, release multicast services with the properties of state-of-art multicast protocols (e.g. standard IP multicast). Use cases 7 to 11 describes principles for QoS based multicasting for next generation multicast services, the use of an active network infrastructure to implement such multicast services, and their use for the realisation of networked games. The use cases are the following:

1. Use case "Initial construction of a multicast tree": This use case describes the use of an active transport network, which is provided and operated by an active network operator, to construct a multicast tree, that will indicated the path for the transmission of multicast packets from a router connected to a sending host system to routers connected to receiving host systems.
2. Use case "Adding a member in a multicast group" : This use case describes mechanisms for adding a new member to a multicast group.
3. Use case "Installing a Multicast Protocol On an Active Node": This use case describes mechanisms for installing multicast protocol components on an active router, prior to the establishment of a multicast session or the routing of a multicast packet.
4. Use case "Deleting A Member From A Multicast Group": This use case describes mechanisms for removing a member from a multicast group.
5. Use case "Deconstructing a multicast tree ": This use case describes mechanisms for releasing a multicast tree that has been built on an active transport network. Such a deconstruction is needed, especially at the end of a multicast session.
6. Use case "Path reconfiguration": This use case describes mechanisms for the reconfiguration of the path used by active packets in a multicast session.

7. Use case "Multicasting with varying reliability": This use case describes principles and mechanisms for the execution of multicasting sessions and the transport of active multicast packets with varying reliability requirements, which migth be a component of general QoS requirements.
8. Use case "Multicasting with varying real-time guarantees": This use case describes principles and mechanisms for the execution of multicasting sessions and the transport of active multicast packets with varying real-time requirements, which migth be a component of general QoS requirements.
9. Use case "Multicasting with dynamic ordering or paquets": This use case describes principles and mechanisms for the execution of multicasting sessions and the transport of active multicast packets with a dynamic ordering of messages and packets, which migth be a component of general QoS requirements related to the computational behaviour of routers.
10. Use case "Multimedia multicasting in a game network": This use case describes the design of a game network and the needs of active routers and game servers to implement multicast protocols thta deal with the communication of multimedia packets with various QoS requirements.
11. Use case "Game application level multicasting": This use case describes examples of multicast communications between game servers, in order to react to communication events expressed by game platforms, game services and environments and client end-systems.

The use case descriptions does not deal with a specific game application, but instead, they focus on computing and communication services and mechanisms for multicasting over an active network, and active multicast network service needed to realise an infrastructure better suited for online, multi-user games.

In FAIN we aim to build an Active and Scalable Multicasting Architecture for the provision of communication services inside the network. It is based on active networking and on a hardware/software co-design in order to improve efficiency. Our primary goal is to provide customized multicast services on demand, thus providing a framework for a flexible open communication platform.

In the past decades a tremendous growth in the use of computer networks in general and the Internet in particular can be noticed. Besides the provided interpersonal communication, like E-Mail, the networks advance in becoming the media for distributed computation, tele-collaboration, distance learning, e-commerce and the like. Many of these applications are inherently based on group communication. Therefore, efficient and scalable group services are needed for proper communication support.

Our Architecture is layered in two main parts. The first layer (lower) concerns the ANSP and the nodes topology, where ANSP can build his routing paths. The second layer concerns the SP and the QoS requirements of the Consumers. According to them the SP collaborates with ANSP for the construction of the QoS paths. Furthermore, SP can contact with SCP and upload code in order to reconfigure the Multicast tree in an Active flexible way for the customized demands.

4.3.3.2 Use Case "initial construction of the multicast tree"

4.3.3.2.1 Use Case Summary

This is the first step in the procedure of multicasting. A service provider wants to construct a multicast tree, in order to send packets to a set of Consumers. He should have knowledge of the exact number of the Consumers as well as their exact location in the network. Using this knowledge, along with additional information such as available bandwidth, QoS etc, the service provider computes the multicast tree and installs the multicast protocol on the necessary nodes.

4.3.3.2.2 Problem Statement

Provide to the active node the essential knowledge of knowing the receivers of the packet it's going to forward.

4.3.3.2.3 Actors Involved in the Use Case

1. Consumer
2. Service Provider
3. Active Network Service Provider

The use case is initiated by the Consumer.

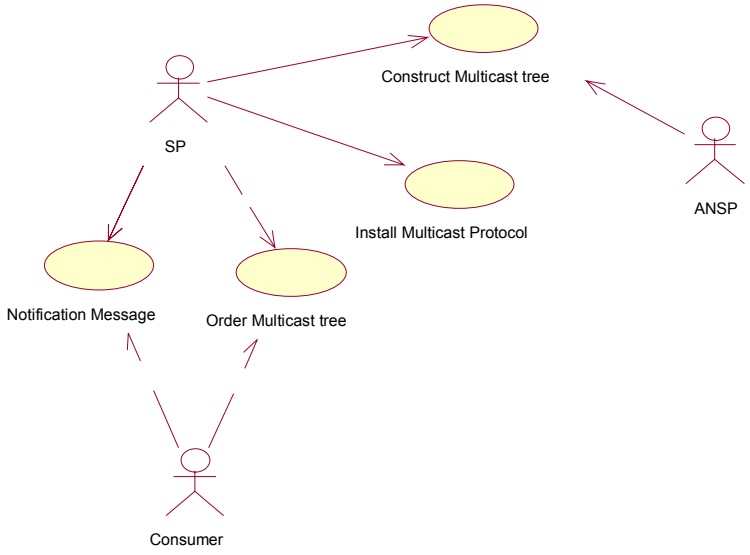


Figure 4-18 actors in the initial construction of a multicast tree

4.3.3.2.4 Use Case Description (Main Flow)

1. A set of consumers contacts with the Service Provider and requests the construction of a multicast tree.

RP4: The Consumer contacts the Service Provider and requests a Multicast tree.

2. The SP requests from the corresponding ANSP to construct a multicast tree according the information given by the consumers. Additionally the ANSP co-operates with SP in order to find the appropriate nodes to host the consumers.

RP2: The Service Provider requests from the Active Network Service Provider to construct the multicast tree.

3. The Active Network Service Provider reserves the appropriate nodes and sets the appropriate routing paths for the consumers and notice the Service Provider that the Multicast tree is constructed.

RP2: The Active Network Service Provider notice the Service Provider that the Multicast tree is constructed.

4. Now the Service provider can install the Multicast Protocol (see use case " Installing a Multicast Protocol On an Active Node"). If the multicast protocol is successfully installed, the Service Provider sends a message to the Consumers to indicate that they are members of the multicast group. If the protocol could not be installed, the Service Provider sends an abort message to the Consumers.

Additionally the Service Provider has the capability to change or modify the installed Protocol because of the Active infrastructure provided by the system.

RP4: The Service Provider informs the consumer that the Multicast tree is ready.

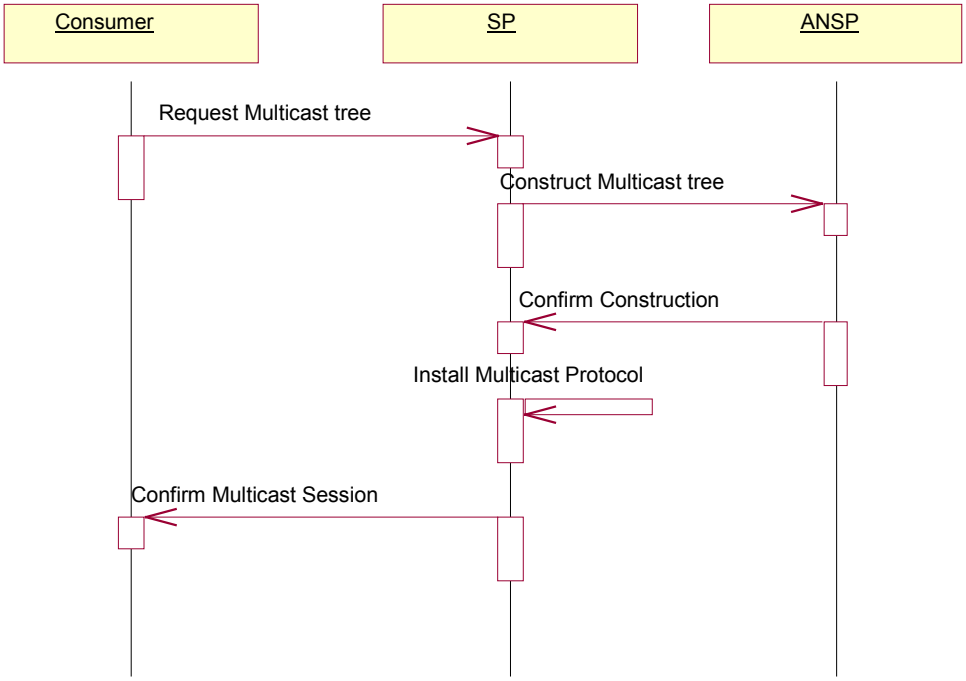


Figure 4-19 flow diagram for the construction of a multicast tree

4.3.3.3 Use Case "adding a member in a multicast group"

4.3.3.3.1 Use Case Summary

In order to be able to receive multicast packets a Consumer must join a multicast group, managed by a Service Provider. The SP checks if he can serve the specific customer and requests from the ANSP to add the node of the customer to the multicast tree. The Active Network Service Provider processes the request and determines which node is needed to serve the specific Consumer. Then he requests from the Service Provider to install the multicast protocol on this node. He also makes the provisioning of the QoS. In this use case we might have to examine two sub-cases. The first and most simple one is when the sender and the receiver belong into the same domain, and their communication can be achieved through one active node that already exist in the multicast tree or in a little more complicated case through the use of additional active nodes of the same domain. The second one is when the sender and the receiver belong to different domains (domains that are managed by different ANSPs).

In the first case where the packets are going to travel inside the regions of the same domain, the procedure that has already been described in the first paragraph is going to be followed. In the second case where different domains are concerned, a more complicated mechanism must be set up. The different ANSP must collaborate to set up an inter-domain multicast tree considering the required resources of the protocol.

4.3.3.3.2 Problem Statement

Provide the capability to a Consumer to be able to receive or sent packets in multicast node at any time, no matter where he is placed in a network or where the sender is located.

4.3.3.3.3 Actors Involved in the Use Case

The actors involved in this use case are the following:

1. Consumer
2. Service Provider
3. Active Network Service Provider

The use case is initiated by the Consumer.

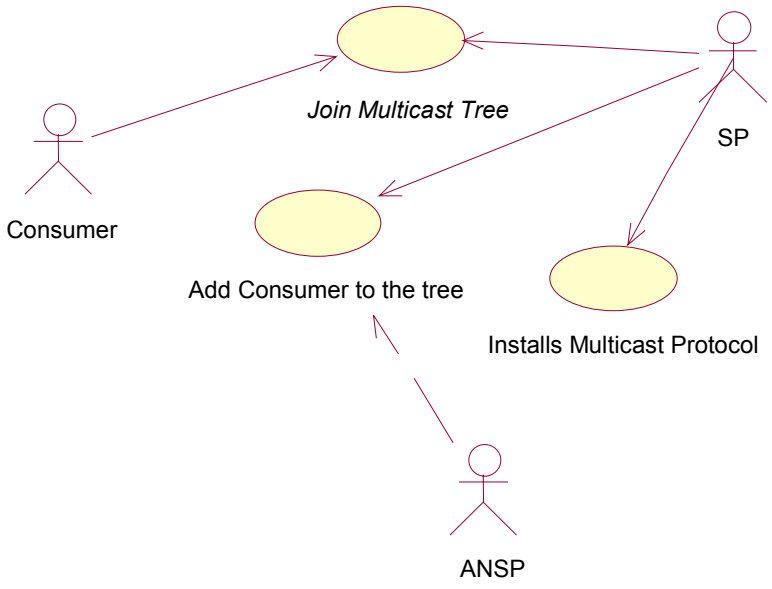


Figure 4-20 actors involved in adding a member to a multicast group

4.3.3.3.4 Use Case Description (Main Flow)

4.3.3.3.4.1 Sender and Receiver are in the same Domain

1. The Consumer makes a request to join a multicast group in order to receive replicates of the data packet that are addressed to the specific group from another Consumer or from a Service Provider. The request is sent to the Service Provider.

RP4: The Consumer contacts with the Service provider to join a multicast group.

2. The SP contacts with the ANSP and asks him to add the consumer in the multicast tree

RP2: The Service Provider co-operates with the Active Network Service Provider in order to identify the node that can serve the requested consumer.

3. The ANSP, using its multicast routing infrastructure, adds the consumer to the tree. The consumer node might be connected to a node, which already is part of the tree, but in another the case the ANSP might have to use additional nodes to connect the consumer to the tree.

4. The ANSP informs the SP about the update of the multicast tree. The Service Provider installs the Multicast Protocol (see use case " Installing a Multicast Protocol On an Active Node"). The procedure for sending multicast packets to the end user is triggered Sender and receiver are in different domains

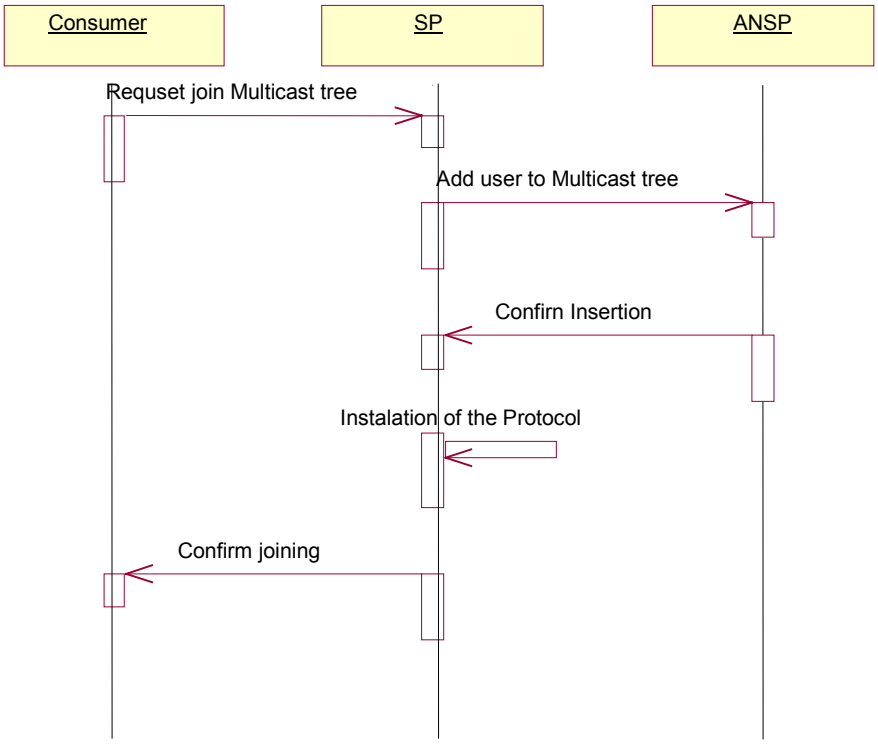


Figure 4-21 flow diagram for adding a member to a multicast group: intra-domain flows

4.3.3.3.4.2 Sender and Receiver are in different Domains

1. The Consumer makes a request to join a multicast group in order to receive replicates of the data packets that are addressed to the specific group from another end user or from a Service Provider.

RP4: The Consumer contacts the Service provider in order to receive multicast packets, or the Service provider can contact the Consumer and to notify him that he will be added in a multicast group in order to receive packets in multicast mode.

2. The Service Provider sends the request to the Active Network Service Provider who will discover that the Consumer that made the request is in different domain than the active node is. So, in order for the consumer to be accessed, the use of additional active nodes in different domains is needed.

RP2: The Service Provider requests from the Active Network Service Provider to add the consumer to the tree

RP6: The ANSP has to contact another Active Network Service Providers in order to establish a multicast connection between the sender and the receiver (in our case the consumer)

3. Through the collaboration of the two ANSP, the different domain nodes are becoming inter-domain (SP co-operates with the second ANSP through the first ANSP) and SP can install the multicast protocol (See use case “Installing a Multicast Protocol On an Active Node”).

4. The procedure for sending multicast packets to the end user is triggered.

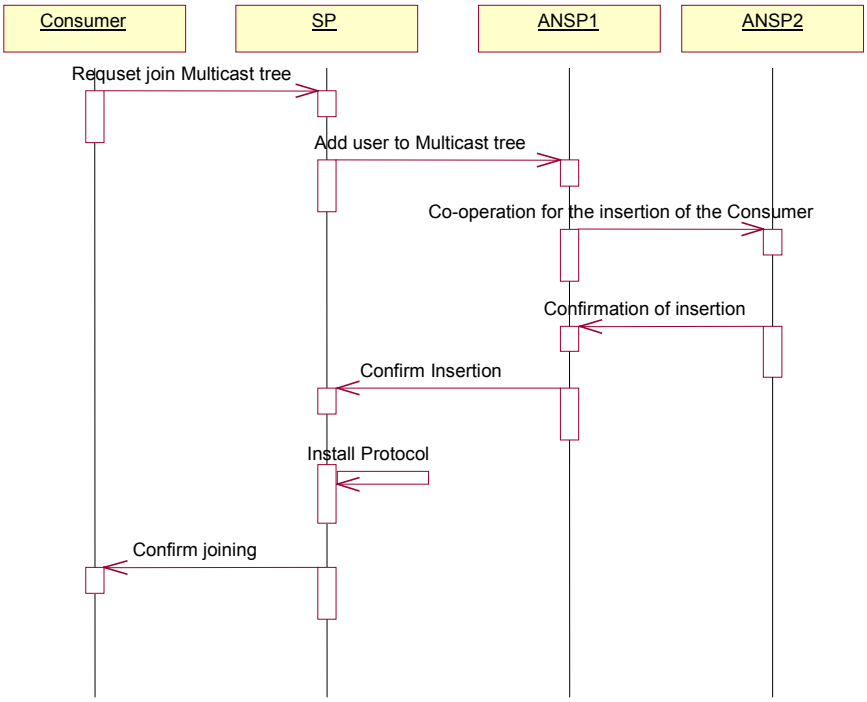


Figure 4-22 flow diagram for adding a member to a multicast group: inter-domain flows

4.3.3.4 Use Case "installing a multicast protocol on an active node"

4.3.3.4.1 Use Case Summary

In this use case, all the interactions that take place during the installation of a multicast protocol on a specific node are going to be examined. In short, when a Consumer request for a service from a Service Provider, the Service Provider should indicate whether the Consumer that requested for the service can be served. That is, whether, for example, some predefined policies are met. This is achieved through the co-operation of the Service Provider and the Active Network Service Provider. Then, and if the request has not been rejected, the Service Provider has to contact the Service Component Provider in order to obtain the appropriate protocols for the service in the intermediate active nodes. The Active Network Service Provider has then to reserve some resources, such as bandwidth or computational power, in order to perform the requested service. Finally the SP installs the multicast code on the node.

4.3.3.4.2 Problem Statement

To explore whether a consumer is allowed to make use of a service, and in the case that he can use it to install it in the active node that will serve this consumer.

4.3.3.4.3 Actors Involved in the Use Case

1. Consumer
2. Service Provider
3. Service Component Provider

4. Active Network Service Provider
5. Network Infrastructure Provider

The use case is initiated by the Service Provider.

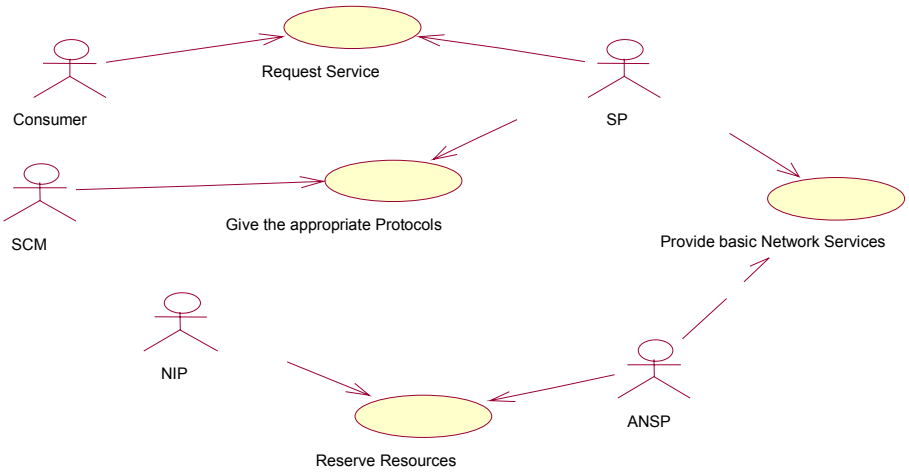


Figure 4-23 actors involved in the installation of a multicast protocol

4.3.3.4.4 Use Case Description (Main Flow)

1. A Consumer makes a request to the Service Provider for a specific service.

RP4: The Consumer Contacts with the Service Provider.

2. The Service Provider has to explore (based on specific agreements or policies) whether the Consumer can have access to this service. The SP checks if it can offer the requested service to the consumer. If the Consumer is not allowed to this service the procedure ends by informing the consumer that his request is rejected.
3. If the Consumer is allowed to use this service, the SP requests from the ANSP to add the consumer to the tree, specifying also the resources (such as bandwidth or computation power), which are needed for the provision of the multicast service.

RP2: The Service Provider contacts the Active Network Service Provider to request the addition of the new Consumer.

4. The ANSP requests from the Network Infrastructure Provider to provide the appropriate resources for the Consumer.

RP3: The ANSP request resources from the NIP.

5. The ANSP adds the Consumer to the multicast tree and built the desired QoS paths above the already done routing paths. If there are not enough resources, the ANSP informs the SP, who in his turn rejects the consumer's request.
6. The Service Provider has also the responsibility to ask from the Service Component Provider the appropriate protocols for the service installation.

RP1: The Service provider downloads the appropriate protocols from the Service Component Provider

7. The SP installs the multicast service code on the node.

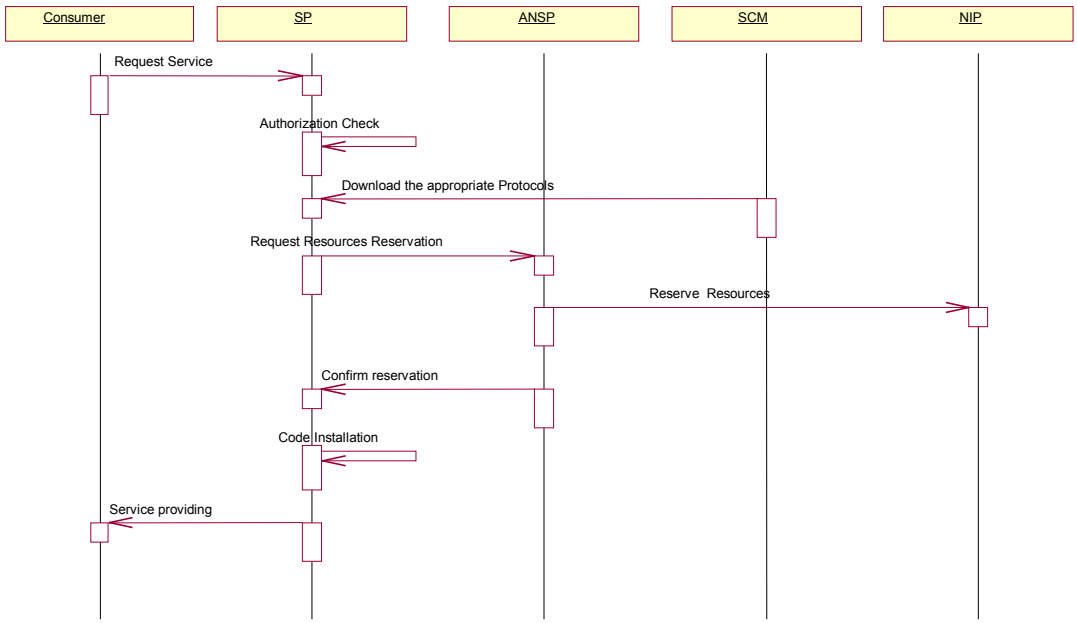


Figure 4-24 flow diagram for the installation of a multicast protocol

4.3.3.5 Use Case "deleting a member from a multicast group"

4.3.3.5.1 Use Case Summary

In this case a Consumer that already belongs to a multicast group may not wish to receive multicast packets any more. So, provision should be taken in order to provide specific actions for the Consumer to be deleted from the multicast group.

4.3.3.5.2 Problem statement

A Consumer should be securely removed from the multicast group. Securely means, that after the deletion only this specific Consumer will not any more have the ability to receive or transmit packets in multicast mode, while all the other members of the group will not undergo any change in their previous behaviour. Additionally, any resources which were reserved in order to serve the specific customer should be released.

4.3.3.5.3 Actors Involved in the Use Case

The actors involved in this use case are the following:

1. Consumer
2. Service Provider
3. Active Network Service Provider

The use case is initiated by the Consumer.

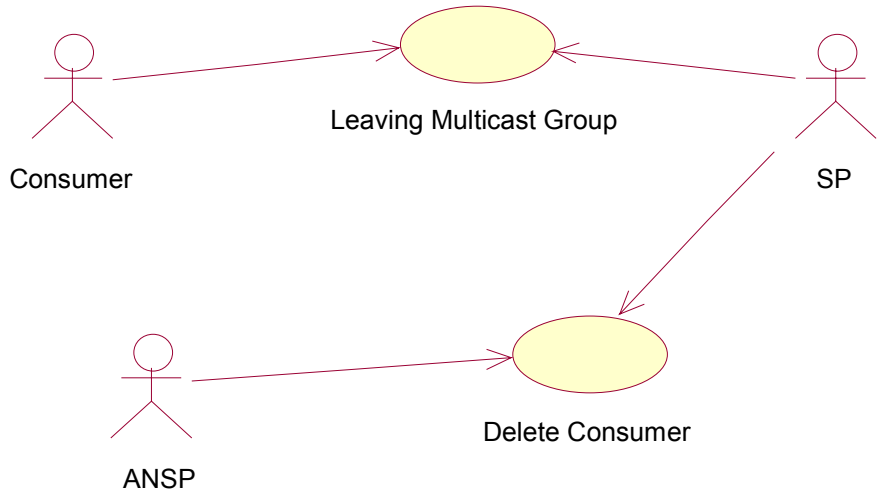


Figure 4-25 actors involved in deleting a member from a multicast group

4.3.3.5.4 Use Case Description (Main Flow)

1. The Consumer makes a request to leave a multicast group in order not to receive multicast packets that are addressed to the specific group, any more. The request is sent to the Service Provider

RP4: a consumer sends towards the service provider a message indicating its will to be deleted from the multicast tree

2. The Service Provider deletes the Consumer from the multicast group. There may exist nodes in the multicast tree, which were used to serve only this Consumer, but now they are no longer needed. The Service Provider un-installs the multicast protocol from these nodes and notifies ANSP to release the resources that had been reserved.

RP2: the Service Provider contacts the Active Network Service Provider in order to release the resources which had been reserved for the multicast protocol, but are no longer needed.

RP6: when consumers belonging to different domains should be able to indicate their will to be removed from a multicast tree that belongs to an active node in a different domain.

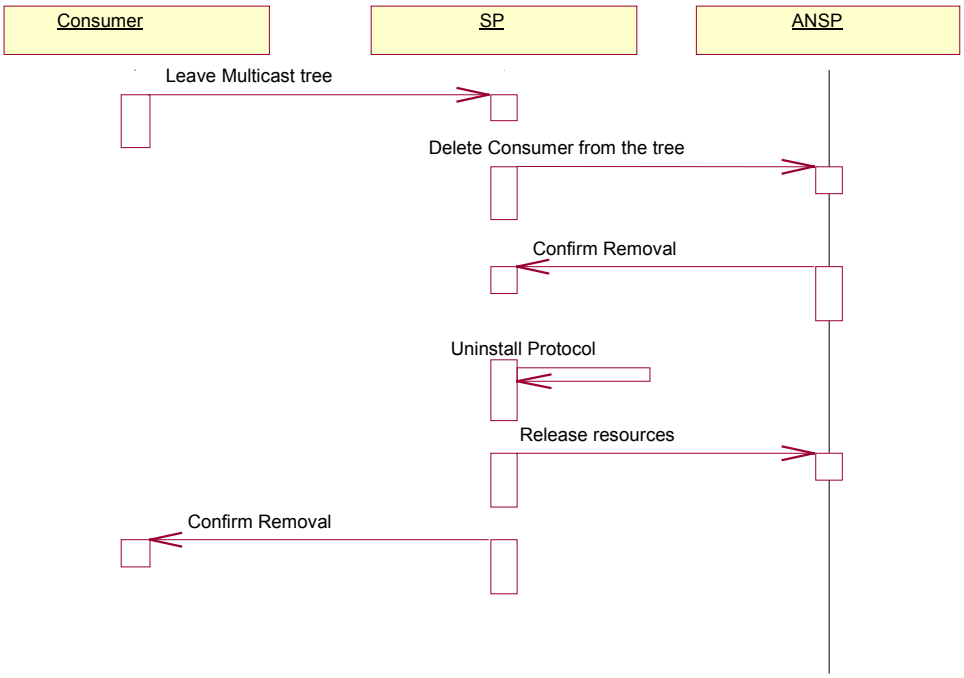


Figure 4-26 flow diagram for deleting a member from a multicast group

4.3.3.6 Use Case "deconstructing a multicast tree"

4.3.3.6.1 Use Case Summary

This use case describes the case where the multicast session finishes either because the source doesn't want to transmit anymore, or because there are no receivers left.

4.3.3.6.2 Problem statement

Due to some reasons the multicasting has come to an end. The active node then has to deconstruct the multicast tree that has been set up for the procedure.

4.3.3.6.3 Actors Involved in the Use Case

1. Consumer
2. Service Provider
3. Active Network Service Provider

The use case is initiated by the Service Provider.

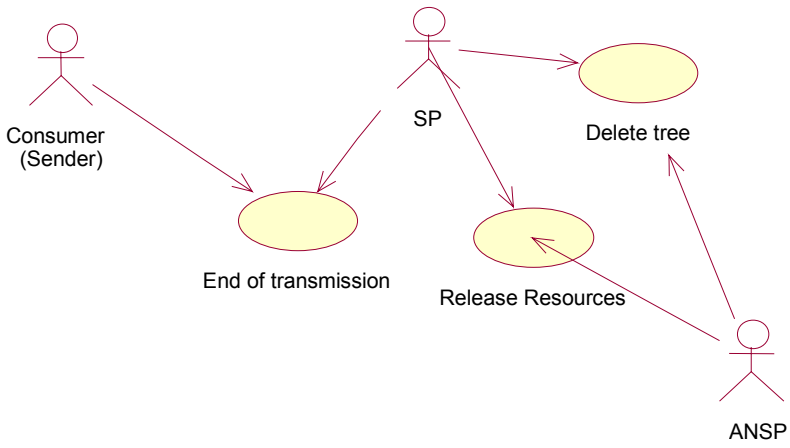


Figure 4-27 actors involved in deconstruction of a multicast tree

4.3.3.6.4 Use Case Description (Main Flow)

1. The Service Provider receives messages from the consumers or from the Sender that multicast session has finished and the multicast tree must be deleted.

RP4: the consumer sends an acknowledgement to the service provider, as their will of deconstructing the multicast tree is concerned.

2. The Service Provider un-installs the multicast protocol from the members of the multicast tree notice the ANSP to deconstruct the multicast tree.

RP2: SP notice ANSP that the Multicast tree must be deconstructed.

3. NSP releases the reserved resources at the nodes of the multicast tree and deletes the tree.

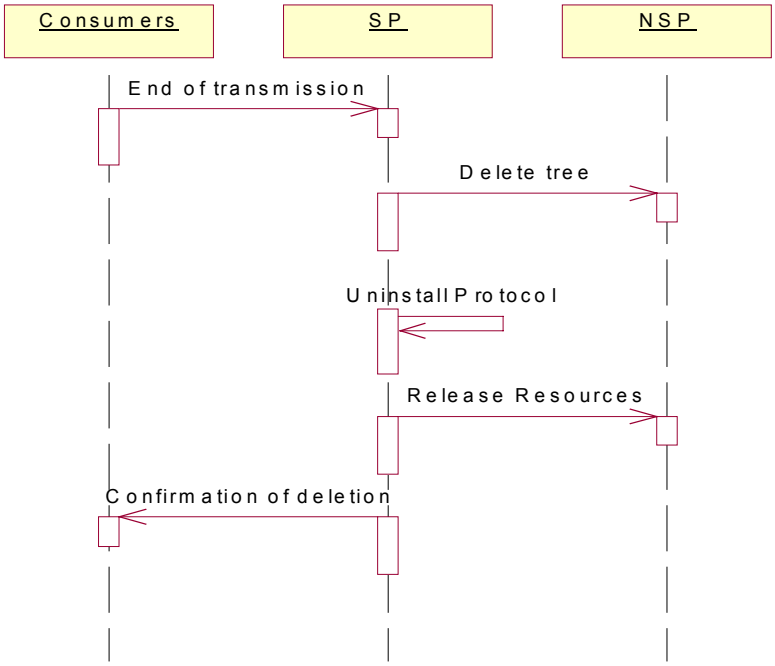


Figure 4-28 flow diagram in the deconstruction of a multicast tree

4.3.3.7 Use Case "path reconfiguration"

When a new Consumer requires upgraded services the multicast tree must reconfigure the bath in order to provide the suitable QoS services.

4.3.3.7.1 Use Case Summary

Provide reliable multicasting so that to meet the QoS requirements.

4.3.3.7.2 Actors Involved in the Use Case

1. Consumer
2. Service Provider
3. Active Network Service Provider

The use case is initiated by the Consumer.

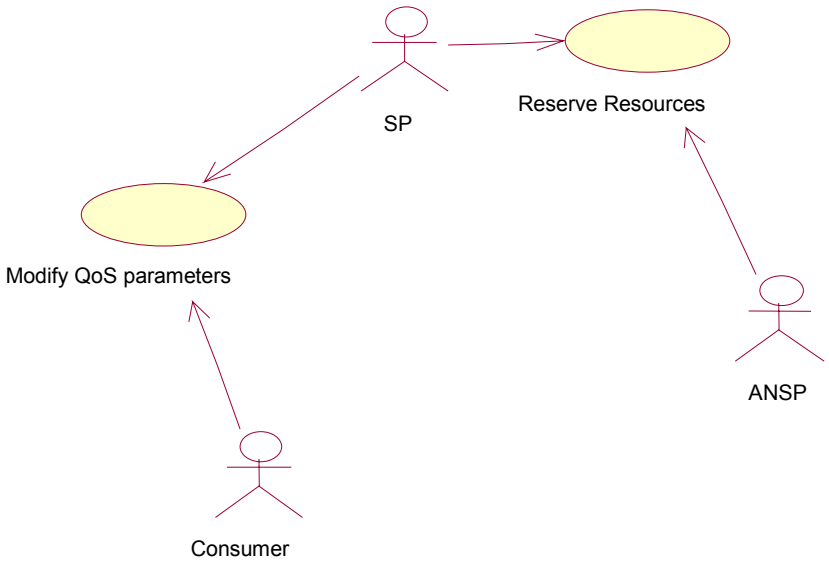


Figure 4-29 actors involved in path recognition

4.3.3.7.3 Use Case Description (Main Flow)

1. The Consumer requests from the Service Provider different QoS service.

RP4: The Consumer request new QoS path.

2. The SP computes the new QoS path and Installs the Protocol if additional nodes are needed. (see use case “Installing a Multicast Protocol On an Active Node”) and notice the ANSP to examine whether or not the node belong to a different domain.
3. If the consumer is hosted in an inter-domain node the Active Network Service Provider notice the other Active Network Service Provider about the new QoS path.

RP6: The collaboration of many Active Network Service Providers provides a wide infrastructure onto which all the consumers, independent to their location in the network might be served.

4. The Service Network Provider transmits the packets to the Consumers through the newly reconstructed multicast tree.

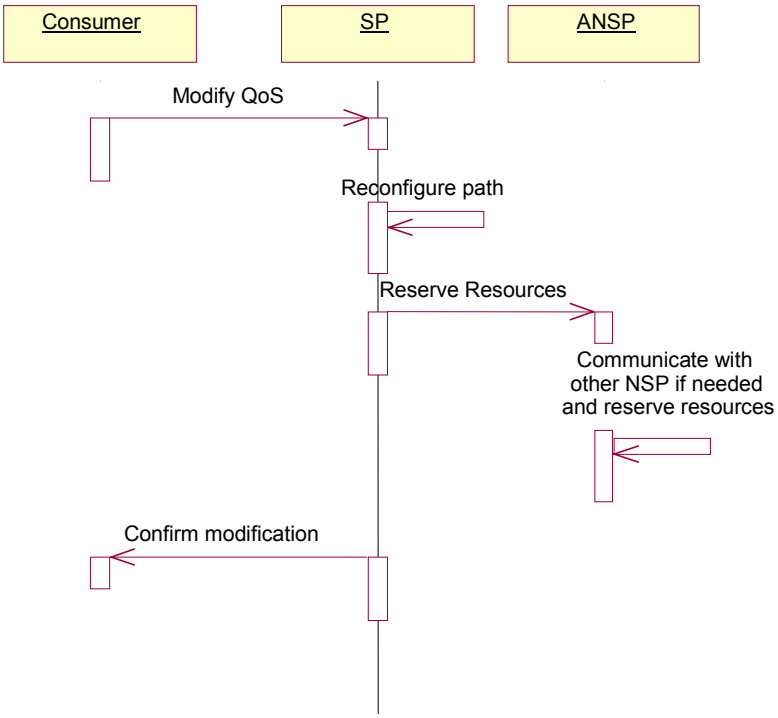


Figure 4-30 flows in path recognition

4.3.4 Use Cases and their Interrelations for Multicasting with Variable QoS and Applications to Networked Games

4.3.4.1 Use Case "multicasting with varying reliability"

4.3.4.1.1 Use Case Summary

This use case describes and illustrates the use of an active IP network to implement multicast protocols that can satisfy per-packet reliability requirements, that can require an arbitrary variation within a multicast session. As an illustration of the applications of such a multicast service, the figure below shows the variation of the percentage of tolerated packet loss for two multicast sessions:

- *session 1* represent a traffic that starts with a strong reliability, and that can tolerate an increasing percentage of packet loss with the time, that culminates with 100% packet loss tolerance at the end of the service session.
- *session 2* represents a traffic that start with the ability to tolerate the loss of almost all the packets, and that requires a rapid fall of this high percentage of packet loss to an acceptable level, and then a progressive decrease of that percentage to meet the requirements of a fully reliable traffic.

The reliability requirements for service session 1 could be those of a simulated (game environment) video sequence in which the initial frames must be transmitted reliably, while the others can be reproduced. Also, we can imagine a service session that starts with a learning and/or path configuration period in which sequences of packets could be sent to network nodes in order to prepare some of the node to the traffic.

We expect an active networks design that will allow a dynamic management of network resources, an end-to-end admission control framework that will enable the implementation of various admission control mechanisms and routing protocols, so as to satisfy reliability requirements expressed by individual multicast sessions or multicast packets.

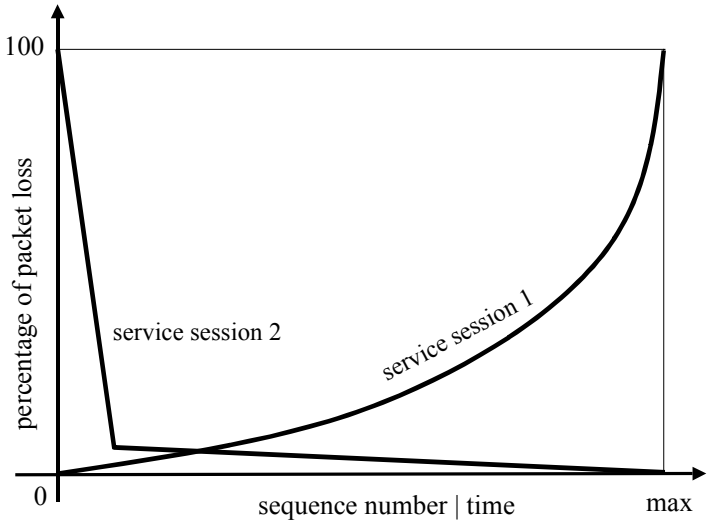


Figure 4-31 multicast service sessions with variable reliability requirements

4.3.4.1.2 Problem Statement

This use case specification is intended to give en overview of service quality required by active network applications, with regards to reliability requirements. Online networked games yield the needs of multicast protocols that can satisfy a varying percentage of reliable delivery, ranging from 0% (unreliable delivery, e.g. IP multicast) to 100% (fully-reliable delivery, fault-tolerant multicast protocols), within the same multicast session. This means that, in the most general case, within the same session, some of the active packets will require an unreliable delivery, others can not tolerate any loss, others can tolerate a relatively low percentage of loss, and so on. In this use case, we illustrate different facets of the problem and the use of such a multicast service.

4.3.4.1.3 Actors Involved in the Use Case

This use case specification involves one or many game network provider (the owners of client systems), one or many Active Network Service Providers, and one or many protocol providers (service component manufacturers). It also involves actors supplying game servers and storage devices, system software running on game servers. However, the specification does not focus on these later actors, since they do not play a key role in the realisation of reliable multicast protocols. Actors playing a key role are the game network providers and some of the actors involved in the provision and the operation of the active IP network infrastructure.

4.3.4.1.4 Resources Assumed for the Use Case

The use case assumes the existence of an active IP infrastructure for the transport of multimedia packets with various QoS requirements, and the can be programmed and configured to meet the requirements of specific packets, using adequate protocol components from multicast protocol providers.

4.3.4.1.5 Pre-Conditions

The active IP network must be operating, along with a set of protocols that can be used for specific packets or multicast sessions.

4.3.4.1.6 Use Case Description (Main Flow)

The figure below show the flow diagram associated to a reliable multicast communication session, in which reliable, unreliable and semi-reliable packets can be sent to at any time during the multicast session. The traffic associated to each of these classes of packets can imply the use of different protocol components, an online negotiation between the active IP network operator and multicast protocol providers that can satisfy specific reliability requirements, dynamic end-to-end network control and resource management operations to satisfy the required reliable delivery.

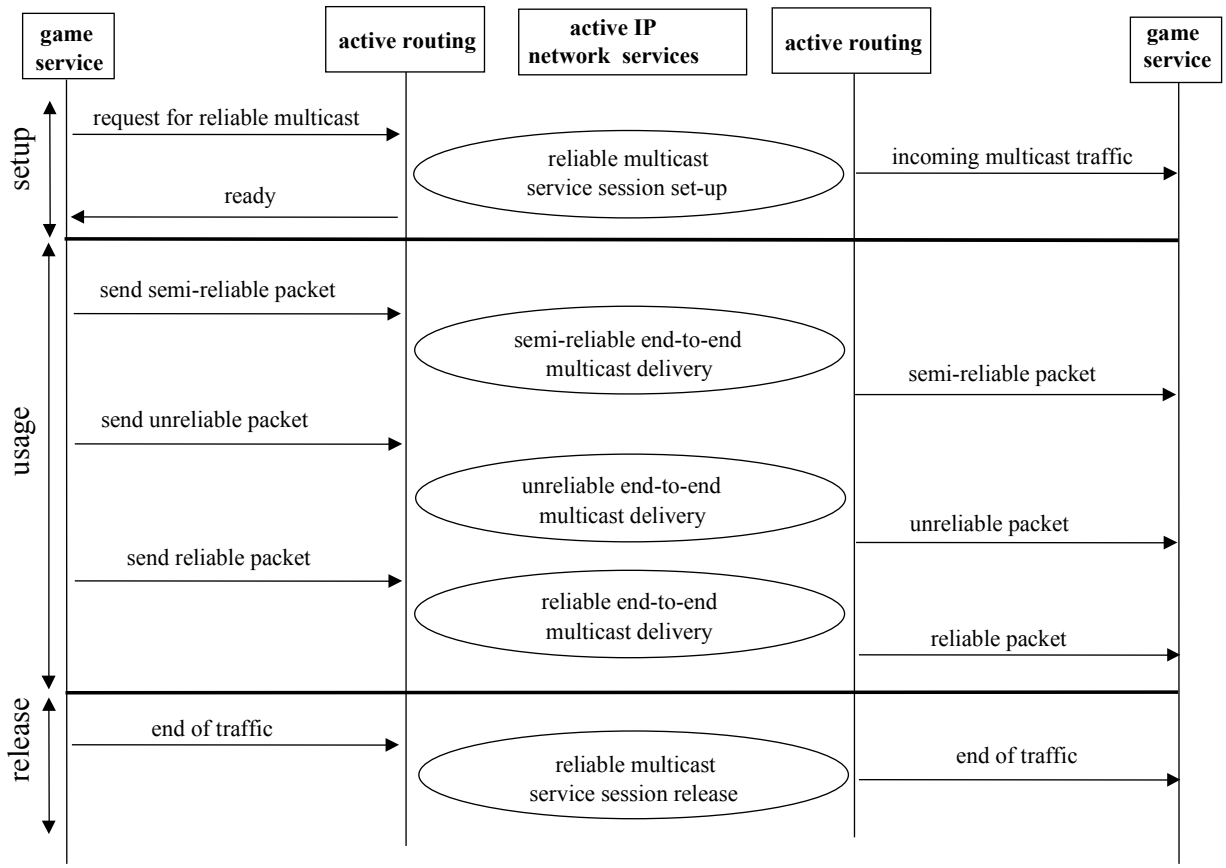


Figure 4-32 flow diagram of multicast communications with variable reliability requirements

At the beginning of a service session, the client game server must collaborate with the transport network to set-up the multicast service session. Then, multicast packets can be sent with their individual reliability requirements. At the end of the service session, its representation within the network and receiving systems must be released.

4.3.4.1.7 Exceptions and Alternative Flows

The figure below shows a situation in which some of the reliability requirements cannot be satisfied by the network. In such a case, depending on the reliable traffic class required by a packet, its transmission may be cancelled, submitted to another active IP network operator (that has both technical and commercial links with the failing operator) or transmitted with service degradation.

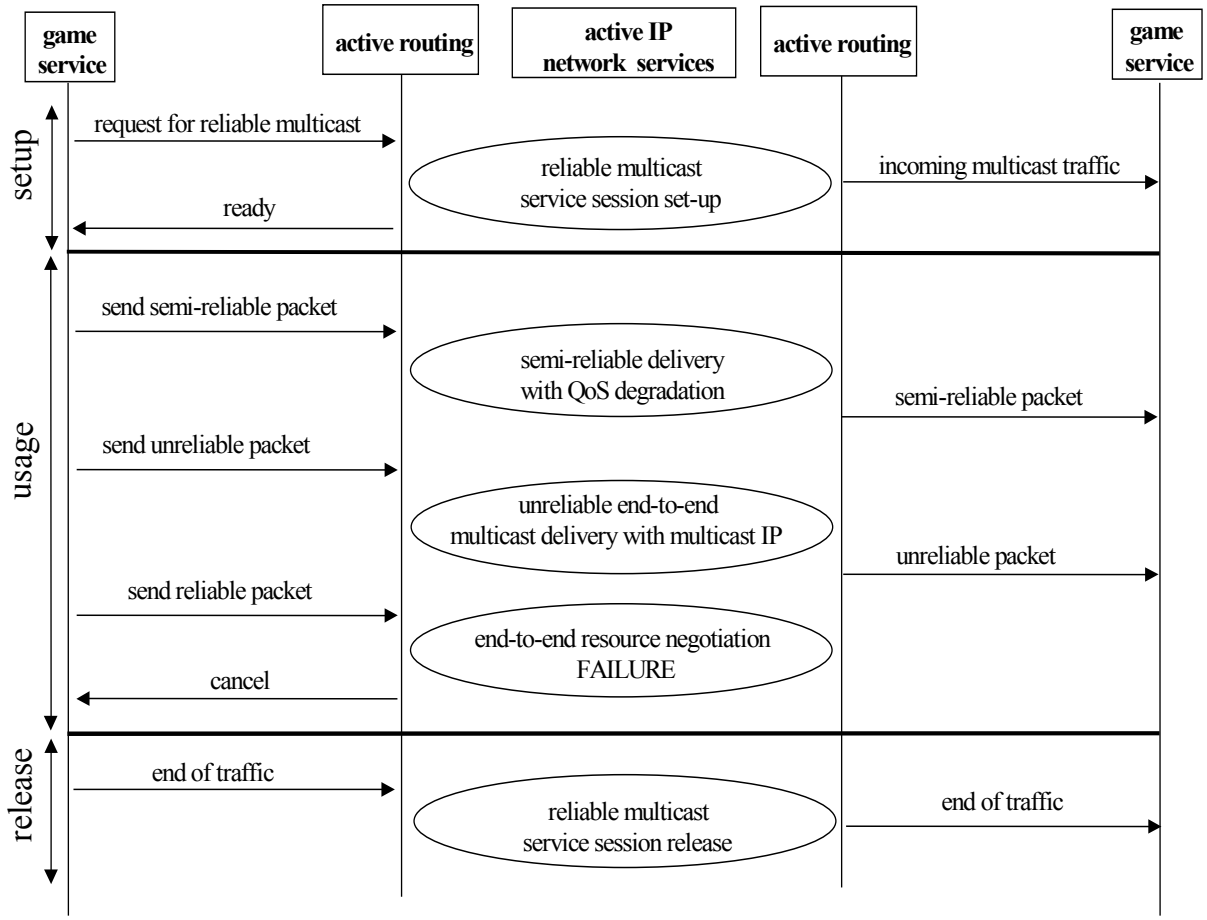


Figure 4-33 Exceptions and alternative flows in reliable multicast

4.3.4.1.8 Post-Conditions

When a flexible reliable multicast protocol is set-up and functioning, game network providers can deploy game platforms, environments and services with arbitrary requirements in terms of reliable communications. Game environments, services and users will be admitted to the platform according to the capability of supporting active IP networks to satisfy their reliability requirements.

4.3.4.2 Use Case "multicasting with varying real-time guarantees"

4.3.4.2.1 Use Case Summary

Like in the case of reliable delivery, we describe and illustrate the use of an active IP network to implement multicast protocols that can satisfy varying, per-packet real-time requirements. Distributed simulation applications and networked games in particular present the need to multicast multimedia packets with variable real-time delivery requirements such as:

- Hard transmission delay, e.g. the packet must be delivered to receivers in a fixed time. Advanced or late delivery cannot be tolerated.
- Packets must be delivered to receivers within in fixed time interval. Advanced or late delivery (related to this time interval) are not tolerated.

- Packets must be delivered within a fixed time interval or a fixed delay, with a certain tolerance in the time or interval of delivery, e.g. packets can be delivered in advance (or late) of a specified time.

Active networks are expected to enable the implementation of protocols that can guarantee all these timeliness requirements within the same multicast session. This will be achieved by the use of multiple protocols from several protocol providers, adequate network service control and resource management strategies within multicast communication services.

4.3.4.2.2 Problem Statement

This specification gives an overview of service quality required by active network applications, with respect to real-time requirements. Online network games require multicast services and protocols that can guarantee variable real-time requirements such as "hard timely delivery", "delivery within a fixed time interval", "delivery within a fixed time or time interval with allow variation". The problem is to implement multicast protocols and service supports that can realise such requirements, within a same multicast session.

4.3.4.2.3 Actors Involved in the Use Case

Main actors involved in this use case specification are game network provider, Active Network Service Providers and protocol providers. The use case also involves actors supplying game servers and storage devices, system software running on game servers, but we do not focus on these later actors, since they to not play an important role in the understanding of this use case.

4.3.4.2.4 Resources Assumed for the Use Case

The use case assumes the existence of an active IP infrastructure for the transport of multimedia packets with various real-time requirements, and that can be programmed and configured to meet the real-time requirements of specific packets, using adequate protocol components from multicast protocol providers.

4.3.4.2.5 Pre-Conditions

The active IP network must be operating, along with a set of protocols that can be used for specific packets or multicast sessions.

4.3.4.2.6 Use Case Description (Main Flow)

The figure below shows the flow diagram associated to a real-time multicast communication session, in which hard real-time, non real-time and soft real-time packets can be sent. Like in the case of reliable delivery, ensuring a real-time communication session with these requirements requires the use of various protocol components, or new generations of protocols that can adapt their service offer to the needs of individual packets. The implementation of such protocols relies heavily on end-to-end network control mechanisms and network resource management.

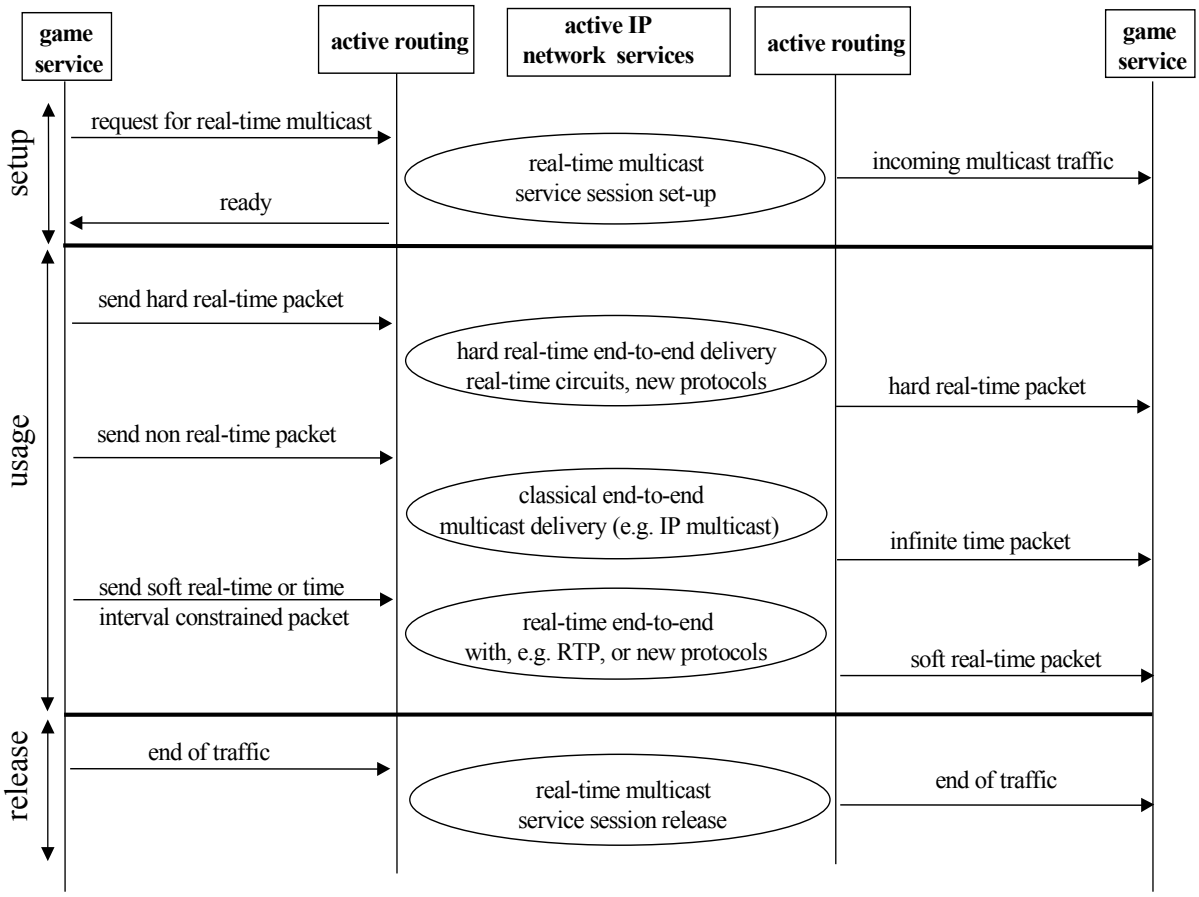


Figure 4-34 flow diagram of multicast communications with variable real-time requirements

4.3.4.2.7 Exceptions and Alternative Flows

The figure below shows a situation in which real-time requirements cannot be satisfied by the network, due to resource limitations or the unavailability of real-time protocols that can satisfy requirements expressed by the incoming packet. In this situation, the incoming packet transmission may be cancelled or submitted to another active IP network operator, that can satisfy the required traffic and that has commercial and technical links with the operator that failed to provide the service.

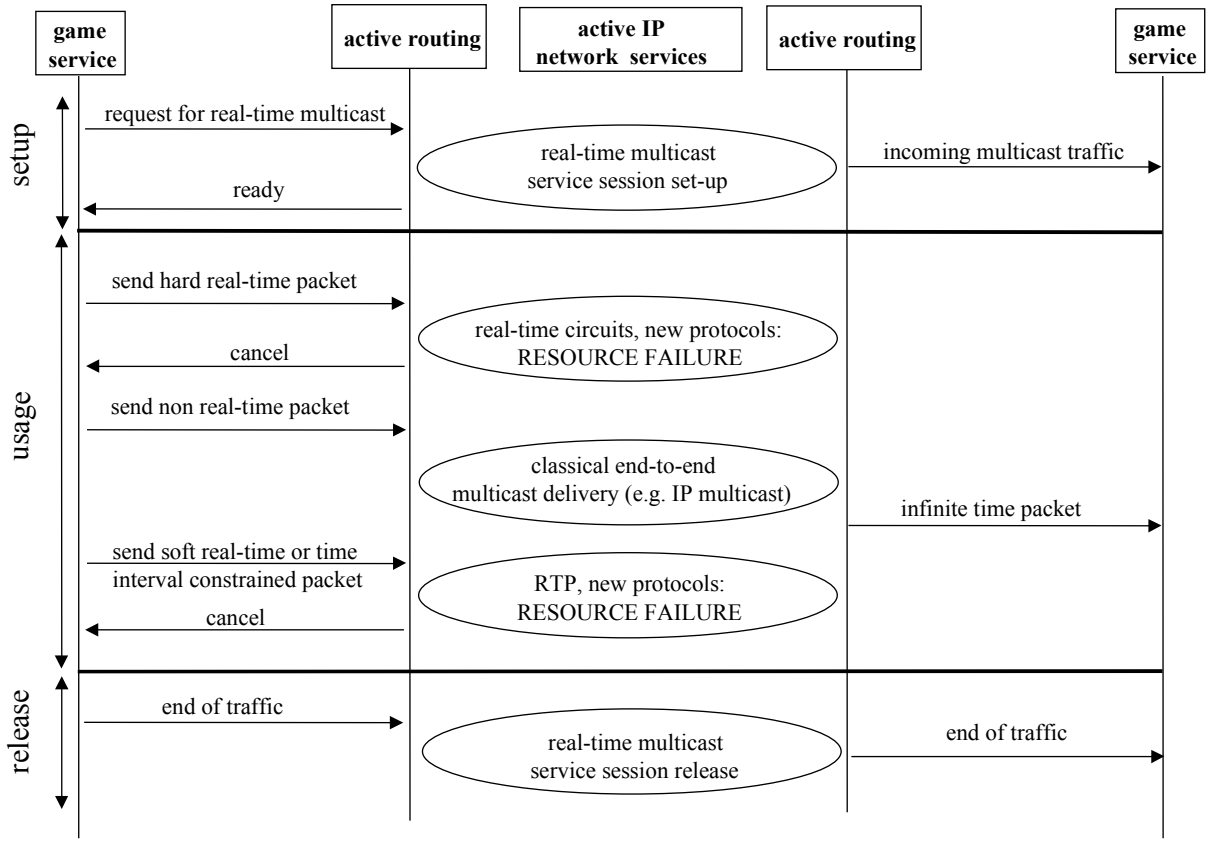


Figure 4-35 Exceptions and alternative flows in real-time multicast

4.3.4.2.8 Post-Conditions

When a flexible real-time multicast protocol is set-up and functioning, game network providers can deploy game platforms, environments and services with a wide range of requirements in terms of real-time communications. Game environments, services and users will be admitted to the platform according to the capability of supporting active IP networks to satisfy their real-time requirements.

4.3.4.3 Use Case "multicasting with dynamic ordering of packets"

4.3.4.3.1 Use Case Summary

This use case describes and illustrates the use of active networks to implement multicast protocols and services that can guarantee the delivery of multicast packets and messages with ordering semantics needed in the control and co-ordination of various distributed applications, the simulation of graphical environments and networked games in particular. Several ordering semantics of messages in reliable multicast communications have been specified in the field of distributed fault-tolerant computing systems [Mul94, VT94], but their implementations are still to be realised, due in part to the lack of an adequate network infrastructures and services. The figure below presents the fundamental ordering semantics have been identified, and their relationships. Other ordering systems can be combined using these fundamental ordering semantics. We expect active networks to allow, through their capability to handle arbitrary computations associated to packets and multicast services, the implementation and combination of ordering mechanisms at transport network nodes.

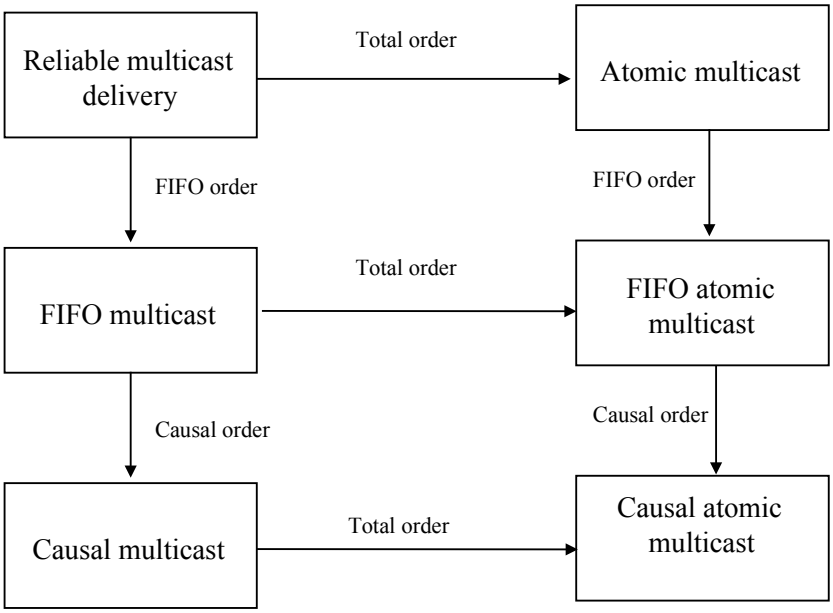


Figure 4-36 ordered multicast semantics

4.3.4.3.2 Problem Statement

This use case specification gives an overview of basic ordering semantics between active packets and messages in a multicast session, and the needs to combine these basic ordering semantics to satisfy arbitrary order notions expressed by game applications. In networked game infrastructure modelled using discrete event simulations, these network service features will enable an efficient implementation of event communications between game servers, game platforms, environments and services, end-users, while guaranteeing a consistent view of the environment in the simulation space and in time.

4.3.4.3.3 Actors Involved in the Use Case

The actors involved are game network provider, Active Network Service Providers and protocol providers. The use case also involves actors supplying game servers and storage devices, system software running on game servers, but the role of these later actors is not described.

4.3.4.3.4 Resources Assumed for the Use Case

The use case assumes the existence of an active IP infrastructure for the transport of multimedia packets with various ordering semantics, and that can be programmed and configured to meet specific ordering requirements between a sequence of packets or messages within a multicast session.

4.3.4.3.5 Pre-Conditions

The active IP network must be operating, along with a set of protocols and services at active nodes, that can maintain temporal relationships between packets and messages.

4.3.4.3.6 Use Case Description (Main Flow)

The figure below shows the flow diagram associated to ordered multicast communications : a FIFO order sequence, a causal order sequence, a totally ordered delivery sequence, and a combination of FIFO, causal and total orders so as to realise a causal atomic order. The active IP network operator must co-operate with one or many active multicast protocol providers, in order to use and combine protocols components with different ordering semantics.

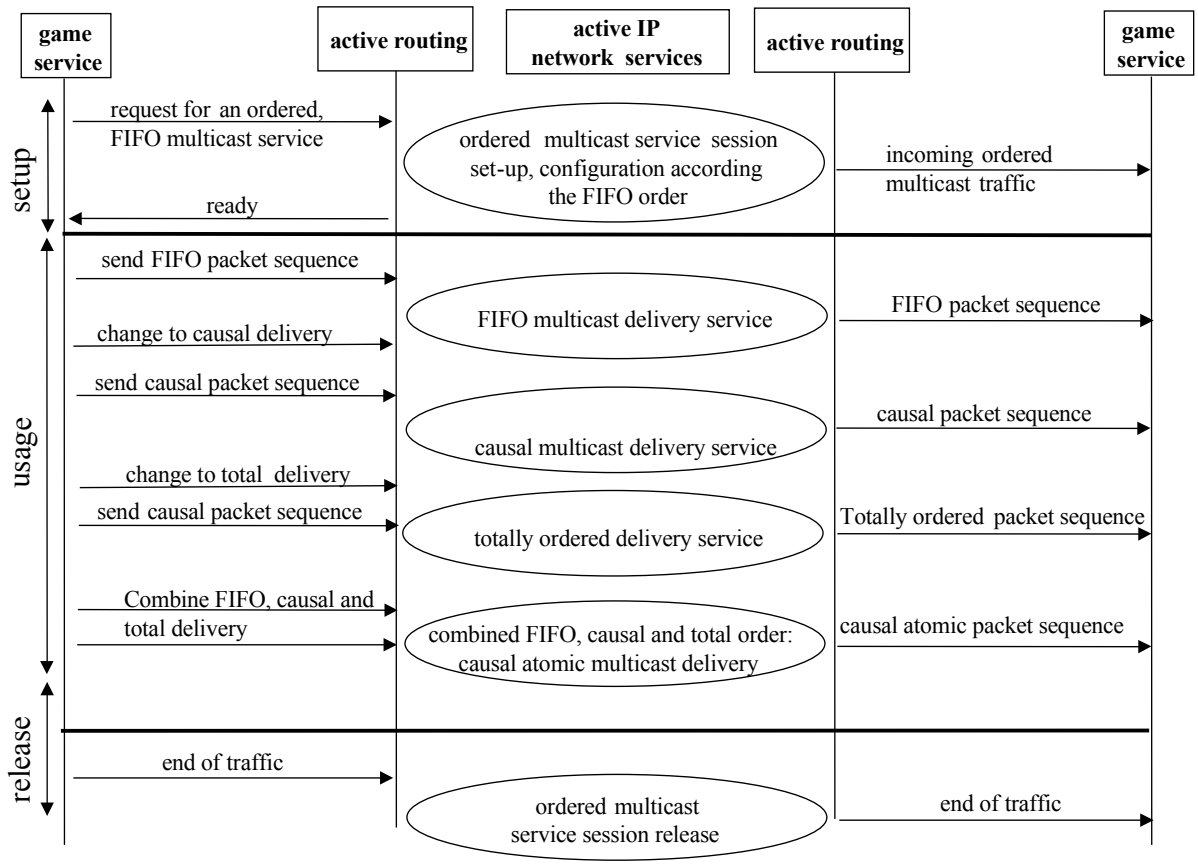


Figure 4-37 flow diagram of multicast communications with variable ordered delivery

4.3.4.3.7 Exceptions and Alternative Flows

The figure below shows a situation in which ordering causal and total orders cannot be implemented by the network, typically because of the lack of a multicast protocol provider that can satisfy such ordering requirements. Service requests with these requirements issued by a client system will be cancelled or directed to another network operator.

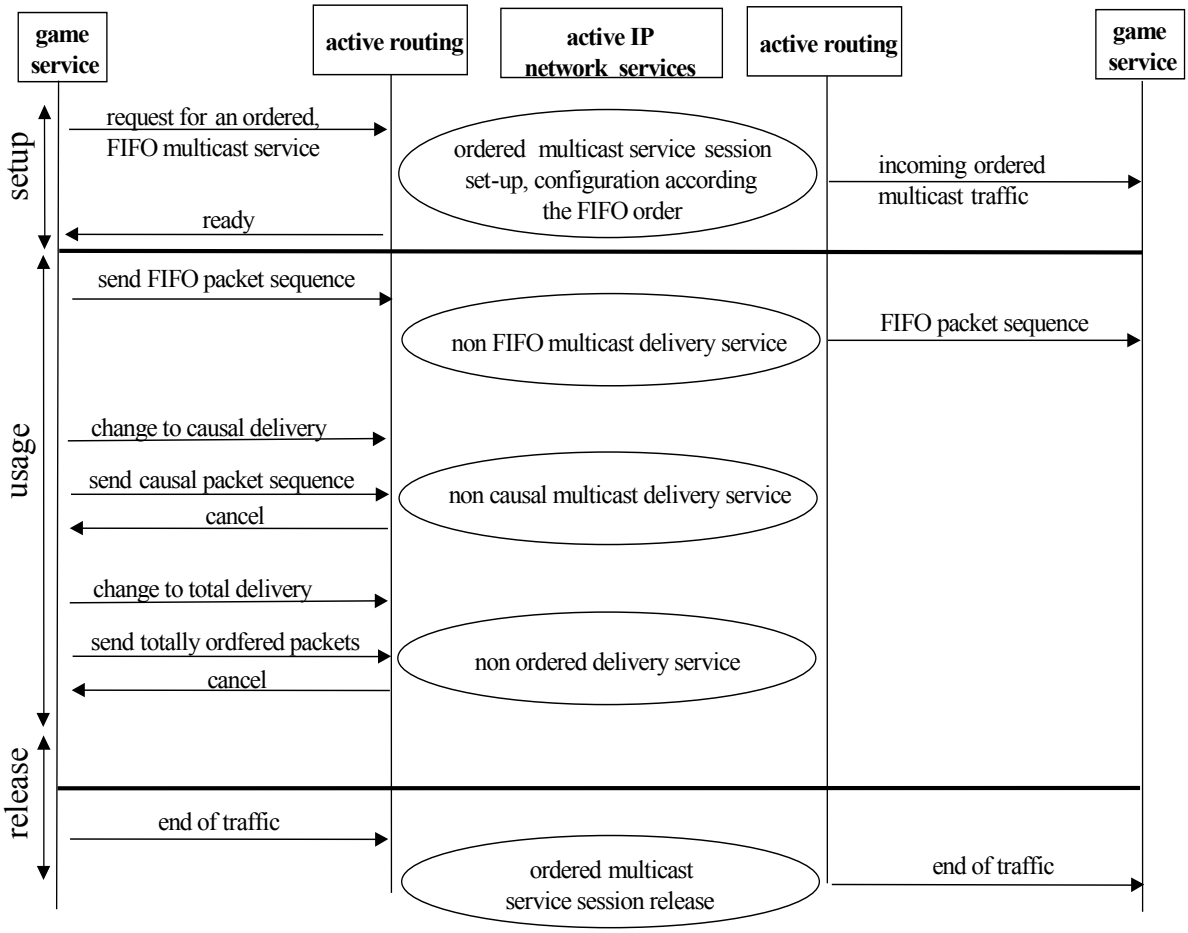


Figure 4-38 exceptions and alternative flows in variable ordered multicast delivery

4.3.4.3.8 Post-Conditions

When a multicast protocol with a flexible ordering of packets and messages is set-up and functioning, game network providers can deploy game platforms, environments and services with a wide range of requirements in terms of temporal relationships between events occurring in sub-systems. In particular, the maintenance of causal relationships between distributed events, along with real-time constraints, represent key network properties that will enable the implementation interactive, online networked games.

4.3.4.4 Use Case "multimedia multicasting in a game network"

4.3.4.4.1 Use Case Summary

This use case describes the use of an active IP media network to implement a game network that will support game platforms and services from one or many game platforms and game service providers. A game network consists of a set of geographically dispersed game servers with adequate processing, storage, network connections and protocols to support the execution of multiple game service platforms and applications. The game servers are to operate outside the active IP network and they are connected through a complete topology, i.e. each server has a direct connection with each of its homologues. When a game network provider wants to realise a game network, it establishes contacts with one or many active IP network operators and one or many processing and storage equipment providers to discuss commercial agreements and technical specifications. Technical specifications will deal with the needs of protocols and communication services, network service level agreements, processing power and storage resources, and the basic software to be supported by game servers and database systems. Commercial agreements concern typically the costs of permanent network connections with some service agreements, the availability of protocols with some technical properties, the costs of game servers with a given processing power and storage amount, etc. Some of the software and communication services may be developed by the suppliers, and others may be developed by the game network providers, according to the commercial and technical agreements. When the game servers (including storage, processing and communication software) and network accesses are available, game networks can be deployed according to a deployment plan previously established and the game network provider is ready to support game platforms and services from several clients.

4.3.4.4.2 Problem Statement

This use case specification is intended to give an overview of the network platforms and multicast services needed to realise multimedia communications between game platform and game service components running on game servers. We illustrate some important operations needed to realise a game network and the use of active multicast services to implement multimedia communications between game platforms and game service components that are distributed and/or replicated on game servers.

4.3.4.4.3 Actors Involved in the Use Case

This use case specification involves one or many media network providers, a game network provider, one or many game platform providers, as actors involved in active networking activities. It also involves actors supplying game servers and storage devices, basic software running on game servers, and possibly external actors that will develop software components required by game network providers. This second group of actors in not represented in this description for the sake of simplicity, since they are not directly concerned by the development of active network platforms and services.

4.3.4.4.4 Resources Assumed for the Use Case

The application assumes the existence of an active network for the communication of control and media information between game servers. It also assumes the possibility of third-party providers to build game servers with powerful processing and storage capability, that can be used to support multiple game platforms, game services and the interactions between thousands of end-systems operating within a geographical area.

4.3.4.4.5 Pre-Conditions

The application assumes the existence of a demand on online networked games, networked game software platforms, providers of game servers and media networks. Also it is assumed that commercial contracts exist between game network providers and the actors directly involved in the realisation and use of game networks.

4.3.4.4.6 Use Case Description (Main Flow)

The figure below shows an example scenario for the set-up, usage and release of a game network. The actors involved in this example are an active media IP network, a set of game servers and client game platforms running on these servers

At the beginning, active media IP network providers and game network providers must collaborate to provide the adequate protocols and install game servers. Then, game network providers and game platform providers must collaborate to provide the adequate protocols and install game platforms and applications. The usage phase shows a loop consisting of multicast communication requests and executions between game servers and the active media IP network.

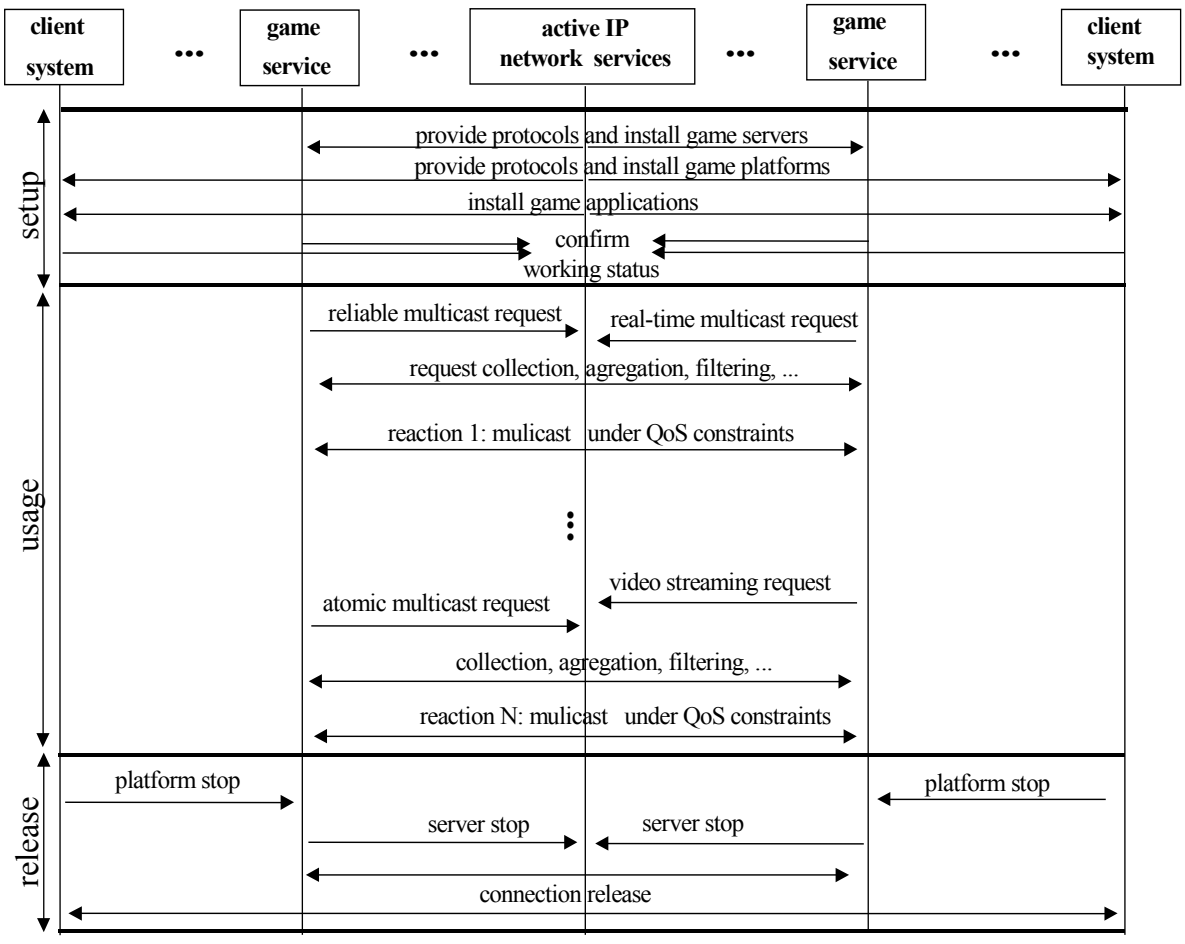


Figure 4-39 flow diagram for multimedia multicasting

4.3.4.4.7 Exceptions and Alternative Flows

The figure below shows an exceptional situation in which the active media IP network functions under limited network resources. In that case multicast communication requests may be cancelled if they require stringent QoS, or served with degraded QoS, etc.

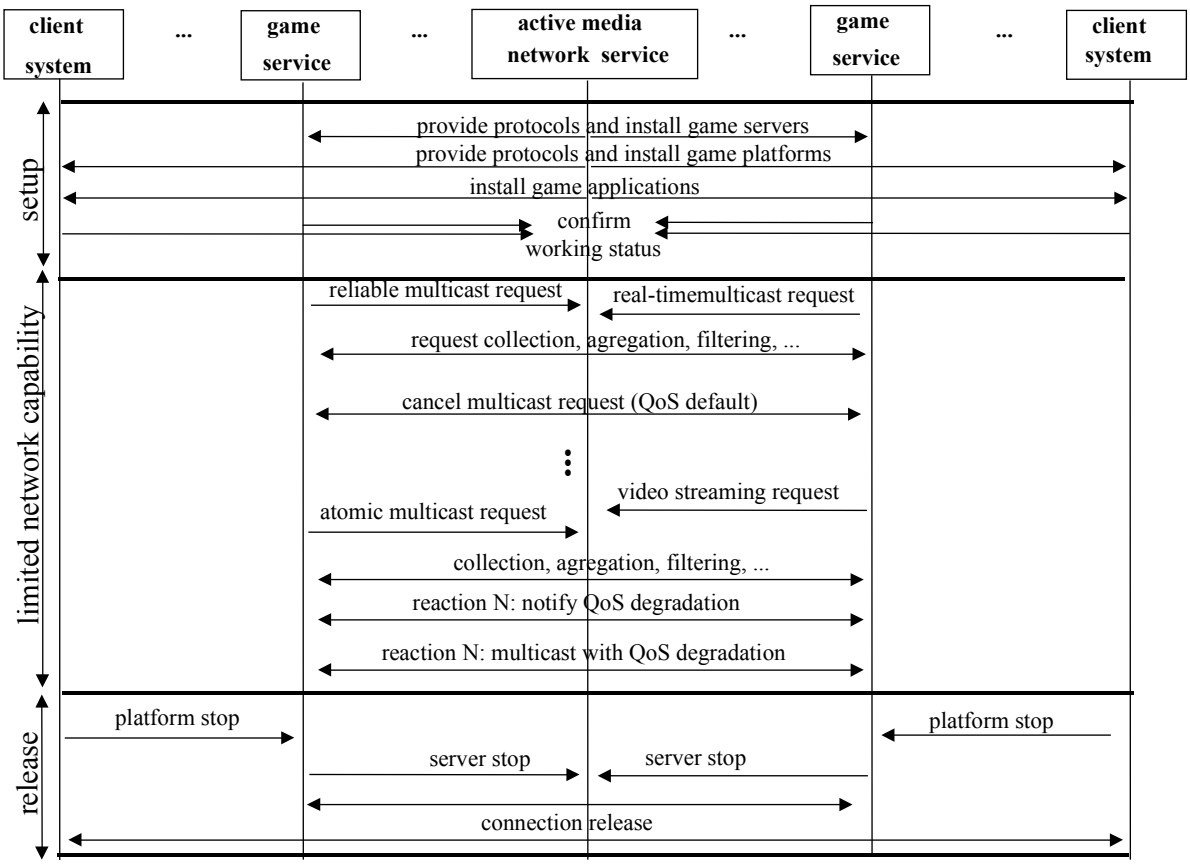


Figure 4-40 exceptions and alternative flows in multimedia multicasting

4.3.4.4.8 Post-Conditions

When a game network is set-up and functioning, computing and communication infrastructures are available to deploy several game platforms and applications with various communication requirements. The ability of the game network do satisfy actual communication service demands depend on the availability of resources, active network service protocols and the power of game servers, and the load of the game network in terms of game platforms, game applications and users.

4.3.4.5 Use Case "game application level multicasting"

4.3.4.5.1 Use Case Summary

This use case describes the use of a game network to implement game platforms and networked game environment, that will be accessed by end-users dispersed geographically. Typically, a networked game is designed as a set of replicated game environments, that are placed on game servers. A game environment is in charge of end-systems within a geographical area covered by its supporting game server, it simulates the same virtual environment as its homologues, it runs the same services and acts on replicas of objects populating the virtual environment. The game environments use a game platform software, which is also replicated on game servers, to ensure replication management functions, distributed control and co-ordination, and so on. Game environment (services) providers establish commercial agreements with end-users, media network providers and game platform providers, for the provision of example services that follow :

- Media network providers give a permanent access to an active IP network, that will be used for the multicast of audio/video streams in particular, and other game control and management communications that do not need to use game platform functions.
- Game platform providers give a permanent access to game control, co-ordination and management functions, game environment replication and consistency functions, and so on.
- End-users use game environments and services to "play" with their homologues .

This use case also describes how end-users can access and use a networked game and use it. Whence a game is available, end-users can subscribe and act on the game environment. The actions undertaken by end-users lead typically to modifications on the game environment and concurrent interactions with their homologues. An end-user is required to have an access to the active media IP network and the game network at the same time. Both accesses can be managed by a same game service provider for simplicity, or separately (one by a game service provider and the other by a media network access provider).

4.3.4.5.2 Problem Statement

This use case specification gives an overview of the network platform and communication services (especially multicasting) needed to establish and use a networked game. We focus on the relationships between game environments, game platforms and the game networks, and the type of communications that will take place between these service elements. The use case also gives an overview of a game environment and end-users accessing that environment. The second description focuses on the types of actions end-users can ask to the game environment and service software, and the multicast communications that could take place between game environments, on behalf to the actions performed by end-users.

4.3.4.5.3 Actors Involved in the Use Case

The use case involves game environment and service providers, game platform providers, game network providers, and many end-users (at-least two) that will interact through game environments. It also involves actors such as those developing game services and platform and active multicast protocols providers, but these actors are not directly concerned by the use case description.

4.3.4.5.4 Resources Assumed for the Use Case

All the resources necessary to an active networking infrastructure for online games over the Internet games are needed : these are mainly an active media IP network, a game network with sufficient computing and communication resources, game platforms for solving problems related to distribution, and the game environments running on servers. End-users also need an access network with sufficient bandwidth to carry stream communications from (or to) the active media IP network. It is assumed that an active media IP network exists and is accessible by game environments and service providers, along with a game network and game platform software running on game servers.

4.3.4.5.5 Pre-Conditions

The resources must be set-up. Also, commercial contracts exist between the principal actors, and especially between and-users and game service providers. Especially, we assume that there exist a demand on online networked games, and commercial contracts between game service/environment providers and end-user, game platform provider, game network providers.

4.3.4.5.6 Use Case Description (Main Flow)

The figure below shows an example scenario for the set-up, usage and release of a game session. The actors involved in this example are a game platform, and a set of couples of actors consisting of a game environment and a pool of users in a geographical area.

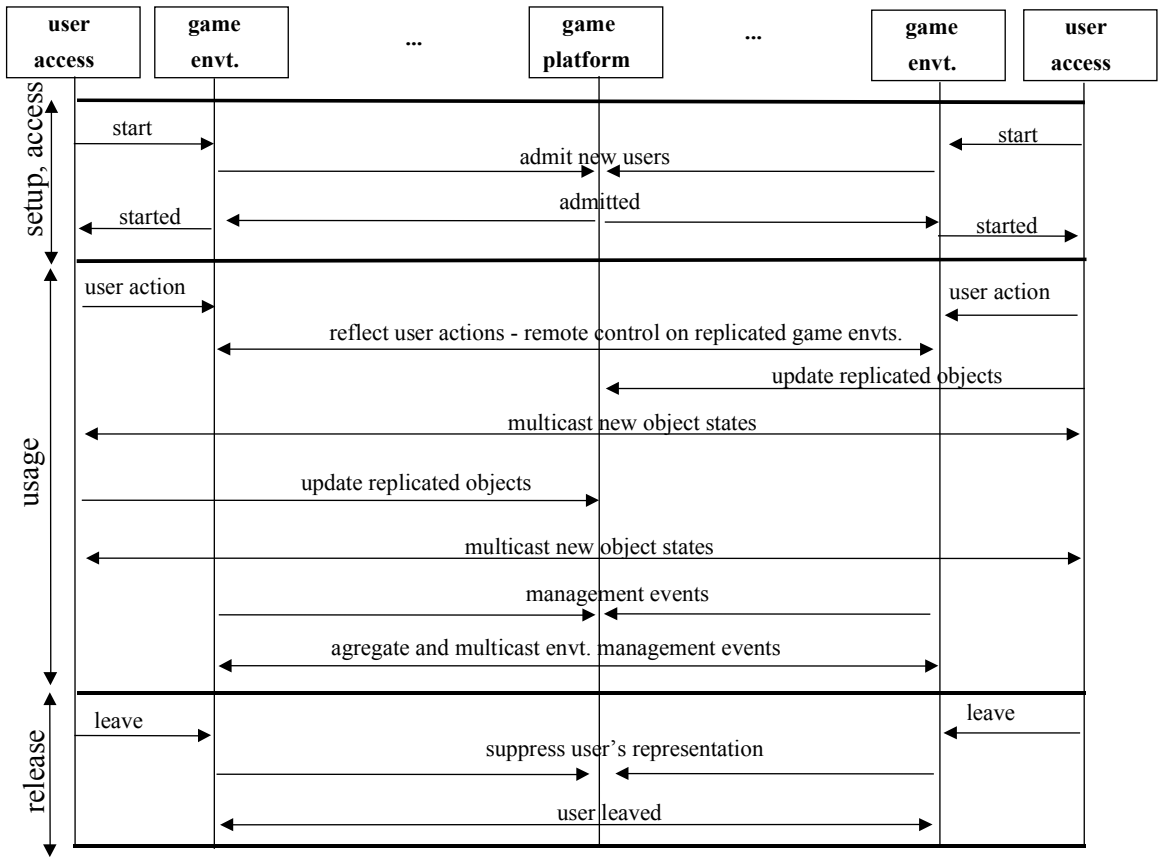


Figure 4-41 flow diagram for the access and use of a networked game

At the beginning, a game environment program is supposed to be running on each server, a long with one or many game service sessions. A user that wants to participate to game session sends an access request to its attached game environment, and the game environment performs and admission control procedure to access the user (access right, authentication, and so on). The admission of new users are multicast to homologue game environments, so that all the servers maintain the same view of the global game environment. Upon admission, users can start to act on the game environment by means of remote control on environment objects. It is the responsibility of game platforms and environments to reflect the effects of the actions of participants (e.g. by means of multicast events), and manage concurrency between several participants.

The figure below shows an example scenario for the set-up, usage and release of a game environment. The actors involved in this example are a game network, and a set of couples of actors consisting of a game platform and a game environment.

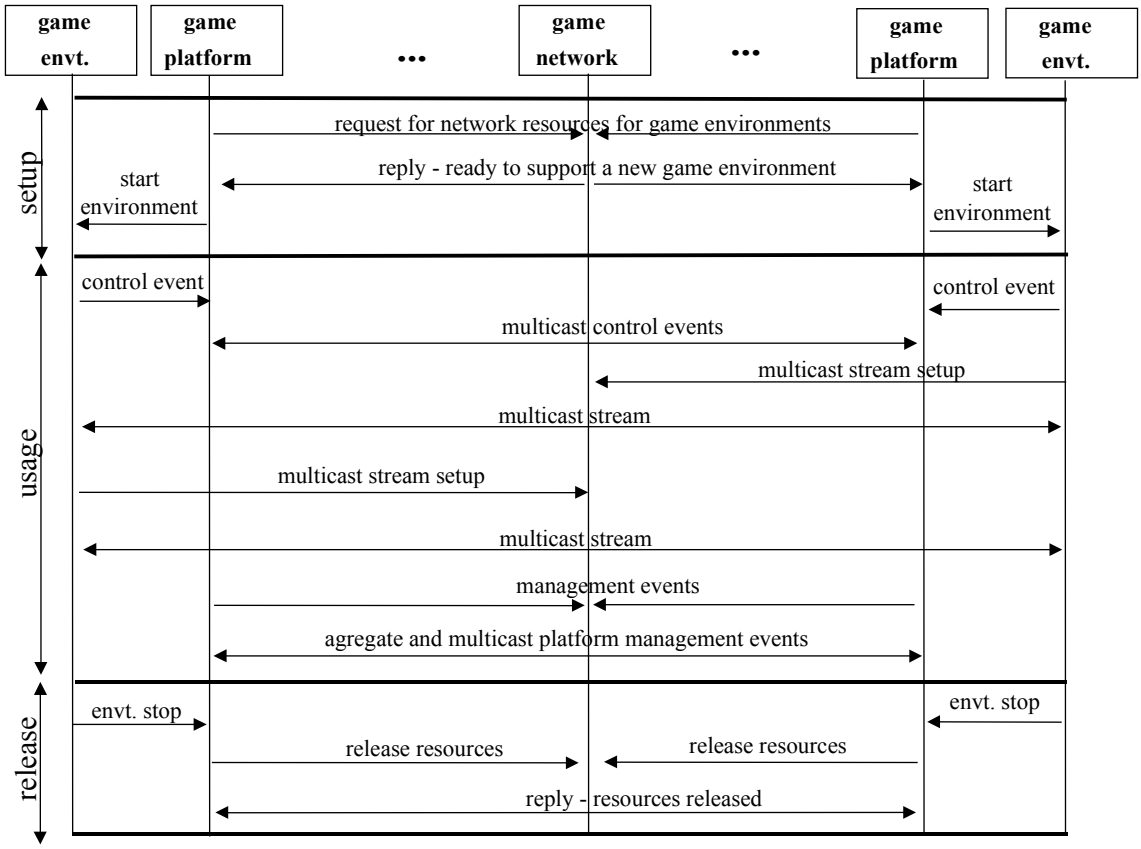


Figure 4-42 flow diagram for the set-up, usage and release of a game environment

At the beginning, the game network and game platforms must co-operate to ensure that resources are available and to start the game environments. Then, the usage phase concerns typically the multicast of game environment control events, stream packets, or management events. When a request is issued to stop a game environment, the game network resources are released accordingly.

4.3.4.5.7 Exceptions and Alternative Flows

The figure below shows an exceptional situation in which the game environment admitted too many end-users. In that case, interactions between end-users may be blocked from time to time, in order to provide an acceptable level of QoS to those already engaged in interactions.

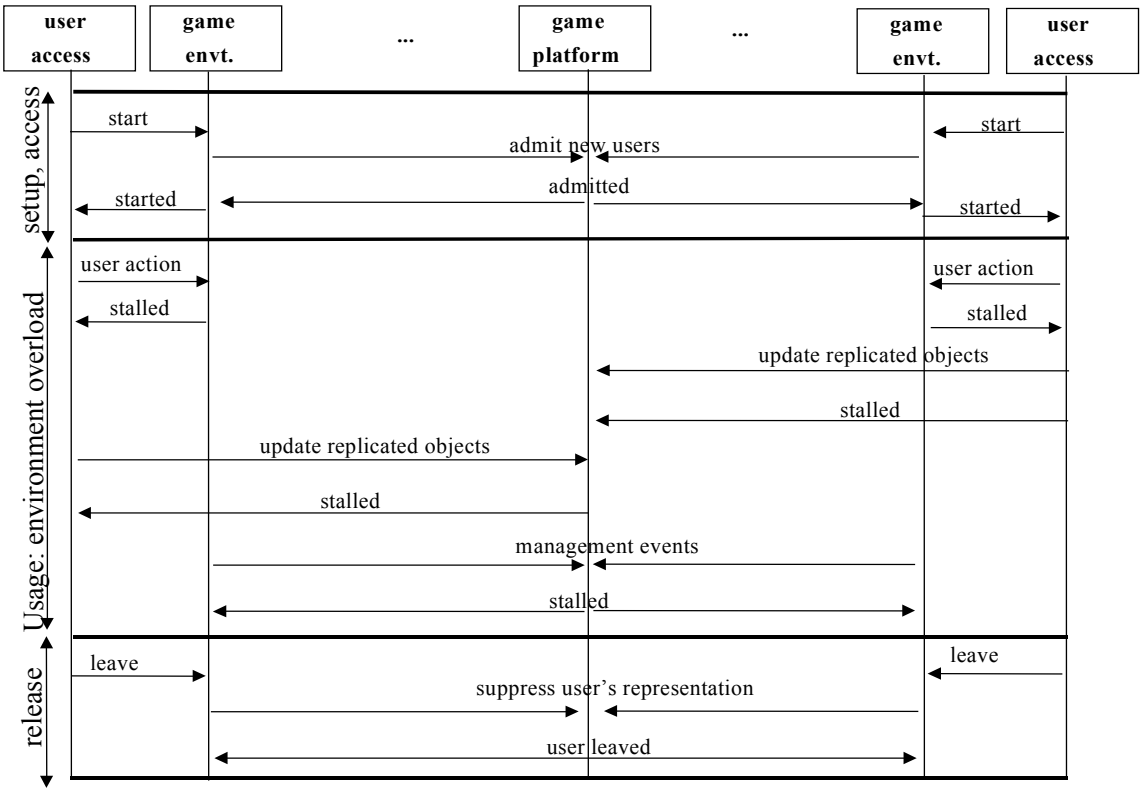


Figure 4-43 exceptions and alternative flows in the access and use of a networked game

The figure below shows an exceptional situation in which event services fail to handle requests properly. In that case, requests coming from game environments or platforms may be cancelled systematically.

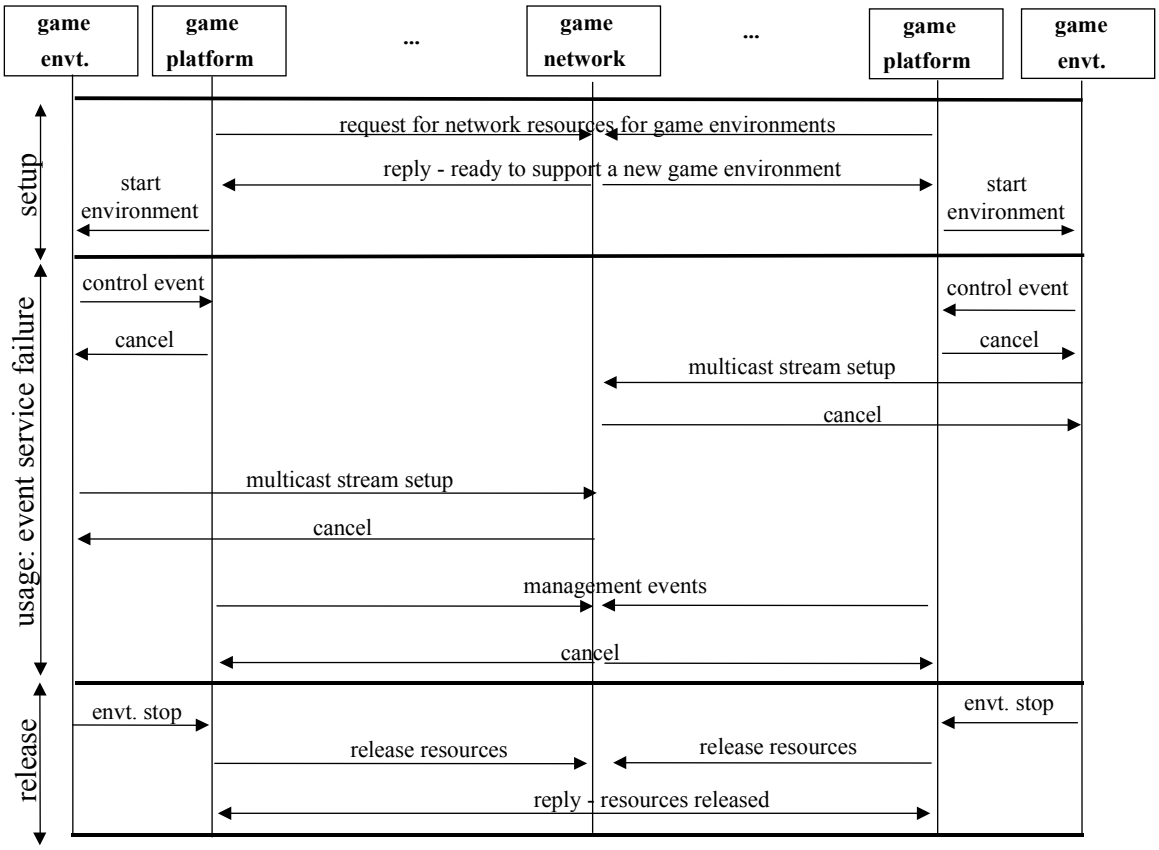


Figure 4-44 exceptions and alternative flows in the set-up, usage and release of game environments

4.3.4.5.8 Post-Conditions

A game environment is running, with services that are ready to handle interactions between end-systems, and reflect actions coming from any of the end-systems to replicated platforms and game environments. Also, a game service session is running, with users accessing the game service, interacting with each over and modifying the game environment, leaving temporally (or definitely) the game, and so-one.

4.3.5 Assessment of the Use Cases

4.3.5.1 The Case of State-of-Art Multicasting

4.3.5.1.1 Assessment of the Use Case "initial construction of the multicast tree"

In this use case we implement the primary infrastructure for the building of the Multicast service. In the Multicast we can create, modify and delete the multicast groups. The group is the fundamental module of the Multicast service.

4.3.5.1.2 Assessment of the Use Case "adding a member in a multicast group"

In this use case we add new users in the Multicast tree. This is a basic idea of the Multicast service as we can dynamically increase the number of the participants in the Multicast group.

4.3.5.1.3 Assessment of the Use Case " installing a multicast protocol on an active node"

This case implies the use of activeness in the Multicast Service. The Consumer has the ability to request new upgrade real time services in an efficient and capable way. Additionally this use case refers to the up layer of our architecture where SP interacts with Consumers in order to serve the QoS requirements.

4.3.5.1.4 Assessment of the Use Case "deleting a member from a multicast group"

In this use case we delete new users from the Multicast tree. This is a basic idea of the Multicast service as we can dynamically decrease the number of the participants in the Multicast group.

4.3.5.1.5 Assessment of the Use Case "deconstructing a multicast tree"

In this use case we can delete the consumers from the reserved nodes. In this way we release the nodes in order to serve other applications running in the Fain Enterprise Model.

4.3.5.1.6 Assessment of the Use Case "path reconfiguration"

This use case upgrades already Multicast services provided to Consumers. This innovation is inserted to delegate the management of the multicast protocol in to different participants inside the network.

4.3.5.2 The Case of Next Generation Multicasting and Game Applications

4.3.5.2.1 Assessment of the Use Case "multicasting with varying reliability"

This use case is essential for describing and illustrating the use of active networks to implement multicast services that can handle communications with variable, dynamic reliability requirements. It is essential as an example of new communication service features allowed by active networks, especially in relation of QoS in multimedia communications and reliable data communications.

4.3.5.2.2 Assessment of the Use Case "multicasting with varying real-time guarantees"

This use case is essential for describing and illustrating the use of active networks to implement multicast services that can handle communications with variable, dynamic real-time requirements. It is essential as an example of new communication service features allowed by active networks, especially in relation with QoS in multimedia communications.

4.3.5.2.3 Assessment of the Use Case "multicasting with dynamic ordering or packets"

This use case is essential for describing and illustrating the use of active networks to implement multicast services that can implement several ordering semantics in multicast communications. It is essential as an example of new communication service feature enabled by active networks, that can be used to satisfy temporal constraints that appear in several Internet services.

4.3.5.2.4 Assessment of the Use Case "multimedia multicasting in a game network"

This use case is essential for describing and illustrating the network infrastructure and multimedia multicast protocols needed for networked games. It is directly concerned with the use of active IP network services in this application domain, and the type of communications required. It is essential to develop and detail this use case. The detailed specification will imply collaborations with WP3 and WP4 activities related to dynamic protocol provisioning and the design and implementation of multicast service supports within active IP network nodes.

4.3.5.2.5 Assessment of the Use Case "game application level multicasting"

This use case is also essential for the understanding of platform and application level active multicast services needed by networked game applications. This use case specification is concerned with the impact of active IP networks on application level communications protocols : it illustrates to show how active IP networks can be used to solve some communication problems easily, that could be hardly solved by the classical networking concepts. Also, the use case is interesting to explain the use of networked game environments and services by several multiple online end-users, this point is not essential for active networking. The use case also illustrates the co-ordination problems involved by the replication of game environments and services, that can be used to derive the communication services needed. This aspect can facilitate the understanding of the interests of active networks, and thus it is nice-to-have, but not essential.

4.3.6 FAIN Reference Points Involved in the Application

4.3.6.1 FAIN Reference Points Involved in State-of-Art Multicasting

RP 1 : The SP contacts with the service SCP in order to download code for the implementation of a new service. In our architecture the SP needs the code to provide a reliable multicast transport.

RP 2 : There take place all the interactions between the Active Network Service Provider(ANSP) and the service provider (SP) in the FAIN enterprise model. ANSP provides the needed infrastructure (routing and the resources reservation) where the SP builds his services.

RP 3 : The ANSP contacts with NIP in order to request the resources for the provision of the Multicast services.

RP 4 : The Consumers (C) can contact with the SP and request the provision of Multicast Services. In our architecture the Consumer has the ability to build (through SP and ANSP) his own services in an efficient and capable way.

RP 6 : There take place the interactions between the ANSPs for the provision of Multicast services into intra-domain Consumers. This Co-operation facilitates the installation of the multicast protocol in to nodes of different domains.

4.3.6.2 The Case of Next Generation Multicasting and Game Applications

The reference points we consider are those of the initial FAIN active network enterprise model, defined in [Mei00], which contains a nearly stable version of the business roles and reference points.

4.3.6.2.1 FAIN Reference Points in Reliable, Real-Time or Ordered Multicasting

RP 1 : The reference point between the service component provider (SCP) and the service provider (SP) in the FAIN enterprise model applies to actors that will develop reliable, real-time or ordered multicast protocols and services, and active media IP network providers.

RP 2 : The reference point between the Active Network Service Provider(ANSP) and the service provider (SP) in the FAIN enterprise model applies to actors that will provide execution environments reliable, real-time or ordered multicast supports and active media IP network providers.

RP 4 : The reference point between the consumers (C) and the service provides (SP) in the FAIN enterprise model applies to game network providers, game platform and environment providers or end-users (as consumers) and IP network providers (as service providers) that provides protocols with reliable, real-time or ordered delivery.

RP 5 : The reference point between service providers (SP) in the FAIN enterprise model applies to IP network providers that provide reliable, real-time or ordered delivery for networked game environments.

RP 6 : The reference point between Active Network Service Providers(ANSP) in the FAIN enterprise model applies to actors involved in the provision of service execution environments with reliable, real-time or ordered multicast delivery constructs for active media IP network services.

4.3.6.2.2 FAIN Reference Points in the Use Case "multimedia multicasting in a game network"

RP 1 : The reference point between the service component provider (SCP) and the service provider (SP) in the FAIN enterprise model applies to actors that will develop network services for networked games and active media IP network providers.

RP 2 : The reference point between the Active Network Service Provider(ANSP) and the service provider (SP) in the FAIN enterprise model applies to actors that will provide execution environments for active media IP network services and active media IP network providers.

RP 4 : The reference point between the consumers (C) and the service provides (SP) in the FAIN enterprise model applies to game network providers, game platform and environment providers or end-users (as consumers) and active media IP network providers (as service providers).

RP 5 : The reference point between service providers (SP) in the FAIN enterprise model applies to active media IP network providers involved in the provision of a networked game service platform

RP 6 : The reference point between Active Network Service Providers(ANSP) in the FAIN enterprise model applies to actors involved in the provision of service execution environments for active media IP network services.

4.3.6.2.3 FAIN Reference Points in the Use Case "game application level multicasting"

RP 1 : The reference point between the service component provider (SCP) and the service provider (SP) in the FAIN enterprise model can be applied to actors that will develop networked game platforms and game platform providers, provided that the scope of RP1 can be extended to include interfaces between actors that developer game platforms, environments and services, ant those that deploy and operate game networks and services.

reference point between the Active Network Service Provider(ANSP) and the service provider (SP) in the FAIN enterprise model can be applied to game Active Network Service Providers and game platform service providers, provided that the implementation of RP2 is not limited to classical network level services.

RP 4 : The reference point between the consumers (C) and the service provides (SP) in the FAIN enterprise model can be applied to game environment providers or end-users (as consumers) and game platform providers (as service providers), provided that the implementation of RP4 is not limited to classical network and transport level services.

RP 5 : The reference point between service providers (SP) in the FAIN enterprise model may apply to game platform providers provided that the implementation of RP5 is not limited to classical network level services.

RP 6 : The reference point between Active Network Service Providers(ANSP) in the FAIN enterprise model may apply to game network services, provided that the implementation of RP6 is not limited to classical network services.

4.3.7 References

[Alm00] Kelvin C. Almeroth, "The Evolution of Multicast: From the MBone to Interdomain Multicast to Internet2 Deployment", EEE Network Special Issue on Multicasting, January/February 2000

[BN00] Ermolaos Zimboulakis and Yiannis Nikolakis, "Multicasting Use Cases: Mapping multicast use cases onto enterprise model ", FAIN internal document # WP2-NTU-003-R32-Int, NTUA FAIN contribution, 12 November 2000

[BP00] Ermolaos Zimboulakis and Odysseas Pyrovolakis, "Multicasting Use Cases: Initial thoughts on use cases, concerning the multicast procedure", FAIN internal document #WP2-NTU-003-R32-Int, NTUA FAIN contribution, 15 November 2000

[CT90] David D. Clark and David L. Tennenhouse, "Architectural considerations for a new generation of protocols", in Proceedings of ACM SIGCOMM '90, September 1990

[DGS99] F. Dang Tran, A. Gérodolle and J.-B. Stefani, "Towards A Distributed Virtual Continuum", in proceedings of the 1999 IEEE Virtual Reality Conference (VR '99), March 13-17, 1999, Houston, TX, USA

[DIS95] IEEE Standard for Distributed Interactive Simulation, Application Protocol - IEEE 1278.1-1995, IEEE Computer Society, 1995

[FJL97] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", IEEE/ACM Transactions on Networking, December 1997, Volume 5, Number 6, pp. 784-803

[Goy98] Jaun-Mariano de Goyeneche, "Multicast over TCP/IP HOWTO", http://notch.mathstat.muohio.edu/html/Multicast/Multicast-HOWTO.html#toc2

[Hou00a] Drissa Houatra, "QoS-constrained Event Communications in Distributed Virtual Environments", DOA'00, Antwerp, Belgium, 21-23 September, 2000

[Hou00b] Drissa Houatra, "Multicast Use Cases, version 1.0: Document analysing the use of active networks to implement multicast protocols for networked games", FAIN internal document #WP2-FT-001-I01-Int, FT FAIN contribution, 23 June 2000

[Hou00c] Drissa Houatra, "Multicast Use Cases, version 2.0: Application: Multicast Services for Networked Games ", FAIN internal document #WP2-FT-002-I01-Int, FT FAIN contribution, 06 October 2000

[Hou00d] Drissa Houatra, "Multicast Use Cases, version 3.0: Infrastructure and Multicast Services for Networked Games ", FAIN internal document #WP2-FT-004-I01-Int, FT FAIN contribution, 28 November 2000

[HT94] Vassos Hadzilacos and Sam Toueg, "Fault-Tolerant Broadcast and Related Problems", in Sape Mullender (Edt.), Distributed Systems - Second Edition, ACM Press 1994, ISBN 0-201-62427-3, pp. 97-145

[Lam78] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System", Communications of the ACM (CACM), Volume 21, Number 7, July 1978, 558-565

[Mei00] Jens Meinköhn, "Initial Active Networks Enterprise Model (Audit Version)", FAIN internal document #WP2-DT-006-I99-Int, NTUA FAIN contribution, 19 November 2000

[Mul94] Sape Mullender (Edt.), Distributed Systems - Second Edition, ACM Press 1994, ISBN 0-201-62427-3.

[Oos] Simon Oosthoek, "Survey of Multicast Support for IP over ATM for Implementation Purposes", http://www.huygens.org/people/oosthoek/thesis1/thesis1.html

[SM] Chuck Semeria & Tom Maufer, "Introduction to IP Multicast Routing", http://www.3com.com/nsc/501303s.html

5 REQUIREMENTS ANALYSIS

5.1 Generic Requirements for the FAIN Architecture

5.1.1 Service Architecture

The Service provision architectural shall be flexible to cope with various underlying connectivity networks. It shall make optimal use of functions provided by these networks.

- Support rapid definition and provision of services, which can be activated, programmed and controlled by the end-userrs and the Service Providers
- Support for service scalability: With the market opening up for communications, it is expected that one operator or service provider will serve an even greater number of users than before. At the same time, there might be markets for highly customised services targeted to small user groups. World-wide markets combined with the capability to customise products for small groups put strong requirements for scalability of all software systems. This necessitates that services should be created with scalability in mind; they should be made to scale well (i.e., be efficient enough to support as little as 10 to over a million users).
- Support multi-domains and multi-service operators and flexibility in inter-operability of services
- Support changes in the business telecommunications models due to growing completion in the Network and Service provision markets
- Support service delivery over various networks
- Support dynamic service provision
- Support for flexible and light-weight service management facilities which can be defined, activated and controlled by the end-userrs and the Service Providers
- Support multi-levels customisation and programmability of services. The service architecture provides a set of components, which can be specialised and enhanced. This capability will enable service providers to differentiate through their own service development
- Support the Open Service Resources APIs. In this way a new service can be provided to end-users that manages the network and service resources as programmable entities.

The user should have the capability to:

- Personalise their subscribed services so that the service is independent of the serving network and terminal in use.
- Personalise their subscribed services, expressing location and temporal preferences, according to different situations or needs, for example being at work, in the car or at home.
- Request new services/facilities to be provided or new value added services to be dynamically deployed.
- Subscribe dynamically to new services or suspend/cancel the subscription of a particular service.
- Access information related with tariffs and service costs, as well as log information about performed calls (globally and on a per service use basis).
- Define cost thresholds in a per service basis.
- Select how networks and their associated QoS are to be displayed for different types of calls and services.
- Define his preferences in case of terminal/network adaptation.

- Management by End-Customer: End-customers must be allowed to perform their own management should they so desire.
- Subscribing/unsubscribing End-Users: It must be possible to subscribe/unsubscribe End-Users if sufficient Management Rights are present.

5.1.2 Service Access Requirements

A set of requirements could be identified to make all these access functionalities, providing commonly used utilities, necessary for the network capabilities to be accessible, secure, resilient and manageable, available and independent of any particular type of service.

- There must be a mutual authentication between application and network. Before an application can use the capabilities from a network, a service agreement has to be established between the application and the network.
- If a service agreement already exists, it will be possible to modify it or start a new agreement superseding the existing.
- An application must be authorized to use one, more or all functionalities from the network; no further authorisation should be required as long as only the "allowed" functionalities are used. Authentication must precede authorisation.
- It will be possible to register new functionalities from the network. Registration must take place before authorized applications can find out which functionalities are available.
- An application will be able to identify the total set of functionalities that it can use, upon request. Before functionality can be discovered by an application it has to be registered.
- An application should be allowed to receive notifications of application related events that have occurred in the underlying network, e.g. indication that a new call is set-up or a message is received.
- An application should be allowed to enable or disable the reception of those events.

5.1.3 Service-to-network Adaptation/Management

The Service-to-network adaptation requirement, enables the user to access any service, regardless of the underlying network technology. Thus, in order to make the service accessible within different networks, the service must be adaptable to the different network capabilities.

- The network adaptation consists mostly of the necessary reservation information, which is downloaded across the network to the terminal and should be adapted to the network capabilities for every single service.
- The user should be able to select which networks to be used to make and receive communications from.
- The user should be able to select how and when to be informed if a change due to QoS reasons is necessary;
- The user should be able to accept or reject those changes, or eventually choose them, if more than one solution is available.

From the experiences of Network Operators and also from the work in several international organisations, some other requirements can be presented in relation to Service management.

- Flexibility of the subscription facilities (dissemination of conditions, easiness of subscription and rapidity of provision, changes support)
- Cost Control must be provided independently for each service, especially if network resources (or others) are shared.

- Effective control on the resources to detect service congestion and to prevent monopolisation from specific customers.
- Keeping track of the history of the customer will be an important aspect of usability: contents accessed, preferences of services, demand on resources, time and place of accesses, etc.
- Management of heterogeneity as multi-platform applications.
- The management service should be able to extend to participants in several geographical locations.
- The management service should be able to extend several Internet Service Providers.

5.1.4 IP-based Network Models

- Support of "Information Model" for IP network services: IP networking technologies will continue to evolve over the years. A good information model of IP connectivity gives Service Provider a service level view, which is to serve as a framework for the Service Provider to build IP network services and application services.
- "Better abstraction" for network/service layers, routers, gateways..: The principle of abstraction and information hiding, aims at providing an essential, core set of connectivity services, which are to be used and expanded for more application level services.
- IP connectivity service: IP connectivity service as it is presented to the Service Provider should support active IP as well as basic IP-based connectivity service classes as they are specified by IETF Integrated Services (IntServ) and Differentiated Services (DiffServ) workgroups.
- Provisioning and monitoring: Provisioning and monitoring for IP connectivity service need to be coordinated in such a way that the user of the service can make sure SLA observance by the Service Provider. The monitoring function, which is complete in the sense of guaranteeing SLA observance, can be offered only through cooperation of the Service Provider.
- Support of "Carrier-IP" features: Carrier-IP features require more rigorous treatment of FCAPS issues. Provisioning and monitoring issues in addition to a part of security issues shall be addressed.

5.1.5 Service Level Agreements

- The service level agreement (SLA) dictates the terms of service and payment for the subsequent service. The SLA is a result of negotiation between two business entities. The SLA is effective over a period, which is also a part of the terms of service dictated by the SLA. Quality of Service (QoS) may be one part of SLA relating to connectivity services.
- The format of the Service Level Agreement should allow on-line specification of service level parameters and automated negotiation of the SLA to allow dynamic establishment of service.
- The format of the SLA should allow automated processing and fulfilment of the service.
- A SLA or SLA negotiation process should contain mechanisms, such as electronic signature, certificate that will allow the SLA to become legally binding,
- Security mechanisms, such as encryption and authentication, should be applied to the SLA and SLA negotiation process in order to ensure that the SLA is only accessible to agreed parties.
- It should be possible to re-negotiate SLA parameters while maintaining an active service.
- A customer should only need to negotiate with a single SP to establish an end-to-end management service even when third parties are involved in providing the service (single point of contact).

5.1.6 Quality of Service

- End-user traffic should be transported according to the SLA negotiated QoS level.
- The end-user should receive alarms / trouble reports, on e.g. Service interrupts, Service degradation.
- The SP should receive, scheduled or on demand, service logs, on e.g. Traffic statistics, Capacity levels, Resource usage, Periods of service interrupt, Periods of service degradation.
- The end-user should have access to information that will allow them to assess the QoS.

5.1.6.1 Static Functionality

- QoS parameters negotiation: The SLA negotiation process should allow negotiation of QoS parameters e.g. bandwidth, delay and jitter bounds, loss packet percentage, and service availability probability, should specify the source and destination prefixes IP addresses and ports and the start and end dates and times. This negotiation process is bilateral.
- Other parameters negotiation: The SLA negotiation process should allow negotiation of service information between ANSPs, e.g. customer care information, service code identification, recovery behaviour, Authorisation, Authentication and Accounting (AAA) policies, out-of-profile traffic handling. This negotiation process is bilateral.
- QoS metrics: The SLA contract should define the metrics for assessing the agreed QoS.
- Standard Based Interfaces: As will be stated later one of the requirements of the system is that it be as heterogeneous as possible. One way of achieving this is, where possible, to use established or emerging industry standards that will allow the finished system to integrate more readily with other third party products.
- Policy Vocabulary Mappings: The QoS Assurance system must process the SLAs it is asked to support into a coherent system policy that can then be distributed to the various elements that make up a service. However the form of the produced policy will depend heavily on what the underlying systems can support. It is necessary, therefore, for mappings to be stored which specify how the policy produced by the system may be translated into other formats.

5.1.6.2 Dynamic Functionality

- SLA Admission: Admission is an important part of any QoS usage stage as, at the very least, it is necessary to check that there are enough resources available to provide the QoS level being requested. However it is possible that admission can also be influenced by some policy concerns although, in the context of the SP, it is thought that most such policy matters will be dealt with during SLA negotiation.
- Policy Construction: The QoS usage system can, and most frequently will, be asked to support more that one SLA at a time. It will, therefore, be necessary for the system to combine the various requirements of the individual SLAs into a single coherent system policy. It will be during policy construction that decisions will be made regarding which resources will be assigned to provide support for each SLA. It is foreseen that this process will initially work to produce the policy in a generic format, which can then be translated, if necessary, into a format that can be understood by the elements/system that must enforce the policy.
- SLA activation: Activation of a SLA may be independent of the service usage itself. Conditions on the usage (e.g. start and end dates) could be part of the SLA.
- A SLA activation process should condition the use of a service. Activation request should specify at least QoS parameters, source and destination prefixes, authentication credentials, and start and end dates and times of use.

- SLA activation failure: The SLA activation process should explain its reasons to the requester in the case that a request can not succeed. It might also propose an altered service to replace the requested one.
- Negotiated QoS fulfilment: End-user traffic should be policed and transported according to the SLA negotiated QoS level.
- Policies monitoring: Policies should be monitored and stored.
- Standards compliance: Policing and policy storage may use IETF, DMTF standards.
- Measurement operations: Measurement (of e.g. delay, jitter, and loss) operations ensure that the network is meeting the required QoS, and report to the information service level.
- End-user measurement monitoring: The end-users should have access to tools that will allow them to assess the end-to-end QoS.
- End-user credentials: Security requirements are needed with regard to stealing of bandwidth, confidentiality of usage, content of information, etc. all-over the path of the communication, in particular for outsourcing and multi-domain QoS provisioning.
- Scalability: The QoS model should be scalable. In particular it should handle aggregate traffics rather than individual ones.
- QoS Monitoring: An import part of any QoS system is the ability to verify that the requested QoS is actually being provided. To this end QoS monitoring mechanisms will periodically collect usage statistics from various service elements are necessary. The statistics produced by such mechanisms not only allow QoS levels to be identified but can also allow warnings or notifications (see Notifications requirement below) to be sent out when certain situations occur.
- QoS Management Policy: During both the process of policy creation and resource adaptation certain decisions will need to be made about which underlying resources to use and how heavily to load these resources. It is therefore necessary to provide a management interface which will allow the system manage to specify the policies about which resources to give preference to. For example such policies might be created on the basis of cost, giving preference to the cheaper resources, or security, giving preference to elements on secure networks.

5.1.7 Charging/Billing

- Service Provider should provide the customers with flexible choices in receiving and paying bills. The main idea behind making billing process flexible is to cut costs through a greater control over service usage.
- Hot billing or real-time billing: Customer may want to receive bills within a few minutes of the end of service usage.
- Bills on Demand: Customers may want to receive bills at any time after the end of service usage.
- Regular and timely billing: Customers may want to receive bills at a regular interval (e.g., quarterly, monthly, weekly, etc).
- Service-sensitive billing The key issue here is the distribution of total cost among various departments that a business may have.
- The end-user should receive, scheduled or on demand, service logs, on e.g. - accumulated cost.

5.1.8 Security

- End-user traffic should be protected according to the SLA negotiated security level.

- The end-user should receive, scheduled or on demand, service logs, on e.g. security violations, violation attempts.
- The SP is responsible for ensuring confidence of information received from end-customer.
- End-customers will receive information in case of abnormal conditions.
- Authentication of End-Users: A set of End-Users that can be authenticated must be derived from the service subscription.
- Authentication of SP: The SP must allow End-Users to authenticate the SP, possibly by maintaining a certificate signed by a Certificate Authority (CA).
- Service Specific Operational Security Parameters: Optionally a set of operational security parameters for authentication, confidentiality and integrity (key-exchange protocol, encryption algorithms, key-lengths, etc.) specific to a service instantiation can be compiled from the high level SLA contents.
- It must be possible to dynamically and on-line access and change the: Business Interfaces for feedback, Service Specific Operational Security Parameters, Service Specific Security Policies, Service Management Rights, Service Access Point Management Rights
- Only authenticated users with specified privileges must perform the changes and the communication must be confidential.

5.1.9 Active Node/Network Control

- AN concept would allow virtual nodes to be created on some set of nodes in the network to form a virtual network. Each virtual network (the virtual nodes that make up the network)can be controlled by an independent Control Plane that should not be aware that it is controlling virtual nodes rather than physical ones. Similarly, each virtual network can be managed by a separate network manager that does not need to be aware of the fact that it is managing a virtual network and not a physical one.
- Standard APIs should be provided for Service Component Providers, so that can develop plug-ins and service components.
- Need to support the creation of multiple Execution Environments on a node, that are completely isolated from each other.
- The Network Infrastructure Provider should provide an open network infrastructure, in a way that networking resources can be divided virtually between different Active Network Service Providers
- Support for rapid service deployment – this is a concern for both active node design and the tools used for the design of Service Components. “Plug-and-play”-like functionality (to create services from service components, and to install these services on the active nodes) is needed to provide dynamic, on-the-fly service creation and deployment
- Management on the super-ordinate level shall allow the creation, modification, and deletion of virtual nodes.
- Management on the super-ordinate level shall partition and virtualise the management information and functions presented to the sub-ordinate management function corresponding to each virtual node.
- Allow competitive procurement of modular and reusable subsystems of active nodes. These sub-systems, under the domain of different controllers or control processes, shall be able to be combined in such a way that they interact with each other only in a predictable and controlled manner.
- Employ standard interfaces and protocols wherever feasible.

- Separate control software (e.g., signalling) from the transport resources (e.g., forwarding functions) using standard interfaces.
- Facilitate vendor, carrier, or third party development of application software.
- Reduce time to market for new technologies and services.
- Reduce operational complexity by providing standardised modular systems.
- Support partitioning of a common network into logically separate virtual networks.
- Support a multiplicity of services and applications each controlled via its own signalling protocols.
- Enable service and application implementations to be independent of changing transport types and technologies.
- Enable placement of service, application and transport/media interworking at the network edge.
- Scalable across a broad range of port counts, aggregate bandwidth, and transaction rates.
- Support a broad range of connection types including reserved bandwidth and a range of QoS capabilities.
- Support efficiency and flexibility of configuration including remote placement of system components.
- Separate the control, routing, adaptation and management aspects
- It shall be possible to combine virtual nodes in such a way that they interact with each other only in a predictable and controlled manner. The sharing of resources between virtual nodes may be either deterministic or statistical. The first application of partitioning is expected to provide multiple instances of services to separate groups of users. These user groups would not expect interference from each others ' activities.. At the same time, the users of each partition are interested in efficient utilization of the resources allocated to their Virtual Switch. This implies that the principle of resource sharing between partitions should be deterministic**.**
- More efficient utilization may be achieved if virtual nodes statistically share pooled resources among several, otherwise non-cooperating control logic instances
- It shall be possible to control nodes by several independent controllers or control processes of different types as well as by several independent controllers or control processes of the same type.
- It shall be possible to control the resources of an active node by deterministic partitioning.
- It may be possible to control the resources of an active node by statistical partitioning.
- It shall be possible to operate active controllers or control processes by one organisation as well as by several independent organisations.
- It shall be possible to implement a physical network of active nodes as one logical network or a set of logical networks realised via partitioning of the same physical resources.
- It shall be possible to perform processing of data streams under the domain of multiple controllers or control processes in such a way that deterministic partitioning of the involved available processing resources is maintained between different controllers or control processes.
- It shall be possible to perform processing of signalling/routing information under the domain of multiple controllers or control processes in such a way that deterministic partitioning of the involved available processing resources is maintained between different controllers or control processes

- It shall be possible to hierarchically structure resources implementing queuing and scheduling so that several Service categories can be supported under the control of each controller or control process.
- A flow rate control (e.g., policing) or queue management shall be implemented to ensure that resources under the domain of each controller or control process stay within the negotiated limits.
- It shall be possible to perform processing of service session signalling under the domain of multiple controllers or control processes in such a way that deterministic partitioning of the involved available processing resources is maintained between different controllers or control processes.
- It shall be possible to hierarchically structure to perform the replication of data in such a way that deterministic partitioning of the available involved processing resources is maintained between different controllers or control processes.
- Partitioning of resources within an active node requires at least two different types of FCAPS management logic. The two types differ in terms of scope: One has a global view, i.e., covering all of the active node, whereas, the other has a subordinate view, i.e., it sees and is able to manage only an a priori allocated partition of the node, a "Virtual Node ".
- It shall be possible to perform execution of multiple instances of FCAPS management logic in such a way that deterministic partitioning of the involved available resources is maintained between different management logic instances.
- It shall be possible to group the management information of Virtual Nodes, so that the group can be managed as one entity with one common management view (e.g., it shall be possible to add the ownership of one more Virtual Nodes to an existing group).
- The Node Control Interface shall provide a standard mechanism for inclusion of vendor-proprietary extensions.
- The Node Control Interface shall allow multiple active Node Control Interface masters to issue requests simultaneously to one Node Control Interface slave. These masters can run on physically separate machines and can connect to the slaves over physically separate links.
- The Node Control Interface should not restrict the number of connections concurrently active on a node, the number of virtual ports on a node, the physical interface types supported by the node, or the throughput supported by each virtual port.
- The Node Control Interface shall not restrict the node hardware redundancy scheme. Furthermore it must provide mechanisms for detecting loss of synchronisation of state and re-synchronising state without disrupting user connections or the ability to create new connections from/to virtual ports whose state is still in sync.
- Connectivity Providers shall ensure fairness in access of the network node is a situation, where all incoming packets and all outgoing packets are ensured a path to get in and go out from the network node. In a malfunctioning situation packets from a certain user can be prevented from reaching a specific network node due to (for example) intervention of a malicious code from another user.
- Connectivity Providers shall ensure fairness in using the resources of the network node is a situation, where all incoming packets are ensured a time space for the 'computation' part of the process. A policy, identified by the operator, or collectively by the end-customer and the operator, identifies the portion of the time space (measured in CPU cycles) and the extent of access to network node resources (files, memory, I/O, etc).

5.1.10 Generic Framework Requirements

The most comprehensive set of management building block requirements currently available is being worked on in the TM Forum's Application Component Team (ACT) [TMF2]. This is based on the requirements established by Telcordia (formerly Bellcore) in their extensive OSCA/INA analysis. This has already seen some application in the use of Ina principle in the work of TINA-C.

These requirement identify a "building block" as an abstract notion of a distributed computing entity, which aids discussion of the deployability and interoperability aspects of server-weight software with multiple interfaces. More simply, it describes a building block is a deployable unit of interoperating software.

5.2 Operators' Expectations & requirements

5.2.1 Influence of the needs to speed Service Deployment and Service Customisation

5.2.1.1 Network Infrastructure Providers

NIPs are required to offer sufficient managed resources on top of which service execution environment runs. With this environment the following features will be supported

- execution environment can be adaptable to new service
- node resources can be allocated/deleted flexibly when new service assigned or terminated.
- select vendor independent network products to support integrated service

These features will be implemented in RP3 and RP7.

5.2.1.2 Active Network Service Providers

ANSPs are required to develop the following features to speed service deployment and service customisations with respect to market needs;

- flexible selection of different service developers for implementing and deploying new service/features to compose the whole network service.
- make modification and the update of services possible at run-time.
- plug-in active properties.

These requirements will be implemented in RP4.

5.2.1.3 Service Component Providers

SCMs are required to provide service components for active application with the following features to speed service deployment and service customisation.

- can be generic service (e.g. multicast)
- make SP possible to ease enhance with reliability, security, real-time, etc..

These features will be implemented in RP1.

5.2.1.4 Service Providers

SPs are required to support the following features to speed service deployment and service customisations,

- chose proper service components which are provided by multiple SCMs and compose new services without many modification to satisfy consumers requests
- deploy and integrate services using proper network service by multiple ANSPs
- negotiate with other SPs to support specified end-to-end services
- offer personalised service to each consumers.
- collect consumers's requirements and satisfy them.

These requirements will be implemented in RP1, RP4 and RP5.

5.2.1.5 Consumers

Consumers require to modify existing services, add new service and guaranteed services. When a consumer starts using a service, this role will make service agreement with SPs. These will be implemented in RP4.

5.2.1.6 Active Middleware Manufacturers

AMMs provide platform with which ANSP can run active services and adapt modification of service behaviour. The platform can be upgraded with their latest network software release.

5.2.1.7 Hardware Manufacturers

HMs are required to realise the following features to speed service deployment and service customisation,

- make their network element products flexible to accept modification or upgrade by active network features.
- reduce lengthy standardisation-phase and technology diffusion phase before being operational.

5.2.2 Influences of the needs to Leveraged Network and Service Management

5.2.2.1 Network Infrastructure Providers

NIPs are required to support the following features for influences to the needs to leverage network and service management,

- provide network infrastructures for multiple third-party SPs. with network management by ANSP.

5.2.2.2 Active Network Service Providers

ANSPs are required to support the following features for influences to the needs to leverage network and service management.

- allow SPs to use network infrastructure under service provider-specific circumstances regarding to (e.g. accounting and billing) in order to comply with the EU ONP.
- delegate full management responsibility to multiple third-party SPs
- avoid the management overhead(e.g. complex and costly overhead of operation and maintenance) and reduce management cost substantially not to push SPs
- use standardised protocol/interface for full network management(FCAPS)

These requirements will be specified in RP2.

5.2.2.3 Service Providers

SPs are required to support the following features for influences to the needs to leverage network and service management,

- select and use network service in multiple network infrastructure

5.2.2.4 Hardware Manufacturers, Consumers, Service Components Manufacturers and Active Middleware Manufacturers

These roles do not act directly to leverage network and service management. Consumers subscribe network service via SPs, but may not care of which NIP support their services.

5.2.3 Influence of the needs to decrease the dependence of Vendor

5.2.3.1 Network Infrastructure Providers

NIPs are required to realise the following features to decrease the dependence of vendor,

- provide basically IP connectivity among heterogeneous HM's network products
- provide open/standardised interface for ANSPs
- available to dynamically introduce the required functionality at network nodes by means of injecting executable code into the node.

These requirements and mechanisms will be specified in RP3.

5.2.3.2 Active Network Service Providers

ANSPs are required to support the following features to decrease the dependence of vendor,

- support management service, which may be deployed by AMM, on active nodes using standardised interface.
- select different manufactures of switches, routers according to market-driven requirements. (e.g. price, availability, functionality).
- responsible for injected active codes.

5.2.3.3 Service Providers

SPs are required to

- integrate services or provide solutions using flexible functionality, which is not prescribed.
- map consumers service requirement into functionality.

These requirements and mechanisms will be specified in RP4.

5.2.3.4 Hardware Manufacturers

HMs are required to support standardised interface on its network hardware, with which NIP will realise IP connectivity requested by heterogeneous services.

5.2.3.5 Service Component Providers, Retailers/Broker, Consumers, Active Middleware Manufacturers

These roles do not act directly to decrease vendor dependency. AMMs may supply platform with execution environment for ANSPs to realise network management.

Influence of networks and service integration and information networking

5.2.3.6 Network Infrastructure Providers

Require to develop network infrastructure with the following properties, concerning the transport network technology, processing power and memory at network nodes :

- transport network technology that can support a wide range of service classes, and that has sufficient bandwidth to support several simultaneous broadband communications.
- network nodes with general processing and powerful processors, to support the execution of arbitrary routing or switching programs, arbitrary processing. Processors dedicated to specific routing or switching programs in particular should be avoided in order to preserve the possibility to support the execution of arbitrary transport services.
- Network nodes with large amount of memory (both for processing and storage). Storage memory in particular is needed to store and use various network service information (e.g. routing tables) on demand of client services.

In addition, network nodes should ease the execution of migrating programs (from one node to another) or remote execution of programs for interactions between network infrastructure providers, or those between network service providers and network infrastructure providers. This requirement will be helpful, especially for the implementation of operations at interfaces defined by RP7 and RP3.

5.2.3.7 Active Network Service Providers

Require to provide multiple service execution environments and generic network services. The service execution environments and generic services must have the following properties :

- Can be adapted to the needs of various network services (flexibility)
- Support the interpretation of active packets developed using various programming languages.
- Support the execution of arbitrary active packets
- Offer means to develop integrated network and information services, i.e., services that are able to combine information and treatments coming from network operation and applications in the same framework.
- Supports the execution and the isolation of network services provided by several service providers and to emulate the execution of "foreign" network services (those that normally run on other platforms)
- Portability to different network infrastructures
- Interactions between network service platforms

Several of the requirements above are of primary importance to implement interfaces that will be specified at RP3, RP6 and RP2.

5.2.3.8 Service Component Providers

Require to develop active network services specified by multiple service providers, according typical generic criteria that follow :

- Use of distributed object platforms and service frameworks.
- Supports mechanisms for remote access and execution of services.
- Supports for remote control, management and co-ordination of network service components

- Support for the concurrent manipulation of network service components, both from homologue network services and information services.
- Supports for the control of network resources, their allocation and the co-ordination of their use.
- Possibility for developed network services to be run on different service execution environments, that belongs to different service provider domains.
- Support for the execution of service components developed by other service component providers, and accessible through brokers and retailers.

5.2.3.9 Service Providers

Require to compose network service components developed by various service component provider, in order to provide complete, operable, end-to-end networks services. Following are some of the properties service providers must realise in order to facilitate information networks and service integration :

- Flexibility mechanisms that allow other service providers and information services to override network service components with new arbitrary network services, provided that some minimal consistency requirements are met.
- Mechanisms for the connection of two or more network service platforms, in order to allow the control of communications that span over several service provider domains than a single actor (a network or an information service provider).
- Interoperability between several network service platforms and their supporting systems (e.g. run-time object platforms).
- Mechanisms for accessing, distributed control, co-ordination and management of network resources from external network service components and information services.

5.2.3.10 Consumers

A consumers is any actor with systems that requests active network services. The role of consumers to facilitate information networks and service integration is very complex. We shall relate a very limited subset of the requirements, that comes from a non detailed analysis :

- Consumers should agree on a set of (prescriptive) active network services currently used in order direct network service providers in the type of services they must implement. These service agreements can be used to derive minimal service interfaces to be implemented by RP4. These network service interfaces can then be adapted by consumers, according to the type of information or communication services they would like to implement.
- Consumers should form special interest groups, par application domain, so as to specify information service frameworks and active network service requirements for their application domain. The interest groups must involve actors concerned by application domain, especially active network service providers. For example, consumer such as next generation telecommunication service providers (e.g. next generation IN service control and management platforms) could form a specification group in order to define and standardise a new service framework based on active networks).
- Consumers can also act as providers of generic distributed information processing software providers, that will hide the use of communications and distributed machines to solve information processing problem. Individual consumers can request active network service interfaces and mechanisms according to the technologies they would like to implement.

5.2.3.11 Hardware Manufacturers, Active Middleware Manufacturers, Retailers and Brokers

These actors are not directly concerned by active network technology, by their activity could be influenced by the development of active networks. For instance, hardware providers could have to develop new generations of network processors or transmission devices that better take into account the needs of active networks. Active middleware providers will be requested to develop middleware systems that satisfy requirements to execute active packets. Brokers and retailers will be requested to offer assess to and active network service components or providers. In the later case, active networks do not necessarily influence their technical activity, they could influence their market opportunities.

5.2.4 Influence of the Needs to Diversify Services and Business Opportunities

5.2.4.1 Network Infrastructure Providers

In addition to requirements from network and service integration and information networking, network infrastructure providers are required to develop mechanisms for :

- the integration of several hardware components, from different hardware vendors,
- the modification of hardware components without important modifications on their network infrastructure.

These requirements will allow network infrastructure providers to decrease their dependence to hardware manufacturers

5.2.4.2 Active Network Service Providers

In addition to requirements from network and service integration and information networking, network service providers are required to develop mechanisms for :

- the integration and use of several network infrastructures,
- the integration and use of different active middleware software,
- the replacement of network infrastructure supports and active middleware software without an important change on their active network service execution environments.

5.2.4.3 Service Component Providers

In addition to requirements from network and service integration and information networking, service components providers are required to develop a wide range of technical competence in network service technologies, and the commercial relations with their suppliers (service providers).

5.2.4.4 Service Providers

In addition to requirements from network and service integration and information networking, service providers are required in particular to :

- Diversify the technologies used and the classes of network protocols and services they provide.
- Their platforms must be able to run programs developed by several service component provides, manage requests coming from several and different type of consumer platforms, and run their network services on platforms belonging to different network service providers.
- Be able to replace change their supplier or network service provider or change their usual clients without an important impact on their activity.

5.2.4.5 Consumers

Consumer must also be able to :

- Use services from several active network service provider.
- Interact with their homologues, to use their services (or offer services to them) even if they do not operate in the same application domain.
- Use several network services with different service qualities in the same application.
- Change their clients or suppliers without an important impact on their service provision.

5.3 Requirements from applications

5.3.1 Requirements Description for the Reference Points

5.3.1.1 RP1 SCP – SP

5.3.1.1.1 Requirements from Multicasting Application

5.3.1.1.1.1 General Description

In this RP the interactions are between the SCP and the SP. We referred to this point in the following use cases: “Installing a multicast protocol on an active node”(explicitly) and the use case “Path reconfiguration”(implicitly though the previous use case).

In the first use case the SP has three responsibilities: Firstly, he has to provide a multicast algorithm, (composes appropriate service components to develop the Multicast Service), in order to achieve a reliable multicast procedure. Secondly, once a consumer has granted that he is authorised to participate in a multicast procedure, SP must contact the SCP in order to ask for the appropriate multicast protocols. Finally, SP has to install these protocols (inject the active service components, to the active nodes).

In the second use case, the SP has first to uninstall the protocols from the active nodes that are about to be removed from the multicast tree and then to install the multicast protocol in the active nodes which were added to the multicast tree, after the reconfiguration process.

5.3.1.1.1.2 Requirements

The requirements for the system that are derived from this RP are the following:

An interface should be provisioned in order for the two different roles (SP and SCP) to communicate. Under this interface SP requests service modules according to service description and SCP provides them including specifications about the requirements of the service module. In this way SCP informs SP about the ‘resources’ needed (VE/EE) from the service module to execute properly (etc: MPEG compression Mobile Agent needs IKV grasshoppers platform, multicast tree table object needs Corba EE).

SCP must organise the service modules into categories according to the functionality provided by the service component (service descriptions). In this way when the SP request a service module the computations taking place in the SCP are limited.

The protocol that will be downloaded from the SCP should also have a mechanism to reassure that this code stems from a trusted SCP (e.g. might have a digital signature that the SP checks). In this way, attacks from malicious code will be prevented.

A check mechanism must be provisioned to check about possible conflicts between the service modules.

The above requirements must be assigned to WP4 (Co-operation between ASP and Management system)

5.3.1.1.2 Requirements from CNM/VPN Application

5.3.1.1.2.1 General Description

The Use Cases involved with this reference point are:

- Use Case “CNM”;
- Use Case “Reconfigure Tunnel”;

In the first use case, the SP has to install the necessary software modules from the SCP for enabling the CNM service, and to remove it when the CNM capability is no longer needed.

In the second use case, the ANSP has to first request for the relevant modules from the SCP via the SP, which will then install the module at all level of the service.

5.3.1.1.2.2 Requirements

- A flexible and scalable management solution that will reduce management traffic and human intervention for running the network and managing the application services provided
- A selection interface should available for the SP to choose the right service modules from the right SCP. The SCP shall provides the SP with specifications about the requirements of the service modules which includes the resources needed in order to have the modules function properly inside an VE/EE.
- A mechanism must exist in order to prevent conflicts between modules.
- From “CNM” use case descriptions above: every active node that is owned by different providers has a common API in order to enable interoperability and common control. The CNM functionalities enable the SP to obtain VPN resources from the ANSP for the creation of the VPN service.
- From “Reconfigure Tunnel” use case descriptions above: In the case where the original VPN tunnel path can no longer guarantee the specified quality of service, the system has to facilitates a mechanism to notifies the ANSP to act upon the situation. The tunnel may be congested, or there are faults within certain nodes or along the path of the tunnel, or there has been a security compromise within tunnel path.

The above requirements have to be realised by work in WP4 on Active Service Provisioning and management systems interactions.

5.3.1.1.3 Requirements from Active Web Service Application

5.3.1.1.3.1 General Description

We did not refer to this point explicitly in use cases.

5.3.1.2 RP2 SP- ANSP

5.3.1.2.1 Requirements from Multicasting Application

5.3.1.2.1.1 General

In this RP interactions between the SP and the ANSP are concerned. We referred to this RP in all the use cases. Since we have made the assumption that ANSP provides the EE of the Multicast Application and implements\managers the interactions between the multicast components (Multicast Agent routers) while SP is responsible for providing the secure and reliable multicast algorithm, these two roles should co-operate for the multicast completion.

5.3.1.2.1.1.1 Requirements

The requirements for the system that are derived from this RP are as follows:

- An interface should be established between these two roles in order to communicate. Under this interface the SP injects the Service modules (Sender Agent, Receiver Agent, Multicast Router Agents, Multicast high quality application Agents, Corba Object Multicast tables) in order to be executed. ANSP checks the service modules according to their service description. If no problem appears then process them to the appropriate EE. If conflicts or other problems appears then informs the SP that the execution cannot be done.
- Since Multicasting is a complex Application the idea of distributed programming would be beneficial. ANSP must build several EE in several VE to distribute the resources and the interactions of the application.
- During the execution of the service modules a mechanism to check and to monitor whether the ANSP can reserve the appropriate resources from the corresponding nodes should be established. In cases that the ANSP identifies that the consumer's request cannot be satisfied (lack of resources, low QoS etc.), this mechanism should also be able to give information about the reasons that have generated the problem.
- Authorisation and authentication mechanisms should also be provisioned for security reasons. That means, that every SP that would like to interact with the ANSP should declare his identity so that he can prove that he is authorised to ask for the ANSP to reserve resources.
- The SP must have the ability to contact more than one ANSPs (the interaction is being made by the associate ANSPs) in order to choose the most eligible one (use case receiver in different domain).
- A mechanism for accounting the resources a user consumes should also be part of this RP.

The above requirements must be assigned to WP4 (Management system).

5.3.1.2.2 Requirements from CNM/VPN Application

5.3.1.2.2.1 General Description

The Use Cases involved with this reference point are:

- Use Case "CNM";
- Use Case "Request VPN";
- Use Case "Create Tunnel";
- Use Case "Modify VPN";
- Use Case "Reconfigure Tunnel";

- Use Case “Terminate VPN”;
- Use Case “Terminate Tunnel”;
- Use Case “Release User”;

It was assumed that the SP handles with Consumer authentications, authorisations and running applications across the tunnel, so only those use cases that are related to the setting-up and termination of the Active VPN tunnel service are relevant to the interactions in RP2.

In the first use case, the SP obtains information about configuration management, performance management, and fault management from the ANSP. The ANSP obtains these capabilities from the NIP in order for him to provide this service to the SP.

In the second use case, the SP notifies the ANSP for the set-up of an Active VPN on behalf of the consumer.

In the third use case, the SP request an Active VPN tunnel from the ANSP and also provide the ANSP with the necessary software modules needed by ANSP for a successful creation of the tunnel.

In the fourth use case, the SP first calculates the amount of changes needed to enable these modifications. These changes could be terminating a tunnel and create another new tunnel, or create a new tunnel in order to extend an existing tunnel. For the next step the SP would request he ANSP for the amount of resources in order to enable the changes and also to collaborate with the ANSP in some of the changes that needs the ANSP to do.

In the fifth use case, the SP acts as a middle person between the ANSP and the SCP. In order to reconfigure the tunnel, the ANSP might need to upgrade existing software modules on NIP nodes from the SCP and must request the modules via the SP as described from the FAIN Enterprise Model.

In the sixth use case, the SP notifies the ANSP on behalf of the Consumer to terminate an Active VPN connection. The ANSP will then send an acknowledgement to the SP that the termination will proceeds.

In the seventh use case, the SP continues from the previous use case “Terminate VPN” and initialises the process of terminating all the relevant tunnels the Consumer currently connects to.

In the eighth use case, the SP release all the record of the Consumer and resources reserved for the Consumer after all the tunnels the Consumer connected have terminated.

5.3.1.2.2.2 Requirements

- A dynamic service deployment mechanism should be included for enabling the SP to install new modules, upgrade components, and protocols onto the ANSP EE. The SCP provides the modules the SP uses. During the code injection process when and if the SP has no rights to access certain parts of ANSP EE, there should be a mechanism for the SP to give the ANSP the relevant modules to complete the code installations.
- There should be a mechanism to priorities resources and therefore reduce resources conflicts and to allows the ANSP to “borrow” more resources from neighbouring NIP should the need arise (e.g. to satisfies all the Consumer’s QoS needs).
- An authentication and authorisation mechanism should be available at all level, including the interactions between the SP and the ANSP. The SP has to be what he says he is and that he has to be authorised in order to utilised service from the ANSP.
- Billing and accounting should be done at this level for how much resource the Consumer has consumed the Active VPN service from the system.

Above requirements has to be realised by work in WP4 on management systems interactions.

5.3.1.2.3 Requirements from Active Web Service Application

5.3.1.2.3.1 General

We referred to this RP in the following use cases: "Configure Distribution Infrastructure AWS", "Distribute AWS ","Modify AWS", "Subscribe AWS", and "Use AWS". In the first use case, SP purchases managed resources from ANSP. In the second use case SP implements active code and installs on appropriate active nodes which are owned by NIP. In the third use case SP modifies individual resources which have been purchased from ANSP in order to accommodate to Consumer. In the fourth use case SP collects usage information which is provided by ANSP according to status of active nodes.

5.3.1.2.3.2 Requirements

The requirements for the system that are derived from this RP are as follows:

- There shall be SLA between SP and ANSP. SP shall purchase single or multiple types of managed resources on active nodes. Resources type shall include cache space, CPU time, or CPU ratio (e.g. assigned 5 % of CPU time), QoS in order to support active web service. Cache space might be provided not only with short-term but with long-term. Short-term may mean controlling per milli-second, second, minute, hour and long-term mean a controlling per day, week, month and year. QoS may be provided in several type, guaranteed, best-effort, etc.
- When a ANSP contracts with multiple SP's, ANSP may act as a delegate of NIP and assign resources to each SP. Inside of a assigned resource, SP might be able to manage its own resource without intervention of ANSP.
- ANSP should provide API for SP to design and implement SP's own service codes which will be executed on active nodes.
- ANSP shall be responsible to installed web service codes, which is implemented by SP, on appropriate active nodes.

The above requirements must be assigned to WP4 (Management system).

5.3.1.3 RP3 ANSP – NIP

5.3.1.3.1 Requirements from Multicasting Application

5.3.1.3.1.1 General

In this reference point, interactions between the Active Network Service Provider and the Network Infrastructure Provider are concerned. We referred to this RP in the following use cases: "Install Multicast Protocol"(explicitly) and "Path Reconfiguration"(implicitly). In both cases, we assumed that since NIP allocates physical resources and builds the VE where the EE are to be developed, the interaction between ANSP and NIP is essential for providing a homogeneous network infrastructure. In this way restrictions because of the consumer's location in the network do not exist.

5.3.1.3.1.2 Requirements

The requirements for the system that are derived from this RP is as follows:

- An interface should be established between these two roles in order for the ANSP to use the NIP. Under this interface the NIP must provides physical active networking resources. In the construction of Multicast tree the NIP must reassure that user tailor multicast application service is going to be provisioned.

- Due to traditional IP-Multicasting limits the group services to the weakest service named within the group the NIP must offer dedicated environment (isolate environments according to the QoS of the receiver paths).
- A computation mechanism must be developed to help the allocation of resources. ANSP must be informed if the NIP cannot provide him with the appropriate physical resources.
- The NIP through Monitoring could inform the ANSP about the resource usage per EE in order to be possible to map the service module in the appropriate EE (optimal resource reservation in different multicast group levels).
- NIP must be able to install and remove code within a VE in order to satisfy new requirements by the application running to the multiple EEs (Path Reconfiguration).
- An event mechanism should be developed to inject events asynchronously in the event detection layer (ANSP). Through these messages ANSP could be implemented an appropriate mechanisms for fault notification and local decision dissemination
- ANSP should authenticate himself to the NIP.
- There should exist a mechanism for the ANSP to deliver the active code to the NIP, who will perform the actual injection of the code.

  The above requirements must be assigned to WP3 (RCF and Security system).

5.3.1.3.2 Requirements from CNM/VPN Application

5.3.1.3.2.1 General Description

The Use Cases involved with this reference point are:

- Use Case “CNM”;
- Use Case “Create Tunnel”;
- Use Case “Modify VPN”;
- Use Case “Reconfigure Tunnel”;
- Use Case “Terminate Tunnel”;
- Use Case “Monitor VPN”;

In the first use case, the ANSP request the NIP to provide him with the basic active node resources in order for him to manage his own Virtual Environment.

In the second use case, the ANSP using the management capabilities enabled by the NIP to interface with NIP nodes in order to obtain the resources needed for creating a tunnel.

In the third use case, the ANSP using the same “CNM” method to change/add new Active VPN tunnel in order to fulfil the consumer’s request.

In the fourth use case, the ANSP again using the same method as above in order to configure the NIP nodes to either reroute the tunnel via other nodes, or to drop packets with the lowest priority.

In the fifth use case, the ANSP received the request from the SP to disconnect a Consumer tunnel. All the resources will be release back to the NIP system that was allocated to the ANSP.

In the sixth use case, the ANSP system detected a fault or there is a severe drop in Consumer QoS. When this happened the ANSP monitoring system would notify the ANSP management system to correct the problem.

5.3.1.3.2.2 Requirements

- The "CNM" capability should be able to allow the ANSP to interface with the nodes of the NIP in order to control some resources of it. In the case of Active VPN the ANSP should be able to control using ports on the node in order to create Consumer specific packets and to route them to the destination.
- The node should have the ability to encapsulate internal Active VPN IP address into each packet so that it would be able to create a "tunnel" environment (IP Masquerade). This environment enables the two distance networks to function as if they are one network.
- A resource analysing mechanism is needed in order to allow the NIP to report to the ANSP the amount of resources available to the ANSP to utilise.
- The NIP should prioritise node resources with sets of rules based on the SLAs with the ANSP. The rules can be a set of policies stored in a resource policy database.
- The NIP can authenticate the ANSP and a mechanism within the NIP authentication system should includes a map to the functions the ANSP is authorise to use.
- The interface mechanism between ANSP and the NIP should allows the ANSP to inject codes onto his VE and subsequently installed onto the node by the NIP.

Above requirements has to be realised by work in WP3 on Resource Control Frameworks and system security.

5.3.1.3.3 Requirements from Active Web Service Application

5.3.1.3.3.1 General

We referred to this RP in the use case of "Configure Distribution Infrastructure AWS", "Modify AWS", "Use AWS", explicitly, however IP connectivity always should be supported by NIP and based on this function ANSP could provide managed resources to SP. In the use case mentioned above, NIP should monitor and measure statistics of active node usage and report to ANSP.

5.3.1.3.3.2 Requirements

The requirements for the system that are derived from this RP are as follows:

- There shall be SLA between ANSP and NIP. SLA would be translated to policy data that are readable by active nodes.
- When a NIP contracts with multiple ANSP's, NIP may assign a VE (Virtual Environment) for each ANSP. Inside of a VE, ANSP may be able to manage its own resource without intervention of NIP.
- NIP shall need to be requested to inform available resource to ANSP.
- NIP shall measure usage information of resources and report it to relevant ANSP periodically. Measurement of resources may be on each VE, i.e. a VE may be assigned to a ANSP and ANSP may get usage information of its own only.
- Granularity of IP packet monitoring should be not only on IP source/resource address, but also on high-level or application-level, e.g. HTTP. In addition markup tags, i.e. HTML tag, XML tag, should be filtered and analysed in active nodes.

The above requirements must be assigned to WP3, especially functionalities of resource control framework and de-multiplexing.

5.3.1.4 RP4 Consumer-SP

5.3.1.4.1 Requirements from Multicasting Application

5.3.1.4.1.1 General

In this reference point, the interactions between the consumer and the Service Provider are to our interest. We proposed the use of it in the following use cases: “Initial Construction of a Multicast Tree”, “Adding a Member to a Multicast Group”, “Installing a Multicast Protocol on an Active Node”, “Deleting a Member from a Multicast Group”, “Deconstructing a Multicast Tree” and “Path reconfiguration”

5.3.1.4.1.2 Requirements

The requirements for the system that are derived from this RP are as follows:

- An establishment of interface, between SP and the consumer. Through this interface, the consumer can ask from the SP the multicast service, or the SP could advertise the available services to the consumer.
- The consumer should authenticate himself to the SP.
- In case that the multicast service cannot be fulfilled, the SP must inform the consumer if a problem in the multicast service has incurred, as well as about the possible reasons that caused the specific problem.
- The consumer should also be able to specify the parameters under which he wants the multicast service to take place (e.g. QoS). If the level of the QoS for example is not acceptable the consumer may have the option either to continue the service with a lower level or to terminate the procedure.

The above requirements must be assigned to WP4 (Management system).

5.3.1.4.2 Requirements from CNM/VPN Application

5.3.1.4.2.1 General Description

The Use Cases involved with this reference point are:

- Use Case “CNM”;
- Use Case “Authenticate User”;
- Use Case “Request VPN”;
- Use Case “Create Tunnel”;
- Use Case “Modify VPN”;
- Use Case “Use VPN”;
- Use Case “Terminate VPN”;
- Use Case “Terminate Tunnel”;
- Use Case “Release User”;

In the first use case, the Consumer has requested a “CNM” related service and functionality from the SP which has triggered a request for “CNM”.

In the second use case, the Consumer has made a request to the system and was authenticated by the SP security system.

In the third use case, the Consumer requests the Active VPN service from the SP and thus started a chain of events that initialise the Active VPN tunnel.

In the fourth use case, the Consumer on the other end (i.e. the server) received a tunnel request from his SP and acknowledges the request. Both SP on the Client and Server side must make sure that the correct software service modules were installed on the relevant nodes in order to allow the set-up of the tunnel.

In the fifth use case, the Consumer requests the SP to make a new tunnel connection to a new Consumer Server. The Consumer request might be involved with the taking down of an exiting tunnel, or to increase the number of tunnel currently connected with the Consumer.

In the sixth use case, the Consumer Client runs applications from the Consumer Server on the other end of the tunnel and the ANSP monitors and maintains the correct QoS with the right packet that is related to the Consumer.

In the seventh use case, the Consumer client sends a request to terminate some or all of his Active VPN connections with the Consumer Server.

In the eighth use case, the Consumer Server was notified about the termination of the tunnel and acknowledges the SP on his side.

In the ninth use case, the Consumers on both side of the tunnel have been notified of the termination of the Active VPN service.

5.3.1.4.2.2 Requirements

- There must be a mechanism to ID tag the Consumer sessions and maps it onto the SP part of the ANSP EE for the packet handler to distinguish different Consumer packet.
- The SP would first authenticates the Consumer and a database should be available to the SP in order to allows him to map the correct identity of the Consumer to the service that he is entitled to use.
- Initially the SP should be able to advise the Consumer the availability of the Active VPN service as one of the many services provided to the Consumer. When the Consumer request the service, there should be a service handler (with a nice easy to use GUI) to obtain details of the destination from the Consumer for establishing an Active VPN connection. The details can be the IP address of the destination Consumer Server gateway, and the QoS the Consumer require.

Above requirements has to be realised by work in WP4 on management systems interactions.

5.3.1.4.3 Requirements from Active Web Service Application

5.3.1.4.3.1 General

We referred to this RP in the following use cases: “Configure Distribution Infrastructure” and “Subscribe AWS”. In the first use case SP advertises its service with using web pages. In the second use case Consumer subscribes service from SP using a SLA. In the third use case, Consumer un-subscribes service and pays for service based on a SLA.

5.3.1.4.3.2 Requirements

The requirements for the system that are derived from this RP are as follows:

- A Consumer shall order web service via SP. A SP shall offer a variety of web service level that may be mapped on differentiated tariff. The contract, i.e. SLA, between a Consumer and a SP should be defined with high-level terms and besides, a Consumer should not be aware of details of assigned resources on active nodes. The SLA may be translated to lower level parameters that would be used for a SLA between SP and ANSP.

- A SP may bill according to usage of web service. Billing may depend on service level which has been chosen by Consumer.

The above requirements might be assigned to WP4 (Management system), especially aspects of SLA specifications and translation of SLA into policies.

5.3.1.5 RP5 SP-SP

5.3.1.5.1 Requirements from Multicasting Application

5.3.1.5.1.1 General

This reference point deals with interactions between different SPs. In the use cases for a multicast service, we have not dealt with this reference point. A single Service Provider offers the multicast service to the consumer. This SP could however use multiple ANSPs in order to fulfil his service.

5.3.1.5.2 Requirements from CNM/VPN Application

5.3.1.5.2.1 General Description

The Use Cases involved with this reference point are:

- Use Case "Terminate Tunnel";
- Use Case "Create Tunnel";

In the first use case, the SP on the Consumer Client side is different from the Consumer Server side. The SP Client has to communicate with the SP Server in order to take down the tunnel connection. The SP on the Server side would first obtain confirmation from the Consumer Server and then would reply back to the SP on the Client side. The SP from both sides would then interface with the ANSP on both sides to disconnect the tunnel.

In the second use case, the SP will first communicates with the other SP on the Consumer Server side in order to obtain the connection parameters. The next step for the SP on the Client side will be to request the ANSP to establish the connection with the given parameters.

5.3.1.5.2.2 Requirements

- The SP will first authenticates the other SP and to confirm with each other the identity of the two Consumers for Active VPN service.
- The must be a mechanism to define the signalling packet (for establishing a Active VPN connection) as a highest priority so that the service can be effectively set-up.

Above requirements has to be realised by work in WP4 on intra-domain management systems interactions.

5.3.1.5.3 Requirements from Active Web Service Application

5.3.1.5.3.1 General

We referred to this point in the following use cases: "Configure Distribution Infrastructure", "Distribute AWS", "Modify AWS", and "Subscribe AWS". In the first use case, SP promotes its service to other SP. In the second use case, SP uses other SP, which could manage resources on active nodes, in order to distribute active codes into active nodes. In the third use case, SP notifies the information of Consumer's joining/disjoining to other SP.

5.3.1.5.3.2 Requirements

The requirements for the system that are derived from this RP are as follows:

- A SP (Web Service Distribution Provider: WSDP) negotiates with other type of SP (Web Service Provider: WSP) in order to offer its web sites for advertising.
- A SP implements service logics based on End User information and uses other SP's facility to inject these service logics into active nodes.

The above requirements must be assigned to WP4 (Management system).

5.3.1.6 RP6 ANSP-ANSP

5.3.1.6.1 Requirements from Multicasting Application

5.3.1.6.1.1 General

This reference point deals with the interactions between different ANSPs. We referred to this RP in the use cases "Adding a member in the Multicast group" and "Deconstructing a Multicast Tree ". This RP covers the situations where different domains are concerned. For example, cases where consumers belonging in different domains need to be served in the same multicast procedure. The ANSPs should co-operate so that the basic network services, such as IP connectivity, will be achieved.

5.3.1.6.1.2 Requirements

The requirements for the system that are derived from this RP is as follows:

- An interface between these roles should exist in order to communicate. Under this interface the second ANSP builds a new Infrastructure to add a new member in the multicast group. The first ANSP is responsible for the connectivity between the SP and the second SP. A migration mechanism must be developed to guide the service modules in the EE of the different domain.
- Each ANSP should notify the ANSPs that are connected with, about his available resources along with any problem that might have occurred and leads the multicast service to a halt.
- Mechanisms for monitoring the network status, as well as the resources of the network should also be provisioned.
- During the ANSPs connectivity, issues about the ANSP's authorisation and authentication arise.
- Since additional ANSPs are used an accounting mechanism should be provided.
- Mechanisms for code transfer between different ANSP.
- Protection of the ANSP from potential attacks of malicious code. That means that the code that is going to travel between the ANSPs should be authenticated (e.g. digital signature, encryption mechanisms etc.)

The above requirements must be assigned to WP4 (Management, Security system).

5.3.1.6.2 Requirements from CNM/VPN Application

5.3.1.6.2.1 General Description

The Use Cases involved with this reference point are:

Use Case "Create Tunnel";

In the first use case, the ANSP would use the parameters given by the SP on his side in order to establish a connection with the ANSP on the other side (i.e. the Server side of the Consumer).

5.3.1.6.2.2 Requirements

The ANSPs would authenticate reach other and should also check whether they have route to each other with their current NIP.

There should also be a mechanism to check whether the service modules need to be installed onto the node across to another ANSP are not conflicting with existing service modules. There should also be a virus scanning mechanism in order to prevent the infection of virus from spreading from one provider to the other.

There should be a mechanism for the two ANSP to exchange their systems resources information so that one could reconfigure the network to suit the other for an optimal interconnection.

There should also be an mechanism to monitor the resources available to the combined network and to report any problem to both ANSPs.

Above requirements have to be realised by work in WP4 on management systems interactions and system security.

5.3.1.6.3 Requirements from Active Web Service Application

5.3.1.6.3.1 General

We refer to this RP in use cases of "Configure Distribution Infrastructure" and "Use AWS". We assume that there may be a variety of active nodes and each could be managed different ANSP's. For instance in AWS use cases we proposes an active web server as service node, an active router as redirect node and an active access gateway. Since these distinct types of active nodes should support end-to-end QoS and service path, multiple ANSPs must co-operate for integrated network services.

5.3.1.6.3.2 Requirements

The requirements for the system that are derived from this RP are as follows:

- ANSP1 could negotiate with ANSP2, which owns different type of active nodes, in order to provide end-to-end QoS guaranteed connectivity or redirect functionality.
- ANSP shall exchange information of its available resources with other ANSP in order to support network service among multiple domains.

The above requirements must be assigned to WP4, especially aspect of inter-domain.

5.3.1.7 RP7 NIP-NIP

5.3.1.7.1 Requirements from Multicasting Application

5.3.1.7.1.1 General

This reference point deals with the interactions between two Network Infrastructure Providers and we referred to it in the use case: "Adding a member in a Multicast Group", "Deleting a Member from a Multicast Group", "Deconstructing a Multicast Tree" and "Path Reconfiguration". In each one of these use cases the aim of this co-operation was to add generalisation to the network. Since, a single NIP is not able to provide global transport capabilities, their federation with other NIPs is essential. This federation is mainly based on geographical considerations.

5.3.1.7.1.2 Requirements

The requirements for the system that are derived from this RP is as follows:

- The development of an interface that will allow the NIPs to federate. Using this interface, the NIPS should be able to co-operate in order to provide inter-domain communication. They should also negotiate the QoS and accounting issues regarding the inter-domain traffic.
- If some NIPs cannot communicate a mechanism for notifying each other, about the reasons of the problems, should be provisioned.

The above requirements must be assigned to WP3.

5.3.1.7.2 Requirements from CNM/VPN Application

5.3.1.7.2.1 General Description

The Use Cases involved with this reference point are:

- Use Case "Create Tunnel";

In the case where there are multiple NIP, the NIP would need to communicates with one another so that they could send Consumer packets through each other's network in order to serve the Consumer.

5.3.1.7.2.2 Requirements

There should be an intercommunication method to enable each NIP to share basic resources and service capabilities. They should be able to guarantee the same level of QoS across their joined domain.

There should be a mechanism that synchronises the service modules available and the QoS across multiple NIPs for the provision of an Active VPN service.

Above requirements have to be realised by work in WP3 on active node platform and environment.

5.3.1.7.3 Requirements from Active Web Service Application

5.3.1.7.3.1 General

We refer to this RP in use case of "Use AWS". We assume that IP connectivity should be supported among multiple NIPs with guaranteed QoS being required by its Consumers.

5.3.1.7.3.2 Requirements

NIP shall exchange information of its available resources with other NIP in order to support network connectivity among multiple domains with reserving resources on their active nodes.

5.3.2 Requirements Description for the Business Roles

5.3.2.1 Requirements from Multicasting Application

Due in part to the complexity of operations, multicasting involves almost all the business roles described in the FAIN business model. Also, interfaces between the equipment of these business roles are must provide means to facilitate the deployment of active multicast services and their operation. This documents presents a set of key requirements that are to be met by an active multicast network infrastructure. The requirements are intended to be used as guidelines for functional specification of active multicast supporting services within WP3 and WP4. These requirements are derived from the multicast application use case. They do not give a complete set of functionalities, nor determine the architecture or the functions of active multicast networks. They represent en initial set of indications that could be refined or extended for actual service specifications or implementations.

In summary, this document derives key requirements for the different reference points and business roles, and (virtually) assigns tasks to the other work packages or conditions their work. All the FAIN reference points and business roles are concerned in our approach. The first part of the document deals with requirements coming from each of the business roles involved in active multicast networks in general (and game networks in particular). The second part gives a short description, followed by a set of multicast requirements, for each reference point.

5.3.2.1.1 End-Users

In the general case, end-users are equipped with end-systems that can access the active network infrastructure. This can be done by means of a LAN connection, a fixed telecommunication network connection or a wireless connection. End systems are typically required to run multicast communication software with:

- The possibility to participate in several multicast communication sessions simultaneously. These communication sessions can be used to handle control (data) related packets, audio-visual packets, or multimedia communications.
- The possibility to initiate, control and manage multicast communication sessions, that will be used by homologue end-systems to participate in a multicast service session
- In addition, the user interface should allow the end-users to specify a network service quality that varies according to its actual need: real-time video quality, control delays, reliable degree of reliability in communications, ordering of messages, synchronisation with homologue end-systems, secure delivery and authentication, etc. This later type of requirement in particular will be enabled (or eased) by active networks.

5.3.2.1.2 Application Providers

Application provides play a key role in multicast communications between stakeholders and system components, since they are responsible for the exploitation and the normal operation of network and information services. Some of the key requirements on application providers are the following:

- The provision of multicast communications between application servers, that runs on top of an active multicast transport network. Application level multicast communications are required to implements enhanced computing and communication software, that rely of multicast transport-level services to realise flexible bindings between application software components with a wide range of QoS requirements. The flexibility of these multicast communications are enabled by computations at active transport network nodes, while the QoS guarantees are achieved by network resource control mechanisms.
- Application providers must also allow multicast communications between end-systems, especially for co-ordinating their use of the multicast network services.
- Application providers are the triggers, the clients and the controllers multimedia communications within an active multicast transport network. Thus, they are required to implement services that take into account all the possibilities offered by the transport network, both for the coordination of the activities of application servers and for services provided by end-users.
- Some of the application providers may also need to implement network management services, if their purpose is related to network management. Typically, a policy-based network management system is seen as an application provider that is specialised in active network management.

5.3.2.1.3 Active Network Providers

Active network providers are required to implement communication software supports in their routers with the following features in particular:

- A set of multicast routing protocol implementation and services, that can interpret and respect QoS requirements expressed by a multicast traffic or individual packets. For example, an active multicast traffic or a packet must be able to configure any router so as to guarantee a real-time delivery at the expense of reliable delivery or vice-versa.
- Also, the set of active multicast routing protocols and services must be modifiable from the outside, on behalf of a multicast traffic or a packet.
- The architecture of service at a router must enable a remote access to some of the service components, for the purpose of remote control, remote modifications or remote access to network resource availability, in order to allow the implementation of end-to-end active network services. This last requirement is of primary importance, for e.g. WP4, where it is expected to define and prototype active network services frameworks for end-to-end active packet processing and new network management concepts.

5.3.2.1.4 Multicast Protocol Providers

Multicast protocol providers are required to establish strong collaborations with active network providers and application providers in particular, so as to specify, develop and ensure the availability of active network protocol components with some given properties and interface specifications. Some of the technical requirements of software components developed by multicast protocol providers are the following:

- Multicast protocol components that can satisfy some specified real-time delivery semantics.
- Multicast protocol components that can satisfy some real-time components.
- And a combination the two properties above: multicast protocol components that can establish a specified trade-off between real-time delivery and reliable delivery.
- Also, multicast protocol components with interfaces that allow modifications on their computational behaviour

5.3.2.1.5 Game Network Providers

As it is presented in the multicast application use case description, game network providers are application provider that are specialised in the provision of a network of game servers, that can support the execution of multiple game platforms and environments. Game network providers are required to implement:

- Generic computing and communication services, that will realise the bindings between service components running on centralised game servers.
- Take into account multicast communication with QoS and flexibility expressed by game applications and platforms.

5.3.2.1.6 Media Network Providers

A media network provider in a game network environment is the transport network that can handle multimedia communications between external systems: game servers and end systems. As such, a media network is required to implement:

- Mechanisms for the communications of data, audio, video sequences between game servers and end-systems, that can rely (for example) on active media interpolation techniques. Routers could implement such techniques within specialised virtual environments (see WP3) or resource control frameworks and services with special purpose algorithms.
- Mechanisms to allow modifications on computations associated to these computations, from a game network.

5.3.2.1.7 Game Platform Providers

Game platforms providers are application providers, that offer distributed toolkits for the development of distributed game services and environments. Game platform provides are require in particular to:

- Generic distributed control services for the control and coordination of several game services and environments.
- Persistence, isolation, consistency, security or directory services, that will be used to by several game environments and services.
- Authentication, isolation or end-to-end security mechanisms for the liaison between environment objects

5.3.2.1.8 Game Service and Environment Providers

Game service and environment providers are application providers that interface end-users, game platform provider and game network providers. Some of their main requirements are:

- To develop and operate game software, using game platform and game networks.
- To develop and operate game access and usage mechanisms.
- To develop various other mechanisms, for e.g. billing, protection, authentication etc., for potential.

5.4 References

[TMF2] Generic Requirements for Telecommunications Management Building Blocks, GB909, Member Evaluation Version 2.0, TeleManagement Forum, September 1999

[IEEE STD 610.12-1991]: IEEE Standard Glossary of Software Engineering Terminology, IEEE

[MSF] MSF-ARCH-001.00-FINAL IA: Implementation Agreements and Architecture Framework. MSFORUM

6 DESCRIPTION OF REFERENCE POINTS

6.1 Introduction

Reference Points are a means to describe the relations between the different components of a system. The introduction of reference points follows the architectural guideline "separation of concerns". We have given a an overview over the reference points in FAIN in Chapter 3, where we have introduced the FAIN Enterprise Model, its different roles and the Reference Points between these roles. While this document provides a high-level overview over the reference points, a more detailed technical specification of these reference points is necessary.

The specification of a reference point should cover the detailed and formalised description of the following aspects of a reference point:

- *Business aspect:* scope limitations, functional and non-functional requirements posed on the business relationship by the business roles. It is derived form the requirements of the business roles for their interaction.
- *Informational aspect*: defines the information, which is shared between the domains.
- *Computational aspect:* defines interfaces on computational objects to be made accessible to the other domain. To promote re-use and manageability the computational interfaces can be grouped into different types according to their use.
- *Engineering aspect:* maps the Reference Points to the FAIN active node model (“the cube”).
- *Miscellaneous:* Defines other constraints, allowed limitations on compliance.

Since the development of viable specifications of reference points is a resource intensive task, which also involves experimentation with and feedback from real world application and implementation scenarios, it is not requireed within WP2 to specify all reference points in a full detail and in a fully formalised fashion:

- RP1 between Service Component Provider and Service Provider. The service provider offers services to end users, which have been subscribed for the end user by some customer. The service provider can offer services, which are composed from services provided by one or several service component providers. Since for RP1 specifications from previous projects [TinaC] could be reused and adapted, RP1 is specified in a relatively formal and detailed way.
- RP2 between Service Provider and Network Service Provider: This reference points describes how external applications, which are to be provided by service providers, and as they have been investigated in Chapter 4, interact with the FAIN system. For this reason, the RP2 represents one of the main contributions of this document. It is defined in detail and with an emphasis on the relationship to the applications.
- RP3 between the Network Service Provider and the Network Infrastructure Provider is in our opinion an important reference point when it comes to the deployment of active network technology e.g. in the public internet. RP3 describes the interface between an owner of physical network infrastructure (NIP) and between an active network services providers who would like to provide an active networking infrastructure to its customers, i.e. to the service providers. RP3 has not been specified in detail, since it is closely tied to business strategies for deploying active networks in the field, which clearly do not exist at the moment. However, the various design options and their implications are analysed and it is suggested to continue this discussion in the remainder of the FAIN project.

- RP4 between Consumer and End User is closely related to the TINA Retailer reference point. The Consumer requests and uses services offered by the Service Provider. The SP provides the service with its functionality. It also subsumes the role of the Retailer as service access point (i.e., it determines the service contract). Since also for RP4 specifications from previous projects [TinaC] could be reused and adapted, RP1 is specified in a relatively formal and detailed way.
- RP5, RP6 and RP7 are federation reference points. Although these are clearly important for real world application scenarios, they are left open for further investigation due to resource limitations.

We try to use the descriptive formalisms of the unified modelling language UML (see e.g. [SW98]) as far as appropriate and possible.

[SW98] Desmond Francis D'Souza and Alan Cameron Wills: Objects, Components, and Frameworks with UML. The Catalysis Approach. Addison-Wesley, 1998.

[TinaC] Business Model and Reference Points Version: 4.0, Date: 22 May 1997 http://www.tinac.com/specifications/abstract.htm

6.2 Reference Point RP2 (Service Provider – Active Network Service Provider)

6.2.1 Information Model

6.2.1.1 The Business Model

The reference point 2 is defined between service providers and active network service providers. Through RP2, the ANSP offers to the SP several network service capabilities –previously negotiated- that should allow the SP to comply with the service requirements demanded by consumers. The term '*service capabilities'* has been adopted to refer to the set of resources4 and functionalities offered by the ANSP to provide customised execution of a service (or related components) within its own domain.

In a wide sense, the ANSP provides an *active network service* by adding a set of virtual environments facilities representing the service capabilities to the IP transport network. Each virtual environment abstracts one or more execution environment services. For example, a set of DPE-like facilities, allowing the interaction between components that execute in different execution environments, provides a distributed active code service within a virtual environment.

The SP accesses the virtual environments through the operational interfaces defined in the reference point with the aim of providing active services to its clients.

6.2.1.1.1 Overall Functionality and Scope

The overall functionality of the RP2 is to provide the access and management interfaces of the service capabilities offered by the ANSP. We distinguish two phases in the interactions that occur between the SP and the ANSP: the initial access phase, and the usage phase of the service capabilities.

To be able to access the service capabilities offered across a domain, a SP is required to establish a communication session with the corresponding ANSP. Establishing a communication session may require an authentication process. Once identities have been validated, the SP and ANSP negotiate the service agreements that should be provided during the session.

[4] Note that execution environments are also considered as resources.

During the communication session lifetime, the SP may request the establishment of execution sessions within the ANSP domain. An execution session is to be opened within an active node in order to obtain the service capabilities required for the actual service execution on it. Thus, the establishment of an execution session is closely related to the virtual environment facilities provided to the SP. The SP may also request to modify and release the execution session. Within an execution session, several active channels can be opened. An active flow transports active packets – that may contain executable code- towards an endpoint, referred in this document as an active flow endpoint (for example, a video compressing active application installed in an active network might act on packet flows containing video data). An active flow endpoint abstracts an entity that is able to gather the active packets coming from an active flow and launch their execution. Such active packets receive the service capabilities negotiated for the active flow –which will be a subset of the service capabilities available on the virtual environment-.

6.2.1.1.2 Business Requirements

In this section the interactions formerly introduced are further described. The set of interfaces and operations required in each phase is analysed as well as the implications on other reference points.

6.2.1.1.2.1 Access Requirements

An initial access part is required to authenticate the SP before usage part interactions with the ANSP actually occur (this requires checking the user's contract profile and includes all the security procedures).

- An authentication interface shall be provided by the ANSP. This shall be used whenever the SP requests a context modification as well as when a usage interface is to be requested or when a periodic authentication checking is to be done -or even at the time of a usage.
- The SP may request a usage interface, after authentication, in order to perform usage part related operations.
    - In case of acceptance, the ANSP shall return the usage interface reference with additional parameters such as security parameters and service level requirements. Once a SP has obtained a usage interface, he is allowed to establish execution sessions within the ANSP domain as appears in the service level agreement. Note that a SP could hold several communication sessions with different service level requirements.
    - In case of failure, the ANSP shall send a failure report in response to a usage interface request containing a *failure code* which may mean either that the security requirements are not met or the usage interface can not be allocated.
- Contract termination: The ANSP shall provide a function for terminating the business contract. The ANSP shall first check the validity of any SP termination request.
    - If the request is invalid (due to security problems,), a notification is sent to the SP and the contract is continued.
    - If the request is valid, the ANSP shall delete the contract profile. A notification is sent to the SP about the contract profile and for termination confirmation

6.2.1.1.2.2 Usage Requirements

Once a usage interface has been obtained, the SP shall use it to set up execution sessions along the active nodes included in the ANSP domain.

6.2.1.1.2.2.1 Execution Session Management

An execution session enables the execution of active code in a virtual environment created on an active node. It abstracts the association of the service provider with a set of capabilities provisioned by the ANSP.

Execution Session Set up

- The ANSP shall provide a function to allow the establishment of an execution session.
- When establishing an execution session, the SP shall use the usage interface reference he has previously obtained. This reference shall be related to the service level agreement. Also, information about the required service capabilities shall be provided to the ANSP. This information will be used by the ANSP to create a proper virtual environment (with the required execution environments, resources and facilities).
- Upon receiving the SP request, the ANSP shall return an interface providing access to the capabilities requested, corresponding to the virtual environment interface. The ANSP shall include a set of credentials to be used when accessing the services and interfaces. Each credential assures a certain service level for each service provided.
- If the set up process failed, the ANSP should inform about the failure reasons. Otherwise, the ANSP shall confirm the establishment of the execution session. Failure codes shall be returned if:
  - The service capabilities can not be granted.
- If the execution session has been successfully established, the ANSP shall be responsible for guaranteeing the agreed service level.

Execution Session Suspension

The ANSP should provide a function to disable an execution session. Suspension is made due to possible failure or inaccessibility of network resources supporting the active flows. Suspending the execution session is also an essential condition to be able to rearrange the service capabilities associated to the execution sessions without interfering in the VE operation. Therefore, active code already running in the virtual environment should remain suspended until the session is explicitly resumed (or released). During suspension, only basic connectivity services could be provided. This operation needs the *execution session reference* as a parameter, along with the criterion of the successful completion of the operation (any, all/none flows).

Execution Session Resumption

The ANSP shall provide a function for setting up the administrative state of an execution session to *enabled* state. The resumption of an execution session is ensured by activating all the network resources that would support the active flows. Execution session resumption is necessary to restore an *enabled* state after an execution session has been suspended.

The processing of the resumption request is deemed successful if the ANSP is able to activate all network resources that would support the active flows. If one or more network resources that support the active flow have failed or are not accessible, the resumption request fails. This operation needs the *execution session reference* as a parameter, along with the criterion of the successful completion of the operation (any, all/none flows).

Execution Session Release

The ANSP shall provide a function to release an execution session. Releasing the session releases the service capabilities (resources, services, etc) associated to a SP in a given virtual environment. An execution session cannot be released unless no active flows are attached to it.

6.2.1.1.2.2.2 Active Channel Management

The ANSP shall provide a function for setting up an active channel (that would support related flows) in the context of an execution session. Once an active channel has been established, the active packets carried in the flow shall receive service capabilities when arriving to the virtual environment.

Active Channel Set up

An active channel requiring service capabilities can not be established unless there are live execution sessions in the virtual environments. Active flows within the channel shall receive, at most, the service level assigned to the execution session.

- The ANSP shall provide a means to attach an active flow to an active flow endpoint within an execution session.
- The SP should inform of the channel characteristics (e.g. the kind of active code carried and the resources necessary for that flow). Also, the SP shall identify the active flow endpoint to which the active channel is to be bound.
- The ANSP should either confirm the channel set up or report the failure. The set up shall fail if there are not enough available resources at the active flow endpoint to fit the active flow requirements.
    - Active flow identifier.

Once an active channel has been set up, it is enabled to transport active flows

In case a non-active flow is used, the bind operation attaches the non-active flow to an endpoint directly related to the NIP, who is the basic IP connectivity provider. The SP shall also provide an active flow endpoint identifier and a service endpoint identifier for the application side. These non-active flows would not receive specific processing in the virtual environments.

- Whenever the SP requests unbinding a non-active flow from a service endpoint, the ANSP shall assure that no packet coming from that flow actually reaches the service endpoint. However, the non-active flow shall still receive the connectivity service.

Active Channel Disengagement

The SP shall request an active Channel disengagement to suspend giving service capabilities to new active packets coming in the flow. Such packets will then only receive basic connectivity services.

- The ANSP should provide a function to allow the SP to request channel disengagement.
- The SP shall provide the identifier of the Channel he wishes to disengage.

This functionality is useful in situations where operations on several flows are closely related one each other. For instance, it could be necessary to avoid further processing the voice channels associated to a video stream whenever a lost of information makes the video data unrecoverable. In this case, the corresponding active channels disengagement would be requested.

Active Channel Release

- The ANSP shall provide a function to allow the SP to release an active channel. The active code that came from related active flows can continue its execution within the virtual environment although the active channel has been released
- The SP shall request an active channel release providing its identifier.
- The ANSP will confirm the termination of the active channel.

6.2.1.1.2.2.3 Active Capabilities Usage

- The ANSP shall undertake the processing of packets coming from enabled active flows.

- The SP should be allowed to request the injection of new service components containing new code.
- The ANSP should allow active policies coming from the SP to be deployed on its own domain according the SLA negotiated between the SP and the ANSP.
- The ANSP shall provide a function by which the SP can find whether two active flow endpoints could be bound.
- The ANSP shall provide query capabilities by which the SP can retrieve information about active flow endpoints, existing execution sessions, and active channels.

6.2.1.1.2.3 Non-Functional Requirements

The ANSP should avoid the operations performed by one SP having influence in other SPs if those SPs are not associated[5]. Also, the ANSP should isolate the effects of the management realised by one SP has on other SP view of the services received.

6.2.1.2 Information Model

The information model defined for the reference point RP2 contains the following information objects and interactions:

[5] Although in composition and federation relationships there might be an influence of this type, the ANSP should avoid any execution interference.

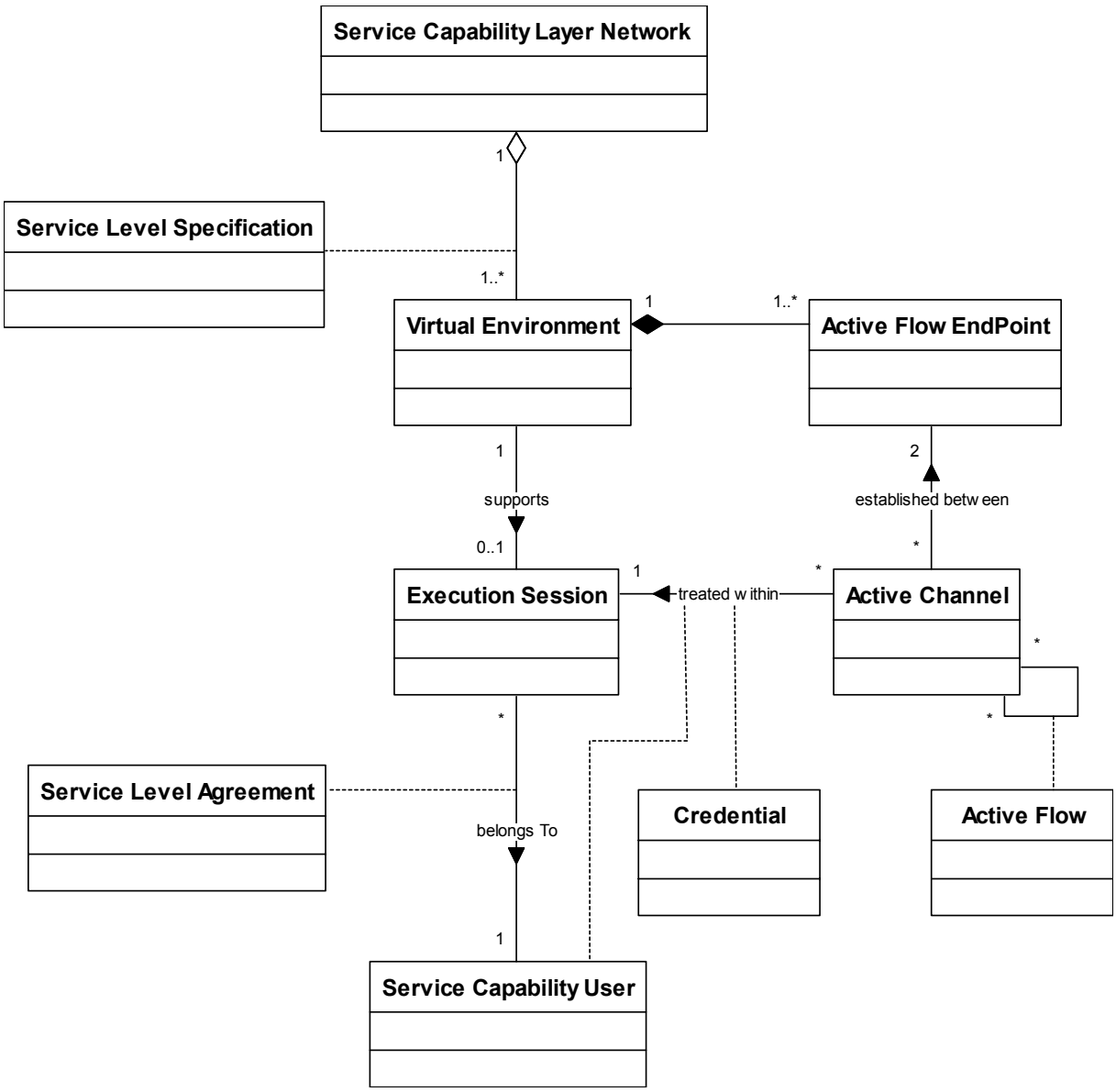


Figure 6-1 Information model usage part for the reference point RP2

Service Capability Layer Network

The service capability layer network consists of a set of virtual environments (at least one). It represents the set of nodes supporting the execution of network services.

Service Capability User

The service capability user represents the entity receiving the active capabilities.

Virtual Environment

A virtual environment shall provide the access interfaces to the service capabilities. A VE can support one or none execution sessions. The VE may handle the interdependency of services running in different execution sessions.

Execution Session

An execution session is a session related to the virtual environment enabling the execution of active code coming from the SP domain in the ANSP domain.

Active Channel

An active channel is a logical link through a set of active nodes in which there exist execution sessions able to handle the active flows corresponding to a certain service. Non-active flows would not be processed by active code in active channel.

To set up an active channel, The ANSP could require the NIP to give information about the active network topology, so that the ANSP can select the active nodes of interest for his service.

Active Flow Endpoint

End point for an active flow, where the active packets are collected and prepared for further processing. Although an active network may contain passive nodes, active flow endpoints are only located in the active nodes.

Two leaf endpoints (source and destination) will be always associated to each active channel.

Active Flow

Packets that have to receive service capabilities are transferred through an active flow. An active flow is associated with a certain execution session, being established between a source active flow endpoint and a destination active flow endpoint. An active flow requires both connectivity services and execution sessions.

Credential

From the relationship between the active flow and the execution session it appears the need to define a set of credentials to authorise the collected active code to access the resources with the corresponding service facilities and conversely, to allow the active code to access particular non-active flows.

Service Level Specifications

The service level specifications are the part of a service level agreement that describes the operational characteristics. It does not include the part directly related to the business details. It would include information on the contracted service capability types and required associated resources (memory, CPU usage profiles, etc).

6.2.1.3 References

[Den00] S. Denazis, “Active Node Reference Architecture”, FAIN Internal Report, December 2000.

[FAIN00] “Initial Active Networks Enterprise Model (Audit Version)”, FAIN Work Package 2 Members, January 2000.

[TIN97] “The ConS Reference Point. Version 1.0”, TINA 1.0 Specification, February 1997.

[TIN95] “Information Modelling Concepts. Version 2.0”, TINA-C Deliverable, April 1995.

[Smi99] J. M. Smith, K. L. Calvert, S. L. Murphy H. K.Orman, L. L. Peterson, “Activating Networks: A Progress Report”, IEEE Computer Magazine, April 1999

6.2.2 Computational Model

6.2.2.1 The Object Model

Figure 6-2 shows the object model for the reference point 2. Since we have used the TINA concepts for describing the access session, the access related TINA computational objects appear in it[6]. We have focused the main contribution of the RP2 computational model on the description of the interactions that come up with the establishment of an appropriate service path within an active network.

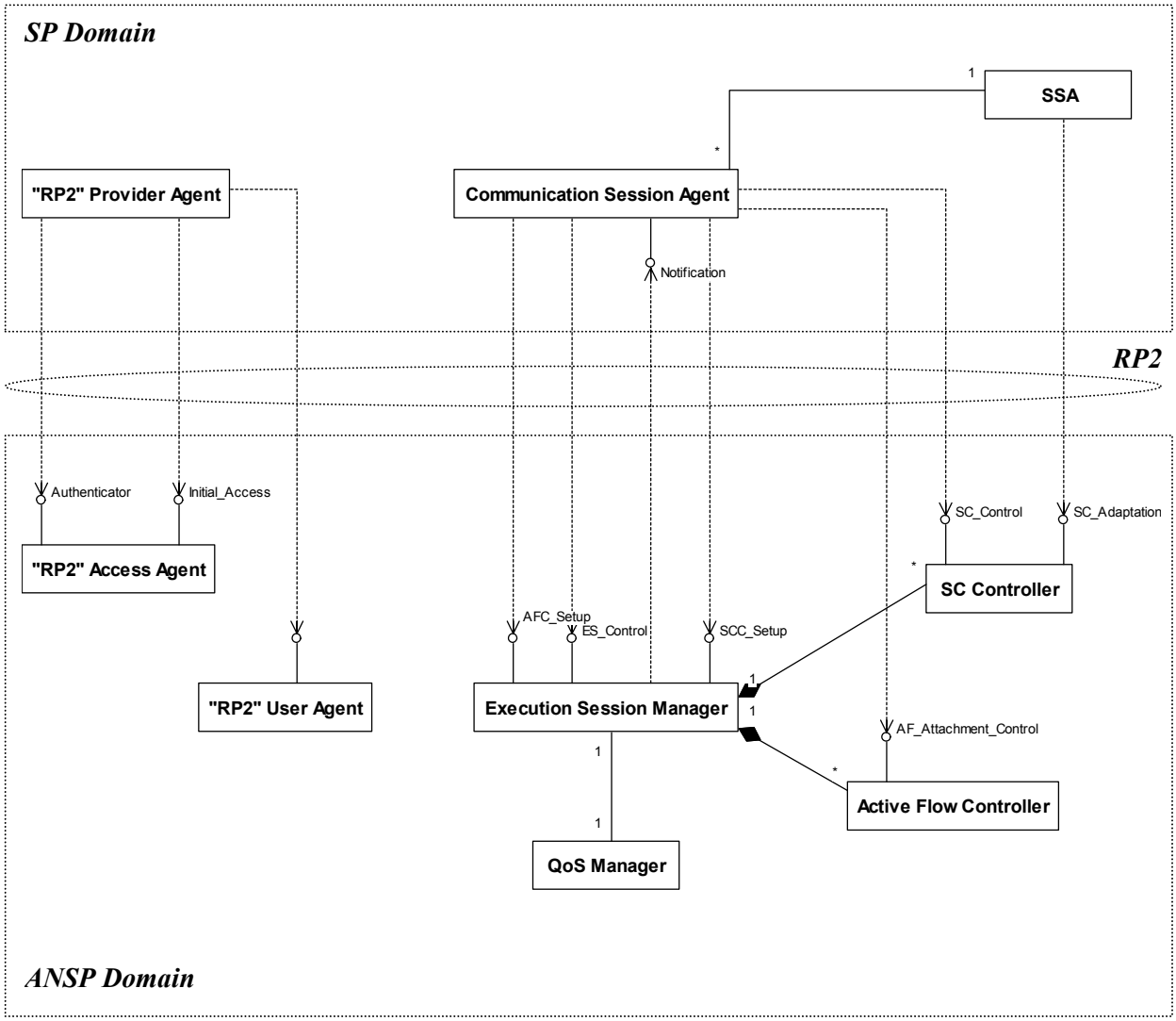


Figure 6-2. Computational Model for the reference point 2

6.2.2.2 Objects for Access Part

6.2.2.2.1 Access Agent Object

The Access Agent object becomes the point of access to the ANSP domain. The reference to the Access Agent shall be obtained by means of a location service such as a name service. Authenticating the SP identity and delivering appropriate interface references are the main responsibilities assigned to this computational object. It offers the Authenticator and Initial_Acces interfaces for these purposes.

[6] Refer to section 3 for a description of the relationship to TINA specifications.

6.2.2.2.1.1 Authenticator Interface

Through this interface the SP is authenticated, so that the ANSP can establish a secure and trusted relationship. Therefore this interface provides the functions to create the appropriate framework for subsequent contract negotiations. In the definition of the authentication interface we have adopted the schema defined in the CORBA Security Service specification [Ham00].

This interface contains the following operations:

`Get_supported_authen_methods`

This operation returns the current methods available in the ANSP domain to perform authentication using a given mechanism. (Definition imported from CORBA specifications)

`Authenticate`

This operation is called by the SP to take his identity to the ANSP with the aim of gaining rights to use the Initial_Access interface. The SP should select an authentication method from the list returned by the Get_supported_authen_methods. (Definition imported from CORBA specifications)

`Continue_authentication`

The method continue_authentication has been found useful since it is desirable to offer the possibility of mutual authentication7, through a challenge/response mechanism, between the SP and ANSP. (Definition imported from CORBA specifications)

6.2.2.2.1.2 Initial Access Interface

This interface can only be accessed by SPs who have been successfully authenticated. It consists of a single operation actually establishing the access session.

`Get_access_interface`

After calling this method, an access session between the SP and the ANSP is considered established. This operation returns a reference to a User_Agent_Access interface.

in: authentication credentials – the authentication credentials are obtained during the authentication phase.

out: user_agent_access_interface_reference – reference to the user_agent_access interface related to the established access session.

6.2.2.2.2 User Agent Object

This computational object allows directing the communication session life cycle. The establishment of a communication session requires a negotiation of the service capability types that are to be provided as well as a definition of acceptable service level margins.

There shall exist one user agent per communication session. Therefore, when several communication sessions pertain to the same SP, each of them will be associated to a corresponding User Agent. When the communication session is released, the associated User Agent and SP information is deleted.

This object offers the following interface:

6.2.2.2.2.1 User Agent Access Interface

The User_Agent_Access interface is used by the SP to manage the communication session. A reference to this interface, obtained from the Initial_Access interface, is used to set up, control and release a communication session.

This interface consists of the next operations:

`Establish_Communication_Session`

[7] Due to potential threats against the SP service agents running within the ANSP domain.

This operation is a means to open a communication session with a set of available service capability types. Whether the ANSP is able to provide the capability types directly or by using composition or federation services is out of the scope of the reference point 2. The security parameters obtained during the authentication phase should be used. When the communication session is established, the ANSP undertake to deliver the agreed service capability types when requested. It should be noted that this fact does not impose any constraint on the service level to be provided by the ANSP.

in: required_service_capability_type_list – list of service capability types required by the SP (e.g. types of execution environments –ANTS, Switchware, Alien,...-, distributed processing environments, security services). The list might include the type of active flow endpoints and attached services necessary along the complete communication session, even although they are not all required during the complete session lifetime.

out: status – informs about the operation result. Either success or failure is reported. The confirmation represents that the ANSP agrees to provide the service capability types and should be a means for avoiding contract repudiation. In this sense, it should act as a service requirements agreement. A failure would be reported if the ANSP is not able to provide any of the service capabilities requested in the service capability list.

out: channel_setup_interface_reference – reference to a channel set up interface that will help on the creation of an appropriate service path.

out: channel_control_interface_reference – reference to a channel control interface for reshaping the partial topology associated to the active service.

`Release_Communication_Session`

This operation is used to release an open communication session. All the execution sessions bound to the communication session should have been previously closed. The profiles containing SP information related to the considered communication session are deleted.

out: status – reports whether the operation succeeded or failed. Failures can arise when there exist execution sessions still opened.

6.2.2.2.3 Provider Agent Object

This object represents an ANSP within the SP domain. It is responsible for initialising and maintaining a communication session suitable for an active service. Thus, the necessary information regarding the concrete service the SP is offering to his customers should be passed to this computational entity8. Based on this information, the Provider Agent should be capable of deciding the service level to be demanded to the ANSP.

6.2.2.3 Objects For Usage Part

6.2.2.3.1 Communication Session Agent

The Communication Session Agent is responsible for starting the execution sessions along a service path. It is also in charge of setting up the partial topology of the active service within the active network (e.g. all border routers, some spanning tree, all active routers within the network, etc). Since topology configuration depends not only on SP requirements but also on ANSP service capabilities availability, a progressive process for realising the actual layout has been found convenient. The communication session agent is in charge of controlling this process. It also adapts the topology to the service needs by closing execution sessions when they are no longer needed or opening new sessions as required to enlarge the channels.

This computational object is also responsible for the establishment and control of active flows as well as for modifying the configuration of the virtual environment to adapt it to new requirements, e.g. by adding new active flow endpoints to a VE.

[8] This information could be obtained from the computational objects involved in the RP4 interactions.

6.2.2.3.1.1 Notification Interface

This interface is used to notify the asynchronous appearance of events. The events could warn about a temporary lack of service due to problems experienced within the ANSP domain. Such problems could lead to unexpected failures in the execution sessions. In this case, the Communication Session Agent should update the partial topology assigned to the active service and take corrective actions to restore the active channels.

This interface contains the following operation:

`Notify_Event`

Notifies that an event regarding the execution session performance has raised.

in: event_type – type of the event

in: execution_session_identifier – identifier corresponding to the execution session affected by the event.

6.2.2.3.2 Execution Session Manager

This object contains the interfaces that allow a SP to configure the service capability platform to adapt it to his needs. The environments created in this way may be composed of different services in distinct locations. It is noticeable that the same virtual environments distributed following different topologies could lead to a service path equally usable, i.e. complying the same topology requirements. Proceeding this way grants a freedom degree to the ANSP for adapting the topology to resources availability.

It also offers interfaces to manage the corresponding virtual environment. Controlling the virtual environment configuration is achieved by adding or suppressing service capabilities (e.g.: active flow endpoints) and binding or unbinding active flows to the active flow endpoints. For this purpose, two additional aiding computational objects are co-ordinated by the Execution Session Manager: the Service Capability Controller and the Active Flow Controller.

6.2.2.3.2.1 Execution Session Set-up interface

This interface includes operations for managing the execution session life cycle, as well as obtaining a clear vision of the partial topology associated to the active service. This interface is not intended to be a capability usage interface but an environment configuration interface.

The methods provided for these purposes are:

`Setup_Execution_Session`

This operation establishes an execution session within an active node. The SP does not specify a concrete active node but informs of the topology requirements for a given service path. The Execution Session Manager is responsible for selecting an active node complying those requirements (e.g. a node having the solicited resources).

in: service_capability_requirements – a list of the service capabilities that should be included in the virtual environment. The service level required for each of the capabilities is also provided. This is necessary since different execution sessions could have different service levels across a service path.

in: layout_requirements – inform to the Execution Session Manager about the new service topological requirements. Those requirements are used by the ANSP to determine the active nodes on which the execution sessions are to be opened.

out: status – reports the status of the execution session, either enabled or disabled due to an error.

out: execution_session_identifier – assigned by the ANSP to identify an opened execution session.

out: current_topology – updated information about the current topology, i.e. the distribution of the execution sessions within the active network.

6.2.2.3.2.2 Service Path Control Interface

`List_All_Execution_Sessions`

Returns the list of execution sessions forming a service path. Neither a service session identifier nor a service path identifier are necessary in this case. This operation may be helpful to recover a topological view after failure conditions are notified by the Execution Session Manager or arise in the Communication Session Agent.

out: execution_session_list –list of execution session identifiers corresponding to the virtual environments attached to the service path being controlled by the Execution Session Manager.

`Get_Execution_Session_Control_Interface`

This method returns a reference to an execution_session_control interface for a previously opened execution session. An exception should raise if the identifier does not correspond to an opened execution session.

in: execution_session_identifier – identifies an execution session. This identifier is obtained when calling the Setup_Execution_Session function.

out: execution_session_control_interface_reference – a reference to an interface for managing the execution session.

`Get_SCC_Setup_Interface`

This method returns a reference to a service_capability_controller_setup interface, allowing the creation of service capability controllers. Thus, the Execution Session Manager becomes the factory for this type of objects.

in: execution_session_identifier – identifies an execution session. This identifier is obtained when calling the Setup_Execution_Session function.

out: service_capability_controller_setup_interface_reference – a reference to the Service Capability Controller set up interface used to create new Service Capability Controllers.

`Release_Execution_Session`

This operation asks for the deletion of the execution session when it is no longer needed. Since the Execution Session Manager is associated with several execution sessions, providing the ES identifier is required. This operation enables the dynamic reconfiguration of the topology to adjust it to changes in service requirements.

in: execution_session_identifier – identifies the execution session to be released.

out: status – report whether it was possible to close the execution session or not. In the later case, a failure code is provided.

6.2.2.3.2.3 Execution Session Control Interface

The Execution_Session_Control interface provides operations allowing the SP to order execution session state changes. Two states are distinguished: enabled and disabled. Whenever the execution session is in disabled state, it is possible to rearrange the service capabilities as required. This means that it is possible to set up new service capabilities on the virtual environment or to destroy existing ones.

On the contrary, when the execution session is enabled, the service capabilities and associated resources are freezed so no change is allowed on them9. However, in either case the resources associated to the execution session do not change but are reorganised among the remaining service capabilities.

`Resume_Execution_Session`

[9] Changes on service capabilities could be requested by service agents, so this is a way to set periods of acceptance or rejection on them.

Enables an execution session controlled by this execution session manager.

in: execution_session_identifier – identifies the execution session to be enabled.

out: status – a flag showing whether the operation succeed or failed.

`Suspend_Execution_Session`

Disables an execution session controlled by this execution session manager.

in: execution_session_identifier – identifies the execution session to be disabled.

out: status – a flag showing whether the operation succeed or failed.

`List_Service_Capabilities`

Provides the list of references that point to the service capabilities included in the virtual environment corresponding to an execution session.

in: execution_session_identifier – identifies the execution session.

out: service_capability_reference_list – list of service capability types and their references.

`Get_Active_Channel_Setup_Interface`

This method returns a reference to an interface for establishing active channels between active flow endpoints.

out: active_channel_setup_interface_reference

6.2.2.3.2.4 Active Channel Set up Interface

Through this interface the SP is allowed to bind an active channel to an active flow endpoint. The execution session manager is responsible for checking that the considered virtual environment is able to offer the service capabilities required by the active packets carried along the active flow. For example, the active flow endpoint must be capable of launching the active code execution. The execution session manager could reject the request if there would be not enough resources to assign to the new active channel.

`Setup_Active_Channel`

Establishes an active channel bound to an active flow endpoint which is the only essential service capability to be given to active flows. Other service capabilities could be requested for the active flow.

in: service_capability_list – list of service capabilities required and resources necessary for the active flow.

out: active_flow_endpoint_identifier – identifies the endpoint for the active channel. It is only valid if the operation succeed as shown by the status parameter.

out: authorisation_credentials – authorise the use of active capabilities.

out: status – its value shows whether the operation succeeded or failed due to resource unavailability.

`Get_Active_Channel_Attachment_Control_Interface`

This operation returns a reference to an interface for controlling the attachment of the active channel.

out: active_channel_attachment_control_interface_reference

6.2.2.3.3 Active Flow Controller

A computational object of this type will exist per each created active channel. The controller represents the entailment of an active flow with an active flow endpoint owned by the virtual environment associated to the execution session opened by the SP.

It provides functions for managing the associated active channel, including its modification, engagement, disengagement, and release. When an active channel is released, the corresponding controller is deleted.

6.2.2.3.3.1 Active Channel Attachment Control Interface

Since several active flows could be linked to the same execution session, although they could have different purposes, it is necessary to be able to control each of the flows separately. For this reason the active channel attachment control interface has been bound to the active flow controller object.

This interface contains the following methods:

`Reengage_Active_Channel`

Only when an active channel is engaged to the active flow endpoint, the active packets coming from the active flow receive service capabilities.

in: authorisation_credentials – authorise the operations on service capabilities.

out: status – flag showing whether it was possible or not to activate the active flow.

`Disengage_Active_Channel`

When an active channel is disengaged, the active packets coming from the corresponding active flows only receive connectivity services, being handled as non-active packets during that period.

in: authorisation_credentials – authorise the operations on service capabilities.

in: timeout – allows the SP to specify the period of time in which the active channel will be disengaged. When the period has expired, the active channel is automatically re-engaged. A value of 0 means the active flow should remain disengaged until the SP explicitly calls the operation Reengage_Active_Channel.

out: status – flag showing whether it was possible or not to deactivate the active flow.

`Release_Active_Channel`

This operation orders the deletion of the active channel and associated profiles. The resources reserved for the active flow are recovered by the execution session manager so that they can be assigned to new active flows.

out: status – flag showing whether it was possible to release the active channel. The active channel should be disengaged to allow deletion.

`Rebind_Active_Channel`

This operation asks for a change in the features of the service capabilities offered to the active flow. The Active Flow Controller and execution session manager will check if there are enough resources to allow the new configuration.

in: authorisation_credentials – authorise the operations on service capabilities.

in: service_capability_list – list of new service capabilities required. The active flow endpoint must remain the same.

out: acknowledgement – acknowledges the redistribution of resources.

6.2.2.3.3.2 Service Capability Controller Set-up Interface

This interface is accessed by the Communication Session Agent to create new service capabilities that are to be used in the corresponding virtual environment and to obtain references to the SCC control interfaces

`Create_Service_Capability_Controller`

It is used for creating a service capability controller with the characteristics and requirements demanded by the SP. A set of resources is attached to the service capability.

in: service_capability_type – type of the service capability required (e.g. an active flow endpoint, or a DPE-like facility)

in: service_capability_requirements – set of resources that should be at the service capability disposal.

out: service_capability_identifier – identifies the service capability.

out: service_capability_control_interface_reference – reference to the service capability control interface.

out: service_capability_adaptation_interface_reference - reference to the service capability adaptation interface.

out: acknowledgement – shows whether the set up succeeded or failed.

6.2.2.3.4 Service Capability Controller

One object of this type is linked to each created service capability and remains until the service capability is no longer needed and therefore can be deleted. It offers two interfaces that allow the SP to dynamically adapt the service capability features to the service needs.

A set of interfaces related to specific usage features should be described for each service capability.

6.2.2.3.4.1 Service Capability Control Interface

The communication session agent will use this interface to obtain the interfaces that are specific to this service capability, being then delivered to the service session agent if necessary.

`Get_Status_Information`

This operation queries for information about the status of the service capability. It could be used for monitoring purposes.

in: authorisation_credentials - authorise operations on service capabilities.

in: information_type – identifies the type of the information required by the SP.

out: status_information – returned information about the service capability.

`Release_Service_Capability`

Deletes the information associated with the service capability. The execution session manager must redistribute the resources among existent service capabilities.

in: authorisation_credentials –authorise the use of the service capability

out: acknowledgement – confirms that the service capability has been correctly deleted.

`Get_Proprietary_Interfaces`

in: authorisation_credentials – authorise the use of the service capability

out: proprietary_interface_list – list of interfaces that are specific to this object.

6.2.2.3.4.2 Service Capability Adaptation Interface

This interface is used to modify the resources given to a service capability for its operation. The existent resources attached to a virtual environment are distributed among the different service capabilities.

`Allocate_Resources`

Adds new resources to a service capabilitiy.

in: resource_requirements – the new list of required resources.

in: authorisation_credentials – authorise to operate on service capabilities.

out: acknowledgement – informs whether it was possible to provide the resources to the service capability.

`Release_Resources`

Release a set of resources associated to this service capability.

in: resource_list – specifies the set of resources to be released.

out: acknowledgement – confirms the release of the resources specified.

6.2.2.3.5 QoS Manager

The QoS Manager component offers network Quality of Service (QoS) management functionality, allowing for the provision of services in accordance with the user's QoS requirements. The provided interfaces allow other SP components to obtain useful network performance information as well as to manipulate the network in order to achieve the required QoS levels. The direct benefit from the QoS Manager component for the SP is that it focuses on ensuring that the service is delivered according to the user defined network QoS preferences at all times.

If an established network connection cannot satisfy the user's QoS requirements, the QoS Manager is responsible to inform the system and take appropriate action. This involves either re-configuration of the connection or the establishment of a new connection using the network of a suitable, alternative network infrastructure provider.

6.2.2.3.5.1 Functionality

Regarding the QoS the following information is required:

- Level of Quality of Service required. (The DiffServ service classes are assumed to be Expedited Forwarding, Assured Forwarding and Default (best-effort)).
- Performance parameters to be monitored.

If the service is adaptable to network conditions, then appropriate thresholds for the monitored performance parameters should be provided as well. These performance thresholds allow the Profiler to adapt service characteristics to the network conditions, and provide the QoS Manager with a set of criteria to notify the Profiler of important changes in the network conditions that can affect the service. The following functions are covered:

6.2.2.3.5.1.1 Performance Related Operations:

The user has the capability to modify the required QoS during a session. The QoS Manager must therefore provide the following functionality to other components:

- Modify the QoS level required by the service dynamically (e.g. Gold, Best Effort etc.).
- Add a performance parameter to be monitored.
- Delete a monitored performance parameter.
- Modify the thresholds of a monitored performance parameter

6.2.2.3.5.1.2 Generation of Reports and Notifications:

The QoS Manager is required to generate reports and notifications that are key to the provision of the service. As mentioned earlier there is a need to notify the Adaptation Manager if a network performance threshold has been 'crossed' in the network. As an example, the following figure shows some desired levels for the delay performance parameter. The values of 50ms and 100ms are service-related thresholds for the delay in the network. The QoS Manager must notify the Adaptation Manager whenever the delay parameter crosses either 50ms or 100ms to allow the service to adapt to the performance of the network. Another important point is that the QoS Manager must also send a notification if the Service Level Agreement (SLA), which is represented by the 150ms boundary, has not been honoured.

If the network is no longer able to deliver the QoS required during a session then the QoS Manager may need to inform the user that a switch to another network may be necessary. At the end of every session, the QoS Manager may also provide a performance report to the Account Manager reflecting on the cost of using the network.

6.2.2.3.6 Service Session Agent

The Service Session Agent computational object represents the active service logic accessing the ANSP domain. It is also the entity that uses the service capabilities. A set of authorisation credentials must be delivered by the Service Session Agent to gain access to such usage.

6.2.2.4 The Dynamic Behaviour

6.2.2.4.1 Service Session Establishment

Once the Provider Agent has obtained a reference to the Access Agent (e.g. using a location mechanism such as a name service), the first step in order for the SP to deploy active services in the active network is establishing a communication session. Since service level negotiations require a secure interaction as well as non-repudiation facilities, an authentication phase must be carried out before actually initiating the establishment process.

After mutual identity authentication, the Provider Agent is allowed to request a reference to a User Agent. The Access Agent is responsible for creating such User Agent that will be associated exclusively to the Provider Agent during the session lifetime. The reference will be provided only upon successful validation of the authentication credentials carried in the access request.

Next, the Provider Agent asks for the establishment of a communication session providing the list of service capabilities that could be requested throughout the session lifetime. The User Agent main responsibility is to verify and confirm that the ANSP is able to provide the solicited service capabilities[10]. By returning a reference to the Execution Session Manager interfaces, the ANSP implicitly commits himself to grant the service capabilities on request. In this sense, it could be considered as a Service Requirements Agreement.

[10] Composition or Federation might be used to offer the service capabilities.

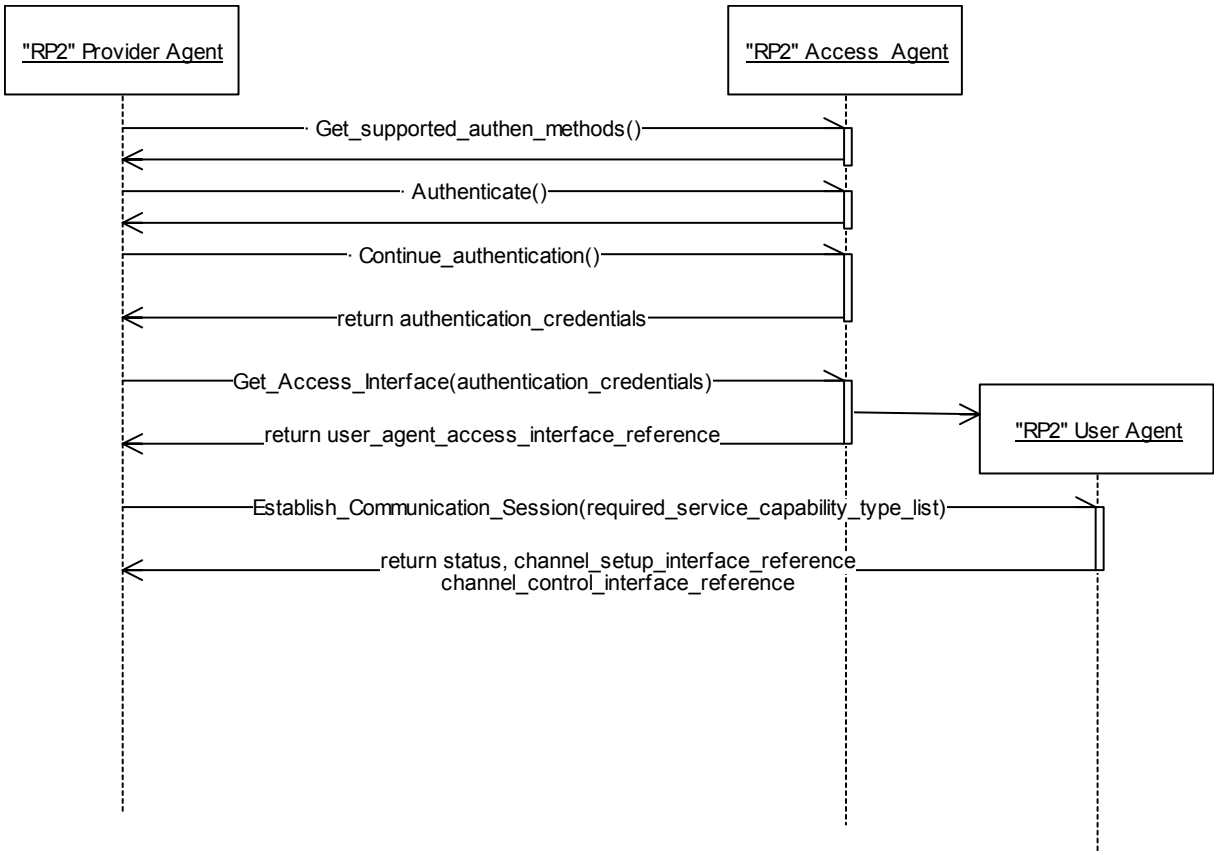


Figure 6-3 Establishment of a service session.

6.2.2.4.2 Partial Topology Creation

Once the communication session is established, the following step lies in opening execution sessions as necessary to support a service path. The Communication Session Agent is responsible for ordering the execution sessions set up. It provides a set of requirements to the Execution Session Manager, which takes the decision on the actual virtual environment location. The current service path topology is analysed by the Communication Session Agent, based on the information returned by the Execution Session Manager, in order to update the layout requirements. When the Communication Session Agent realises that the layout requirements are fulfilled –i.e. the service capability platform is configured- it stops opening execution sessions.

The Execution Session Manager creates a Virtual Environment for each opened execution session, which identifier is returned to the CSA. This way the Communication Session Agent has a complete view of the topology, being able to act on each individual virtual environment.

It should be noticed that the service capability requirements contain not only the list of initial service capabilities to be included in the virtual environment but also the service level required from each one. The Execution Session Manager will be in charge of checking resource availability within the network and selecting an appropriate node to set up the virtual environment.

The Communication Session Agent is able to adapt the topology to his needs by opening new execution sessions or removing existing ones.

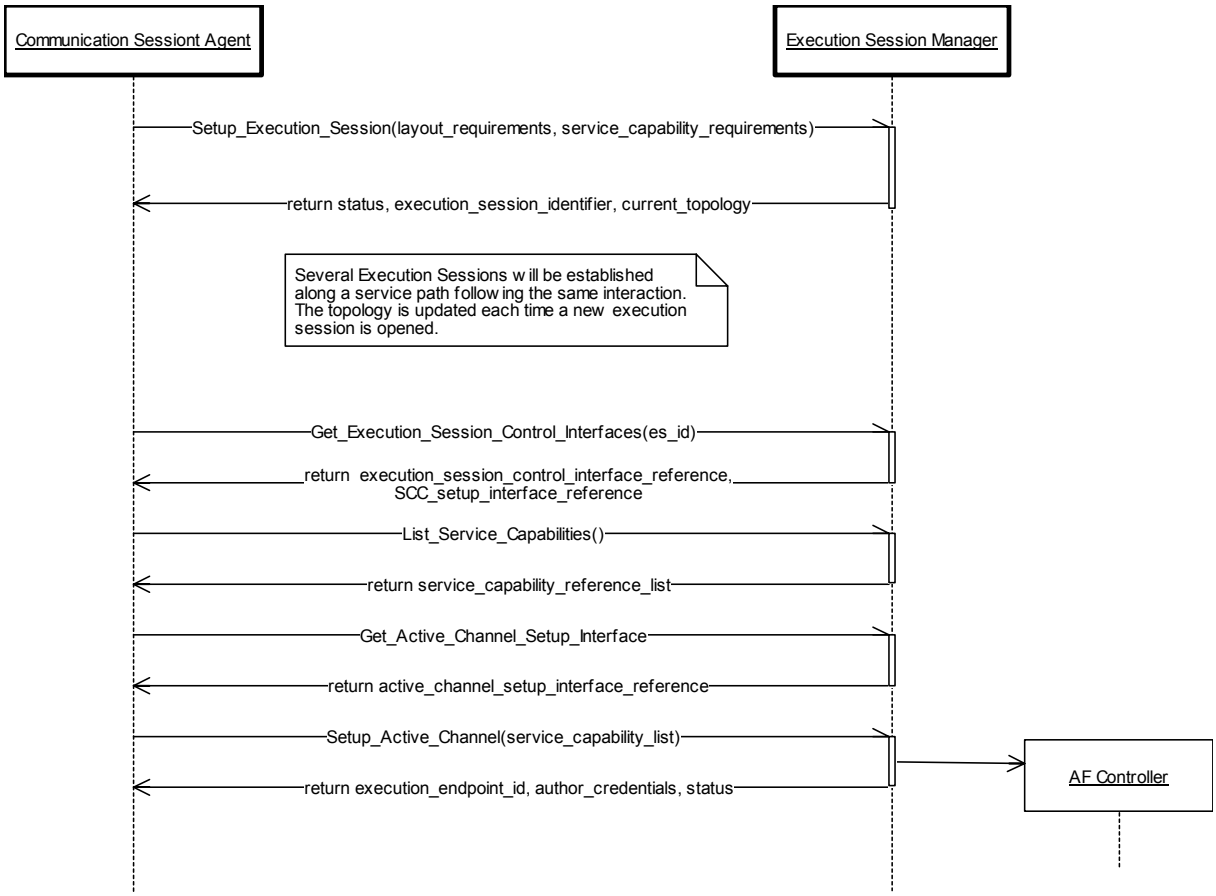


Figure 6-4. Creation of a service path

Once the execution session has been set up, the communication session agent requests to the execution session manager a list of references to the service capabilities forming a virtual environment in order to be able to control them.

Although this action completes the basic service path infrastructure, further configuration is necessary before the Service Provider can actually install his services in the network. The following section describes the required steps.

6.2.2.4.3 Virtual Environment Configuration

After the virtual environment has been set up, additional configuration tasks could be carried out in order to dynamically adapt the VE to the changing Service Provider requirements.

Also the appropriate active channels have to be opened so that active components can be deployed and run within the virtual environment. To manage to do it, the Communication Session Agent calls Setup_Active_Flow_Channel delivering the service capability list parameter, which informs to the Execution Session Manager of the service capabilities that should be attached to the active flow. The active packets coming from the considered active flow will therefore receive, at most, those service capabilities. The active flow endpoint identifier provided by the Execution Session Manager will be useful to identify the active flow an active packet belongs to and therefore dispatching it to the appropriate execution environment.

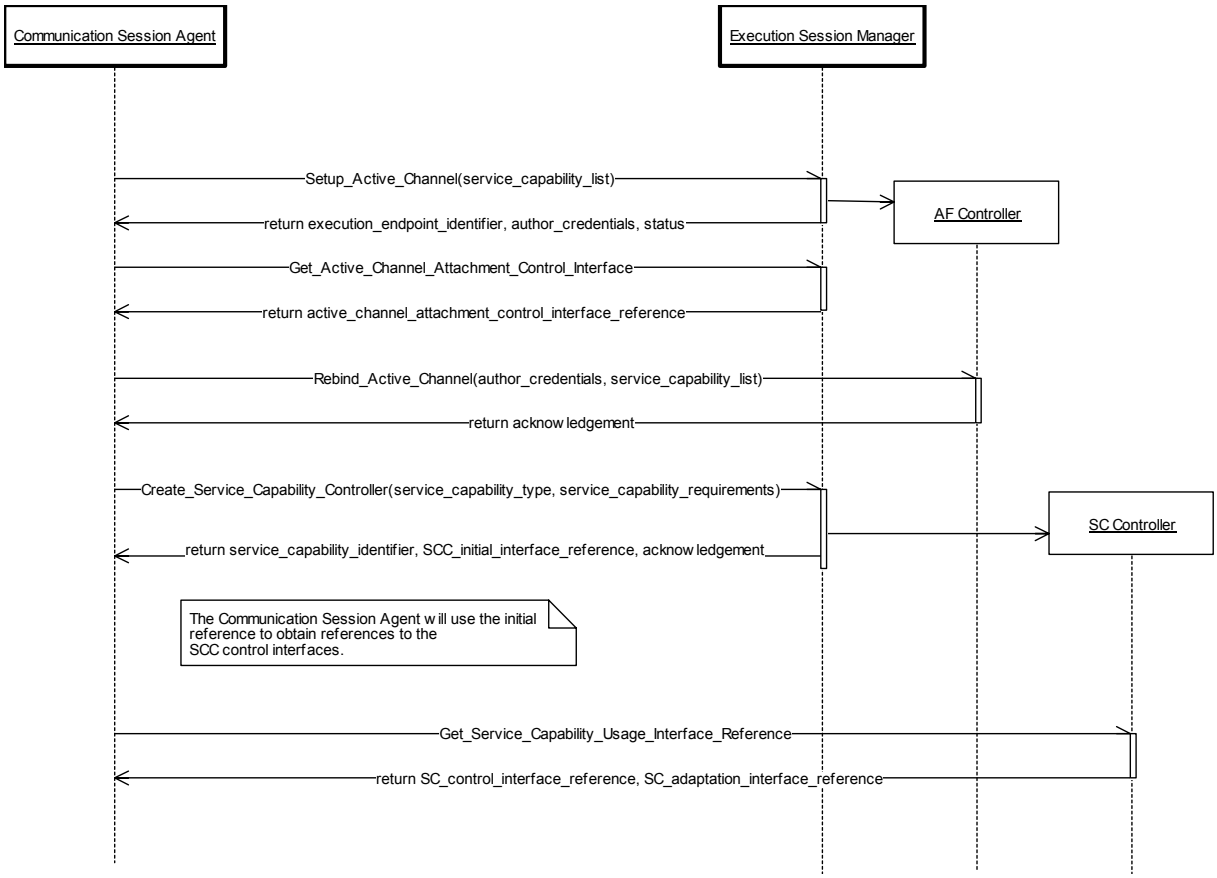


Figure 6-5. Configuration of the Virtual Environment

To manage each active channel, an Active Flow Controller is created by the Execution Session Manager (one per active flow). This computational object corresponds to the Flow Connection Controller entity as defined in [TIN97]. The Communication Session Agent subsequently obtains the reference to the Active Flow Controller interfaces.

As explained formerly, the Execution Session Manager is even able to request a change in the resources associated to the active channel by issuing Rebind_Active_Channel and presenting a set of authorisation credentials. If any of the service capabilities now required by the active packets was not reserved during the virtual environment set up, the Communication Session Agent is allowed to demand the inclusion of new capabilities.

Each service capability is controlled by an associated Service Capability Controller. The Execution Session Manager, acting as a factory object will be used to dynamically create them under demand by calling Get_Service_Capability_Controller. The returned service capability identifier will be used to retrieve the references to the SCC interfaces whenever necessary. This is a mechanism flexible enough to be used by service session agents.

6.2.2.4.4 Virtual Environment Usage

Accessing service capability interfaces requires obtaining a reference to them by calling Get_Proprietary_Interfaces[11]. The authorisation credentials will inform to the SCC about the maximum service level that could be granted to the SSA. As a security measure, all the operations on the SCC require presenting those credentials.

[11] When sending the Service Agent, the Execution Session Manager provides the references to the SCC interfaces.

The service session agent maintains references to the Service Capability Controllers, which are used to obtain the rest of the control interfaces. To perform its functionalities, the service session agent can obtain information on the service capability status by calling Get_Status_Information as well as allocate resources when necessary. Nevertheless, in the last case the SC Controller may reject the request if there are not enough resources available to satisfy it.

Finally, the Service Agent will be able to use the service capabilities through the previously obtained proprietary interfaces (e.g. accessing the security services).

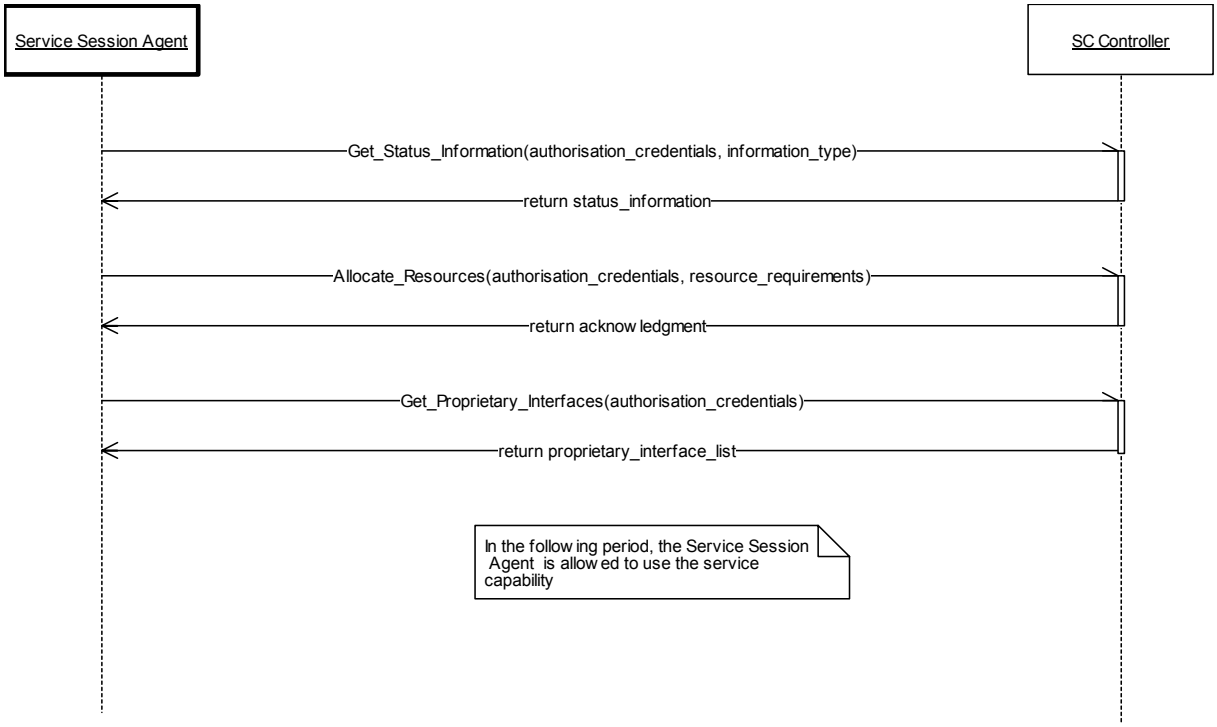


Figure 6-6. Usage of the VE service capabilities

6.2.2.4.5 Service Session Release

The process of releasing a service session can be initiated by the Provider Agent when an active service is no longer provided. In this case, the Provider Agent will proceed to orderly release the resources attached to the computational objects associated to a particular service path.

Firstly, the active channels and service capabilities will be disabled and released. Next, the execution sessions will be closed, causing the associated Execution Session Manager to be destroyed. When all the execution sessions are removed, it is possible to release the communication session. The Communication Session Agent will notify this fact.

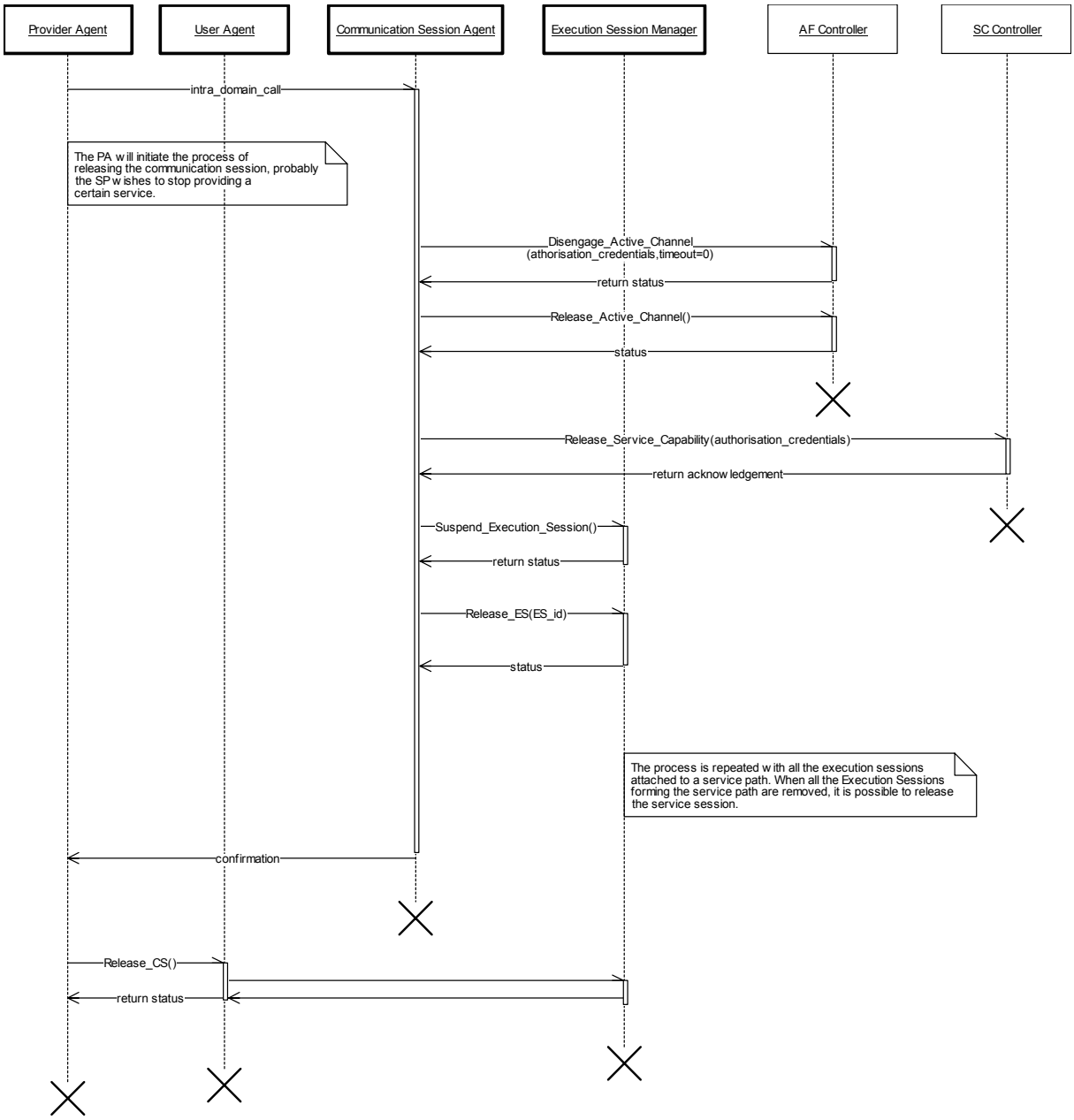


Figure 6-7. Release of a Service Session

The Provider Agent will then ask for the communication session release to the User Agent, which will cause the destruction of any remaining ANSP computational objects or profiles associated to the considered session.

6.2.3 References

[Den00] S. Denazis, “Active Node Reference Architecture”, FAIN Internal Report, December 2000.

[FAIN00] “ Initial Active Networks Enterprise Model”, FAIN Work Package 2 Members, January 2000.

[TIN97] “The ConS Reference Point. Version 1.0”, TINA 1.0 Specification, February 1997.

[TIN96] “Computational Modelling Concepts. Version 3.2”, TINA-C Deliverable, May 1996.

[OMG00] “Security Services Specification”, Object Management Group Inc., May 2000.

[Ham00] T. Hamada, "A Framework for Connectivity Service Delivery Process", TINA-IPCM Work Group RFP, April 2000
http://www.tinac.com/specifications/documents/ConSv10.doc

6.3 Reference Point RP3 (Active Network Service Provider – Network Infrastructure Provider)

RP3 defines the interface between ANSP and NIP, i.e. between the following roles:

- **Network Infrastructure Provider (NIP):** A NIP provides managed network resources (bandwidth, memory and processing power) to Network Service Providers. It offers an ANN-based platform to the Network Service Providers who can build their own Execution Environments, and proffers basic transmission infrastructure which may be based upon traditional transmission technology as well as emerging ones (both wired and wireless).
  Since FAIN focuses on Internet technology, a basic network infrastructure consists on the following characteristics: presence of a physical network; IPv4 and/or IPv6 forwarding mechanism; default IP routing protocol; if RP 7 is implemented: peering service to other NIPs or ISPs; management interface.
- **Active Network Service Provider (ANSP):** An ANSP provides network services and Execution Environments (one or more) for active service components to Service Providers (e.g., it features component provisioning, security, and management capabilities). Together with the Network Infrastructure Provider, it forms the communications infrastructure.

In contrast to RP2, where the requirements can be derived from the applications, the definition of RP3 is mainly a design decision. In WP2, we have decided to postpone this decision to a later stage in the project. This is not only for reasons of resource limitations within WP2, but also due to the fact that in our opinion the proper definition of RP3 is closely related to deployment scenarios of AN technology by operators. Instead, we have decided to show up the design space we have for this interface. We see mainly the following options:

Layer 2 vs. Layer 3 service offered by NIP: With respect to transmission resources, NIP may provide link layer (Layer 2) functionality (ATM, Ethernet, Wireless ...) or it may provide IP connectivity.

In the case of the NIP providing Layer 2 functionality, the NIP basically provides a managed transport network, while all IP functionality is realised by the ANSP. Note that we are only interested in the functionality the NIP offers, not in the way it is implemented. For instance, NIP may also operate an IP network and provide L2TP tunnels between the active nodes.

In the other case, the NIP operates an IP network as it is done by today's ISPs, and provides IP connectivity services of various kinds (see below) to the ANSP. The operation of an IP network usually involves the provisioning of IPv4 or IPv6 forwarding capabilities, routing protocols, peering services with other ISPs as well as management operations for the individual nodes and the whole network.

The implications of these choices are:

- In case NIP provides only Layer 2 capabilities, the ANSP (and in turn the SP) may exploit fully the characteristics of the underlying transport network. This may be beneficial if services are highly dependent on such characteristics – which are "hidden" by IP - such as bandwidth, delay, latency. An example may be terrestrial (GSM, UMTS...) or satellite wireless links. This option may also be beneficial in fixed networks in order to improve the basic IP service as such, for instance for traffic engineering purposes along the lines of MPLS.
- In case the NIP provides IP connectivity, the NIP can be an existing ISP. While the skills of an ISP are in the operation of IP networks, the new role ANSP can concentrate on providing higher level active network services on top of an IP network. This may be easier for application oriented services.

In the context of the applications investigated within FAIN, all are more oriented towards application services. For instance, Active Web Service (see section 4.1) fully operates on the HTTP layer or above. For this reason, we decide for the second choice at the moment. We also favour this choice, since otherwise in FAIN within the ANSP role we would have to reinvent the necessary functionality to operate an ordinary IP network (e.g. execution of routing protocols), which is clearly undesirable.

Basic IP services offered: Besides best effort IPv4 forwarding, the NIP may also provide more advanced network services such as IPv6, Integrated Services (IntServ), Differentiated Services (Diffserv) or even Multi Protocol Label Switching (MPLS). However, IPv4 is currently the only service which is available on a global scale and it is difficult to predict how fast and broad these more advanced technologies will be deployed. Consider for instance IPv6, which is standardised and available within products since several years, while still there is no pressure to deploy it in the field.

Active nodes owned by NIP or ANSP: Active nodes are physical components, which may be owned by the ANSP or by the NIP. The "ownership" relation is important, since ownership usually implies physical location as well as some management responsibilities.

In case that the active nodes are owned solely by the ANSP, the NIP is not aware of any "activeness" of the network. In this case, RP3 specifies what functionality the FAIN system (or the ANSP) expects from a network infrastructure provider to be operational. However, it prohibits a scenario where a traditional network operator can deploy active nodes without providing the full functionality of an ANSP, unless additional reference points are specified. However, new market entrants can act as an ANSP, but without fully exploiting the benefits of being physically located within the network – they only can provide an overlay network, which is connected to the NIP like any other network.

In case that the active nodes are owned by the NIP, we have to further investigate the kind of active functionality they provide and the kind of management operations they perform:

- *NIP provides raw hardware:* In this case, the NIP sells the physical co-location of active nodes within its network, but leaves the operation of these nodes to one ANSP. Access to these nodes is possible via a lower-layer node interface. All issues of resource management, isolation... are dealt with at the ANSP level. This approach may be advantageous if a operator who owns a physical network (e.g. a ex-PNO) may want to make money out of this fact, but who would like to leave the operation of the Active part of the network to a separate legal entity.
- *NIP provides virtual hardware:* In this case, the hardware is virtualised on a lower-layer interface in order to be able to serve several ANSPs. This involves advanced support for resource management, isolation of the different ANSPs, security etc. at the NIP level. This approach may for instance be advantageous if the owner of a physical network has to (or would like to) open the physical network to a small number of competitors (i.e. ANSPs).
- *NIP provides higher-level interface* (e.g. VEs). In this case, the NIP provides a higher-level interface (e.g. Virtual Environments) to the ANSP.

Network wide vs. Node Management: Regarding the management of active nodes, the NIP may provide management operation on a per-node or on a network wide level.

To sum up, we have a variety of different choices regarding the specification of the NIP/ANSP interface. These different choices enable different options and strategies for the deployment of Active Network technology by operators and service providers. Since these are not yet clear, we consider it as too early to precisely specify the reference point in its full beauty. We will rather investigate the above-mentioned options in more detail during the remainder of project FAIN and to decide on a small number (may be one) of scenarios later on. It is expected that the implementation of the FAIN, which is structured around clearly defined interfaces anyhow (see [IPBNM]), can easily be restructured to make these later defined reference points explicit.

The only decision we make in the moment is that the NIP offers Layer 3 (IP) connectivity services to the ANSP, which has been justified above.

[IPBNM] FAIN project, Workpacke 4: Initial Policy Based Management. FAIN project, internal report R11 & R12, FAIN consortium, 2001.

6.4 Reference Point RP4 (Consumer-Service Provider)

6.4.1 Information Model

6.4.1.1 Definition

The RP4 information model describes two categories of information objects: 1) roles derived from the roles of people and organisations within the business model, and 2) services. The business model identifies three different roles*: service providers*, *subscribers* and *users*. A service provider enables subscribers to make use of different services.

The service description differentiates between two types: the *service template* type and *user service profile* type. Each service is described by a *service description* that is divided into two parts: *a service template* and *user service profile*. The service template defines how the service is to be administered including information about the start of service sessions and service user interfaces, the required runtime environment as well as general authorisation rights. Also included is configuration information and information on the available tariff schemes.

In addition to the service template, there is also the *user service profile*. This information object defines service-specific attributes that can be manipulated by a user while the service is running. These attributes are interpreted solely by the service.

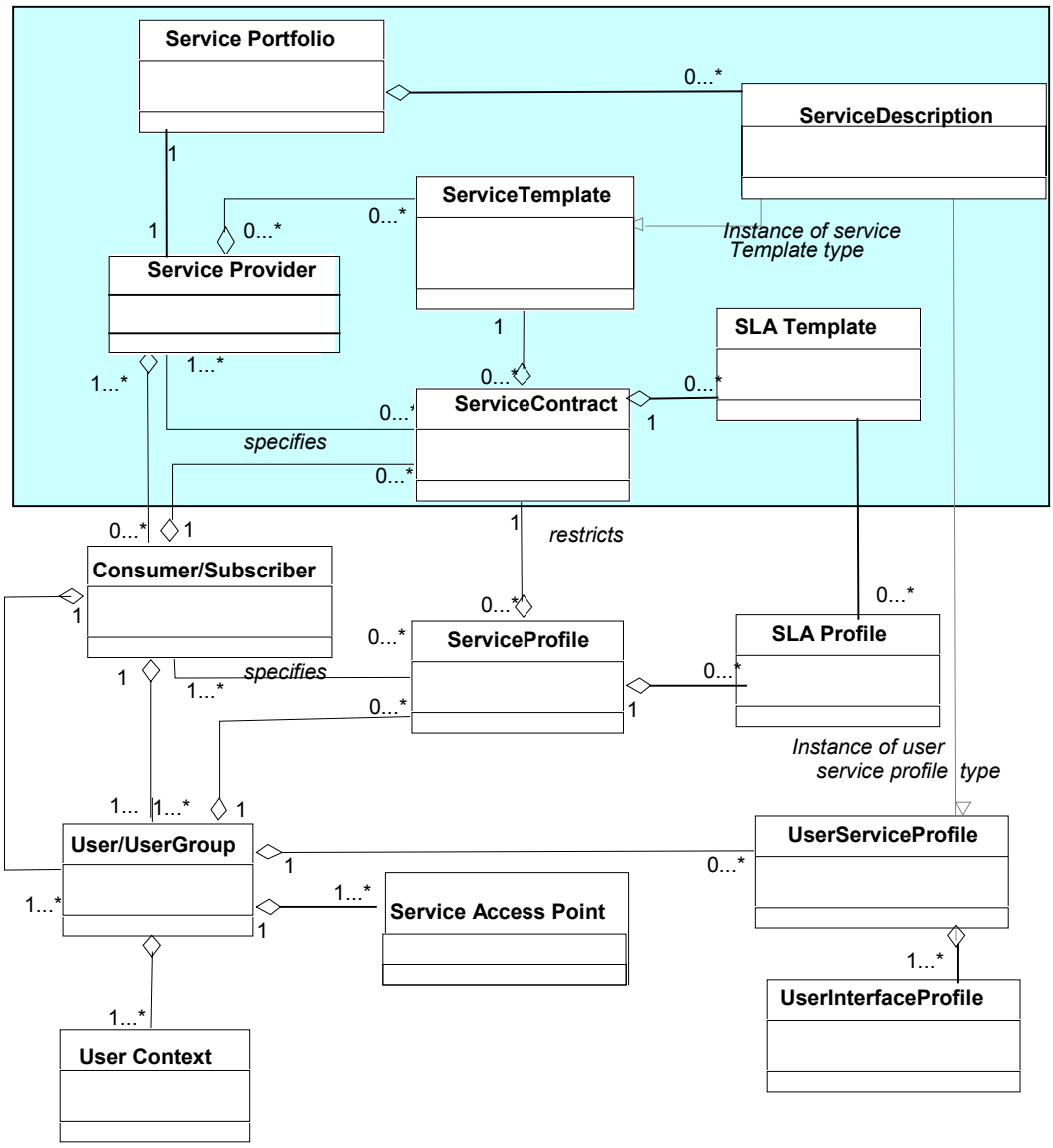


Figure 6-8 RP4 Information Model

6.4.1.2 Description

Descriptions of the individual objects are given below:

Service Portfolio: This represents the collection of the separate services offered by this Service Provider.

Service Description: Represents the service specification produced by standardisation bodies, enterprises or service providers.

Service Template: Described the generic information and behavioural characteristics of a service as offered by a *Service Provider* according to a *Service Description*.

Service provider: enables subscribers to make use of different services.

Service Contract: This represents the agreement between the Customer and Service Provider to use and pay for the Service. It represents the tailoring of a *Service Template* to the specific requirement and needs of a consumer to be provided by the Service Provider. It holds contract-specific information such as the time of service provision, the names of contact people for technical and financial issues, the tariff for the subscription and a link to a corresponding service description. The service contract describes the service characteristics for a specified customer and restricts the corresponding service template. It describes the service-specific part of the contract. Its service settings apply to any of a customer's users.

SLA Template: This defines the information structures and value ranges within which SLAs may be established for the use of the service defined by the related Service Contract.

Service Profile: This is a constrained usage of the service defined in a Service Contract associated with the containing Subscription. It may consist of a specific SLA configuration selected from within the restrictions of the SLA Template.

SLA Profile: This is an SLA that applies to the containing Service Profile.

Consumer/Subscriber: In our case, the Consumer role defined in the business model may include the role of customer, subscriber or end-user. It may contain information about a customer such as name, address, payment method and bank-account transfer data. Customer of the system may be people, institutions or other service providers. A consumer may subscribe to several services from a provider. If a customer wishes to specify different user rights and service characteristics for different users, then subscription Groups can be defined.

Service Access Point (SAP): This represents a point administered by the Service Provider from which End-Users/SAG are able to access the Service.

User/UserGroup: each user of a service is represented by an object containing information about the user, for example the user-ID and user name. The user object mainly aggregates the information objects *user context* and *user service profile*. A group of End-Users that may use the service and/or a group of SAPs from which they may access the service. The user, usergroup and service profiles are customer-specific information objects. UserGroup is assigned a service profile object that correspondingly further restricts the service contract settings. A UserGroup may have a SAP filter setting and an End-User filter setting both of which can be positive or negative with the following effects on the term (User filter, SAP filter):

Case 1: (+,+): Only End-Users in the SAG may use an associated Service Profile and only from the included SAPs. If no SAPs are included none of the End-Users can use the Service Profile.

Case 2: (+,-): Only End-Users in the SAP may use an associated Service Profile but not from the included SAPs. If no SAPs are included the End-Users can use the Service Profile from any of the Customer's SAP.

Case 3: (-,+): End-Users in the SAP may not use an associated Service Profile and only from the included SAPs. All other registered End-Users may use the Service Profile from the included SAP. If no SAPs are included none of the End-Users can use the Service Profile. If no End-Users are included than any End-User may use the Service Profile from any of the included SAPs.

Case 4: (-,-): End-Users in the SAP may not use an associated Service Profile All other registered End-Users may use the Service Profile, but not from the included SAP. If no SAPs are included all registered End-Users except those in the Group can user the Service Profile from any register SAP not in the group.

User Context: For each access session, there will be a specific user context for the user, and it will constrain the invocation of service sessions within that access session. The usage context specifies the configuration of network and terminal equipment in the user domain.

User service profile: specifies the preferences set by the user for service execution. A user service profile may refer to security and accounting preferences and may be constrained by one or more user contexts. A user service profile is always service-specific so there must be one for each service the user is allowed to use.

User service profile: specifies the preferences for the user interfaces

6.4.2 Computational Model

6.4.2.1 Introduction

The computational model corresponding to RP4 may be decomposed into an access, session, subscription, accounting, and profiler components

With respect to an access session, the information describing the user domain is separated from the one describing the provider domain. The concepts of user domain access and provider domain access are reflected in the model, in order to clearly separate the responsibilities and the information contents of the two areas. An access session is established when the provider domain access session and one user domain access session interact. A user must have an agreement (subscription contract or unilateral declaration for anonymous access) with a provider, in order that an access session can be established between a provider domain access session and a user domain access session. A user is related to zero or more user domain access session objects by an owning/controlling association. A provide is related to zero or more provider domain access session objects by an owning/controlling association. Each access session is responsible of controlling the lifecycle of a number of service sessions.

The mapping of the information model results in the access session related service components in the computational viewpoint, which are the Provider Agent, the Access Agent and the User Agent. The Session Manager SM is at the border between access session and service session. The access session components also have relations with SSA, the session segment of AA as well as the subscription component Sub.

The usage part deals in controlling and managing service sessions (e.g. announce, stop, suspend, invite, notify changes, negotiate transfer of control rights),

The subscription process involves the *customer* and the service *provider*. Before a service can be used in the subscriber organisation by the *users*, subscription to the service must take place. A subscriber subscribes to a service by signing a contract about the service usage with the service provider. A provider provides a service according to the subscription contract. A subscriber authorises the user for the service usage. Finally a subscribed service can be used by a particular *user*.

Components:

- Access Component: This subsystem handles the access sessions which represent a secure association between one consumer and one SP. This subsystem consists of a component in the consumer domain, the Provider Agent (PA) and Access Agent (AA); and in the SP domain, the Access Agent (AA), the User Agent (UA).
- Session Component: This subsystem manages the sessions. It contains, Session Manager (SM), Service Session Agent (SSA), and the Communication Session Agent
- Subscription Component: This subsystem allows the management of subscribers, users, subscriptions, and service templates.
- Accounting Component: This subsystem provides a basic and an advanced accounting functionalities. It is composed of a Usage Metering Data (UMData) per session. For active services specific accounting methods are provided
- Profiler Component: This component contains all objects relating to profiles: user-profiles, user-service-profiles, user-interface-profiles, service profiles.

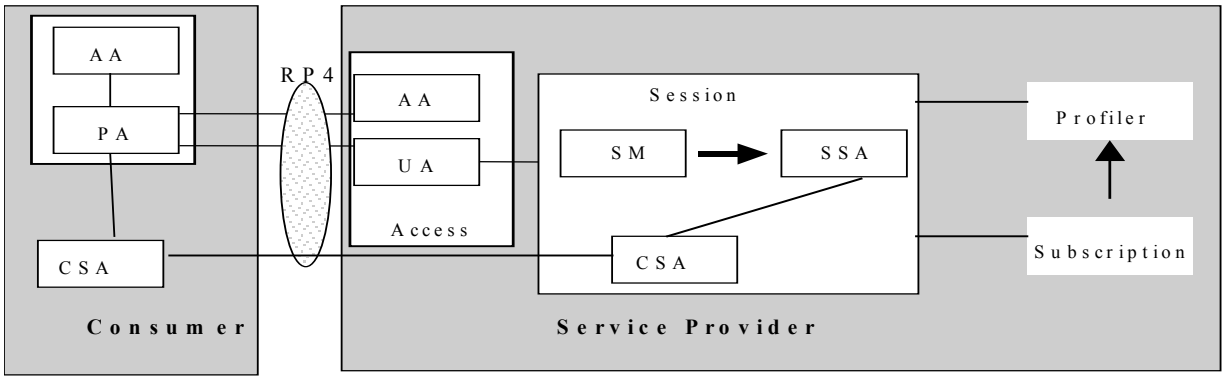


Figure 6-9 RP4 Computational model

6.4.2.2 Scenarios

6.4.2.2.1 Service Access

The Service Access use case describes the initial phase of the service access process. The primary actor is the Customer who initiates the Service Request.

	Service Access
Actors	Primary: Customer. Secondary: Service Provider
Pre-conditions	The End Customer does not have an account and an ID number yet.
Description	1. The Customer provides the company name, the address, the name of the service administrator in the Customer site, its telephone number. 2. The Customer indicates the service type it wants to subscribe for. Examples of services are active services, billing and charging related services providing usage record data based on QoS etc. 3. Service Provider indicates the result of service discovery & creates a customer account based on the information received from the customer. 4. The Customer requests an access session to the service provider 5. Authentication is done between the Customer and Service Provider 6. The Service Provider sets-up the access session including user-context 7. The Customer closes the access session
Post-condition	The Customer has an account and an access and can subscribe to services

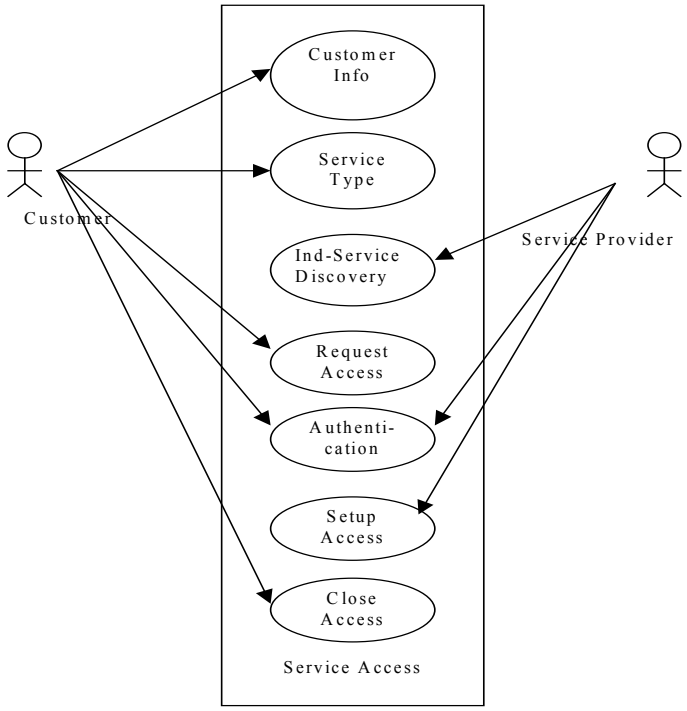


Figure 6-10 Service Access Use Case

6.4.2.2.2 Service Subscription

The Service Subscription use case describes the final phase of the pre-service process. It consists of the negotiations as goal to come to an agreement between the Customer and the Service Provider on the level of service to be provided.

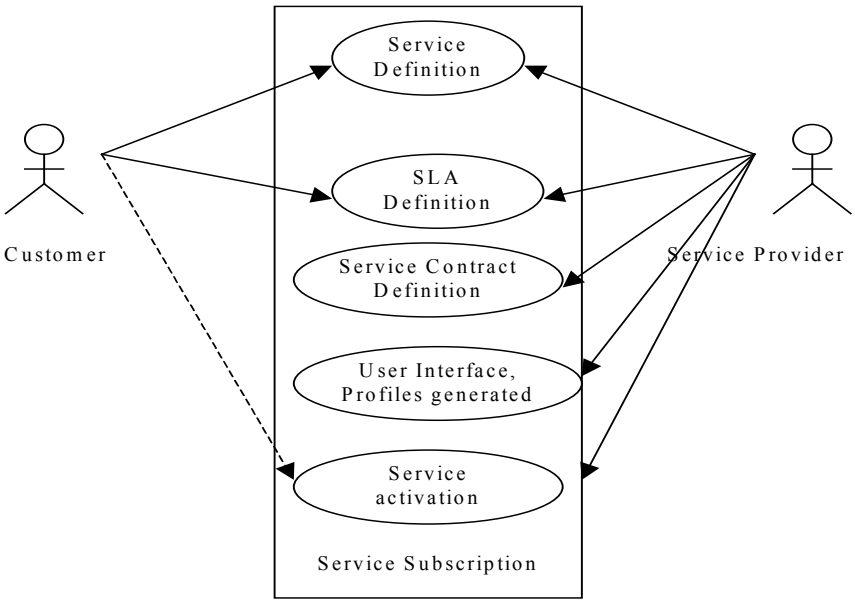


Figure 6-11 Service Subscription Use Case

	Service subscription
Actors	Primary: Customer

	Secondary: Service Provider
Pre-conditions	The Customer has a account and an ID number
Description	1. The Customer selects additional service attributes from the service portfolio, defining details of the service to subscribe to. The attributes are constrained by the service template. The Service Provider validates the service definition. 2. The Customer negotiates the SLA that will apply to the overall subscription. The SLA negotiation is constrained by the rules defined in the SLA template. The Service Provider validates the SLA definition. 3. The Service Provider creates a contract composed of service profile within the profiler containing information on the Customer, the service, the policy and the SLA etc. 4. Profiles are selected and User Interface related to active service is generated and new subscription objects instantiated. 5. The Service Provider as well as the Customer may activate the service according to the contract. Accounting objects are instantiated and specific accounting schema for active components is loaded.
Post-condition	The subscription process is completed, the Customer can afterwards use the service

6.4.2.2.3 Service Usage

The Service Usage use case describes setting up the service sessions and Communication session for the subscribed service to be run.

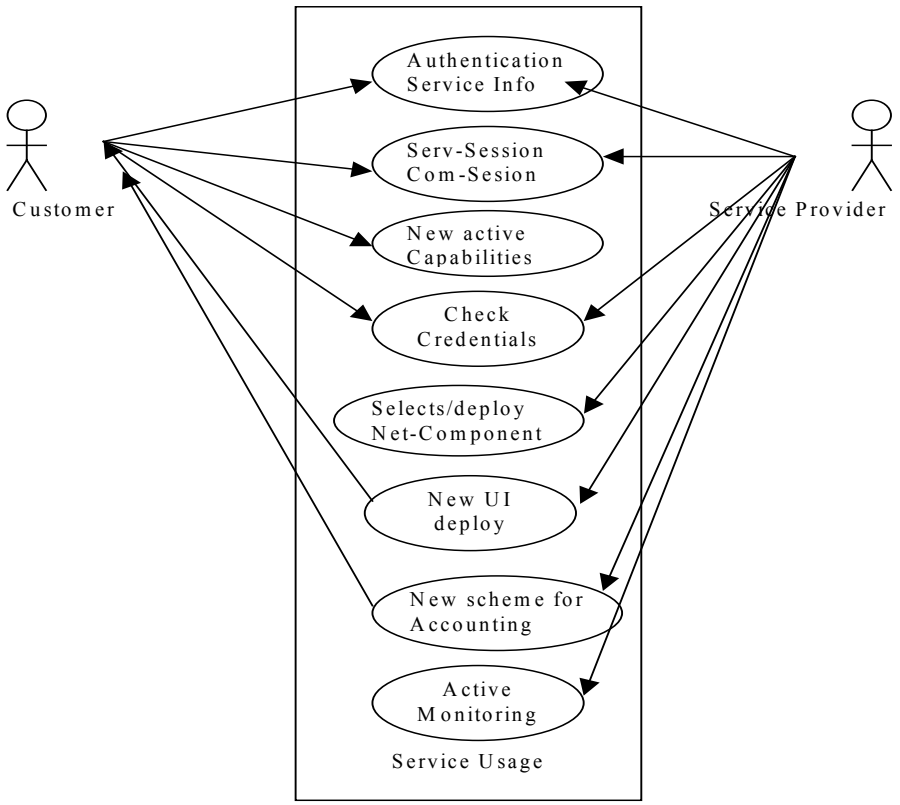


Figure 6-12 Service Usage Use Case

	Service modification
Actors	Primary: Customer Secondary: Service Provider
Pre-conditions	A customer has subscribed to a service. The service is already activated
Description	1. The Customer initiates the service usage, providing Authentication information and service information (Service Type, etc.) 2. The Customer is authorized to use the service by creating associated service sessions and communication session 3. The Accounting is started by the Service Provider considering profiles of the user and of the service 4. User requests new active capabilities 5. SP checks if the user has the credentials for the requests, if yes it would proceed. If not, SLA re-negotiation should be made. 6. SP discover components and deploy the selected component towards the ANSP domain 7. The SP selects the required UI supporting the user control (if any) related to the component deployed. The SP deploy this UI in the consumer equipment. 8. New accounting scheme is created considering the new component and the user-profile records are updated 9. The SP could monitor the service usage using the facilities of the deployed components.
Post-condition	The service is under usage

6.4.2.2.4 Service Modification

The Service Modification use case describes an option that allows the Customer to re-negotiate the SLA. The Customer might need additional features of a service (requests delegation facilities), change the terms and conditions of the service delivery or QoS parameters. If necessary the Service Provider will be consulted and may take-over the service modification process.

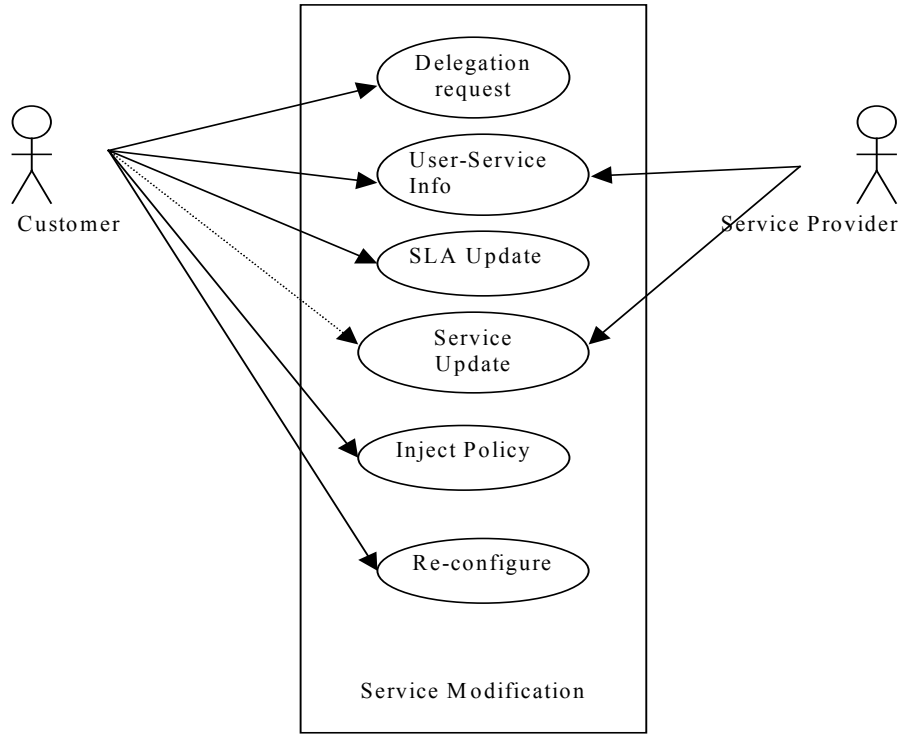


Figure 6-13 Service Modification Use Case

	Service modification
Actors	Primary: Customer Secondary: Service Provider
Pre-conditions	A customer has subscribed to a service. The service is already activated or is scheduled.
Description	1. The Customer initiates the service modification asking delegation privilege, providing 2. The consumer credentials are checked and service information (Service Type, etc.) 3. The Customer is authorised to perform the service modification (using different active plugins..) by updating the SLA and the service profile. The Service Provider will be the main actor to perform the service modification. 4. Service Provider updates configuration and the composition of the service. The new UI is sent to the user and new accounting scheme take action 5. The consumer has the capability to request new management policy to be handled by the system 6. The consumer has the capability to adapt the configuration of the service according to his needs
Post-condition	The service has been modified and re-activated.

6.4.2.2.5 Service Subscription Termination

The Service Subscription Termination use case describes the termination process of a service at the end of the service contract. The Service Provider initiates the process and notifies the termination of the service to the Customer. But before the end of the contract the End Customer can request a service subscription termination.

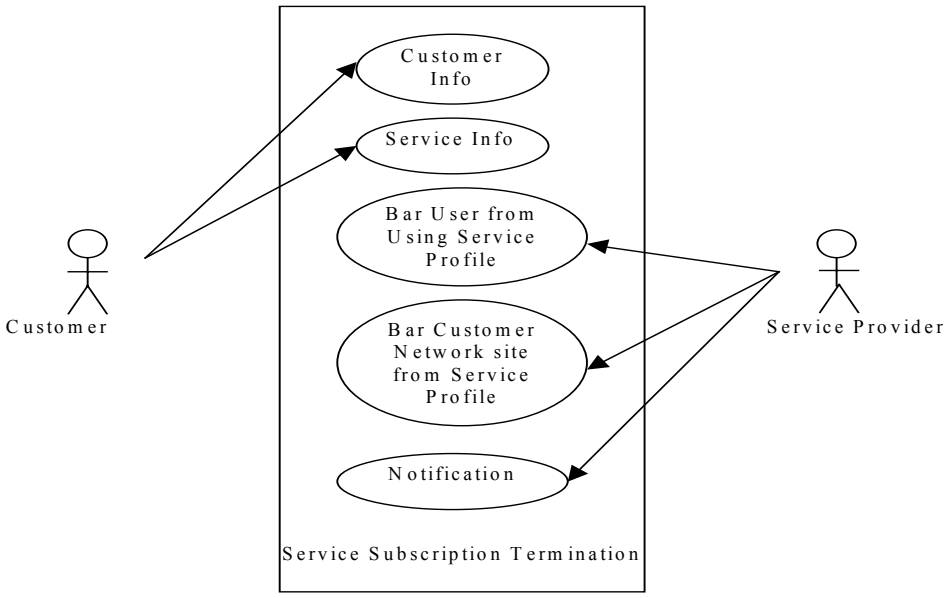


Figure 6-14 Service Subscription Termination Use Case

	Service subscription termination
Actors	Primary: Customer Secondary: Service Provider
Pre-conditions	A Customer has a subscription to the service, which is to be terminated.
Description	1. The Customer sends a request for service subscription termination to the Service Provider . The request contains the Customer information as well as the information on the service for which the subscription is to be terminated. 2. Following the end of the service contract, the Service Provider initiates the termination of the subscription. The service profile is selected and any Customer Network site is barred from the service profile. 3. The End-User is barred from the service profile. So, no user from the Customer side has access to the service for which the subscription is being terminated. 4. The Service Provider notifies the Customer that the subscription to the service is terminated. The Notification contains the day, the time, the service type, etc.
Post-condition	The service subscription is terminated, any user from the End Customer side has no access to the service.

6.4.2.3 Dynamic Model

6.4.2.3.1 Access Session

This scenario shows how a consumer can establish an access session with his UA in the domain of a certain service provider, so he can make use of this service provider's services to which he has previously subscribed.

6.4.2.3.1.1 Preconditions

It is assumed that the consumer has already had an initial contact with this service provider by means of which the consumer's PA obtained a reference to the `Service providerInitial` interface of an access agent of the service provider.

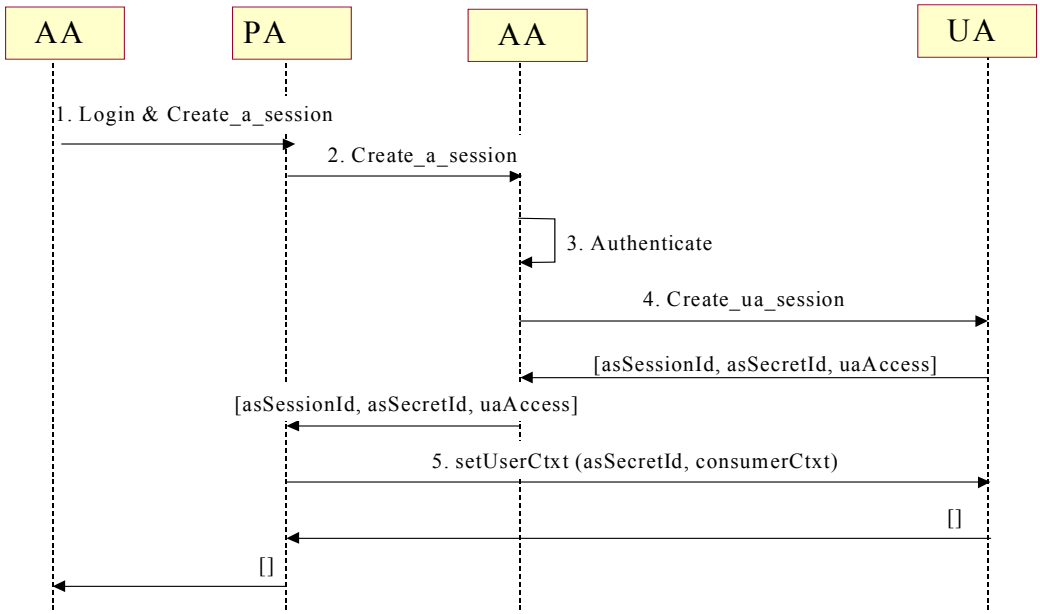


Figure 6-15 Sequence diagram of Start Access Session scenario

6.4.2.3.1.2 Scenario

1 AA -> PA::login()

The consumer uses an access session related AA to forward to the PA the request to log in to the service provider as a known consumer. The access session AA requests the consumer's user name (id) and password.

2. PA -> AA:: requestAccess()

The PA invokes requestAccess() on the service provider's AA to establish an access session that allows the consumer to access the services he is subscribed to.

3. AA -> Authenticate

4. AA -> UA:: setupAccessSession()

5. PA -> UA:: setUserCtxt()

The PA starts access session interactions invoking the setUserCtxt() . This gives the consumer's UA some information about the consumer's domain, such as interface references, available user applications, or the operating system used.

The consumer's UA acknowledges the receipt of this consumer's domain information. The PA should have this acknowledgment before considering the establishment of the access session complete, because some operations may rise an exception if the UA does not have this information.

6.4.2.3.2 Register of New Customer

This scenario shows how an anonymous user can register as a customer of the service provider.

6.4.2.3.2.1 Preconditions

The service session components for are already instantiated.

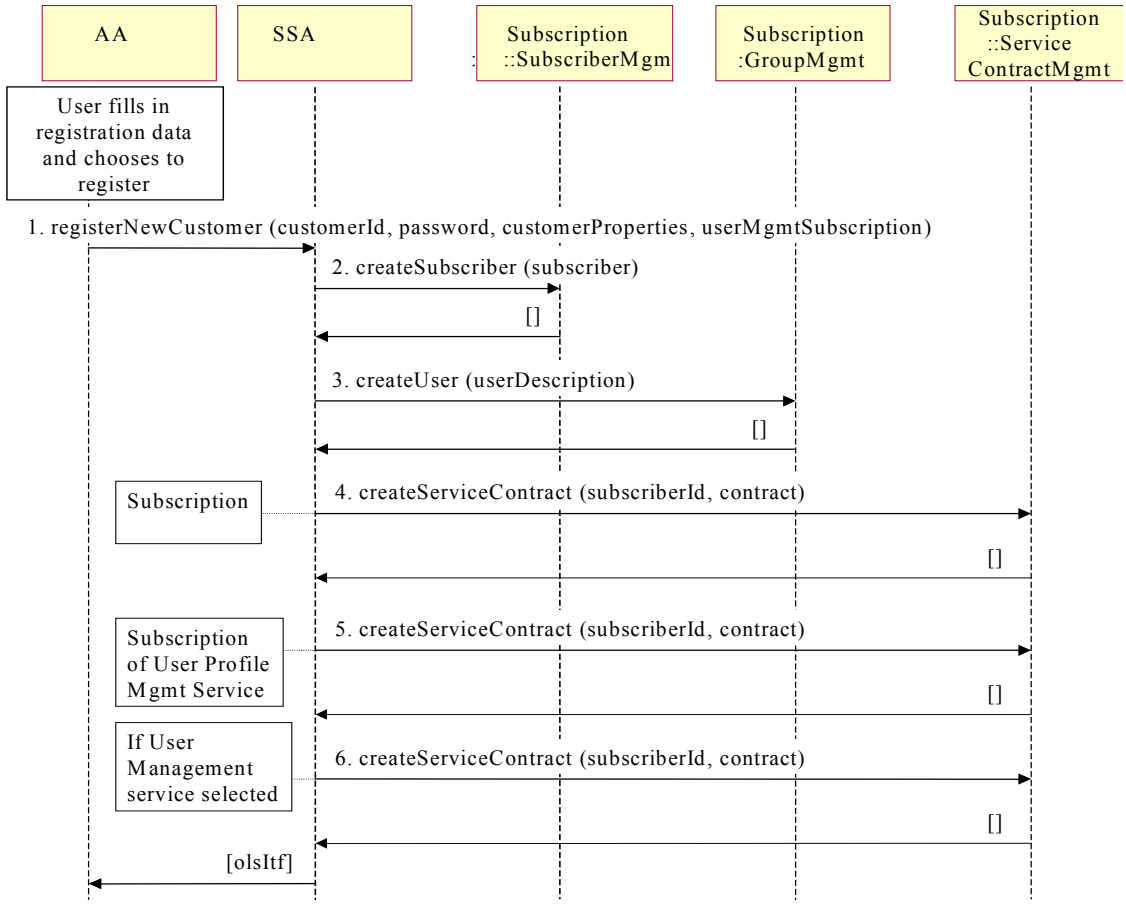


Figure 6-16: Registration of new customer

6.4.2.3.2.2 Scenario

1. AA -> SSA::registerNewCustomer() The user uses the AA to forward to the SSA the request to register as a new customer. The user must supply a customer name (id) and password. He may optionally supply a list of additional properties like postal address, email address etc. As a fourth parameter the user may choose to subscribe to user management as a subscription feature. Then, he can create additional user records to authorize these users to consume the subscribed services.

2. SSA -> Subscription:: createSubscriber() The SSA requests the Subscription component to add a subscriber record for the new customer.

3. SSA -> Subscription:: createUser() The SSA requests the Subscription component to add a user record for the new customer. This user record is needed in addition to the subscriber record, in order to make the customer known as a user of services.

Through the registration the user obtains the authorization to use several services implicitly The next three operations establish the service contracts for these services.

4. SSA -> Subscription:: createServiceContract(). The SSA requests the Subscription component to add a service contract which describes the conditions for the new customer to use the online subscription service for the subscription of services.

5. SSA -> Subscription:: createServiceContract(). The SSA requests the Subscription component to add a service contract which describes the conditions for the new customer to use the user profile management service.

6. SSA -> Subscription:: createServiceContract(). If the user has chosen to subscribe the user management feature a service contract is also added for the user management service.

Finally, a reference is returned to the AA. The user can then continue to subscribe other services via this interface.

6.4.2.3.3 Subscription of a Service

This scenario describes the possibility for a customer to subscribe a service.

6.4.2.3.3.1 Preconditions

The user has been registered as a customer.

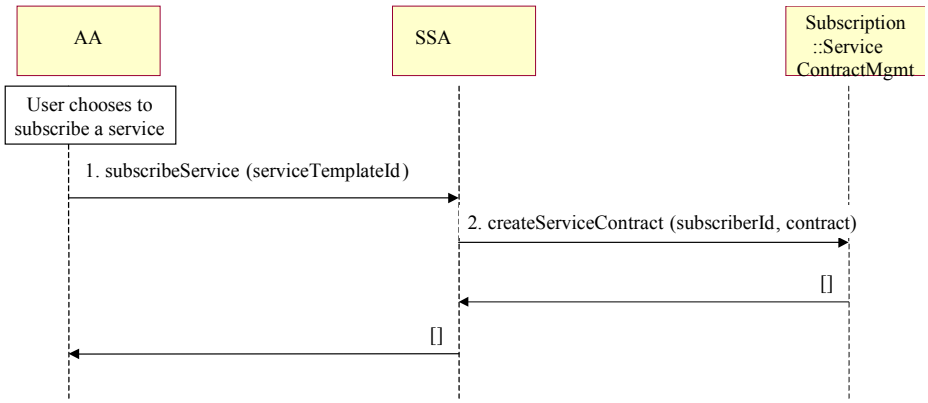


Figure 6-17 Subscription of a service

6.4.2.3.3.2 Scenario

1. ssAA -> SSA:: subscribeService() AA forwards to the SSA the request to subscribe a service. The service identifier (serviceTemplateId) is supplied as parameter.

2. SSA -> Subscription:: createServiceContract() The SSA requests to the Subscription component to create a service contract for the service to be subscribed. The customer name and the contract information are passed as parameters.

6.4.2.3.4 Start Service

This scenario allows a user involved in an access session with a named UA to start a service session.

6.4.2.3.4.1 Preconditions

The access session was successful and the consumer authorised

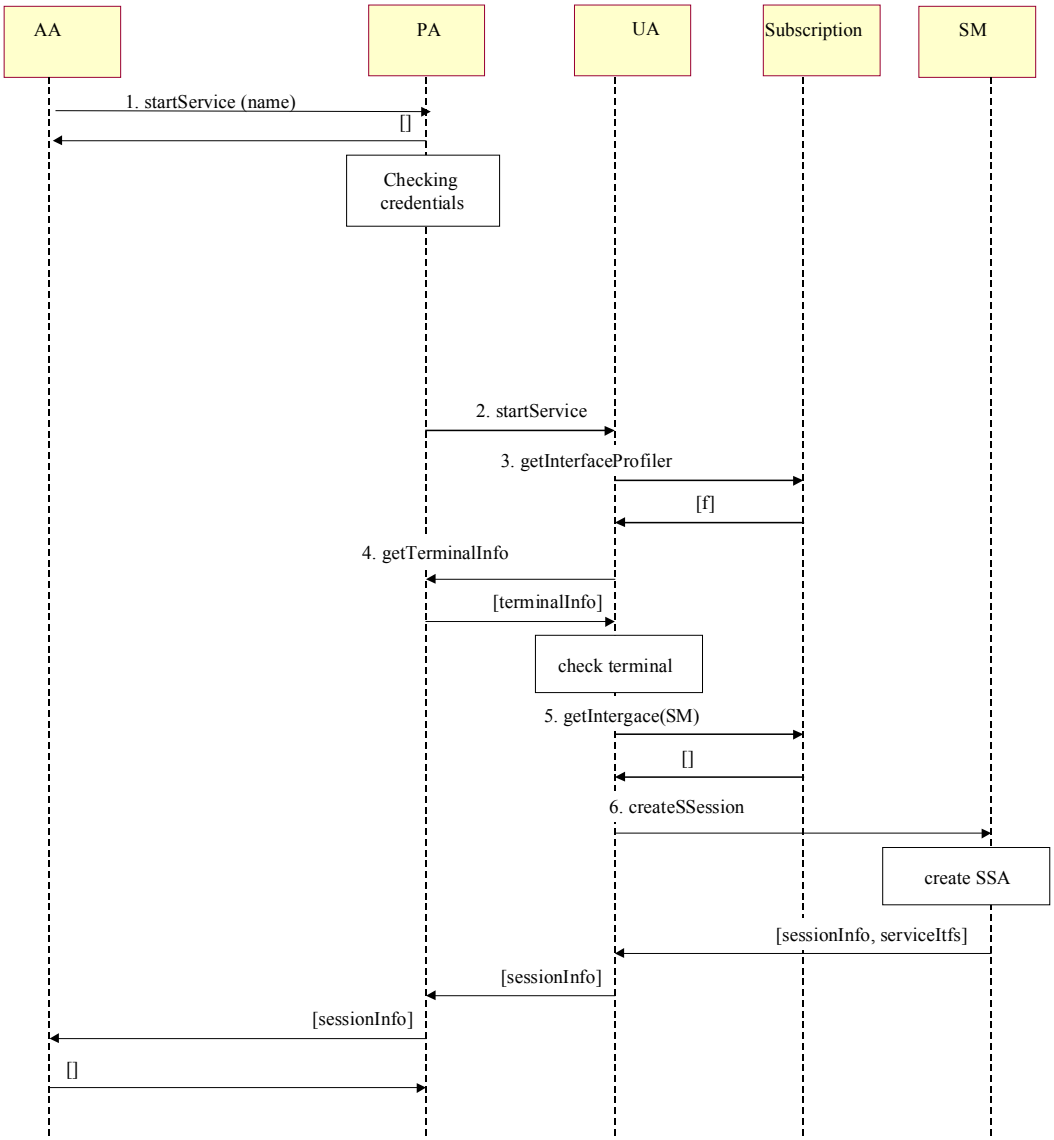


Figure 6-18 Sequence diagram of Start Service scenario

6.4.2.3.4.2 Scenario

1. AA -> PA:: startService()

The consumer selects a service from the list of subscribed services. The AA requests from the PA a new service session of the given service type. The PA then initiates a procedure to load the corresponding session segment. This may also involve the download of an applet. Subsequently, the session segment of AA is instantiated in the end system.

2. PA -> UA:: startService ()

The PA in turn calls startService at the UA. In the following, the UA performs (various non-prescriptive) authorization and personalization actions (steps 6 to 8) before continuing. The UA may return unsuccessful and raise an exception to the PA if the service request is declined.

3. UA -> Sub:: getProfilerByServiceTemplateId ()

The UA requests from the Sub component a reference to a profiler interface which allows to check subscription and terminal constraints with respect to the requested service. The successful creation of the profiler interface also implies that the consumer is authorized to start the service.

4. UA -> PA:: getTerminalInfo ()

The UA proceeds to perform a check whether the consumer terminal is capable of supporting the requested service. Therefore, the terminal information is interrogated from the PA by getTerminalInfo. In the following, this information is checked against the terminal requirements of the service (specific interaction with the profiler interface not shown).

5. UA -> Sub:: getSMRefByServiceTemplateId ()

The UA must obtain a reference to a service factory to create the session components. This reference is predefined and queried by invoking getSvcFactoryRefByServiceTemplateId at the Sub component.

6. UA -> SM:: createSSession()

The UA requests the SSA session components to be instantiated and initialized with optional properties specified by the UA. The SM creates an SSA and initializes them in a service specific manner.

6.4.2.3.5 Modify of Customer Registration

This scenario describes the possibility for a customer to modify the registration data.

6.4.2.3.5.1 Preconditions:

The user has been registered as a customer.

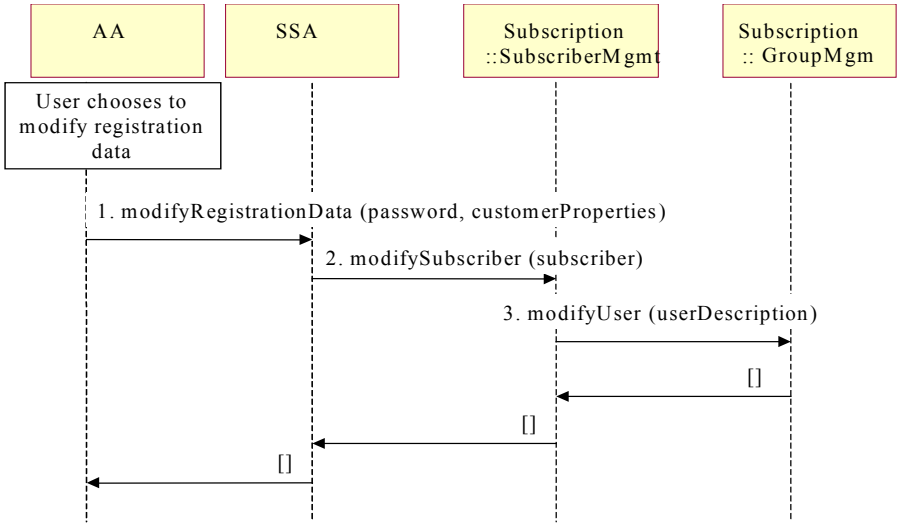


Figure 6-19 Modification of customer registration data

6.4.2.3.5.2 Scenario

1. AA -> SSA::modifyRegistrationData()

The user uses the AA to forward to the SSA the request to modify the registration data. The (potentially modified) password and the list of modified customer properties are supplied as parameters.

2. SSA -> Subscription:: modifySubscriber()

The modified subscriber record is transferred to the Subscription component.

3. Subscription -> Subscription:: modifyUser()

The user record associated with the customer is modified accordingly.

6.4.2.3.6 Deletion of Customer Registration

This scenario describes the possibility for a customer to withdraw the registration.

6.4.2.3.6.1 Preconditions

The user has been registered as a customer.

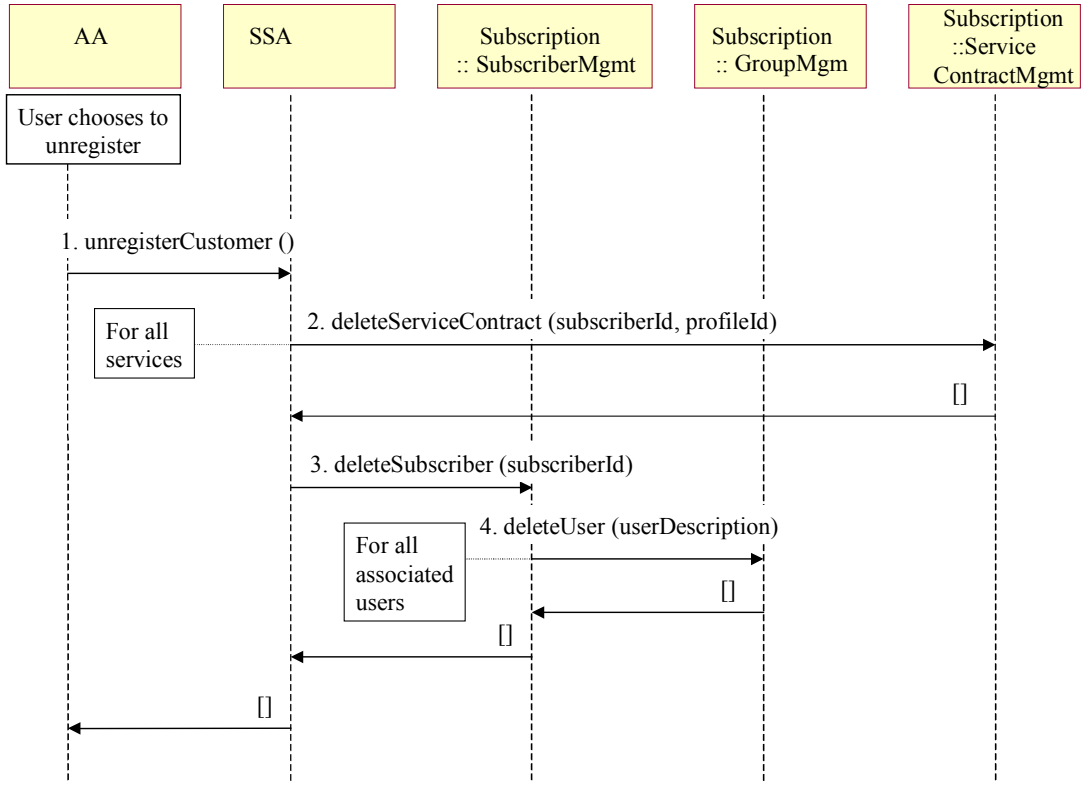


Figure 6-20 Deletion of customer registration

6.4.2.3.6.2 Scenario

1. AA (session segment) -> SSA::unregisterCustomer()

The user uses the session segment of the AA to forward to the SSA the request to unregister as a customer. The (potentially modified) password and the list of modified customer properties are supplied as parameters.

2. SSA -> Subscription:: deleteServiceContract()

For all services the customer has subscribed to, the SSA requests to the Subscription component to delete the service contracts.

3. SSA -> Subscription:: deleteSubscriber()

The SSA requests to the Subscription component to delete the subscriber record for the customer.

4. Subscription -> Subscription:: deleteUser()

All the user records associated with the customer are withdrawn. Consequently, the associated users will not be able to access services anymore.

6.4.2.4 Components Description

6.4.2.4.1 Access Component:

This component exists in both domains (SP, Consumer) with different configurations. In the consumer domain, it encapsulates the segments of the Provider Agent and the Access Agent. In the SP domain, it encapsulates the segments of the User Agent and the Access Agent.

6.4.2.4.1.1 Access Agent (AA)

6.4.2.4.1.1.1 Purpose

The Access Agent (AA) mainly implements the Core Segment of TSAS with some adaptations. It provides the initial access session contact point of the Consumer within the Service Provider domain to initiate the establishment of an access session with that Service Provider. Also, the AA within the consumer domain enables users/applications to use/exploit the PA capabilities. The AA is service specific and logically supports authentication and a combination of session controls: starting/ending/suspending/resuming the session...

6.4.2.4.1.1.2 Interface Description

6.4.2.4.1.1.2.1 Access Segment

6.4.2.4.1.1.2.1.1 Initial

this interface implements one part of TSAS interface while the other part would be addressed by UA and PA

`initiate_authentication`

allows the user to initiate an authentication procedure. If the Security Service is supported by both the user and the provider, then it may be used to mutually authenticate the user and the provider.

in: user_domain - identifier for the user domain.

in: auth_type - type of authentication mechanism requested by the user.

out: provider_domain - identifier of the provider domain.

6.4.2.4.1.1.2.1.2 Authentication

this interface has been extended compared to TSAS to include authentication for getting management capabilities related to the active service.

`select_auth_method`

for selecting the authentication procedure. The user uses this method to gain access to the provider domain, by means of an access session. The user may choose if the service invoked involve active capabilities or not.

in: ActiveCap - Indicates if the invoked service is active

in: auth_caps - the type of access interface requested by the user.

out: selected_cap - the reference for the provider to call the access interface of the user.

out: ActiveCapabilityselection – the reference for active service capability list

`authenticate`

to perform the authentication for the service and for each active service capability . (It can be invoked several times to complete the authentication procedure).

in: AuthCapability selected_cap - (returned by select_auth_method)

in: ActiveCapability selection – (returned by select_auth_method)

in: string challenge

out: string response - provides the response of the provider to the challenge data of the user in the current sequence.

`abort_authentication`

to abort the authentication procedure. If this method has been invoked, calls to the request_access operation will raise the AccessError exception, until the user has been properly authenticated. It contains no attributes.

6.4.2.4.1.1.2.1.3 Access

This interface allows an authenticated user to access services that may contain active capabilities. It contains extensions of TSAS related interface. The Access interface allows the user to access services offered by the provider, and to gain references to other interfaces, as part of segments.

`select_active_service`

This operation is used by the user to identify the (active) service that they wish to use.

in: service_id identified the service required. If the service_id is not recognised, then a ServiceError exception is raised

in: service_properties - is a list of the active service properties that the service instance should support. (These properties are used to initialise the service instance for use by the user.)

in: service_properties - is a list of the basic service properties (not including active properties) that the service instance should support. (These properties are used to initialise the service instance for use by the user.)

out: service_token - is a free format text token returned by the provider, which can be used to start a service session with the selected service properties.

`end_access`

This operation is used to end the user's access session with the provider. The user requests that it's access session is ended. After it is invoked, the user will not longer be authenticated with the provider.

in: end_access_properties - properties is a PropertyList defining the actions to be taken by the provider in ending the access session. If the properties are invalid, a PropertyError exception is raised.

`list_segments`

This operation is used to list the segments offered by the provider.

out: segment_ids - should only include segment identifiers to segments that are offered by the provider, and are available to this user.

`get_segment`

This operation is used to establish a segment between the user and the provider. Segments may define interfaces to be offered by both the user and provider, when establishing a segment.

in: segment_id

in: user_refs,

out: provider_refs

`release_segment`

This operation is used to release a segment between the user and the provider.

in: SegmentId segment_id. If the segment_id is invalid, a SegmentError exception is raised with an InvalidSegmentId error code.

`sign_initial_service_agreement`

This operation is used by the user to request that initially the provider sign a service agreement on the service, before the user is allowed to use the service. The service agreement provides non-repudiation that the user requested to use the service and gain access to a service session. This SLA could be renegotiated dynamically later, if the user would like to add new active capabilities in the service.

in: service_token - is the token returned by the provider in the call to select_active_service().

in: agreement_text - is the service agreement text that is to be signed by the provider.

in: signing_algorithm - is the algorithm used to compute the digital signature of the service agreement.

out: signature_session_info - is a structure containing the digital signature of the provider for the agreement_text, and the session information.

`sign_dynamic_service_agreement`

This operation is used by the user to renegotiate the SLA with the provider to possibly include new active capabilities (or components) within the service

in: service_token - is the token returned by the provider in the call to select_active_service().

in: new_agreement_text - is the service agreement text that is to be signed by the provider.

in: signing_algorithm - is the algorithm used to compute the digital signature of the service agreement.

out: SignatureAndSessionInfo signature_session_info - is a structure containing the digital signature of the provider for the agreement_text, and the session information.

`Terminate_Service_Agreement`

This method is used by the client application to terminate an agreement for the service.

in: service_token: This is the token passed back in a previous selectService() method call. This token is used to identify the service agreement to be terminated.

in: string agreement_text: This is the termination text describes the reason for the termination of the service agreement.

in: signature_session_info: This is a signed version of a hash of the service token and the termination text.

6.4.2.4.1.1.2.2 Access Control Segment

6.4.2.4.1.1.2.2.1 AccessControl

The AccessControl interface allows a known user to control its access sessions.

list_access_sessions: *returns a list of access sessions. The list contains all the access sessions the user currently established with this provider.*

out: AccessSessionList

`end_access_sessions`

allows the user to end one or more access session. The operation can end the current access session, a specified access session, or all access sessions (including the current one), through the use of the SpecifiedAccessSession parameter

in: SpecifiedAccessSession as

6.4.2.4.1.1.2.2.2 AccessRegister

This interface allows the client to register/unregister interfaces to be used within or outside the access session by the PA or the UA

`RegisterInterface`

allows the AA interfaces to be registered for use within the current access session. The registrations ends when the access session ends, or when the unregisterInterface operation is called. An interfaceIndex is returned to allow the interface to be unregistered.

in: AccessSessionSecretId

in: InterfaceStruct

out: InterfaceIndex

`registerInterfaces:`

allows the client to register a list of interfaces to use within the current access session. The registrations ends when the access session ends, or when the unregisterInterfaces() operation is called. An interfaceIndexList is returned to allow the interface to be unregistered.

in: AccessSessionSecretId

in: RegisterInterfaceList

out: InterfaceIndexList

`registerInterfaceOutsideAccessSession:`

allows the AA to register an interface for use outside an access session. (The interface registered should still be available when no access session exists between the user and provider).

in: AccessSessionSecretId

in: InterfaceStruct

out: InterfaceIndex

`listRegisteredInterfaces:`

allows the AA to list the interfaces which have been registered with the PA by her. The list defines which interfaces are registered for use inside an access session, and which are for use outside.

in: AccessSessionSecretId

in: SpecifiedAccessSession

out: RegisteredInterfaceList

`unregisterInterface:`

allows the AA to unregister an interface, so that the PA will not attempt to use that interface, (either inside or outside the access session).

in: AccessSessionSecretId

in: InterfaceIndex

`UnregisterInterfaces:`

allows the client to unregister alist of interfaces, so that the PA will not attempt to use that interface, (either inside or outside the access session).

in: AccessSessionSecretId

in: InterfaceIndexList

6.4.2.4.1.1.2.3 Service Discovery segment

6.4.2.4.1.1.2.3.1 ServiceDiscovery

This interface allows a known user to access information about its subscribed services and to discover new services. This interface is returned as a result of the segments framework control operation establishing this segment.

`list_user_services`

The list_user_services returns a list of the services to which the user has previously been subscribed.

in: desired_properties- can be used to scope the list of subscribed services

out: ServiceList - identifies the properties that the subscribed services must match.

`discover_services`

returns a list of the services available via this service provider. This operation is used to discover the services provided via the service provider, for use by the service provider.

in: desired_properties,

in: how_many - defines the number of ServiceInfo structures to return in the services parameter.

out: ServiceList,

`get_service_info`

The get_service_info returns information on a specific service, identified by the service_id.

in: service_id,

in: desired_properties- can scope the information that is requested to be returned.

out: service_properties

`list_Service_Types`

This operation returns the names of all service types that are in the repository. The details of the service types can then be obtained using the describeServiceType method.

out: listTypes - The names of the requested service types

`Describe_Service_Type`

This operation lets the caller obtain the details for a particular service type.

in: TpServiceTypeName - The name of the service type to be described

6.4.2.4.1.1.2.4 Session Control Segment

6.4.2.4.1.1.2.4.1 Session Control

This interface is related to control service sessions

`start_session`

This operation is used by the user to start a service session. The service session corresponds to the service token i.e. the service session is a session of the service type, and has the service properties selected when the service token was generated (using select_active_service).

in: service_token - is the token returned by the provider in the call to select_active_service.

out: session_info - is a structure containing information about the service session. It includes the SessionId, SessionPropertyList, and a list of interfaces relating to the service session.

`end_session`

This operation is used to end a service session. After it is invoked, the service session associated with the SessionID will have ended, and will not be accessable to the user.

in: SessionId session_id - identifies the session to end. If the session_id is invalid, a SessionError exception is raised with an InvalidSessionId error code.

leave_ssession

allows the AA to request the deletion of this Consumer from an existing service session (without deleting the service session itself)

in: aSessionSecretId - private id

in: rServiceId - identifies the Service Provider service

in: SessionId - identifies the service session

in: applicationInfo - cause of deletion

out: operationId - returned in case of asynchronous confirmation

cancel_ssession_op

allows the AA to request the abortion of an operation which has reported that it will confirm asynchronously

in: aSessionSecretId - private id

in: rServiceId - identifies the Service Provider service

in: SessionId - identifies the service session

in: operationId - identifies the operation that is to be aborted

in: applicationInfo - application-specific cause of abortion

join_ssession

allows the AA to request to join an existing service session

in: aSessionSecretId - private id

in: rServiceId - identifies the Service Provider service

in: SessionId - identifies the service session

in: applicationInfo - application-specific information

out: partyId - public id generated by Service Provider

out: partySecretId - private id generated by Service Provider

out: retSSessionReq - interface reference to contact point in the Service Provider domain for the service session

out: operationId - returned in case of asynchronous confirmation

accept_invite_ssession

allows the AA to asynchronously accept the invitation generated by the Service Provider to join a service session

in: aSessionSecretId - private id

in: rServiceId - identifies the Service Provider service

in: SessionId - identifies the service session

in: operationId - identifies to which operation this is a response

refuse_invite_ssession

allows the AA to asynchronously refuse the invitation generated by the Service Provider to join a service session

in: aSessionSecretId - private id

in: rServiceId - identifies the Service Provider service

in: SessionId - identifies the service session

in: operationId - identifies to which operation this is a response

in: applicationInfo - cause of refusal

`list_service_sessions`

The list_service_sessions returns a list of the service sessions, which the end-user is involved in. This includes present and suspended sessions. SessionSearchProperties identifies the properties that the sessions must match. It also defines whether a session must match one, all or none of the properties

in: aSessionSecretId - private id: scopes the list of sessions by the access session in which they are used.

in: desired_properties, parameter can be used to scope the list of sessions.

out: SessionList

`resume_session`

resumes a service session. It is used on a session that is suspended.

in: session_id, is the identifier of the session to resume.

in: ApplicationInfo, structure containing information on the application, which will be used to interact with the service session.

out: session_info: structure, which contains information that allows the consumer domain to refer to this service session using other operations on this interface.

6.4.2.4.1.1.2.5 Delegation Segment

Operationally, the client must setup (or reuse) a login context that is appropriate for its role as either initiator, delegate, or impersonator, and perform the operation under that login context. We provide three calls that setup login contexts: become_initiator, become_delegate, become_impersonater. All three calls allow the setting of delegation-related restrictions, extensible restrictions etc. Each creates a new login context as a return value. The calls differ only with regard to the identity information passed in.

Clients are primarily concerned with establishing their identity and the necessary controls on how that identity is used. The model provides a *login context* as an abstraction of the client's identity.

6.4.2.4.1.1.2.5.1 DelegationClient

This interface is the for the client side of the delegation

`become_initiator`

This operation takes an existing login context as in input parameter.

in: my_login_context,

in: delegation_type_permitted, parameter is an enumeration. The value *no delegation* means that this caller's identity may not be used in a service chain. With this value specified, *delegate_restrictions* parameter is ignored.

in: delegate_restrictions, The value *delegate* means that the initiator allows its identity to be delegated but not impersonated.

in: target_restrictions,

in: optional_restrictions, lists of the respective additional restrictions

in: required_restrictions, lists of the respective additional restrictions

in: permit_initiator_compat_mode, comes into play if the initiator's identity is delegated in the service chain (as opposed to a chain of impersonations). The initiator may either permit or deny the use of initiator_compat_mode.

out: *new_login_context* is an output argument that is the new login context that effectively refers to other relevant security attributes (e.g. delegation restrictions etc.)

raises: error status

`become_delegate`

This operation takes an existing login context (i.e. the delegates identity) plus a reference to the identity to chain with.

in: callers_identity,

in: delegation_type_permitted,

in: delegate_restrictions,

in: target_restrictions,

in: optional_restrictions,

in: required_restrictions,

in: compatibility_mode, parameter is an enumeration with the following values: *initiator, direct_requester, initiator_if_possible, none* .

out: *new_login_context* is an output argument that is the new login context that effectively refers to other relevant security attributes (e.g. delegation restrictions etc.)

raises: error status

`become_impersonater`

This operation takes only the identity to impersonate, but needs no existing login context. There is no compatibility mode argument. If the direct requester's identity was actually a chained identity, whatever compatibility mode was used there is retained.

in: callers_identity,

in: delegation_type_permitted,

in: delegate_restrictions,

in: target_restrictions,

in: optional_restrictions,

in: required_restrictions,

out: *new_login_context* is an output argument that is the new login context that effectively refers to other relevant security attributes (e.g. delegation restrictions etc.)

raises: error status

6.4.2.4.1.2 User Agent (UA)

6.4.2.4.1.2.1 Purpose

The User Agent (UA) represents a user in the service provider domain. The creation and deletion of user sessions and service sessions are done via the SM (session manager). The UA also provides the list of services available in a service domain via the Subscription object and the list of existing and public sessions via the SM. The UA provides an interface to SM called uaInit that allows to create and delete UA instances.

6.4.2.4.1.2.2 Interface Description

6.4.2.4.1.2.2.1 Access Control Segment

6.4.2.4.1.2.2.1.1 ProviderAccess

This interface allows the user to get type descriptions from the provider

`getInterfaceTypes`

allows the user to discover all of the interface types supported by the provider domain.

in: AccessSessionSecretId

out: InterfaceTypeList

`getInterface`

allows the user to retrieve an interface reference, giving the interface type and properties.

in: AccessSessionSecretId

in: InterfaceTypeName

in: desiredProperties,

out: InterfaceStruct itf

`getInterfaces`

allows the user to retrieve a list of all the interfaces supported by the provider.

in: AccessSessionSecretId asSecretId,

in: MatchProperties desiredProperties,

out: InterfaceList itfs

6.4.2.4.1.2.2.1.2 uaInitial

allows the client to request the creation of a new access session.

create_ua_session

allows the client to request the creation of a new access session

in: termConfiguration - terminal type and identifier

in: paAccess - interface offered by the PA, user

out: aSessionId - public id generated by Service Provider

out: aSessionSecretId - private id generated by Service Provider

out: uaSession - interfaces offered by the UA

6.4.2.4.1.2.2.2 Invitation Segment

6.4.2.4.1.2.2.2.1 UserInvite

The interface allows a service provider to invite an end-user to join a service session, and to cancel pending invitations when required.

`invite_user`

allows a service provider to invite an end-user

in: invitation,

out: InvitationReply

`cancel_invite_user`

This operation allows a service provider to cancel an invitation to join a service session that has been sent to an end-user.

in: UserId invitee_id,

in: InvitationId id

6.4.2.4.1.2.2.3 Accounting Segment

6.4.2.4.1.2.2.3.1 AccountingPull

This interface allows its clients to retrieve accounting data from the UA. It provides the following operations:

`GetUserLogEntries`

allows its clients to retrieve accounting data about the UA's user specifying a time interval.

`GetSessionLogEntries`

allows its clients to retrieve accounting data about a specific service session the UA's user is/has been taking part of.

6.4.2.4.1.2.2.3.2 AccountingPush

This interface allows its clients to store accounting data in the UA. It provides the following operations:

`StoreBillingEvent`

allows its clients to store a specific billing event in the UA.

`StoreBillingEventList`

allows its clients to store a list of billing events in the UA.

`RemoveBillingEvent`

allows its clients to remove a specified billing event previously stored in the UA.

`RemoveBillingEventList`

allows its clients to remove a list of billing events previously stored in the UA.

`RemoveUserLogEntries`

allows its clients to remove log entries corresponding to a specified time interval.

6.4.2.4.1.2.2.4 Context Segment

6.4.2.4.1.2.2.4.1 ProviderContext

This interface allows a user to set context information related to its domain, into the provider domain.

`set_user_ctxt`

allows the user to inform the provider about the configuration of the consumer domain. In the particular case of the end-user, it can inform the service provider of user applications available in the consumer domain, operating systems, etc.

in: user_ctxt

`get_user_ctxts`

This operation allows the user to retrieve information about user contexts that have been registered with the provider.

in: ctxt,

out: user_ctxts: user_ctxts is a list of structures each containing user domain configuration information and properties such as, possibly, interfaces.

`void get_user_info`

allows the user to request information about himself. This operation returns a UserInfo structure as an out parameter.

out: user_info

6.4.2.4.1.3 Provider Agent (PA)

6.4.2.4.1.3.1 Purpose

The Provider Agent implements the access session contact point of the Service Provider within the Consumer domain. Access Agent AA to implement the PA control of a single access session, i.e. the access session association between a single Consumer and a single Service Provider.

6.4.2.4.1.3.2 Interface Description

6.4.2.4.1.3.2.1 Access Control Segment

6.4.2.4.1.3.2.1.1 paAAControl:

This interface allows the AA to control the lifecycle of the access sessions in which this Terminal is involved.

create_asession

allows the AA to request the creation of a new access session

in: userId - of the Consumer

in: password - of the Consumer

in: aaAccess - interface offered by the AA

out: aSessionId - public id generated by Service Provider

out: aSessionSecretId - private id generated by Service Provider

out: paAA - interface offered by the PA

6.4.2.4.1.3.2.2 Invitation Segment

6.4.2.4.1.3.2.2.1 ProviderInvite

The interface allows a user to retrieve information related to announced service sessions or to invitations meant for that user. The user can use this interface to reply to invitations or request to join announced service sessions or service sessions it has been invited to.

`list_session_invitations`

returns a list of the invitations to join a service session, which have been sent to the end-user through this retailer.

out: InvitationList

`list_session_announcements`

returns a list of the session announcements, which have been announced through this SP. This operation is provided in order to allow an end-user to request a list of service sessions that have been announced.

in: desired_properties,

out: AnnouncementList

`join_session_with_invitation`

allows the end-user to join an existing service session, for which it has received an invitation.

in: invitation_id,

in: ApplicationInfo

in: join_properties,

out: session_info

`join_session_with_announcement`

allows the end-user to join an existing service session, for which the end-user has discovered an announcement.

in: announcement_id,

in: ApplicationInfo

in: join_properties,

out: session_info

`reply_to_invitation`

allows the end-user to reply to a received invitation.

in: invitation_id,

in: InvitationReply

6.4.2.4.1.4

6.4.2.4.1.4.1.1 Accounting Segment

6.4.2.4.1.4.1.1.1 AccountingPull

This interface allows the asUAP to retrieve accounting data from the PA. It provides the following operations:

`GetUserLogEntries`

allows AA to retrieve accounting data about the user specifying a time interval.

`GetSessionLogEntries`

allows AA to retrieve accounting data about a specific service session the user is/has been taking part in.

6.4.2.4.1.4.1.2 Context Segment

6.4.2.4.1.4.1.2.1 UserContext

This interface allows the provider to gain information about the user domain's configuration, and eventual applications.

`get_user_ctxt`

This operation allows the provider to receive all the information about the user domain's configuration, that the user accepts the provider to have access to. In particular it can be used by the service provider to gain access, via the retailer, to information related to the end-user's consumer domain, such as e.g. terminal and applications used.

out: user_ctxt: structure that contains e.g. a property list enabling to include provider specific information.

6.4.2.4.2 Session Component

This component is mainly located at the SP domain and is concerned with sessions. It includes the session manager, the service session agent and the communication session agent.

6.4.2.4.2.1 Session Manager (SM)

6.4.2.4.2.1.1 Purpose

The Session Manager (SM) is responsible for the creation and deletion of SSA service session instances for a specific service.

6.4.2.4.2.1.2 Interface Description

6.4.2.4.2.1.2.1 Session Control Segment

6.4.2.4.2.1.2.1.1 smCreate:

This interface allows the UAs to control the lifecycle of the SSA service sessions for a specific Service Provider service

create_ssession

allows the UAs to request the creation of a new service session - this results in the creation of an SSA service session

in: cServiceId - identifies the consumer service

in: userId - of the Consumer

in: termId - terminal identifier

out: sSessionId - id generated by Service Provider

out: partyId - public id generated by Service Provider

out: partySecretId - private id generated by Service Provider

out: ssaSession - interfaces offered by the SSA

delete_ssession

allows the UAs to request the deletion of a service session - this results in the deletion of all relevant SSA service sessions

in: sSessionId - identifies the service session

create_user_ssession

allows the UAs to request the addition of new User to an existing service session - this results in the creation of a SSA service session

in: sSessionId - identifies the service session

in: newParty - partyId, userId

in: newPartySecretId - private id

out: ssaSession - interfaces (ssaAASession) offered by the SSA

delete_user_ssession

allows the UAs to request the deletion of a party from an existing service session - this results in the deletion of the relevant SSA service session

in: sSessionId - identifies the service session

in: partySecretId - private id

list_ssessions

allows the UAs to request for the list of service sessions for this Service Provider service

out: sSessionList - every element contains sSessionId, sSessionName, rServiceId, cServiceId and partyList (partyId, userId)

get_ssa

allows the UAs to request for the interface reference of the SSA service session for a specific sSessionId

in: sSessionId - identifies the service session

out: ssaSession - interfaces offered by the SSA

MetaSsa offers operations to instantiate meta-objects related to active components.

Instantiate_meta

Allows the UA to request the creation of meta-objects associated with new active components to be injected

in: cServiceId - identifies the consumer service

in: sSessionId - id generated by the SM

in: userId - of the initiator

in: actComp – reference of active component control interface

out: response – success response

out: metaId – reference of the meta- object created

out: umData - interface offered by UMData for new accounting

push_meta

request binding the meta-object with associated active component control interface

in: metaId – Reference of the Object

in: actComp – reference of the active component control interface

in: userId - of the initiator

in: actComp – reference of active component control interface

out: response – success response

6.4.2.4.2.1.2.1.2 ssaMeta

interfaces offered by SSA for the object

pull_meta

request unbinding the meta-object with associated active component control interface (i.e if the active component is deactivated)

in: ssaMeta - interfaces offered by SSA for the object

in: metaId – Reference of the Object

in: actComp – reference of the active component control interface

out: response – success response

6.4.2.4.2.2 Service Session Agent (SSA)

6.4.2.4.2.2.1 Purpose

The Service Session Agent implements the personalised management of the service session for a single party in the session. A distinct SSA instance is deployed in each of the Service Provider domains for each Consumer which has a service subscription with that Service Provider.

6.4.2.4.2.2.2 Interface Description

6.4.2.4.2.2.2.1 Session Control Segment

6.4.2.4.2.2.2.1.1 ssaControl

This interface offers operations to control the life cycle of SSA service session contexts.

create_ssa_session

request the creation of a new SSA service session context

in: cServiceId - identifies the consumer service

in: sSessionId - id generated by the SM

in: userId - of the initiator of the service session creation

in: termId - terminal identifier of initiator

in: sSessionName - application-determined service session name

in: uaSMEvent - interface offered by UA of initiator

in: umData - interface offered by UMData (accounting C)

out: ssaSession - interfaces offered by SSA

delete_ssa_session

request the deletion of a SSA service session context

in: sSessionId - id generated by the SM

in: partySecretId - private identifier of party whose SSA context needs to be deleted

multiparty features

add_party

invoked on SSA associated to relevant owners when a party which is already part of the service session has requested to add, i.e. invite, an additional user

in: operationId - to be used when responding asynchronously

in: requestingParty - userId, partyId of requester

in: newParty - userId, partyId & defaultControl of new party

in: applicationInfo - application-specific information - copied from add_party_req

join_ssession

invoked on SSA associated to relevant owners when a user has requested to join, i.e. add himself to, the service session

in: operationId - to be used when responding asynchronously

in: joiningParty - userId, partyId, defaultControl of new party

in: applicationInfo - application-specific information - copied from join_ssession_req

delete_party

invoked on SSA associated to relevant owners when a party which is already part of the service session has requested to delete another party from the session

in: operationId - to be used when responding asynchronously

in: requestingParty - userId, partyId of requester

in: deleteParty - userId, partyId & defaultControl of party that is to be deleted

in: applicationInfo - application-specific information - copied from delete_party_req

leave_ssession

invoked on SSA associated to relevant owners when a party which is already part of the service session has requested to leave, i.e. remove himself from, the session

in: operationId - to be used when responding asynchronously

in: leavingParty - userId, partyId & defaultControl of party that wants to leave the session

in: applicationInfo - application-specific information - copied from leave_ssession_req

delete_ssession

invoked on SSA associated to relevant owners when a party has requested to delete the complete service session

in: operationId - to be used when responding asynchronously

in: requestingParty - userId, partyId of requester

in: applicationInfo - application-specific information - copied from delete_ssession_req

6.4.2.4.2.2.2.1.2 ssa_meta

ssa_meta is an interface for the meta object to initiate control actions on AFC associated with the active component within a specific communication session

ctrl_meta

allows meta abject to control (adapt_Qos_AFC, partition_AFC, policy_AFC) ANSP component

in: ssaMeta - interfaces offered by SSA for the object

in: comSessionId – Id of communication session

in: metaId – Reference of the Object

in: actComp – reference of the active component control interface

in: ctrlMeta – ref of operation to be executed

out: response – success response

6.4.2.4.2.3 Communication Session Agent

6.4.2.4.2.3.1 Purpose

The communication session Agent is responsible for the creation, the release and the control of the communication session. CSA supports many sessions and offers SSA two interfaces.

6.4.2.4.2.3.2 Interface Description

6.4.2.4.2.3.2.1 Session Control Segment

6.4.2.4.2.3.2.1.1 ComSSetup

This interface allows the Service Session Agent to create, to release, to activate, deactivate and list information about communication sessions toward the ANSP for providing active services.

setup_communication_session

allows the SSA to request the creation of a new communication session

in: SuccessCriterion - Criterion for the creation of the channel

in: DescList - Description of the channel

out: ComSessionName - Name of the session

out: ComSCtrl - Interface for control the session

out: CRespList - Results of the channel created

release_com_session

allows the SSA to request the release the communication session

in: ComSessionName - Name of the session

activate_communication_session

allows the SSA to request the activation of the communication session

in: ComSessionName - Name of the session

deactivate_communication_sessionç

allows the SSA to request the deactivation of the communication session

in: ComSessionName - Name of the session

list_all_communication_sessions

allows the SSA to request the listing of information about the communication session

out: ComSessionNameList - Name of the session

6.4.2.4.2.3.2.1.2 ComSCtrl

This interface allows that the Service Session Agent (SSA) to control the execution session at the ANSP. Through this interface the active flow controller (ACF) is partially dynamically configured according to the requirement of active services. This part is incomplete and would be investigated in RP2.

setup_active_channel

allows the SSA to request the creation channels within a communication session

in: SuccessCriterion - Criterion for the creation of the channel

in: CDescList - Description of the channel

out: CRespList - Results of the channel created

out: RefAFC - Reference of AFC Control interface

release_active_channel

allows the SSA to request the release of the channels in the communication session

in: SuccessCriterion - Criterion for the release of the channels

in: CNameList - List of the names of the channels released

out: CRespList - Results of the channels released

adapt_QoS_AFC

allows the SSA to request from the AFC to adapt QoS parameters

in: RefAFC - Reference of AFC Control interface

in: QoSCriterion - Criterion for QoS adaptation (structure with SLS params)

out: AFCResp - Results of the requests

partition_AFC

allows the SSA to request from the AFC to partition the resources used for the active channel

in: RefAFC - Reference of AFC Control interface

in: ParCriterion - Criterion for new resource partitioning

out: AFCResp - Results of the requests

policy_AFC

allows the SSA to request the execution of a policy by the AFC

in: RefAFC - Reference of AFC Control interface

in: PolCriterion – Policy provided

out: AFCResp - Results of the requests

6.4.2.4.3 Subscription Component

6.4.2.4.3.1 Subscription Component

6.4.2.4.3.1.1 Purpose

The basic functionality provided by the service subscription subsystem can be summarised with the following list of capabilities:

- To collect subscription information from a service provider operator.
- To subscribe/cancel subscribers to a service
- To authorise/bar service use for Groups associated to Subscribers.
- To determine the subscription information available to the subscriber during service access.
- To support modification/interrogation of subscription information.

6.4.2.4.3.1.2 Interface Description

6.4.2.4.3.1.2.1 Subscriber Management Segment

6.4.2.4.3.1.2.1.1 SubscrptCntrl

Control Subscription information : This interface contr the life-cycle of the Subscription Contract, Subscription and Service Profile information objects.

subscribe

Creates a subscription contract

in: Account number

in: Subscription contract

cancel

Terminates a subscriber's subscription to a service

in: Account number

setSubscription

Sets the subscription characteristics for a subscriber to a particular service

in: Account number

in: Subscription

Activate

Activates subscription for a particular Group

in: Account number

in: Group identifier

Deactivate

Prevents a particular Group from using a service

in: Account number

in: Group identifier

6.4.2.4.3.1.2.1.2 SubscriberMgmt

Subscriber management :

createSubscriber

Creates a subscriber

in: Subscriber

out: Account number

modifySubscriber

Modifies attributes for a particular subscriber

in: Subscriber

deleteSubscriber

Deletes a particular subscriber

in: Account number

6.4.2.4.3.1.2.1.3 SubscriberInfoQuery

Subscriber information Query:

getSubscriber

Provides attribute values associated with a subscriber

in: Account number

out: Subscriber

checkSubscriber

Checks whether a subscriber is already created

in: Account number

out: ACK

list_of_Subscribers

Provides a list of subscribers

out: Subscriber list

getSubscription

Obtains global subscription characteristics for a subscriber to a particular service.

in: Account number

out: Subscription

get_required_configuration

Gets the service configuration information for a given account and a service

in: Account number

in: Service identifier

out: Required terminal configuration and service access attributes

list_Subscriptions_of_a_Service

Gets a list of subscriptions to a given service

in: Service identifier

out: Subscription list

list_Subscriptions_of_a_User

Gets a list of subscriptions for a given user

in: User identifier

out: Subscription list

list_Subscriptions_of_a_Subscriber

Gets a list of subscriptions for a given subscriber

in: Account number

out: Subscription list

list_Users_of_a_Service

Gets a list users of a particular service

in: Service_id

out: User_list

6.4.2.4.3.1.2.1.4 GroupMgmt

This interface deals with group of users management

defineGroup

Defines a Group of users

in: Account number

in: Group selection

out: Group id

modifyGroup

Modifies a Group

in: Account number

in: Group selection

deleteGroup

Deletes a Group

in: Account number

in: *Group* identifier

6.4.2.4.3.1.2.1.5 GroupInfoQuery

Group information query:

getGroup

Provides attribute values associated with a Group

in: Account number

in: *Group* identifier

out: *Group* selection

list_of_users

Provides a list of users of a particular subscriber

in: Account number

out: Users list

list_Users_of_a_ListofGroups

Provides a list of users of a list of Groups

in: Account number

in: *Group* list

out: Users list

get_Subscriber_Of_a_User

Gets the subscriber to whom belongs a user

in: user identifier

out: account number

6.4.2.4.3.1.2.2 SP Administration Segment

The service template management can be used in order to introduce new services, modify the properties of existing services or delete offered services.

6.4.2.4.3.1.2.2.1 ServiceTemplateMgmt

This interface deals with service template management

`deploy_templ_service`

The operation deploy_templ_service allows the service provider to define a new instance of an active service template. The provider fills in the provided service properties, the service template properties for the select active service and the possible end-user application properties, active capabilities..

in: provider_id,

in: service_template is given by the service provider

`modify_templ_service`

The operation modify_templ_service allows a service provider to dynamically modify existing active service templates (service offers).

in: oproviderId,

in: oservice_template

`withdraw_templ_service`

The operation withdraw allows a service provider to delete existing service template

in: oproviderId,

in: service_template_id

6.4.2.4.3.1.2.2.2 ServiceTemplateInfoQuery

`list_service_templates`

The operation list returns a list of all service templates for a service provider. There may be an indication the service sought is active service.

in: provider_id

`get_service_template`

The operation get returns the structure of a single service template.

in: provider_id,

in: service_template-id

6.4.2.4.3.1.2.2.3 SLAMgmt

SLA management

CreateSLA-Section

Creates SLA section in contract

in: Subscriber

out: Reference to Account

modifySLA-Section

Modifies SLA attributes for a particular subscriber if the active service has been dynamically updated

in: Subscriber

deleteSLA-Section

Deletes SLA section for a particular subscriber if an active capability has been removed dynamically form the service

in: Reference to Account

6.4.2.4.3.1.2.2.4 SLAInfoQuery

SLA information Query:

list_of_SLA_profiles

Provides a list of SLA profiles, particularly for active services (accounting models..)

out: SLA list

Choose_SLA

Select an SLA profile in the list dynamically in case of new active capability requested

out: SLA reference

getSLA

Provides attribute values associated with a SLA Profile

in: subscriber reference

out: SLA profile reference

checkSLA

Checks if SLA profile conforms to template

in: SLA reference

6.4.2.4.3.1.2.2.5 AdaptConfig

Allows the SP to deploy new configuration (UI, accounting) on the consumer domain if after the re-negotiation new user service control capabilities should be added

`deploy_UI`

The deploy_UI allows the SP to deploy new UI at the consumer premises.

in: UIref ui_mod - structure containing user domain: new UI model

in: ServiceID

`deploy_Acc`

The deploy_Acc allows the SP to deploy new accounting scheme at the consumer premises.

in: Accref Acc_mod - structure containing user domain:new UI model

in: ServiceID

6.4.2.4.3.1.2.3 Delegation Segment

The server-side delegation segment is extended to allow applications to extract the privilege attribute set for each participant in a chained identity. We consider calls to extract the privilege attributes of the initiator of the operation the privilege attributes of each delegate the operation the delegation and extensible restrictions placed by each participant. When a server needs to perform an operation on another target on behalf of its client, it must become a delegate for that client. Servers are the reference monitors for the data they manage. In general they are concerned with extracting the privilege attributes associated with a given remote request. These attributes are then generally passed on to the standard access control algorithm to determine if the client is authorised to perform the requested operation. The caller's privilege attributes may also be used for auditing the operation or otherwise recording information about the participants in the service chain.

6.4.2.4.3.1.2.3.1 DelegationServer

This interface deals with delegation permission/control at the service provider side.

`sec_login_enable_delegation`

operation is used by a client to enable delegation.

in: login_context, argument identifies the login handle to be operated on.

in: legit_delegates, argument provides a list of principals that are to be considered legitimate delegates. This information will be encoded into the delegation token obtained for this context. If this list is empty, any server principal will be allowed to be a delegate for the client.

in: target,

in: optional_restrictions, lists of the respective restrictions to be applied to the delegation request.

in: required_restrictions, lists of the respective restrictions to be applied

to the delegation request.

out: st: error_status_t

`sec_login_disable_delegation`

Delegation is turned off for a given login handle

in: login_context, argument identifies login handle to be operated on.

out: st: error_status_t

`sec_login_become_delegate`

operation used by server processes wishing to be a delegate for their caller. This operation will yield a new login handle which refers to the principal identity composed from the server's login handle and the privilege attributes delegated by the client.

in: login_context, argument identifies the server's identity.

in: priv_attributes, argument refers to the clients privilege attributes -- including the needed delegation token

in: target,

in: optional_restrictions,

in: required_restrictions,

out: new_login_context, argument is an output handle that refers to the composite principal information.

out: st: error_status_t

`sec_login_enable_impersonation`

This operation behaves analogously to the sec_login_enable_delegation function, except that it transmits a delegation token that allows the server to be indistinguishable from the initiator.

in: login_context, argument identifies login handle to be operated on

in: legit_delegates,

in: target,

in: optional_restrictions,

in: required_restrictions,

out: st: error_status_t

6.4.2.4.3.1.2.3.2 Sec_cred Interface

The sec_cred interface provides accessor functions for extracting privilege attributes from the authorization data

`sec_cred_get_initiator`

extracts the privilege attributes for the initiator of an operation.

in: priv_attributes,

out: pac, *privilege attribute certificate*

out: st: error_status_t

`sec_cred_get_delegate`

is an iterator function to extract the privilege attributes of each delegate involved in an operation.

in: priv_attributes,

out: pac,

out: st: error_status_t

`sec_cred_get_optional_restrictions`

iterator functions to extract the optional restrictions from the authorization data

in: priv_attributes,

out: optional_restrictions,

out: st: error_status_t

`sec_cred_get_required_restrictions`

iterator functions to extract the required restrictions from the authorization data

in: priv_attributes,

out: required_restrictions,

out: st: error_status_t

6.4.2.4.4 Profiler Component

6.4.2.4.4.1 Profiler

6.4.2.4.4.1.1 Description:

A profiler object is created upon start of each service session. It holds information about the user, his access rights as well as the user's personal preferences. The profiler implements the information objects: Service Template, Service Profile, User Service Profile and User Profile. The service implementation gets this information from the profiler and writes the user's personal settings.

6.4.2.4.4.1.2 Interface Description

6.4.2.4.4.1.2.1 Profiler Segment

6.4.2.4.4.1.2.1.1 ServProfileMngt:

Query Subscription information : This interface queries Subscription Contract, Subscription and the Service Profile information objects.

defineSvcProfile

Defines the service profile for a given subscriber

in: Account number

in: Service Profile

out: Service Profile identifier

modifySvcProfile

Sets service profile parameters without overriding mandatory settings

in: Account number

in: Service Profile

delSvcProfile

Deletes the service profile for a given subscriber

in: Account number

in: Service Profile id

getSvcProfile

Obtains the service profile for a Group for a particular service.

in: Account number

in: Group identifier

out: Service Profile

check_service_profile

Ckecks validity of service profile for a Group

in: Account number

in: Group identifier

in: R_service_id

in: Terminal Configuration

out: boolean ack

assign

Assigns the service profile to a Group

in: Account number

in: *Group* identifier

in: Service Profile identifier

deassign

Deletes the assignment of a service profile to a Group

in: Account number

in: Group identifier

list_ServiceProfiles_of_a_Subscriber

Gets a list of service profiles for a given subscriber

in: Account number

out: Service Profile list

list_ServiceProfiles_of_a_Subscription

Gets a list of service profiles for a given subscriber to a *particular service*

in: Account number

in: Service identifier

out: Service Profile list

`Group_of_a_ServiceProfile`

Gets the sag associated wit a given serviceProfile

in: Service Profile identifier

out: Group identifier

6.4.2.4.4.1.2.1.2 UserProfileMngt

allows end-user to modify the user profiles and the user service profile settings.

modify_password

The operation modify_password provides the possibility to change the user password.

in: subscriber_id,

in: user_id,

in: old_password

in: new_password

modify_user_profile

The operation modify_user_profile allows and end-user to detail its personal entries in the user_profile. Subscriber and user_id are used to identify in the retailer domain the end-user.

in: SubscriberId subscriber_id,

in: user_id,

in: user_profile

modify_user_service_profile

The operation modify _user_service_profile provides end-user with the ability to change the personal preferences for the usage of a service predefined by the service provider.

in: subscriber_id,

in: user_id,

in: servic_template_id,

in: user_service_profile

delete_user_service_profile

The operation delete_user_service_profile removes the user service profile in the retailer. The subscriber_id and the user_id are used identify the user in the retailer, the service_template_id to identify for which service the profile should be deleted.

in: subscriber_id,

in: userId,

in: service_template_id

6.4.2.4.4.1.2.1.3 UserDescriptionQuery

The query interface allows end-user to question about its user descriptions and user service profiles

get_user_description

The operation get_user_description provides the end-user with information about its user properties and password.

in: subscriber_id,

in: user_id

list_user_service_profile_ids

The operation list_service_profiles_ids returns a list of subscribed end-user service profiles which are associated with the retailers service template_ids.

in: subscriber_id,

in: user_id

get_user_service_profile

The operation get_user_service_profile returns the end-user service profiles

in: subscriberId,

in: userId,

in: serviceTemplateId

6.4.2.4.4.1.2.1.4 UI-Adapt

This interface allows the user to update and downolad to its terminal/system the adapted UI to the active service requested

`set_UI_ctxt`

The set_UI_ctxt allows the user to inform the provider about the requirement or capabilities of the service requested.

in: ui_ctxt - structure containing user domain configuration info

`get_UI_ctxt`

This operation allows the user to update the UI of the active service dynamically each time a new component/capabilities are installed

out: ui_ctxt – reference for new ui

6.4.2.4.4.1.2.2 Context Segment

6.4.2.4.4.1.2.2.1 UserContext

This interface allows the provider to gain information about the user domain's configuration, and eventual applications, or allows a known user to set context information related to its domain, into the provider domain.

`get_user_ctxt`

This operation allows the provider to receive all the information about the user domain's configuration, that the user accepts the provider to have access to.

out: user_ctxt **-** structure containing property list

`set_user_ctxt`

The set_user_ctxt allows the user to inform the provider about the configuration of the consumer domain.

in: user_ctxt - structure containing user domain configuration info

`get_user_ctxts`

This operation allows the user to retrieve information about user contexts that have been registered with the provider.

in: ctxt - is a list of structures each containing user domain configuration information and properties.

out: user_ctxts - list of specified contects

`get_user_info`

allows the user to request information about himself.

out: user_info - This contains the user's UserId, its name, and a list of user properties.

6.4.2.4.5 Accounting Component

6.4.2.4.5.1 Accounting

6.4.2.4.5.1.1 Description

The Account Component (AC) main idea is to handle the charging and billing procedures. One issue that has to be decided before if the charging scheme for active services: when the user requests an active service, he should also include in the SLA a clause on charging for active capabilities with the SP. Each time a new component is installed, the SP may need to update the charging metrics.

6.4.2.4.5.1.2 Interface Description

6.4.2.4.5.1.2.1 Accounting Segment

6.4.2.4.5.1.2.1.1 UMInit

This interface acts as a Control interface of the UMData objects and in general, supports operations for the manipulation of their life-cycle.

init

Instantiates a new UMData computational objects for a subsciption.

in: Ssession - Information of the service session with which the new UMData will be associated.

in: IntRef - The stringified reference of the UMLog object to which will be forwarded the Usage Metering Data Records.

out: IntRef - The stringified reference of the UMData computational object that has just instantiated by the Control.

terminate

Terminates a specific UMData computational object.

in: Ssession - Information of the service session that corresponds to the UMData object that will be terminated.

terminateAll

Terminates all the instantiated UMData computational objects.

check

Retrieves the stringified references of the existed UMData computational objects.

out: IntRefList - A list with the stringified references of all the existed UMData computational objects.

6.4.2.4.5.1.2.1.2 UMActivManagement

This interface supports operations for the administration for metering related to active services

`SetAccountingContext`

This operation notifies the Accounting Component that the End-user has chosen an active service in access

in: IntRef - The reference of the UMData computational object

`StopAccountingContext`

This operation notifies the Accounting Component that the End-user has ended the active service usage

in: IntRef - The reference of the UMData computational object

`StartActCharging`

This operation notifies the Accounting Component that the End-user starts the active service.

in: IntRef - The reference of the UMData computational object

in: Ssession - Information of the service session

`ModifyACtCharging`

This operation notifies the Accounting Component that a new active component has been dynamically injected, and request modifying the metering accordingly. AC is responsible for modifying the account settings after the provider's request.

in: IntRef - The reference of the UMData computational object

in: Ssession - Information of the service session

`StopACtCharging`

This operation notifies the Accounting Component that the End-user quits the active service.

in: IntRef - The reference of the UMData computational object

in: Ssession - Information of the service session

`SuspendActCharging`

This operation notifies the Accounting Component that the End-user suspends a service session dedicated to an active service.

in: IntRef - The reference of the UMData computational object

in: Ssession - Information of the service session

`ResumeActCharging`

This operation notifies the Accounting Component that the End-user resumes an active service.

in: IntRef - The reference of the UMData computational object

in: Ssession - Information of the service session

6.5 Reference Point RP1 (Service Provider- Service Component Manufacturer)

6.5.1 Information Model

6.5.1.1 Introduction

RP1 deals with the interaction between two different important roles: the service provider and the Service Component Provider. The service provider can offer active services to consumers relying on active components provided by a service component provider.

The RP1 reference point in the service component provider domain should allow that active components which are offered by a service component provider, who has a contract with a service provider, to be offered (indirectly as new capabilities) to consumers without any changes in the implementation of his components.

The specifications will cover the following aspects:

- A kind of access session which covers a trusted relation between service providers and service component providers
- The operations which are necessary to exchange information which is needed to start a service, and to discover and select a component form the service component provider domain

- After selection, an agreement is signed between them and the component could be provided to the SP after ordering it.

6.5.1.2 Description

The RP1 information model describes two categories of information objects: 1) roles derived from the roles of people and organisations within the business model, and 2) services. The business model identifies two associated different roles*: service providers*, *service component provider*. A service provider provisions services and may need to make use of different components from different service component providers. The service provider may navigate in the s. component portfolio before choosing a specific component of interest.

Each service component is described by a *service component description* that is restricted by a component template. The component template defines how the component is to be administered including information about the sessions, the required runtime environment as well as general authorisation rights. Also included is configuration information and information on the available tariff schemes.

In addition to the component template, there is also a description of the component interfaces that would restricts how the component would be integrated within a service by the service provider. This information object defines service-specific attributes that can be manipulated by a service provider while the service is running.

Once a component is chosen, registration could take place between the SP-SCP using the Service Component register, and then the component is transferred and integrated within the service and the related service profile (service component flow profile object) is updated.

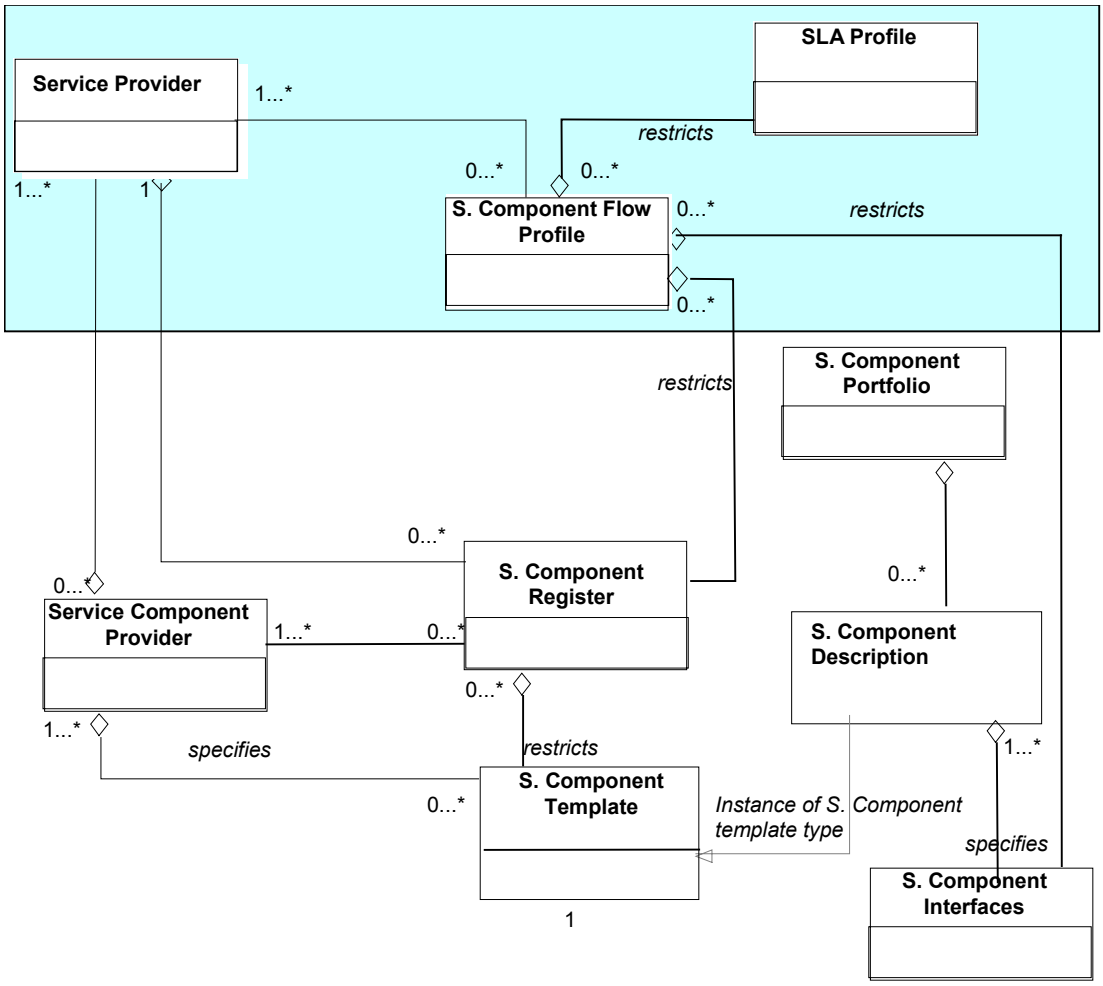


Figure 6-21 RP1 Information model

Descriptions of the individual objects are given below:

Service Component Portfolio: This represents the collection of the separate services offered by this Service Provider.

S. Component Description: Represents the service component specification produced by service component providers.

S. Component Template: Described the generic information and behavioural characteristics of a service component as offered by a *Service component Provider* according to a *S. Component Description*.

Service Component Provider: enables Service Providers to make use of different service components.

S. Component Register : This represents the agreement between the *Service Component Provider* and Service Provider to provide and pay for the S. Component. It represents the tailoring of a *S. Component Template* to the specific requirement and needs of a Service Provider to be provided by the Service Component Provider. It holds contract-specific information such as the time of S. Component provision, a link to a corresponding service component description, and the infrastructure to be used for provisioning the S. Component, and related security issues. The S. Component Register describes the service characteristics for a specified customer and restricts the corresponding S. Component template.

Component Interfaces: The interfaces provided by individual servers that allow monitoring and management function to be performed on them.

SLA Profile: This defines the SLA requirements to be considered when using a service component or a flow of a service components. may be established for the use of the service defined by the related Service Contract.

S. Component Flow Profile: This describes the usage of a flow of S. Component to compose a service to be delivered by the service provider. The usage of the service component is constrained by SC Register. The configuration of the used SC is constrained by related component interfaces. It may take into consideration a specific SLA configuration and profile.

Service Provider: It may contain information about a service provider such as name, address. Service Providers may retrieve S. Component portfolio of several SC providers. He could choose a specific SC, negotiate acquiring this SC in SC Register, including infrastructure to be used for delivery of the SC.

6.5.2 Computational Model

6.5.2.1 Introduction

RP1 scenario takes place when a service provider wants to integrate a component from a service component provider and initiates the deployment of the necessary elements in his domain.

The procedure to be followed can be outlined as follows:

- A Service provider Manager logs on to his own domain and calls an Online Subscription service. This service may include a navigation service.
- An object configuration for the interaction is spawned and a service component provider to connect to is chosen. Then, an access session is opened to the service component provider and a specialised provider subscription service is started.

The service provider can choose one or more components to integrate. For each component, the related information objects are transferred from the Subscription model in the service component provider domain to the service provider domain. For each component that the service provider would use, a subscription entry is created in the component service provider domain.

A Service .component Template holds the necessary information to instantiate a component. For a service provider to be able to integrate a component, the component template for the mediated service needs to be known in his own domain. The Service provider SSA thus also coordinates the transfer from the Subscription component in the service component provider domain to the Subscription component in the service

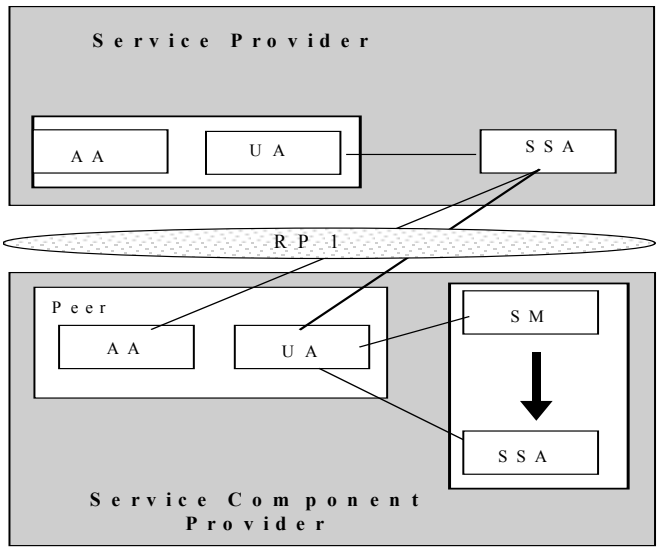


Figure 6-22 RP1 Computation model

6.5.2.2 Scenarios

6.5.2.2.1 Service Access

The Access use case describes the phase of the access process between the service provider and the service component provider. The primary actor is the Service provider who initiates the Request for component.

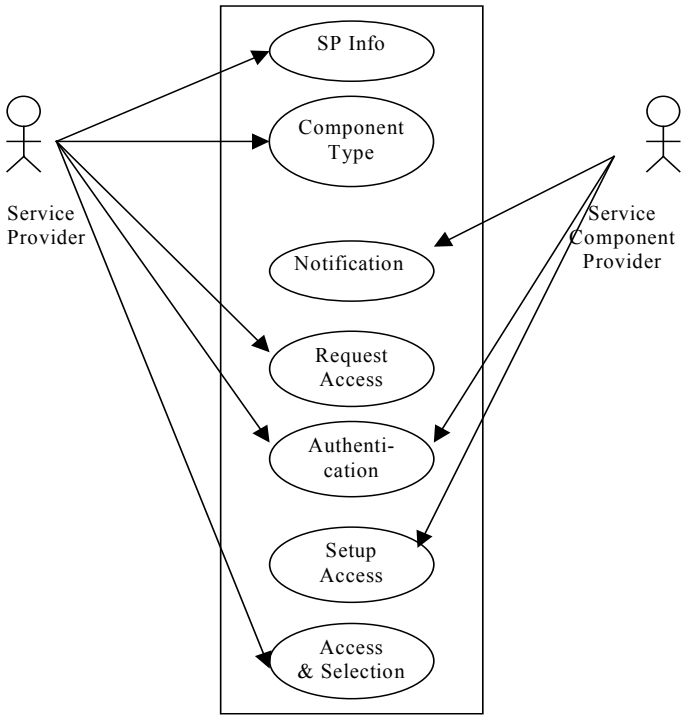


Figure 6-23 Service Access Use Case

	Service Access
Actors	Primary: Service provider. Secondary: Service component provider
Pre-conditions	The Service provider does not have an account and an ID number yet.
Description	1. The Service provider provides the company references 2. The Service provider indicates the component type it wants to subscribe for. If browsing is provided for free, the SP could look himself for a specific component. 3. Service component provider creates a service provider account based on the information received from the service provider, and send a notification to the service provider containing the service provider ID, the account number etc. 4. The Service provider requests an access session to the service component provider 5. Authentication is done between the Service provider and Service component provider

	6. The Service component provider sets-up the access session and make available its s.component portfolio to be browsed with customised features. 7. The SP access the portfolio and may select a component of interest
Post-condition	The Service provider has an account and an access and can subscribe to get components

6.5.2.2.2 Component Subscription/Acquisition

The Component Subscription use case describes the negotiation/provisioning process. It consists of the negotiations as goal to come to an agreement between the Service provider and the Service component provider on the component to be provided, and the transfer of the component from SCP to SP.

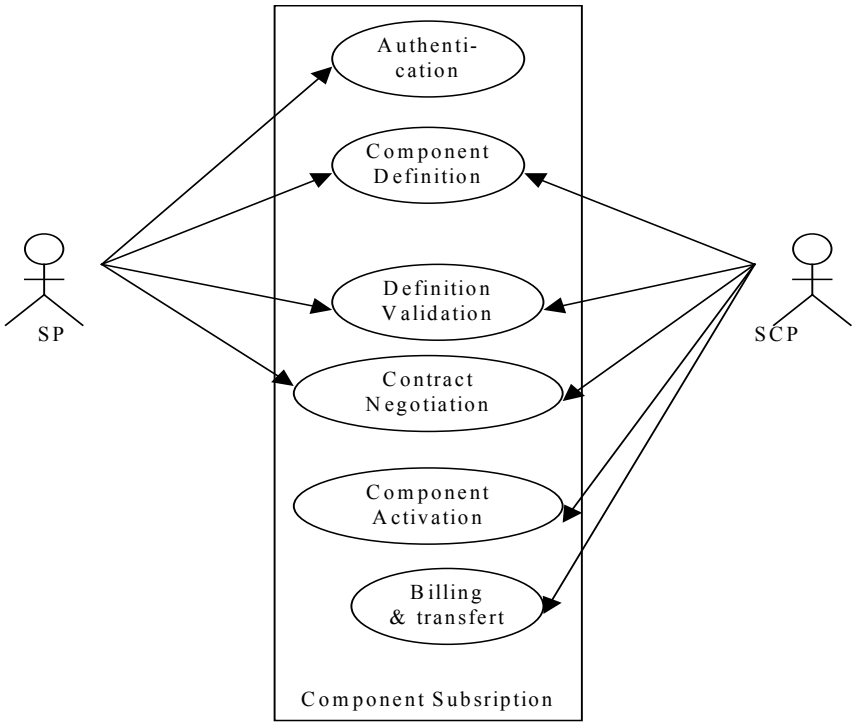


Figure 6-24 Component Subscription Use Case

	Component subscription
Actors	Primary: Service provider Secondary: Service component provider
Pre-conditions	The Service provider has a account and an ID number
Description	1. The Service Provider Authenticate himself to get access to the service component provider domain 2. The Service provider selects additional attributes from the s.component portfolio, defining details of the component to be provided. The attributes are constrained by the service template. 3. The Service provider negotiates the adaptation of the component

	interfaces to his overall service details including SLA profile requirements 4. The Service component provider validates the component features. 5. The Service component provider creates a contract composed of the component facilities provided, the usage/billing agreements, the SP references. 6. The Service component provider may activate the service according to the contract, the SP may instantiated the required objects . 7. The component is transferred from the SCP to the SP domain
Post-condition	The subscription process is completed, the Service provider can afterwards use the component

6.5.2.3 Dynamic Model

6.5.2.3.1 Service Access

This scenario describes how a service component provider service is subscribed by the service provider. The figure below depicts the part of the scenario which is the opening of an access session with the provider and the start of the service session.

6.5.2.3.1.1 Preconditions:

A service provider manager has logged in to the service component provider domain and has started the interaction

6.5.2.3.1.2 Scenario

1. Service provider-SSA -> Peer:: requestNamedAccess()

The Service provider-SSA contacts the provider domain to establish an access session by calling requestNamedAccess() at the Peer.

2. Service provider-SSA -> Peer:: setUserCtxt()

The Service provider-SSA communicates the access session interfaces available in the service provider domain (e.g. i_ConsumerSessionInfo) to the Peer by calling setUserContext().

3. Service provider-SSA -> Peer:: startService()

The start of the Provider service is requested by the Service provider-SSA to the Peer.

4. Peer -> SM::createSession()

Peer finds the appropriate SM and calls createSession(). The necessary component Provider-SSA is then created.

The session information structure is returned from the provider domain to the service provider domain.

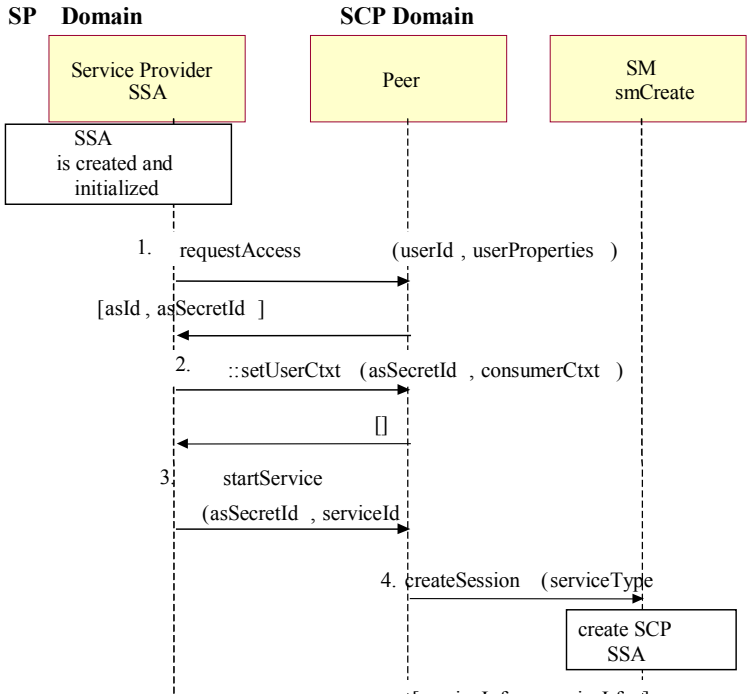


Figure 6-25 Access session

6.5.2.3.2 Selection and Deployment of Components

6.5.2.3.2.1 precondition

The SP has made access session. He likes to get new component

6.5.2.3.2.2 Scenario

1. Service provider (SP)-SSA -> Provider-UA::get_interface_type

The SP get the interface type related to the component provider

2. SP-SSA-> AA:: describe_type

A request to subscribe a component is sent from the Service provider-SSA to the SCP-AA. A service template identifier is passed as a parameter to identify the requested component, e.g.

3. SP-SSA -> AA:: list_components

request to the SCP to list the components

4. SP-SSA-> SSA:: getComponentTemplate

SSA requests the component template instance for the requested component. The type structure is returned to the SSA

5. SP-SSA -> AA:: sign_service_agreement

SCP creates and signs a service contract SLA with SP for providing the required component

6. ComponentProvider-SSA -> SP-SSA:: billing

The SCP charges the SP for the component that has been discussed in the SLA above

7. SP-SSA -> AA:: get_component

The component is transferred to the SP domain

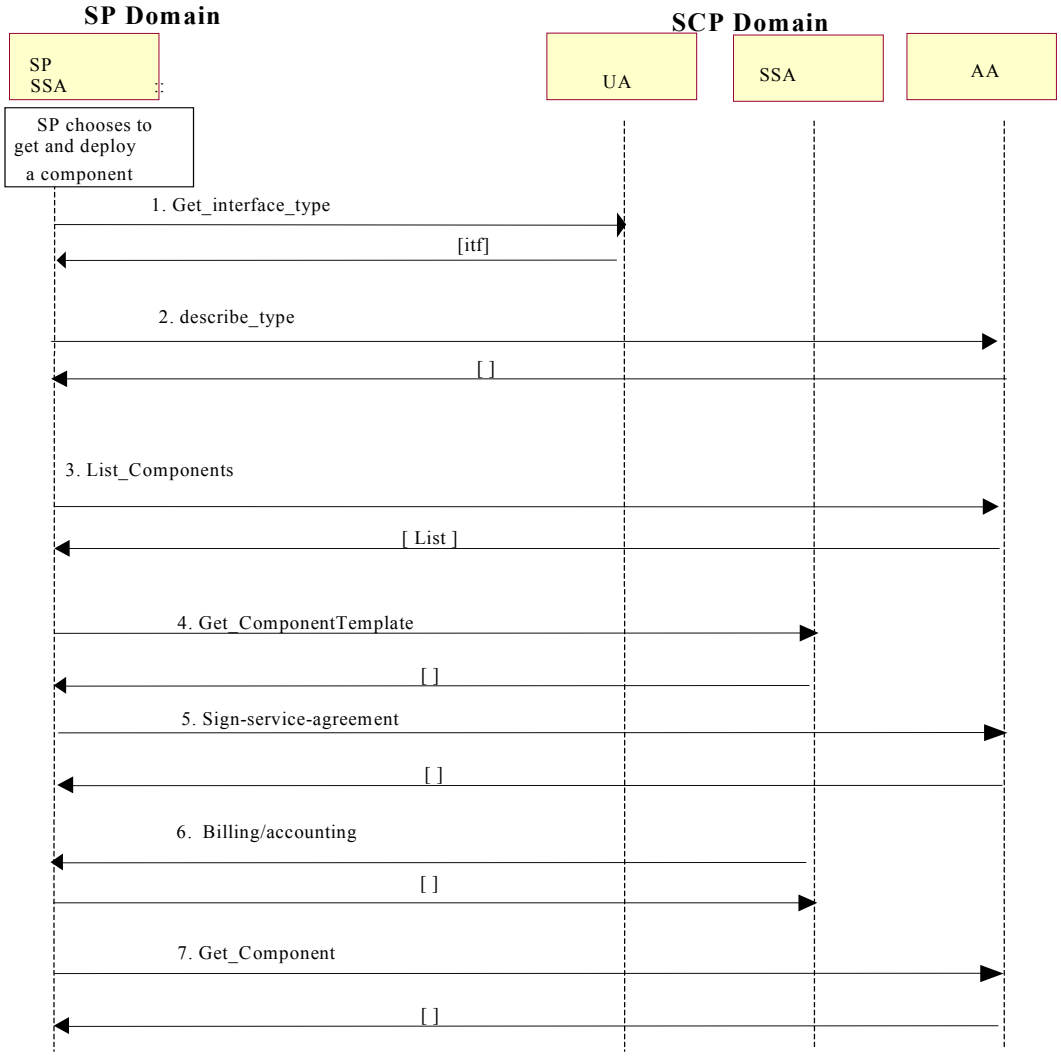


Figure 6-26 Selection & deployment of a service component

6.5.2.4 Components Description

6.5.2.4.1 Access Component:

This component exists in both domains (SP, Consumer) with different configurations. In the consumer domain, it encapsulates the segments of the Provider Agent and the Access Agent. In the SP domain, it encapsulates the segments of the User Agent and the Access Agent.

6.5.2.4.1.1 Access Agent (AA)

6.5.2.4.1.1.1 Purpose

The Access Agent (AA) mainly implements the Core Segment of TSAS with some adaptations. It provides the initial access session contact point of the Consumer within the Service Provider domain to initiate the establishment of an access session with that Service Provider. Also, the AA within the consumer domain enables users/applications to use/exploit the PA capabilities. The AA is service specific and logically supports authentication and a combination of session controls: starting/ending/suspending/resuming the session...

6.5.2.4.1.1.2 Interface Description

6.5.2.4.1.1.2.1 Access Segment

6.5.2.4.1.1.2.1.1 AA-Authentication:

`initiate_authentication`

allows the SP to initiate an authentication procedure.

in: AuthSP_domain - identifier for the SP domain.

in: auth_type - type of authentication mechanism requested by the SP.

out: provider_domain - identifier of the SCP domain.

`select_auth_method`

for selecting the authentication procedure. The SP uses this method to gain access to the SCP domain, by means of an access session. This operation must be invoked only after SP and SCP are authenticated.

in: List auth_caps - the type of access interface requested by the SP.

out: selected_cap - the reference for the SCP to call the access interface of the SP.

`authenticate`

to perform the authentication. (It can be invoked several times to complete the authentication procedure). This procedure may consist of a number of messages e.g. a challenge/ response procedure.

in: selected_cap - (returned by select_auth_method)

in: challenge

out: response - provides the response of the provider to the challenge data of the SP in the current sequence. The response will be based on the challenge data, according to the procedure selected by select_auth_method.

`abort_authentication`

to abort the authentication procedure. If this method has been invoked, calls to the request_access operation on the Initial interface will raise the AccessError exception, until the SP has been properly authenticated. It contains no attributes.

6.5.2.4.1.1.2.1.2 Access:

This interface allows an authenticated SP to access service components. It contains extensions of TSAS related interface. The Access interface allows the SP to access services offered by the service component provider, and to gain references to other interfaces, as part of segments.

`select_component`

This operation is used by the SP to identify the service component that they wish to use.

in: comp_id - identified the component required. If the comp_id is not recognised, then a ServiceError exception is raised

in: component_properties - is a list of the component properties that the service instance should support. (These properties are used to initialise te service instance for use by the SP.)

out: component_token - is a free format text token returned by the provider, which can be used to start a service session with the selected component properties.

`start_session`

This operation is used by the SP to start a service session. The service session corresponds to the component token i.e. the service session is a session of the service type, and has the component properties selected when the component token was generated (using select_component()).

in: component_token - is the token returned by the provider in the call to select_component().

out: session_info - is a structure containing information about the service session. It includes the SessionId, SessionPropertyList, and a list of interfaces relating to the service session.

sign_service_agreement

This operation is used by the SP to request that the provider sign a service agreement on the service, before the SP is allowed to use the service. The service agreement provides non-repudiation that the SP requested to use the component and gain access to a service session.

in: component_token - is the token returned by the provider in the call to select_component().

in: agreement_text - is the service agreement text that is to be signed by the provider.

in: signing_algorithm - is the algorithm used to compute the digital signature of the service agreement.

out: signature_session_info - is a structure containing the digital signature of the provider for the agreement_text, and the session information.

end_access

This operation is used to end the SP's access session with the provider. The SP requests that it's access session is ended. After it is invoked, the SP will not longer be authenticated with the provider.

in: end_access_properties - properties is a PropertyList defining the actions to be taken by the provider in ending the access session. If the properties are invalid, a PropertyError exception is raised.

end_session

This operation is used to end a service session. After it is invoked, the service session associated with the SessionID will have ended, and will not be accessable to the SP.

in: session_id - identifies the session to end. If the session_id is invalid, a SessionError exception is raised with an InvalidSessionId error code.

list_segments

This operation is used to list the segments offered by the provider.

out: segment_ids - should only include segment identifiers to segments that are offered by the provider, and are available to this SP.

get_segment

This operation is used to establish a segment between the SP and the component provider. Segments may define interfaces to be offered by both the SP and provider, when establishing a segment.

in: segment_id

in: InterfaceList SP_refs,

out: provider_refs

6.5.2.4.1.1.2.2 Access Control Segment

6.5.2.4.1.1.2.2.1 AccessControl:

The AccessControl interface allows a known SP to control its access sessions.

list_access_sessions:

returns a list of access sessions. The list contains all the access sessions the SP currently established with this provider.

out: AccessSessionList

end_access_sessions:

allows the SP to end one or more access session. The operation can end the current access session, a specified access session, or all access sessions (including the current one), through the use of the SpecifiedAccessSession parameter

in: SpecifiedAccessSession

6.5.2.4.1.1.2.2.2 AccessRegister:

This interface allows the client to register/unregister interfaces to be used within or outside the access session by the PA or the UA

`RegisterInterface`

allows the AA interfaces to be registered for use within the current access session. The registrations ends when the access session ends, or when the unregisterInterface operation is called. An interfaceIndex is returned to allow the interface to be unregistered.

in: AccessSessionSecretId

in: InterfaceStruct

out: InterfaceIndex

`registerInterfaces`

allows the client to register a list of interfaces to use within the current access session. The registrations ends when the access session ends, or when the unregisterInterfaces() operation is called. An interfaceIndexList is returned to allow the interface to be unregistered.

in: AccessSessionSecretId

in: RegisterInterfaceList

out: InterfaceIndexList

`listRegisteredInterfaces`

allows the AA to list the interfaces which have been registered with the PA by her. The list defines which interfaces are registered for use inside an access session, and which are for use outside.

in: AccessSessionSecretId

in: SpecifiedAccessSession

out: RegisteredInterfaceList

`unregisterInterface`

allows the AA to unregister an interface, so that the PA will not attempt to use that interface, (either inside or outside the access session).

in: AccessSessionSecretId

in: InterfaceIndex

`UnregisterInterfaces`

allows the client to unregister alist of interfaces, so that the PA will not attempt to use that interface, (either inside or outside the access session).

in: AccessSessionSecretId

in: InterfaceIndexList

6.5.2.4.1.1.2.3 Component Discovery segment

6.5.2.4.1.1.2.3.1 ComponentDiscovery:

Interface allowing a known SP to access information about its subscribed components and to discover new components. This interface is returned as a result of the segments framework control operation establishing this segment.

`list_Component_Types`

This operation returns the names of all component types that are in the repository. The details of the component types can then be obtained using the describeComponentType method.

out: listTypes - The names of the requested component type.

Describe_Component_Type

This operation lets the caller obtain the details for a particular component type.

in: ComponentTypeName- The name of the component type to be described

list_components

The list_components returns a list of the components to which the SP has previously been subscribed.

in: desired_properties- can be used to scope the list of subscribed components

out: ComponentList - identifies the properties that the subscribed components must match.

discover_components

returns a list of the components available via this component provider. This operation is used to discover the components provided via the component provider, for use by the service provider.

in: desired_properties,

in: how_many - defines the number of ComponentInfo structures to return in the components parameter.

out: ComponentList

get_component_info

The get_component_info returns information on a specific component, identified by the component_id.

in: component_id,

in: desired_properties- can scope the information that is requested to be returned.

out: component_properties

6.5.2.4.1.1.2.4 Session Control Segment

6.5.2.4.1.1.2.4.1 Session Control

This interface is for service session control

start_session

This operation is used by the SP to start a service session. The service session corresponds to the component token i.e. the service session is a session of the component type, and has the component properties selected when the component token was generated (using select_component).

in: component_token - is the token returned by the provider in the call to select_component.

out: session_info - is a structure containing information about the service session. It includes the SessionId, SessionPropertyList, and a list of interfaces relating to the service session.

end_session

This operation is used to end a service session. After it is invoked, the service session associated with the SessionID will have ended, and will not be accessable to the SP.

in: SessionId session_id - identifies the session to end. If the session_id is invalid, a SessionError exception is raised with an InvalidSessionId error code.

list_service_sessions

The list_service_sessions returns a list of the service sessions, which the SP is involved in. This includes present and suspended sessions. SessionSearchProperties identifies the properties that the sessions must match. It also defines whether a session must match one, all or none of the properties

in: aSessionSecretId - scopes the list of sessions by the access session in which they are used.

in: desired_properties, parameter can be used to scope the list of sessions.

out: SessionList sessions

resume_session

resumes a service session. It is used on a session that is suspended.

in: session_id, is the identifier of the session to resume.

in: ApplicationInfo , structure containing information on the application, which will be used to interact with the service session.

out: SessionInfo : structure, which contains information that allows the consumer domain to refer to this service session using other operations on this interface.

6.5.2.4.1.1.2.5 Accounting Segment

6.5.2.4.1.1.2.5.1 Accounting

This interface allows its clients to retrieve accounting data from the UA.

It provides the following operations:

GetUserLogEntries –

allows its clients to retrieve accounting data about the UA's user specifying a time interval.

GetSessionLogEntries –

allows its clients to retrieve accounting data about a specific service session the UA's user is/has been taking part of.

6.5.2.4.1.2 User Agent (UA)

6.5.2.4.1.2.1 Purpose

The User Agent (UA) is a computational object that represents an SP in the SCP domain (Peer). The UA is the single point of contact to access the available services within a service domain.

The UA allows to authenticate the SP and to change the authentication data. The uaAccess interface offers operations to get the references to the rest of interfaces of the UA instance.

6.5.2.4.1.2.2 Interface Description

6.5.2.4.1.2.2.1 Access Control Segment

6.5.2.4.1.2.2.1.1 ProviderAccess:

This interface allows the SP to get type descriptions from the provider

getInterfaceTypes

allows the SP to discover all of the interface types supported by the component provider domain.

in: AccessSessionSecretId

out: InterfaceTypeList

getInterface

allows the SP to retrieve an interface reference, giving the interface type and properties.

in: AccessSessionSecretId

in: InterfaceTypeName,

in: desiredProperties,

out: InterfaceStruct

getInterfaces

allows the SP to retrieve a list of all the interfaces supported by the provider.

in: AccessSessionSecretId,

in: desiredProperties,

out: InterfaceList

6.5.2.4.2 Session Component

This component is mainly located at the SP domain and is concerned with sessions. It includes the session manager, and the service session agent

6.5.2.4.2.1 Session Manager (SM)

6.5.2.4.2.1.1 Purpose

The Session Manager (SM) is responsible for the creation and deletion of SSA service session instances for a specific service.

6.5.2.4.2.1.2 Interface Description

6.5.2.4.2.1.2.1 Session Control Segment

6.5.2.4.2.1.2.1.1 smCreate:

Interface allowing the UAs to control the lifecycle of the SSA service sessions for a specific component provider

create_ssession

allows the UAs to request the creation of a new service session - this results in the creation of an SSA service session

in: userId - of the SP

out: sSessionId - id generated by Service Component Provider

out: ssaSession - interfaces offered for the SSA

delete_ssession

allows the UAs to request the deletion of a service session - this results in the deletion of all relevant SSA service sessions

in: sSessionId - identifies the service session

list_ssessions

allows the UAs to request for the list of service sessions for this Component Provider

out: sSessionList - every element contains sSessionId, rServiceId

get_ssa

allows the UAs to request for the interface reference of the SSA service session for a specific sSessionId

in: sSessionId - identifies the service session

out: ssaSession - interfaces offered for the SSA

6.5.2.4.2.2 Service Session Agent (SSA)

6.5.2.4.2.2.1 Purpose

The Service Session Agent in SP domain has to establish an access to the service component provider domain, start the service and interact with the SCP SSA which manages the subscription procedure on behalf of the SCP. Moreover, the SCP SSA has to coordinate the transfer of necessary type information from the service component provider domain to the service provider domain.

6.5.2.4.2.2.2 Interface Description

6.5.2.4.2.2.2.1 Session Control Segment

6.5.2.4.2.2.2.1.1 ssaControl

Interface offers operations to control the life cycle of SSA service session contexts.

create_ssa_session

request the creation of a new SSA service session context

in: sSessionId - id generated by the SM

in: sSessionName - application-determined service session name

out: ssaSession - interfaces offered by SSA

delete_ssa_session

request the deletion of a SSA service session context

in: sSessionId - id generated by the SM

6.5.2.4.3 Accounting Component

6.5.2.4.3.1 Accounting

6.5.2.4.3.1.1 Description

The Accounting Component (AC) main idea is to handle the charging and billing procedures. A simple billing procedure is made when a component is acquired by the SP

6.5.2.4.3.1.2 Interface Description

6.5.2.4.3.1.2.1 Accounting Segment

6.5.2.4.3.1.2.1.1 UMInit

This interface acts as a Control interface of the UMData objects and in general, supports operations for the manipulation of their life-cycle.

init

Instantiates a new UMData computational objects for charging a component.

in: Ssession - Information of the service session with which the new UMData will be associated.

out: IntRef - The reference of the UMData computational object that has just instantiated by the charging.

terminate

Terminates a specific UMData computational object.

in: Ssession - Information of the service session that corresponds to the UMData object that will be terminated.

7 CONCLUSION

The deliverable D1 is the result of FAIN workpackage 2, which will complete its objectives with the production of D1. In this conclusion, we are briefly summarising our main achievements and give an outlook on continuation of WP2 work within the other FAIN workpackages.

The IP & telecom services market is becoming increasingly complex and the players in this market are changing rapidly. Relationships between players have to be able to confirm to the this market dynamics. FAIN has developed an enterprise model for Active Networks marketplace that is capable of the required dynamics. As far as we know, the FAIN enterprise model is the first enterprise model which has been designed specifically for active networks, at least in this level of detail.

After an initial phase of continuous revision and adaptation it has being validated by mapping a number of active services/applications onto it. Nevertheless, we are aware that there are various aspects, which need further discussion and investigation.

Probably the most important one is a more detailed description of the reference point (RP3) between ANSP (Active Network Service Provider) and NIP (Network Infrastructure Provider). This activity will be the basis for the definition of node interfaces and implementation scenarios within WP3 and WP4 in the next two years of the FAIN project. The more precise definition of RP3 should be guided by migration scenarios, which show how today's' IP network architecture may evolve into an architecture incorporating AN based concepts. In the section on RP3, we have already discussed some directions and scenarios, which we propose to refine and to discuss under the involvement from operators.

Another issue, which deserves attention in WP3 and WP4, is to structure the artefacts (designs, implementations, ...) along the lines of the enterprise model. At the moment, the activities in these workpackages are driven mainly by a network-oriented view. In order to support the decomposition of a FAIN system into the roles as described in the enterprise model, the respective interfaces have to be present within the designs and implementations created in these workpackages. These objective will be pursued within the work area WA3.9 in the revised project plan of FAIN.

The work in WP2 was driven by three active services / applications, which we think clearly benefit from AN technology. These applications have been used to validate the enterprise model and to derive requirements from them. We are convinced that this using this approach, we were able come up with a set of requirements and an architecture, which is viable to support a large variety of AN applications in the future. The application description will also form the basis for the more detailed architectural investigations on the network- and node level within workpackages WP3 and WP4 and for the demonstration scenarios within WP5.

Another source of AN requirements were the expectations of operators on Active Networks, which at the end of the day would like to make money out of active networks. Contrast this with existing work on Active Network, which is mainly technology driven. In all our activities, we were guided by these expectations, although we are aware that we did not always reference to them as explicit as it would be desirable.

Finally, the glossary, which for technical reasons has been submitted as a separate document, is an living document, which will be continuously maintained also in the future. The preparation of the glossary helped in identifying ambiguous or undefined terms and in defining a common FAIN "vocabulary". It often triggered discussions, which sooner or later not only resulted in precise definitions of the terms, but also in the ability of the various partners in FAIN to work together as a team.

8 APPENDIX

8.1 State of the Art in Enterprise Models

The world of business is changing rapidly, so there has been an increasing need for people to understand their enterprises in a more integrated manner. There is a need to see just what happens in our enterprises. Business leaders are asking, "How? When? Where? Why? Who does what? What tools do they need to do it?" The answers to these questions form the basis of Enterprise modelling. An Enterprise may be considered as a complex system of cultural, process and technology components engineered to accomplish organisational goals.

The identification of business- and enterprise models are also an important starting point when designing a service architecture like the one for FAIN. This means the identification of actors, roles and relationships from the point of view of the organisation of the telecommunications system. Once the roles have been identified, an association is made between the roles and the management related interactions between the roles being investigated.

In this appendix, we briefly review the state of the art in enterprise modelling with an emphasis on models which have been developed within the IT and telecommunication domains. Before doing that, we give an introduction to modelling in general.

8.1.1 Enterprise Modelling

Modelling is an expression of concepts that allows each part of an organisation to understand and contribute to its own development. Models become more valuable as additional parts of an enterprise are added for all to use. Models also promote understanding across different business groups in an organisation. A good model can communicate much of a company's purpose to stakeholders in the business, whether they are employees, shareholders, or customers. Modelling can (and, as we will see, should) be applied to all stages of business and systems development. Models in the area of business processes may be expressed in a specific language, primarily graphical in nature, meant to be used to communicate specific concepts. These languages become useful and generally understood when they are formalised and standardised by standards bodies such as ISO, Industry use and practice, or vendors. There are many modelling methods available. Most of these methods are designed to serve a specific purpose (modelling Object Oriented Systems for instance) and are ill suited to perform outside their intended role (e.g., modelling business processes). An Enterprise Model is composed of models using one or more modelling methods sufficient to describe the different parts of the Enterprise. Enterprise Modelling is the act of providing a set of descriptive representations (e.g., models) that are relevant for describing an Enterprise such that it an be produced to management's requirements (quality) and maintained over the period of its useful life (changed).

What is a Model ?

A model is a representation of concepts that

- can be validated
- can be checked for rigor & robustness
- captures and communicates ideas
- can be changed
- can provide scenarios ('what if' and 'where do I fit in?')

What is not a Model ?

- A collection of drawings and documents
- Anything that can't reflect the business changes

- Anything that can't derive the impact of change
- Anything that can't be navigated

What models ?

When building an Enterprise Model a common question that arises is: what to model and when. There are a number of different models that a business analyst might use in order to complete his part of the picture. These are:

- organisational models.
- process flows.
- functional decomposition.
- requirements definitions.
- technology models.
- use case models .

8.1.2 TMF Business Model

The Telecommunication Management Forum (TMF) is an international forum with approximately 230 member organisations. Overall direction for the forum is governed by a Board of Trustees. This board is made up of approximately 16 member companies - usually with half of these being Telecommunications Service Providers and the others being Telecommunication Vendors. TMF's principal mission is to enable the development of 'standardised' Operational Support System (OSS) or Management System solutions.

8.1.2.1 TMF Business Reference Model

The TMF has developed a Business Reference Model involving three kinds of actors:

- Service Providers;
- Service Customers and;
- Service Providers' Suppliers.

It shows the relevant business relationships between:

- Service Customers and Service Providers;
- Service Providers and;
- Service Providers and their Suppliers, providing the network infrastructure;

Current interfaces for exchanging management information, on which service and network infrastructure providers currently depend, tend to be manual or involve proprietary, low-level interactions. There is therefore an opportunity to establish common specifications and agreements which will allow providers, their customers and their suppliers to work together more effectively, than currently possible. Achieving this goal depends on first identifying the business objective behind the presence of every interface, the roles established, and then ensuring that technical work to implement the interfaces is well founded and delivers the required business benefits.

The Business Reference Model shown in Figure below illustrates the principal points of contact between service providers, their customers and suppliers.

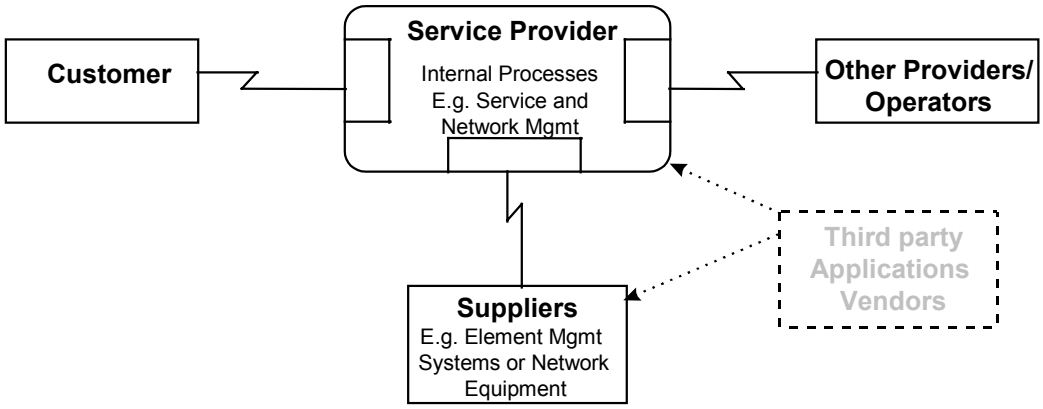


Figure 8-1 - TMF Business Reference Model

8.1.2.2 Business Process Model

Building upon the above reference model, the TMF has defined a Business Process Model (BPM) [NMF-BPM]. The BPM (depicted in Figure 8-2) provides some level of agreement between the parties identified above in terms of their views of the business process they support and the information they consume and generate as a result of the business process activities.

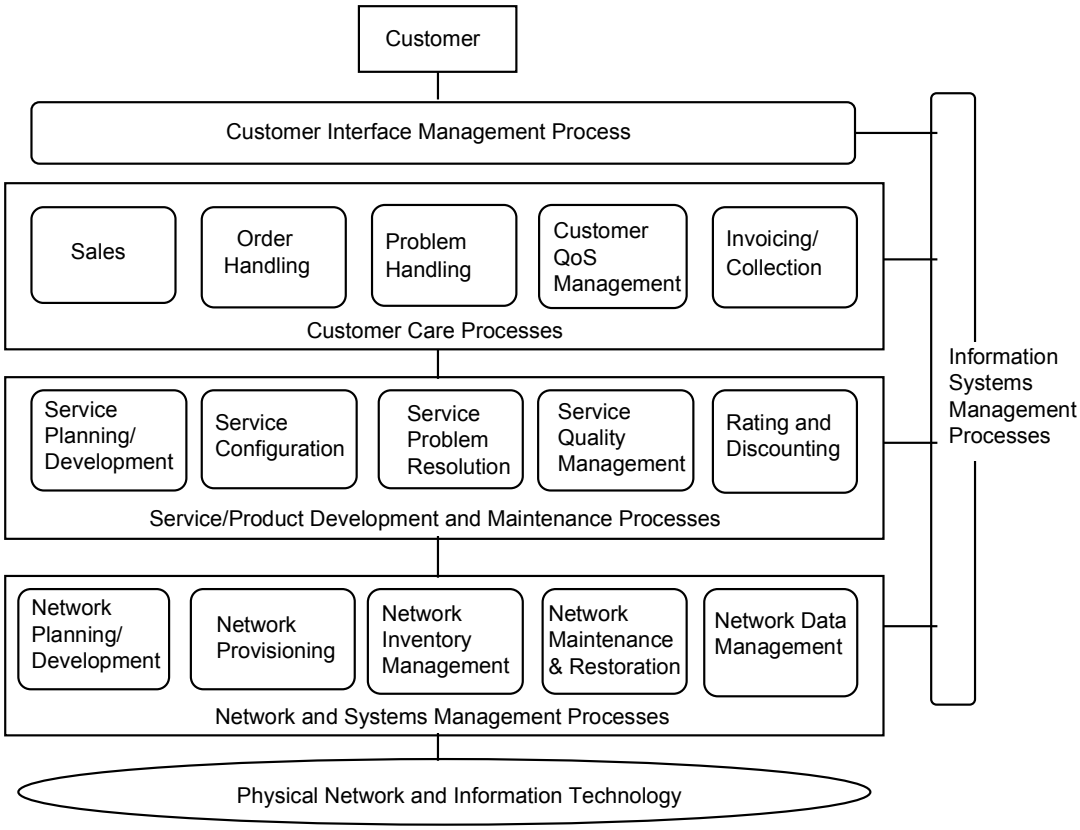


Figure 8-2 - TMF Business Process Model (BPM)

The BPM developed by the TMF covers both Service Management and Network Management business processes – as shown in the figure.

.

8.1.3 TINA Business Model

TINA-Consortium has released the service architecture model and the business model. TINA-C defines a specific user/provider paradigm which reflects the inter-domain interactions between TINA stakeholders. The stakeholders are able to act in the same business role or in different ones. However, TINA stakeholders are more related to telecommunication infrastructure more than to information services. The user could be: an end-user, a subscriber on behalf of a group of end-users, a management operator, a provider asking services of a third party, a provider asking transport services. The provider could be: a retailer, a broker, a third party provider, a connectivity provider.

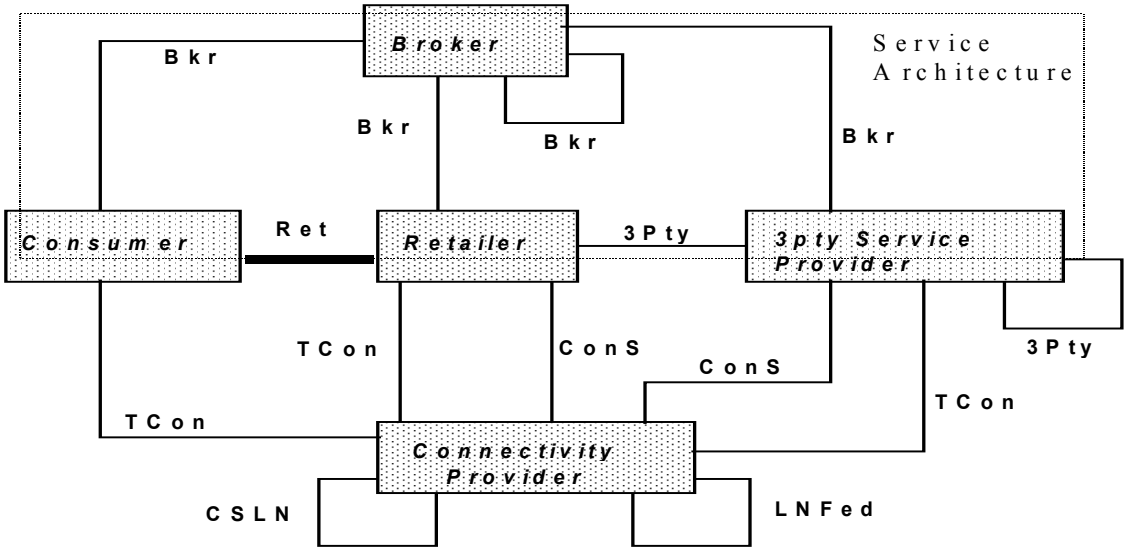


Figure 8-3 - TINA Business Model

The descriptions of the interdomain integration reference points are given below:

- **Ret**

This reference point is in the relationship between Consumer and Retailer domains. It gives the user access to services provided through the retailer. The main functions required are: authentication of the Consumer, (un)subscription of services and the ability to launch a selected service.

- **Bkr**

This reference point provides access to the Broker functionality. Each of the other domains requires access to the Broker to find any stakeholder they need. The main functions required on this reference point are: to enter a search string and obtain references to stakeholders and the ability to (de)register once services as a specific stakeholder.

- **3Pty**

The 3rd Party Service Provider provides services on a wholesale basis through a Retailer. This reference points allows a 3 rd Party Service Provider to interact with a Retailer in order to offer services to Consumers. The main functions required are: the ability to register services with the Retailer and launching a service when requested by one of the Consumers served by the Retailer.

- **RtR**

The Retailer-to-Retailer reference point is defined to allow a Retailer to get services from other Retailers as well and offer these to its Consumers. This allows for a Consumer to only register with one Retailer and use services from other Retailers as well. This is the equivalent of the real-life experience of walking into a store wanting to buy a product which is out of stock or not sold there, but can be ordered from another store..

- **ConS**

The Connectivity Service reference points provides access to some of the functions offered by a Connectivity Provider. Through ConS the Connectivity Provider can be requested to provide an end-to-end network connection for transporting user data. This function is invoked by the Party Service provider for a service requiring a transport network connection.

- **TCon**

The Terminal Connection reference point is used by a Connectivity Provider to confer with the parties involved in an end-to-end network connection. These parties can be in the Consumer, or Party Service Provider role. The reference point allows the Connectivity Provider to retrieve the information required to properly attach? the network connection at the border of each of the domains at the endpoints of the connection.

- **LNFed**

The Layer Network Federation reference point is defined between two parties in the Connectivity Provider role. Since most Connectivity Providers are not able to provide global connections, they need to federate with other Connectivity Providers to provide many of the end-to-end connections. This is the equivalent of a long distance telephone call spanning the domain of various carriers.

- **CSLN**

The Client-Server Layer Network reference point is also defined between parties in the Connectivity Provider role. This reference point is used to provide connections which need the support of an underlying network technology.

8.1.4 ITU-T Model for GII

ITU-T SG 13 Working Party 1/13 has developed the following Recommendations:

- Y.100 – General overview of the Global Information Infrastructure standards development
- Y.110 – Global Information Infrastructure principles and framework architecture
- Y.120 – Global Information Infrastructure scenario methodology

Global Information Infrastructure (GII) will be an infrastructure which facilitates the development, implementation and interoperability of existing and future information services and applications within and across the telecommunications, information technology, consumer electronics and content provision industries. Current areas of applications considered include electronic commerce, telemedicine, teleworking, city information services, intelligent transport systems, distance learning, electronic publishing and libraries. The generic concept of GII calls for an added-value chain-based business model for identifying roles and relationships, ref. Figure 8-4.

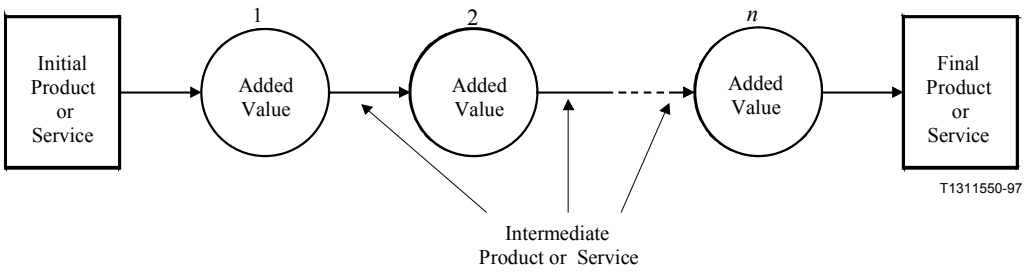


Figure 8-4 Added-value chain model, from [Y.100]

In the following subsections a selection of basic definitions and main ideas is given for these models. The text in these subsections is taken from [Y.110].

Simple Enterprise Model, [Y.110]

According to the simple enterprise model, goods/services flow through various functions or organisations such as service creation function or service provider, brokerage function or broker, etc., based on the infrastructure roles of GII. Based on this value flow, it is possible to classify interconnection types. The next figure illustrates various interconnection types.

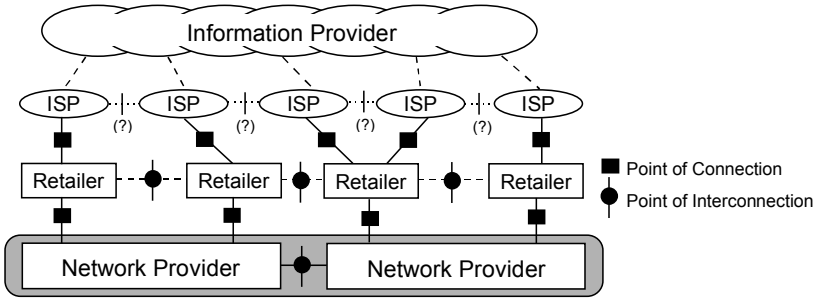


Figure 8-5 Inter-connection[Y.110]

The simple enterprise model is concerned only with roles. This method of definition allows a clear translation from business activity to the relationships which may be the subject of standardisation.

The information industry has a set of structural roles which form a primary value chain including:

- *Information ownership role* – There is frequently value in owning the source of information; for example, a gallery owns pictures images of which can be sold.
- *Provision of information and related content role* – This role takes raw information and makes it available for inclusion in information services and information-based services; for example, it could provide a library of photographs which could be used by travel agents in describing their services.
- *Provision of information-based services role* – The role builds information and information-based services which it makes available to end-users. This includes services which deliver goods or information – for example, a yellow pages service – and also includes services where information is only part of the service – for example, home shopping – where, in addition to the information transactions, physical goods must be delivered to the end user.
- *End user role* – The end user can be either a private individual or a role in another industry which requires information-based services.

The main focus of the GII is on the infrastructural roles. The structural roles, as well as the relationships between them, require support services and these are supplied by infrastructural roles.

The set of infrastructural roles within the GII includes:

- *Communication and networking of information* – This role provides a general, distributed platform on which information applications and services can be supported. It provides the means by which an application and its users can be distributed without being aware of the nature of the distribution. It will invoke the application when requests are sent to it, support messaging to and from the application and also between components of an application. It will include capabilities to support directory services, navigation, security, payments, browsing and searching, etc.
- *Distributed information processing and storage service provision* – This role provides a processing platform on which applications can run and can store data. This role will involve the use of an information appliance.

- *Generic communications service provision* – This role provides telecommunications services for the transport of data, voice and video.
- *Service and application creation support* – This role supplies features which facilitate the production of services and applications which can use communication and information networking.
- *Terminal equipment supply* – This role supplies information appliances to end users which can form an integral part of the distributed platform operated by communication and information networking.

8.1.5 ODP Enterprise Modelling

The ISO Reference Model for Open Distributed Processing [RM-ODP] is an international standard that can serve as a framework for describing architectures of open distributed systems. An RM-ODP specification of a system consists of five different specifications, corresponding to five different, but related, viewpoints on a system. They are the enterprise, information, computational, engineering, and technology viewpoints. Each viewpoint is an abstraction of the whole system focusing on a specific area of concern. The enterprise viewpoint is concerned with the purpose, scope and policies of the ODP system. When describing a distributed system using the RM-ODP framework, the RM-ODP enterprise viewpoint is typically used as a first step in the process.

The enterprise specification for an ODP system specifies the purpose, scope and policies for that system in terms that are meaningful for the stakeholders for that system. This specification is expressed in terms of the behaviour expected of the system by other entities within the enterprise. The term “enterprise” is used here in its widest meaning, i.e. a configuration of entities that may be very large (e.g. a group of co-operating companies), more limited (e.g. a particular service inside a company), or much smaller (e.g. a service provided by an IT system).

The enterprise language introduces the concepts necessary to represent the behaviour expected of an ODP system by other entities within the enterprise and the structuring rules for using those concepts to produce an enterprise specification.

8.1.6 European Project PREPARE

PREPARE introduces the developments leading to the global information society and the characteristics of this environment that are creating new challenges for telecommunications management. In PREPARE a *domain* is associated with an organisation. The set of resources, both virtual and real, belonging to an organisation comprises a management administrative domain, which is defined as a "management domain where the managed objects in the domain are all under the responsibility of one and only one administrative authority" [X.701].

PREPARE distinguishes between intra-domain, inter-domain, and multi-domain management. Management activities concerned with the management of network and multimedia service resources locally within a domain are termed *intra-domain* management. A domain has its own management system which is responsible for managing such resources locally, and how it manages these resources is not visible from outside the domain. *Inter-domain* management refers to the management activities across a management interface between any two separate domains. The management activities at this interface include the exchange of management information and co-operation to support the management of a service across the interface. These activities must be understood by each of the domains’ management systems and adhere to a set of agreed constraints. The management information required to support inter-domain management must be made visible at each domain’s interface and be accessible externally. This is a key interface and is the subject of much of the work elaborated in PREPARE. *Multi-domain* management refers to the management activities required for the end-to-end management of services spanning several domains and comprises the management activities at all the inter-domain management interfaces involved in the management activities for any one service.

8.1.7 Eurescom Project P610

The Project P610 has been addressing the problem of management of multimedia services and TMN system development guidelines. It defined a Management Framework for multimedia services and a set of concepts and principles for specification, analysis, design, reuse and operation of service-oriented management components for multimedia services, as well as the identification of some reusable management components, were produced. In addition it established methodological guidelines for the design of distributed management systems.

At a high abstraction level, a generic business model is comprised of actors, i.e. a person or organisation acting in a service market, the various roles these actors play and the contractual relationships between these roles in the realm of a specific business case (see Figure 8-6)

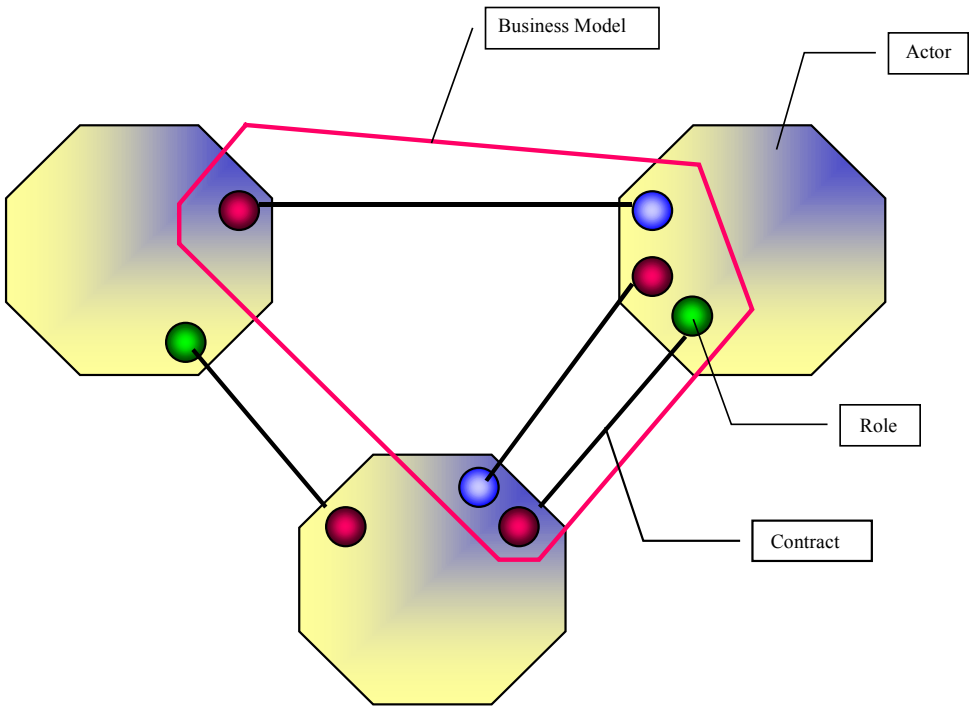


Figure 8-6 Top-level Generic Business Model

At the next level of detail, the roles and the contracts are further specified. The roles are defined in terms of responsibilities, obligations and authorisation.

The ***responsibilities*** describe the mission of the role in terms of **what** it must accomplish or which levels of quality it must achieve. To satisfy these responsibilities they need to be supported by a contract and subsequently this contract must contain, at least, the responsibilities of two associated roles.

The ***obligations*** describe all necessary actions which must be undertaken in order to fulfil the responsibilities of a role, i.e. **how** to execute the role. This is still on a high level, whereas the following Use Case Model specifies the details of the obligations, i.e. each obligation is a point of departure for a specific Use case.

To enable the role to execute its obligations and thus fulfil its responsibilities, ***authorisation*** from the actor point of view is required.

The P610 Business Model is an adaptation of the TINA Business Model [TINA-BM], albeit generalising the notion of retailer into a common service provider role and omitting the details of the reference points and IDL interfaces pertaining to the implementation level.

8.1.8 Internet Business Model

The current Internet is essentially a multi-provider environment due to the principle of the internetworking supported by Internet protocols and by various agreements between *Internet Service Providers*. However, the notion of the *Internet Business Model* is not commonly used within the engineering community. The reason might be the absence of QoS guarantees and, as a consequence of this the absence, the need for *monetary relationships* definitions between the ISPs.

This section develops a business model for Internet services. Considering the current market structure, a general model is presented which identifies major roles for Internet stakeholders and describes interfaces between them.

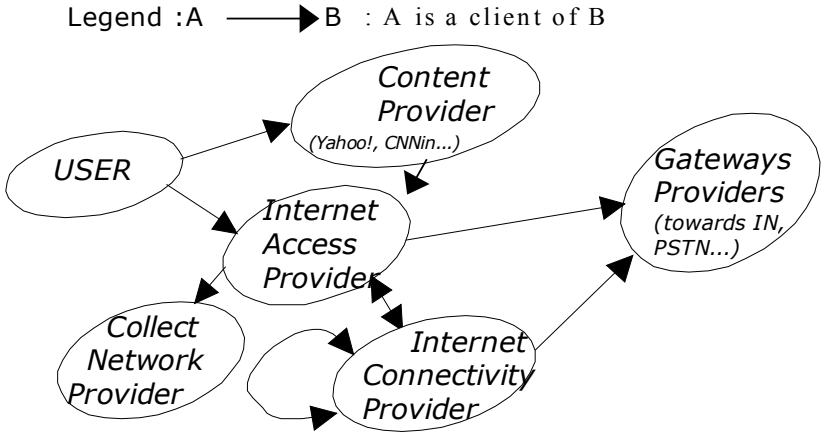


Figure 8-7 Business/role and technical views on Internet-based service provision

User/Customer

In the Internet market two segments of customers could be identified. There is the group of residential users (small office/home office) customers (using dial-up access to Internet) and the group of larger enterprises and organisations connected on a permanent basis. These two groups differ in the technologies they apply to access Internet and in their service requirements and expectations.

Internet Service Provider

A number of different types of ”providers” involved in the provision of Internet-based services are considered. The term *Internet Service Provider* is rather broad. We can differentiate roles oriented towards the wholesale side selling Internet transport services to other smaller ISPs (*Internet Access, Backbone providers)* from a role oriented to the retail side interacting with end-customers *(Collect Network Provider)*.

Collect Network Provider: Its role is to provide connections between end-users and the Internet access provider(s) for a certain region or even on a national and trans-national level. A number of Points of Presence (PoP) distributed over a region, to which the users connect via an access network (CATV, PSTN, etc.), is managed by a collect network provider.

Internet Access Provider: It is dealing with the end-user information, managing accounts, billing, hot line, etc. Traffic is received through the collect network, and passed on via Internet connectivity providers. Offering dial-up access or other access methods to the Internet, e.g. via RAS (remote access servers), they provide extensive customer care including training and helpdesk services.

Internet Connectivity (Backbone) ***Provider***: It offers global connectivity to Internet by being connected to a number of Internet access providers, or to other connectivity providers. An NAP (Network Access Point) connects, in one point, a number of Internet access providers and would be managed by an Internet connectivity provider. They operate at the root level of interconnection hierarchy and handle aggregated IP traffic.

Gateway Provider: It offers gateways towards other types of network (PSTN, mobile network, ..).

Content Providers

Besides the roles dealing with pure IP transport service, additional roles will appear with further development of Internet. Providers of additional services such as content, customer charging and billing, clearing houses, authentication services, service brokering will turn up with the maturing Internet.

8.1.9 DARPA Model for Active Networks

Actually, there's no explicitly defined and commonly accepted business model for active networks. However, from the technical activities within DARPA an initial business model could be synthesised. The DARPA Working Group on Active Networks has defined an architectural framework for active networks. It is mainly based on identifying the structure of node architecture.

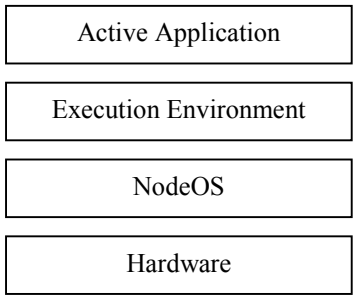


Figure 8-8 DARPA Active Network architecture

The layers of the architecture could be mapped to different manufacturers/producers of software/hardware.:

- Producers of Active Node Hardware
- Producers of Node Operating
- Producers of Execution Environments.
- Producers of Active Application software.

The different manufacturers could then be attached to associated providers interacting together as indicated in the figure. These providers would provide solutions to service providers that would combine these solutions to provide full services to consumers. This scenario is possible, but has not identified by DARPA AN Working Group.

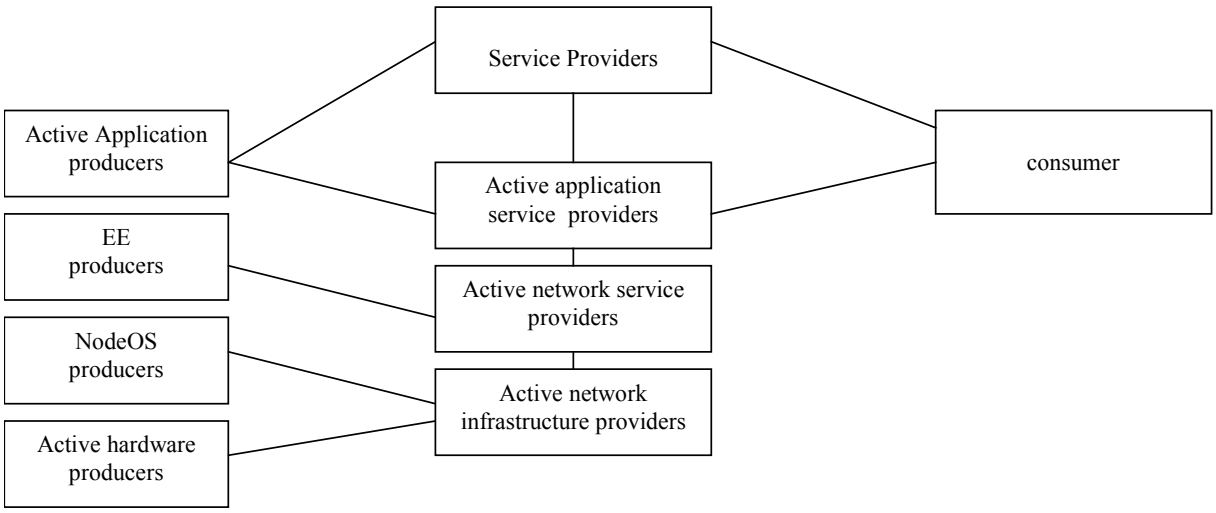


Figure 8-9 AN DARPA synthesised business model

8.1.10 Intelligent Network Business Model

The initial objective of IN is "*to allow the inclusion of additional capabilities to facilitate provisioning of service(s), independent of the service/network implementation in a multi-vendor environment*." The IN conceptual model is defined independently of the actual service/network realisation, but it also introduces functional components allowing for new service implementations, and therefore new QoS requirements and measurements.

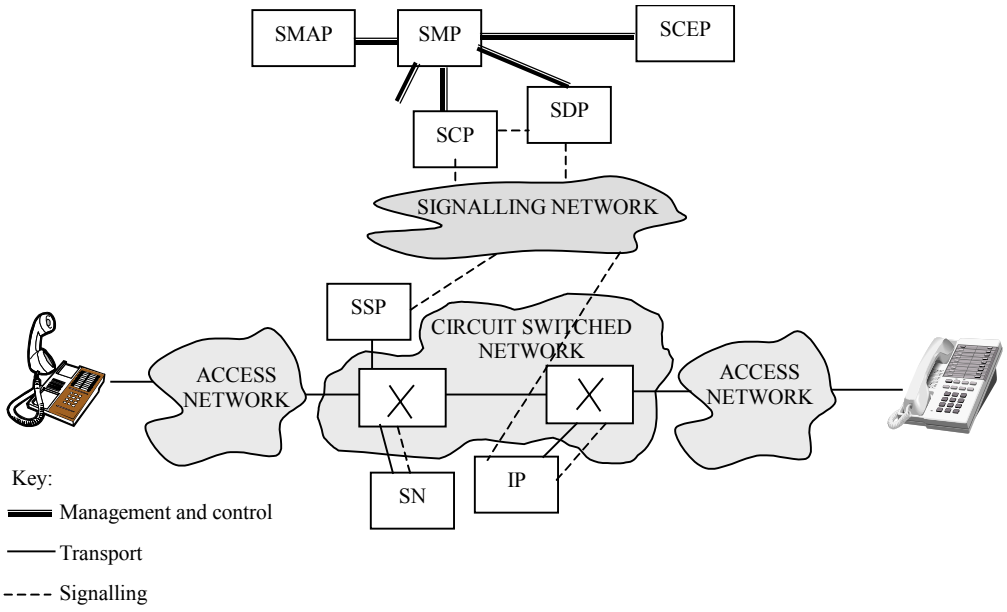


Figure 8-10 Intelligent Network architecture

In principle, each of the IN functional components (like e.g. SCF- Service Control Function, SMF – Service Management Function, SSF – Service Switching Function) can be managed by different providers. IN-based services involve resources that might be managed by different actors.

A selection of physical elements that can take roles as "stand-alone"/autonomous actors is depicted in Figure. Each of them can be considered to represent a separate actor or an element belonging to different companies. At least four types of interfaces could be identified: i) User-Operator, ii) Operator-Provider, iii) Provider-Provider, and iv) Operator-Operator.

8.1.11 References

[NMFBMP] *A Service Management Business Process Model*, NMF, Morristown, NJ, 1996.

[P610D4] P610 EURESCOM Project *Management of Multi-Media Services* Deliverable 4 "Management Framework and Methodology", 1998.1997.

[TINA-BM] *Reference Points and Business Model*, TINA-C, Version 3, June 1996

[TINA-SA] TINA Consortium: *TINA-C Service Architecture*, Version 5.0, 1997.

[TINA-CMC] TINA Core Team: *Computational Modelling Concepts*, Version 3.2, 17th May 1997.

[X.700] ITU-T Recommendation X.700 ISO/IEC 7498-4, *Management Framework for Open Systems Interconnection (OSI) for CCITT Applications, / Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework*, 1989.

[X.701] ITU-T Recommendation X.701 / ISO/IEC 10040, *Information technology - Open Systems Interconnection - Systems Management Overview*, 1992.

[X.702] ITU-T Recommendation X.702 / ISO/IEC 9595, *Common Management Information Service Definition* [*for CCITT Applications*], 1991.

[X.710] *Information technology - Open Systems Interconnection - Application Context for Systems Management with Transaction Processing*, ITU-T Draft Recommendation X.710, ISO/IEC Draft International Standard 11587, 1994.

[X.703] *INFORMATION TECHNOLOGY – Open Distributed Management Architecture,* ISO/IEC Draft International Standard, Draft Recommendation X.703, June 1996.

[X.711] ITU-T Recommendation X.711 / ISO/IEC 9596-1, *Information technology - Open Systems Interconnection - Common Management Information Protocol Specification*, 1991.

[X.720] ITU-T Recommendation X.720 / ISO/IEC I 10165-1, *Information technology - Open Systems Interconnection - Structure of Management Information - Part 1: Management Information Model*, 1993.

[X.721] ITU-T Recommendation X.721 / ISO/IEC 10165-2, *Information technology - Open Systems Interconnection - Structure of Management Information - Part 2: Definition of Management Information*, 1992.

[X.722] ITU-T Recommendation X.722 / ISO/IEC 10165-4, *Information technology - Open Systems Interconnection - Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects*, 1992.

[X.749] ITU-T Recommendation X.749 (08/94) - Information technology - Open systems interconnection - Systems Management: Management Domain and Management Policy Management Function, August 1994.

[X.901] ITU-T Draft Recommendation X.901 / ISO/IEC Draft 10746-1, *Information Technology - Open Distributed Processing - Reference Model - Part 1: Overview*, 1995.

[X.902] ITU-T Recommendation X.902 / ISO 10746-2, *Information Technology - Open Distributed Processing - Reference Model - Part 2: Foundations*, 1995.

[X.903] ITU-T Recommendation X.903 / ISO 10746-3, *Information Technology - Open Distributed Processing - Reference Model - Part 3: Architecture*. 1995.

[X.904] ITU-T Draft Recommendation X.904 / ISO Draft 10746-4, *Information Technology - Open Distributed Processing - Reference Model - Part 4: Architectural semantics*. 1995.

[X.641] ITU-T Recommendation X.641: *Information Technology – Quality of Service: Framework*.

[Y.100] ITU-T Recommendation Y.100: *General overview of the Global Information Infrastructure standards development*. 06/98.

[Y.110] ITU-T Recommendation Y.110: *Global Information Infrastructure principles and framework architecture*. 06/98.

8.2 Comparison between FAIN interface/reference-point descriptions and related work

Approach for describing reference points in FAIN could be compared to other models. We distinguish especially, TINA, TSAS, Parlay, OSA, SPAN, P1520 and MSF.

8.2.1 MSF

The MSF (Multiservice Switching Forum) is trying to consider open model for supporting heterogeneous switching technologies (ATM, FR, MPLS, IP..). It distinguishes between intra-system and inter-system interfaces considered as reference points. An intra-system interface defines the information exchange between the forwarding and control components within a network element. Intersystem interfaces handles the interactions across the network between components of the same layer. As MSF is not exclusively focusing on IP networks and as almost of the technical documents are not public, it may be difficult to compare it at the actual stage with FAIN architecture. However, we may need to evaluate its solutions in further development on control plane as more mature MSF architecture would be released.

8.2.2 IEEE P1520

It defines several interfaces:

- *V interfac*e for User level programming
- *U interfaces* exist between the value-added services level and network generic services level.
- The *L interface* provides programming interfaces between the network generic services level and the virtual network devices level (VNDL). The L interface defines the API to directly access and manipulate local device network resource states.
- the *CCM interface* is a collection of protocols that enable the exchange of state and control information at a very low level between a device and an external agent.

Mostly only L interface is being investigated and is still in progress. This interface may correspond to RP3 level. RP3 has not been fully specified until now in FAIN and is subject to further design specification in WP3.

Other interfaces: U, V may be relevant to FAIN, but there's no input from P1520 workgroup

8.2.3 Parlay, OSA, SPAN

The Parlay architecture (from Parlay Group), the Open Service Architecture (from 3GPP) and SPAN (from ETSI) share all a common framework based on describing Application Programming Interfaces (APIs) between networks and service providers. The Parlay architecture/specification is the most complete compared to OSA or SPAN. All of them address interfaces at framework level (client view), the generic call control service level, mobility, generic user interactions levels. Both Parlay and SPAN address connectivity managers, generic messaging. Only Parlay addresses interfaces at the framework service view with facilities on integrity and security management.

In OSA, the access to network functionality is offered by different Service Capability Servers (SCSs) and appear as service capability features in the OSA interface. The different servers interwork via the OSA interfaces. The service capability servers will serve as gateways between the network entities and the applications.

At the Framework Client view, the access/authentication phase is approximately in alignment with TINA and TSAS. However, these models support an new operation for terminating a service agreement between roles that is not available in TSAS and TINA. As making explicit the termination of the service agreements is considered important, we adopted this operation at the access part for FAIN reference points.

At the Generic User Interface Level, several operations are provided within Parlay, OSA, SPAN, to manage the user interface. These interfaces are not supported in TINA or TSAS. In FAIN, we consider that the user interface should reflect the activeness of services. In case of new active component is added to the service, the user interface may need to be updated subsequently e.g. to allow the user to perform more operations involving the new component. So we added in FAIN some interfaces at RP4 relating to the user interface. We have not adopted Parlay/OSA/SPAN operations as they relate generally to basic calls.

Parlay and SPAN address the issue of IP Connectivity Manager. We could relate this to TINA connectivity provider for non IP connections. Basically, they propose two sets of interfaces: QosMenu and Enterprise Network. The Enterprise Network set deals with setting up a VPN of IP-based pipes. The Qos Menu interface deals with selecting QoS for associated IP-based pipes.

In FAIN as we divide the connectivity provider into two entities: ANSP and NIP. Each of NSP and NIP concentrate on specific issues. In fact the Parlay connectivity manager could only relate to the NIP level. We're not dealing specifically with VPNs as unique solution based on IP. This makes FAIN framework more general than Parlay at this level, so we could propose for example a third set of interfaces relating to active facilities/networks.

The main difficulty to implement Parlay/OSA/SPAN is related to the mapping toward the underlying protocol used to build the experimentation. These models/APIs have in fact complex data structure. There's a need for a higher level of abstraction. In FAIN, we consider different levels of abstractions by separating ANSP and NIP.

According to Parlay, edge-of-network applications should also provide some interfaces of their own, which are used by the network-located service components as callbacks for notifications, event reports. The applications interact with the parlay framework by providing it with the pointers to these callback methods. The relationship between a Parlay application and the Parlay service components is thus markedly similar to the relationship between the service switching and the service control functions in Intelligent Network only that here methods are used instead of protocols and the services have a much broader scope of control.

What is necessary is the definition of a component object model for Parlay services. Once such model is available, any supporting architecture will be able to transparently take care of issues like replications, migration and activation. Services will then be executed in specialized container (virtual) environments where they will enjoy the services mentioned above plus access to the Parlay interfaces exposed by the underlying network.

In FAIN, we define the new functionality of ANSP that would reside between service providers and network infrastructure providers and provides the appropriate execution environments into which services developed by independent software vendors will execute. These execution environments may allow further economies of scale to develop as hundreds of services will be executed there. Service providers relying on Parlay will still have the option to keep service execution within their premises as the Parlay APIs are implemented over a DPE making the actual run-time location of the services insignificant. For services, however, that impose rigorous standards of performance, use of the operator-provided execution environments can be obtained. From the point of view of the network provider on the other hand, no additional risk is incurred due to the co-location of externally developed services with critical network equipment as exactly the same remote interfaces are used in both cases.

8.2.4 TSAS

The Telecommunication Service Access and Subscription (TSAS) model provides segmentation of interfaces relating to access and subscriptions. This segmentation is useful in FAIN for usability and scalability. However, when considering implementation issues, other aspects are discovered, that are important. Notions like User Agent or Provider Agent are considered to be important because they relate to real models. The notion of User Agent is stable and is been used in different frameworks for supporting customization and intelligence of the local system. In FAIN we have decided to specify the reference points in terms of segments, and to allocate these segments to some computational objects that would form the building blocks of the system. When considering the system at a certain level of abstraction, we could ignore this mapping and consider only the interfaces, but we consider that these computational objects are important as components of the system. This idea may be similar to the conclusion related to Parlay APIs, as a new object based component model is needed to be supported to help developing the system.

The TSAS specifications are not dealing with user interfaces or accounting issues that are important during subscription.

TSAS doesn't support explicitly service types, but relies on some internal trader interfaces. In our case we need to make these explicit as done in Parlay to enable interoperations between different implementations

TSAS does not support termination of service agreements, but these are important in an active network context as for each dynamic update of the active service, re-negotiation/termination of service agreements is needed.

TSAS is not dealing with aspects related to RP2 as it relate to the communication session.

In RP4 and RP1, we deal not only with access, but also the usage part, with some functionalities provided for enabling delegation, meta-level control, accounting and user interfaces. We have not explicitly adopted the notion of SAG (subscription access group) or the notion of session graphs, and so the subscription administration includes simple or groups of users.

8.2.5 TINA

TINA provides an important background for specifying reference points. It specifies mainly the Retailer and partially the Connectivity Service reference points. TINA specifications contain several objects and there are some redundancy between the interfaces. One of the objectives of TSAS was to improve the scalability of the specification and remove the redundancy using segmentation.

The RP1 has not been specifically addressed by TINA.

The RP4 is considered an adaptation of the Ret reference point to active network facilities with redefinition of interfaces, objects and responsibilities.

In FAIN the connectivity provider has been divided into the ANSP and the NIP and the RP2 is only concerned with interactions between ANSP and SP. No direct interactions are considered between SP and NIP.

The RP2 introduces the concept of active IP flows and execution sessions and virtual environments where the active flows should be processed. As the ANSP may be considered as the most important role for active networking by providing associated virtual networks where active code is been processed, it was necessary to add more control facilities between the NSP and the SP (and with the NIP to be addressed later). One of these control facilities is assumed to be the meta-control from SP to the NSP. This would enable some SP entities (interfaces defined at RP4) to do meta-control on the ANSP active-flow controller.

Meta-programming permits the separation of functional from non-functional code. Non-functional code resides at some meta-level, supervising the execution of the functional code. A system supporting a meta-level will allow the incorporation of meta objects that may obtain information about any base-level object and/or control its execution. The system enables the redefinition of interactions between base-level entities (e.g. method calls, instantiations) through their management at the meta-level.

One of the major advantages of using meta-managers is that the information obtained from the execution of a service can be gathered and combined with similar information handed over by other meta-managers in order to enhance the overall service management. This leads to the possibility of modifying the behaviour of the distributed service as a whole. The principle is to enforce the introduction of a control interface already during the design phase and to treat application management as an orthogonal aspect allowing management modules to be incorporated afterwards. It is thus possible to deploy new management functionalities a posteriori since management tasks are performed through the meta-level.

As stated in the enterprise model, active networking should allow delegation of responsibilities between roles. In a distributed object oriented environment, intermediate objects hide the details of complex system interactions. Unfortunately the interposition of the abstracting object prevents the target services from securely determining the identity of the initiator of the operation. All requests arriving at the target services appear to be the action of the intermediary rather than the true initiator. Delegation is necessary to be supported in FAIN enterprise model. This would allow the migration of management functionalities from NIP up to the consumer. In the case of meta-management of components between SP and NSP, delegation would allow these facilities to be shared with other roles (not only between NSP and SP) in certain cases by allowing delegation of component meta-management.

A delegation architecture adopted is concerned about three issues:

- - First, we could allow an intermediary to operate on other objects in a manner that reflects the initiator's identity as well as its own. A target server receiving such a chained request would see the privilege attributes of each participant in the chain.
- - Second, we need to extend the authorization model to allow target servers to make use of the distinction between initiators and intermediaries. Target servers may grant rights to principals acting as intermediaries on behalf of authorized initiators without granting rights for those principals to act on their own.
- - Lastly, we should allow clients performing operations to place restrictions on the uses of their identity in chained calls to protect itself by placing limitations on who may project its identity and to whom its identity may be projected. A client may simply allow or disallow use of its identity in a chained call. The two types of delegation related restrictions are *target restrictions* and *delegate restrictions* . Target restrictions set by a given client apply to all servers in a call chain that are down stream from the immediate client-intermediary pair. Delegate restrictions set by a given client limit who may act as an intermediary.

8.2.6 References

[Parlay] http://www.parlay.org

[OSA] http://www.3gpp.org/ftp/TSG_CN/WG5_osa/

[SPAN] http://www.etsi.fr

[P1520] http://www.ieee-pin.org/

9 GLOSSARY

Active Middleware Manufacturer

An *active middleware manufacturer* (AMM) produces platforms for executing active services (including the execution environments) to the network service provider.

The active middleware manufacturer is not a genuine FAIN business role since it is not directly involved in the main FAIN business process.

Actor

An *actor* is an entity of any kind that owns, uses, supports and/or administrates parts of an (active networking) system. An actor may be a person, company, or organisation; it may interact with the networking system directly or via a hardware or another system.

The FAIN actors are the following entities:

- the providers, i.e., companies operating networks (by managing the respective hardware and software resources), offering services over the network (employing their own resources or those of the network operators), or proffering repositories containing active code and components. The FAIN providers occur in their roles as FAIN service provider (SP), FAIN network service provider(ANSP), FAIN network infrastructure provider (NIP), and FAIN service component provider (SCP).
- the users, i.e., individuals or companies which use the proffered connectivity and services. They correspond to the FAIN role of the consumer (C).
- the manufacturers, i.e., companies that produce the software and hardware components used by the providers. They are reflected in the roles of the FAIN service component manufacturer (SCM), the FAIN active middleware manufacturer (AMM), and the FAIN hardware manufacturer (HM).
- additional provider companies that offer auxiliary services, like the broker (BR) role with its mechanism for the distribution of software components, provider and domain information and the retailer (RET) role where access mechanisms for (FAIN) services are offered.

Broker

The *broker* (BR) is a business role that provides fair and equal access to information about how and where to find certain services, service components, administration domains, and stakeholders in a networking system. Thus, it collects, stores and distributes information about (1) available services, (i.e., those offered by the service providers), (2) service components (i.e., code offered by the service component providers, in particular active code components, which may be injected into the active network), (3) the administrative domains in the active networking system, and (4) the instances of the business roles, that are active in the administrative domains. The Broker may show a variety of auxiliary features, such as service property descriptions, service version management, service discovery, etc.

The broker is no genuine FAIN business role and will not be detailed further in the FAIN enterprise model.

Business Role

A *business role* is acted by an actor and refers to the specific behaviour, properties and responsibilities assigned to an actor with respect to an identifiable part of a networking system and the related activities that are necessary to keep this part of the system operational.

FAIN identifies consumer, service provider, network service provider, and network infrastructure provider, and service component provider as core business roles. The roles service component manufacturer, active middleware manufacturer, hardware manufacturer, broker, and retailer are considered not to be in the centre of FAIN interest.

Cache

A program's local store of response messages and the subsystem that controls its message storage, retrieval, and deletion. A cache stores cacheable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server that is acting as a tunnel. [HTTP]

Consumer

The *consumer* (C) is the end user of the active services offered by a service provider. In FAIN, a consumer may be located at the edge of the information service infrastructure (i.e., be a classical end user) or it may be an internet application, a connection management system, etc. Depending upon the selected service, a consumer may need to inject active code into the network which may possibly be retrieved from a service component provider, or may be otherwise obtained from a service component manufacturer. In order to find the required service among those proffered by the service providers or in order to obtain the needed code components from the service component provider, the consumer may contact a broker and, e.g., use its service discovery features.

The consumer role is often split into the end user and the subscriber. For FAIN, the subscriber is considered an ephemeral role upon which the project will not elaborate.

Corporate User

A *corporate user* is an end user belonging to or making use of a corporate network.

Corporate users get specific service profiles offered by the corporate network. In order to be able to offer such services, the corporate network (customer) may have to subscribe to certain services of service providers.

End User

The *end user* is a subtype of the consumer role. It is the one who makes use of a service. The end user is or represents an individual, a group of individuals (e.g., a household) or a company. It either is a human being or a software application. In the later case, the end user is usually termed user application.

See also: subscriber.

Enterprise Model

An *enterprise model* consists of the business model addressing business and administrative issues and the corresponding business processes addressing technical issues.

Hardware Manufacturer

A *hardware manufacturer* (HM) produces programmable networking hardware for the network infrastructure provider.

The hardware manufacturer is no genuine FAIN business role and will not be detailed further in the FAIN Enterprise Model.

Network Infrastructure Provider

A *network infrastructure provider* (NIP) provides managed resources (bandwidth, memory and processing power) to network service providers. it offers a platform to the network service providers who can build their own execution environments, and proffers basic transmission infrastructure which may be based upon traditional transmission technology as well as emerging ones (both wired and wireless).

The network infrastructure provider, together with the network service provider, corresponds to the role of the ”connectivity provider” in the TINA business model.

Active Network Service Provider

An *active network service provider* (ANSP) provides network services and execution environments (one or more) for active service components to service providers (e.g., it features component provisioning, security, and management capabilities). Together with the network infrastructure provider, it forms the communication infrastructure.

The network service provider, together with the network infrastructure provider, corresponds to the role of the “connectivity provider” in the TINA business model.

Provisioning

Provisioning is the setting up of a service, including, if necessary, the setting in place and configuring of hardware and software. It results in creating new instances of the objects that represent the service.

Retailer

The *retailer* (RET) provides an integrated view of services to the customer and presents a single point of contact to the customer in both its subroles as end user and subscriber. Thus, it has to be capable of negotiating and managing the end user contract (handling authentication and security issues as well as accounting and billing, etc.) and it has to allow the service life cycle management by the user (subscription and customising of a service, administrative interactions, etc.). Retailers ensure ease of access and quality of service guarantees to consumers.

Role

A *role* is an expected behaviour of each of the actors in a particular context.

Service Component Manufacturer

A *service component manufacturer* (SCM) builds service components and active code for active applications and offers them to service providers, consumers and service component providers in appropriate form (e.g., as binary or as source code). Descriptions of the service components provided by the SCM may be published via a broker service.

> The service component provider is not a genuine FAIN business role since it is not directly involved in the main FAIN business process.

Service Component Provider

The *service component provider* (SCP) provides active code that may be used by the consumer and the service provider. A service component provider acts like a repository. It may publish its components via a broker.

Service Provider

A *service provider* (SA) composes active services from components delivered by a service component provider or obtained form a service component manufacturer, deploys these services in the network and offers them to the consumers. The services provided by a service provider may be end user services or communication services. A service provider may federate with other service providers in order to build more complex services. Descriptions of the offered services are published via the broker service. Access to the service provider is provided by the retailer according to the TSAS specifications.

Service Role

A service role is a refinement of a business role used to further structure service descriptions.

Stakeholder

Stakeholder are an individual, team, or organisation that are affected in some way by a certain (networking) service. Stakeholders are instances of business roles.

Subscriber

A subscriber is a subtype of the consumer role who interacts with a service to obtain the effect of a service. It holds the contract with the service provider or its representative and subscribes to a service on behalf of its users.

> See also: end user.

User

The term *user* is synonymous with the term end user. Thus, it refers to the entity that makes use of a service that is provided by a service provider.

