OPENING THE NETWORKS WITH PARLAY / OSA APIS: STANDARDS AND ASPECTS BEHIND THE APIS

Ard-Jan Moerdijk[1] and Lucas Klostermann[2]

Draft version, to be resubmitted to IEEE Communications Magazine.

[1] Ard-Jan Moerdijk, Chairman 3GPP CN5 (Joint API Group), Ericsson Eurolab Netherlands, e-mail: Ard-Jan.Moerdijk@eln.ericsson.se
[2] Lucas Klostermann, Product Manager, Ericsson Eurolab Netherlands, e-mail: Lucas.Klostermann@eln.ericsson.se

1 ABSTRACT

The past years have shown an enormous increase in efforts to open up telecom network functionality for application development. This means that applications can access core network functionality by means of open standardised APIs based on open technology.

One value of opening up of the network is the emergence of new business models where applications can be developed and provided by enterprises outside the traditional network operator domain.

Combined with the fact that applications can be built with standard IT technology and tools, new innovative applications will hit the market with development cycles shorter than ever shown before.

This paper provides an overview of the Parlay / OSA initiatives on the specification of a set of open, standardized APIs. Furthermore, the paper outlines some architectural aspects that are implied and implicitly contained in the Parlay / OSA specifications, but are nevertheless critical for understanding the implications when opening up the network by means of Parlay / OSA. The paper concludes with a characterization of applications that can be built using Parlay / OSA.

2 INTRODUCTION

In today's telecommunication networks, applications and services are a part of the network operator's domain and are primarily built using Intelligent Network (IN) technology. This approach is well suited for simple mass-market and carrier class applications. However with the emergence of mobility and IP, easy creation and rapid deployment of innovative applications that combine different features and critical enterprise data, become a challenge beyond the capability of IN.

In the past initiatives like TINA (see Ref 1) tried to address this challenge, but the results have so far largely remained at research level. Nonetheless, inspired by the TINA work, different groups like Parlay (Ref 2) and JAIN (Ref 3) continued with efforts to develop APIs, based on open technology that allows applications to access core network functionality. This is referred to as the opening up of the network and will make the telecom functionality residing in the network, accessible for a large developer community. This will lead to fast creation of innovative applications. Also, by opening up its network, a network operator can invite 3rd parties, operating outside their own domain, to develop these innovative applications. This will provide the network operator with a means to extend their service portfolio.

These APIs naturally form the interface between what is referred to as the application layer or Service Network on one hand and the core network on the other hand. Applications are logically positioned in the Service Network and can be deployed independent of the core network and access network that the end-user is using. This means that instead of the current approach where applications are tied to one specific network (fixed, mobile, IP) applications can be accessed and used from different types of networks or domains. This is called the vertical approach versus the horizontal approach. The benefits of a horizontal approach for the operators and end-users are evident.

The APIs this paper will mainly be focussing on, were originally defined within the Parlay Group (Ref. 2) and standardized in the context of 3GPP (Ref. 4) and ETSI (Ref. 5). Parlay is not considering any specific underlying network technology or architecture, whereas 3GPP and ETSI focus on 3rd generation mobile and fixed networks. The Service Network concept and the collection of APIs are called Parlay within the Parlay group and Open Service Access (OSA) in 3GPP and ETSI. Within the OSA concept, so-called Service Capability Servers offer the API interface implementation and hence these nodes can be found on the border between the Service Network and the core network, see Figure 1. In section 4 the architecture is outlined in more detail.

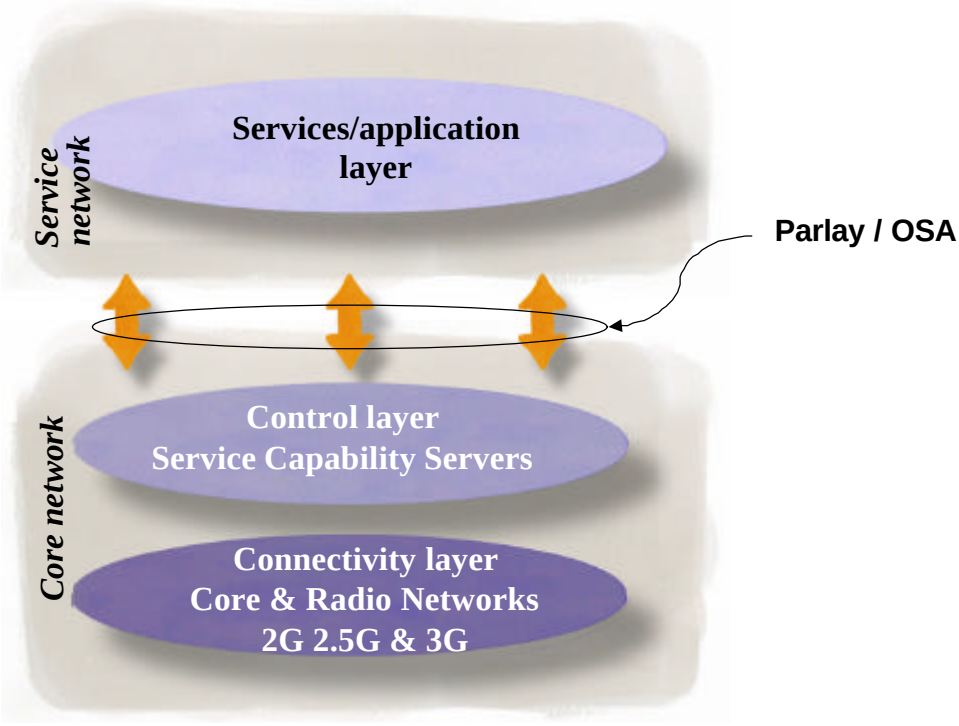

Figure 1: logical network architecture showing the Parlay / OSA APIs on the interface between the Service network and core network. The Parlay / OSA APIs are offered by Service Capability Servers (SCSs) and applications access network capabilities through the APIs and get accessed via the Parlay / OSA APIs.

In the following sections the paper gives an overview of the history and current status of the Parlay / OSA API work. Next we look at the implied architecture of OSA and outline some concepts that are key to OSA, but also somewhat implicit in the specifications. Finally the application side of the interface will be addressed and some typical example applications will be given.

3 PARLAY / OSA API DEVELOPMENT AND STANDARDS: THE JOINT API GROUP

In 1998, the Parlay group was established as an industry forum with initially 5 member companies. The group's goal was to develop APIs that would allow enterprises to access core network capabilities. The aim was that these APIs would be network independent in order to allow the development of applications that do not rely on minute network details and particular network intricacies. This independence enables a smooth evolution of core network technologies while the applications stay untouched. It also allows easy migration and portability of applications to other network technologies. Note that in addition to these "network agnostic" APIs there are also APIs that are intended for a particular network technology and allow the applications to access this network's specific functionality as well. The first release of Parlay was realized within one year and work continued as Parlay 2.

Around the same time as the work on Parlay 2 commenced, 3GPP and ETSI started working on APIs to be used for service development for 3rd generation networks. Having a large overlap in scope, with 3GPP focussing on UMTS and ETSI SPAN on fixed 3rd generation networks, the benefits of working together were promptly recognized and all the work has been done jointly between ETSI and 3GPP from the outset[3]. It was then realized that the Parlay APIs can be used in the 3GPP / ETSI context as well. Therefore the relevant Parlay APIs were brought to 3GPP and ETSI for standardization in the context of what was called at that time Open Service Architecture (OSA). Nowadays the OSA acronym has been re-termed Open Service Access. In addition to the relevant Parlay APIs new ones were defined. Furthermore, mappings from the API to specific network signaling protocols have been produced. This latter work was primarily to show that the API could be implemented using the 3GPP core network and is provided as informative recommendation.

Needless to say, a lot of emphasis was placed on aligning the APIs of Parlay and 3GPP / ETSI. To achieve one developer community Parlay, 3GPP and ETSI decided to collaborate towards their common goal and to date the work on the OSA/ Parlay APIs done jointly in what is called the Joint API Group. In addition to the joint work between these three organizations, there is an informal collaboration with the JAIN community (Ref. 3), mainly with the purpose to align the Call Control APIs[4]. At this moment 3GPP has produced Rel.4, while ETSI and Parlay have released a common draft specification called ETSI OSA version 1 / Parlay 3.0. These three specifications are fully aligned. One can also say that the specifications are different exposure channels of the same work! Furthermore, within JAIN there is work ongoing to develop Java versions of the APIs, called JAIN SPA (Service Provider Access).

The figure below outlines the relations between the various versions and releases of the specifications.

[3] The specific working groups responsible for API development are 3GPP CN5 and ETSI SPAN12.
[4] The JAIN Call control 1.0 APIs were based on the Parlay 2.0 Call control APIs but some additional modifications were introduced. Work has been ongoing in order to align the JAIN Call Control with the Parlay / OSA Call Control.

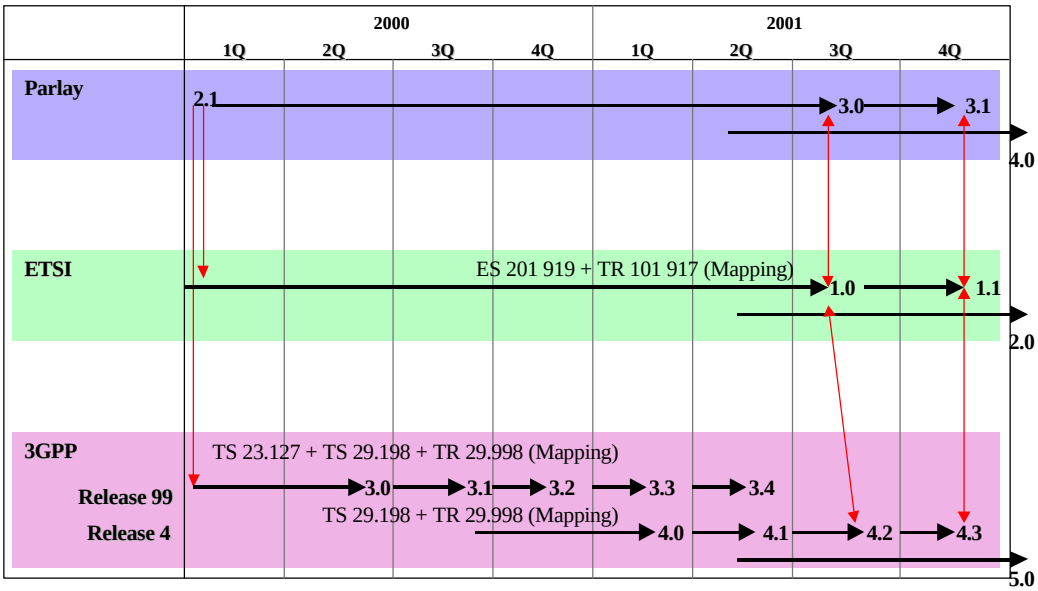

Figure 2: Relation of the different specifications

For the reader of the different specifications the concepts used may still be a bit confusing, caused by different use of terminology. In the Parlay specifications the term Service is used for the collection of interfaces that provide a specific function, while 3GPP and ETSI use the notation SCF (Service Capability Feature) for the same concept. In the 3GPP context the term service usually means an end-user service.

In the remainder of the paper the term application will be used for end-user service and the term Service Capability Features will be used instead of the Parlay term Service.

For Parlay version 3.0, 3GPP Release 4 and ETSI version 1, the specification set consists of 12 parts (see Refs 6,7,8) :

part #	SCF	Description	Comments
1	General	Contains the introduction and methodology used.	
2	Common data	Generic data definitions, used in other parts.	
3	Framework	Defines the infrastructure capabilities like authentication, SCF discovery, SCF registration, fault management, etc.	
4	Call Control	Defines the call control family with capabilities ranging from setting up basic calls to manipulating multimedia conference calls.	Multi-media and conference call control are not part of 3GPP Rel.4 OSA as this functionality is considered to be outside the 3GPP Rel. 4

			network.
5	User Interaction	SCF to obtain information from the end-user, play announcements, send short text messages, etc.	
6	User location / User status	SCF to obtain location and status information.	
7	Terminal capabilities	SCF to obtain the capabilities of an end-user terminal.	
8	Data session control	SCF to influence data sessions.	
9	Generic Messaging	SCF for access to mailboxes	not part of 3GPP OSA Rel. 4 as this functionality is considered to be outside the 3GPP Rel. 4 network.
10	Connectivity Management	SCF for provisioned QoS	not part of 3GPP OSA Rel. 4 as this functionality is considered to be outside the 3GPP Rel. 4 network.
11	Account Management	SCF to access end-user accounts	
12	Content based Charging	SCF to charge end-users for use of applications / data.	

Note that apart from the APIs within the scope of the Joint API group, additional APIs are being developed within specific Parlay working groups. This allows to work on a specific task with a focussed group and try out new ideas. When work of such a group is successful, the Joint API Group will adopt these additional APIs.

For the time period after June 2001, the focus of the joint group is on extension of the APIs like for example multi-media aspects, the extension of content based charging and inclusion of policy management and presence and availability management (PAM). In addition to these functional aspects, the joint group will work on the support for an XML version of the APIs, which is to be published in parallel to the current IDL version.

In the next sections the paper will focus on surrounding aspects of the Parlay / OSA APIs.

4 BEHIND THE APIS: ARCHITECTURAL ASPECTS

In this section aspects of the OSA architecture are described. Firstly, the logical entities are explained, secondly, the aspects related to deployment in the network, and finally some key architectural aspects like security and scalability are outlined. As this paper is intended as an overview, a comprehensive and exhaustive look at all architectural aspects is out of the scope.

4.1 THE LOGICAL ARCHITECTURE

Figure 3 shows the logical architecture behind OSA. The logical entities to be distinguished are: Applications, Application Servers, Service Capability Servers (SCSs), the Parlay / OSA Framework and core network elements.

Applications, deployed on Application Servers, which can be any standard IT platform, use the capabilities defined in Parlay / OSA, provided by Service Capability Servers and offered to applications through the APIs: the SCS implements the server side and the application the client side of the API. Communication between the application and the SCS is done using standard IT middleware infrastructure. Service Capability Servers (SCSs) are thus logical entities that implement the API (that is, the interface classes of the Service Capability Features) and potentially interact with core network elements. These may include Home Location Registry (HLR), Mobile Switching Center (MSC), Service Switching Point, etc. As such, an SCS serves as a proxy or gateway to the core network. Application Servers could be either in the same business domain as the Service Capability Servers or in a different domain.

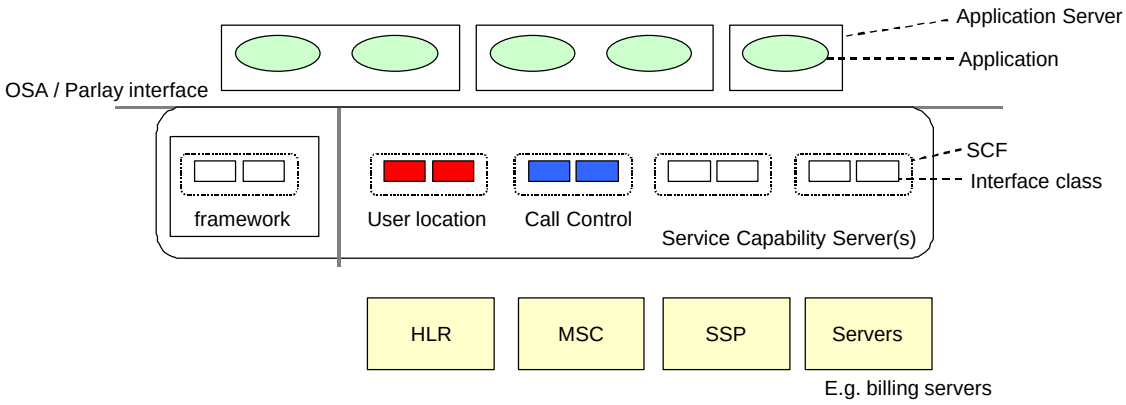

Figure 3: Overview of the logical entities involved with Parlay / OSA

A Service Capability Server may implement multiple SCFs, although one usually talks about a specific SCS per SCF, e.g. the Call Control SCS.

As an SCS is a logical entity it does not need to be implemented as a separate box. For instance the Content Based Charging SCF might very well be directly offered by the charging and billing server itself. This is further detailed in section 4.2.

The Parlay / OSA Framework, which is also shown in Figure 3 offers the Framework capabilities to applications. This entity is the essential part in enabling openness, and making it possible to go beyond traditional IN when it comes to openness, discovery and integration of new features, as described below. The following section explains this crucial part of Parlay / OSA on a high level.

4.1.1 The role of the Parlay / OSA Framework

The Parlay / OSA Framework builds upon ideas from TINA (Ref 1) and provides controlled access to the Service Capability Features (SCFs), which in combination with distributed technology supports flexibility in application location and business scenarios. Furthermore the Framework allows multi-vendorship and even the inclusion of non-standardized SCFs, which is crucial for innovation and service differentiation.

The Parlay / OSA Framework consists of a family of interfaces with the core part consisting of:

- Trust and Security Management: authentication of domains.
- SCF Registration: registration of new SCF to the Framework.
- SCF Factory: creation of new SCF instance.
- SCF Discovery: discovery of SCFs provided by the operator

In addition there are also Framework interfacess for:

- Integrity Management: load balancing, fault management, heart beat
- Event notification: notifications for specific events (e.g. registration of new SCS).
- Contract management: management of contracts between different domains (e.g. between application provider and network operator).

To explain the core part of the Framework it is beneficial to look at a complete business process where a new SCS is installed and an application starts using the capabilities offered by this SCS. The scenario is outlined in Figure 4. Suppose that the SCS implements the Multi-Party Call Control SCF. Three different stages can then be distinguished:

1. Registration

During this stage a new SCF is added.

First the SCS contacts the Framework and requests its the Registration interface. The Framework then returns a reference to it. These are steps 1-2 in Figure 4. Next, the Multi-Party Call Control SCS uses this interface to publish its SCF type, in this case Multi-Party Call Control and its capabilities, for example how many parties per call this specific implementation is able to handle. Furthermore, the SCS will supply the Framework with its own reference (called Service factory[5]) via the Registration interface. This is step 3 in Figure 4. At this point in time the Framework and the SCS know each other[6].

2. Setup of Service Agreement

In this phase the conditions under which an application is allowed to use the SCS are established.

From this point on, the Framework operator is able to enter the conditions under which an application is allowed to use the SCS. One of these conditions could be that application X is only allowed to use a maximum of 4 parties per call. All of this information is captured in what is called a Service Agreement and the information is entered via a management system.

3. Run-time communication establishment.

At this stage the run-time communication between the application and the SCS is established.

[5] The factory pattern is a general design pattern and allows the Framework to request the SCS to create an SCF instance for each application.
[6] It is important to outline that the framework will also allow registration of non-standardised SCFs: in principle one can thus publish any API that is useful for applications using the above mechanism.

The final step occurs when an application contacts the Framework and uses the Discovery interface to find out what SCFs are supported (steps 4-6). Suppose that the application wants to use the Multi-Party Call Control SCF and it wants to be able to control calls with a maximum of 4 parties per call. The application will therefore ask the Framework via the Discovery interface, to return all SCF implementations that fulfill this request of 4 parties per call. The Framework then checks the Service Agreement to find out whether the application is indeed allowed to use the Multi-Party Call Control SCF. If this is the case, it will return a list of implementations of Multi-Party Call Control that are able to handle at least a maximum of 4 parties per call. For instance it might be that there are two Multi-Party Call Control SCSs, where one is able to handle 8 parties per call and another one 6 parties per call. As the information about the capabilities of the SCS is also provided to the application, it can then select one from the list (Step 7 in Figure 4). In this case it is most likely that the second SCS might be cheaper to use. Next, the Framework will request the specific SCS (via the Service Factory interface) to create an SCF instance that is to be used by this application. The Framework will also send the information about the conditions under which the application may use this SCF, for example, the maximum of 4 allowed parties per call (Step 8). The SCS will now create the SCF instance and return a reference to the Framework (Step 9). The Framework will then return it to the application (Step 10). From this moment on, the application is able to use the Multi-Party Call Control SCF.

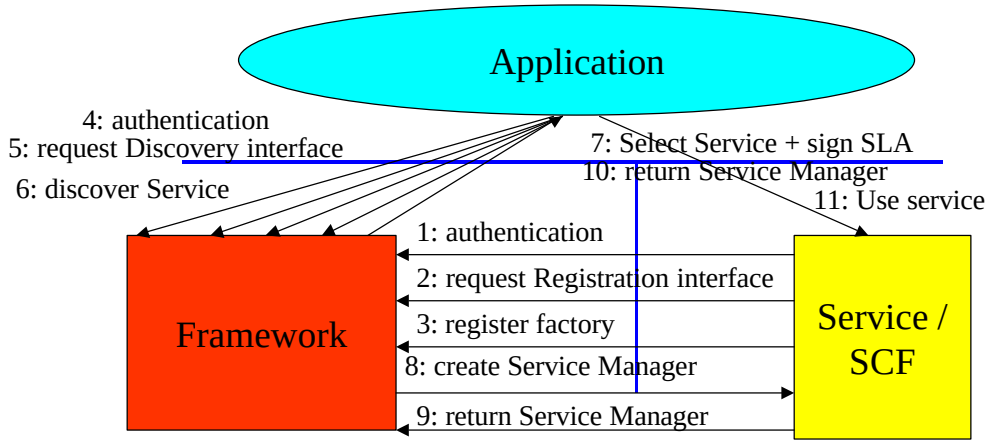

Figure 4: Service registration and discovery

Steps 6 – 10 are to be repeated for each SCF the application wants to use.

Also note that Parlay / OSA supports the authentication steps (1 and 4) to be skipped for entities within the same domain, for example, the case where an operator adds a new SCS to its Framework or in case the application is in the same domain as the Framework and SCSs. It is a general misconception that Parlay / OSA is focussing only on 3rd party applications.

4.2 PHYSICAL ARCHITECTURE

There can be numerous physical entities offering the functionality that the API provides the open access to: switches, Interactive Voice Response systems (IVRs), Home Location Registries (HLRs), mobile positioning systems, GPRS Support Nodes (GSSN), billing servers, etc. As indicated, Service Capability Servers are logical entities and the specifications do not prescribe whether SCSs should be separate boxes in the network. In principle an SCS can be deployed as a stand-alone node in the network or it can be deployed as a core network node. For example the user status SCS could be deployed directly on the HLR or it could be deployed on a separate stand-alone node that uses the CAMEL protocol for communicating with the Home Location Register (HLR).

In the case where the SCSs are really deployed on separate stand-alone nodes, one does distinguish a physical Service Enabler sub-layer in the core network, see Figure 5. One solution here is to provide all API implementations within one physical node. Usually this is referred to as the physical OSA gateway. This gateway has protocols / interfaces to all the various core network entities. The other approach is a distributed approach. In this case the OSA gateway node contains the Framework and maybe a few Service Capability Server components, but the rest of the Service Capability Servers run on different nodes. This means that the OSA gateway is a logical gateway and API implementations can run on distributed nodes, i.e. the different network entities provide their own APIs. This alternative can of course also be a mix of SCSs as separate nodes and core network nodes providing the SCF implementations.

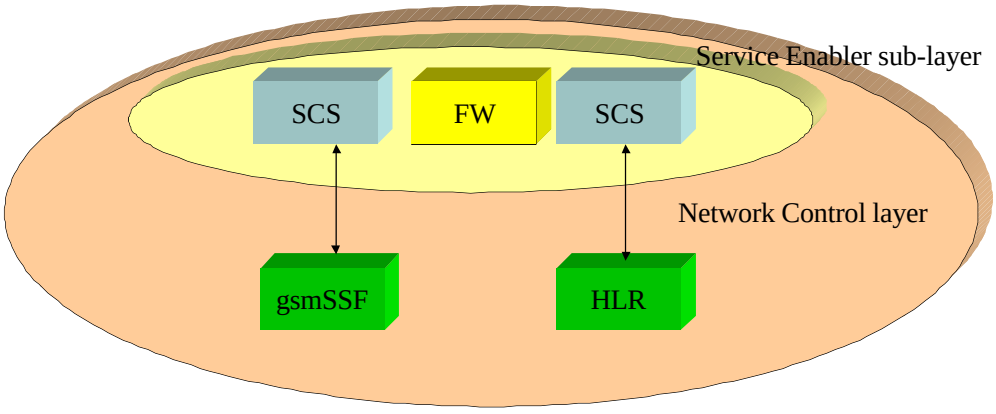

Figure 5: One specific network configuration where all Service Capability Servers are deployed on separate nodes. In this configuration one can distinguish a Physical Service Enabler sub-layer in the network architecture.

All deployments are in principle possible, provided the appropriate middleware infrastructure between different nodes is in place. However, there are cases where it is not desirable to deploy SCS software directly on a core network node. It is especially in cases where end-user triggers are stored dynamically based on e.g. location of the user or the signaling or processing load. In these cases it would mean that all possible nodes in the network where the user trigger might be located, must be able to setup communication with the application, in order to be triggered when the user initiates an event of interest. For example, a mobile subscriber can be attached to any serving switch in a mobile network, based on location of the subscriber. Setting up a call origination trigger for example, would require updating all SCSs deployed on the switches or it would mandate communication between one switch and any other switch to which the mobile subscriber can possibly connect and originate calls from.

One can conclude that in practice there will be a mixture of all indicated deployment scenarios. There will most likely be a gateway node offering the Parlay / OSA Framework that includes the SCF registration interface and some core SCSs. Additional SCSs can then run on separate nodes, and be registered to the Framework in the gateway node. This also allows phased introduction of SCSs. Furthermore, as the Framework is a critical entity, it should be deployed on a node that provides the usual telecom carrier class characteristics like 99.999 % availability.

4.3 SECURITY CONSIDERATIONS

In order to allow applications from other domains to use the SCSs in the network, it must be ensured that the communication between the Application Server and the SCSs is secure. As mentioned in Section 4.1.1, Parlay / OSA includes authentication between domains on API level. This means that there are mechanisms defined for authentication between domains on application level. Furthermore, these mechanisms rely on well-known encryption techniques and are also extendable with new techniques. However, apart from authentication on the application level, one also has to ensure that the underlying communication is secure and that access to capabilities is controlled. These two topics will be addressed below.

4.3.1 Secure communication

Secure communication can be realized with standard IT technology, firewalls, SSL, IPSec, etc. It is also not expected that Application Servers will communicate directly over the internet with the SCSs, mainly for security and performance reasons.

A more likely scenario is that dedicated connections are to be used. Thus it should be ensured that only Parlay communication is allowed over the dedicated link and also that the applications can be reassured that nobody else is able to address the application peer objects present at the gateway. This can be achieved by introducing a firewall that only allows the specific communication protocol (e.g. CORBA) to pass and, by using SSL to prevent other Applications from “stealing” object references.

Another security issue is how the integrity of Service Agreement can be guaranteed. For this we have to focus on the service properties and the Service Agreements.

4.3.2 Controlled access: Service Agreements and service properties

In Section 4.1.1 it was briefly outlined that during registration an SCS publishes its capabilities. Based on these capabilities, the Framework operator is able to enter the conditions under which an application is allowed to use the SCS. The example we looked at was that the Multi-Party Call Control SCS was able to support a maximum of 8 parties per call, while the application was allowed to use only a maximum of 4 parties per call. The capabilities of an SCS are described by the Service Properties in Parlay / OSA. In the following discussion we will briefly outline this concept and show how this is used to control access to the capabilities offered by an SCS.

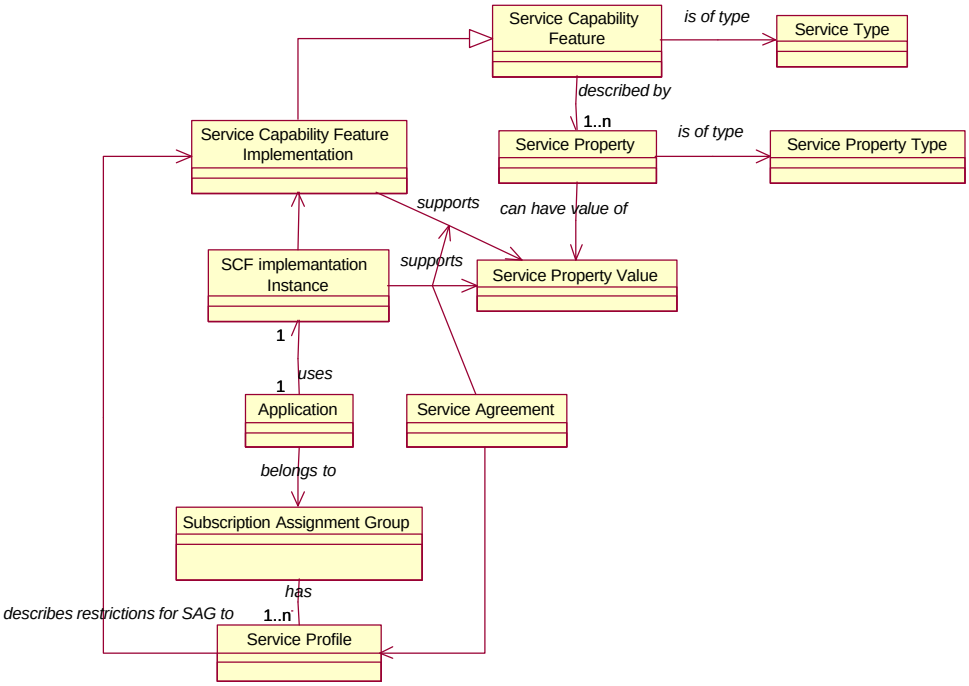

Figure 6: Model showing the concepts related to Service Agreements.

In Figure 6 a UML model outlines the concepts related to the Service Properties and the so-called Service Agreement. The model shows that an SCF is of a certain type (e.g. Call Control, User Interaction, etc.) and is described by a set of Service Properties. Service Properties are of a certain type (e.g. String) and can have a certain value. General properties include the name of the SCF, version, name of the SCF implementation, version of the SCF implementation, etc. Furthermore, each SCF has it's own properties like for instance Call Control has a property to indicate the maximum number of parties in one call session.

For a specific SCF implementation the properties are assigned values and this allows the SCS to promote its capabilities and restrictions. It also allows applications to know about these, without knowing the exact underlying technology that supports these capabilities or leading to certain restrictions.

The concept of a Service Profile is used to specify what values within the range of supported service property values, applications are allowed to use. For example it might be that although the SCF implementation supports up to 8 parties per call, the application is allowed to use only up to 4. A run-time instantiation of a Service Profile is called a Service Agreement.

The final concept shown in the UML model, is the Subscription Assignment Group. The Subscription Assignment Group contains Service Profiles for all the SCFs needed by the applications within such a group.

In Section 4.1.1 it was shown how SCF implementations are registered to the Framework, how they can be discovered by applications and how the SCF implementation instance is created in case the application decides to use the SCF. At registration the Framework thus receives all the supported property values of the SCS. The Framework should then store these values in some kind of database. Each time a new application is introduced, it should be assigned to a Subscription Assignment Group. After the discovery procedure, when the application selects the SCF instance to use and signs the Service Agreement, the Framework propagates the values of the service properties the application is allowed to use, to the SCS. The SCS then creates an SCF instance that is one-to-one related to the application and it should also check the application does not violate the Service Agreement.

This clarifies the power of the Service properties and Service Agreement when it comes to security. One can easily indicate what applications are allowed to do, e.g by creating the Profile in such a way that the application is e.g. not allowed to use certain methods or to alter certain parameters. At this moment the Service Property related data-types are somewhat complex. Work is ongoing to allow for specification of the Service Properties in XML.

4.4 QUALITY OF SERVICE

Service Agreements can also be used for QoS aspects. Service Properties for e.g. guaranteeing a minimum number of sessions per second or maximum number of allowed sessions per second, will soon be introduced.

4.5 SCALABILITY

Another major non-functional architectural aspect is scalability. Parlay / OSA allows for scalability on different levels.

First of all, it is possible to add more SCSs and distribute the load from different applications over multiple SCSs. The way this works is that at the Service selection phase, the Framework diverts one application to one SCS and another application to another one. Of course in the case where the specific SCS depends on network entities, this scalability is limited by the capacity of the core network.

Furthermore, with underlying middleware like CORBA it is possible to distribute load on session basis without the application being aware that different sessions run on different processors.

Finally, in some of the APIs it is possible to add multiple application call-backs to the SCS, which allows the SCS to distribute the load of different sessions over different application instances that could run on different servers. However, there are at present no means for applications to indicate a distribution policy.

4.6 PERFORMANCE, AVAILABILITY, ROBUSTNESS, REDUNDANCY

As the interface between the telecom networks and the datacom networks, the SCS of course has to provide the usual telecom characteristics like 99.999 % availability. The main challenge will therefore be to provide applications with the same characteristics.

5 THE APPLICATION LAYER

5.1 APPLICATION SERVERS.

Besides the Service Capability Servers much is also expected for ease of application development from the Application Servers themselves. From the previous sections it is clear that some kind of distribution of SCSs is very likely and therefore middle-ware like CORBA is needed. However, application servers allow the hiding of the distributional aspects and offer application developers a standard programming environment in a favorite language. Application Servers are also the most suited environments for the work done in the JAIN community. Here one focuses on the specification of the APIs in Java. At this moment most of the Parlay APIs are thus specified in Java.[7]

[7] One exception is Call Control, here JAIN has a version of its own, called JCC/JCAT. Much effort has been spent to align the JCC and Parlay / OSA Call Control and it is to be expected that a fully aligned API will become soon available.

SCF functionality on Application Servers may even be captured in development components (e.g. Java Beans) that allow application developers to link in and tune the needed components. It is foreseen that these Service Development Kits (SDKs) further speed the application development process. One crucial aspect of these development kits is that it should be easy to link in new components that implement the APIs in the target language, and also communicate with the specific SCS.

5.2 SAMPLE OSA APPLICATIONS

OSA allows applications to combine capabilities (SCFs) that were not easy to combine with technologies like IN. Setting up calls can be easily combined by requesting the status of a user, and requesting a reservation for charging purposes. The first type of OSA application that one can distinguish is thus an application that combines different SCFs. A typical scenario is shown in Figure 7. The example is an application that the subscriber can use to compile a list of people (buddy list) and request to be notified when these people are available or e.g. when they get in the neighborhood. Once everyone is available, the application could set up a multi party call, for example, within a given time frame, or at a certain time connect all parties who are available.

The required OSA interactions would look like the following:

1. The non-traffic part: the application contacts the Framework, gets authenticated depending on whether it is in a different domain and then requests for access to the user location /user status SCS and the call control SCS. The Framework will check that the application is allowed to use the requested SCSs and if so will request the different SCSs to create an SCF instance that is to be used by the application. The references of the SCF instances will be returned to the application.

2. An end-user is now able to access the application, e.g via the internet or WAP and set-up the buddy lists.

3. When this list is complete, the application contacts the user location / user status SCS and requests to be notified when the people within the list become available (turn their mobile on).

4. The SCS will initiate installation of triggers in the network in order to be notified when people from the buddy list switch on their mobile.

5. Each time one of the people from the list turns his/her mobile on the User location/ user status SCS is informed.

6. The User location / user status SCS notifies the application.

7. In case the application finds out that enough people are available, it informs the user, either by popping up a window, or by using the call control SCS to place a call with an announcement or by using the user interaction SCS to send an SMS message.

8. The user decides to request the application to set-up a conference call with all parties available.

9. The application could check whether the end-user still has enough credits by contacting the Charging SCS.

10. The application contacts the call control SCS to initiate the conference call.

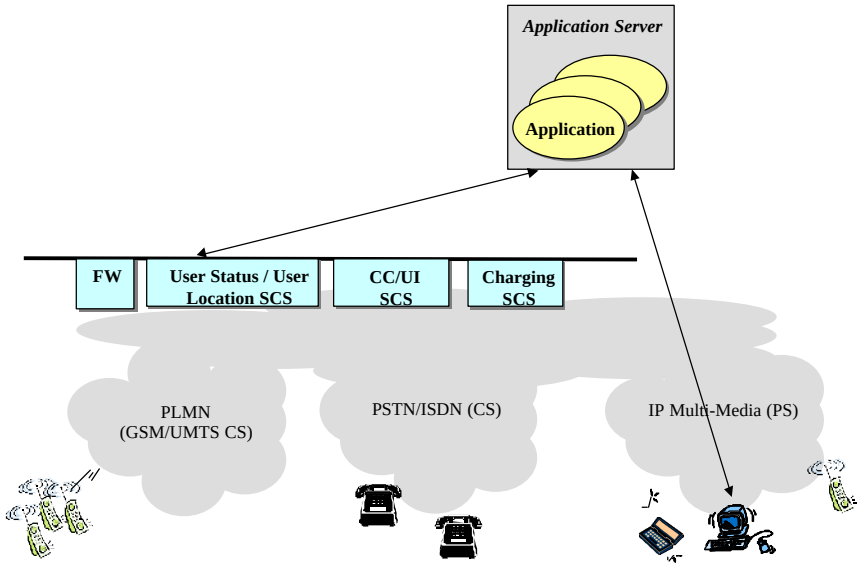

Figure 7: application combining different capabilities

OSA also allows a combination of capabilities with data not traditionally found in the network. The second type of application is then one that combines network capabilities with enterprise data. One characteristic of OSA applications is thus the data-owner. A typical example would be an application that can direct the end-user to the location of e.g. the nearest ATM machine. The location of the ATM machines is usually data that is not found in the traditional telecom network and also often “belongs” to the enterprise in question, e.g. the bank.

With the network convergence achieved by means of OSA, it is also possible to deploy SCSs in the enterprise domain, e.g PBX with Call Control API, etc. Enterprise applications can thus be located in the enterprise domain and the end-users can then access these applications when they are within the corporate network and also, even when they are outside the enterprise network, e.g. when they are travelling. To set this up would require an agreement between the application provider (enterprise) and the network operator. In this case the application is not part of the standard application portfolio of a certain operator, but is only accessible to the group of corporate users. We will refer to these types of applications as corporate applications. Of course, it might also be the case that when present inside the corporate domain, the corporate application is accessed by means of other mechanisms than OSA, outside the corporate environment however, OSA is used to access the application. A typical example would be the enterprise mail system, accessible also by dialing into an application that is able to send the mail messages to a text2speech engine via the OSA SCFs.

One extreme case here is the mission critical application that uses some SCSs of the network operator, and is used only by a very limited set of corporate users. A typical example would be an application that can inform a worker in the field closest to the location of a certain event. What would be needed here is an application that is able to access the user location SCF to track the mobile stations of the workers and is also able to access e.g. the Call control SCF to place a call or the User Interaction SCF to send an SMS message.

6 FUTURE AND EVOLUTION OF PARLAY / OSA

The set of APIs developed within Parlay / OSA has grown substantially within a few years. This comes with a drawback and in this case the accessibility to the specifications has suffered somewhat. Therefore, the Parlay / OSA specifications should be made more readable and more simplified. A high-level description is needed and the Parlay / OSA data-types are especially rather complex. There is also the lack of a clear model that describes the relations of the different data-types related to registration and discovery. However, this has been recognized and is under further development.

A next item that is under development is the further integration of Parlay / OSA with IP-Multi media (e.g. the 3GPP all-IP based core network, see Ref 9) and the possibility to link in with other SIP based networks. At this moment the Joint API group is working on these aspects and it is foreseen that some specific APIs like multi-media call control will need some evolution. One important aspect to consider here is that Parlay / OSA and SIP based approaches (e.g. SIP servlets) are not competing with each other, but rather complementary, see also Refs 10 and 11.

Work is also in progress in the area of policies. The central idea here is to define policies that can be checked by the different SCSs and define a way to introduce and manage these policies. This work might influence the definition of the Service Agreement and Service properties.

Finally, Parlay / OSA will also soon support XML / Webservice based versions of the APIs in order to allow the webservice developer community to use services offered by the capabilities in the network. This work might have some impact on the Parlay / OSA architecture in general as mechanisms like Service Discovery are already supported with webservice technology. However, it would open up the networks to a new developer community and allow for multiple implementation options. Nonetheless real time applications for session control are better off using CORBA based technology, but the rest of the applications could be based on webservice or CORBA technology.

7 CONCLUSION

Parlay / OSA is becoming a powerful toolbox that provides the necessary openness and flexibility for fast application development. By means of the joint API standardization working group, one standard is achieved and one common developer community is addressed.

Parlay / OSA is flexible when it comes to actual deployment in the network. For instance it allows the introduction of new SCFs on the fly, even ones that are not standardized. We have also shown there are several options to implement SCSs and deploy them in the network. Furthermore, Parlay / OSA provides the necessary security for the core network to be opened-up for 3rd party application providers.

We have outlined some typical OSA applications. Today there are already SCS products and applications that prove the OSA concept and show the integration between telecom and datacom, without the end-user being aware of this.

In order to become really accessible for the common IT development community, the Parlay / OSA specifications should, in certain areas, be simplified and made more readable. Also it is important to link up with XML based initiatives (Webservices, SOAP).

8 ACKNOWLEDGMENT

The authors gratefully acknowledge Erik van der Velden, Christophe Gourraud, Jörgen Dyst and Chelo Abarca for input and helpful discussions. Furthermore we would like to thank Musa Unmehopa, Andy Bennett and Paulus Karremans for reviewing the paper and finally Kimberleigh Anderson for a grammar review.

9 TERMINOLOGY

3GPP	3rd Generation Partnership Project, body responsbile for UMTS standards.
CAMEL	Customized Applications for Mobile network Enhanced Logic, IN standard for mobile networks.

CAP	CAMEL Application Part, protocol in the CAMEL standard between Service Control Point and Mobile Switching Center.
CORBA	Common Object Broker Architecture, specification defined by the Object Management Group (OMG) for application interoperability independent of platform, operating system, programming language.
ETSI	European Telecommunications Standards Institute, body responsible for (European) telecommunication standards.
GPRS	General Packed Radio System, mechanism for packed switched communication in mobile networks.
HLR	Home Location Register, database for permanent storage of subscriber data. The HLR keeps track of the locations and status of the subscribers.
IDL	Interface Definition Language, technology independent language for specifying interfaces.
MAP	Mobile Application Part, protocol for signaling and database communication in mobile networks.
MSC	Mobile Switching Centre, node responsible for switching in 2nd generation mobile networks.
OSA	Open Service Access, term use in 3GPP and ETSI for APIs that open up the network.
SCF	Service Capability Feature, OSA API standardized in 3GPP.
SCS	Service Capability Server, server implementing one or more Service Capability Features.
SMS	Short Message Service, allows mobile subscribers to send and receive short text messages.
SSL	Secure Socket Layer, security protocol defined by the Internet Engineering Task Force (IETF) that provides communications privacy over the Internet
UML	Unified Modeling Language, technology independent language for specifying, visualizing, constructing, and documenting the artifacts of software systems.
UMTS	Universal Mobile Telecommunications System, successor to GSM.

WAP	Wireless Application Protocol, enabler for web like applications on mobile terminals.
XML	Extensible Markup Language, universal format, defined by W3C, for structured documents and data

10 REFERENCES

1 See for instance: TINA-Consortium, TINA Service Architecture Version 5.0, 1997, available at http://www.tinac.com/

2 Parlay Group, http://www.parlay.org/.

3 JAIN, http://java.sun.com/products/jain/index.html

4 3GPP, http://www.3gpp.org/.

5 ETSI, http://www.etsi.org/.

6 The Parlay Group, “Parlay specifications 3.0”, http://www.parlay.org/specs/

7 3GPP “Open Service Access”, Application Programming Interface, 3G TS 29.198 + 29.998: http://www.3gpp.org/ftp/TSG_CN/WG5_osa/Specs/

8 ETSI “Open Service Access”, Application Programming Interface, ES 201 915 + TR 101 917: http://docbox.etsi.org/tech-org/span/Open/Span12/osa.html

9 Lieve Bos and Suresh Leroy, Towards an all-IP-UMTS based system architecture, IEEE network, January /February 2001.

10 Pailer R., Stadler J., A Service Framework for Carrier Grade Multimedia Services using PARLAY APIs over a SIP System, ACM Wireless Mobile Internet Workshop WMI2001, Rome, Italy, July 2001.

11 Stephane Desrochers, Roch H. Glitho, Kindy Sylla, Roch Glitho et al, “Experimenting with PARLAY in a SIP Environment: Early Results”, IP Telecom Services Workshop 2000 (IPTS 2000) Atlanta, September 2000.