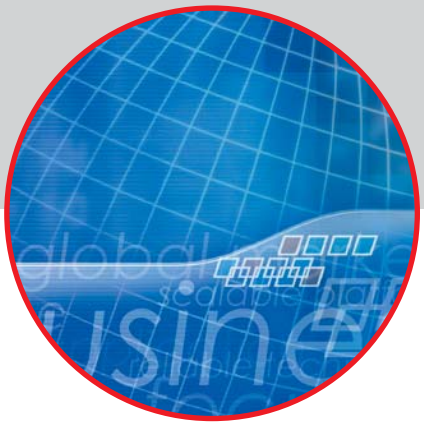


The Death of Network Intelligence?

R. Minerva, C. Moiso

ABSTRACT - The goal of the paper is to demonstrate that: a) the paradigm of Network Intelligence has to be adjusted in order to cope with the challenges posed by next generation networks and services (e.g., 4G, grid computing, ...); b) there is a need for a flexible service platform tailored for multimedia personalized services; c) there are technologies that can shape networked systems and services in a new fashion. The service platform should accommodate for services executing on top of different networks (from personal to global networks) and for a highly personalized control of services. There is the need to bypass the argument of "Stupid vs. Intelligent Network" linking together the edge, the terminal, and the network intelligence. These requirements will be the drivers for designing a simple, highly programmable and adaptive service architecture that exploits the capabilities of the Networked Intelligence. Such architecture is strongly based on Service Discovery, Network Identity, and APIs. A stratification of APIs will provide different views of the underlying heterogeneous network resources. The major technical challenges are identified and analyzed in the paper.

API: *Application Programming Interface*

Introduction

In the debate about Next Generation Networks, a rough reality is emerging: the major application for NGN still remains VOICE. At the same time, the Next Generation Control Architectures (for NGN, 3GPP, and 4G alike) still reproduce existing Telecom architectures on top of an "all-IP" infrastructure. This because a) Operators tend to reapply the "known" business models and solutions in the new context; b) established Vendors push for

a "smooth" evolution of their product lines; c) new comers in the "control platform" field disregard the fact that they are proposing old services with a new technology (e.g., SIP).
The victim of this quarrel is the "Network Intelligence". The reason is that in an IP dominated world "the network" has to be STUPID (1). Nevertheless, it is surprising how much network intelligence is in an IP infrastructure (e.g., DNS, DHCP, SIP Proxy, Policy servers) and how much it is neglected. If 4G networks are to be "all-IP" then there is the need to make the IP black boxes programmable, but how? In other words, what is the new service architecture?
There is not a crystal ball for guessing killer applications of the future nor it is possible to suggest new services (that's quite a sensitive information!), the investigation focuses, instead, on some communications metaphors that could bring in requirements for new services and features. Consequently, a new service architecture is sketched out identifying good technologies that are applicable from now on,.and some issues that the research should cover.

A look Ahead into the Future

How do you imagine the communication world in ten years from now? So far it has been described in terms of tons of sensors, high capacity wireless access networks, faster than light wireline access networks, lots of distributed computing nodes (grids?), multi-modal terminals, wrappable/foldable high definition screens. All very real, but these seem to be technical solutions. If they are solutions … then what are the problems they try to solve? What services are envisaged? What the customers' requirements… From the service perspective the innovations aim at the Multimedia and Value Added service markets over different types of networks. Those networks span from Personal to Global networks.
In order to get requirements we have used some metaphors and services scenario. The first one is the Universal Remote Control metaphor. The terminal is used for interacting with the surrounding environment by pointing it towards sensors. The other metaphor is the Virtual Assistant, i.e., a software capability able to filter and control events, information and communication when the user is at home (home networking) in specific areas (WLAN) and under the coverage of public (mobile) networks.
In both cases we have considered multimode terminals with the possibility to roam to the best servicing network.
Some other metaphors and scenarios can be seen in (2). The requirements we got do confirm requirements identified within (3). They fall in these areas: federation; user awareness; context awareness; personalization of services; advanced user interfaces, adaptation of services to networks capabilities and terminals .

The importance of Business Models

The paper benefits also from an extensive analysis carried out internally to TILAB on the importance of Business Models in the definition of viable architectures. The starting point was considering what business models the industry is going to use for future networks. Essentially two: the internet and the telecommunications models. Are they adequate for the new perspectives opened by the upcoming technologies? Are they sufficient? Are they applicable?
The two models currently used are not totally applicable because they are poor with respect to the service scenarios that we have considered (centralized and distributed intelligence working together). There is the need for a new model: the Networked Intelligence Business Model. This model is strongly based on the federation of functionalities: each functionality, each resource, each terminal is seen as a node of the network (or better a resource that can be networked) that offers functions to other "nodes" in order to compose and deliver a valuable service to the clients. Different Providers can offer specialized and intelligent

resources, information, contents, functions, chunks of networks or simply the service logic that will bring things together (i.e., that controls the networked resources). In this way different "service implementations" can occur within the grid of networked nodes. The important point here is that terminal nodes, edge nodes and network nodes are ready to share information and processing with each other.

The business model is an open one leaving the possibility to existing and new actors to have a role in the communications business. The competition will be based on the ability to provide valuable services leveraging the assets and the capability of each actor. In addition the business model allows for the aggregation of actors in order to propose new value chains for services and solutions.

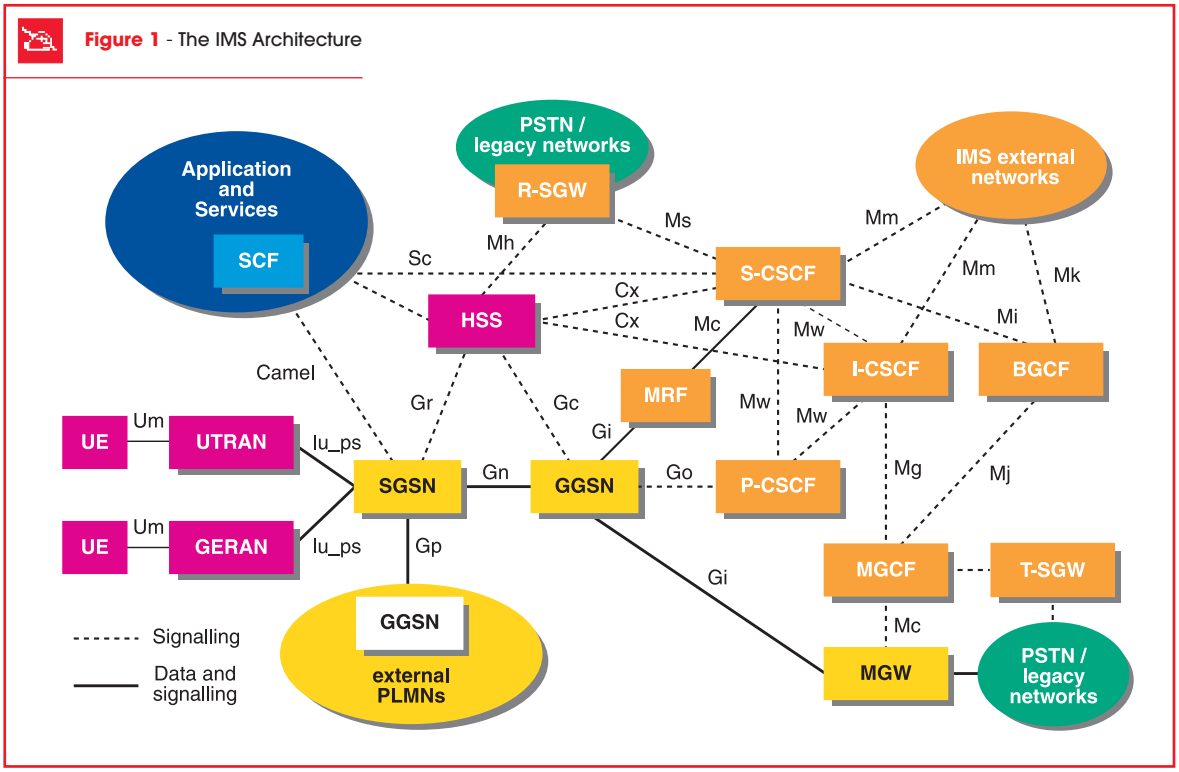
So the questions: "What about the network (or edge) intelligence of this new communications environment?" and "Where do the service control mechanisms reside?" are out of scope. Intelligent mechanisms will be distributed according to the Networked Intelligence business model.

The Need for a New Service Architecture

The architecture supporting the Networked Intelligence business model should encompass a few basic components and rules for service composition and programming. In the paper the term **Networked Intelligence** is used opposed to the identification of a specific place in which the intelligence should "occur". The requirement on simplicity of the architecture is a stringent one. The 3GPP focuses on both circuits and packet domains. The circuit one is based initially on traditional techniques and then will gradually move to packet switching aiming at the possible convergence with the packet domain. For the packet domain, the IP Multimedia Subsystems IMS architecture is defined (see ■figure 1) (4).

It extends the GPRS functionalities and introduces the abilities to support session-based services and levels of QoS. The IMS architecture identifies a sort of extended repository of user data named HSS, and the structure of an Open Service Architecture. Some major concerns are:

Figure 1 - The IMS Architecture



- the need to invest first on circuit platforms for supporting advanced services and then to reinvest in order to migrate towards a packer infrastructure;
- the service architecture definition confuses the usage of a protocol (i.e., SIP) with the definition of a software architecture (i.e., OSA/Parlay);
- the IMS proposes an uncertain Business Model from Operators point of view;
- the focus on session based services neglecting other classes of services.

These characteristics lead to the caution of Mobile Operators to deploy UMTS and IMS in particular. Many established Operators are considering the empowerment of current GPRS infrastructure (EDGE solutions) for providing data services leaving the circuit domain to support traditional voice services. The video communication seems to be a small market yet not too much worth for huge investments. In addition the data services to be delivered to users in the short/medium term seems not be "session based" (but mainly supporting vanilla client - server relations) so that investing in SIP has to prove yet as a good move.

The IMS architecture could also be interpreted as a way for Service Providers (e.g., Virtual Mobile Operators) to have the Operator to invest on the transport infrastructure (a sort of commodity) and let them to invest in the valuable service layer. As seen, the IMS architecture is strongly based on SIP. This protocol is also capable of requesting and dealing with QoS requests. Unfortunately, SIP and QoS mechanisms are put together in the functions of the CSCF (encompassing QoS control and Session control). This means that non session-based services have not an easy access to QoS functions. There is a strong requirement to decouple the QoS functions from the session function. The QoS functions must be provided in a general way and independently from the protocol used to control the conversational services.

The IMS architecture of 3GPP is very complex in certain aspects and poor on the service architecture definition. It is full of options, so much that it will be very hard to converge into a viable platform. In any case the definition of the IMS is (so far) the best attempt to introduce a new control architecture for all-IP networks.

On the other side, the broadband market is taking up. The users like an internet approach, and they demand for a few things: more bandwidth, multimedia content, more mobility and interoperability between different networks (i.e., Mobile and Fixed), and Safety and Security. In addition there is the emerging requirement to provide connectivity with Quality of Service (i.e., this means essentially lower downloading times, better streaming, and the possibility to voice and video communicate). In contrast to requirements, the definition of the broadband architecture (5), (6) has focussed on communications services leaving outside many simple information services and the peer to peer communication.

In this context, the Triple Play approach (i.e., the possibility of providing a bundle of voice, video and data services) for broadband is now surprisingly following the indications stated by 3GPP. The reasons are simple: IMS is an architecture that gives control functionalities to the network (through SGSN and GGSN or their broadband equivalent: Border Gateways), it is based on SIP, it can support levels of Quality of Service, it could be integrated with existing control systems (e.g., the IN infrastructure), it allows the Operator to offer a smooth interaction with the PSTN services, and it supports multiple accesses (e.g., WLAN) and multi-mode terminals.

With all the caution possible for predictions, it seems that the Mobile and Broadband environments will blend making (at least for the advanced customer) the Mobile and Broadband Operators undistinguishable.

Some Challenges to a New Paradigm for Networked Intelligence

It is a long time that the industrial community is looking for a viable Service Architecture able to provide information, content, and conversational services on circuit AND packet networks. Typically, the Telecom industry unravelled the concept of Service Platform in two ways: 1) Telecom horizontal platforms (e.g., Service Control Points of Intelligent Network): with (limited) programmability of services and service components; 2) closed vertical platforms – supporting a single class of services, and difficult to extend. On the IP side, the control resources (e.g., routers, but also servers) are intrinsically closed, and usually not focused on the service creation (e.g., autonomous systems providing well established services).

The paper advocates an approach that tries to combine the possibility and the richness of horizontal platforms with the effectiveness and lower cost of vertical solutions. The goal is to keep the future service architecture simple and slim allowing for a progressive enrichment of the functions.

The starting point is to identify a few functions that are common to the major solutions under development today, to integrate them, and … keep the architecture as simple as possible.

Requirements for a Lightweight Software Architecture for New Services

Requirements for a Lightweight Software Architecture for New Services

Many new multimedia services will not be conversational. So the architecture should be able to support different classes of services and eventually to allow a mix of their functionalities.

Req 1: to design an architecture that is not oriented to a single class of services.

In highly distributed platforms there is the problem of how to identify and associate to the service instances the right services/ resources/controllers that provide the networked functionalities. In plain terms there is the need for abstracting them as service component and for introducing efficient mechanisms for service component discovery and allocation.

Req 2: the service platform should have registration, discovery and allocations mechanisms for exploiting the full potentiality offered by a networked environment.

The envisaged services will be so many that the platform should be scalable and programmable in the sense that new components could be added under the need of services. The platform should be so flexible to readily encompass and integrate the new components without requiring a heavy reworking of existing components and services.

Req 3: the service platform should support the service development and deployment, through the assembling of service components, and be able to readily and easily integrate new service components by means of composition and integration of functions.

In order to play, to exchange messages or to share file and to guarantee level of trustiness and protection, the Provider needs to know who the user is, (that is Authentication, Authorization and Accounting functions). In addition the collection and interpretation of User Data and preferences will be needed in order to personalize services.

Req 4: the service architecture should support Network Identity mechanisms.

Req 5: the service platform should be able to access user data scattered all over the networking environment.

As said the service platform should be able to support higher levels of cooperation between different actors maintaining for each individual company a well recognized domain:

Req 6: the service platform should allow the federation and the brokering of functionalities and data offered by different "entities".

Many services over an IP infrastructure would have an advantage if the underlying networks do provide dynamic and on-demand QoS. On the other hand, they could adapt their behaviour according to the characteris-

tics of the available network resources and terminals (i.e., the execution context).
Req 7: there is a need to have interfaces for controlling the underlying resources of the networks in order to achieve end-to-end QoS;
Req 8: the service platform should be adaptive to the different characteristics of the networks used (context-awareness).
The relationship between the "service components" and the underlying networked resources cannot be bound to a single signalling protocol. The service architecture is "by nature" multi-protocols in order to encompass existing and future signalling means.
Req 9: the service platform is able to support, integrate and use different control protocols.
In the following sections the paper analyzes some technologies that could satisfy the previous requirements. Many of them are not focussing on communications services and they pose some technical challenges in order to be used in a different context:
Technical Challenge #1: *How a Web Services Architecture (encompassing the UDDI mechanisms for distributed registrations) is adequate as a horizontal (and programmable and extensible and layered) framework for building communication services.*
Technical Challenge #2: *how to devise a technical solution for the User Representation in the network (encompassing the User Profile and Network Identity) that makes the Customer free to choose the best Service Offering of the market according to his/her needs.*
Technical Challenge #3: *how to define and demonstrate a viable framework for programming and orchestrating IP services on demand.*
Technical Challenge #4: *how to establish SOAP as the Signalling Protocol, pushing for highly distributed and programmable control architectures.*
Technical Challenge #5: *how to develop an open Service Brokering models that allows users to access to the best service offering, and allows service providers to offer services to customer with the minimum mediation of other parties.*

The paper will try to discuss these technical challenges aiming at demonstrating the applicability of the solution within the context of the Networked Intelligence.

Service Oriented Architectures

The IT industry has worked a lot on service architectures. Its latest contribution is a technology that is quite promising in terms of programmability, and composition: Service Oriented Architectures.

Web Services

The classical IT model for the service architectures is centred on the client-server paradigm. This approach yielded to the definition of IT platforms and related standards such as CORBA (7). Recently the focus is on the definition of a architecture that allows the offering of "services" outside of a specific domain using mechanisms of the Web. The service is not anymore a Web Page, but it could be the result of some processing (e.g., access to data base, or a map calculation,...). It come out that many Companies could export services to a wider community. The SOAs and in particular the Web Services Architecture allow for a better offering of IT services over the Web infrastructure.
The Web Services Architecture is using very simple mechanisms: software components can publish their interfaces; specialized servers make those interfaces (a sort of advertisement) available to applications. In this way published interfaces can be used to compose services, and components could expose multiple interfaces to accommodate for personalization of services. The approach differs from CORBA, DCOM, RMI because it is based on simple Web protocols.
The Web services architecture paradigm has found the right attention even in the W3C context and in the grid community becoming one of the features of OGSA (8).
The Web Services architecture uses the http protocol for requesting the execution of a

remote procedure. The call and its parameters are formalized using the XML language. The standardization of this proposal yields to the definition of SOAP (i.e., Simple Object Access Protocol (9)). It is a text protocol able to support an RPC mechanism usable all over the web. By means of SOAP an URL (Uniform Resource Locator) it is not anymore just a access point for Web Pages, but it becomes the access point for an IT service.
In addition, thanks to the usage of XML, SOAP is independent both from distributed processing platforms (such ad DCOM and CORBA) and from specific programming languages (as Java).
Many standard bodies have tried to provide the Web Services Architecture with a more complete functional architecture.
Currently there are languages for the service description, some mechanisms for accessing to the services, and functions for the discovery of available services so that it is possible to dynamically figure out what services are available and who the provider is.
The standards that describe the mentioned functions are:

- SOAP, it is the fundamental mechanisms for requesting the execution of a service and for receiving the results of the processing.
- WSDL (Web Services Description Language (10)), it is the language to be used for the description of services' interfaces, their methods, the input and output parameters and the different data types that can be used. The language is analogous to the IDL language of CORBA, but it is richer and more extensible;
- UDDI (Universal Description, Discovery, and Integration (11)), it is the mechanism that supports the publishing and the search of the available services (and their provider). It is as a sort of broker between the several providers and the users of the services.

Figure 2 depicts the basic scenario of the Web Services Architecture.

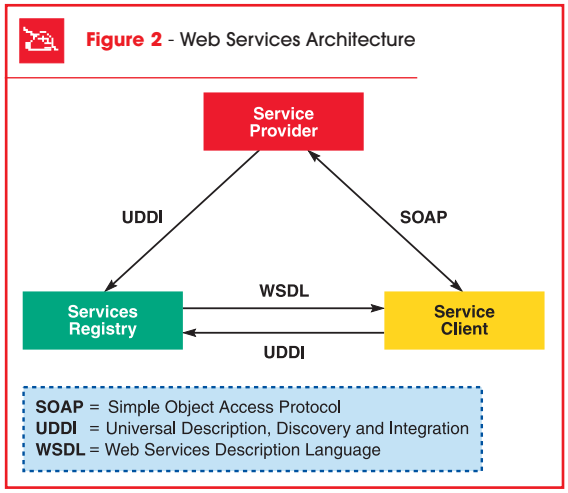


Figure 2 - Web Services Architecture

The UDDI is the enabling server for supporting a sort of service market place. It keeps track of all the registered services. Then client applications access to the service functions. If mechanisms for Network Identity are introduced (12) then the support to really personalized services could be granted. The user would access to only a part of available services and related interfaces, but that part is highly tailored on needs, terminals, subscriptions, and willingness to pay of that specific customer.
As seen the Web Service Architecture is a sound foundation for building highly distributed architectures, it is a fundamental part of the promising approach of the GRID (13) computing. It shows an elegant simplicity in the overall architecture, and it can be used also for communications services and components. A service platform based on Web Services Architecture can start small, i.e., providing a small number of components and interfaces (a sort of vertical solution), but it can scale up to manage many standardized and de facto components (i.e., providing communications, information, and other capabilities). In order to apply Web Services Architecture to the communication environment there are issues to be consolidated and solved:

- There is the need for Notification interfaces in order to support also an event-notification

paradigm (beyond the client – server one)
- Some services have a transactional nature and there is a need to support transaction
- The UDDI server should be resilient and efficient in order to provide high availability. It must also be extended so that the relationship between registered services and their availability can be monitored (it would be pitiful whether the UDDI server dispatches a reference to a non existing service).
- Security mechanisms are to be added in order to guarantee the secure usage of services
- AAA functions are to be added in order to monitor who, how, when and how much has used the service.
- Management of the Network Identity (it intermingles with previous points). Users are to be presented with their own services, supporting an easy way of accessing the services (e.g., single sign on) and personalized interfaces and functions;
- Federation of different UDDI servers. There could be the case that a specific UDDI server registers the services available in a particular domain (e.g., the Operator Domain). In order to provide a full range of services and functionalities, the UDDI servers could exchange info so that they can federate the offering of services making easier for application developers to compose services spanning over different domains.
- …

For these reasons we are "picking cherries" from other solutions and architectures that show a good similarity with the Web Service Architecture. In particular we consider OSA/Parlay (14) and Parlay X (15).

OSA/Parlay

Even if the Parlay initiative was born as a Telecom Operator initiative, we deem such architecture as pertaining to SOA. In fact OSA/Parlay is designed for enabling the development of services by discovering and aggregating components. OSA/Parlay presents functions that are helpful for handling communications and secure interworking of components. These functions are the registration and discovery of service components, the authentication of the applications invoking a service component, the authorization to invoke a specific service component, the handling of configuration data associated to a specific application and the instantiation of service components according to them, the management of the interactions among service components and applications, by monitoring conditions on fault, loading, etc.

The Framework is the key element backing up such functions. The Framework must be contacted by the applications in order to access a Parlay/OSA service component (named SCF, i.e., Service Component Feature). It handles the phases for authentication, authorization, and service component instantiation, according to the data stored in application profiles. Summarizing, the principal functions provided by a Framework are:

- Secure, controlled and accountable access to the Services
- Incremental introduction of new Services through the Service registration process
- Management of the integrity of the whole Parlay/OSA system (i.e., Applications and Service Components), such as fault handling and load control.

OSA/Parlay has a broader definition of mechanisms facilitating the deployment and operation of the architecture. The main differences are in handling the relationship between the Registrar and the components (i.e., the web services). An UDDI server mainly provides functions for the registration and the discovery of service components (i.e, Web Services). Additional functions, such as authentication, authorisation, instantiation and configuration are handled by different mechanisms (e.g., SAML for authentication/authorization, WS-Security for secure invocations, OGSA for instance handling). A comparison among the

Parlay/OSA model and the Web Services model can be found in (16).

Parlay X

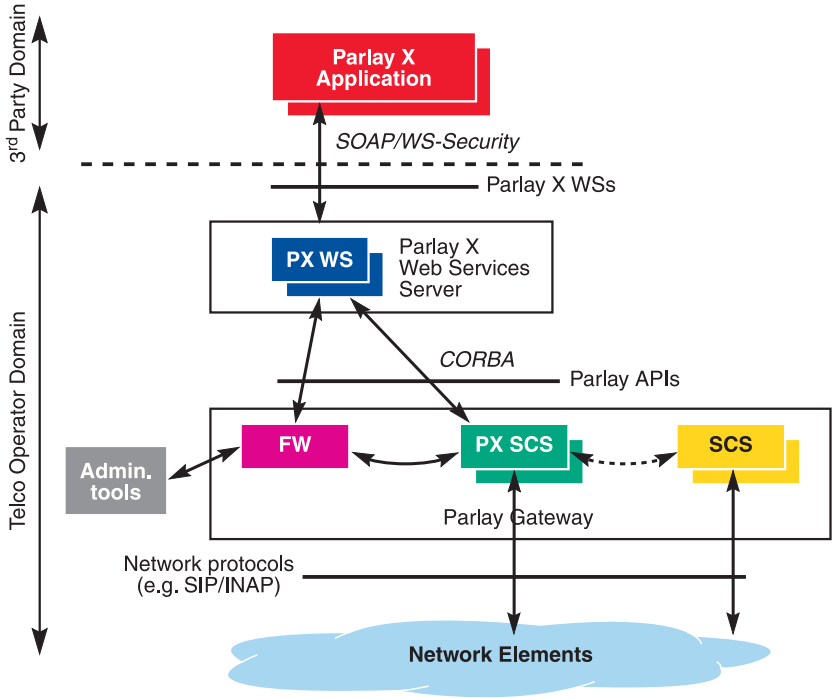
Parlay/OSA solutions are converging towards Web Services: Parlay X, a new set of APIs, based on Web Services, was recently defined, in order to simplify the usage of Telco Service Components by application developers with very limited knowledge of telecommunications.

Solutions to provide Parlay/OSA Framework functions for a secure and controlled access to Parlay X Web Services are currently under discussion, not only in the Parlay/OSA communality, but also in other contexts. OMA (17) is planning to define an architecture based on "Mobile Web Services" (18) and encompassing Parlay X features. Such functions should be based on Web Services mechanisms, in order to maintain a uniform technological context. If Parlay X Web Services are made available by a Telco Operator which already deployed a Parlay/OSA Gateway, one of the requirements to be considered is the reuse of the Framework component in order to provide a secure and controlled access, in particular:

- a Parlay X Web Service could perform the authentication of the software applications that want to access it by exploiting the authentication mechanism provided by the Framework functions in a Parlay Gateway;
- a Parlay X Web Service could verify whether a software application that wants to access it is authorized or not, by exploiting the authorization mechanism provided by the Framework functions in a Parlay Gateway;
- the Parlay X Web Service behaviour could be customized, by exploiting the configuration support provided by the Framework functions in a Parlay Gateway.

In this way, the Telco Operator has not to deploy additional servers external to the Parlay/OSA gateway in order to implement such "framework" functions for Web Services. A possible solution to reuse the Parlay/OSA Gateway Framework (see ■figure 3) requires

Figure 3 - An implementation of the Parlay X Architecture



splitting the implementation of a (Parlay X) Web Service in two parts, named respectively PX Proxy and PX SCS that implements the Web Service and the SCS part.

Summarizing, the Parlay X could be the first implementation of the new service architecture. It could act as a bridge between the next releases of the architecture and the current systems deployed or under deployment within the Operator networks. It offers the simplicity and the functions of the envisaged architecture, it is able to interwork with existing systems, and it is open and flexible for a smooth evolution.

"The user in the net": somz related issues

The representation of users' characteristics in the Network (who the user is, what his preferences, what his rights and subscriptions, what security levels and privacy should be granted) will be of paramount importance in terms of service offering. The user representation in the network is a concept that is taking shape under the push of several forces originated from the IT and the Telecom worlds. It brings to light two main issues:

- The representation of the user within the networked environment, and
- The collection, access, management of user related information that describes the user and his preferences.

These issues are often mixed together causing a little bit of confusion and the difficulty to achieve a viable architecture for services.

The Network Identity Concept

The definition of mechanisms such as DNS, DHCP, ARP and so on allows for a complete mapping of IP addresses to logical names. Actually those simple servers are one of the key elements of the IP networks. Also the Web has used the concept in order to associate addresses to logical names of pages. DNS is now moving from being a Name Server to be an Identity Server, i.e., a system that univocally identifies the user resolving the mapping of logical names that he may use, to addresses and to a unique identifier. Think for example to the need to access to different web pages by means of different passwords and login names. It would be quite nice for users to have mechanisms supporting the single sign-on. This requires the ability to relate all the user identities and to have mechanisms for keeping together all the user related info. Also in this area XML based solutions and software architecture as Web Services Architecture are leading. Not all the "service architecture" community is aware of the needs to integrate into the SOAs the Identity management. Under this respect, the paper supports the following statement:

"… Web Services must be inherently bound with (digital or network) identity to be able to behave the way they are designed to. Identity isn't an add-on, it is a central part of "their nature" (19).

So far the users have been identified by logical names, in a web services environment the logical endpoints can be applications, people, specialized resources dedicated to specific functions and accessible only by specific users. Practically an endpoint can be any thing that can be identified by an URI and accessed by a WSDL interface (20). An entity can have multiple addresses, multiple logical names, multiple associations to identity (think for example to a person that has several e-mail addresses and many login to access to a Web Site). There are stringent requirements on privacy and on how and when identity information can be shared among different providers. One point is quite important. An entity (a user) can even change his names, but the Network Identity should still refer to the same entity. Many Fora are trying to solve the federated management of Network Identity (namely the Liberty Alliance project (21), the OASIS organization (22), …). They follow different approaches but they have in common at least these characteristics:

- The usage of XML for describing the Identity features (Identity Documents).
- The usage of XML based communication means (namely the SOAP protocol).
- The introduction of Servers for supporting the Identity resolution (namely the identity server in the OASIS architecture, the Identity Providers in the Liberty Alliance architecture).
- The need for the federation of Identity Servers.

One key difference between the two Identity solutions is related to the need to have "absolute" Identifiers for univocally determine the user. In ■figure 4 the OASIS solution is represented. The point here is to introduce a new layer based on the URN (Uniform Resource Name) in order to decouple Logical Identities from the names used to refer to the entity. In other words an Identity Id (i.e., the E-Number) relates multiple network names into a single logical identity.
So far the "killer application" for the Network Identity has been the single sign-on. Major points under discussion focus on security, privacy, repudiation, personalised Service Level Agrement (SLA) support, and federation issues.

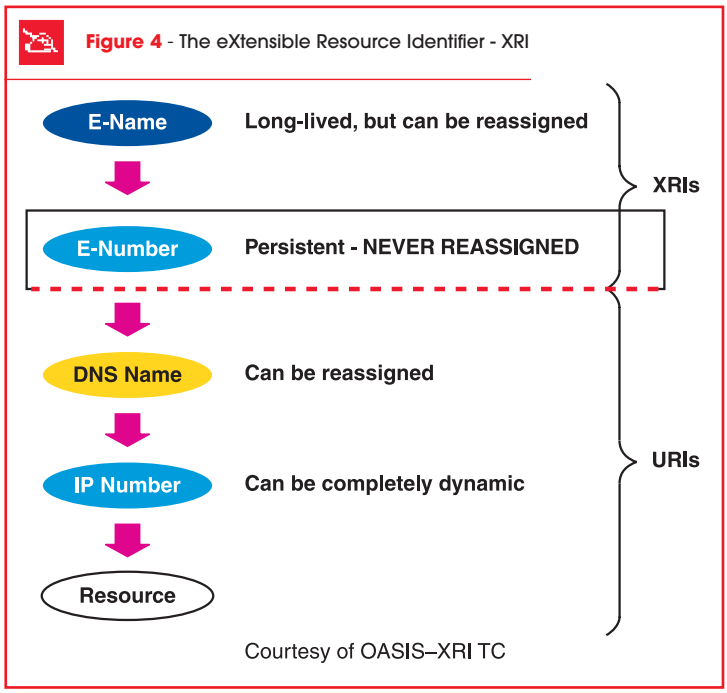


Figure 4 - The eXtensible Resource Identifier - XRI

The IT industry has already understood the importance of the Network Identity (e.g., Microsoft with the Passport system).
Also the Telecom world has used the concept of Network Identity. In fact the GSM Association has clearly separated the identity of the SIM card (namely the International Mobile Subscriber Identity IMSI Number) from the Phone Number (i.e., Mobile Subscriber ISDN MSISDN Number). This has facilitated a certain number of services such as, roaming and mobility, the dual number on the same SIM, or number portability. The Network Operator can be seen as a sort of Identity Provider while the SIM is acting as the Identity client. This also shades light on the quarell within 3GPP for the separation between USIM and ISIM.
The solution to the Identity Management is not complete; nevertheless it proved to be a powerful enabler for new and better services. Even if some work is to be done, the Network Identity is one of the key components of the Architecture.

The User Profile

For the GSM Architecture, the decoupling of user data from the central office (i.e., the usage of the Home Location Register) has enabled a large number of services (namely mobility management) that customers have greatly appreciated. This trend has to be pursued, i.e., a clear functional separation between the control functions and the used data has to be stated. This clearly identifies a fundamental component of the architecture: the User Profile.
On the IP side the solution for handling user data (e.g.,AAA) are well know and they are consolidating towards new protocols such as Diameter (23) and COPS (24).
The information about users (in IT and Telecom alike) has been scattered over different reposito-

ries. There is the need to figure out technical solutions for recomposing an integrated logical view of User Data (maintaining the ability to access data in real-time).
The HSS system has been defined in IMS for dealing with the collection of User information. In addition the 3GPP is working on the definition of a Generic User Profile (GUP) (25) for putting together the needed information that describes a User. However these efforts focus on the information collected and stored in a single environment and still keep a "centralized" approach. Data are logically organized by a single entity that passes references pointing to the relevant chunks of information to authorized and federated "requestors" (26). The ownership of data is quite sensitive from a business perspective and more federated solutions have more chances to be accepted. An interesting approach for describing user terminal characteristics and preferences is tried by the W3C with the CC/PP (27).
The static information related to the terminal features is stored in a specific location (e.g., the web site of the Vendor of the specific terminal) and the info related to dynamic features or preferences of the user is stored in the terminal itself (or in the network as a terminal agent). In this way, when the user is requesting to use a service by means of a specific terminal, the service logic could retrieve the static and dynamic information, recompose them and profiling the service accordingly to the terminal possibilities. The combination of HSS (the mobile world), AAA (the IP world), and the Profile definition (the IT - XML world) can bring to a powerful solution for logically representing the user data in the network keeping them distributed in different places.
The service logic could be helped by the platform's services making available both a unified access mechanisms (let's say a single component offering a unified API for getting User Data) or by means of several APIs that will query the different sources of information by means of specific protocols or languages (e.g., LDAP, SQL, …).

Other notably examples of User Profile are the SIP Registrar and the ENUM initiative. The latter tries to solve the problem of mapping E.164 numbers onto IP addresses, in doing this it also introduces the possibility to add ordered records of data about the preferences of the user. The Enum database started out to be a sort of DNS, but actually it can be defined a sort of User Profile.

Programmability within IP networks

At least three different paths can be identified in the area of Network programmability. In the Telecom industry, programmability has been pursued since the launch of the Intelligent Network. The idea was to decouple services functions from the switches introducing ad hoc control protocols (INAP, MAP, …) and systems. The IT industry has systematized the concept of programmable platforms in different branches: distributed computing platforms, Web Services Architecture, Application Servers (i.e., processing systems that offer a layered set of functionalities and tools in order to support the programmer to develop distributed services focussing on the business logic). The IP community has granted the programmability of nodes by means of protocols. The interacting systems use specific protocols in order to offer functionalities and services. The inner part of the system are not visible (and programmable in the IT sense) from outside.
The IP infrastructure is getting more complex and user requirements are pushing for dynamic QoS. There is the need for an IP control infrastructure that allows the extreme programmability of IP resources so that users can dynamically demand and be supported by appropriate network features (e.g., QoS on demand, routing services, security, reconfigurability, etc.). The definition of such an architecture should take into account existing IP resources, but also new types of "actuators" such as appliances, sensors and new kind of terminals.

The solution lies around the capability to decouple the logic of IP services from the actuators, so that a limited number of "intelligent nodes" can orchestrate the functionalities offered by a large number of executors.

In this section the paper analyses how different philosophies are approaching the problem of network programmability and then will indicate a few solutions to be considered in building the service platform.

The IETF Proposition

Within IETF, some initiatives tried to open up programmable interfaces for IP systems: primarily SIP, and, more recently, the Middlebox Communication (MIDCOM) working group (28). The results achieved so far are briefly.

The MidCom Approach

The MIDCOM working group started up with the goal of promoting the definition of an architecture able to integrate different IP control functions, for instance the Firewall traversal, the quality of service and other functional properties. The driving idea is to decouple the functions of network elements from the service logics. The model assigns to Policy Decision Points the goal of determining which policy to apply and to Policy Enforcement Points the execution of the policies. The value of MIDCOM is not in the proposed architectural model (it recalls a lot the IN and it is reusing the model already proposed by the COPS - Common Open Policy Service - protocol), but in the fact that it recognizes the need to decouple the services from the transport and routing mechanisms.

The dialogue between the involved entities is still based on protocols (and not on RPC and APIs). The debate on what protocols to use was a little bit amazing. The requirement of independence of the protocol from the services has been correctly stated, but the considered protocols were SNMP, COPS and SIP, the SOAP protocol was totally overlooked. Such a protocol is totally agnostic with respect to services, and it introduces the possibility to make the PDPs and the PEPs programmable by means of APIs. If SOAP was chosen then the MIDCOM architecture would have been easily converged towards a Web Services Architecture exploiting also many relationships with OSA/Parlay. With respect to the QoS, the SOAP solution could bring the advantage of a Framework for registering network functions as web services and could enrich the definition of Policy Decision Points with the already available specifications of Parlay for the Data Management. ■Figure 5 represents the possible architecture.

The SIP revolution ?

The Session Initiation Protocol (29) is seen as the future of communications: services will be provided and programmed in a much faster and simpler way than traditional telecom services, end-points will be offering a lot of intelligence to be used for new and exciting services. Those were the premises, but actually the simplicity of the protocol is proportional to the things that the protocol aims at doing. Now SIP is trying to be THE protocol for any service spanning from communications, to messaging, to instant presence, and so on. SIP specifications exceed 20 documents and are comparable in size and complexity to H.323 and possibly to the definition of some traditional telecom protocol. The bigger the specification is the more interpretations arise; in fact some interworking problems are emerging even in the SIPit testing (30.). The criticisms are not only related to the simplicity of the protocol, but to some misinterpretation of it. SIP is still a protocol and it is not to be confused with a software architecture: the comparison and the controversial with OSA/Parlay is totally out of scope, in fact OSA/Parlay can accommodate SIP as a component that offer SIP APIs. The SIP session model can be compared with other Call Models (in this case the comparison with Jain Call Control or H.323 or Parlay Call control is correct), and the simplicity is evident. But it is amazing to note that SIP started out promoting the concept of *session* (being quite different from the notion of call). Now the session and call are almost the same.

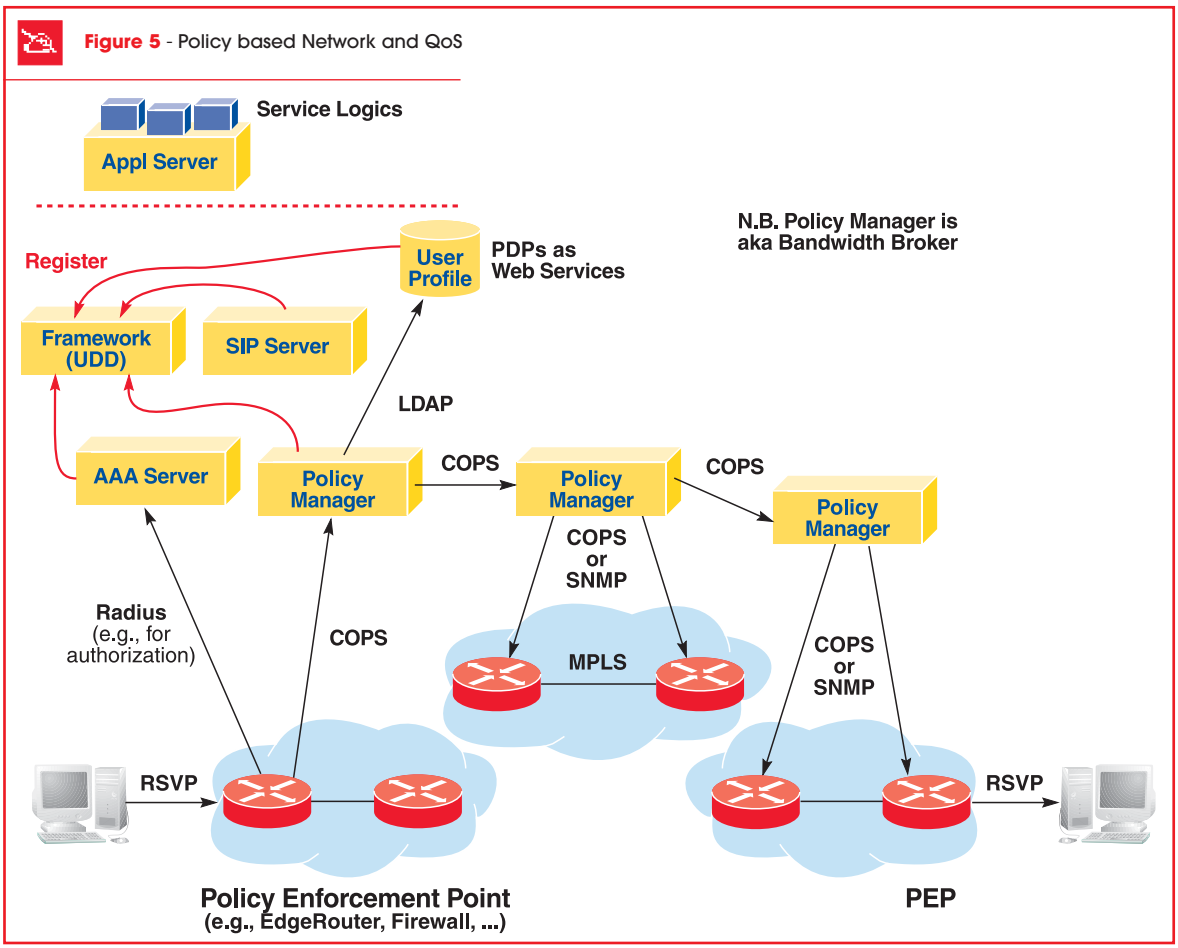


Figure 5 - Policy based Network and QoS

The SIP community has also tried to support programmability proposing the concept of SIP Application Server, i.e., an IT system able to program services on top of the SIP protocol. Such a system should be deployed within the network. Actually the term SIP Application Server seems to be out of scope because it is nothing more than an Application Server with a SIP protocol stack and it dramatically recalls the old centralized Service Control Point.

The concept of session is quite crucial: it is a means to associate in a computational activity different end-points with different processing capabilities. It is not constrained to deal with Call Models and signalling transport. Currently this feature is neglected, while it would be of paramount importance: think to the ability to identify resources and users by means of Network Identifiers and to associate them into a processing and communication session by means of an identifier that univocally relates participating nodes in the networked environment. The SessionId. could have a big role in coordinating the execution of services in a highly distributed processing environment. SOAP could easily transport Session Id in each remote procedure call, XML could allow for a complex and extensible representation of attributes of the SessionId (and the session itself), some Session Server could provide unique and network wide Session identifiers to users, nodes, and applications.

From a service platform perspective the usage of SIP will be a necessity, but it should be used in a wise way (i.e., as a good protocol for supporting some communications services).

The IT proposition

Another significant contribution of he IT world is the concept of Applications Server. They are processing systems that offer some levels of abstraction to the programmers in terms of

protocol transparency (i.e., the system is able to interact with external systems using heterogeneous protocols but internally it offer a single interface) and data access transparency (i.e., the system is able to access and to deal with heterogeneous sources of data, it can normalize the data so that the programmer has a unique view on them). In addition the Application Server is able to support high levels of robustness.

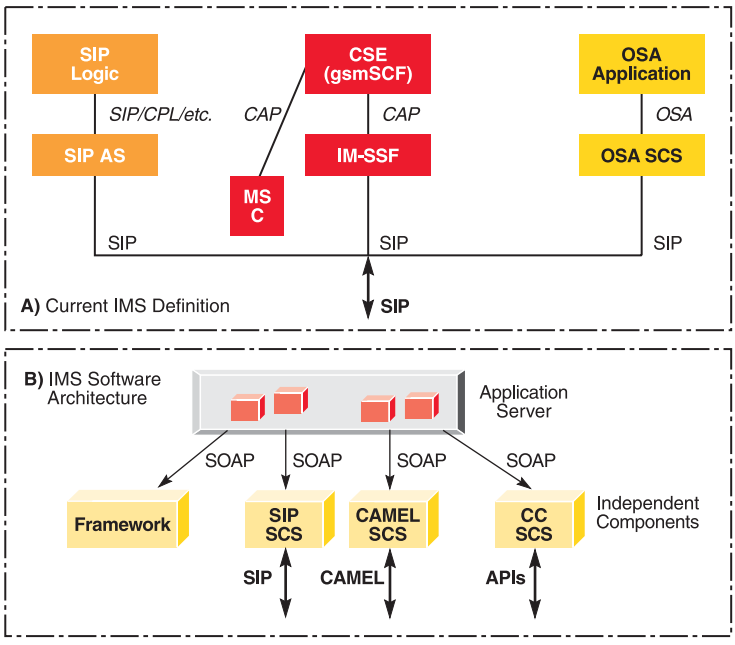
The IT world has also coped with the problem of resources discovery and allocation. Major (and confronting) architecture are: Sun's Jini, Salutation, and Microsoft's UPnP. The main concept is to create a networked environment that supports plug and play mechanisms. This is particular important when the resources are not using the same network technology. The concept around which these architectures have been built is the Service Brokering (31). It comprises functions like Service Registry, Service Discovery, Service Session and Service Availability: i.e., something similar to Web Services Architecture, but extended in order to encompass session and service assurance management. All the initiatives do end up to similar general conclusion, but different implementation (those implementation do leverage the Vendor asset: the Java language for Jini, the Windows OS for UPnP, ...). Also in the IETF the Service Location Protocol (32) has tried to cope with similar problems, but, obviously, just from a protocol point of view.

The Telecom Proposition

The Telecom industry has tried to define the evolution of programmability in different consortia. The most notably ones are the IPCC and MSF for broadband networks and the 3GPP for UMTS. The predominant idea in the industry is to integrate different ways of controlling heterogeneous network having SIP as a sort of "Lingua Franca". ■Figure 6 A) depicts the 3GPP approach for building the service platform. Some remarks are to be made:

- If all the IMS signalling is SIP, then the Service Logic resides naturally on the SIP Application Server (why bother with IWU and other systems?)

Figure 6 - IMS Service Architecture



- OSA is not used as a software architecture for integrating components, but just for opening interfaces up to third parties and for integration of functions in the old PLMN.

In our opinion the Service Platform for the 3GPP (and for other advanced networks) is quite different. ■Figure 6 B) gives a hint of the target architecture.
It is quite important that the service platform is built around an agreed software architecture (OSA and Web Services Architecture must be an important component of the IMS architecture). The Framework functions (or the Service Registry or the UDDI server) has a central role that should be clearly recognized and appreciated. All the network functions should be componentized and provided with APIs, then the components should be registered and their interfaces published. The components interact with the underlying networks by means of appropriated protocols or APIs.
The service platform for 4G is by nature multi protocol, because different services are to be provided on top of a heterogeneous networked infrastructure. Many resources will be programmable and the dialogue between the Application Server and the Resources should be as specific as possible. The platform should not be anymore centred on a single protocol (like the IN's SCP or the new version of it: the SIP application server). In simple terms, it is a mistake duplicating the software functions in two different systems: the OSA platform and the SIP application servers.

Network Transparency vs. Context awareness

One of the goals of the Parlay initiative is to support the concept of Network Transparency, i.e., the possibility to write a service logic independently from the heterogeneous networks that will be used for the transport of information and communication. This approach is open to discussion, because the network capabilities are the richness of the Network Provider, hiding the specific network functions is not convenient. On the other side, it is true that a level of abstraction is helpful, but it should be turned on-off depending on the type of service and the functionalities needed to support the specific service.
Mobile networks are strongly posing the problem of context-awareness. With multi-mode terminal the condition of provisioning of the service could vary very rapidly even during the same instance of the service (imagine a user entering and exiting from an area covered by a WLAN). The bandwidth, the security, the QoS vary during the lifetime of a session, however the user should be offered the best context of execution. (33). The context-awareness is not important only for mobile and heterogeneous networks, but even for a single network. In general, the network transparency can be detrimental in terms of functions available for the construction of services. A common principle is: the middleware can abstract and simplify the functions supported by the protocol, but it is not possible the vice versa, i.e., that the middleware adds functionalities to the protocol (we call this the "middleware dependence on protocol (34)). This means that if abstraction is introduced then the expressiveness of APIs is lower than the one of the protocol. There is a 1:1 mapping between the protocol and the APIs expressiveness when the APIs are a plain translation of protocol primitives. When state control is introduced in the protocol stack than some parameters and functions are hidden or interpreted or merged and their representation by APIs is simplified. If the goal of protocol stack is to support high level of abstraction it happens that many parameters and functions are concealed to the upper layer. The more control is exerted, the more abstraction is granted, but less expressiveness is offered to applications.
If more than one protocol is to be handled by the same component, then the simplifications, the need for mapping different states of the protocols and reconciling them will result in a even greater loss of expressiveness. The resulting APIs can offer functions to manage the proto-

cols, but many characteristics are lost. In addition the protocols to merge and integrate can have a different nature (consider for example INAP and SIP), so the abstractions are so many that the resulting APIs can be very generic (an abstract method as *make_a_call* plus some in and out parameters) and the work needed to develop the component can be huge. In this case the component would offer poor APIs with respect to the internal processing and respect to the expressiveness of the protocols.
We are more in favour of the identification of a set of basic components that map 1:1 with specific protocols needed to control and manage networked resources. The needed abstractions are introduced in the service platform by abstractor (or adaptors). The abstractors could be bypassed anytime the service need to access to the very basic features of the protocols.

XML as a Signalling Means

SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment
With SOAP, it is easy to define a dialogue between two remote entities. The protocol is "semantically agnostic" with respect to the service; in fact the application level dialogue is defined in terms of XML documents (DTD or Schema). The XML documents to use are identified in the header of the SOAP packet. The payload is then interpreted accordingly to the document at the end-point. The extension of functionalities and parameters is a simple operation having impact on the document definition. The expressiveness of XML is very high, so the language can represent very complicated data structure as well as methods and parameters of a remote call. XML documents can be also used to describe protocols' primitives. If the SOAP capability to allow a remote binding between remote end-points is added, it can be inferred that the combination of SOAP and XML documents represents and supports the exchange of primitives between nodes. If SOAP were transported on top of a resilient protocol (e.g. SS.7 or better SCTP, or new transactional protocols like Beep (35)) then SOAP could be a sort of new TCAP that carries a new kind of Application level Protocols. Considering that SOAP is an RPC, the dialogue between end-point ends up to be a sequence of invocations on remote APIs.
The SIP community has started to consider the usage of SOAP in combination with SIP, the idea is to use SIP as the transport (instead of http) and using SOAP in order to invoke remote functions on end-points. In this sense it is correct to talk about "SOAP over SIP" (36) (37). However there is another view for relating SIP and SOAP, i.e. to carry SIP primitives over SOAP payload. The SIP protocol primitives and parameters could be defined in terms of XML documents and transported between nodes by means of SOAP. Each networked node should have just a simple protocol stack for supporting SOAP and an interpreter for decoding the SOAP payload (containing SIP primitives as well other protocols' primitives): SOAP supports then a real multi protocol communication. This yields to a simple and slim distributed architecture based on a single protocol that simplifies the structure of distributed nodes. Wouldn't that be a real revolution? In addition why bother anymore about protocols; it is time for the definition of standardized schema or DTD describing the dialogue between networked end-points.
SOAP is also ready for supporting new computing models à la GRID.
Clearly SOAP is not thought to substitute the protocols, in fact there are some major issues to be considered in order to make the substitution happen. In particular the verbosity of the protocol has the drawback to make SOAP bandwidth consuming and this will still be a problem for mobile networks. SOAP should support transactional capability in order to guarantee features of current signalling protocols. The SOAP model is inherently client – server; in many communication services there is

the need to have also event notification. Such problems are under evaluation within the Standard Bodies dealing with the specification of Web Services Architecture. The usage of XML documents poses the need of interpreting all the remote calls introducing delay. In spite of all these problems, SOAP is a serious candidate to play a major role in the definition of future Service architectures (e.g., OGSA).

Service Brokering

The Service Brokering is another important issue: how the control infrastructure can support a variety of Actors that offer different services or networked capabilities, how to aggregate the service offering, how to allow a fair access, how to regulate the information flows between different administrative boundaries. It is clear that the Entity that will be able to play the role of service broker will have a chance to attract more customers. So far very limited (in scope) and proprietary solutions are offered, there is the need for an open platform that supports several business model.
In the UMTS business model, new services are likely to be delivered to the end users through a long and complex chain of business relations. Service capabilities and resources, service logic, service execution environment, service subscription management, service administration, service billing are probably aspects that different actors will take care of, with shared responsibility.
In general, a customer can access a service through some transport bearer or channel, e.g., voice connections, SMSs, IP connections, WAP connections, USSD. The channel can be used for multiple purposes: to activate the service, to control it and to get the results. The service could be operated by an actor, which plays the role of service provider, potentially distinct from the network operator that provides the channel.
Moreover, innovative technologies enable the development of services by assembling "service components" that could be provided by multiple actors. Parlay/OSA solutions and their evolution towards Web Services (i.e., Parlay X) enable to enrich the service provisioning business models by introducing new actors: the service component providers. They offer to service developers the usage of their resources through well-defined public interfaces, in a secure, controlled, and accountable way. The service providers can develop their services by assembling elements provided by service components providers, potentially distinct from the network operator (e.g., providers of payment mechanisms, information providers).
In this multi-actor scenario the mobile operator must face the problem of maintaining a central role, keeping the contact with customers. The main risk is otherwise to become a pure "connectivity provider", which would not be enough to justify the remarkable amount of infrastructure investments needed for the UMTS. The Service Broker role is a key factor to maximize the role that a mobile operator can play in a multi-actor scenario. The "rough" definition of a Service Broker may be the following:

1. It is the unique point of delivery to customers of a rich set of personalized services provided by many parties (e.g., Service Provider, Mobile Operator, etc.).
2. It is the unique point of access for 3rd parties to a rich set of network capabilities and service components needed to build services.

The core value of the above definition is that the Service Broker may simplify the relationships among the involved actors and provide advantages to them, including service providers.
A solution (a set of components) for supporting the role of service broker could enhance a platform for service component provisioning. For example a SOA platform (including Parlay/OSA) with a few extension such as the capability to hide the Network Identity of the user (and his logical names and addresses, and the capability to enforce policies related to SLA conditions, could be a starting point for offering service broker functions.

The service broker platform should provide functions for the brokering of customers' profile data to the service providers (e.g., preferences, payment account numbers), by taking into account privacy constraints or service subscription requirements. Moreover, it should provide functions for Identity management towards the service providers, in order to guarantee the identities of the customers. In addition the service broker platform has to handle the lifecycle of service, including registration, subscription, usage parameter negotiation, personalization, etc.

Concluding Remarks

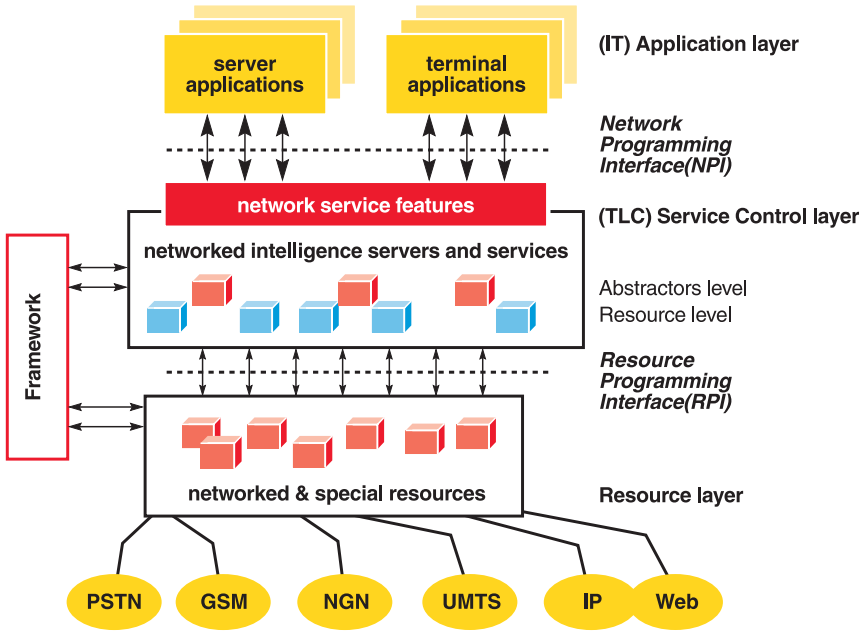
The current architectural definition of the service architecture layer follows two major approaches: to derive the Control Architecture directly from the single underlying network (all services based on a single – or very few – protocols), or to adopt the principle of network transparency, i.e., the service platform is unaware of the underlying transport complexity. In both cases the service layer is flawed because, in the first case the control is strongly biased towards a single class of services: those supported by the protocol(s); while, in the second case, the control functions tend to be too generic and abstract so that a service does not take advantage of the (heterogeneous) underlying transport capabilities (that are the major investments of the Operator). The paper has provided evidence for the end of the Network Intelligence as understood today. A set of guidelines for building a new Service Architecture according to the requirements of the Networked Intelligence have been discussed. The architecture trades off between the expressiveness power of (many) control protocols and the easiness of programming an abstract network (overcoming the issue of the "network transparency"). Service Oriented Architectures, (i.e., the Web Services Architecture) integrated with some OSA/Parlay functions (e.g., Service Availability) are the way to easily integrate (i.e., register) new network functions and to support service composition capabilities. The Web Services offer the advantage to be based on XML technologies, so that they are the natural candidate for the framework of the future.

New resources and related controller can be progressively added; components interfaces can be standardized and/or be personalized in order to tailor to specific customer requirements. In addition the Network Identity can help in offering personalized Service Level Agreements (SLA) for using networked resources (services, but also resources).

The abstraction is provided by means of abstractors that can mask the low level APIs used to control network resources. Depending on the need, the platform can support any level of context awareness (it depends on the capability of network resources to register on the framework). The User is at the centre of the networked environment because each function can identify and adapt to the specific needs of that customer. The Service Platform is ready to support several business models, so the Service Broker role is not an exclusive role of the Network Operator, but different actors can potentially play it. ■Figure 7 depicts the Network Intelligence Architecture.

The major findings are: there is the need to propose a set of APIed components that can form horizontal and layered solutions. The service platform can start small in terms of integrated components (in order to reduce the initial investments) and can incrementally grow introducing new control functions and services. The Service Oriented Architectures, (i.e., the Web Services Architecture) taking advantage of some Parlay functions (namely the Framework functionality for supporting the Service Availability) are a way to easily integrate (i.e., register) new network functions and to support service creation capabilities. Resources should expose low-level API so that no low-level features are lost due to abstraction. The abstraction should be introduced in terms of components that "simplify, but do not hide" the low level APIs. The SIP protocol is going to be one of the components of the

Figure 7 - the Networked Intelligence Architecture



architecture, but the problem related to the association of networked resources to sessions (the Session Id) still remains open, even if SOAP solutions could be proposed.

Acknowledgements

The authors wish to thank Dr Drummond Reed for his support in documenting the work of Oasis. Our greatest estimation goes to all those colleagues that have supported the architectural view with their daily work building software solutions. THANKS!

Glossary

3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorization, Accounting
ADSL	Asymmetrical Digital Subscriber Line
ARPU	Average Revenue Per User
BCSM	Basic Call State Model
BGCF	Breakout Gateway Control Function
CAMEL	Customized Applications for Mobile Network Enhanced Logic
CAP	CAMEL Application Part
CC/PP	Composite Capability/Preference Profiles (W3C)
CCITT	Comitè Consultif International Tèlègraphique et Tèlèphonique
CCXML	Call Control Extensible Markup Language
COPS	Common Open Policy Service
CORBA	Common Request Object Broker Architecture
CPL	Call Processing Environment
CS.x	Capability Set number x (Intelligent Network)
DCOM	Distributed Component Object Model (Microsoft)
DNS	Domain Name Server
E2E	End to End
EDGE	Enhanced data rate for GSM (Global) Evolution
ETSI	European Telecommunications Standards Institute (Cedex - F)
GCP	Gateway Control Protocol (es. MGCP - Media Gateway Control Protocol, SGCP Simple Gateway Control Protocol)
GERAN	GSM/EDGE Radio Access Network
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	originariamente Groupe Spéciale Mobile, in seguito Global System for Mobile telecommunications
GUP	Generic User Profile
HLR	Home Location Register
HSS	Home Subscriber Server
HTTP	HyperText Transfer Protocol (WWW)

HTTPR	Reliable Hyper Text Transfer Protocol
I-CSCF	Interrogating – Call State Control Function
ICT	Information and Communication Technology
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem (UMTS)
IN	Intelligent Network
INAP	Intelligent Network Application Part
IP	Internetwork Protocol (layer 3)
ISV	Independent Solution (o Software) Vendor
ITU	International Telecommunications Union
IVR	Interactive Voice Response
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JAIN	Java for Advanced Intelligent Network
JDBC	Java Data Base Connectivity
JTAPI	Java Telephony Application Program(ming) Interface
MAP	Mobile Application Part
MEGACO	Media Gateway Control (IETF working group)
MGCF	Media Gateway Control Function
MGCP	Media Gateway Control Protocol
MGW	Media GateWay
MIDCOM	MIDdlebox COMmunication
MMS	Multimedia Messaging Service
MSF	MultiService Switching Forum
NGN	Next Generation Network
NTT	Nippon Telegraph and Telephone Corporation
OMG	Object Management Group
OSA	Open Service Access
P-CSCF	Proxy – Call State Control Function
PC	Personal Computer
PLMN	Public Land Mobile Network
PSTN	Public Switched Telephone Network
R-SGSN	Roaming Signaling Gateway
RI	Rete Intelligente
RPC	Remote Procedure Call
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
QoS	Quality of Service
S-CSCF	Serving – Call State Control Function
SCE	Service Creation Environment
SCF	Service Capability Feature
SCF	Service Control Function
SCP	Service Control Point
SCS	Service Capability Server
SDK	Software Development Kit
SGSN	Service GPRS Support Node
SIB	Service Independent Building Block
SIM	Subscriber Identity Module
SIP	Session Invitation Protocol
SLA	Service Level Agreement
SLEE	Service Logic Execution Environment
SMS	Short Message Service
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPA	Service Provider API
SS.7	Signaling System number 7
SSP	Service Switching Point
SW	Software
T-SGW	Transport Signaling Gateway
TINA	Telecommunication Information Networking Architecture. TINA-C = TINA Consortium
TLC	Telecomunicazioni
UDDI	Universal Description, Discovery and Integration
UDP	User Datagram Protocol (layer 4)
UE	User Equipment
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
UTRAN	UMTS Terrestrial Radio Access Network
VLR	Visitor Location Register
VOIP	Voice over IP
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WSDL	Web Services Description Language (IBM/Microsoft - XML format for describing network services)
XML	eXtensible Markup Language
XTML	eXtensible Telephony Markup Language (XML-based service description language)

References

(1) D. Isenberg "Rise of the Stupid Network" in http://www.rageboy.com/stupidnet.html
(2) http://www.salutation.org/scenarios.htm
(3) http://www.wireless-world-research.org/
(4) 3GPP TS 23.228 "IP Multimedia (IM) Subsystem" - Stage 2, in www.3gpp.org
(5) http://www.softswitch.org
(6) http://www.msforum.org/
(7) http://www.omg.org/technology/documents/formal/corba_2.htm
(8) http://www.globus.org/ogsa/
(9) "Simple Object Access Protocol (SOAP)" version 1.1 W3C Note in http://www.w3.org/TR/SOAP
(10) Web Services Description Language (WSDL) 1.1W3C Note http://www.w3.org/TR/wsdl
(11) http://www.uddi.org/specification.html
(12) http://wwws.sun.com/software/sunone/identity/
(13) I. Foster, C. Kesselman, S. Tuecke "The Anatomy of the Grid" in http://www.globus.org/research/papers/anatomy.pdf
(14) http://www.parlay.org

(15) Parlay X in http://www.parlay.org/specs/index.asp
(16) G. Di Caprio, C. Moiso, Web Services and Parlay: an architectural comparison, in Proc. ICIN 2003, pp. 195-200
(17) http://www.openmobilealliance.org/
(18) Info about OMA Mobile Web Services in http://member.openmobilealliance.org/ftp/public_documents/mws/charter/
(19) The Digital ID World Newsletter - February 12, 2004 Issue
(20) XNS White Paper "From Name Service to Identity Service: How XNS Builds on the DNS Model" in http://www.xns.org/pages/tech_wp.html
(21) http://www.projectliberty.org/specs/liberty-idff-arch-overview-v1.2.pdf
(22) Oasis XDI Technical Committee White Paper, D Reed, G. Strongin "The Dataweb: an introduction to XDI" in http://www.oasis-open.org/committees/download.php/5115/wd-xdi-intro-white-paper-2004-01-20.pdf
(23) http://www.ietf.org/rfc/rfc3588.txt
(24) http://www.ietf.org/html.charters/rap-charter.html
(25) 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3GPP Generic User Profile - architecture;Stage 2 (Release 6) in http://www.3gpp.org/ftp/Specs/html-info/23240.htm
(26) A. Sahuguet, R. Hull, D. Lieuwen and M Xiong "Enter Once, Share Everywhere: User Profile Management in Converged Networks" in http://citeseer.nj.nec.com/554172.html
(27) http://www.w3.org/Mobile/CCPP/
(28) http://www.ietf.org/html.charters/midcom-charter.html
(29) http://www.ietf.org/html.charters/sip-charter.html
(30) http://www.nwfusion.com/news/2004/0202sip.html
(31) Salutation Architecture Specification (Part-1) Version 2.0c in Http://ww.salutation.org
(32) http://www.ietf.org/html.charters/OLD/svrloc-charter.html
(33) L. Capra, W. Emmerich, and C. Mascolo "Middleware for Mobile Computing: Awareness vs. Transparency" in http://www.cs.ucl.ac.uk/staff/c.mascolo/www/hotos1.pdf
(34) D.G. Messerschmitt, "Convergence of telecommunications with computing", invited paper in special issue on "Impact of Information Technology", Technology in Society, Vol. 18, No. 3, Elsevier Science Ltd..
(35) http://www.beepcore.org/beepcore/home.jsp
(36) Ubiquity White Paper " SIP and SOAP" in http://www.ubiquity.net/pdf/SIP_and_SOAP_1-3.pdf
(37) "SOAP a Lather for SIP?" in http://www.e-principles.com/Article_14.htm

Roberto Minerva, Manager, is responsible of the Competence Area "Wireless Network Architecture" of Telecom Italy Lab within the Function Network Innovation. He graduated cum Laude at the Bari University University. Since 1987 he has worked on the development of distributed applications in CTI (Computer Telephone Integration) environments and on the definition of service architectures for broadband networks. He has contributed to the definition of the Service Architectures of TINA, initially within the TINA-C CoreTeam, later on as Project Leader of various international initiatives. From 1996, he has been responsible of TILAB projects with the objective of an incremental implementation of an advanced platform for services for Convergent Networks (fixed and IP and mobile) using technologies like VoIP and SIP within an OSA/PARLAY architecture framework. Currently his search activities include: the realization of platforms and related middleware components (such as User Profile, Netwok Identity) based on the OSA architecture, the study of programmability solutions for the data architecture of 3GPP (IMS), the study of programmable solutions for controlling the QoS in "all-IP" networks.

Contacts

Roberto Minerva
Telecom Italia Lab
Tel. +39 011 228 7027 – Fax +39 011 228 5069
roberto.minerva@telecomitalia.it

Corrado Moiso
Telecom Italia Lab
Tel. +39 011 228 6780 – Fax +39 011 228 5069
corrado.moiso@telecomitalia.it