Service Provision Evolution in Self-Managed Future Internet Environments

Apostolos Kousaridas. Panagis Madgalinos, Nancy Alonistioti
Department of Informatics and Telecommunications,
University of Athens, Athens, Greece
{akousar, panagis, nancy}@di.uoa.gr

ABSTRACT

Future Internet is based on the concepts of autonomicity and cognition, where each network element is able to monitor its surrounding environment, evaluate the situation and decide the action that should be applied. In such context, the traditional service provisioning approaches necessitate a paradigm shift so as to incorporate the Cognitive Cycle. Towards this end, in this chapter, we introduce a Cognitive Service Provision framework suitable for Future Internet Networks. The proposed approach supports cognition by modeling a Service as an aggregation of Software Components bundled together through a graph. Consequently, each Service is composed by various components and is tailored to the operational context of the requestor. In order to prove the viability and applicability of the proposed approach we also introduce the enhancement of the IP Multimedia Subsystem through our Cognitive Service Provision framework. Finally, based on our work, we discuss future research directions and the link between service and network management.

FUTURE INTERNET NETWORKS

Future network systems design principles are based on high autonomy of network elements in order to allow distributed management, fast decisions, and continuous local optimization. The Cognitive Cycle model, as it is depicted in Figure 1, is envisaged to be in the heart of Future Internet Elements and it leads to their autonomy [1], [2]. A Future Internet Element could be a network element (e.g., base station, mobile device), a network manager, or any software element that lies at the service layer.
The three distinct phases of the Generic Cognitive Cycle Model are the following:

- Monitoring process involves gathering of information about the environment and the internal state of a Future Internet Element. Moreover, the Monitoring process receives, internally or externally, feedback about the effectiveness of an execution that took place, after the last decision.
- Decision Making process includes the problem solving techniques for reconfiguration and adaptation, utilizing the developed knowledge model and situation awareness. The Decision Making supports the optimal configuration of each element, considering its hypostasis and the organization level that it belongs. Decision making mechanism identifies alternatives for adaptation or optimization and chooses the best one, based on situation assessment, understanding of the surrounding context, and the preferences of the element. After decision making, the execution process undertakes to apply the decision that will change the behaviour of the element.
- Execution process involves (self-) reconfiguration, software-component replacement or re-organization and optimisation actions.

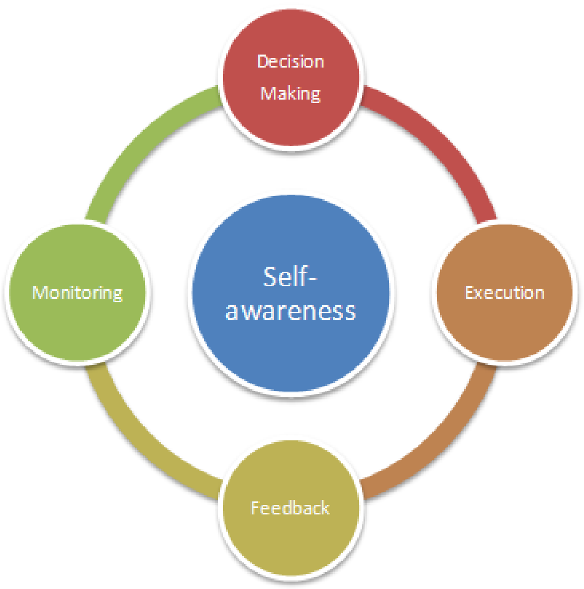


Figure 1: Generic cognitive cycle model

The scope of this book chapter is to discuss the challenges and describe the path for the evolution of the Future Internet services synthesis, delivery and adaptation, by exploiting the cognitive cycle paradigm. The cognitive cycle is placed at each network element that provides, consumes, or forwards one or more end-user services, and thus affects their performance and consequently users' experience. Even the software that undertakes to deliver (i.e. service provider) or consume (i.e. user application) the respective end-user service (Service Layer) is designed following the cognitive cycle model and consequently interacts with the other cognitive cycles, thus affecting its behavior.

The continuous increase of the number of user equipments, in combination with the evolution of the traditional client/server model [3], for service provision towards more distributed application structures have paved the way for more advanced services. Especially, by taking into account that even simple users through their own devices (e.g., smart phones) can concurrently have the role of service consumer and service provider.

The rest of this chapter is organized as follows: In the following section the service management background is investigated, by presenting key research outcomes for service delivery, service publish and service discovery. Thereinafter, a platform independent and system agnostic framework for the evolution of the Future Internet service management is described, by adopting the cognitive cycle paradigm. In section 4, the baseline 3GPP IP multimedia sub-System (IMS) architecture is studied and its extensions in order to support the cognitive service provision framework are proposed. Finally, we conclude with future research directions and especially we discuss the cooperation between the service and network management, which is a major challenge that lies ahead.

SERVICE MANAGEMENT BACKGROUND

In the context of this section we will elaborate on the presentation of some key concepts of the service management area. We commence by providing a number of definitions which set the methodological and theoretical foundations of our work and proceed by identifying the state-of-the art paradigms of the area.

Our work is based on the assumption that a *Service* instantiates a specific functionality offered to a user by the execution of one or more applications ([4], [6]). Therefore, we employ the notions of *Software Component* and *Application*. Specifically, a *Software Component (SW Component)* is a standalone software module that when combined appropriately with others form an *Application*. The latter comprises the software bundle that upon execution provides a *Service* to the users.

In order to capture and represent in a coherent manner the information related to a service we employ a number of profiles. Following a bottom-up approach we first define the *Software Component Profile (SW Component Profile)* which contains information regarding the requirements of a specific SW Component in terms of execution environment (both hardware and software platform) and collaboration with other SW Components. The latter identifies a specific part of the *Software Component Profile*, namely the *Binding Rules*. The *Application Profile* aggregates information deriving from various *SW Components Profiles* and provides information regarding the requirements posed for the successful provision of the identified *Application*. However the *Application Profile* contains additional, *Application* specific, information such as a provision/download URL. Finally, similarly to the *Application Profile*, a *Service Profile* comprises the aggregation of one or more *Application Profiles* as well as additional information related to charging and billing. A high level, UML based depiction, of this hierarchy is provided in Figure 2.

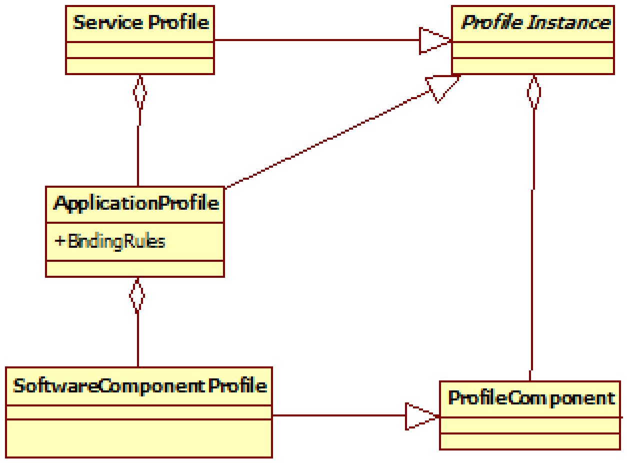


Figure 2: High level depiction of the cognitive service provision framework concepts

The fundamental entities that exist in every service provisioning framework are depicted in **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**. Initially, a *Service Provider*, which essentially comprises an actor, internal or external to the actual system, provides *Services* to terminals (end users) of one or more networks. The provision requires the initial registration of the service. The action of service registration essentially corresponds to the notion of *Service Publishing* which captures all functionality related to service registration and advertisement so a *Service* can become accessible and consumable by any *Service Requester*. The latter, discovers a *Service* by utilizing functionality identified by the concept of *Service Discovery*. Finally the service is delivered to the requestor through predefined protocols (*Service Delivery*) and can be adapted in order to match the requester's requirements (*Service Adaptation*). In the following paragraphs we attempt a short overview of the requirements posed by the *Service Publish*, *Service Discovery* and *Service Delivery* functionalities as well as state-of-the-art research in these areas, always from the perspective of a dynamically composed number of *Applications*, which upon deployment/composition provide a specific *Service* to a user.

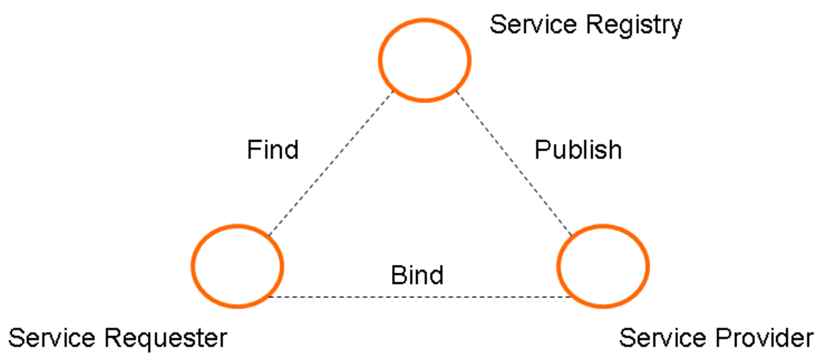


Figure 3: Traditional Service Provision Approach

Service Publish

The *Service Publish* aggregates all functionality related to the registration and possible advertisement of a service by a service provider. This module implements the following:

- *SW Component Profile* registration
- *SW Component* upload
- *SW Component Binding Rules* registration(Application Registration)
- *Application Binding Rules* registration(Service Registration)
- *Service Profile* registration

The most fundamental part of this procedure is the description of the procedure through which *SW Components* are bonded into *Applications* and *Applications* into *Services*. The implementation of this composition should provide two kinds of guarantee, namely, low computational cost and scalability. The former denotes the requirement for low delivery time with respect to the time required for the delivery of an off-the-shelf service, while the second the ability of a service to be composed of a large number of applications without imposing significant load compared to a lower cardinality composition approach.

Attempting to address these issues in the context of SELF-SERV [7], a declarative composition language has been defined. The idea is based on statecharts [8] and supports the successful deployment of composite services (services composed of other services) as well as their concurrent or distributed execution. Each composite service is perceived as a statechart, where states represent services and transition arrows between them simulate input and output. A more lightweight approach is outlined in [9], where the use of RDF is considered for the description of networks of applications that involve discrete processes.

Service Discovery

The *Service Discovery* aggregates all functionality related to the discovery of a specific *Service* or a set of services by a registered client. The implementation of this action is directly influenced by the structure of the communication scheme. Therefore, two cases are distinguished, specifically centralized (traditional client-server communication) and ad hoc (peer to peer scheme).

There are several protocols and platforms which have been proposed and developed up-to-date with respect to the requirements posed by the discovery process. *Service Discovery* necessitates protocols which allow automatic detection of devices and services offered by these devices through communication networks. In addition, it dictates negotiation in terms of user preferences and terminal or network capabilities (e.g. profile Specification). Sun's JINI [9], "Universal Description, Discovery and Integration" protocol (UDDI) [10] , IETF Service Location Protocols (SLP) [11], UPnP [12], DNS-SD

[13] and Bluetooth's Service Discovery Protocol (SDP) [15] are some of the most important service discovery protocols that were designed for wired and wireless networks.

In the ad hoc case, the discovery phase becomes more elaborated, since virtually every peer of the network has to be queried. A straightforward approach would be the adaptation of well known content and query distribution techniques in peer to peer networks, such as those proposed in [16], [17] or [18]. On the other hand a service discovery protocol specifically designed for mobile ad hoc networks such as GSD [20], Konark [21] or HESED [22] could directly be adopted. However, the ad hoc network outlined previously is envisaged as being of low node cardinality with small lifetime.

Service Delivery

The *Service Delivery* aggregates all functionality related to the delivery of a specific service or a set of services to a registered client. The term delivery embraces all procedures involved from the selection of a service (right after the discovery phase) until its consumption by the client. The implementation of this action, as in the case of the *Service Discovery* is directly influenced by the structure of the communication scheme.

Web Services is one of the most important technologies, which incorporate the functions of the afore-discussed modules (Service publish, service discovery and service delivery).Web Services are a collection of protocols and standards used for exchanging data between applications or systems. Web services model uses Simple Object Access Protocol (SOAP) [32], Web Services Definition Language (WSDL) [33] and UDDI protocol ([35]), for service provision, service description and service discovery respectively. WSDL describes how to use the software service interfaces. A WSDL description is retrieved from the UDDI directory and the services are invoked over the World Wide Web using the SOAP, which is XML based, used for exchanging structured data and type information in a decentralized and distributed environment.

Web services allow programming language independence, and assure the desired platform interoperability and openness. Furthermore, the combination of semantic web and Web Services (OWL-S) provides greater automation for service provision, selection, invocation, composition and negotiation. The IP Multimedia Subsystem (IMS) is the service and session control platform, which allows the core network to be independent of a particular access technology (e.g. WLAN, UMTS), to support end-to-end QoS guaranteed connections, and to integrate mobility for all network applications.

COGNITIVE SERVICE PROVISION

In the context of this paragraph we will present in details the Cognitive Service Provision framework. As stated in the previous paragraph, the work is based on the following assumptions:

- An *Application* is considered as the composition of *Software Components*
- A *Service* is the user experience provided by the execution of one or more *Applications* or equally the functionality offered to the user by the execution.

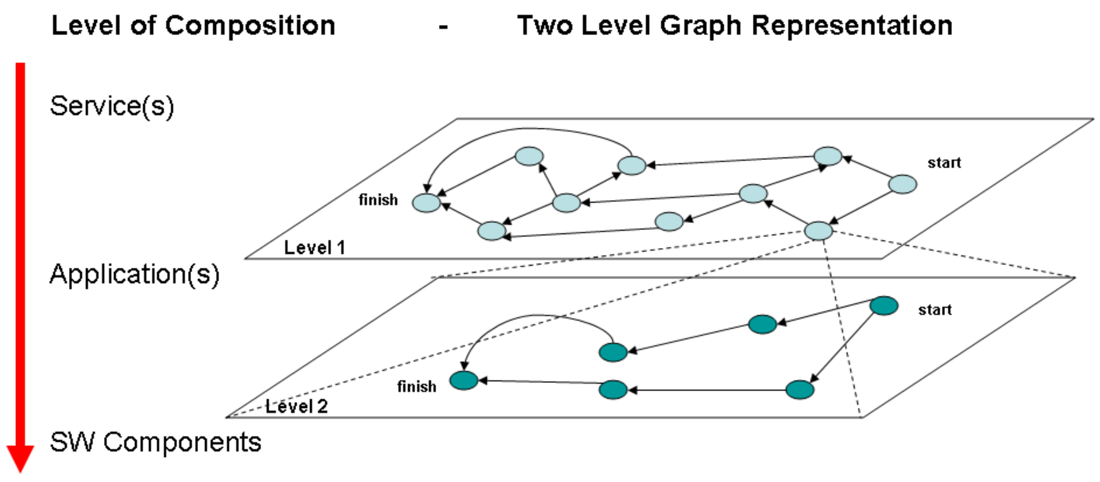


Figure 4: Service representation as a graph

Each service is modeled as a two level hierarchy graph (Figure 4). Each registered *Service Provider* enlists its *SW Components*, their binding-into-applications schemes as well as the applications' combination rules. In the upper layer the nodes represent the applications and links between them input and output information. The various paths defined between the Start and Finish nodes of the graph denote the various instantiations of this specific service. In the second layer, each application node is depicted as a graph, where this time nodes represent *SW Components*, links transfer of control from component to component and paths adapted versions of the same application.

The proposed hierarchical scheme for services' description gives us the capability to specify the architectural framework for Future Internet services provision in the context of a complex and distributed network environment, where the cognitive cycle paradigm is present. In the remaining of the paragraph we will provide a platform independent and system agnostic architecture (UML Modeling, Functional Blocks and Signaling) for the evolution of service delivery, discovery, adaptation and composition mechanism by capitalizing on the cognitive cycle model. The presentation follows a bottom-up approach; initially we present the fundamental operations outlined above and then we provide the holistic view of the framework and the employment of the cognitive cycle.

Service Publish

In the context of the CSP *Service Publish* operation each registered *Service Provider* enlists its *SW Components*, their binding-into-applications schemes as well as the applications' combination rules that will result in the provided services. Due to the fact that our framework is targeting a large set of devices, ranging from personal computers to mobile phones, the representation should be lightweight and easily processed. Consequently the information is encoded with a set of adjacency matrixes. Figure 5 illustrates the registration of a service that is the result of the execution of an application made up of one component.

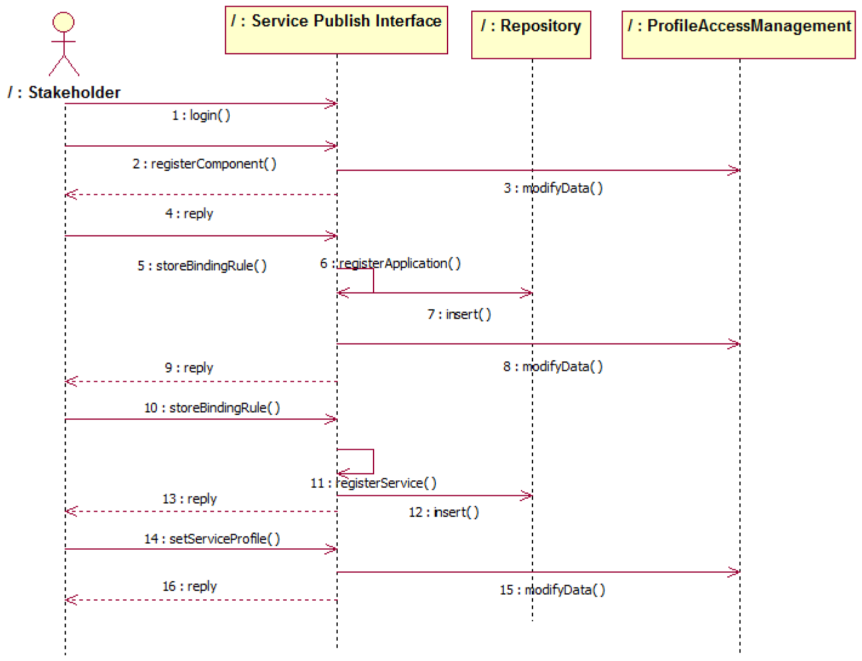


Figure 5: Service Publish Module

The *Stakeholder* logs into the systems and registers a new *SWComponent*. Essentially the registration triggers a series of events described by the action *modifyData*(). As soon as the operation is successfully concluded a reply is issued back at the stakeholder, who in turn starts inserting the binding-rules. The latter are dynamically placed in the graph and give the possibility to register a new *Service* through the *setServiceProfile*() action.

Rule-based approaches are combined in the process of service publishing. More specifically, the publishing system, based on the available binding information and the new service profile defines the service using only the available *SW Components* and *Applications* just like in [26]. Moreover the arcs are populated with weights that are calculated using the quality criteria outlined in [27] together with policy rules defined by the *Service Provider*. In addition the Web Ontology Language (OWL) which provides interoperability and inference features and in sequence the OWL Services (OWL-S [22]) ontologies is used to express detailed semantic information facilitating the subsequent service discovery mechanisms, by describing the meaning of the service through ontological annotation.

All the previous and forthcoming theoretical analysis is more effective by using a simple and efficient way of implementing the concepts of *SW Components*. It is obvious that the implementation of tens or hundreds of software modules is simply unaffordable and inapplicable in large scale communication environments. However, technologies such as AspectJ, JMangler and AspectS ([28], [30], [31]) allow the design of one simple component and its subsequent dynamic differentiation. In this way, one executable can represent a whole class of software modules.

The *Service Publishing* approach introduced by the proposed *Cognitive Service Provision Framework* is depicted in Figure 6. The heart of the system is the *ServicePublish* class, which implements the *ServicePublishInterface*. The latter is exploited by the Stakeholder in order to insert new services. The *Service* comprises the aggregation of the numerous *SWModules* and is described by a *ServiceProfile*.

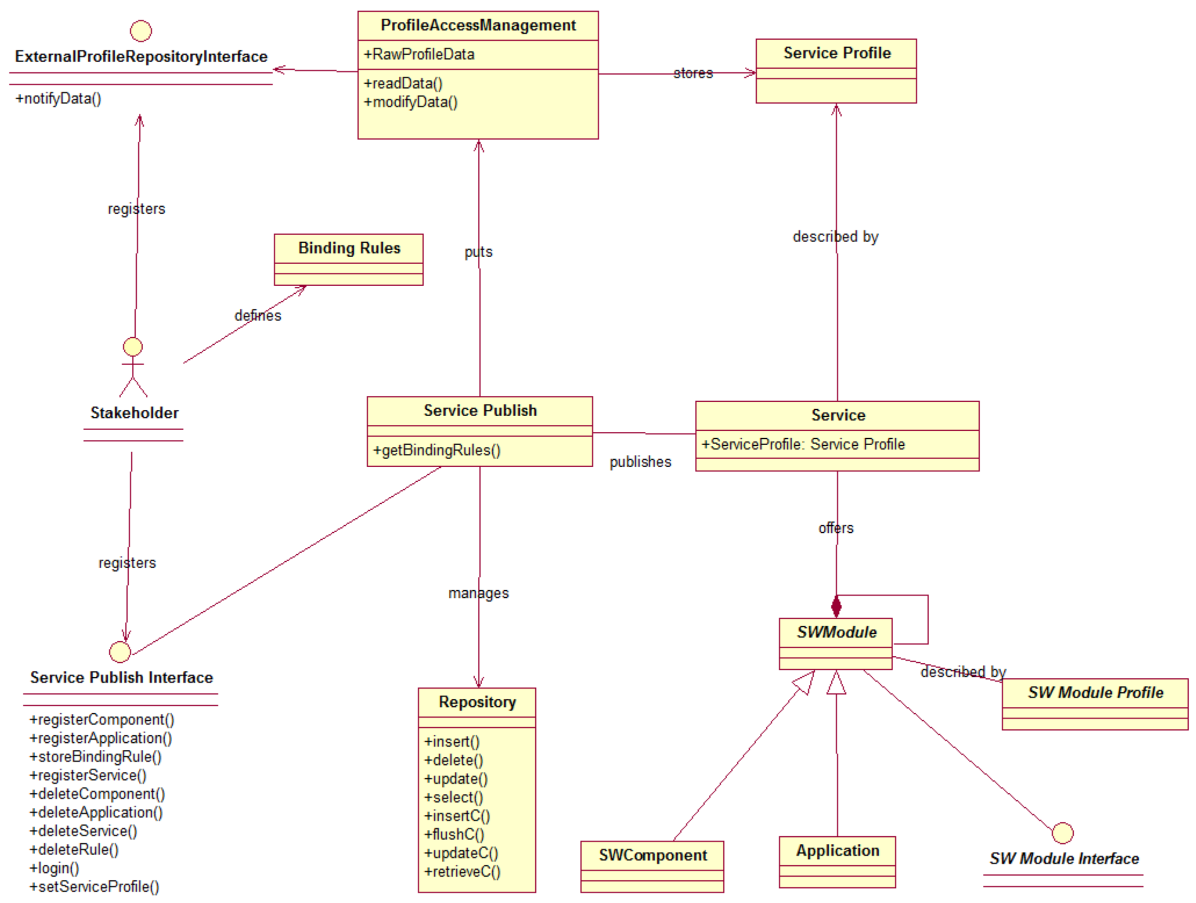


Figure 6: Service Publishing in the context of the proposed Cognitive Service Provision framework

Service Discovery

As explained in the previous section *Service Discovery* aggregates all functionality related to the discovery of a specific service or a set of services by a registered client and its implementation is influenced by the structure of the communication scheme; therefore, two cases are distinguished, centralized (traditional client server communication) and ad hoc (peer to peer scheme). The proposed protocol for the traditional client server model is presented in Figure 7. The ServiceDiscoveryModule of the client retrieves information from its local information base and requests a service from the ServiceDiscoveryModule of the server. The latter, retrieves all available information and proceeds in a twofold filtering. At first, the rules are filtered and the available Applications are extracted. These Applications will serve as basis for the formulation of the final Service. Then, based on the issued request, a new filtering is applied which identifies the various Service compositions that can be offered to the requestor. The most important part of this procedure is the profile filtering. All nodes of the graph hierarchy outlined in the *Service Publish* are examined with respect to their hardware and software requirements. As a result, a set of nodes is removed together with their incoming and outgoing arcs. The aforementioned procedure is carried out from bottom to top. Specifically:

- ***SW Component Level:***
    - Each component is evaluated concerning its hardware and software requirements. All those that do not conform to the limitations posed by the execution system are removed.
    - The remaining components form the various application graphs.

- ***Application Level:***
    - Each application graph is examined for the existence of at least one path from start to finish. If no such path exists then the application is ignored.

- ***Service Level:***
    - The service graph is examined with the same procedure as above. If no path exists from start to finish then the service cannot be provided, otherwise, all the existing paths depict various versions of the same service and are transmitted back to the querying node.

In the end, this procedure incorporates one more assessment step, namely the filtering of policy information. Therefore, services are defined in accordance with the rules provided by the current network operator, service provider and service consumer.

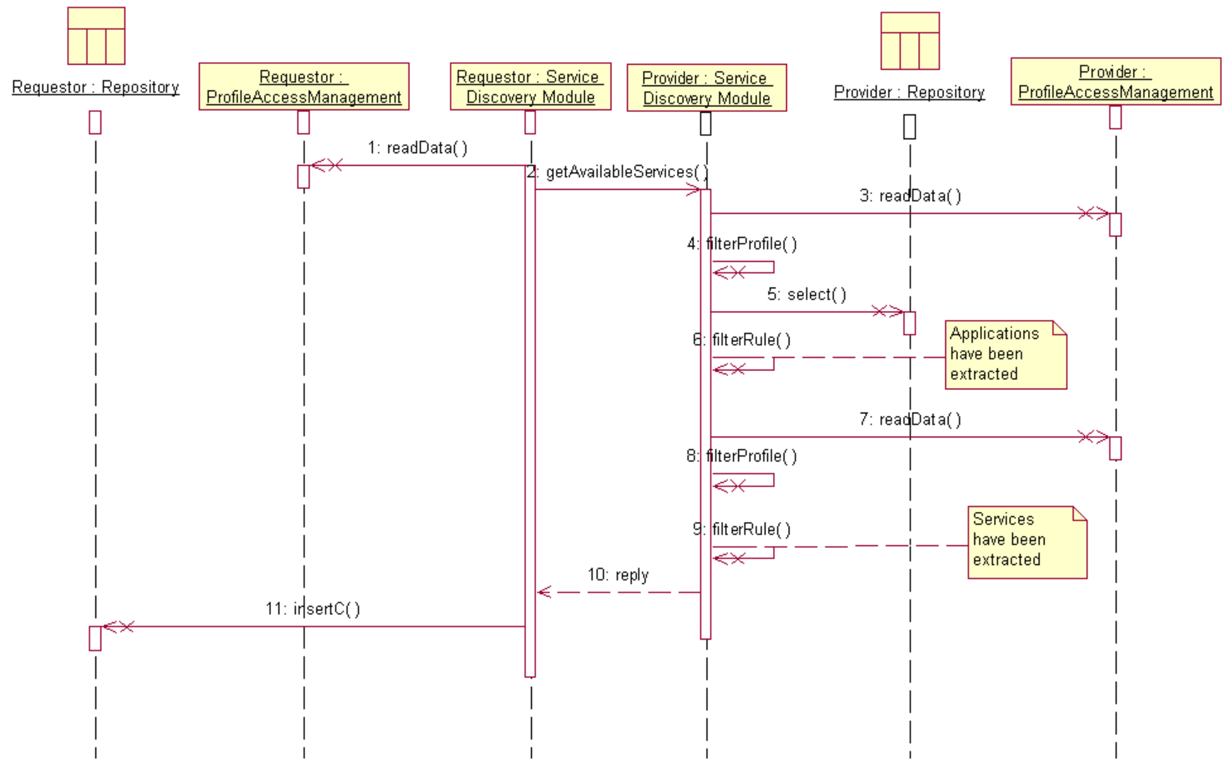

Requestor : Repository
Requestor : ProfileAccessManagement
Requestor : Service Discovery Module
Provider : Service Discovery Module
Provider : Repository
Provider : ProfileAccessManagement
1: readData( )
2: getAvailableServices( )
3: readData( )
4: filterProfile( )
5: select( )
6: filterRule( )
Applications have been extracted
7: readData( )
8: filterProfile( )
9: filterRule( )
Services have been extracted
10: reply
11: insertC( )

Figure 7: Service Discovery in a client-server query scheme

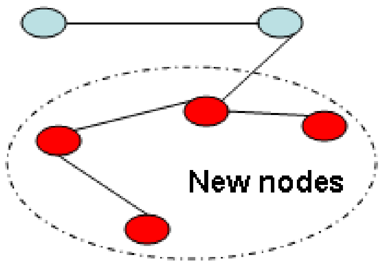


Figure 8: An ad hoc network with new peers joining (red)

However, an ad hoc network outlined previously is envisaged as being of low node cardinality with small lifetime. Therefore, in the context of the proposed *Cognitive Service Provision* framework, a simple store and forward technique together with caching is proposed.

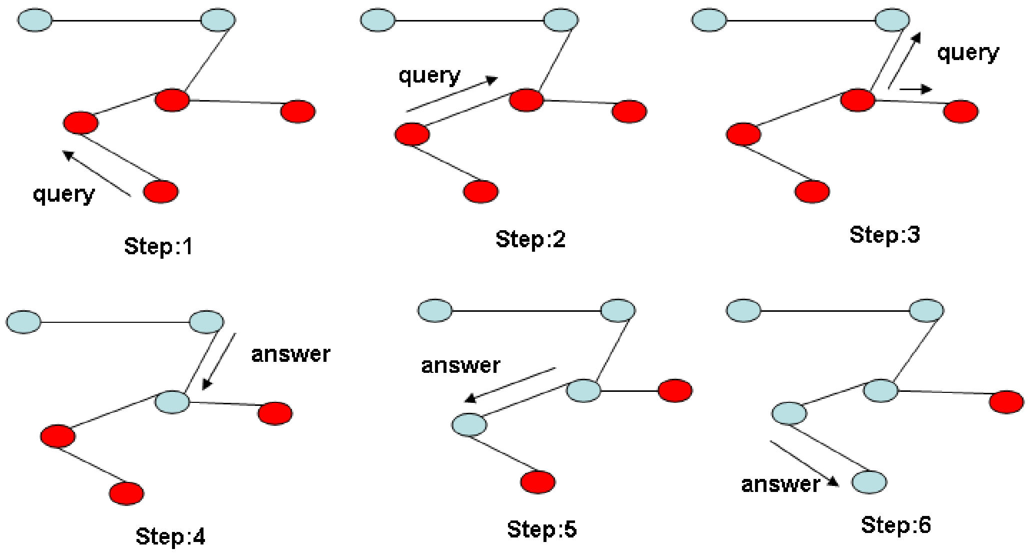


Figure 9: Illustration of the query process

Each peer upon necessitating a service initiates a query procedure in order to receive all services available in the vicinity. The query is forwarded by neighbor peers until it reaches a node that has already knowledge of available services. These services are stored in a cache memory repository. This event triggers a cache exchange between the querying device and the queried one, since the former could also be a service provider. A simple example is illustrated in Figure 8 and Figure 9. Initially two nodes are interconnected and form a small network which is expanded when four new devices (red circles) are connected to it. For simplicity reasons we assume than none of the new nodes can offer a service, therefore they have an empty service cache.

Thereinafter, one of the new peers requires a service. Since it cannot provide it to itself, initiates a query procedure. The query is forwarded by adjacent nodes until it reaches one with already populated cache. The generated reply is forwarded to all nodes of the query path until it reaches the source.

In order to ensure that a query is not endlessly forwarded in the network or get stuck in a loop, two simple techniques are used. Every query message has a predefined TTL, which is decreased by one every time it reaches a new node. Moreover, a query carries a specific identification number together with the MAC address of the initiating peer. These fields are stored in each peer's cache memory and checked whenever a message requires forwarding. If a match between the newly arrived fields and the already stored appears then the query is discarded.

Figure 10 depicts a simple example of this procedure. Profile filtering is carried out as in the client-server example presented previously, but this time, each peer decides locally based on its context information which services are supported for execution.

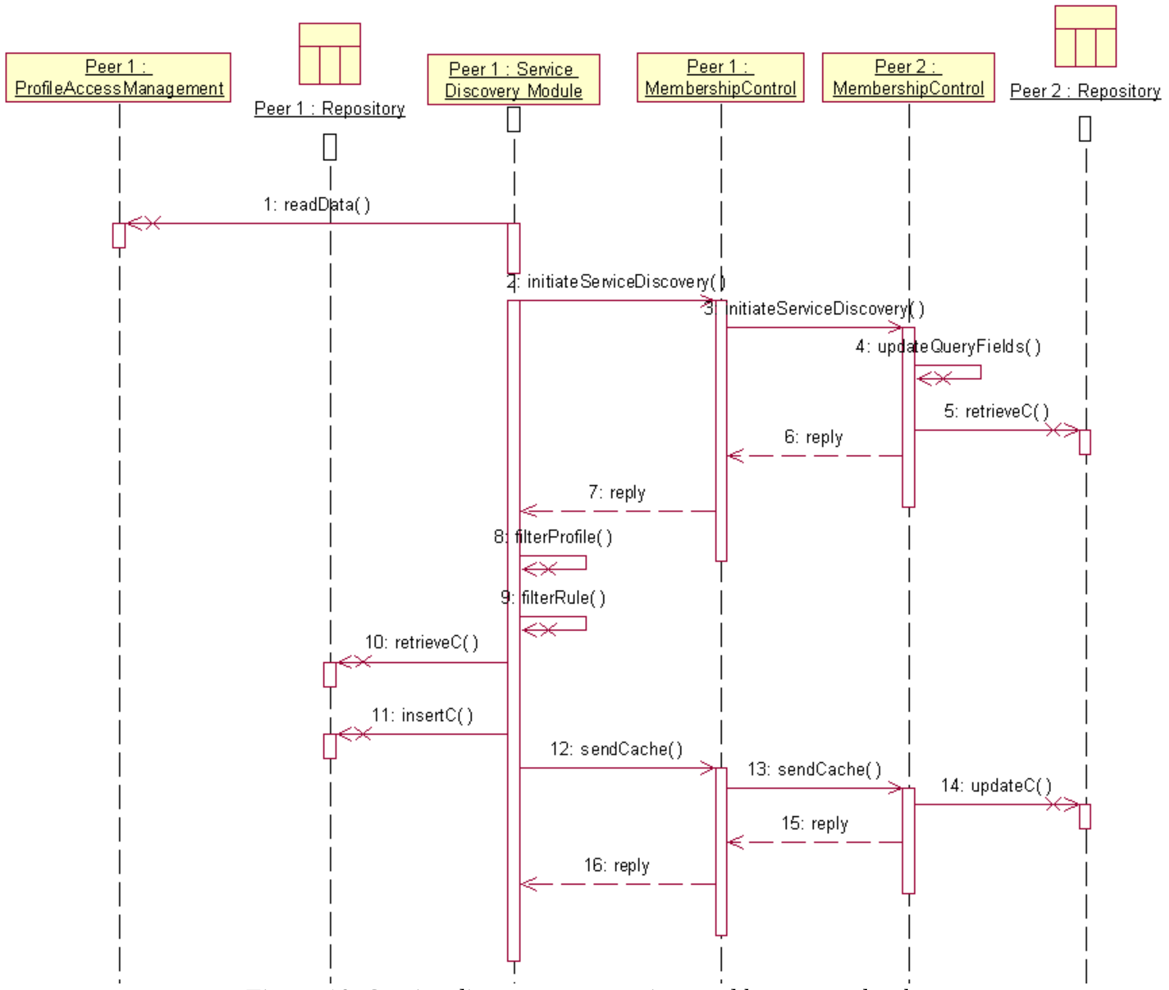


Figure 10: Service discovery process in an ad hoc network scheme

The *Service Discovery* approach introduced by the proposed *Cognitive Service Provision Framework* is depicted in Figure 11. The important functionality of the system is the *ServiceDiscovery*, which inherits and extends the *DecisionMaking* class. The latter provides support for decision making actions which are required in the context of service discovery. Essentially, the *ServiceDiscovery* class retrieves information through the *Repository* and implements the actions of filtering as depicted in the *ServiceDiscoveryInterface*.

Service Delivery

Following the two envisaged cases of discovery, namely traditional client-server and ad-hoc/p2p schemes the *Service Delivery* is also studied in these contexts.In both cases three are the basic steps of this procedure: service composition, downloading and application execution.

The basic part of this module is the implementation of the service composition scheme. Based on the analysis of the prior paragraphs, the following types of composition have been identified:

- **Server Side Composition:** The server takes charge of the binding procedure based on the available binding and context information. This approach is used when the terminal (user equipment) is not able to sustain the load imposed by this operation.

- **Client Side Composition:** The server forwards the binding rules and the components to the client machine. In this case, the client undertakes the binding procedure based on the forwarded binding information. This approach is followed when the terminal (user equipment) is able to sustain the load imposed by this operation.

A second classification could be used with respect to the time the service is composed ([23]):

- **Proactive Composition:** Proactive composition refers to the offline composition of services on the provider side. This type of composition is used when service composition is time consuming or the specific service is often requested.
- **Reactive Composition:** Reactive composition refers to the on-the-fly creation of a service upon request. It is employed in order to provide a personalized version of the service to the client.

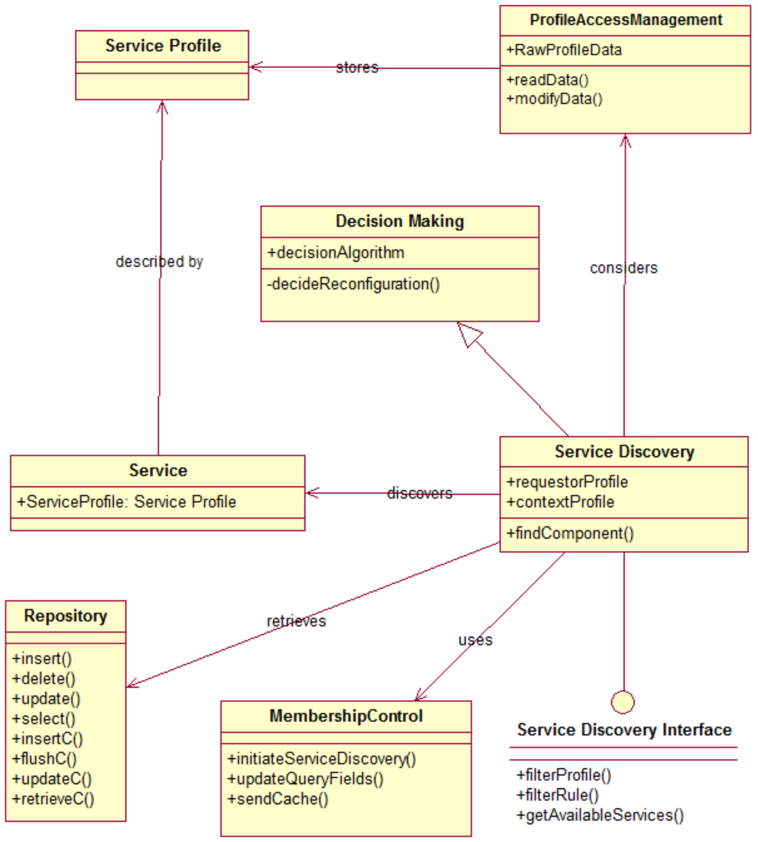


Figure 11: Service Discovery in the context of the proposed Cognitive Service Provision framework

The outlined classification schemes are not disjoint; on the contrary they can be combined as it is depicted in Table 1. The main difference between the service delivery process between an ad hoc network and a fixed topology lies in the fact that the proactiveness feature of a composition cannot be used as well as the delivery of an already composed set of applications. Consequently, peers on their own gather and combine the various *SW Components* into *Applications* and then into *Services*.

	Proactive	Reactive
Server Side Composition	X	X
Client Side Composition	X	X

Table 1: Service Composition Schemes

As far as execution is concerned, the following schemes are derived:

- **Centralized Execution:** All applications offering the requested service are executed locally.
- **Distributed Execution:** Applications offering the requested service are executed distributed and their results are aggregated in the client node. The idea is that the applications already exist somewhere and the client essentially instead of downloading them locally acts as the aggregator of the final output. It has to be stressed out however that the applicability of this execution scheme necessitates a set of robust machines, a stable network and synchronization according to the composition graph. Various implementation of this idea appear in [20], [24], [25].

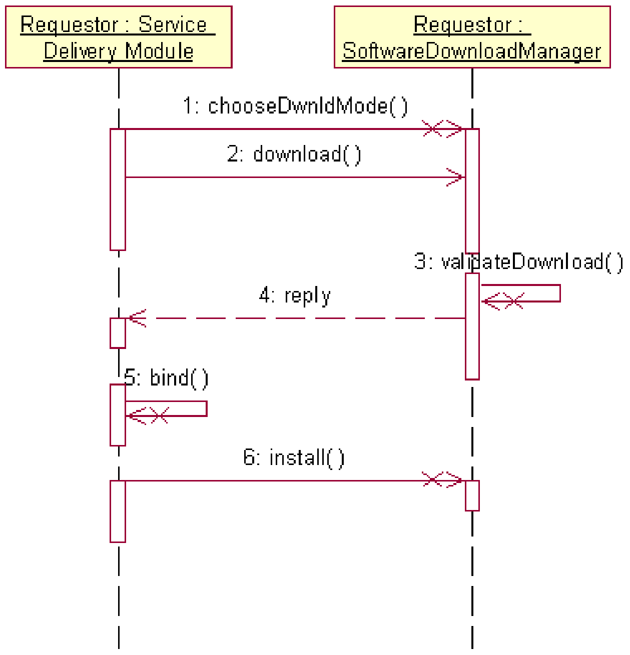


Figure 12: Service Delivery (client side, reactive composition - central execution)

The client side, reactive composition and execution sequence chart is depicted in Figure 12, while Figure 13 provides an overview of our proposal. The main module in this case is the *SoftwareDownloadManager*, which undertakes all tasks related to the downloading of the software as well as the *BindingRules*. Additionally, the module is able to install and uninstall the software (i.e. delete an application) or in case of malfunctioning roll back to the previous stable situation or re-initialize the downloading session. The *ServiceDownloadManager* is triggered by the *ServiceDeliveryModule*.

Service Adaptation

Service Adaptation includes the procedures that uppon their enforcement achieve to differentiate the result that a user experiences, through the execution of one or more applications. The feature of adaptability is an important characteristic of pervasive services. An important phase of service adaptation procedure is the transition from the current state to the most suitable one of the entity being adapted, considering:

- The specified policies of the involved entities and the contextual environment
- Minimum effort from the provider and the client with the minimum consumption of resources.

The capability of *Services* to autonomously adapt to the context from which they are requested and in which they execute includes mechanisms, like:

- Dynamic adaptation of *Applications*
- Parameter level adaptation techniques
- Dynamic aggregation of components

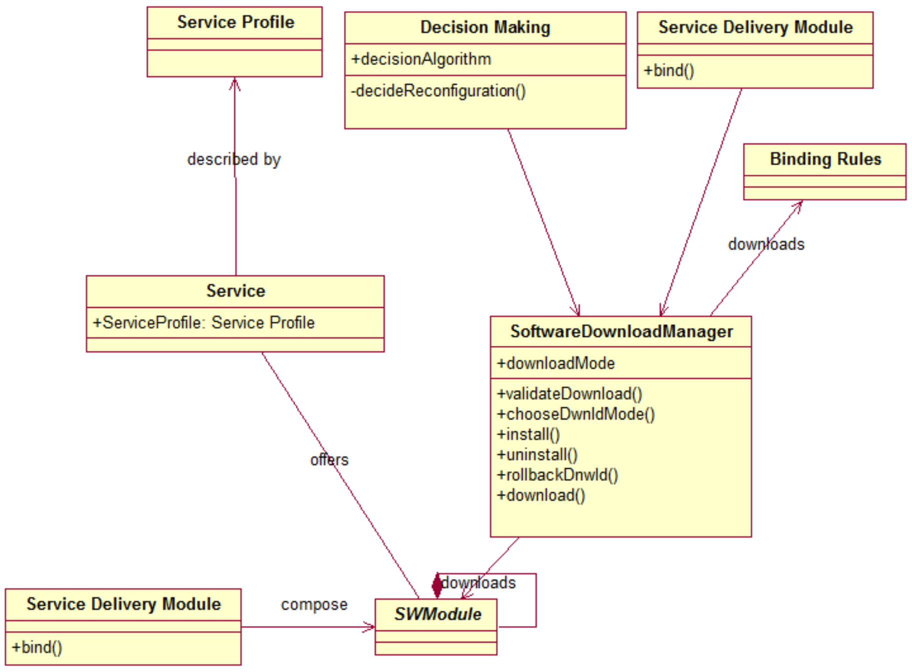


Figure 13: Service Delivery in the context of the proposed Cognitive Service Provision framework

It is straightforward therefore to consider that this differentiation can be triggered by various actions, such as:

- Change of the environment in which the *Application* is executed
- Change of the content consumed by the executed application
- Change of the *Applications* interactions that compose the *Service*
- Change of the *SW Components* that define one or more *Applications*

Two algorithms that can directly be applied to this problem are defined in [37] and [19]. Upon detecting a node failure in the graph, the aforementioned algorithms backtrack to the closest node with outgoing arcs and try to find another path to the final destination. The general idea is largely based on the well known principal of optimality [38] that is also used in the context of our work.

An example of this procedure is illustrated in Figure 14, Figure 15 and Figure 16. Upon delivery request, the paths on the applications and service graphs have been created and the service is being initialized. If we consider that one of the SW Components is not working properly and therefore the application in which it operates is also malfunctioning a remedy action should be triggered. The solution that will ensure service continuation lays in the selection of an alternate path either in the service graph or the in specific application graph where the malfunctioning component exists.

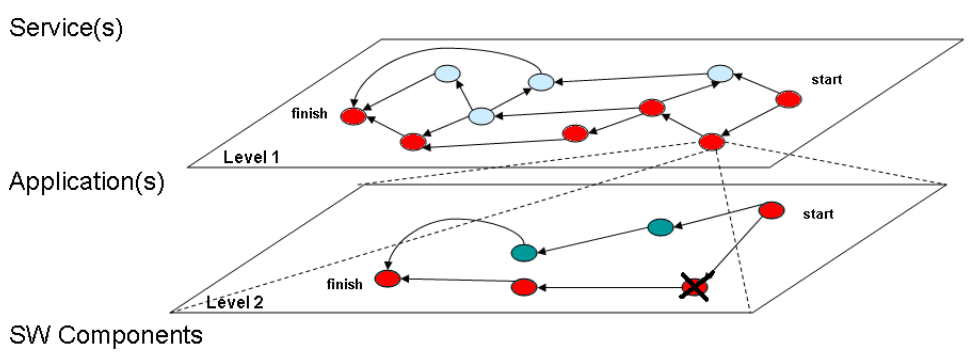


Figure 14: Service Adaptation triggering - Component Failure (red signifies components/ applications in use)

For example, the *Service Adaptation Module* may choose to completely ignore the problematic application and select another combination of applications for the service (alternate path in the service composition graph), as it is depicted in Figure 15.

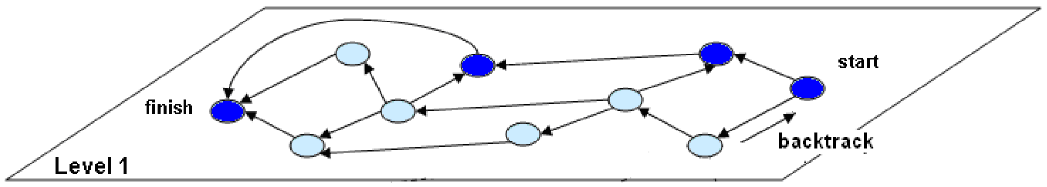


Figure 15: Adaptation on the service composition layer (blue signifies adapted service)

Alternatively, the adaptation may take place at level of the application graph. In this case the overlaying graph remains unaltered, and the malfunctioning application is adapted my changing the components combination sequence (alternate path in the application composition graph), as it is depicted in Figure 16.

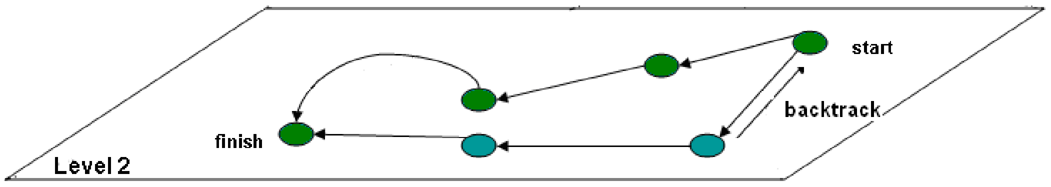
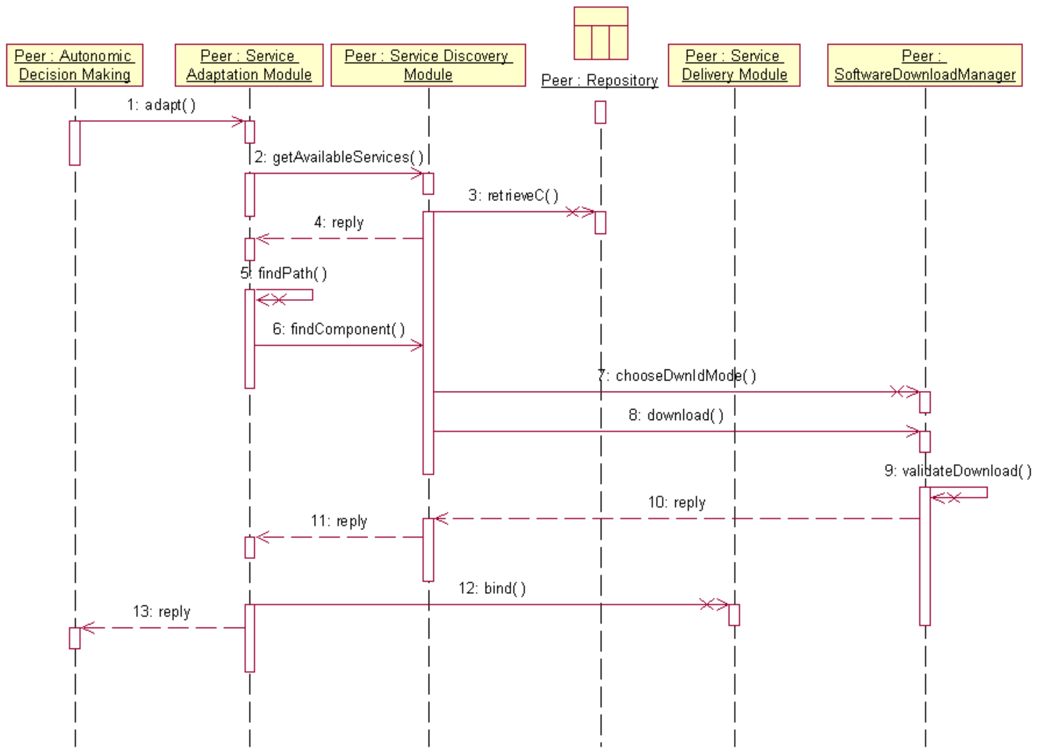


Figure 16: Adaptation on the application layer (green signifies adapted application - Level 1graph remains unaltered)

One of the issues that has not been addressed by the aforementioned analysis is the decision process and specifically the reason for which a service version is selected from the set of available ones. In the discovery phase we can assume that the user chooses the service, while in the adaptation phase the device itself has to make the decision and select the new version.

A simple technique, which is employed in our framework, is to find the path with the highest node intersection comparing to the old one. High intersection provides smaller differentiation between the two versions of the service, namely the initial and the adapted one. The enhancement of weights in the graph will enable the use of path selection techniques such as Dijkstra’s or Bellman’s algorithms and consequently the definition of sophisticated service adaptation techniques.

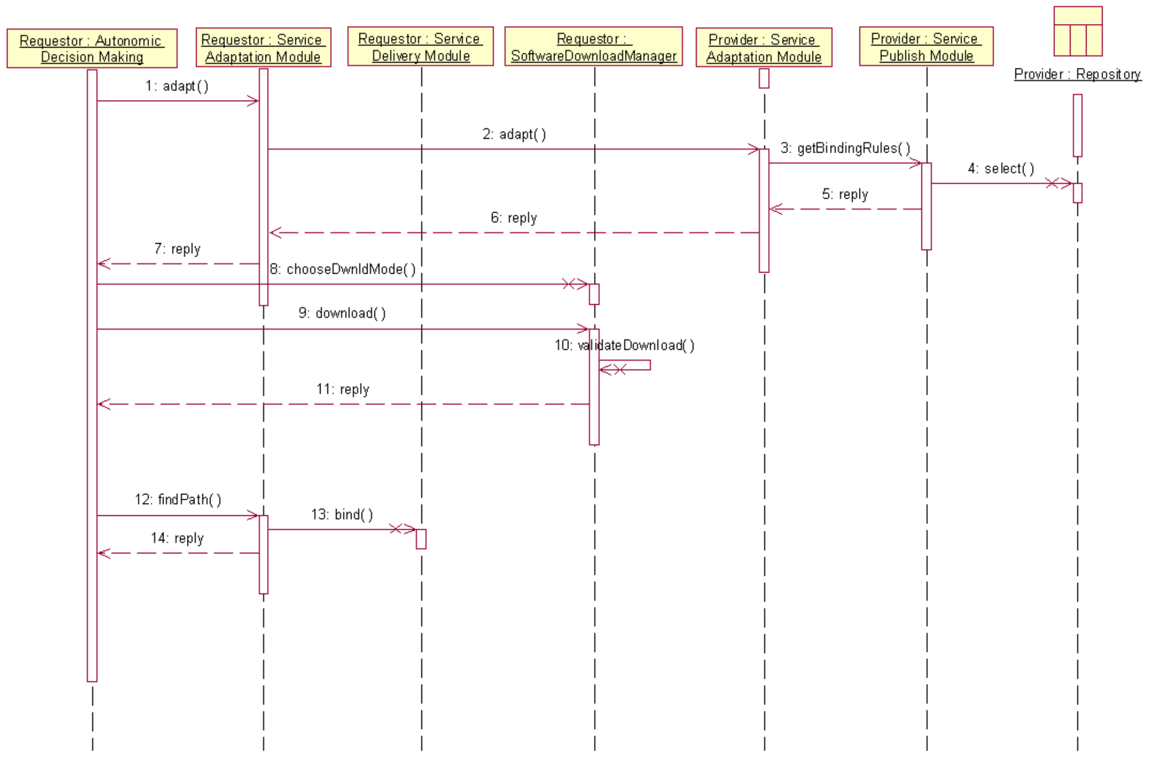

Peer : Autonomic Decision Making
Peer : Service Adaptation Module
Peer : Service Discovery Module
Peer : Repository
Peer : Service Delivery Module
Peer : SoftwareDownloadManager
1: adapt( )
2: getAvailableServices( )
3: retrieveC( )
4: reply
5: findPath( )
6: findComponent( )
7: chooseDwnldMode( )
8: download( )
9: validateDownload( )
10: reply
11: reply
12: bind( )
13: reply

.

Figure 17: Service Adaptation in an ad hoc network scheme

Another issue that is discussed below is the location where the adaptation process takes place. It is straightforward to assume that the selection algorithm will be executed at the network node, where the required information is aggregated. Consequently, in the ad hoc scenario, service adaptation takes place in the service consumer side, while in the client-server communication scheme it can take place either on the provider or on the consumer node. In both cases, if a SW Component is missing it can be downloaded from the possessing node. The flow charts of this procedure are depicted in Figure 17 and Figure 18. Figure 17 presents the messages that are exchanged in the case of an ad-hoc network. When a service adaptation action is triggered by the decision making module of a peer node the first step is to retrieve the list of the available services through the *Service Discovery* module. The *Service Adaptation* module of the initiating node selects the new path of the service and applications graph. If the download of an application component is necessary then the service discovery in cooperation with SoftwareDownload agents of peer nodes retrieves the missing components. Finally the new service is deployed in the *Service Delivery* module. In the case of the client-server scheme the steps are similar. The main difference is the process to retrieve the binding rules of the service and application layer graphs. The *Adaptation Module* of the client side controls the whole process for finding the new path.

Figure 19 provides the high level view of the *Service Adaptation* approach of the proposed framework. The key introduced feature is the inclusion of the Cognitive Cycle in the context of *Service Provision*. As the reader may notice, the Monitor-Decision Making-Execution approach is inherited by *Service Adaptation* and specialized for the case of *Service Provision*. The *Monitoring* process is implemented as *Service Monitor*, as specific entity which constantly monitors the validity of the binding of the application, while the *Decision Making* specializes the *Decision* procedure of the MDE cycle for the identification of *Service* related problematic situations. Finally, the *Service Adaptation* corresponds to the *Execution* part of the MDE cycle and translates the directives of the *Decision Making* entity to specific commands on the graph of the *SW Components*.



Requestor : Autonomic Decision Making
Requestor : Service Adaptation Module
Requestor : Service Delivery Module
Requestor : SoftwareDownloadManager
Provider : Service Adaptation Module
Provider : Service Publish Module
Provider : Repository
1: adapt( )
2: adapt( )
3: getBindingRules( )
4: select( )
5: reply
6: reply
7: reply
8: chooseDwnldMode( )
9: download( )
10: validateDownload( )
11: reply
12: findPath( )
13: bind( )
14: reply

Figure 18: Service adaptation in a client server communication scheme (adaptation in client side)

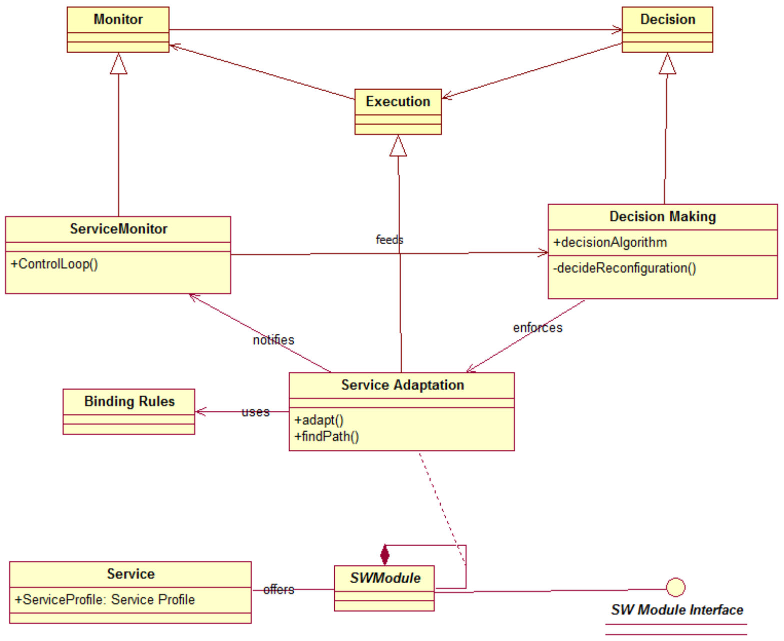


Figure 19: Service Adaptation in the context of the proposed Cognitive Service Provision framework

IMS EVOLUTION FOR SERVICE COMPOSITION AND SERVICE DISCOVERY

The IP Multimedia Subsystem (IMS) architecture is considered by the majority of telecom carriers and service providers as the unifying technology, which brings wireless, wireline and Internet systems together into a seamless environment. IMS provides to the service providers the capability to integrate voice, video, data services as well as advanced services, and provide them on a single platform. Furthermore, IMS provides the opportunity for ubiquitous access for users regardless of service, terminal type or location as well as a uniform environment for billing. In this section the architectural and protocol aspects of the IMS architecture are described, while the evolution of the IMS infrastructure in order to incorporate the proposed hierarchical scheme for services' description and features of the cognitive service provision framework are presented.

IP Multimedia System

IP Multimedia System is an emerging technology, which main goal is to provide to the end user converged application services (data, speech) that are based on the Internet Protocol (IP). IMS is an open and standardized architecture that attempts to integrate service provision by using features and advantages of both mobile communications and Internet world [39], [40]. The main protocols that constitute IMS are Session Initiation Protocol (SIP) [41], Session Description Protocol (SDP) [42], Diameter [43], and Real-time Transport Protocol (RTP) [44], while the main goals through IMS establishment are:

- to provide a common platform for the development, the provision, and the delivery of multimedia services,
- to offer fully integrated real and non-real time services,
- to facilitate the interaction between the user and the services that each user consumes,
- to assure the efficient classification of a group of user that are consuming the same service, under one session or under multiple synchronized sessions.

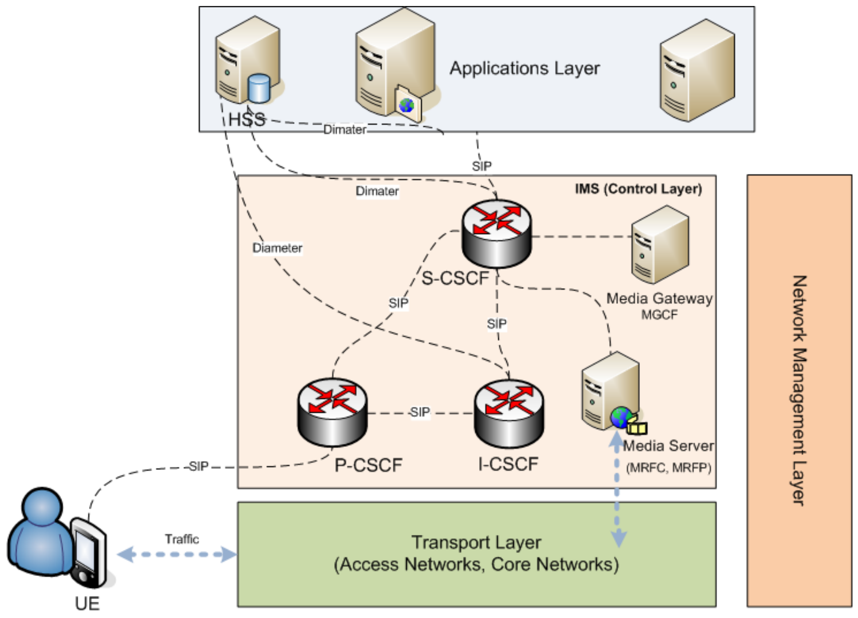


Figure 20: IP Multimedia Sub-System

As depicted on Figure 20, a high level IMS–enabled communications network architecture consists of four layers: a) the Application Layer, b) the core IMS Layer, c) the transport layer, and d) the network management layer. The transport layer includes all those wireless or wired access networks (e.g., LTE, Wifi, ADSL) that are available for the exchange of data packets (e.g., services) among the terminals of the end user and/or application servers. The Network Management (NM) layer comes into the scene in order to highlight the importance of the control of the NM system (NMS) on the transport layer and the necessity of collaboration between the NMS and the service management entities, as it is analysed in the following section. The main functional entities that constitute IMS are described as follows:

- The Home Subscriber Server (HSS) is the main data base of IMS. It is used to store and provide user profile information, security information as well as location information. It is also used for user authentication purposes as well as for services authorization support.
- The Proxy Call Session Control Functions (P-CSCF) server is the first contact interface for SIP messages from the IMS-enabled terminals to the rest IMS network. Terminal attaches to the P-CSCF prior to performing IMS registrations and initiating SIP sessions. P-CSCF operates as the local registrar and as the firewall for the infrastructure. P-CSCF undertakes a) to protect IMS by informing other IMS entities about user's identity, b) to act as the bridge between the visited networks and the home network of the end user, c) to provide charging information in cooperation with the respective policy and charging control entities.
- The Interrogating CSCF (I-CSCF) server is at the edges of an administrative domain. Remote servers (i.e. remote P-CSCFs) discover I-CSCF through DNS calls and contact the latter in order to forward SIP messages to the respective domain. I-CSCF contacts HSS that is associated with in order to find the address of the S-CSCF, where it will select to forward the SIP messages and consequently serve the UE. Thus, S-CSCF could be used for load sharing purposes among multiple S-CSCF nodes.
- The Serving CSCF (S-CSCF) is the most important entity of the IMS infrastructure, where fundamental IMS functions are taking place and all SIP messages are passing through. S-CSCF is used as the local registar server for the users that have been served, and authorizes them by downloading (or updating) user's profile from the HSS. Furthermore, S-CSCF routes the SIP messages to the appropriate application server, by applying routing rules according to the service profile information and taking into account one or more initial Filter criteria (iFC). iFC are stored in the HSS, and provide information about the services that each user is subscribed to. More discussion about iFC is provided below.
- The media gateway and specifically the Media Gateway Controller Function (MGCF) is responsible to control a media gateway.
- The media server and specifically the Multimedia Resource Function Controller (MRFC) provides those functionalities for the adaptation and manipulation of services (e.g., transcoding).

The application Layer includes all application servers that provide an IMS service and interact with the S-CSCF server via SIP messages for control purposes. More information and details about the IMS building blocks and the specified interfaces are available in [45], [46], [47], [48].

IMS and Cognitive Service Provision

Taking into account the cognitive service provision framework for future Internet environments that has been presented in section 3, it is studied below how the IMS infrastructure could be evolved in order to incorporate the proposed hierarchical scheme for services' description and the cognitive cycle paradigm (MDE cycle) for the service management evolution. Service Management in future Internet environments will allow the provision of more sophisticated and personalized services, by exploiting the thousands of services that will be available for delivery either by network operators and third parties or even by simple users that will be able to deliver an application and have the role of a service provider. Thus, in a dynamic and ubiquitous environment with high user mobility, there is the need for the monitoring of the services ecosystem by incorporating also discovery mechanisms for service components (Figure 11). Based on the outputs of the service discovery phase, decision making and execution schemes for services composition are necessary for the realization of the hierarchical model as well as for more efficient reuse of distributed applications and services (Figure 19). Through services composition new sophisticated services will be developed by reusing and correlating existing service components/applications and thus avoiding the continuous specification of new services and the redundant development/delivery of the same application/service by multiple service provision points.

Specifically, the analysis hereinafter is focusing on the service discovery and service composition phase and how these two capabilities are affecting the structural features and the signalling of the IMS environment. **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** illustrates the building blocks and interfaces that are introduced in the IMS architecture, following the design principles and the philosophy of IMS. The service discovery manager and service composer are incorporated in the IMS architecture as trusted SIP servers. The last specification releases of the IMS do not provide the capability for dynamic service discovery and service composition, since the address of the service providers (i.e. application servers) are static and predefined by the network operator in the iFC of the service profile. Hence, an end user knows the service that is subscribed to and according to the specified priorities the appropriate application servers are called. In the literature there are some works that are also proposing the incorporation of the service composer in the IMS infrastructure [49], [50], [51]. At this point it is useful to describe iFC and highlight its importance for the proposed evolution of IMS.

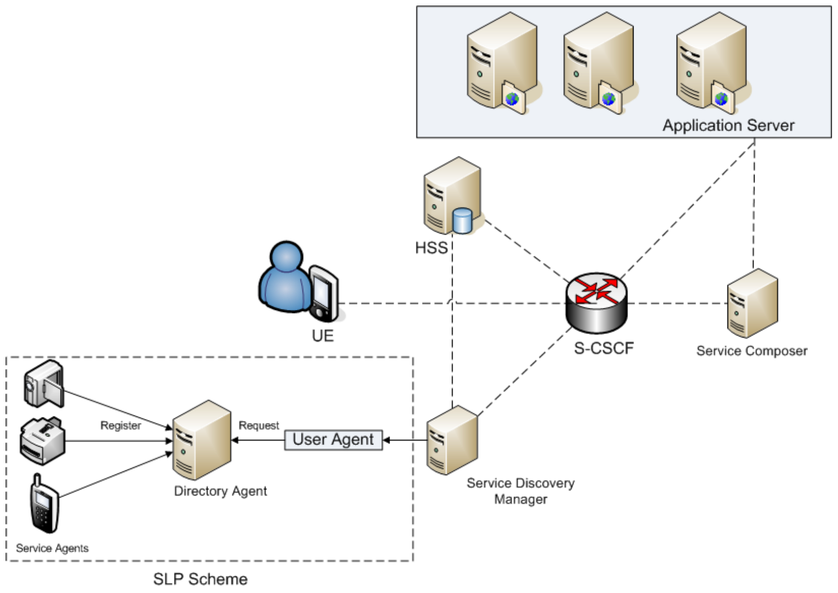


Figure 21: Proposed scheme for the Dynamic Service Composition and Discovery in IMS

In the IMS architecture the coordination of the application servers that provide a service is undertaken by the S-CSCF server. The latter coordinates the available application servers for the provision of a service based on the iFC information of the user profile. The service profile affects the routing of the SIP messages that are exchanged for the service provision. The profile of a service is always associated with a subscriber and includes one or more iFCs that have different priorities. iFC includes those information that will help the S-CSCF server to decide whether an application server should be invoked in order to deliver a service to the subscriber. The service profile is retrieved by the HSS server. The main entities of the iFC profile are depicted in Figure 22 [52].

From the iFC structure it is obvious that composite services have not been taken into account. In the case that more than one application servers are involved the S-CSCF undertakes to forward the SIP message to the appropriate application server based on the “priority field” of the iFC. This mechanism allows only the sequential interaction between the application servers. Thus, a new functionality is necessary in the IMS architecture that will undertake to compose services (i.e. Service Composer) and will undertake the coordination of individual applications. Moreover, the update of the iFC structure is necessary in order to inform the S-CSCF, whether the requested service is a composite one or not. For that purpose, the parameter “Composite Service” could be introduced as a “Service Information” sub-field. This Boolean-type parameter (composite = 1, not-composite = 0) indicates whether the requested service is a composite or not. In the case of a Composite Service the Service Composer undertakes to handle the request, according to the available services. The operator of the end user knows the initial constituent services of the requested composite service or alternatively in a more future scene the user can described the type of the requested composite services.

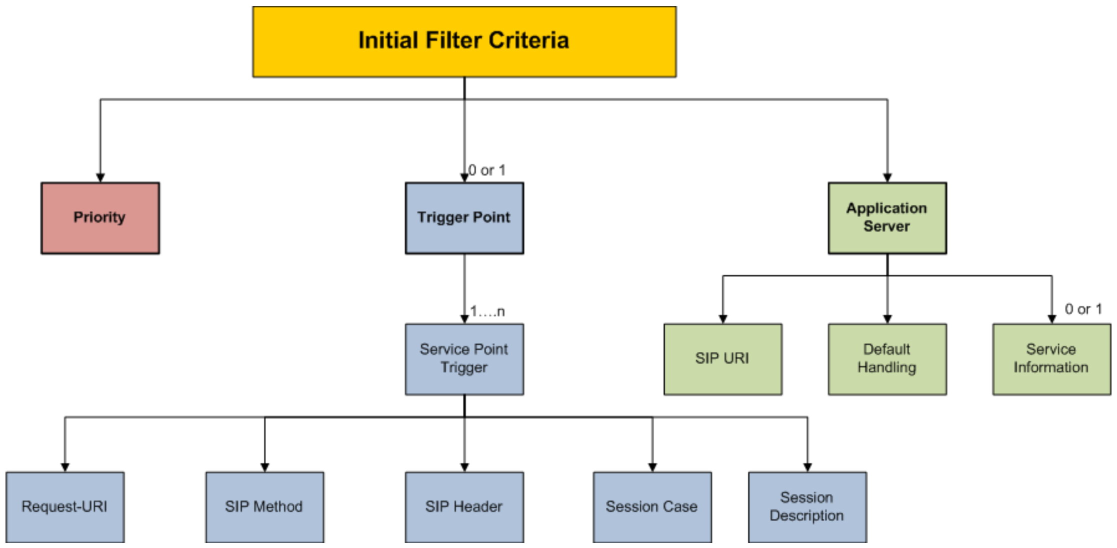


Figure 22: Initial Filter Criteria Information Model

The development of the dynamic discovery of application servers is another capability that will enable the evolvement of the service management, in the context of the IMS, towards a more dynamic and distributed service provision paradigm. As it is mentioned above iFCs and service profiles are predefined by the network operator. In order to allow the dynamic update or indication of the application servers that are described in the iFC, we are proposing the usage of the Service Location Protocol (SLP) and its integration with IMS entities. SLP [53] is a service discovery protocol that will allow the searching of services that Service Agents (SA) publish and User Agent (UA) request. SLP considers also the existence of a Directory Agent (DA) that collects and stores locally discovered services and facilitates the UA discovery process by reducing searching time and signaling e.g., by sending unicast discovery messages instead of multicast packets. The application servers that are provided by after the SLP searching are recorded to the HSS data base in order to be available for other service provision requests.

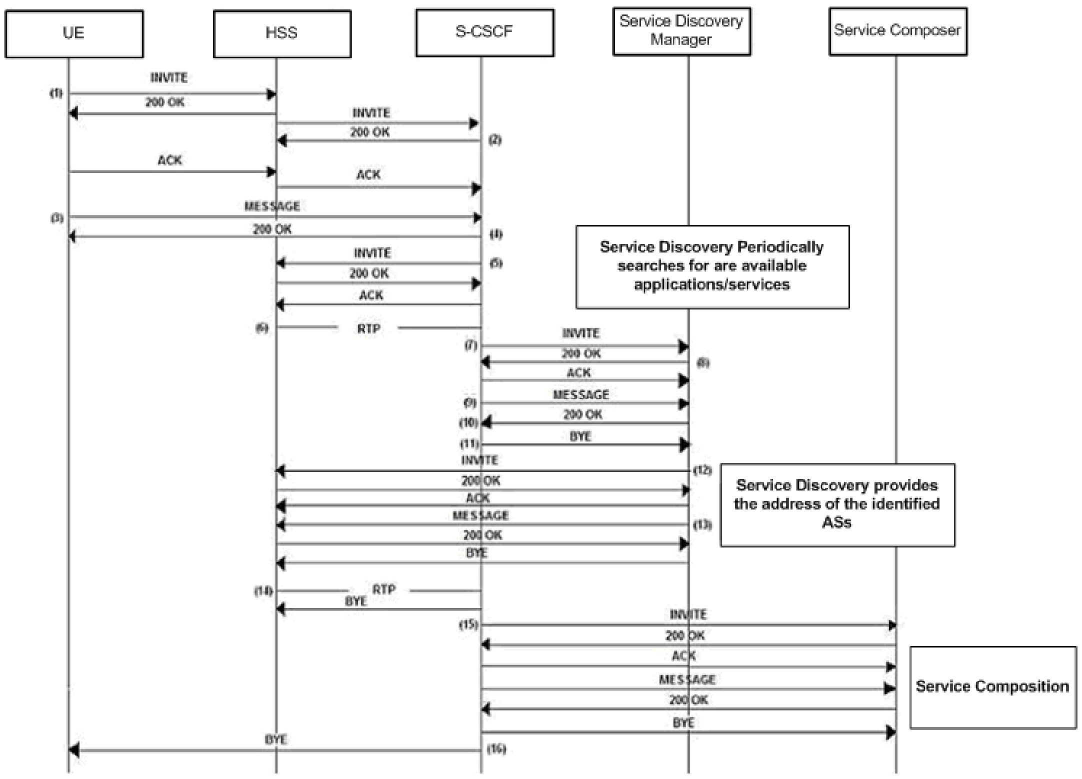
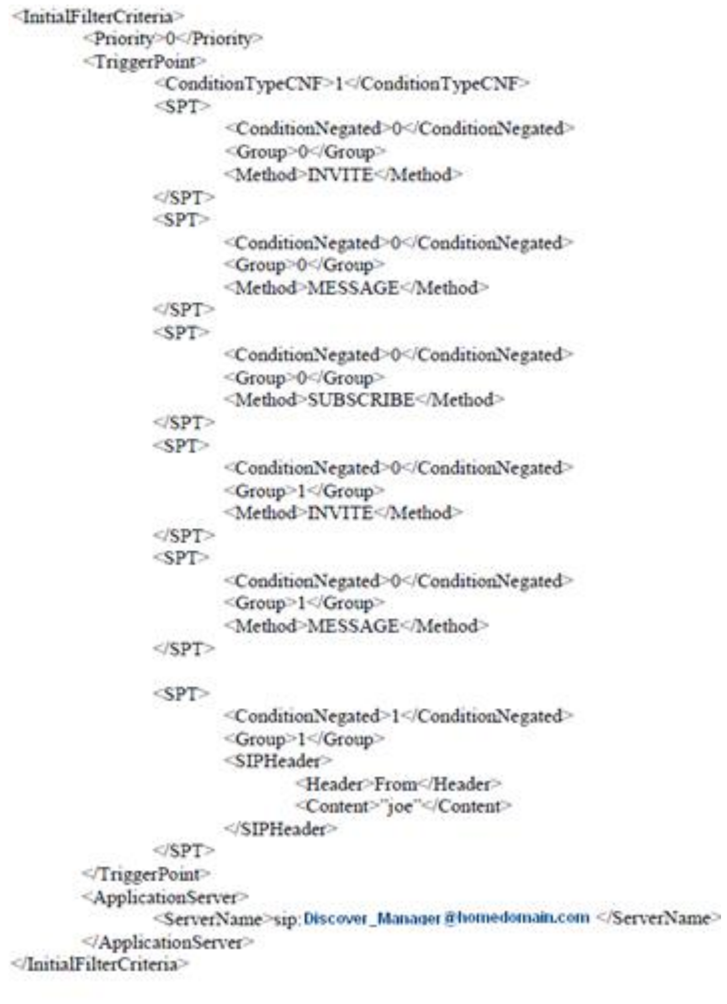

UE
HSS
S-CSCF
Service Discovery Manager
Service Composer
INVITE
200 OK
ACK
MESSAGE
BYE
RTP
Service Discovery Periodically searches for are available applications/services
Service Discovery provides the address of the identified ASs
Service Composition

Figure 23: Signaling for the Discovery and composition of services

The joint usage of the service discovery and service composition entities is described below and depicted in Figure 23, by using SIP messages. The IMS system checks whether a service discovery phase is necessary and whether a composite service has been requested by the end-user. The proposed scheme intends to resolve the service composition for services that are either predefined or not to the IMS. A non-predefined composite service means that the agent of the user is aware about the type of the constituent components of the requested service, but the HSS of the IMS does not know the IP address of the respective service providers. In that case the service discovery mechanism intervenes for the identification of the appropriate service providers. The steps of the proposed scheme are as follows:

1. The end user requests a service either composite or not.
2. In case that the application server that is requested by the user is not registered in the IMS data base (i.e. HSS) then the field “Application Server” of the iFC of the respective service profile is completed with the address of the Service Discovery server of the local IMS area. Thus, it is indicated that a discovery phase by using SLP is necessary to be initiated by the Service Discovery server for the detection of the application server that is associated with the requested service.
3. The S-CSCF retrieves the user’s profile (Initial Filter Criteria) from the HSS. An example of the relative iFC is described in Figure 24.
4. If the address of the Service Discovery Manager has been indicated in the ‘Server Name’ field then C-SCSF calls the latter and indicates the type of the applications components that are required.
5. The Service Discovery Server sends a discovery request message to the User Agent in order to start searching for application servers.
6. The User Agent returns to the Service Discovery Manager the addresses of the application servers that have the ability to deliver the service (composite or not), that the user has requested.
7. The Service Discovery Manager having collected the addresses of the application servers updates the profile of the user on the HSS.

8. After the update of the user profile the S-CSCF downloads the updated user profile (iFC).
9. The S-CSCF has collected the necessary application server, checks the proposed ‘Composite Service’ subfield of the ‘Service Information’ filed in order to detect whether a composite service has been requested or not. If ‘Composite Service = 1’ then the S-CSCF triggers the Service Composer entity, which undertakes to make the composition of the service and thereinafter S-CSCF contacts the services (application servers) that the user has requested.



```
<InitialFilterCriteria>
        <Priority>0</Priority>
        <TriggerPoint>
                <ConditionTypeCNF>1</ConditionTypeCNF>
                <SPT>
                        <ConditionNegated>0</ConditionNegated>
                        <Group>0</Group>
                        <Method>INVITE</Method>
                </SPT>
                <SPT>
                        <ConditionNegated>0</ConditionNegated>
                        <Group>0</Group>
                        <Method>MESSAGE</Method>
                </SPT>
                <SPT>
                        <ConditionNegated>0</ConditionNegated>
                        <Group>0</Group>
                        <Method>SUBSCRIBE</Method>
                </SPT>
                <SPT>
                        <ConditionNegated>0</ConditionNegated>
                        <Group>1</Group>
                        <Method>INVITE</Method>
                </SPT>
                <SPT>
                        <ConditionNegated>0</ConditionNegated>
                        <Group>1</Group>
                        <Method>MESSAGE</Method>
                </SPT>

                <SPT>
                        <ConditionNegated>1</ConditionNegated>
                        <Group>1</Group>
                        <SIPHeader>
                                <Header>From</Header>
                                <Content>"joe"</Content>
                        </SIPHeader>
                </SPT>
        </TriggerPoint>
        <ApplicationServer>
                <ServerName>sip:Discover_Manager@homedomain.com </ServerName>
        </ApplicationServer>
</InitialFilterCriteria>
```

Figure 24: iFC for service discovery activation

The flexibility levels of the service management that could be introduced and efficiently implemented is a fundamental issue. The description of the services components that constitute a service, their priorities and in general the place for the publishing and description of a composite service (end user terminal or network operator side) are issues that affect the flexibility of the system and the changes that should take place to existing service provision platforms e.g., IMS.

CONCLUSION AND FUTURE RESEARCH DIRECTIONS

From the above analysis it is obvious that new capabilities could be deployed for advanced service provision. However the increased complexity for service management is one of the issues that should be addressed. For the fast evolution of the Future Internet services ecosystem it is necessary the specification of the interfaces among the various modules of the Cognitive Service Provision framework and mainly the provision of APIs and tools (e.g., Web 2.0 tools) that will allow the application developers to deploy fast and easy software components (*SW Components*) for both the service providers and the end users. Furthermore, there is the need of common information models for the uniform description of services and application as well as of a common representation scheme for the graph of services and the graph of software components.

The service management behavior is affected by the operation of the network management systems and vice versa. For that reason the federation and the cooperation of these two domains is a very important parameter and a challenge for future research. Before presenting how this link could be used in practice it is useful to sketch out the future Internet network management systems.

The last decade there is a lot of literature and research work for the automation of network management by reducing the human intervention and handling complex situations. The self-management of Future Internet infrastructures necessitates the introduction of decision making techniques and the capitalization of the existing knowledge and policy frameworks, as it is depicted in **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**, where the cognitive cycle is also present for the automation of network management tasks. For efficient and scalable network management, where various stakeholders participate (network operators, service providers, end users), a distributed approach is required. Dynamic network (re)-configuration, in many cases, is based on cooperative decision making of various Future Internet devices and distributed network management service components. Hints and requests/recommendations are exchanged among the layers, in order to indicate a new situation or an action for execution. The automated and dynamic incorporation of various layers/levels requirements (e.g., SLAs) into the management aspects, provides also novel features to network management capabilities. Moreover, the resolution of conflicting requests is an issue of situation awareness and elements' domain policy prioritisation.

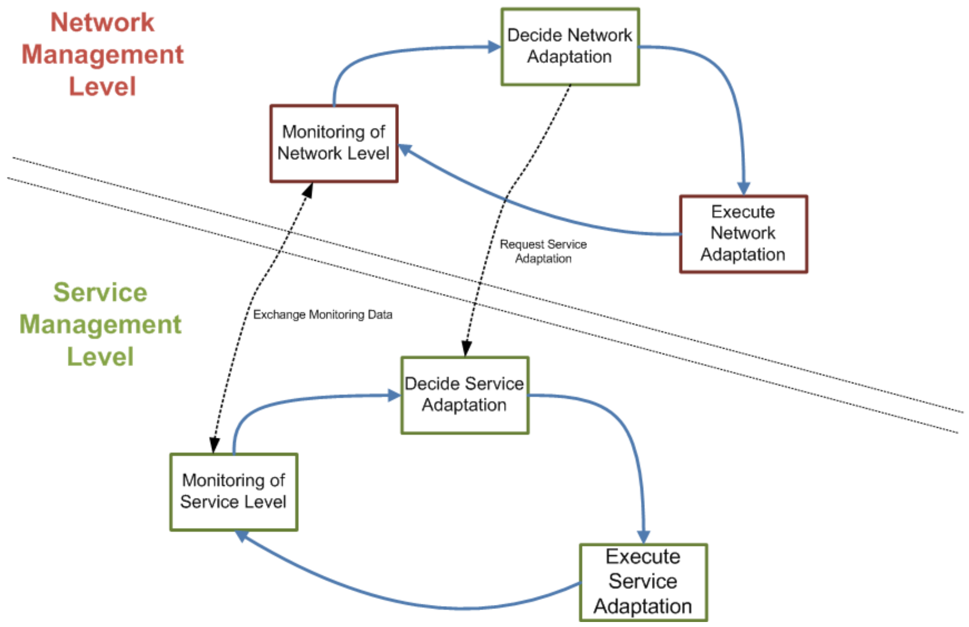

Network Management Level
Decide Network Adaptation
Monitoring of Network Level
Execute Network Adaptation
Request Service Adaptation
Exchange Monitoring Data
Service Management Level
Decide Service Adaptation
Monitoring of Service Level
Execute Service Adaptation

Figure 25: Service and Network Management Cognitive Cycles interaction

For the cooperation between the network and service management systems the following communication channels should be specified:

a) The first one is used for the exchange of monitoring data that is sensed locally either at the service or at network management level. The network management system could use service-level data as an input for the situation deduction and specifically for the identification of possible faults or optimization opportunities. Furthermore, service-level parameters could be used by the objective functions that the network management system should solve for the decision taking phase, according to the identified fault. On the other hand the service management level could exploit network-level monitoring data e.g., for the selection of the most efficient application server taking into account network conditions or for the building of the service path.
b) The second communication channel is used by the network management system in order to trigger service adaptations. Each network management system has a list of configuration actions that could be triggered as a remedy to a detected fault. Hence, service adaptation (e.g., service re-composition) is an additional configuration that could be triggered by the network management system.

Existing network management systems have limited capabilities for their cooperation with service management systems e.g., IMS. Hence, this could be considered as an important area for further research and specification.

In the context of this chapter we presented a novel approach for Service Provision which incorporates cognitive features and promotes itself as an ideal paradigm for the Future Internet era. Initially we discussed the challenges of the Future Internet Networks and then presented in details the proposed Cognitive Service Provision framework. Its key differentiating factor lays in the introduction of the cognitive cycle in the context of Service Delivery and Service Adaptation and supported by the modeling of a Service as a bundle of Software Components. Through extensive UML class and message sequence diagrams we analyzed the key concepts of the proposed framework and the extension of the IP Multimedia Subsystem through our approach.

ACKNOWLEDGMENT

This work is supported by the European Commission 7th Framework Programme through the Self-NET project (http://www.ict-selfnet.eu) and E3 project (https://ict-e3.eu). The authors would like to acknowledge Ms. Lorena Kolai and Ms. Penelope Loucareli help for IMS investigation.

Keywords list: service management, cognitive service provision, IMS, service composition, service discovery, network management

REFERENCES

[1] Apostolos Kousaridas, Gérard Nguengang, Julien Boite, Vania Conan, Vangelis Gazis, Tilemachos Raptis, Nancy Alonistioti. (2010). "An experimental path towards Self-Management for Future Internet Environments", In: "Towards the Future Internet - Emerging Trends from European Research". Edited by Georgios Tselentis, Alex Galis, Anastasius Gavras, Srdjan Krco (pp. 95 – 104).

[2] Ganek, A. G. (2003). The dawning of the autonomic computing era. IBM Systems Journal.

[3] Nikos Houssos, Vangelis Gazis, Athanassia Alonistioti (2004) "Enabling Delivery of Mobile Services Over Heterogeneous Converged Infrastructures. Information Systems Frontiers 6(3): 189-204.

[4] Tselikas, N. D., Dellas, N. L., Koutsoloukas, E. A., Kapellaki, S. H., Prezerakos, G. N., and Venieris, I. S. (2007). Distributed service provision using open APIs-based middleware: "OSA/Parlay vs. JAIN" performance evaluation study. J. Syst. Softw. 80(5): 765-777

[5] 3GPP TS 22.127. (2009). Service Requirements for the Open Services Access (OSA); Stage 1 (Release 7)", V7.1.0.

[6] 3GPP TS 22.105. (2006). Services and Service Capabilities (Release 8) V8.0.0.

[7] Boualem Benatallah et all. "Declarative Composition and P2P Provisioning of Dynamic Web Services", ICDE 2002.

[8] David Harel and Amnon Naamad. (1996) .The STATEMATE semantics of statecharts", ACM Transactions on Software Engineering Methodology, vol. 5, num. 4.

[9] Joshua Lubell, "Professional XML Meta Data", Chapter 14, Wrox Press, 2001, ISBN 1861004516 (http://ats.nist.gov/psl/xml/process-descriptions.html)

[10] Sun's JINI, http://sun.com/jini

[11] Universal Description, Discovery and Integration protocol (UDDI), http://www.uddi.org

[12] IETF Service Location Protocols (SLP), http://www.ietf.org/html.charters/OLD/svrloc-charter.html

[13] UPnP, http://www.upnp.org

[14] DNS-SD, http://www.dns-sd.org

[15] Bluetooth's Service Discovery Protocol (SDP), www.bluetooth.com

[16] Stephanos Androutselis and Diomidis Spinellis. (2004). A Survey of Peer-to-Peer content distribution technologies. ACM Computing Surveys, vol.36, no.4.

[17] Demetrios Zeinalipour-Yazti, Vana Kalogeraki, Dimitrios Gunopulos. (2004) .Information Retrieval Techniques for Peer-to-Peer Networks. Computing in Science and Engineering, vol. 06, no. 4, pp. 20-26.

[18] X. Li and J. Wu. (2006) .Searching Techniques in Peer-to-Peer Networks. Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks, J. Wu (ed.), Auerbach Publications, pp. 613 - 642.

[19] Tao Yu and Kwei-Jay Lin. (2005). Service Selection Algorithms for Composing Complex Services with Multiple QoS Constraints. In proc. of the 3rd International Conference on Service Oriented Computing (ICSOC2005), Amsterdam, The Netherlands.

[20] D.Chacraborty, A.Joshi, Y.Yesha, T.Finin, "GSD: A Novel Group based Service Discovery Protocol For MANETS", MWCN 2002

[21] A. Helal, N. Desai, V. Verma, and C. Lee "Konark:A Service Discovery and Delivery Protocol for Ad-Hoc Networks", Proceedings of the IEEE Wireless Communication and Networking Conference (WCNC 2002), New Orleans, LA, March 2003.

[22] Yu Yang; Hassanein, H.; Mawji, A. (2006). Efficient Service Discovery for Wireless Mobile Ad Hoc Networks. Computer Systems and Applications, 2006. IEEE International Conference on. Page(s):571 – 578.

[23] Dipanjan Chakraborty, Anupam Joshi, Yelena Yesha, and Tim Finin. (2002). GSD: A novel group-based service discovery protocol for manets in Proc. of 4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN).

[24] Dipanjan Chakrabory, Anupam Joshi. (2001). Dynamic Service Composition: State-of-the-Art and Research Directions. Technical Report TR-CS-01-19, University of Maryland.

[25] D. Chakraborty, Y. Yesha, and A. Joshi. (2004). A Distributed Service Composition Protocol for Pervasive Environments. In IEEE Wireless Communcations and Networking Conference (WCNC). vol.4, pp. 2575- 2580.

[26] Basu P. Wang Ke. Little T.D.C. (2002). A novel approach for execution of distributed tasks on mobile ad hoc networks. IEEE Wireless Communications and Networking Conference, 2002. WCNC2002., vol.2, no., pp. 579- 585 vol.2.

[27] Shankar R.Ponnekanti and Armando Fox. (2002). SWORD: A Developer Toolkit for Web Service Composition, Technical Report, Stanford University.

[28] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. Z. (2003). Quality driven web services composition. In Proceedings of the 12th international Conference on World Wide Web pp. 411 – 421.

[29] Hirschfeld R. (2003) AspectS - Aspect Oriented Programming with Squeak, Objects, Components, Architectures, Services and Applications for a Networked World, LNCS 2591, Springer.

[30] Kiczales G. et all, (2001). An Overview of AspectJ. In proceedings of the 2001 European Conference on Object-Oriented Programming.

[31] Kniesel G. et all. (2001). JMangler- A Framework for Load Time Transformation of Java Class Files. Source Code Analysis and Manipulation Workshop.

[32] OWL-S 1.0 Release, http://www.daml.org/services/owl-s/1.0

[33] SOAP Specification, http://www.w3.org/TR/soap

[34] WSDL Specification, http://www.w3.org/TR/wsdl

[35] UDDI specification http://www.oasis-open.org/committees/uddi-pec/doc/spec/v3/uddi-v3.0.2-20041019.htm

[36] 3GPP Technical Specification. (2010). 23.228 v10.1.0, ”IP Multimedia Subsystem (IMS); Stage 2”

[37] Tao Yu, Kwei-Jay Lin, “Adaptive Algorithms for Finding Replacement Services in Autonomic Distributed Business Purposes”,

[38] Thomas H. Cormen, Charles E. Leirson, Ronald L. Rivest, Clifford Stein. (2004). Introduction to Algorithms. Second Edition, MIT Press.

[39] M.A Qadeer, A.H. Khan, J.A. Ansari, S. Waheed, “IMS Network Architecture," International Conference on Future Computer and Communication (ICFCC 2009), vol., no., pp.329-333, 3-5 April 2009

[40] S. Loreto, T. Mecklin, M. Opsenica, H.-M. Rissanen, "IMS service development API and testbed," IEEE Communications Magazine, vol.48, no.4, pp.26-32, April 2010

[41] IETF RFC 3261: "SIP: Session Initiation Protocol".

[42] IETF RFC 4566: " SDP: Session Description Protocol".

[43] IETS RFC 3588: “Diameter Base Protocol”.

[44] IETF RFC 3550: “RTP: A Transport Protocol for Real-Time Applications”.

[45] 3GPP TS23.228, “IP Multimedia Subsystem (IMS);Stage 2”.

[46] 3GPP TS24.229, “IP Multimedia Call Control Protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3“.

[47] 3GPP TS23.218 “IP Multimedia (IM) session handling; IM call model; Stage 2”.

[48] 3GPP TS29.228, “IP Multimedia (IM) Subsystem Cx and Dx interfaces; signaling flows and message contents”.

[49] Juan Miguel Espinosa Carlin, (2008). Realizing Service Composition in the IP Multimedia Subsystem. In: Proceedings of the 1st. ITU-T Kaleidoscope Conference Innovations - Future Network and Services. Geneva, Switzerland.

[50] Brajdic, A.; Lapcevic, O.; Matijasevic, M.; Mosmondor, M.; (2008). Service composition in IMS: A location based service example. Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium on, vol., no., pp.208-212.

[51] Dinsing, T, Eriksson, G, Fikouras, I, Gronowski, K Levenshteyn, R, Pettersson, P and Wiss, P., (2007). Service composition in IMS using Java EE SIP servlet containers. Ericsson Review, Vol. 84 3, pp. 92-96.
[52] Poikselka M., Georg Mayer (2009), "The IMS: IP Multimedia Concepts and Services, 3rd Edition", ISBN: 978-0-470-72196-4.
[53] IETS RFC: 2608 "Service Location Protocol".