Context-Aware Service Provisioning in All-IP Networks

Dimitris Karteris, Christos Xenakis, and Lazaros Merakos

Communication Networks Laboratory,
Department of Informatics and Telecommunications,
University of Athens, 15784, Athens, Greece
E-mail: {dkart, xenakis, merakos}@di.uoa.gr

ABSTRACT

An architecture that supports context-aware service provision in the All-IP network is proposed and analyzed. The main goal of this architecture is to provide representation of a nomadic user in each visited computing and communication environment so that his services are tailored to his preferences and needs. The user representing entity collects context information that pertains to the interactions between the user and his services in the current environment. The collected information is fed to the services along with any relevant preferences defined in the user profile. The services use the received information in order to adapt to their context. Actors in this architecture such as the users, the network providers and the service providers are represented by software agents, which exchange context information. A nomadic user is represented by a mobile agent, which follows him wherever he may roam. User agent mobility management is carried out by means of the cooperation of agents that belong to the network providers of the visited environments, and their interaction with the Mobile IP protocol that is used for terminal mobility in All-IP networks.

I. INTRODUCTION

The evolution of mobile networking has introduced a new telecommunication *status quo* whose main characteristic is the coexistence of wired (LAN, PSTN/ISDN/xDSL, etc.), and wireless (WLAN, GSM, UMTS, etc.) networks, which are managed by different principles, and cover both indoor and outdoor environments. In an attempt to integrate the various telecommunication systems, current research efforts in networking lead to architectures that are known as All-IP networks. These architectures aim to provide fully IP-based service offering that is voice, data, and multimedia services over an IP bearer [1]. The Internet protocol is used for the deployment of a unified backbone that federates the different access technologies (see Figure 1). A key issue in the All-IP architecture is the provision of a terminal mobility mechanism that ensures seamless roaming between the access technologies. As potential solutions, the Mobile IPv4 [2] and Mobile IPv6 [3] protocols have been considered.

As the Internet is expanded to a real ubiquitous network, it accommodates the needs of nomadic users who move between several places, such as their home, their office, conference rooms, hotels, automobiles, airplanes, etc. [4]. So, nomadic users operate in several environments, which may be quite different from each other in terms of computing platforms, communication infrastructure, as well as services provided in their realm.

While moving between different environments, nomadic users wish to receive personalized services from specific service providers by using whatever access technology and terminal. Moreover, when they receive a specific service from different service providers, they wish to experience it as close as possible to a common and predefined *look and feel*. However, the interaction between a user and his services is not affected only by the access technology, and the terminal used. Physical variables such as time and environmental conditions may also affect this interaction. Moreover, information such as his location or his current activity should also be taken into account. The whole of the aforementioned information, which is relevant to the user-service interaction, constitutes the *context* of each service. Influenced by the definition of context by Dey *et al*. [5], a more formal definition of the service context is given: *service context* consists of any information that can be used to characterize the situation of any entity (person, place, or object) that is considered relevant to the interaction between a user and a service, including the user and the service themselves. The service provisioning requirements mentioned above call for *context aware services* [6]. These services adapt their behavior, or the content they process, to their service context in a transparent way.

In this paper, a new architecture, which aims to support the provision of context-aware services in the All-IP network architecture, is proposed. It is called CASPAIR (Context-Aware Service Provisioning in All-IP netwoRks). The main goal of this architecture is to provide representation of a nomadic user in each visited computing and communication environment so that his services are tailored to his preferences and needs. The user representing entity collects context information that concerns the interactions between the user and his services in the current environment. The collected information is fed to the services along with any relevant preferences defined in the user profile. The services use the received information in order to adapt to their context.

The proposed architecture uses the mobile agent technology, which provides a number of benefits in the creation of distributed systems [7, 8]. Due to their autonomous and independent nature, mobile agents may act on behalf of the application that spawned them, even if communication between them is temporarily lost, thus providing support for disconnected operations. Moreover, the ability of mobile agents to migrate to the location of the resources they access, results in less remote communication (i.e. reduced bandwidth consumption), and less overall latency in a series of interactions. Another benefit of mobile agents is their ability to adapt dynamically, as they can sense their execution environment and react autonomously to changes. In addition, the mobile agent paradigm offers more flexibility than the traditional client-server communication, as mobile agents can be used to dynamically change the interface between a client and a server. Finally, mobile agents are generally platform and transport layer independent thus providing an optimal solution for heterogeneous distributed computing environments.

The rest of this paper is organized as follows. In Section II, the basic principles of the proposed architecture are pointed out. Section III gives a description of the main components that comprise the CASPAIR architecture. In Section IV, a set of basic operations is analyzed in order to showcase the system functionality. Finally, the conclusions are discussed in Section V.

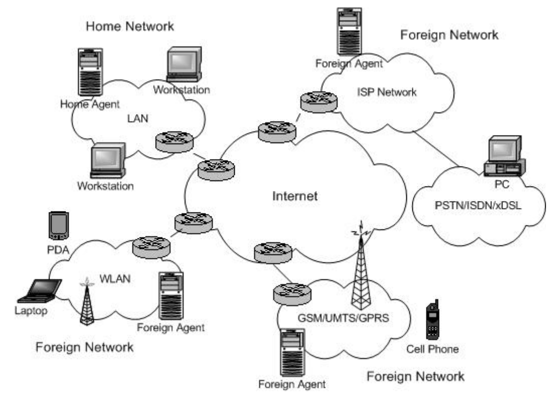

Figure 1 An All-IP network that federates several wired and wireless access technologies.

II. THE CASPAIR APPROACH

The basic goal of the CASPAIR approach is to provide a framework that supports continuous context acquisition in every environment visited by nomadic users, and facilitates the contextualization of the services used. This is achieved by using software agents to represent the entities that are relevant to the interaction between the user and his services. These entities are illustrated in Table 1, where information characterizing each one of them is also given.

Each agent's responsibility is to constantly gather all the context information concerning its entity and provide this information to other agents in the framework. Context information is eventually supplied to the service agents, who use it in order to adapt the behavior of their corresponding services.

The key agent in the CASPAIR approach is the one representing the user. User representation is based on the *user profile*, which contains a set of service-related preferences that designate how the services should be experienced, a set of preferences that concern the user's CASPAIR sessions, and, finally, the capabilities of all the user terminals. Besides the user profile, the agent in question obtains context information collected by agents residing in the currently visited environment. In this way, the user agent gains knowledge of the environment in which the user and his services operate.

When a service is activated, the user agent identifies the service relevant context components. Then, it extracts all the information that is relevant to the user-service interaction from the user profile, and constructs a *service specific profile*. The service specific profile is a specification of all possible user configurations that a service may assume, and depends on a subset of the service relevant context components. The service specific profile along with the values of the service relevant context components is eventually sent to the agent that represents the service. The service agent applies the values of the context components to the service specific profile and produces a specific configuration that holds under these values. In the following, adaptation is carried out based on the current instance of the service context, which is the aggregation of the currently holding user defined configuration and the current values of the context components. The service context is not invariable; any updates to the values of the service relevant context components detected by the user agent are communicated back to the service agent and adaptation recurs.

Entity	**Information**
User	identity, location, time, activity, service preferences, current terminal, environmental conditions
Service	adaptable characteristics, current configuration, capabilities
Network Provider	network type and capabilities, network load, local service providers, billing scheme, security policies
Service Provider	available services, available execution resources, billing scheme, security policies
Terminal	terminal type and capabilities, available resources, installed software

Table 1 Entities that are relevant to the user-service interaction and corresponding information used to characterize them.

Before advancing to the details of the CASPAIR architecture, a categorization of the service types that are assumed is defined. The three service categories are: *subscribed*, *guest* and *hybrid services*. Subscribed services are offered by home or third-party service providers and require previous subscription in order to be used (e.g., stock market quotes report, electronic mail service, personal bookmarks, Video on Demand, etc.). Guest services are offered by a local service provider to the users (both home and visiting) currently attached to the network without requiring any subscription (e.g., printing, slide projection, room interaction, etc.). Finally, hybrid services are subscribed services that can be offered by a service provider of a visited network, even if the visiting users have not subscribed to this provider (e.g., nearby restaurant retrieval, local weather report, emergency calls, etc.).

III. THE CASPAIR COMPONENTS

In the proposed architecture, a number of logical entities are defined in order to group the functionality provided by agents: the Profile Entity (PE), the Network Entity (NE), the Service Provider Entity (SPE), and the Terminal Entity (TE). Each entity corresponds to an agent system that is an agent execution environment where agents reside and perform portions of their life cycle. Agents may migrate from one entity to another during their lifetime.

A. Profile Entity Agents

The Profile Management Agent (PMA) is a stationary agent that resides in the PE and handles the activation of CASPAIR sessions as well as the management of user profiles. When a PMA receives a request for the activation of a user's session, it searches for the user's profile in the profile database, and, after a successful authentication, it creates the user's Profile Agent (PA).
PAs are mobile agents that act as user representatives. Each PA remains alive throughout a user session and follows the user, while he roams between different networks, by migrating to the NE of each visited network. The PA receives user requests for the activation of services and forwards them to appropriate service providers. The most important functionality of a PA is the collection of service related context information from other agents in its local CASPAIR system. The collected context information is sent to the service agents that use it in order to adapt the behavior of their corresponding services.

B. Network Entity Agents

Network Agents (NAs) are stationary agents that reside in NEs and provide access to CASPAIR. They receive session initiation/resumption requests, which they forward to appropriate CASPAIR agents. For each user, there is a Home Network Agent (HNA), which is the NA that resides in the home network, i.e. the network to which the user has subscribed for the provision of CASPAIR functionality. All other NAs in foreign networks are considered as Foreign Network Agents (FNAs) to the user in question.
Agents that reside in user terminals can contact the NA of their local network by obtaining its identifier and the address of the corresponding NE, either through manual configuration or by means of an automated way, such as DHCP or router advertisements.
NAs that belong to different network providers perform mobility management for the PAs so as to enable them to follow the roaming users. Each NA holds three lists that assist him in PA mobility management: a *network list* that contains information for other CASPAIR-providing networks, a *home users list* that contains information for all the users to whom the NA in question acts as a HNA, and a *visiting users list* that contains information for all the users currently visiting the local network. The network list consists of (*networkID, naIdentifier, naLocation, subnetPrefixesList*) quadruplets where *networkID* is the network's identity, *naIdentifier* is the identifier of the network's NA, *naLocation* is the address of the NE where the NA resides, and *subnetPrefixesList* is the list of the subnet prefixes of the IP addresses that belong to this network. The *home users list* contains (*userID, paIdentifier, paLocation, addressList*) quadruplets where *userID* is the user's identity, *paIdentifier* is the identifier of the user's PA, *paLocation* is the address of the NE where the user's PA is currently located, and *addressList* is a list that contains the home IP addresses of all the user's terminals that run Mobile IP. The *visiting users list* contains (*userID, homeNetworkID, paIdentifier*) triplets where *userID* is the user's identity, *homeNetworkID* is the identity of the user's home network, and *paIdentifier* is the identifier of the user's PA.
Finally, each NA provides network related context information to the PAs that reside in its NE.

C. Service Provider Entity Agents

The Service Provider Agent (SPA) is a stationary agent that resides at the SPE, and handles requests for the activation of services. Each user may subscribe to a number of service providers in order to receive services. The SPAs of the providers in question are considered as Subscribed Service Provider Agents (SSPAs) for this user. A user may also receive services by the SPA of a visited network, even if he has not subscribed for service provision by that network. However, in such cases, he may access only the guest and the available hybrid services. An SPA that provides only guest/hybrid services to a user is considered as Guest Service Provider Agent (GSPA) for the user in question. Each SPA is a context information provider for the PAs. An example of such information is the set of offered subscribed, guest, or hybrid services.
When a user activates a service, the corresponding request is initially sent to the user's PA, which in turn forwards it to the appropriate SPA on the basis of the service type. The SPA creates a corresponding Service Agent (SA) that handles service execution and representation in the CASPAIR framework. The SA contains the service logic, the service user interface, the service specific profile, and the current values of the service relevant context components. Although, an SA is created and executed in an SPE, it may migrate to the NE of the network that the user currently visits. However, this depends on the visited network's policy as well as the NE's resource availability.

D. Terminal Entity Agents

The Terminal Agent is a stationary agent that resides in the TE and enables the interaction of the terminal with the

CASPAIR system. When the user logs in successfully, the TA receives the graphical user interface (GUI) software modules for the management of the CASPAIR session and the activation of services. Moreover, when a service is activated, the TA receives the GUI modules for the interaction with the service in question. Besides service presentation, the TE can be also used for service logic execution either by the migration of the SA to the TE, or by the dispatch of service logic code from the SA to the TA. However, this feature depends on the terminal capabilities as well as the available terminal resources.
Finally, the TA provides context information, such as terminal type, GUI module versions, resource availability, etc., to PAs.

IV. BASIC CASPAIR OPERATIONS

Based on the description of the CASPAIR components, the most important CASPAIR operations are presented in order to give a better idea of the overall system functionality.

A. Session Initiation

A user may initiate a CASPAIR session either when he resides at his home network or at a foreign one. Figure 2 illustrates a session initiation procedure at a foreign network. The TA in the user terminal calls the *initiateSession()* method on the FNA (1). Among the information provided to the FNA, through the previous method call, are the user credentials (user id, home network id, password, etc.), the TA identifier, the TE address, as well as terminal related context information. A user id is unique inside a network provider's domain so the combination of the username and the network provider id uniquely identifies the user in the global network. The FNA identifies that the user is not at his home network, so it consults its network list in order to retrieve the identifier of the user's HNA as well as the address of the corresponding Home Network Entity (HNE). By using the retrieved information, the FNA contacts the HNA, and forwards the session initiation request to it (2). The request is enhanced with the address of the Foreign Network Entity (FNE). In the following, the HNA relays the session initiation request to the PMA (3). The PMA authenticates the user, and searches for his profile in the profile database. Then, the PMA creates the user's PA, based on the retrieved profile, and instructs it to migrate to the FNE (4, 5). So, although the PA is created at the home network, it migrates to the currently visited network in order to follow the user.
As soon as the PA is established at the FNE, it requests context information from the context providers of the visited network by means of a procedure that is described in a following section. Finally, the PA dispatches the CASPAIR session management GUI (CSMGUI) to the TA (6). The dispatched GUI depends on the current terminal and network context, as well as the user preferences. By means of this GUI, a user may pause/resume/terminate CASPAIR sessions, set session or GUI specific preferences, and, eventually, activate his services.

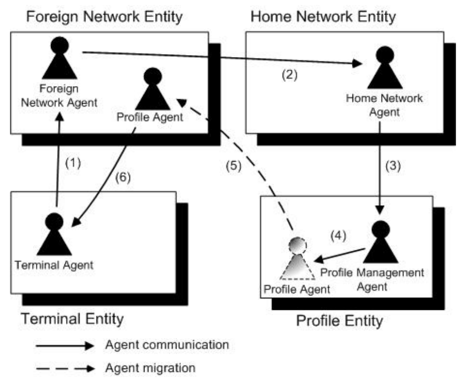

Figure 2 Session initiation at a foreign network.

B. Session Resumption

Session resumption implies that the user has previously paused his CASPAIR session. During the pause interval, the PA stays "alive" in some NE. The key issue in session resumption is for the PA to be tracked down, and associated with the user's TA. Moreover, if the session is not resumed in the network where it was paused, the PA should be instructed to migrate to the currently visited network. Figure 3 illustrates a scenario in which the user pauses his session at his home network and resumes it in a foreign one. Additionally, the session is resumed in another terminal (1). First, the TA calls the *resumeSession()* method on the current FNA (2), providing the user's id, the home network id, the TA identifier, the TE address, as well as context information about the new terminal. Then, the FNA consults its visiting users list in order to obtain the PA identifier. After failing to track down the PA locally, the FNA consults its network list and obtains the information needed to contact the user's HNA. Following, the FNA forwards the session resumption request to the HNA by calling the *resumeSession()* method (3). The HNA, in turn, accesses its home users list in order to retrieve the PA's current location. The HNA locates the PA at the HNE and instructs it to migrate to the FNE by calling the *moveTo()* method (4, 5). As soon as the PA migrates to the FNE, it gathers context information concerning the current network. Finally, the PA sends the CSMGUI to the new terminal (6).

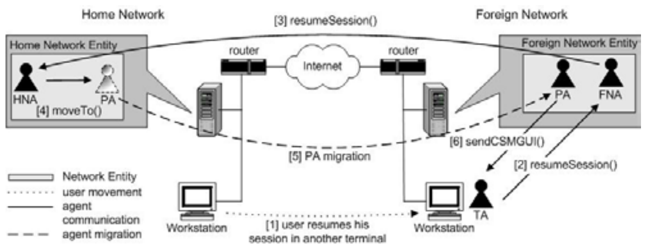

Figure 3 Session resumption at a foreign network.

The terminal switch that takes place in the previous scenario does not prevent session resumption. This is due to the fact that session state information (e.g., listing of all the activated services and references to their corresponding SAs) is not preserved in the terminals, but in the PA instead. Moreover, any information that is held by the PA and concerns the currently used terminal is updated in each session resumption request. As a result, a user may transfer his session from one terminal to another.

C. Handoff triggered PA migration

In the previous cases, the PA migration was the result of a user action: a session initiation request in a foreign network or a session resumption request in a network different from the one the session was paused. Although the PA mobility management mechanism works under the circumstances just mentioned, this is not the case when the user roams between networks and keeps his session active in a terminal. In such a case, the CASPAIR system has no way of knowing that a handoff occurred since the MIP protocol ensures roaming transparent to the overlying layers. As a consequence, the PA will not be instructed to migrate to the currently visited network.

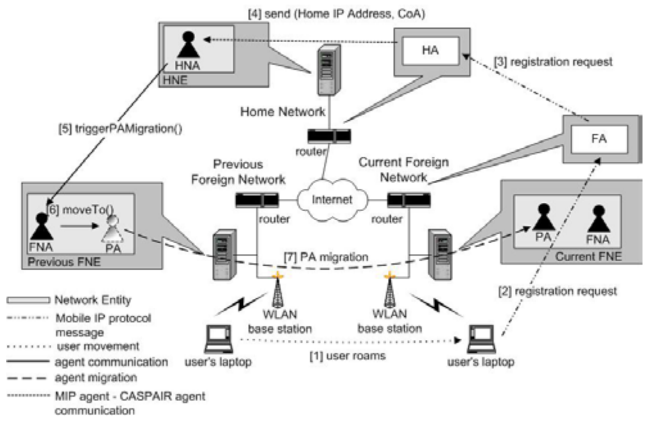

Figure 4 Handoff triggered PA migration.

A solution to this problem is to uncover MIP protocol information to the CASPAIR system. When a Mobile Node (MN) roams to another network and obtains a new care-of address (CoA), then, it should register the CoA with its Home Agent (HA). In case Mobile IPv4 is used, the registration is realized by a Registration Request message, which is sent by the MN to the HA, either directly or through the FA. In case Mobile IPv6 is used, the registration is realized by a Binding Update message, which is sent by the MN directly to the HA. In CASPAIR, after a successful registration, the HA sends the terminal home IP address and the CoA, contained in the registration message, to the network's HNA. Next, the HNA queries the network list in order to find a subnet prefix that matches the subnet prefix of the received CoA. In this way, the HNA obtains the location where the PA should migrate in order to follow the user. By using the terminal home IP address, the HNA queries the home user lists and obtains the current location of the user's PA. If the PA resides at the HNE, then, the HNA instructs its migration directly. However, if the PA resides in any other FNE, then, the HNA delegates the migration instruction to the corresponding FNA, through a *triggerPAMigration()* call. The mechanism that was just described is called *handoff triggered PA migration*. Figure 4 illustrates this mechanism in the case MIPv4 is used. The terminal roams between two foreign networks and the registration of the CoA is realized through the FA.

D. Service Activation

Figure 5 illustrates the activation of a service independently of its type. First, the user issues a service request through the CSMGUI running on his terminal. The request is received by the TA, which in turn calls the *activateService()* method on the PA (1). This method call provides the PA with the requested service id. Next, the PA produces the service specific profile, and calls the *activateService()* method on the SPA that provides the requested service (2). Among the method call arguments are: the service id, the service specific profile, and the currently available values of the service relevant context components. The SPA searches for the implementation of the requested service, and, then, creates a corresponding SA that handles service execution (3). As soon as the SA is created at the SPE, it uses the values of the service relevant context components in order to extract an initial user-defined service configuration from the service specific profile. Next, the SA sends the service GUI (SGUI) to the TA, so as to enable the user-service interaction, as well as the presentation of the service's results (4). The SGUI is adapted according to the directives of the currently holding configuration. The SA is continuously adapting the service behavior according to the values of the service context components that it receives by the user's PA.

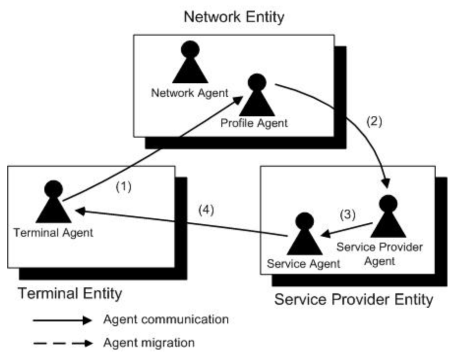

Figure 5 Service activation procedure.

E. Service Session Management

The activation of a service implies that a service session is initiated. The state information that specifies a service session is handled by the corresponding SA. All service sessions initiated by a specific user are managed by its PA.

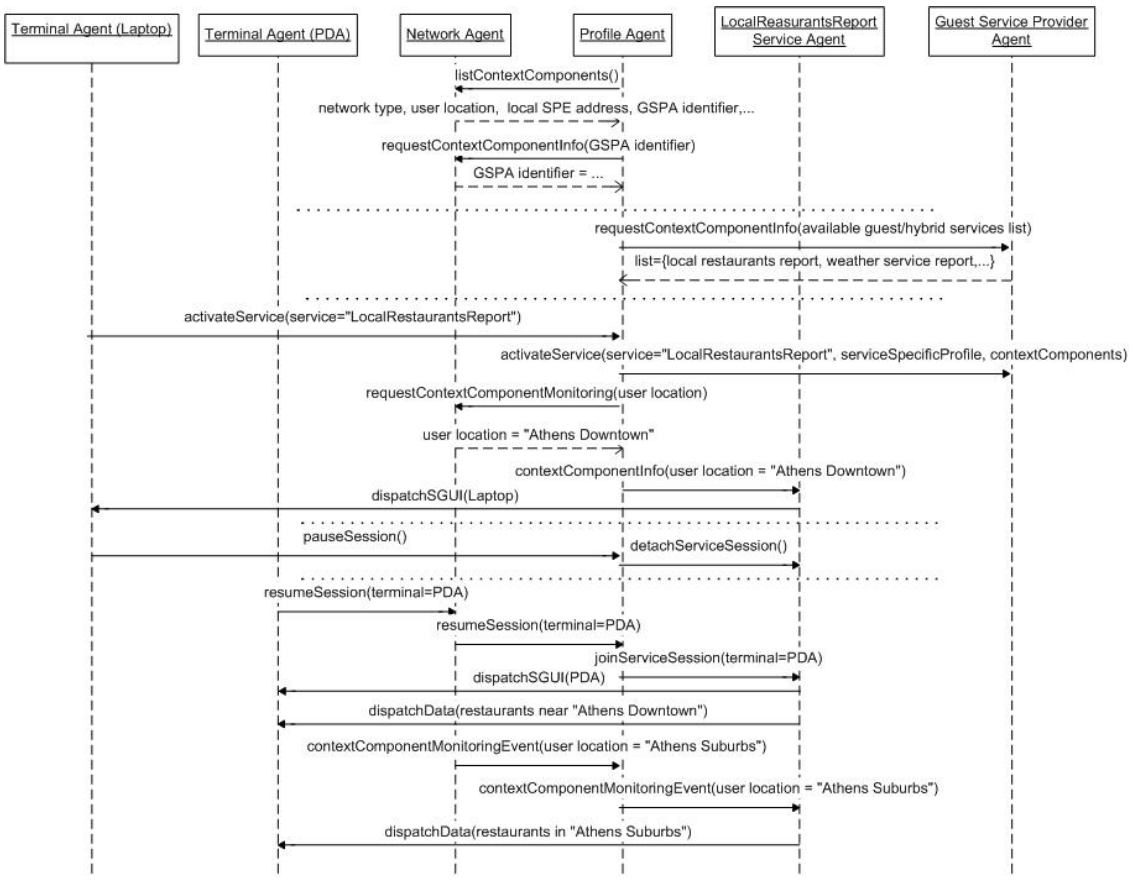
Terminal Agent (Laptop)
Terminal Agent (PDA)
Network Agent
Profile Agent
LocalReasurantsReport Service Agent
Guest Service Provider Agent
listContextComponents()
network type, user location, local SPE address, GSPA identifier,...
requestContextComponentInfo(GSPA identifier)
GSPA identifier = ...
requestContextComponentInfo(available guest/hybrid services list)
list={local restaurants report, weather service report,...}
activateService(service="LocalRestaurantsReport")
activateService(service="LocalRestaurantsReport", serviceSpecificProfile, contextComponents)
requestContextComponentMonitoring(user location)
user location = "Athens Downtown"
contextComponentInfo(user location = "Athens Downtown")
dispatchSGUI(Laptop)
pauseSession()
detachServiceSession()
resumeSession(terminal=PDA)
resumeSession(terminal=PDA)
joinServiceSession(terminal=PDA)
dispatchSGUI(PDA)
dispatchData(restaurants near "Athens Downtown")
contextComponentMonitoringEvent(user location = "Athens Suburbs")
contextComponentMonitoringEvent(user location = "Athens Suburbs")
dispatchData(restaurants in "Athens Suburbs")

Figure 6 A service scenario that illustrates context information communication as well as session management in CASPAIR.

The interface between the PA and the SAs defines two service session operations: the *service session pause*, and the *service session detachment*. Service pause implies that service execution is interrupted, and, then, resumed at a future point in time. After resumption, the service execution continues from the point it was stopped. Service session detachment, on the other hand, implies that the service is detached from the user session, and may run independently. In this way, the service may continue to execute even when the user has paused his CASPAIR session. The results produced are stored in the SA in order to be delivered to the user, as soon as he resumes his session, and the association between the user terminal and the service is restored. On service session resumption, any context information that concerns the currently used terminal is updated in the corresponding SA. In this way, the static binding between the services and the currently used terminal is removed and the user is given the ability to transfer a service session between terminals.

F. Context Information Exchange

Agents have two roles as far as context information exchange is concerned. They may provide context information about the entities they represent, in which case they are called context providers. Moreover, they may request context information from providers in order to exploit it, in which case they are called context consumers. Context information can be provided either in a synchronous, or in an event-driven way. A consumer uses the synchronous way if he wishes to obtain immediately the value of one or more context components observed by a provider. In the event-driven way, a consumer receives asynchronously the state of a context component only if it has been updated. The context provider publishes the set of update events for the different context components that it observes, and receives subscriptions by context consumers interested in notifications for such events.

G. Sample Service Scenario

A service execution scenario is presented in order to showcase the context information flow between the agents, the management of the service session, and the adaptation of service logic to the changing context. The sample scenario, which is illustrated in Figure 6, concerns a hybrid service that provides a list of restaurants, which are located nearby the current user location and match a set of user preferences. The user uses a laptop with a GPRS network connection.

The user's PA requests the list of context components observed by the local NA by means of a *listContextComponents()* call. The received list contains several context components including the network type, the current user location, the identifier of the local guest SPA (GSPA), and the address of the local SPE, where the GSPA resides. In this example, we assume that the currently used access network provides location tracking. The PA issues a *requestContextComponentInfo()* to the NA in order to obtain the current value of the local GSPA identifier. With a similar request, the PA obtains the address of the local SPE. Then, the PA issues a context information request to the GSPA in order to obtain the current list of available guest and hybrid services. The aforementioned list is presented to the user. Following, the user activates the local restaurants report service. The PA identifies that one of the context components that constitute the service context is the user's current location. Moreover, the PA knows that the NA can monitor the user's current location. So, the PA subscribes to the NA for update events, concerning the user location, through a *requestContextComponentMonitoring()* call. Through this call the PA also obtains the current value of the user location, which it forwards to the SA. The SA sends the appropriate GUI, for the presentation of the service, to the TA of the currently used terminal. Following, the SA executes the service logic in order to obtain the list of restaurants near the current user location.

When the user pauses his session, the SA is not instructed to pause its service session. On the contrary, it is instructed to continue the service execution by means of a *detachServiceSession()* method call. In this way, the service keeps collecting and storing information in order to present it to the user when he resumes his session. Later on, the user resumes his session by using a PDA. The update of the terminal context is sent to the SA through the *joinServiceSession()* method call, which is used to associate the SA with the new TA. Following, the SA sends the new GUI that is appropriate for the PDA, and, then, all the information that was collected by the service, to the new TA.

When the user moves to a new location, this fact triggers a context component update event, which is sent by the NA to the PA through a notification. The notification that carries the new user location is forwarded to the SA, which exploits the received information in order to adapt the service execution. Notifications are sent through the *contextComponentMonitoringEvent()* call. Finally, the service collects information about restaurants near the new user location and sends it back to the TA in order to be presented to the user.

V. CONCLUSIONS

In All-IP network architectures, nomadic users are moving between different computing and communication environments. This fact calls for the introduction of context-awareness in the provision of personalized services. In this paper, an architecture that supports context-aware service provisioning in All-IP networks has been proposed and analyzed. This architecture provides continuous context acquisition, in every environment visited by nomadic users, and facilitates the contextualization of the services used. Context acquisition is carried out by the Profile Agent, which follows the nomadic user, and gathers context information by the agents that represent the currently visited environment. Context information, which is communicated either in a synchronous or an event-driven way, is eventually fed to the agents that manage the activated services along with any relevant preferences defined in the user profile. The services exploit the received information and adapt their behavior or the content they process. The mobility management of the Profile Agent is carried out by the cooperation of the Network Agents of the visited networking environments, as well as their interaction with the Mobile IP protocol whenever that is necessary.

REFERENCES

[1] L. Morand and S. Tessier, "Global mobility approach with Mobile IP in “All IP” Networks", In Proceedings of the 2002 IEEE International Conference on Communications, 2002, pp. 2075-2079.

[2] C. Perkins, "IP Mobility Support for IPv4", IETF RFC 3344, 2002.

[3] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6", IETF Internet Draft, June 2003.

[4] L. Kleinrock, "Nomadic Computing - An Opportunity", ACM SIGCOMM Computer Communication Review, vol. 25, 1995, pp. 36-40.

[5] A.K. Dey and G.D. Abowd, "Towards a Better Understanding of Context and Context-Awareness", Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999.

[6] H.-G. Hegering, A. Küpper, C. Linnhoff-Popien, and H. Reiser, "Management Challenges of Context-Aware Services in Ubiquitous Environments", In Proceedings of the 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2003), 2003, pp. 246-259.

[7] D.B. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents", Communications of the ACM, vol. 42, 1999, pp. 88-89.

[8] G. P. Picco, "Mobile Agents: An Introduction", Journal of Microprocessors and Microsystems, vol. 25, 2001, pp. 65-74.

[9] D. Mandato, E. Kovacs, F. Hohl, and H. Amir Alikhani, "CAMP: A Context-Aware Mobile Portal", IEEE Communications Magazine, 2002, pp. 90-97.

[10] P. Farjami, C. Gorg, and F. Bell, "Advanced service provisioning based on mobile agents", Computer Communications, vol. 23, 2000, pp. 754-760.

[11] K. Raatikainen, "Middleware for Future Mobile Networks", in Proceedings of the IEEE International Conference on 3G Wireless and Beyond, 2001, pp. 722-727.