Towards efficient support of context-awareness in mobile systems

Nikos Houssos, Athanassia Alonistioti, Lazaros Merakos
Communication Networks Laboratory, Department of Informatics & Telecommunications,
University of Athens, 157 84, Athens, Greece,
email:{nhoussos, nancy, merakos}@di.uoa.gr

Abstract: Future mobile communication systems beyond 3rd generation (3G) are expected to signal the transition from inflexible, vertical infrastructures to reconfigurable, horizontally integrated networks that will incorporate a vast, ubiquitous space of devices and radio technologies and will be able to deliver to end-users a previously unimaginable range of valuable services. A fundamental characteristic for systems and services that operate in such a pervasive, heterogeneous environment is *context-awareness* that enables dynamic adaptation according to rapidly changing situational parameters. Challenging technical issues, however, need to be solved on the route to realization of this feature, including the optimal representation, placement, replication and management of the corresponding context information. This paper discusses some of these problems and introduces data management schemes that facilitate the seamless incorporation of re-usable situation-awareness functionality into a middleware platform for beyond 3G service delivery and management.

I. INTRODUCTION

The evolution of mobile networks to 3rd generation (3G) and beyond is expected to bring about substantial changes to the way telecommunication services are being created, delivered and consumed. Technological advances in various fields (e.g., wireless communications, embedded systems, microelectronics, distributed systems, middleware) have enabled the research community to start shaping a vision of future mobile systems, commonly termed as "4G" or "beyond 3G" [1][2], comprising ubiquitous, heterogeneous infrastructures that will support the delivery of a plethora of functionality-rich applications, offering end-users valuable assistance with all kinds of their everyday activities. In such a dynamic environment, services and systems should have the capability of being aware of the constantly changing context. In this paper, we introduce context management schemes that have been employed to support the successful incorporation of context-awareness features in a middleware platform for beyond 3G service management and provision [4].

The rest of this document is organized as follows: In Section 2 we present basic concepts regarding context-awareness and discuss the significance of the latter in next generation mobile service provision, as well as the fundamental requirements of a context management system. In Section 3, we present the environment where the proposed context management mechanism has been integrated, namely a software platform for provision of services over 3G mobile networks. Furthermore, we elaborate on the introduced schemes, including elaboration on the design choices made and their benefits and presentation of the interactions through which context-awareness functionality is accomplished in our system. The last section of this paper is dedicated to summary and conclusions.

II. CONTEXT AWARENESS – CONCEPTS AND REQUIREMENTS

A. Context awareness – Basic concepts

Context can be defined as any information that can be used to characterize the situation of the entity [5]. This information includes data about the entity itself (the entity's *profile*, which is typically a small subset of the context data in its entirety). Context awareness typically refers to the capability of "sensing" and exploiting context information. Context-awareness is closely related and bound with adaptability, which can be defined as the ease with which an entity can be dynamically modified. The dynamic modification (adaptation), which can be considered as a transition to another state of the entity and could include an alteration in the entity's behavior, is performed according to the current context. In this paper, while recognizing their interdependence, we clearly distinguish between the notions of context-awareness and adaptability; the former is restricted to *obtaining* and possibly *interpreting* situational data, while the latter concerns the ability to adapt (most probably based on context information). In general, we consider context collection as one of the tasks comprising any adaptation procedure [7], and therefore context-awareness can be considered a prerequisite of adaptability.

Considering the applicability of the above definitions in the field of mobile service provision, one could identify several types of entities that can be subject to adaptation and thus benefit from context-awareness. These include end-user services, protocols at various layers of the networking protocol stacks, control plane mechanisms (e.g., handover) as well as content that is provided to the mobile user.

B. Requirements and issues in context management

Situation-awareness requires context management mechanisms that can make environmental data available to systems and services that are able to utilize it in a useful way. In the present section we identify the basic requirements from such a system and certain issues that important in its design. These issues, which we try to address in our system (see Section 3), are the following:

- Level of abstraction. An extremely tedious task in the development of situation-conscious applications is the collection of context data [5] from a large variety of sources, which may support diverse interfaces and communication mechanisms, provide low-level, raw information that is not useful without pre-processing and belong to different administrative domains
- Consistency. Context data should be consistent and up-to-date; otherwise inappropriate effects to context-sensitive applications may be incurred.
- Efficiency. Context management should be efficient, so that the corresponding functionality does not heavily burden network and system resources (e.g., link capacity, memory).
- Context meta-data availability. Attributes of situational information should be available to the system, in order that improper utilization of this information is avoided [6] [16]. For example, location information may not be valid for an application if a certain degree of accuracy is not guaranteed.
- Modularity and independence from other components. Context management logic should be as far as possible decoupled from other functions (e.g., context-based adaptation intelligence [7]), so that re-usability, scalability and extensibility are facilitated.
- Generality. A context management system should be generic and dynamically extensible to handle a wide range of context parameters with diverse encodings.

III. CONTEXT MANAGEMENT MECHANISMS FOR BEYOND 3G SYSTEMS

A. Putting the context management system in context: Service provision platform overview

The proposed context management scheme has been incorporated in a distributed software platform (called *RCSPP, Reconfiguration Control and Service Provision Platform*) for the provision and management of value-added services over mobile networks in 3G and beyond [4]. It is worth noting that an earlier version of this platform has been developed in the frame of project IST-10206 MOBIVAS. The platform aims to address major issues regarding the deployment and management of services offered to users of next generation mobile networks. These applications are typically provided by third-party software vendors, commonly termed Value-Added Service Providers (VASPs) and can be delivered over different types of networks in distinct administrative domains. A key innovation is the RCSPP provides a single point of contact (a "portal" in a certain sense) for mobile users, VASPs and network operators

The platform can be administered by a new actor in the mobile provision value chain, the *service platform operator* [4].

The architecture of the platform is depicted in Figure 1.

The *Reconfiguration Control and Service Provision Manager (RCSPM)* is the central platform component in that it co-ordinates the entire service provision and management process. The RCSPM is the place where the complete, authoritative copy of the available context information pertaining to each user session with the platform is maintained. It includes modules that undertake on-line service deployment by VASPs, network reconfiguration, maintenance of service- and user-related data in suitable databases and repositories, as well as customized service discovery, downloading and adaptation.

The *Charging, Accounting and Billing (CAB)* system [9] is responsible for producing a single user bill for service access and apportioning the resulting revenue between the involved business players.

The *End User Terminal Platform (EUT)* [10] includes functionality such as service downloading management, GUI clients for service discovery and selection, capturing of event notifications as well as service execution management. The EUT contains a context repository, which is a subset of the primary copy residing within the RCSPM. The EUT is able to track context information that is communicated to the RCSPM.

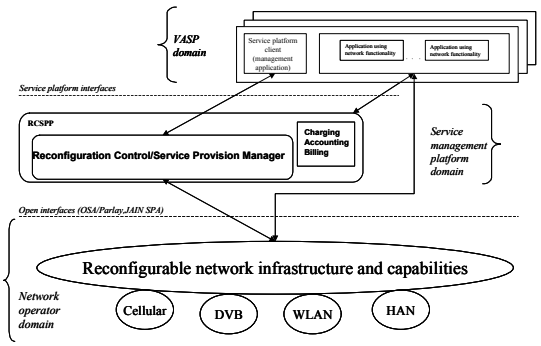

Figure 1. Architecture for flexible service provision in 3G and beyond networks

B. Context management in the RCSPP

1) Architectural overview

Context management in the RCSPP is based on the architectural components depicted in Figure 2. It is worth noting that the figure does not provide a complete picture of the RCSPM architecture; a number of RCSPM components, interfaces and repositories, which are not directly related to context management operations, are not depicted. The main elements of the architecture are the following:

1. The context sources, which includes every possible entity from which context data can be retrieved, including low-level sources (e.g., sensors) as well as higher-level interfaces like OSA/Parlay gateways providing information

concerning terminal capabilities as well as user location and presence

2. Adaptable applications, whose context-awareness features are supported by the RCSPP. Notably, the case of situation-aware services that do not utilize the RCSPP for context management is not precluded, e.g., direct interaction with the context sources is perfectly valid. This has been taken into account in our design in recognition of the fact that occasionally ad-hoc context management performed from within the application could be preferable for some reason (e.g., a service needs limited context information, which can be easily acquired with proprietary methods).
3. The EUT software residing in the mobile terminal.
4. A subset of the components and interfaces of the RCSPM, which are described below:

   The ***User Interaction Management Module (UIMM)*** is responsible for providing the user with a highly personalizable, context-aware mobile portal. It manages user sessions with the RCSPP and co-ordinates user-related operations like service discovery, selection, adaptation and downloading as well as user profile management (user data is persistently stored in an appropriate database). The UIMM is the place where dynamically updatable user session information is maintained. This information is accessible to clients (e.g., applications) through generic open APIs.

   The ***VAS Registrar Module (VASREGM***) is responsible for interacting with 3rd party service providers. Through the VASREGM the platform operator provides VASPs with a way to automatically deploy their services. The VASP compiles a profile, encoded in XML, of service attributes. Based on these attributes, the VASREGM co-ordinates service deployment, including various actions like reconfiguration of the underlying infrastructure and uploading of service components to the RCSPM. The service provider is able to manage (add/delete/update) its services via a convenient web interface.

   The ***Network Registrar Module (NetREGM***) enables network operators to register their network with the platform. The network providers feed into the NetREGM a descriptor of their infrastructure, according to a common, extensible network information model. Based on this descriptor attributes, the NetREGM triggers all the required (automatically carried out) actions like storage of appropriate data in network profile database as well as instant deployment of the available VAS that are compatible with the newly registered network(s). In the same way as the VASREGM, NetREGM functionality is remotely accessible to network operators via a convenient web interface.

   The ***Reconfiguration Manager (RCM)*** undertakes network, platform and service reconfigurability. The RCM is responsible for executing the appropriate reconfiguration actions on the underlying network during VAS management procedures (registration/de-registration/update), triggered by the VASP. The RCM also comprises a generic adaptation module [7] that is used for supporting adaptation through functions like intelligent profile matching.

   The ***Packaging and Downloading Module (PDM)*** is responsible for dynamically creating, in a context-aware manner, a single downloadable application bundle.

   The RCSPM also includes database managers that provide interoperable access to the persistent service, user and network profile repositories hosted by the platform. In particular, the maintenance and management of network profile information is vital to situation-awareness support, since it enables the instant availability of data regarding the constantly changing infrastructure that is close to the user as he/she moves within beyond 3G, ubiquitous/ambient computing environments.

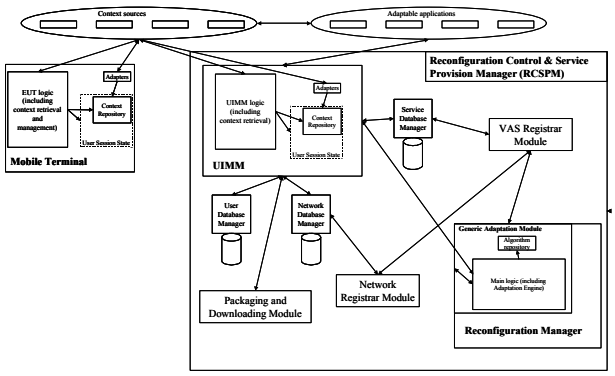

Figure 2. Context management architecture.

C. *Context management design – critical choices*

a) *Context definition, placement and update*

A fundamental question that a context management system designer faces is to what type of entity (e.g., person, application, terminal?) context information is bound. Related issues concern:

- The location and distribution of context data, including synchronization techniques in case replication is employed.
- The retrieval of context information from its original sources.

In the rest of this section we describe how we addressed the above issues in our mechanism.

Regarding the first question, we chose to relate context information not the user, an application or their combination, but to a *user session* with the RCSPP. Such a session begins with security procedures (e.g., mutual authentication of user and platform provider) and includes service discovery, selection, downloading and execution by the mobile user. All the above procedures are customized to context. It is possible for a user to concurrently execute multiple services from one or more terminals and over different access networks.

We did not choose to bind context solely with applications, since in that case the characteristics of the user situation, which are

usually valuable for adaptation, would be ignored. A combination of user and application (an obviously good choice when the context management logic resides within the application) would not allow the re-use of context management features by multiple services and would create heavy overlapping and duplication of context data. Finally, the end-user himself is a serious candidate; however, multiple user sessions from different contexts like different terminals, locations (note that non-interactive services can continue to be executed without the user presence) and connectivity networks are possible.

As far as the location, distribution and update of context data is concerned, our approach was to maintain only a single complete and authoritative copy of the environmental information concerning each user session; this copy is kept in the internal of the RCSPM and in particular within the UIMM. The context information maintained by the UIMM is accessible to RCSPP components and external services through generic, open APIs. It is worth noting that this is *not* persistent storage; user session context data is assembled as a unit from the context sources (e.g., databases, sensors, network infrastructures) only for the lifetime of a user session (persistence can of course be useful for fault tolerance and recovery purposes, e.g., through periodic "dumping" or the UIMM non-persistent context repository but this feature has not yet been implemented in our prototype). The choice to place the authoritative copy of context data on the server-side (unlike other approaches [11] that are oriented towards terminal devices as hosts of the "master" copy of environmental information) has been made for the following reasons:

- The majority of context sources are more efficiently accessible from the RCSPM than by the terminal. Notably, a large part of context data is immediately available inside the RCSPM (user, service and network infrastructure profiles), while other information (e.g., terminal hardware capabilities) is easily retrievable from fixed network servers. Transferring and keeping all this data to the terminal would be largely inefficient.
- Placing the context repository on the server-side makes it available (via open interfaces) to server-side parts of applications, which usually contain the core of the service business logic. If context was kept in the terminal, this open access would be impractical due to the limited capacity of the radio link.
- In contrast with the typically restricted capabilities of mobile terminals and the limitations of the wireless links, on the server-side there are typically plenty of computational and communication resources for easily accomplishing the tasks of retrieving and processing (e.g., dynamically enrich through inference techniques) situational data.

The UIMM is responsible for retrieving context from all possible sources. The latter mainly include persistent repositories (like the databases maintained within the RCSPM), as well as (registered with the RCSPP) network infrastructures accessible through open interfaces (e.g., OSA/Parlay, JAIN SPA). The UIMM retrieves data from the latter not only in a "pull" approach, but also is able to receive event notifications form the corresponding open API gateways. The contact information of the latter (e.g., network addresses, access terms of use) is included in the network profile information provided by the network operator at the time of registration of the corresponding network with the RCSPP.

An interesting issue relates to the fact the UIMM constantly attempts to collect every piece of user session context information that can be possibly retrieved. This "greedy" approach results to faster response times in context retrieval requests made towards the UIMM and simplifies management of situational data at the UIMM, coming at the price of an one-off overhead at the beginning of the session when the bulk of available context is retrieved. Alternative schemes and their implications are subject to ongoing work.

In certain cases, the retrieval of some parts of the user session context data is feasible only from the EUT (or more efficiently performed that way). In such a case, the EUT retrieves the data, keeps it in its local context repository (which has the same form as the one kept in the UIMM, but contains only a subset of the information) and updates the UIMM with it. If the update is valid, the UIMM commits it and acknowledges it to the EUT, which keeps it locally; in case that the UIMM has a more accurate or more recent copy of this information (e.g., originating from another context source), the EUT discards it from its local cache and optionally retrieves from the RCSPM the up-to-date copy of this specific parameter.

b) Context representation

In our implementation (that was performed in Java), we defined generic, data type-independent containers for profiles. The class diagram of Figure 3 depicts the common internal representation of profiles that we have used. All the context information reaching the UIMM is transformed to this form. For that reason, appropriate adapters [12] can be dynamically "plugged-in" our system. In our current implementation we have incorporated adapters for XML-encoded RDF [13] (using the open source DELI and JENA tools [14][15]), plain XML as well as relational database data. The Composite design pattern [12] has been applied for profile representation. We have defined the classes ProfileAttribute (for single attributes) and Profile (for compound profiles). Both classes inherit from the ProfileElement abstract base class, while Profile objects also aggregate ProfileElement objects. ProfileElement objects are identified their name and fully qualified type (e.g., gr.uoa.di.cnl.TerminalCapabilities) and essentially act as generic containers of arbitrary data structures that can be retrieved and iterated (in case of compound profiles) using generic logic. Thus, introducing new types of profiles and profile parameters does not require the creation of new classes (e.g., subclassing) and can be performed at runtime, without any changes in the profile processing code of the adaptation engine.

Each ProfileElement contains a single reserved attribute called "metadata". This is a generic container (thus, it is itself a ProfileElement) that integrates all the available metadata related to a specific context profile. It is worth stressing that this information does not pertain to the entity represented by the profile, but to the quality of the profile data [16]. Apparently, the set of applicable types of metadata depends on the profile type may be bound to different types of metadata elements (e.g.,

attributes like estimated accuracy and various parameters that concern when, where and by whom this profile. However, we have identified certain parameters (e.g., time of profile element retrieval, administrative domain from which the profile element was acquired) that apply to a wide range of context data and have been employed for all the context information types that we used in our prototype.

An important capability that is made possible by the design of Figure 3 is the processing of context information in the course of making intelligent context-aware adaptation decisions by algorithms that are specific to the application/service that will be affected by this decision. Note that different services may require different algorithms for matching service requirements with context parameters. As a simple example one could consider a context parameter that is expressed in terms of dimension (e.g., terminal screen resolution). Two services could have the same value in their profile as a dimension requirement, but the algorithm for matching it with the corresponding context value could differ (e.g., one algorithm would require the currently supported screen resolution of the terminal to be just greater than or equal to the value in the service profile, while another one could additionally require that the width/height ratio would be equal to a specific quotient). Thus, in the service profile, each context parameter is annotated with a descriptor of the corresponding algorithm. In the adaptation engine internal representation of the service profiles these descriptors have the form of *ComparatorDescriptor* objects that correspond to the *Comparator* (*Matcher* or *Adaptor*) objects that encapsulate the appropriate algorithm. An important feature is that these algorithms can be specified and loaded at run-time by third-parties (e.g., VASPs) that are this way able to tailor the context-awareness decisions behavior of the system to their needs (e.g., to the requirements of a particular service), without modifying the system itself. More details on these mechanisms are beyond the scope of the present paper and can be found in [7].

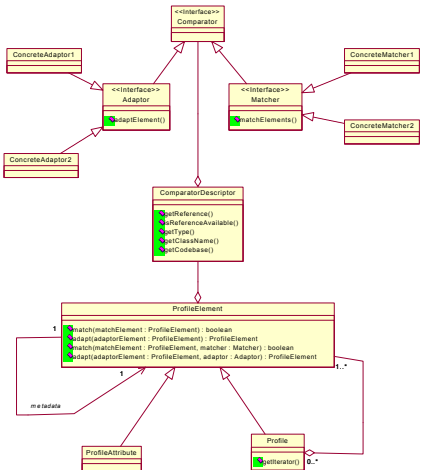

Figure 3. Context profile representation.

IV. SUMMARY – CONCLUSIONS

Support for context-aware service provision in the envisioned heterogeneous and dynamic beyond 3G environments is a very challenging task. This paper has proposed context management mechanisms that constitute a basis for the incorporation of situation-aware functionality in mobile systems and applications. Key issues for the design of our context retrieval and processing system have been identified, such as architectural choices, generic profile representations as well as flexible object-oriented design that enables run-time extensibility. The design and implementation of the proposed scheme is greatly facilitated by the adoption of the service provision model that is based on a mediating middleware platform for service delivery and management [4].

V. REFERENCES

[1] A. Jamalipour, S. Tekinay (eds.), "Fourth generation wireless networks and interconnecting standards", Special Issue of IEEE Personal Communications Magazine, October 2001.

[2] J. Pereira, "Beyond third generation", Wireless Personal Mobile Communications (WPMC) 1999, September 22, 1999, Amsterdam, The Netherlands.

[3] M. Dillinger, N. Alonistioti, K. Madani, "Software Defined Radio: Architectures, Systems and Functions", John Wiley & Sons, Jun 2003.

[4] A. Alonistioti, N. Houssos, "The need for network reconfigurability management", in [3].

[5] A K. Dey, "Providing Architectural Support for Building Context-Aware Applications", PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.

[6] E. Mohyeldin, M. Fahrmair, Christian Salzmann, "Communication Profiles", in [3].

[7] N. Houssos, S. Pantazis, A. Alonistioti, "Generic adaptation mechanism for the support of context-aware service provision in 3G networks", 4th IEEE MWCN 2002, Stockholm, Sweden, 9-11 September 2002.

[8] UMTS Forum Report No. 9, http://www.umts-forum.org/.

[9] M. Koutsopoulou, A. Kaloxylos, A. Alonistioti, "Charging, Accounting and Billing as a Sophisticated and Reconfigurable Discrete Service for next Generation Mobile Networks", Fall VTC2002, Vancouver, Canada, September 2002.

[10] O. Fouial. K. A. Fadel, I. Demeure, "Adaptive Service Provision in Mobile Computing Environments", 4th IEEE MWCN 2002, Stockholm, Sweden, 9-11 September 2002.

[11] S. Riche, G. Brebner, "Storing and Accessing User Context", Mobile Data Management (MDM) 2003, LNCS 2574, pp.1-12, January 2003.

[12] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns: Elements of Reusable Object Oriented Software", Addison Wesley Longman, Inc., 1995.

[13] RDF: Resource Description Framework, http://www.w3.org/RDF/.

[14] M. Butler, "DELI: A DElivery context LIbrary for CC/PP and UAProf", HP Labs Technical Report, HPL-2001-260.

[15] The jena semantic web toolkit, http://www.hpl.hp.com/semweb/jena-top.html.

[16] K. Henricksen, J. Indulska, A. Rakotonirainy, "Modeling Context Information in Pervasive Computing Systems", Pervasive 2002, LNCS 2414, pp.167-180.