COSMOS: A Context-centric Access Control Middleware for Mobile Environments

Paolo Bellavista, Rebecca Montanari, Daniela Tibaldi

Dipartimento di Elettronica, Informatica e Sistemistica
Università di Bologna
Viale Risorgimento, 2 - 40136 Bologna - Italy
{pbellavista, rmontanari, dtibaldi}@deis.unibo.it

Abstract. User/terminal mobility during service provisioning and high heterogeneity of wireless portable devices identify novel challenges for service delivery in ubiquitous pervasive environments. An emerging architecture solution in the wireless Internet is to have middleware components (mobile proxies) over the fixed network that follow the movements and act on behalf of the limited wireless clients. It is crucial that mobile proxies have full visibility of their context, i.e., the set of available and relevant resources depending on access control rules, client location, user preferences, privacy requirements, terminal characteristics, and current state of hosting environments. The paper presents the design and implementation of a context-centric access control middleware, called COSMOS, for the wireless Internet. COSMOS dynamically determines the contexts of mobile proxies, and effectively rules the access to them, by taking into account different types of metadata (user profiles and system/user-level authorization policies), expressed at a high level of abstraction and cleanly separated from the service logic. The paper also shows how COSMOS facilitates the development of articulated access control strategies in the case study of a context-dependent movie-info service deployed over IEEE 802.11 network localities.

1 Introduction

Telecommunication systems and the Internet are converging towards an integrated pervasive scenario that permits the development and the access to services in highly dynamic mobility environments. Mobile users can connect to the Internet from ubiquitous access points, possibly by preserving their personal preferences and the state of their ongoing sessions. Mobile devices can either nomadically connect/disconnect to/from different points of attachment or continuously maintain connectivity during their roaming [1]. In particular, the recent proliferation of heterogeneous portable devices and of different technologies for wireless connectivity in home/office environments suggests not only extending to mobile users/devices the access to traditional Internet services, designed and implemented for the fixed network infrastructure, but also to develop new classes of services that can provide results depending on the relative position of clients and on the consequent resource visibility. For instance, a city

guide assistance service should dynamically retrieve maps, audio descriptions, and detailed textual information about the buildings, restaurants and theatres close to the current user location. Service results, possibly retrieved from traditional Web servers, should be filtered and downscaled at provisioning time depending on the hardware/software limitations of client access devices.

The complexity of the above scenario calls for novel middleware solutions for facilitating service development and for supporting service delivery to wireless devices in mobility environments. We claim that these novel middlewares should exhibit the non-traditional property of context-awareness. Different definitions of context have been recently proposed: for example, Schilit and Theimer identify context as the identities and locations of nearby users and objects, and their modifications during service sessions [2], while Pascoe uses the term context to indicate the subset of physical and logical states of interest for a specific subject [3]. In the following, we adopt a large definition of context as the collection of any information useful to characterize the runtime situation of a user during her service session [4]. In addition, the middleware should enable the dynamic installation/migration of client-side middleware/service components and their seamless discarding when no longer needed.

By focusing on a commonly addressed networking scenario, in the following we will use the term *wireless Internet* to indicate the integrated networks where wireless solutions extend the accessibility of the fixed Internet via service access points working as bridges between fixed hosts and wireless devices, such as in the case of IEEE 802.11 access points providing Wi-Fi equipped laptops with connectivity to a traditionally wired local area network.

Few research activities have recently started to investigate middleware solutions for the wireless Internet: an emerging guideline is to have active middleware components that are deployed at service provision time to act as user/device proxies over the fixed network and to carry on the needed service configuration, tailoring and management. The relevance of implementing middleware components as mobile proxies that follow the client movements and execute in their same network locality starts to be recognized [5, 6]. We claim the suitability of designing and implementing mobile proxies for the wireless Internet in terms of Mobile Agents (MAs). MA-based proxies can carry on service requests autonomously even in case of temporary device disconnection, can migrate among different network localities by maintaining the session state, and can exploit the full context visibility typical of the MA programming paradigm to support context-based service configuration and tailoring [4].

However, the deployment of MA-based proxies also raises novel and challenging security concerns [7, 8, 9]. On the one hand, the possible injection of malicious proxies can compromise the security of the wireless Internet fixed nodes similarly to the case of viruses and worms. On the other hand, malicious nodes may try to disclose the private information carried by the hosted proxies and to tamper with the MA code and state. In addition, adopting a middleware solution based on mobile proxies requires establishing a secure user-proxy trust relationship to ensure that proxies actually operate according to the user requirements [10].

The paper focuses on the novel access control issues coming from the adoption of MA-based proxies for the support of context-dependent service provisioning over the wireless Internet. Several practical techniques have been already proposed to control

and confine the interactions between MAs and hosting execution environments. Type-safe languages permit to determine whether incoming MAs respect safety properties, such as address space confinement. Sandboxing techniques can be used to rigidly limit the resource visibility and access scope of MAs while executing and have evolved to propose more advanced access control solutions [11]. However, these solutions are not flexible enough for the addressed highly dynamic pervasive environments. Frequent and unexpected proxy mobility, to follow the recurrent movements of wireless devices, significantly increases the complexity of access control decisions. Hosting environments interact with MA-based proxies they are not familiar with and, most important, whose identity may be un-informative or not sufficiently trustworthy. It is unlikely for hosting environments to know in advance the identities/roles of all subjects that will request the access to their managed resources/services, thus making inappropriate traditional access control policies based on subject identity/role. In other words, we claim that context-dependent service provisioning in the wireless Internet requires a deep paradigm shift from subject-centric access control models to context-centric ones: access control policies should also depend on the dynamic evaluation of the applicable context.

The paper presents the design and implementation of a highly dynamic and flexible security middleware, called COSMOS (**CO**ntext-aware **S**ecurity Middleware for **MO**bile Agent **S**ystems), for context-centric access control in the wireless Internet. COSMOS dynamically determines the contexts of MA-based proxies, and effectively rules the access to them, by taking into account different types of metadata (user/device/resource profiles and system/user-level authorization policies), expressed at a high level of abstraction and cleanly separated from the service logic. COSMOS provides an integrated environment for both the specification of context-dependent access control policies and for their runtime enforcement. The proposed access control middleware integrates with the SOMA framework which offers the support facilities for user/terminal mobility and for the proxy-based discovery/binding of wireless devices to the needed resources in their contexts [4, 12].

The paper finally presents the case study of a context-dependent movie-info service prototype, built on top of COSMOS and deployed over a group of coordinated IEEE 802.11 localities, to evaluate the usability and effectiveness of the proposed middleware. The case study exemplifies how COSMOS supports articulated context-based access controls and shows how the different context evaluation strategies implemented allow the dynamic tuning of the middleware overhead depending on application-specific requirements.

2 COSMOS Security Framework

COSMOS is an access control middleware for securing agent-to-agent and agent-to-environment interactions in service provisioning scenarios with context awareness requirements. In particular, COSMOS focuses on three main peculiar aspects: flexible solutions for context-centric access control, active context view provisioning to middleware components, and privacy support in the propagation of user context information. COSMOS access control decisions depend on dynamic context attributes, such as resource state and availability, in addition to more traditional attributes, e.g., the

identity of the MA code implementer, or the identity/role of the principal on behalf of whom MAs are executing.

Another distinctive feature of COSMOS is to provide MAs entering a new locality with a controlled visibility of the directly accessible physical/logical resources and of the other MAs locally executing (*active context views*). Active context views contain resources that both MAs are willing to access and the COSMOS access control function have qualified as accessible. The provision of active context views to MAs has many benefits. MAs can exploit the visibility of available resources to adjust their behavior accordingly and to reduce the risk of undesired task failure during execution. Active context views can also help MAs to decide whether it is more profitable to stay in a locality than to move from it and to explore a new computing environment.

COSMOS addresses also the privacy issues that arise in context-aware service scenarios [13, 14]. In fact, context awareness requires computing devices that gather, collect and propagate up to the service level both user- and environment-specific information to take more informative service management strategies. However, the visibility of user-specific information, such as user location, could be exploited to infer user tasks or preferences, thus violating user privacy. COSMOS protects user privacy by enabling users to specify which personal context information they are willing to make public. User specifications guide and automate COSMOS access controls to user personal context information.

2.1 Security Model

To support context-centric access control, COSMOS allows system administrators and final users (on behalf of whom MAs act over the network) to specify their security requirements at a high level of abstraction in terms of metadata. Metadata are declarative rules that describe the attributes of users, devices and service components and the desired access control requirements.

A primary advantage of exploiting metadata is the possibility to separate security logic from security control. Metadata govern access control decisions, but are decoupled from the implementation of the system components in charge of enforcing access control accordingly to the metadata specifications. This favors the rapid prototyping of secure MA-based services, their runtime configuration and maintenance. By changing metadata to accommodate evolving security requirements, the behavior of a running agent system can be dynamically and rapidly varied without any intervention on agent and system code. In addition, developers do not need to manually insert calls to security checking code inside each resource that a host may want to protect from illegitimate agent usage. Metadata can also facilitate automated security reasoning: all basic elements involved in access control decisions can be easily extracted from declarative notations, analyzed and checked for conflicts. The relevance of metadata adoption to decouple management logic from mechanism implementation details has been recently recognized in network, systems and service management [15].

Figure 1 shows the different kinds of metadata supported in COSMOS, i.e., profiles and policies. *Profiles* provide the explicit description of user/device/resource characteristics; in the following, we focus only on user profiles, while details on how device and resource profiles affect the context determination can be found in [4]. In

particular, COSMOS decomposes the user profile in three sub-structures: user properties, desired view, and privacy requirements. User properties specify user characteristics relevant to qualifying the user to the system, such as subscribed services, user identity or role.

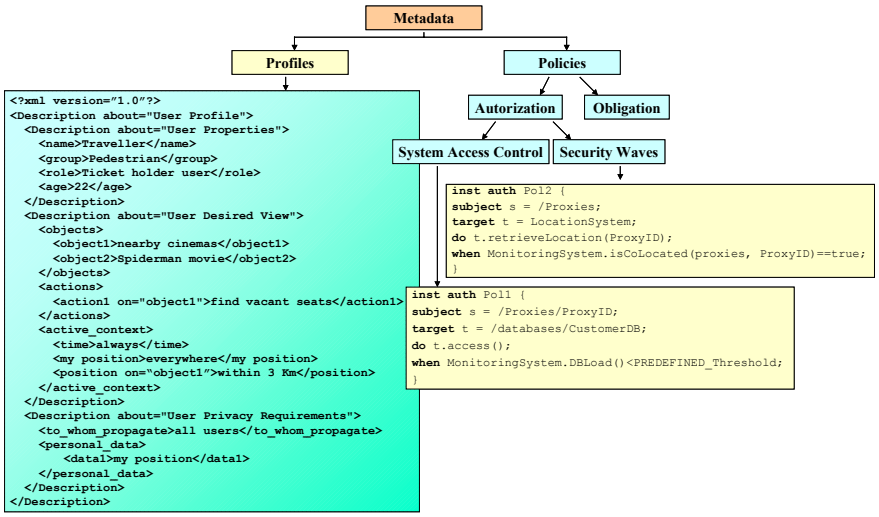

Figure 1. Metadata taxonomy and examples in COSMOS.

Desired views express user preferences about the desired context information visibility (such as resources, other co-located users, ...). For instance suppose that a user is willing to know if there are cinemas within 3 Km. As Figure 1 shows, his desired view is composed by the set of desired objects, i.e., cinemas (the `objects` tag), the set of actions that the user would like to perform on these objects, i.e., find vacant seats (the `actions` tag), and the context conditions in which the user desires to perform the specified actions, i.e., within 3 Km (the `active_context` tag).

Privacy requirements specify which personal information users are willing to propagate. This ensures a controlled disclosure of user-specific context information. In the profile example of Figure 1, the user allows the underlying system to propagate the information about her current position to all potentially interested users.

COSMOS adopts XML-based standard formats for profile representation to deal with heterogeneity of data representation over different architectures: standard XML for user profiles, the World Wide Web Consortium Composite Capability/Preference Profiles (CC/PP) for device profiles and the Resource Description Framework (RDF) for the interoperable description of resource components.

Policies are high-level directives regulating resource access control and defining choices in security management operations. COSMOS distinguishes between *authorization* and *obligation policies*. *A*uthorization policies rule access control by defining what a subject can or cannot do on specific target resources if certain context conditions are met. These conditions may depend on the runtime resource/system state and on the subject identity/role. Obligation policies allow to automate security manage-

ment tasks by specifying when the system must perform a specific management action on a set of target objects. For instance, MA system administrators can exploit obligation policies to block the execution of an agent when its CPU consumption is higher than a tolerated threshold. In the following, we focus only on authorization policies, essential for COSMOS context-centric access control model, whereas for details about obligation policies please refer to [16].

COSMOS distinguishes between system- and user-level authorization policies (*system access control* and *security waves* in Figure 1). The main difference is that the former ones are specified to protect system resources from illegitimate use by incoming MAs, the latter ones to reduce user privacy violations.

COSMOS exploits the Ponder language for expressing both types of authorization policies [17]. In Figure 1 `Pol1` is an example of system access control policy that allows an entering MA (the `subject` clause) to access the Customer database (the `target` clause) depending on the current database overload status (the `when` clause). Note that a subject can operate on target objects, by only invoking methods visible on the target interface and that the `when` clause allows to limit the applicability of authorization policies on the basis of context conditions. `Pol2` is an example of security wave allowing the propagation of the user location information to all interested users. Note that `Pol2` is the Ponder-based representation of the privacy requirements contained in the user profile shown in Figure 1. It is COSMOS that automatically generates `Pol2`: user privacy requirements are transformed into authorization policy rules that define which entities can have access to the user context information and under which circumstances. This transparent mapping relieves users from dealing with the details of the underlying system and facilitates the enforcement of user privacy requirements.

COSMOS exploits all the described metadata information to perform access control decisions and to determine the active context view to return to incoming MAs. In particular, as it stems from Figure 2, an active context view results from the intersection between a desired view and an allowed view. An allowed view contains the collection of local resources accessible to one MA depending on both system access control policies and security waves.

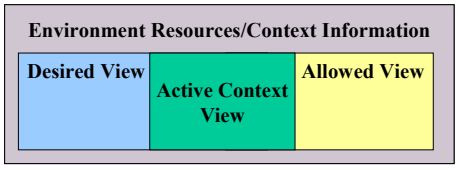

Figure 2. Active context view.

3 COSMOS Middleware

COSMOS has been built on top of the Java-based SOMA system that supports the accessibility of mobile users/terminals to both traditional Web and to new context-dependent services. SOMA is centered on the distributed deployment of active middleware proxies over the fixed network to support service provisioning to portable devices. SOMA provides any portable device with a companion middleware proxy

(*shadow* proxy) that autonomously acts on its behalf, possibly negotiates service tailoring to fit user/device characteristics and follows user/device movements among network localities. SOMA implements shadow proxies by exploiting the MA programming paradigm. In particular, SOMA provides proxies with execution environments, called places, that typically model nodes. Places can be grouped into domains that correspond to network localities, e.g., either Ethernet-based LANs or IEEE 802.11b-based wireless LANs. With a finer degree of detail, a shadow proxy is implemented by one SOMA agent running on a place in the domain where the portable device is currently located. SOMA domains can facilitate policy evaluation and enforcement for context-centric access control. In fact, the domain abstraction allows to define a well-specified management boundary: each domain holds references to the entities currently members of the domain (both MAs and resources) and to the applicable metadata. In particular, profiles and security waves are maintained in the SOMA directory service with global visibility and are accessible via the local domain directory component [6]; system access control policies are stored locally at the domains where they have to be enforced to increase management decentralization and local policy access.

At first proxy instantiation, COSMOS creates a secure trust relationship between the device and the proxy according to the protocol steps described in [18]. At runtime, proxy interactions with the hosting environments are protected by means of the COSMOS middleware facilities shown in Figure 3. The **Context-aware Security Manager** returns to proxies active context views on the basis of specified metadata, the **Authorization Enforcement Manager** performs context-centric access control decisions and the **Metadata Manager** enables users to specify the needed metadata. COSMOS facilities exploit SOMA lower-level functions for proxy identification, resource discovery, directory, context monitoring, and event registration/dispatching [6].

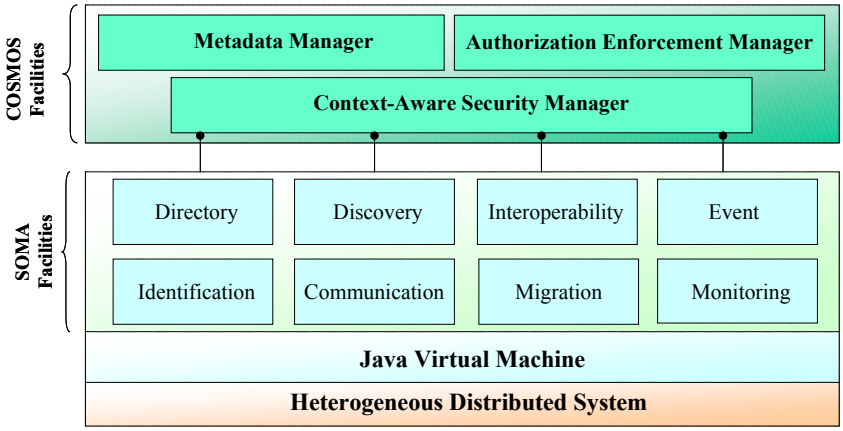

Figure 3. COSMOS middleware facilities.

Metadata Manager (MM). MM provides various tools for metadata editing, updating, removing, and browsing. In particular, with regards to authorization policies,

MM integrates tools for syntactic analysis of policy specifications, for translating user privacy requirements into security waves, and for transforming both system access control policies and security waves into Java objects, which act as policy information containers to be interpreted at runtime [16]. MM is also in charge of distributing specified metadata to the SOMA directory component responsible for storage, i.e., the one in the domain of first user registration (user home domain). At its first instantiation, any shadow proxy exploits the SOMA directory to retrieve its profile and security wave, which become part of its carried state.

Context-Aware Security Manager (CASM). CASM is responsible for establishing the active context view of any entering MA. When a new shadow proxy enters a domain, CASM calculates and returns to the proxy a valid active context view on the basis of the proxy responsible user profile, of the active system access control policies and of the security waves imposed by the other proxies currently executing in the domain. The active context view sent to the proxy is a copy of the active view maintained by CASM for any proxy in the locality, until the proxy exits from the domain.

CASM is also in charge of maintaining active context views up-to-date when relevant variations in context information occur, such as changes in the MAs executing in a locality, in resource availability, or in security wave specifications. CASM allows service providers to choose among differentiated view update strategies. At the two extremes, CASM supports eager and lazy update strategies. The *eager* strategy requires CASM to constantly coordinate with the underlying monitoring and event services in order to update active context views at the occurrence of any relevant context change. This strategy is expensive especially in environments with frequent context modifications, but provides proxies with an immediately and always updated active view. This is necessary when the updated active view is crucial to continuously re-adapt service tailoring/configuration strategies and to frequently evaluate the convenience to stay in the current domain.

The *lazy* strategy consists in updating the active context view only on user-demand, i.e., when the user requests the access to a resource in her current active view. This reduces CASM management load, but limits the possibility for a continuous service adaptation.

The choice of the most appropriate strategy to adopt depends on several factors, ranging from service requirements to the characteristics of the service deployment setting and to the tolerated trade-off between tailoring/configuration optimal choices and context update overhead. Similar considerations guided the implementation of protection domains in the JDK 1.2 security architecture [11].

Authorization Enforcement Manager (AEM). AEM mediates proxy-resource interactions by granting/denying proxies the access to resources, possibly depending on runtime conditions. Shadow proxies cannot directly access the resources included in their active context view, but have to interface with AEM at any resource access request. Active context views provide proxies with only the identifiers of accessible resources along with the permitted actions; no direct handles to the resources listed in the active view are returned to proxies.

When a proxy requests to access a resource, AEM intercepts the request and evaluates whether to deny/accept it. The type of access control checks needed de-

pends on the update strategy adopted by CASM. In the case CASM adopts an eager update strategy, AEM grants/denies the proxy request by simply checking whether the requested resource and action are included in the already updated active context view maintained by CASM. Let us note that if CASM, at the moment of the access control check request, is engaged in a view update task, AEM waits until CASM terminates the update and then performs the requested check. The eager strategy can speed up access control checks: at the resource access request, AEM does not need to evaluate all the applicable specifications of system access control policies and security waves; CASM have already calculated all granted permissions to maintain a fresh active context view.

On the contrary, in the case of lazy update strategy, AEM has to calculate the set of permissions that currently applies to the requesting proxy. AEM has to evaluate all authorization policies specifications and also to coordinate with the monitoring system to check if the context information specified in the `when` clause holds in the system.

4 Case Study

We have tested COSMOS in the design and implementation of a Context-centric Movie Assistant (CMA) that allows mobile users to find nearby cinemas and to exchange opinions about cinema characteristics, such as seat comfort, air conditioning, sound and screen resolution. The service exploits the visibility of user location to retrieve cinemas in the neighborhood performing the desired movie and the awareness of propagated user opinions to enable the choice of the most comfortable cinema according to the user-specific requirements.

Our testbed setting for CMA consists of a wireless metropolitan network composed by several 802.11 network localities, with each locality modeled as a SOMA domain. Each domain provides execution environments for shadow proxies, offers COSMOS middleware facilities and hosts info service components providing information about the locally available cinemas.

CMA users interact with the COSMOS infrastructure via device-specific clients running on their wireless access devices (Toshiba e740 Pocket PCs with Wi-Fi connectivity). The clients allow users to subscribe to CMA, by filling in a form that specifies user profile information and to authenticate to the service before starting any CMA session. After successful authentication, CMA instantiates a shadow proxy in the domain where the user is currently attached, and the proxy loads the user profile in its state part. At service provision time, the clients are only in charge of forwarding user requests (and of visualizing the received service results) to (from) their responsible proxies.

Let us consider the case of a user willing to see the “Spiderman” movie. As Figure 4A) shows, the user specifies in her user profile that she desires to know if there are nearby cinemas performing “Spiderman” and if they still have seats available. The user is also interested in using a local service of opinion management not only to browse other user opinions about the cinema characteristics but also to insert her personal judgment on a visited cinema.

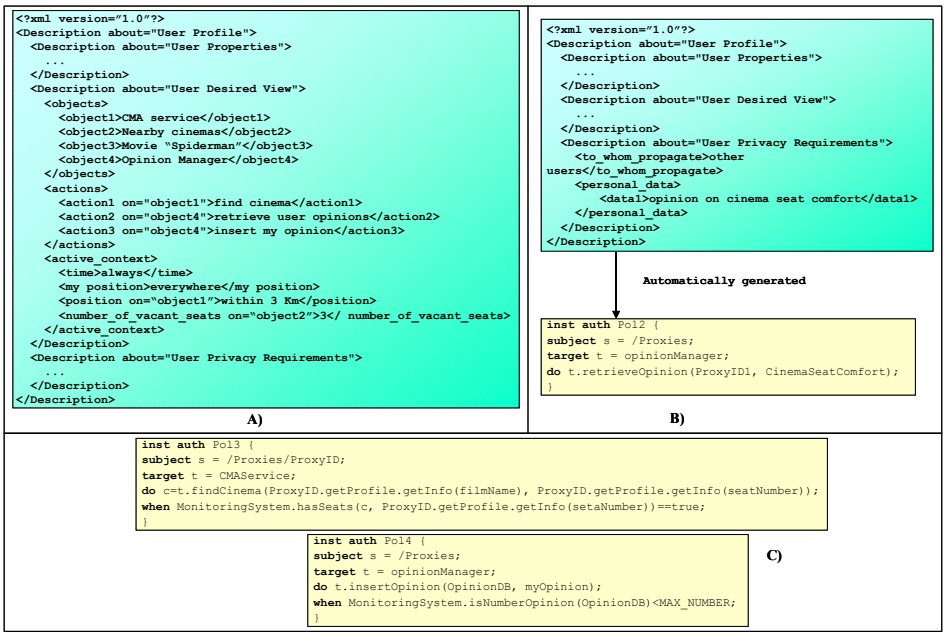

Figure 4. User profiles and authorization policies.

Another user may have already visited one of the nearby cinemas and expressed an opinion about its characteristics. The excerpt of the user profile in Figure 4B) shows the privacy requirements of this user, which allow the system to propagate the opinion about cinema characteristics to all other users co-located in the domain.

Figures 4B and 4C) describe some authorization policies that rule the proxy visibility of local resources. `Pol3` allows the proxy active context view to include nearby cinemas performing the desired movie if they have enough seats available. `Pol2` is the security wave that allows user opinion propagation. In particular, `Pol2` is automatically generated by COSMOS and derived from the user privacy requirements shown in Figure 4B). `Pol4` regulates proxy write operations on the opinion database managed by the opinion manager service. In particular, the policy states that all users can add their opinion to the opinion database if the number of already stored opinions does not exceed a threshold.

Given these configuration settings, when the proxy acting on behalf of the user represented by the profile in Figure 4A) enters the network locality, CASM retrieves and interprets the user profile and the policies to apply to the proxy, i.e., the policies where the proxy compares as the subject, and evaluates the policies by coordinating with the SOMA monitoring facility. On the basis of profile/policy metadata and of context conditions currently holding in the system, CASM generates the user active context view, composed by the nearby accessible cinemas with needed seat availability, by the local opinion manager service component where it is possible to retrieve the opinions about cinemas without violating user privacy, and by the list of actions that the proxy can perform on resources.

CASM can adopt various strategies to update the active context view. In the case of the eager strategy, CASM constantly monitors cinema seat availability and the status of the opinion database to check whether new opinions have been inserted. For instance, as long as the number of vacant seats is higher than the number requested by the user and no new opinions are available, the active context view does not change. Otherwise, CASM immediately updates the active context view, e.g., by removing from the nearby list of cinemas the one that have no more seats available. In the case of the lazy strategy, CASM calculates the active context view only once at first proxy arrival in the locality. Once the proxy receives the active view, it can decide the cinema to reach.

Let us note that the two update strategies have a different impact on the level of quality of service offered to the user. If CASM adopts the eager strategy, the user has online visibility of vacant seats in the cinema. This means that the user can see if other users are buying the last tickets and consequently can change her destination. On the contrary, if CASM adopts the lazy strategy, its management overhead is reduced, but the user could have an obsolete view of the cinema seat availability status. So she can run the risk of going to the cinema, of parking her car and of finding out only at destination that the needed tickets are no more available.

Suppose now that at the end of the movie the user is willing to express her opinion about the cinema characteristics. Her device-specific client forwards the request along with the opinion content to the responsible proxy that tries to access the opinion database. AEM intercepts the request and `Pol4` rules the proxy access to the opinion database. AEM access control checks depend on the active context view update strategy adopted by CASM. In the case of the eager strategy, CASM has already evaluated `Pol4` at proxy request time by maintaining the "insertOpinion" action in the list of permitted actions as long as the opinions in the database remain under the predefined maximum number. This relieves AEM from repeating policy evaluation: if the "insertOpinion" action is included in the fresh active context view maintained by CASM, AEM allows the proxy to insert user opinion. In the case of the lazy strategy, AEM has to evaluate `Pol4` and check the conditions in the `Pol4 when` clause.

5 Conclusions and Ongoing Work

Effective service provisioning over the wireless Internet requires the full visibility and the flexible management of context information. Requirements for context visibility start to be recognized in the security area also for traditional fixed networks, where interesting novel proposals are emerging to enhance protection techniques with context awareness [19, 20]. By focusing on mobile environments, COSMOS proposes and implements a novel security model for context-centric access control that exploits different types of metadata to express articulated security strategies, separately from the application logic. This separation of concerns increases flexibility, dynamicity and reusability of middleware/service components. In addition, the choice of the most proper evaluation strategy allows administrators to tune the access control overhead, which is acceptable for most wireless Internet services, usually having no hard real-time constraints.

First experiences in implementing services on top of COSMOS are encouraging further research to extend the middleware prototype. We are working on how to deal

with possible runtime conflicts among policies to be enforced, and we are investigating and prototyping solutions based on policy prioritization. Finally, we are integrating COSMOS with mechanisms for inter-cell mobility prediction based on IEEE 802.11 signal strength variations, in order to anticipate the migration of (copies of) shadow proxies towards the new locality and the determination of the new active views in advance.

References

1. L.F. Akyildiz, J. McNair, J. Ho, H. Uzunalioglu, W. Wenye, "Mobility management in current and future communications networks", *IEEE Network*, Vol. 12, No. 4, July/August 1998.
2. A.K. Dey, G.D. Abowd, "Towards a Better Understanding of Context and Context-Awareness", *Proc. of CHI*, The Hague, The Netherlands, April 2000.
3. T. Rodden, K. Cheverst, K. Davies, A. Dix, "Exploiting Context in HCI Design for Mobile Systems", *Proc. of Workshop on Human Computer Interaction with Mobile Devices,* Scotland, May 1998.
4. P. Bellavista, A. Corradi, R. Montanari, C. Stefanelli, "Dynamic Binding in Mobile Applications: a Middleware Approach", *IEEE Internet Computing*, Special Issue on "Mobile Applications", Vol. 7, No. 2, March/April 2003.
5. IKV++ Technologies AG, enago Open Service Platform, http://www.ikv.de
6. P. Bellavista, A. Corradi, C. Stefanelli, "The Ubiquitous Provisioning of Internet Services to Portable Devices", *IEEE Pervasive Computing*, Vol. 1, No. 3, July-September 2002.
7. G. Vigna, (ed.), "Mobile Agents and Security", Springer-Verlag, LNCS 1419, 1998.
8. R. Oppliger, "Security issues related to mobile code and agent-based systems", *Computer Communications*, Elsevier Science, Vol. 22, No.12, 1999.
9. R. Montanari, C. Stefanelli, N. Dulay, "Flexible Security Policies for Mobile Agents Systems", *Microprocessors and Microsystems*, Elsevier Science, Amsterdam, Olanda, Vol. 25, No. 2, 2001.
10. N. Mitrovic, U. Arronategui Arribalzaga, "Mobile Agent security using Proxy-agents and Trusted domains", *Proc. of SEMAS 2002*, Bologna, Italy, July 2002.
11. L. Gong, "Inside Java 2 Platform Security", Addison Wesley, 1999.
12. P. Bellavista, A. Corradi, C. Stefanelli, "Protection and Interoperability for Mobile Agents: A Secure and Open Programming Environment", *IEICE Transactions on Communications (Special Issue on "Autonomous Decentralized, Systems")*, Vol. E83-B, No. 5, May 2000.
13. J. Al-Muhtadi, R. Campbell, A. Kapadia, M. D. Mickunas, S.g Yi, "Routing Through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments", *Proc. of ICDCS'02*, IEEE Press, Vienna, Austria, July 2002
14. S. Wright, R. Chadha, G. Lapiotis (eds.), Special Issue on "Policy Based Networking", *IEEE Network*, Vol. 16, No. 2, March 2002.
15. G. Myles, A. Friday, N. Davies, "Preserving privacy in environments with location-based applications", *IEEE Pervasive Computing*, Vol. 2, No.1, January/March 2003.
16. A. Corradi, N. Dulay, R. Montanari, C. Stefanelli, "Policy-driven Management of Mobile Agent Systems", *Proc. of Policy 2001*, Springer-Verlag, LNCS 1995, Bristol, January 2001.
17. N. Damianou, N. Dulay, E. Lupu, M. Sloman, "The Ponder Policy Specification Language", *Proc. of Policy 2001*, Springer-Verlag, LNCS 1995, pp. 18-39, Bristol, January 2001.

18. M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, L. Yu, "Negotiating Trust on the Web", *IEEE Internet Computing*, Vol. 6, No. 6, November/December 2002.
19. M. J. Covington, W. Long, S. Srinivasan, A.K. Dey, M. Ahamad, G. D. Abowd, "Securing Context-Aware Applications Using Environment Roles", ACM, *SACMAT'01*, Chantilly, Virginia, USA, May 2001.
20. G. K. Mostéfaoui, P. Brézillon, "A Generic Framework for Context-Based Distributed Authorization", *CONTEXT'03*, LNAI 2680, pp. 204-217, 2003.