Context-based Access Control Management in Ubiquitous Environments

Antonio Corradi, Rebecca Montanari, Daniela Tibaldi

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

Abstract

Wireless connectivity and the widespread diffusion of portable devices raise new challenges for ubiquitous service provisioning. Mobility of users causes frequent and unpredictable changes in user location and in consequently available resources. Access control to resources is crucial to leverage the provision of ubiquitous services and calls for novel solutions based on various context information, e.g., user location, device properties, user needs, local resource visibility. This paper presents a novel access control model that proposes the adoption of context as a first-class design principle to rule access to resources. The paper proposes a context-centric access control middleware, called UbiCOSM, that dynamically determines the contexts of mobile users and effectively rules the access to them, by taking into account different types of metadata: user profiles and system/user-level authorization policies. The paper also presents a context-dependent movie-info service to evaluate the functioning of UbiCOSM.

1. Introduction

The widespread availability of wireless network connectivity in the environments where users live and work together with the increasing diffusion of portable devices creates novel opportunities for users to access services anywhere, at any time and from various access devices. In particular, mobile users/devices can have access not only to traditional Internet services, designed and implemented for the fixed network infrastructure, but also to new classes of services that can provide results depending on the relative position of clients and on the consequent resource visibility.

However, the design and deployment of ubiquitous services lead to serious security risks and access control problems and impose new challenges to the secure retrieval and operation on distributed resources, undermining several assumptions of traditional security solutions. Traditional solutions typically evaluate permissions depending on the identity/role of the client requesting access to resources. However, the new ubiquitous scenario makes service providers deliver services often to unknown entities and, more important, whose identity may be un-informative or not sufficiently trustworthy. In fact, it is almost impossible for service providers to know in advance the identities/roles of all subjects that are likely to request access to their managed resources/services. Instead, service providers can more easily define the conditions for making resources available and for allowing/denying users resource visibility and access according to the context operating conditions. In the following, we define context as the collection of any information useful to characterize the runtime situation of a user during her service session [1].

Some initial research works are starting to recognise that ubiquitous service provisioning requires a paradigm shift from subject-centric to context-centric access control [2, 3, 4, 5]. Novel access control middleware solutions should consider context as a first-class principle to guide both policy specification and enforcement process. In this perspective, the changes in context should trigger the evaluation process of applicable permissions. Drawing inspiration from the RBAC model that exploits the concept of role as a mechanism for grouping subjects based on their properties [6], we state that, the same as with role, the concept of context can provide an indirection level between users and permissions. Instead of managing subjects and their permissions individually, a system administrator defines for each context the set of applicable permissions. When a subject operates in a specific context, she instantaneously acquires the set of permissions active for the related context. When she changes her operating context, her previous permissions are automatically revoked and the new permissions acquired.

The paper presents the design and implementation of a highly dynamic and flexible security middleware, called UbiCOSM (Ubiquitous Context-based Security Middleware), that adopts context as the basic concept for security policy specification and enforcement processes. Unlike traditional access control models, permissions are directly associated with contexts, instead of user identities/roles: any mobile user/device acquires a set of permissions by entering a specific context. In UbiCOSM context-based access control policies are expressed at a

Proceedings of the Third IEEE International Symposium on Network Computing and Applications (NCA'04)
0-7695-2242-4/04 $ 20.00 IEEE

high level of abstraction in terms of metadata and they are cleanly separated from the service logic. UbiCOSM allows not only security administrators to specify system access control policies in order to avoid illicit accesses to local resources, but also users to specify their privacy requirements in order to regulate the disclosure of their personal information when entering a new context.

The rest of the paper is organized as follows. Section 2 presents the UbiCOSM security framework and Section 3 the proposed middleware architecture. Section 4 describes a context-dependent movie-info service prototype, built on top of UbiCOSM and deployed over a group of coordinated IEEE 802.11 localities, to evaluate the usability and effectiveness of the proposed middleware. Finally concluding remarks follow.

2. UbiCOSM Security Framework

UbiCOSM is an access control middleware for securing service provisioning with context awareness requirements. In particular, UbiCOSM focuses on three main peculiar aspects: flexible solutions for context-centric access control, active context view provisioning to mobile users, and privacy support in the propagation of user context information. UbiCOSM access control decisions depend on dynamic context attributes, such as resource state and availability, in addition to more traditional attributes, e.g., the identity/role of user requesting a resource access.

Another distinctive feature of UbiCOSM is to provide users entering a new locality with a controlled visibility of the directly accessible physical/logical resources and of the other users locally acting (*active context views*). As Figure 1 shows, active context views contain resources that both users are willing to access (*desired view*) and the UbiCOSM access control function have qualified as accessible depending on currently active context-dependent access control policies (*allowed view*). The provision of active context views to users has many benefits. Users can exploit the visibility of available resources to adjust their behaviour accordingly and to reduce the risk of undesired task failure during execution. Active context views can also help users to decide whether it is more profitable to stay in a locality than to move from it and to explore a new computing environment.

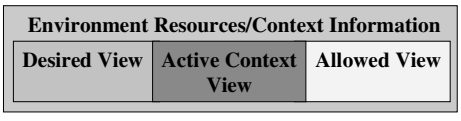

Figure 1. Active context view.

UbiCOSM addresses also the privacy issues that arise in context-aware service scenarios. 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. UbiCOSM protects user privacy by enabling users to specify some special kinds of security policies, thus defining which personal information they are willing to make public. User specifications guide and automate UbiCOSM access controls to user personal information.

2.1 UbiCOSM Context Model

UbiCOSM access control management distinguishes two different kinds of context: *physical* and *logical* (see Figure 2). Physical contexts identify physical spaces, delimited by specific geographical coordinates. A user operates in a particular physical context depending on her current location. At any time, one user can belong to only one physical context. Physical contexts define specific boundaries for access control policy management: each physical context holds references to the resources to be protected.

Logical contexts identify logical states of both physical contexts and entities composing an ubiquitous service deployment scenario, e.g., users and resources. Logical states depend on logical properties, e.g., temporal conditions, resource availability and status, user activities and also user device characteristics. At any time, entities may be associated with different logical contexts. It is worth stating that UbiCOSM considers user identities and roles as specific types of logical contexts. This allows UbiCOSM to handle also subject-based access control if needed.

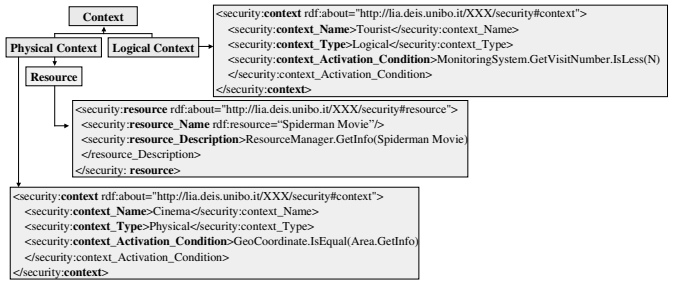

Figure 2. UbiCOSM Context Model: physical and logical contexts.

UbiCOSM adopts an RDF-based standard format for context representation to overcome heterogeneity of data

representation over different architectures. As it stems from Figure 2, both physical and logical contexts have a ***Name*** that uniquely identifies the context, a ***Type*** qualifying the context type, either logical or physical, and a set of ***Activation Conditions*** that represent the physical/logical conditions that determines the activation of the context. For instance, the *Cinema* context example shown in Figure 2 represents a physical area delimited by the geographical coordinates defined in the activation conditions. The *Tourist* context represents a logical context modelling the Tourist user role within the environment where UbiCOSM executes. In particular, this logical context is active for users who have visited the monitored location less than a threshold number of times.

Figure 2 also shows an example of UbiCOSM RDF-based resource representation including a ***Name*** that identifies the resource and a ***Description*** that reports the resource properties. In the figure, the resource representing a *Spiderman Movie* has some properties given via a local resource manager. The resource description contains information about resource type, resource owner and resource usage modalities.

While the description of physical and logical context properties is static, the association between contexts and the entities operating in them is dynamic to more easily accommodate system evolutions.

2.2 UbiCOSM Security Model

UbiCOSM allows system administrators and final users to specify their security requirements at a high level of abstraction in terms of metadata. Metadata are declarative rules that describe both user/device/resource profiles and authorization policies. Metadata permit to separate security logic from security control and facilitate automated security reasoning: all basic elements involved in access control decisions can be easily extracted from declarative notations, analysed and checked for conflicts.

Figure 3a) shows the two kinds of metadata supported in UbiCOSM: *profiles* and *access control policies*.

Profiles. 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 [1]. In particular, UbiCOSM decomposes the user profile in two sub-structures: *user properties* and *desired view*. User properties specify user characteristics relevant to qualify the user to the system, such as user physical characteristics, user logical properties like name or role and so on. In what follows, we will focus on the desired views that express user preferences about the desired action and resource visibility. The excerpt of profile shown in Figure 3b) illustrates an example of a desired view where the user is willing to know if there are cinemas within 3 Km. This desired view is composed by the set of desired objects, i.e., cinemas (the objects tag), the set of actions that she would like to perform on these objects, i.e., find vacant seats (the actions tag), and the context conditions in which she desires to perform the specified actions, i.e., within 3 Km (the active_context tag).

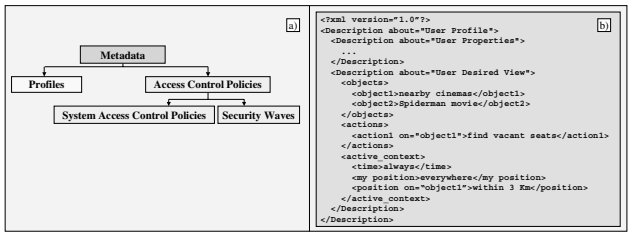

Figure 3. Metadata taxonomy and examples in UbiCOSM.

UbiCOSM adopts XML-based standard formats for profile representation to deal with the heterogeneity of data representation over different architectures.

Access Control Policies. UbiCOSM access control policies rule context-based access control and determine the user *allowed view*. UbiCOSM distinguishes between system- and user-level authorization policies (*system access control policies* and *security waves* in Figure 3a). The formers are specified to protect system resources from illegitimate use by incoming mobile users. Through system access control policies, security administrators define which permissions are activated on target resources by the specified context conditions. Security waves are used to reduce user privacy violations. Through security waves, users specify which personal information UbiCOSM can propagate to other users in specific context situations. This ensures a controlled disclosure of user-specific critical information.

UbiCOSM exploits an RDF-based standard format to express access control permissions. As shown in Figure 4a), the definition of a permission includes a ***Name*** that identifies the permission, an ***Action*** that specifies an allowed operation, a ***Target*** representing the resource the specified action can apply on and a ***Kind*** representing the positive or negative meaning of the permission. Figure 4a) shows an example of permission (*P1*) that allows to see *Horror Movies*.

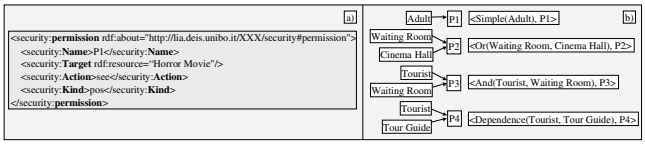

Figure 4. UbiCOSM RDF-based permissions and context-centred access control policies.

UbiCOSM context-centric access control policies are defined as association rules between a set of permissions and a set of contexts. Thus, policy specification requires not only to define permissions, but also to associate them with the contexts in which they should apply. In particular, UbiCOSM allows to associate permissions either to individual contexts (*simple association*) or to multiple contexts composed according to *and*, *or,* and *dependence* relationship (*and, or, dependence association*).

As it stems from Figure 4b) policy semantics changes according to the type of association. In the case of a simple association, access control permissions are associated to only one context and they apply when it is active. Thus they are enforced on any mobile client currently operating in that context. For example, referring to Figure 4b) and 4a), the permission *P1* is granted to every *Adult* through a *simple association,* thus only adults are allowed to see *Horror Movies*.

The other kinds of relationships permit to express some more complex association rules. In the case of an *or, and, dependence association*, access control permissions are associated to more than one contexts. Referring to an *or* association permissions are activated if at least one of the associated contexts is active. For example, if the permission *P2* has to be granted to users located either in the *Waiting Room* or in the *Cinema Hall*, it has to be associated with the collection of both physical contexts composed through an *or* association. In this case, *P2* applies to users whose active physical context is either *Waiting Room* or *Cinema Hall*. On the other hand, referring to an *and association*, permissions are activated only if all the specified contexts are active at the same time. As shown in Figure 4b) *P3* is associated with the *Tourist* and *Waiting Room* contexts and is activated when both are active. Finally, the *dependence association* permits to address more complex situations that may occur during an ubiquitous service provisioning session. In particular, it permits to rule access behaviour of a mobile client operating in a specific logical context only in presence of other mobile clients operating in the same physical context. In the case of a dependence association, access control permissions are activated when the logical context for a specific mobile client is active and when the contexts of all co-located mobile clients in the *dependence* relationship with the mobile client context are active. Figure 4b) shows an example of a permission *P4* activated for a mobile client in the *Tourist* logical context only if she is co-located with users operating in the *Tour Guide* logical context.

As shown in Figure 4b), UbiCOSM access control policies are expressed in terms of tuple with the following format:

<*association_Name*(*context_collection), permissions*>.

The first argument identifies a collection of one or more contexts to which associate the set of permissions. In the case of a *simple* association the collection is made by one context, while in the case of other associations the collection consists of at least two contexts joined by an *and, or, dependence* association. The *association_Name* identifies the association itself, e.g., *Simple, And, Or, Dependence*.

3. UbiCOSM Access Control Middleware

Figure 5 shows the UbiCOSM middleware services built on top of the context-aware CARMEN middleware [7]. CARMEN provides lower-level functions for entity identification, resource discovery, directory, context management, and event registration/dispatching. On top of these services UbiCOSM furnishes the additional facilities. We herein detail the key services for the support of access control management.

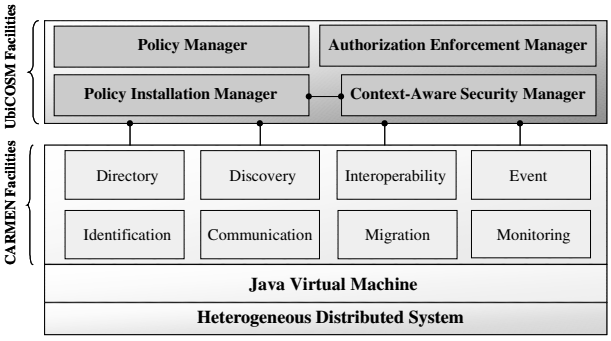

Figure 5. UbiCOSM middleware services.

The **Policy Manager** (PM) provides security administrators and final users with various tools for system access control policy and security wave editing/browsing.

The **Policy Installation Manager** (PIM) is in charge of maintaining associations between contexts and their applicable permissions. Thus, PIM installs the specified *system access control policies* and *security waves* into the

system, by storing associations between contexts and permissions into hash-tables. In particular, PIM creates four tables, one for each kind of context association: *simple, and, or, dependence*. Figure 6) shows some examples of hash tables. In particular, as it stems from figure 6a), the table corresponding to a *simple association* is a mono-dimensional array whose each cell maintains a reference to a permission list storing all the permissions applicable to the context of the corresponding cell. Context is the key of the hash table, thus getting a reference to a context name, it is possible to directly access its applicable permissions.

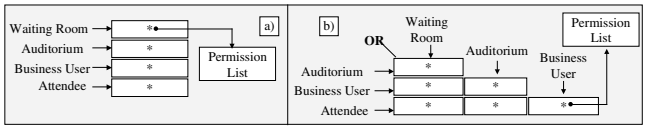

Figure 6. Hash tables for context-permission relationships.

Figure 6b) shows an example of a hash table corresponding to an *or association*. This table has two dimensions and each cell maintains the reference to the permission list applicable to users in the context of correspondent row or column. In order to consider associations of more than two contexts, it is possible for PIM to store context-permission associations into multi-dimensional arrays, where each dimension is a context, argument of the association. Thus, for example, it is possible to consider a three-dimensional array whether it is necessary to activate a set of permissions when the *or(a, b, c)* context association is active.

The **Context-Aware Security Manager** (CASM) is responsible for computing the set of applicable context-centric access control policies for mobile clients thus establishing the active context view of any entering user. Therefore CASM calculates and returns to the user a valid active context view on the basis of the active context situation, of the active system access control policies and of the security waves imposed by other users currently executing in the domain. In particular, when a mobile client enters a physical context, CASM evaluates the permission set associated with the physical context and with the logical contexts of all involved entities, e.g., of entering mobile clients, of resources in the current physical context and of other co-located mobile clients. CASM coordinates with the CARMEN context management service to retrieve all significant physical/logical contexts. For each of them, CASM has access to hash tables storing context-permission relationships in order to extract permissions that apply. Among the whole set of retrieved permissions, CASM considers only the ones that apply to resources locally available in the physical context and generates the active context view. The view sent to the entering user is a copy of the active view maintained by CASM for any user in the locality, until the user 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 users 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.

An eager strategy requires CASM to constantly coordinate with the CARMEN context monitoring service in order to update permission views whenever any significant change in the involved entity context occurs. This strategy is expensive especially in environments with frequent context modifications, but provides users with an immediately and always updated view. This is necessary when an updated permission view is crucial to continuously re-adapt service provisioning and to frequently evaluate whether it is convenient for a mobile client the permanence in the current domain.

On the other hand, a lazy strategy updates the view on user-demand, i.e., when the user requests access to a resource in her current active view. This reduces CASM management load, but limits the capacity of 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 trade-off tolerated between service provisioning optimal choices and context update overheads. Similar considerations guide implementation of protection domains in JDK 1.2 security architecture [8].

Authorization Enforcement Manager (AEM). AEM mediates mobile clients-resource interactions by granting/denying clients the access to resources according to the active context-centred policy specification. Users 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 users 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 user requests to access a resource, AEM intercepts the request and evaluates whether to deny/accept it. The type of needed access control checks depends on the update strategy adopted by CASM. If CASM adopts an eager update strategy, AEM grants/denies the user request by simply checking whether the requested resource and action are included in the already updated permission view maintained by CASM. At the moment of the access control

check request, if CASM 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 applicable specifications of system access control policies; CASM has already updated all granted permissions to maintain fresh visibility of accessible resources. On the contrary, with a lazy update strategy AEM coordinates to CASM to calculate the set of permissions that currently applies to the requesting user and then grants/denies the access according to the evaluation.

4. Case Study

We have tested UbiCOSM in the design and implementation of a Context-centric Movie Assistant (CMA) that allows mobile users to find nearby cinemas, to book cinema seats/tickets and to exchange opinions about cinema characteristics, such as seat comfort, air conditioning, sound and screen resolution. In particular, CMA exploits the visibility of context information, such as user location and past booking behaviour to control ticket booking: users who are near the cinema and who have never cancelled their reservations have booking priority.

In addition, CMA exploits the visibility of user location to retrieve nearby cinemas 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. Each locality offers UbiCOSM middleware facilities and hosts info service components providing information about the locally available cinemas. CMA users interact with the UbiCOSM 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 and to authenticate to the service before starting any CMA session.

To show the functioning of UbiCOSM services, let us consider the case of a user, Lucy, willing to see the "Spiderman" movie. Lucy exploits CMA to check whether there are nearby cinemas performing Spiderman and whether there are still seats available. Before booking the tickets Lucy has access to a local CMA opinion management service to browse posted opinions about the nearby cinema characteristics to select the most comfortable one. Then she chooses the cinema where to watch the desired movie and has access to the CMA ticket booking service to reserve a couple of tickets for a specific showing. Finally Lucy can insert her personal judgment on the visited cinema.

Figure 7 shows a subset of access control policies that govern Lucy access to the CMA ticket booking and opinion management services (Figure 7a, 7b and 7c) and that rule user opinion visibility (Figure 7d).

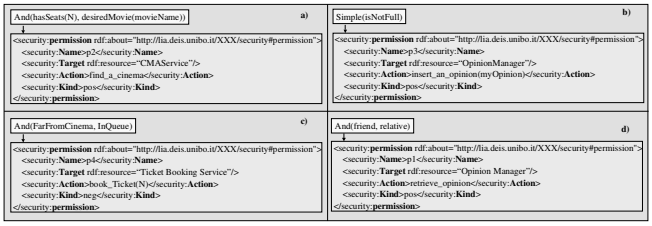

Figure 7. Sample of context-permission association

In particular, Figure 7a) shows a system access control policy that allows the access to the CMA service and to find a nearby cinema with *N* available seats and showing the movie *movieName*. Figure 7b) shows an example of a system access control policy that allows users to insert their opinion if the opinion management service is not full. The p4 policy of Figure 7c) denies a user to have access to the ticket booking service if there is someone else in queue for the same desired movie and if the user is far away from that cinema. It is important to note that the context *FarFromCinema* represents the maximum distance from which the user is allowed to reserve a ticket. The distance varies depending on the starting show time. This security policy attempts to favour the nearest users to cinema.

Figure 7d) shows the Lucy security wave, which allows the system to propagate her opinion about cinema characteristics only to her *relatives* and *friends*.

When Lucy moves through different localities and context conditions change, CASM coordinates with the underlying CARMEN system to retrieve Lucy active context situation. For each of the obtained context, CASM considers the correspondent entries in the hash tables storing context-permission associations. CASM extracts applicable permissions and generates the Lucy active context view. In particular, Lucy active context view includes the nearby accessible cinemas with needed seat availability, the local opinion manager service component where it is possible to retrieve the opinions about cinemas without violating other user privacy and the list of accessible resources on the basis of context-centric security policies.

It is worth stating that contexts are not statically associated with users. For example, if other users buy tickets for a cinema performing the desired movie, it is

possible that the cinema has not enough seat available. Thus CASM has to update the user active context view by removing from the nearby list of cinemas the one that have no more seats available. UbiCOSM automates this active context view update because any detected context change activates/deactivates the correspondent security policies.

The UbiCOSM exploitation of the context concept as a mechanism for grouping applicable policies simplifies policy specification and policy update/revocation.

Traditional access control systems exploit user identity/role information to determine the set of user permissions. Context information can only further limit the applicability of the available permissions. It is worth stating that the specification of traditional access control policies tightly couples identity/role of client users with their permissions and with the operating conditions in the system to grant the permitted actions to take place. This coupling requires security administrators to foresee all contexts client users are likely to operate. In ubiquitous environments where client users are typically unknown and where context operating conditions frequently change even unpredictably, the traditional approach to specify access control policy may lead to a combinatorial explosion of the number of policies to be written and may force a long development time and can induce potential bugs. This approach also lacks flexibility: new access control policies need to be designed and implemented from scratch for any user when new context situations occur.

L	t(msec)
10	0.26
20	0.48
30	0.75
40	0.96
50	1.25
60	1.50
70	1.75
80	1.93
90	2.13
100	2.52

Table 1. Performance overhead to compute an active context view.

However, we have to state that UbiCOSM access control model and enforcement introduces some performance overhead depending on several factors, e.g. user physical and logical context determination, associations evaluation between contexts and permissions, and access control checks. In particular, the number of permissions UbiCOSM has to evaluate depends on the number of contexts associated (directly or through an *association*) with both the entering user and co-located users and resources. In particular, we herein focus on the performance tests we have carried out to evaluate the overhead introduced by CASM to build an active context view.

Table1 shows some performance test results that report the time delay for computing a user active context view for different values of user and resource logical contexts (L), considering a fixed number of co-located users, resources and policies (20 co-located users, 20 co-located resources and 20 security policies for each context).

Figure 8 shows the time needed to update an active context view when the number of access control policies for each context raises. In the figure a fixed number of contexts, co-located users and resources is considered.

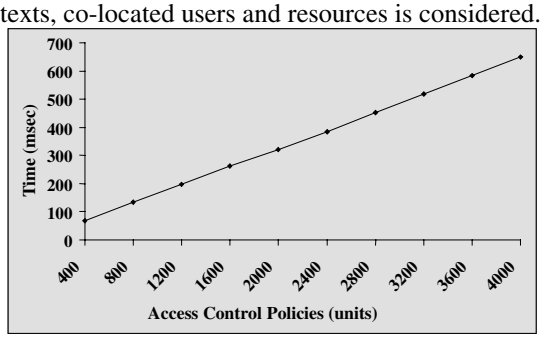

Figure 8. Performance overhead to update an active context view.

It is worth stating that the two update strategies for the active context view have a different impact on the level of quality of service offered to the user. If CASM adopts an eager strategy, the user has online visibility of vacant seats in the cinema. This means that the user can immediately see if other users buy the last tickets and consequently she can change her destination. Thus, having an immediate controlled visibility of directly accessible physical/logical resources in a network locality facilitates user decision on the profit of the permanence in the network locality compared with the migration to another one. On the contrary, if CASM adopts a lazy strategy, its management overhead is reduced, but the user could have an obsolete view of the cinema seat availability status. 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.

5. Conclusions

Service provisioning in ubiquitous environments requires full visibility and flexible management of context information. The need to propagate system information up at the application level and to exploit context visibility start

to be recognized also in the security area, where interesting novel proposals are starting to emerge to build context-aware protection techniques with context awareness [2, 3, 4, 5]. By focusing on mobile environments, UbiCOSM proposes and implements a novel context-centric access control framework specifically designed to protect resource access and to provide a privacy support in the propagation of user personal information in ubiquitous service provisioning scenarios.

First experiences in implementing services on top of UbiCOSM are encouraging us enough to work on the extension of the middleware prototype. We work on possible conflicts, by exploiting solutions based on policy priority. Finally, we are integrating UbiCOSM with mechanisms for inter-cell mobility prediction based on IEEE 802.11 signal strength variations, in order to anticipate the migration of mobile users towards the new locality and the determination of the new permission views in advance.

References

[1] 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.

[2] 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.

[3] G. Neumann, M. Strembeck, "An Approach to Engineer and Enforce Context Constraints in an RBAC Environment", ACM, *SACMAT'03*, Como, Italy, June 2003.

[4] P. McDaniel, "On Context in Authorization Policy", ACM, *SACMAT'03*, Como, Italy, June 2003.

[5] G. K. Mostéfaoui, P. Brézillon, "A Generic Framework for Context-Based Distributed Authorization", *CONTEXT'03*, LNAI 2680, pp. 204-217, 2003.

[6] R. S. Sandhu, E. J. Coynek, H. L. Feinsteink, C. E. Youmank, "Role-Based Access Control Models", *IEEE Computer*, Vol. 29, No. 2, February 1996.

[7] P. Bellavista, A. Corradi, R. Montanari, "Context-Aware Middleware for Resource Management in the Wireless Internet", *IEEE Transaction on Software Engineering*, Special Issue on "Software Engineering for the Wireless Internet", Vol. 29, No. 12, December 2003.

[8] L. Gong, "Inside Java 2 Platform Security", Addison Wesley, 1999.