Design of a Context Privacy Service for Mobile Collaboration

Vagner Sacramento, Markus Endler and Fernando Ney Nascimento [*]

[1]Departamento de Informática, PUC-Rio
R. Marquês de São Vicente 225
22453-900, Rio de Janeiro, Brazil

{vagner,endler,ney}@inf.puc-rio.br

Abstract. *Privacy related to context information is becoming an increasingly important issue due to the wide range of new applications using computational, location and personal context information. In this article, we propose a privacy service, called* ***Co****ntext* ***P****rivacy* ***S****ervice (*CoPS*), to control how, when and to whom disclose a user's context information. In order to identify the end-user privacy concerns, we carried out a survey where we asked approximately 100 people about their willingness to use context-aware services and to disclose their context information. Based on the results of this survey and experience reported by other research groups, we identified the main service requirements and designed* CoPS *aiming flexibility, generality, simplicity and fine-grained privacy control.* CoPS *is supposed to aid the end-user to define and manage his privacy policies regarding his context information.*

Keywords:
Privacy, Context Service, Context-awareness, Middleware, Collaboration.

Resumo. *Privacidade de informações de contexto está se tornando cada vez mais importante diante da grande diversidade de novas aplicações que utilizam informações de contexto pessoal e computacional. Neste artigo, propomos um serviço de privacidade,* ***Co****ntext* ***P****rivacy* ***S****ervice (*CoPS*), para controlar como, quando e para quem as informações de contexto do usuário podem ser compartilhadas. Para identificar as necessidades de privacidade do usuário realizamos uma pesquisa com aproximadamente 100 pessoas sobre o interesse em utilizar serviços com percepção de contexto e mecanismos que controlam como tais informações serão divulgadas. Com base nos resultados dessa pesquisa e na experiência relatada por outros grupos identificamos os principais requisitos do serviço de privacidade e, projetamos o* CoPS *visando um serviço flexível, simples e com controle de privacidade mais preciso. O serviço proposto tem o objetivo de auxiliar os usuários a definir e administrar suas políticas de privacidade em relação a suas informações de contexto.*

Palavras-chave:
Privacidade, Serviço de Contexto, Percepção de Contexto, Middleware, Colaboração.

1. Introduction

For more than a decade, context-provisioning services and middleware have been extensively investigated and proposed, with the goal to support the development of

[*]Partially supported by CNPq research grants no. 552.068/02-0 (Project ESSMA) and no. 55.2028/02-9 (Project IMPAR).

context-aware applications, specially for mobile networks. [Chen and Kotz, 2000, Dey et al., 2001, Schilit et al., 1994, Schmidt et al., 1999] have proposed two major groups of context information: physical and human factors. The first group was further divided into three sub-groups: *environment* (e.g. light, pressure, temperature, time), *computational* infra-structure (e.g. resources, communication capabilities, system load), and *location* (coordinates, symbolic, absolute, relative, etc.). Human factors were also classified into sub-groups: *personal* information (e.g. preferences, interests, skills, etc.), *social relationships* (groups, family), and *activity* (e.g. occupied, travelling, resting, etc.). Of all these, only *physical* context can be automatically measured, and in this category, *location* has probably been the mostly explored aspect of context, specially for *Location-based Services* [Schiller and Voisard, 2004].

In fact, access to computational context and location information of end-user devices opens a wide range of new possibilities for implementing distributed collaborative applications. For example, location information can be displayed by instant communication services, physical proximity among users can be used to select collaborating peers, information about the connectivity status can enhance mutual collaboration awareness. This motivated us to design and implement a service-oriented middleware architecture called ***Mo****bile* ***C****ollaboration* ***A****rchitecture* - MoCA [Sacramento et al., 2004, MoCA Team, 2004b] for collecting and processing context data from mobile devices in 802.11 networks, and making it available to applications. We are currently using this middleware to implement context- and location-aware applications for mobile collaboration.

While some users may want to use such applications to facilitate group coordination or convey a sense of presence with friends or co-workers, there are also some serious concerns about the risk of disclosing personal context information. Hence, there is a huge demand for tools which provide end-users with the ability to define organization- and person-specific privacy control over context data. But since privacy is a very broad concept that entails very different interpretations and requirements, a context privacy solution should be flexible and adaptive to the specific needs of individuals, user communities and applications.

In order to identify the most relevant user needs and privacy concerns, we carried out a survey with approximately 100 end-users of different background and age. The results of this survey, as well as experience reported by other groups, helped us to identify the main system requirements and inspired our design of a flexible, powerful and generic privacy control mechanism for context information, which we named "**Co**ntext **P**rivacy **S**ervice" (CoPS). This service allows the end-users to share their context data with the right people, at the right level and at the right moment through the following features: group-based access control, hierarchical privacy rules, rule specificity analysis (based on the requester ID, spacial and temporal precision, and information freshness), optional user notification, logging and plausible deniability mechanisms.

The remainder of this paper is structured as follows. Section 2 presents a discussion about some related work. Section 3 shows the results of our survey about requirements for privacy control, and Section 4 presents a short overview of MoCA and the type of context data that it makes available. Section 5 introduces CoPS's main features and describes the typical pattern of interaction among CoPS, a context service and an application. Then, Section 6 presents the main components of CoPS, giving emphasis on the structure of the privacy rules, the policy hierarchy, group definitions, the algorithm for rule specificity analysis, and some concrete scenarios illustrating how the algorithm works. Finally, in Sections 7 and 8 we discuss future work and give concluding remarks.

2. Related Work

Recently, research about mechanisms for privacy of control of context information has received increased attention. In the following we discuss the work most related to this field.

In Confab [Hong and Landay, 2004], people, places, things, and services are assigned to *info-spaces*, which are tuple-spaces storing static or dynamic context data about any of the info-space's entities. The context data stored in *info-spaces* are *contexts tuples*, and are populated by sources of information, such as sensors. Similar to other tuple spaces, an *info-space* supports *in-* and *out-methods*. *In-methods* affect what data is stored within an *info-space*, and include add and remove methods. *Out-methods* govern any data leaving an info-space, and include query, subscribe, unsubscribe, and notify. Privacy mechanisms are enforced through *info-space operators* (*in-* and *out-operators*), which govern what data can enter or leave the *info-space*. *In-operators* and *out-operators* are run on all tuples coming in/out. These operators can apply, for example, the info-space's access control policies to ensure that a tuple is allowed to be added/removed, or that this tuple should be blocked for reading or removal.

Project Aware Home [Covington et al., 2001] focuses only on home environments, where a variety of data about home residents and their activities is captured (by sensors), processed and stored. The access control mechanism uses an extension for Role-based Access Control (RBAC) [Sandhu et al., 1996]. Similar to the subject roles of RBAC, the authors defined *environment roles*, which can be used to capture security-relevant aspects of the environment in which an application executes.

In [Smailagic et al., 2001] privacy of location information is described and controlled by simple rules based on set theory. Each rule establishes a list of users who are allowed (or disallowed) to know the location of a user for a given period. The rule specifies the authorizations based on one of four visibility modes: *Visible to All, Invisible to Some, Visible to Some*, and *Invisible to All*, and using boolean operators *AND* or *OR*. When conflicting rules (e.g. R1 grants but R2 denies access), are combined using *AND*, the location information is not made available, and when using *OR*, it is made available to the requester.

Most of the approaches exclusively focus on location context, and propose specific mechanisms for controlling access and disclosure of this information. We however, take a broader approach considering that *any* context data, specially computational context that can be automatically collected and which apparently does not reveal relevant information, should be subject of access control.

Most of the related work adopts a centralized approach for storing and controlling access to context data, even though this surely gives end-users less control of their context information. Only [Hong and Landay, 2004] proposes a fully decentralized approach. Although this seems more reasonable from the end-user perspective, which does not have to trust a centralized context-provisioning infrastructure, it entails some problems when this context information has to be shared (albeit in a controlled way) among many users. Due to the huge amount of resources necessary to store, process and distribute this information, and the intrinsic limitation of mobile devices, it turns out that, at least with our current technology, only centralized approaches are feasible from a software engineering point of view.

Moreover, network use experience of the past decades has shown that despite the real threats of using unknown and remote services, users have largely trusted network infra-structures because of the obvious benefits that they gain. In fact, in most of our daily activities we, *de facto*, rely on social protocols and law enforcement, and expect that other people will indeed obey the rules and follow the social norms.

3. End-User Needs

In order to identify the real expectations, concerns and needs of users with respect to privacy issues, we carried out a survey with people of different age, educational background, field of work/studies, and different familiarity with information and communication technology. The survey was based on a questionnaire which initially described, in non-technical terms, a hypothetical technology supporting the probing, processing and sharing of data about the computational, environmental and personal context of mobile users, and would be aimed at enabling new applications and services for spontaneous communication and collaboration among users. The questions then evaluated following privacy aspects: the degree of willingness/suspicion in using such technology; the groups of people with which the person would share its context data; the demand for anonymity; the desired degree of access control, notification and traceability options, etc. So far, the questionnaire was answered by more than 100 people from Brazil, Italy and Germany.[1]

The most relevant results of our survey indicated the following facts (with the corresponding percentages of votes):

- a cautious attitude toward such technology prevails. For example, 45 % would check the options to selectively disable monitoring of some data, 26 % are seriously concerned about privacy issues. However, only 16 % would refuse the technology;
- users demand for detailed information about which context data was monitored (63 %);
- since mutual discovery and communication is mediated through a yet unfamiliar technology, most users would provide a nickname instead of their real identity (64 %);
- majority of users would only share their context data with people they know, such as friends, colleagues, relatives, (74 %);
- concerning access control users demand: that *a priori* all access be denied, and that they need to explicitly define which access is allowed (76 %),and 66 % want to be explicitly queried for permission at each access request;
- concerning traceability and notifications, most want to be able to check at any moment (at a log) who requested the data, when and through which application (57 %), and some want to be immediately notified at each request (29 %).

Based on the results of this survey, and experience with privacy issues obtained in related work [Hong and Landay, 2004], we identified the following generic requirements for a privacy service:

Flexibility users should be able to define their privacy options with different levels of detail;

Appropriate Control and Feedback users should be notified of, and be able to trace, any attempt to access their context data;

Plausible Deniability in addition to “Grant” and “Deny”, a third option “Not Available” should be allowed as a request result. With this, the requested user can deny access without giving a clue to the requester.

[1]Actually, it is an ongoing survey, since we have not established a final date for the replies. The interested reader can answer the questionnaire on-line [MoCA Team, 2004c].

Precision Control the users should be able to adjust the spacial, temporal precision and the freshness of their context information being disclosed;

Visibility Control at any moment users should be able to block access to any (or all) context data;

Exceptions for Emergencies it should be possible to define an exception policy, which overrides any other privacy policy;

Simplicity users must not be burdened with much work to configure their privacy options;

Efficiency privacy enforcement should not cause significant overhead to the context-provisioning services.

4. MoCA's Context Provisioning Services

The *Mobile Collaboration Architecture* (MoCA) [Sacramento et al., 2004, MoCA Team, 2004b] consists of client and server APIs, basic services supporting context acquisition, storage and processing, and a framework for implementing application proxies (*ProxyFramework*). The APIs and the basic services have been designed to be generic and flexible, so as to be useful for different types of context-aware collaborative applications, e.g. synchronous or asynchronous interaction, message-oriented or artifact-sharing-oriented. MoCA is intended for use in an infra-structured wireless LAN (such as 802.11), and the current version runs on WinXP/CE and is based on TCP/IP.

In MoCA the following are the core services and components responsible for probing, storing and inferring computational and location context.

Monitor is a daemon executing on each mobile device that is in charge of collecting data concerning the device's execution state/connectivity, and sending this data to the *CIS* (*Context Information Service*) executing on one (or more) node(s) of the wired network. The collected data includes the quality of the wireless connection, remaining energy, CPU usage, free memory, current Access Point (AP), list of all APs and their signal strengths that are within the range of the mobile device.

Context Information Service (CIS) is a distributed service where each CIS server receives and processes devices' context data, sent by the corresponding *Monitor*s. It also receives requests for notifications (aka subscriptions) with SQL-like, context-based *interest expressions* from applications, and delivers notifications to the applications whenever the corresponding interest expression matches a new state of the context variables.

Location Inference Service (LIS) infers the approximate *symbolic* location of a device. It does this by comparing the device's current pattern of RF signals received from *CIS* (from all "audible" 802.11 Access Points) with the signal patterns previously measured at pre-defined *Reference Points* in a Building or Campus. For this, LIS periodically queries *CIS* to update the device's pattern of RF signals. Since the RF signal is subject to much variation and interference, the location inference is only approximate: its precision depends on the number of access points and the number of the reference points. *LIS* allows the administrator to define symbolic regions of arbitrary size and shape, and a hierarchical description of regions with its nested sub-regions.

So far, we have used MoCA's API and services to develop some context- and location-aware prototypes of collaborative applications, such as NITA [Gonçalves et al., 2004], WhoAreYou?(WAY), BuddySpaceLive, WirelessMarketingService, and others [MoCA Team, 2004a]. When we demonstrated these applications to

Context Variable	**Privacy Risk**
CPU usage (%)	gives a clue about the device processing load
Free Memory (in kB)	gives a clue whether the user's device is short of resources
Battery Power (%)	gives an estimate for how long the device will be available
IP Addr/Mask	network point of attachment, owner's network access rights, and rough information about device location
Current AP's MAC-Addr, RSSI	connectivity status and rough information about device location
List of all APs in the range	gives a clue about device's approximate location
LIS' Symbolic location	device's approximate location

Table 1: Context data collected by MoCA

other students and faculty, we realized the importance of privacy issues related to context information, i.e. while context data is useful for implementing adaptable and context-aware applications, it can also be used to derive information of where and how a user is using his device. Table 1 shows all computational and location context variables made available by MoCA's services, and some privacy risks of disclosing each such information.

Although our original goal was to develop a privacy mechanism for MoCA's context-provisioning services *CIS* and *LIS*, we later realized that it would be better to design CoPS as an independent and generic service that could be easily integrated with other context-provisioning services.

5. CoPS **Overview**

CoPS is in charge of controlling when, how and to whom context data will be disclosed. This service implements an engine that evaluates privacy policies and checks whether access to context data from one *subject* (i.e. user) will be granted to a specific *requester* (i.e. a user or application). A privacy policy is set up by a *policy maker*, which may or not be the subject himself. By using a policy management GUI, the policy maker specifies the rules that dictates the access restrictions to the subject's context information.

The proposed service implements fine-grain control, feedback and logging mechanisms, which give the subject different means of avoiding abuse of his context information usage. For instance, by setting the notification option in his rules (in addition to the appropriate access control) Bob would probably prevent others, for example his boss, from trying to periodically query his location. The feedback mechanism may use any appropriate form of notification, such as e-mail, SMS, ICQ, etc. In addition, having access to CoPS' log, Bob would be able to check who accessed (or could not access) his context data, when and how many times it occurred, etc. Feedback and logs have also been identified elsewhere [Hong and Landay, 2004] as a simple, yet effective, means of controlling access abuse through social visibility. For example, it is less likely that a boss will repeatedly query an employee's location if he knows that the employee gets notified at every request, and moreover can use the log to prove the abuse, and blame him of this action.

In CoPS, privacy policies are organized in a three-level hierarchy: organization-specific, individual-specific, and default policies. In this hierarchy, the organization-specific policy overrides the individual-specific policy, which in turn, overrides the default one. Hence, for organization-specific policies the policy maker may not be the same as the subject (e.g. the employee). For example, a policy maker responsible for a university may define that the location of each member of a department staff must be made available to the director to facilitate delegation of tasks or finding a nearby member of the network support team to fix a problem of the secretary's desktop.

CoPS also supports two general approaches to specify a default access policy, an optimistic and a pessimistic one. With the pessimistic policy, by default all requests are denied, except those that match some rule specified by the policy maker; and with the optimistic approach, by default all requests are granted, except those matching some rule specified by the policy maker. Thus, using the pessimistic access policy end-users need only define the rules specifying under which circumstances their personal information should be disclosed. In contrast, using the optimistic access policy the end-users should set up rules that explicitly deny access and define side-effect actions (e.g. logging, notifying), related to a specific context variable, (group of) requester, time of the day, etc. For example, the user may define a rule denying access to his location data to any requester which is not affiliated with his department. As a general rule, the optimistic approach is easier to use, because the users can hardly predict all possible scenarios for which he wants to grant access in the pessimistic default policy.

By supporting these two approaches for default policies CoPS gives end-users a convenient, simple and flexible means of defining their rules according to their individual privacy preferences. Unlike other work, [Hong and Landay, 2004, Grudin and Horvitz, 2003], CoPS' dual approach helps to reduce the number of rules necessary to define a privacy policy. After choosing either the pessimistic or the optimistic approach the policy maker will have only to specify a few rules, each of them producing the following results: "Grant" or "Deny" (but not both), "Not Available" or "Ask Me".

The policy maker may use the "Not Available" result when he wants to take advantage of plausible deniability (cf. section 3), since this is also the default result for a request when a context information is in fact not available. Returning "Not Available" as the result of a request thus enables a subject to make "white lies" where he in fact denies access, but does not make this explicit to the requester, who will not know whether the context information could not be obtained due any technical failure, due to access restriction, or lack of the data.

5.1. Typical Interaction Pattern

The Context Privacy Service comprises a server (CoPS) and a client (CoPS Client API). It makes available to the application developer two client APIs, one for the Context Service, and the other for the application clients (at the subject's and requester's side). These APIs provide methods for authentication and session key generation, communication with the CoPS server, for checking the consistency of policy rules, for accessing and analyzing the log, and a protocol used by the CoPS server to send notifications and queries to the subject's application asking the user for a final decision regarding an access request.

Figure 1 illustrates the CoPS general architecture, and shows how its components and the context provisioning service (Context Service) interact with each other. In the case of MoCA, *CIS+LIS* would be the Context Service, and the *Monitor* would be executing on the subject's device.

Initially, the policy maker (e.g. the subject) has to define the privacy policy to be applied to a subject's context data (1): he chooses the default access policy (optimistic or pessimistic) and uses the policy management GUI to write the corresponding privacy rules. In parallel, context data from the subject's device will be periodically received by the Context Service (2) but will only be disclosed upon evaluation of the appropriate privacy policy. Before a requester is able to submit an access request, he has to authenticate

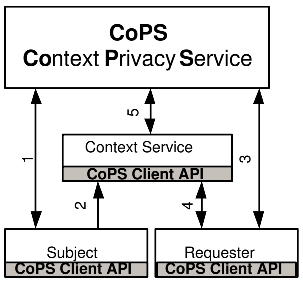

Figure 1: CoPS **general Architecture**

himself with CoPS (3). This authentication will produce a session key which will be used to create an **U**ser **I**dentification **T**oken (UIT) for future requests. The UIT is a ciphered challenge text with the user's session key. The generation and distribution of user session keys will be done by a protocol similar to WPA [Wi-Fi Alliance, 2004]. When the access request arrives at the Context Service (4), it will forward the request and the UIT to CoPS and wait for the result. If the requester is successfully authenticated and the request is granted (5), CoPS replies with a "Grant", otherwise with a "Deny" or "Not Available" result.

5.2. Controllable Properties

According to the results of our survey about requirements for privacy control and related work (see section 3), most users demand means of interactively deciding when requests should be granted or not. In other words, in this approach of interaction, called mixed-initiative, end-users are interrupted and are asked *on the fly* to decide whether to grant or deny the request. If the policy maker set up the privacy rule with the result value "Ask Me", the CoPS server will forward the request to the subject's application. The application may interact with the subject exporting a high level view of the request asking for the Subject's decision, such as: "Can requester A using application Y be granted access to the context information I?". And the answer options could be, for example, "Always allow", "Just this time", "Only for the next 2 hours", "Never Allow", etc. The CoPS server waits some time for the reply, and if no reply is received, CoPS simply denies access to the context information returning the default reply value "Not Available".

CoPS also supports adjusting the precision of the dynamic contextual information being disclosed. It does so by allowing the policy maker to specify a *spatial precision*, *temporal restriction* and *freshness* of the contextual information in the privacy rules. For example, consider a scenario where some service provides location information, Alice is sharing her location with classmates, but maybe is not feeling comfortable letting them know precisely where she is. In this case, she would be able to adjust the level of disclosure by defining the spatial precision of her location information (e.g. "PUC-Rio" or "Department of Computer Science Building" instead of "Room 205"). She could also set some temporal restriction, by defining, for example, the time interval (e.g. "9:00 to 11:30 am AND Monday to Thursday") when the information should be made available. Moreover, she could also specify the freshness of the disclosed information, determining that instead of her current location, only her location 30 minutes ago shall be disclosed.

6. CoPS's Architecture

The service has been designed to offer fine-grained and flexible control over privacy policy evaluation, using the following components: the Privacy Policy Engine, the Dynamic User Management and Access Control (DUMAC), a notification dispatcher and the client APIs. Figure 2 illustrates the main components of the CoPS architecture.

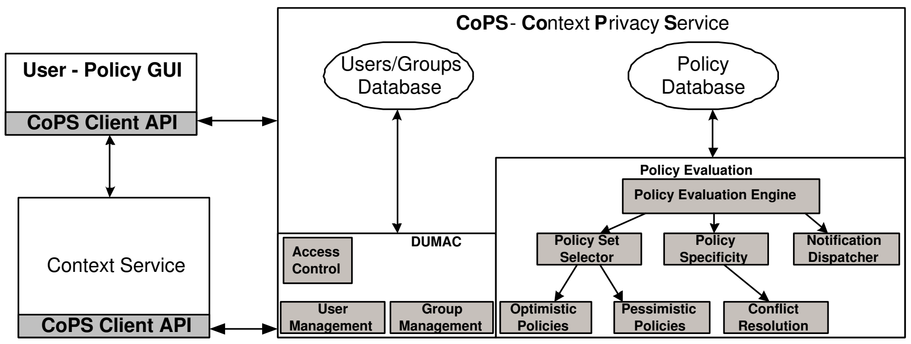

Figure 2: CoPS **Architecture**

As mentioned in Section 5, the client APIs hides from the Context Service and application developer many details related to CoPS-specific interaction and processing. By designing CoPS independently from the Context Service, we obtain more flexibility and reduced complexity. Flexibility, in the sense that CoPS becomes independent of a specific Context Service, and that Privacy Management can be incorporated as an optional and complementary feature of a context provisioning middleware. Thus, the Client API is very important for enabling a simple and transparent integration of CoPS with both the Context Service and the application clients.

In order to provide support to the mixed-initiative interaction, the CoPS server and the application client API use event-based asynchronous communication. The client API subscribes itself at the CoPS server informing the address in which it is supposed to receive requests for the subject's final decision (Grant or Deny). After receiving a request from the CoPS server, the client API forwards it to the application client and waits some time interval to send a reply. If the CoPS server receives a reply from the client API before the timeout, the subject's decision is sent to the context service, otherwise, the default reply ("Not Available") is dispatched.

The Dynamic User Management and Access Control (DUMAC) component is in charge of implementing user authentication and management of groups and users. Although CoPS offers its own authentication method, in principle it can be integrated with any other similar authentication system, such as NIS, SAMBA or Windows Domain Controller, facilitating the deployment in different administrative domains.

The Policy Evaluation is the central component within the CoPS server. It processes the access request taking into account all privacy policies related to a subject. It first selects the rules of the default access policy chosen by the policy maker, and then evaluates policy specificity, by selecting the most specific rules that match a given request. Based on the set of selected rules, it then checks and resolves possible conflicts in order to compute the final result ("Not Available", "Ask Me","Grant" or "Deny"). The result is then returned to the client API at the Context Service.

6.1. Structure of the Privacy Rules

The structure of a CoPS privacy rule is composed of several fields, which are also present in the requests. Any privacy rule is associated with a default access policy (optimistic or pessimistic). This must be chosen in beforehand by the policy maker, and it will determine the basic evaluation algorithm for each request. The proposed rule fields and their semantics are described as follows.

Policy Maker: Individual who defined/created the privacy rule (may or may not be the same as the subject).

Subject: User or entity whose context data is controlled by this rule.

Requester: User or software component requesting access the subject's context data.

Context Variable: The specific type of context data being requested (e.g. location, energy level, IP address, etc.).

Application: List of application names that can be used by the requester to access the context variable. The wildcard '*' represents any application.

Precision: Specifies the value precision of the context variable (e.g. for location information, this could be the spatial precision like state, city, ZIP code, building, room, etc.).

Temporal Restriction: Date and time interval restrictions for disclosing the context information (e.g., weekdays, from 9 am to 6 pm).

Freshness: Specifies the freshness (in milliseconds) of the disclosed context information (e.g. location 15 minutes ago, or current location). The default value is 0 ms.

Policy Level: Hierarchy level of this rule. Initially, CoPS will support only following three possible values "organization", "individual" or "default".

Result: Outcome of applying this rule to a request. Possible values: "Not Available", "Ask Me", "Grant" and "Deny".

Notify Me: If the policy maker wants to be notified when the rule is applied. The options available are "NoNotification", "E-Mail", "ICQ", "MSN" or "SMS".

6.2. Group Definitions

Groups provide an additional facility for the management of privacy rules and also decrease the processing effort during evaluation of the requests. The *Subject* or *Requester* field of a privacy rule can be either individual users or groups.

There are two general categories of groups: *administrator* and *user-defined* groups. The first ones are structured hierarchically to reflect the organizational structure, and define the corresponding user roles, similar to RBAC [Sandhu et al., 1996]. Groups in a higher level of the hierarchy include all of its descendant groups at a lower level, e.g. the group "puc.employee" comprises the group "puc.employee.prof", which in turn comprises the group "puc.employee.prof.cs". *User-defined* groups, are not hierarchical for the sake of efficient evaluation and maintenance.

Initially, all users in CoPS belong to group "Anonymous", which facilitates the specification of access rules for unknown users, i.e. the policy maker is able to set up a privacy policy for unknown ("Anonymous") requesters. Moreover, this group can also be used for anonymity, i.e. users can send a request as an "anonymous user" if they want to hide their real identity.

6.3. Policy specificity

During the evaluation process, more than one rule may match the request, for many reasons. For instance, when the requester belongs to several groups mentioned in field "Requester" in some rules (e.g. "Alice" belongs to groups "MyCo-worker" and "MyFriend"), then all these rules match the request. CoPS's specificity algorithm aims to determine the *most specific* privacy rule that applies to a request and, if necessary, resolve possible conflicts among the rules.

The specificity algorithm works as follows: Given a set of rules previously selected (by the engine) to evaluate a request, the algorithm identifies the most specific rule of the set by comparing their structure fields in the following order of priority: *Subject*, *Requester*, *Application*, *Temporal Restriction*, *Precision* and *Result*. When comparing rules with respect to a field, only the ones with the most specific value in this field are selected for the further specificity analysis, while all other rules are not considered for selection. This way, even if two or more rules have different relative specificity (i.e. they differ in two or more fields) the algorithm can identify the most specific rule analyzing these fields according to their priorities. For all fields, wildcard “*” means least specific.

For the specificity of the *Subject* and *Requester* fields, privacy rules mentioning an individual user (e.g. “Alice”) are more specific than rules containing a user-defined group (e.g. “MyFriend”), which in turn is more specific than the ones mentioning an administrator-defined group. The administrator-defined group specificity follows the usual interpretation of a hierarchy: groups at a lower hierarchy level are more specific than groups at a higher level (e.g. “puc.employee.prof.cs” is more specific than group “puc.employee.prof”). With regard to field *Application*, specificity has only two possible levels: any application (represented by “*”) and a list of applications.

The same hierarchy-induced specificity applied to the administrator-defined group is used also for the *Precision* field[2]. For example, when comparing rules concerning location information, the most specific ones are those where field *Precision* mentions the lowest level in the location-hierarchy, e.g. “country.state.city.zip” (level 4) is more specific than “country.state.city” (level 3). Two or more privacy rules can be at the highest level of specificity with regard to their *Precision* field if they have the most specific value, and are at the same level in the hierarchy. When this happens, the next field (according to the priority) of these rules is compared to identify the most specific rule. In order to allow for such specificity analysis the developer of the Context Service has to define the syntax (e.g. campus.building.floor.room) of the name hierarchy for this specific field. It will be a configuration parameter of CoPS.

The field *Temporal Restriction* represents the time interval and date at which the requester is granted or denied access to the context information, depending on the access policy approach used (optimistic or pessimistic). This field is very useful when the user wants to restrict the access in some special situations (e.g. at lunchtime or at working hours). Even though the policy maker specifies a time interval (e.g. “9:00am-11:30am”), CoPS represents it in seconds, to allow for an accurate rule selection. The specificity for this field is evaluated in three phases: (1) select the rule(s) that match the time and date of the request; (2) identify the rule with the largest time interval and check whether the time interval of the other rules are its proper subsets (e.g. Temporal Restriction “Feb 5, 10:30am-2:00pm” is a proper subset of restriction “Feb 5, 10:00am-6:00pm”). Rules are considered to be at the same level of specificity either if they have identical time intervals, or if the time interval is not a proper subset of the largest time interval; (3) select the rule with the smallest time interval, when they are not at the same level of specificity.

Finally, if all previously considered fields are at the same level of specificity, the *Result* field is the one used to select the most specific rule to evaluate the request. The possible values for this field are: “Not Available”, “Ask Me” and “Grant” (or “Deny”). The “Not Available” result has precedence over “Ask Me”, which in turn has precedence over the others (i.e. result “Not Available” is more specific than “Ask Me”, which in turn is more specific than “Grant” and “Deny”). The reason is that “Not Available” implicitly means “Deny”and “don’t let requester know it”, while “Ask Me” may be interpreted as “Deny” or “Grant”, depending on my mood. A conflict is detected when there is more than one rule with a result “Not Available” or “Ask Me”, or when all rules have either a “Grant” or “Deny” result. In this case, the last rule with greatest specificity created by the policy maker will be selected. It is necessary to define a deterministic choice for these situations because the conflicting rules may have different notification methods and only

[2]Although a hierarchy-induced notion of precision is more easily understood in terms of location information, it can be applied also to other context information, such as sub-domains of an IP address.

Rules	Subject	Requester	Application	Temporal Restriction	Precision	Result	Context Variable	Freshness	Policy Level	Notify Me
R1	Puc. Student	Puc.Manager	Ap1	*	puc	G	Location	0	O	e-mail
R2	Bob	Puc.Student	*	9:00am to 6:00pm	*	G	Energy	5	U	ICQ
R3	Bob	MyFriend	*	9:30am to 12:30am	*	G	Energy	0	U	ICQ
R4	Bob	Coworker	*	12:00am to 2:00pm	*	NA	Energy	0	U	NoNotify
R5	Bob	Coworker	*	9:00am to 12:00am	*	G	Location	0	U	NoNotify
R6	Bob	Alice	*	9:00am to 11:00am	campus. building	G	Location	0	U	MSN
R7	Bob	Alice	*	10:00am to 4:00pm	campus. building. floor.room	G	Location	15	U	e-mail

Table 2: Example rules.

Assumptions		
	Group	**Members**
user-defined	Bob.MyFriend	Bob, Alice, John
	Bob.Coworker	Alice, Jane, John
administrator-defined	Puc.Student	Bob, Alice, Jane, John
	Puc.Manager	Jane, Paul

Table 3: Assumptions about User Groups

a single rule must be chosen to evaluate the request.

6.4. Privacy Policy Evaluation Example

In this section, we show an example of possible privacy rules for user Bob, assuming that the pessimistic default policy has been chosen, i.e. whenever a request does not match any rule, it will be denied. These rules (shown in Table 2) determine how and when Bob's location and energy context variable will be disclosed. In this example, we also assume the existence of some *user*- and *administrator-defined* groups (Bob's and PUC's groups are shown in Table 3), which are mentioned in some of the rules.

Through some scenarios, we will now explain how the privacy rules are selected and used to evaluate a request, using the algorithm explained in Section 6.3.

As already mentioned, the rule to be applied to the request is always the most specific one, and comparison of the rule's specificity takes into account fields *Subject*, *Requester*, *Application*, *Temporal Restriction*, *Precision* and *Result*, in this order. Thus, intuitively, the algorithm compares the values in the corresponding columns (from left to right), and as soon as one (or several) rules have a more specific value in one of the columns, they are candidate for further comparison.

Scenario1: If Jane makes a request for Bob's location, both R1 and R5 would apply. However, the request would be granted by R1, because this rule belongs to a higher level than rule R5 and, consequently, the first rule overrides the others.

Scenario2: Consider a request from John to get the energy level of Bob's device. In this case, R2, R3 and R4 are the related rules. But among those, rules R3 and R4 are selected because the user-defined groups mentioned in these rules are more specific than the administrator-defined group of R2. Finally, the request will be evaluated by R4 because, despite their fields *Requester*, *Application*, *Temporal Restriction* and *Precision* having the same level of specificity, their *Result* value differs, and "Not Available" has precedence over "Granted".

Scenario3: For Alice's request to get Bob's location rules R5, R6 and R7 should be examined. Among those, R6 and R7 take precedence over R5 because they apply to an individual user, "Alice", rather than to a group, as specified by R5. Although the R6 and

R7 are at the same level of specificity in the *Application* and *Temporal Restriction* fields, R7 is more specific than R6 in the *Precision* field, and therefore will be applied to grant the request.

7. Future Work

We intend to extend CoPS's engine to handle context-dependent privacy policies, allowing the policy maker to set privacy rules which depend on dynamic context data. For example, a policy could specify that access to some context data is granted only when the requester is within the university campus, or even in a specific building. In addition, we plan to develop a privacy policy management GUI, which supports end-users to define their privacy rules. This GUI ought to be simple and effective to motivate end-user to adopt CoPS.

Furthermore, we are studying the "Platform for Privacy Preferences (P3P)" specification [Cranor et al., 2002] with the purposes of using it to represent the privacy policy structure. P3P supports the encoding of privacy policies into machine-readable XML, making it easier to interpret these policies and execute the corresponding actions.

We also intend to develop a trust model [Wagealla et al., 2003] for context-aware computing, exploring some properties of trust evaluation (e.g. diversity, transitivity, and combination), in order to facilitate the definition of privacy policies. For example, assuming the transitive property of trust we could have the following scenario "if Alice trusts Bob who trusts Jane, then Alice will also trust Jane". This way, Alice would not need to explicitly set up privacy rules to handle Jane's request. Instead, the system could be able to infer the Alice's risk level of disclosing her information to Jane, and it would be able to apply the appropriate privacy policy.

8. Conclusion

Since context-awareness has been recognized as a key element for the development of adaptive applications in mobile environments, many efforts have been made to design and implement context-provisioning middleware infrastructures. We have implemented such a middleware, called MoCA which we are now using to implement context- and location-aware applications for mobile and spontaneous collaboration.

Results of a recent end-user survey, where we assessed the acceptance of such applications and privacy concerns, helped us to identify the main requirements for a context privacy service. We then designed the Context Privacy Service (CoPS), trying to address all these requirements.

CoPS is intended as an optional, generic service to enforce the controlled access to context information. Prior to releasing any context information requested by a user or application, CoPS would be queried to decide if access to a subject's context is granted or denied. One of the most interesting feature of this service is its support for a rich set of options for privacy policies, such as user and organization-level rules, both optimistic and pessimistic default privacy rules, group-based rules and group management, specificity analysis considering the subject, the requester, spatial and temporal restrictions, information freshness, as well as the allowed applications. We have implemented a first prototype of CoPS which we make available (as an optional service) in MoCA.

References

Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

Covington, M. J., Long, W., Srinivasan, S., Dev, A. K., Ahamad, M., and Abowd, G. D. (2001). Securing context-aware applications using environment roles. In *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 10–20. ACM Press.

Cranor, L., Langheinrich, M., Marchiori, M., Presler-Marshall, M., and Reagle, J. (2002). Platform for privacy preferences 1.0 (p3p) specification. W3C Recomendation, HTML version at http://www.w3.org/TR/P3P/ (Last visited December 2004).

Dey, A., Salber, D., and Abowd, G. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications.

Gonçalves, K., Rubinsztejn, H., Endler, M., Santana, B., and Barbosa, S. (2004). Um aplicativo para comunicacão baseada em localização. In *VI Workshop de Comunicação sem Fio e Computação Móvel*, pages 224–231.

Grudin, J. and Horvitz, E. (2003). Presenting choices in context: approaches to information sharing. In *Workshop on Ubicomp communities: Privacy as Boundary Negotiation*.

Hong, J. I. and Landay, J. A. (2004). An architecture for privacy-sensitive ubiquitous computing. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 177–189. ACM Press.

MoCA Team (2004a). Moca applications home page. http://www.lac.inf.puc-rio.br/moca/applications.html (Last visited December 2004).

MoCA Team (2004b). Moca home page. http://www.lac.inf.puc-rio.br/moca (Last visited December 2004).

MoCA Team (2004c). Questionnaire about privacy and spontaneous collaboration. http://cis.lac.inf.puc-rio.br:8080/lac/questionnaire.jsp (Last visited December 2004).

Sacramento, V., Endler, M., Rubinsztejn, H. K., Lima, L. S., Goncalves, K., Nascimento, F. N., and Bueno, G. A. (2004). Moca: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*, 5(10):2.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, 29(2):38–47.

Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US.

Schiller, J. and Voisard, A. (2004). *Location-Based Services*. Morgan Kaufmann.

Schmidt, A., Beigl, M., and Gellersen, H.-W. (1999). There is more to context than location. *Computers and Graphics*, 23(6):893–901.

Smailagic, A., Siewiorek, D. P., Anhalt, J., Kogan, D., and Wang, Y. (2001). Location sensing and privacy in a context aware computing environment. In *Pervasive Computing*.

Wagealla, W., Terzis, S., and English, C. (2003). Trust-based model for privacy control in context aware systems. In *Second Workshop on Security in Ubiquitous Computing at the Fifth Annual Conference on Ubiquitous Computing (UbiComp2003)*.

Wi-Fi Alliance (2004). Wi-Fi protected access web page. http://www.wi-fi.org/OpenSection/protected_access_archive.asp (Last visited December 2004).