Privacy-aware Context Discovery for Next Generation Mobile Services

Cristian Hesselman, Henk Eertink, and Martin Wibbels
Telematica Instituut, The Netherlands
{Cristian.Hesselman, Henk.Eertink, Martin.Wibbels}@telin.nl

Abstract

We present a system that enables applications to discover and obtain information that describes the context of a particular entity (e.g., a user or a device). Our system revolves around the notion of a context agent, which is a service that represents an entity and provides access to context information about that entity. Context agents facilitate the enforcement of an entity's policies regarding the release of context information (e.g., to applications or visiting users), even while these entities roam across different administrative domains. Context agents form an overlay on top of traditional local area service discovery infrastructures (e.g., based on SLP or WS-Discovery) and are enablers for more intelligent pervasive computing environments. In this extended abstract, we outline the architecture of our system based on a simple scenario.

1. Introduction

Context-awareness is a key function of systems that operate in pervasive computing environments. It is made possible by recent innovations in sensor technology, wireless communications, and personal devices. Context-aware systems are able to dynamically adapt the delivery of a service, for instance to the environment of a service user or to the activities that user is currently engaged in.

In this extended abstract, we consider the problem of discovering and obtaining context information about entities (e.g., users or devices) that roam across intelligent environments in different administrative domains. By an intelligent environment we mean a place that provides context information (e.g., about itself) and also collects it (e.g., about users in the place). We illustrate this problem by means of a simple scenario in which a user Bob carries a smart phone and enters an intelligent environment in a foreign domain. The objective of our system is to enhance the information that describes Bob's context with context information that the foreign environment gathered about Bob (e.g., using temperature sensors in the foreign environment). An important requirement is that the system should enforce the policies of Bob and those of the foreign domain regarding the release of context information. Bob should furthermore be able to remain relatively anonymous in the foreign domain.

To accomplish this, we introduce the notion of context agents. These discoverable software entities provide a single point of access to the (aggregated) context information about a particular entity (e.g., a user or a device), including context information gathered by a foreign domain. A context agent can be located through an identity such as bob@domain. In the scenario, we assume that Bob's friend Alice wants to obtain context information about Bob and that she is aware of his identity.

We show how context agents play an important role in the interface between 'raw' sources of context information (e.g., GPS devices and temperature sensors) and context-aware applications and services. From a privacy perspective, a context agent operates as a policy enforcement point on behalf of the entity that it represents.

The rest of this extended abstract is organized as follows. First, we introduce the basic features of our Context Management Framework (CMF) [4], which is a distributed system on which context agents build. We illustrate some of the shortcomings of the CMF in inter-domain scenarios. Next, we describe our context agent extension, which adds the controlled exchange of context information. This work is an extension of the work presented in [3] and fits within the service platform concepts introduced in [6].

2. Context Management Framework

The CMF [4] consists of different types of services, instances of which may be distributed across different administrative domains. The main types of services in the CMF are context sources and context brokers. A *context source* stores context information and makes it available to applications. Applications access a context source through a well-defined interface, which provides access to the context information stored by the context source (both synchronously and asynchronously). Context sources can wrap physical sensors or entire sensor networks, but they can also aggregate context information from other context sources or reason about that information to infer new context information (bottom-up inference).

A *context broker* enables applications or other context sources to discover context sources that can provide a particular type of context information. A context source registers with a single context broker, which maintains a directory of registered context sources. A context broker is part of the same domain as the context sources that register with it and may reside on the same (mobile) device. The latter will enable the context broker to manage the lifecycle of its registered context sources. A context broker is discoverable in its own network, using local service discovery protocols like SLP or WS-Discovery.

The CMF is very flexible in terms of distribution properties (it can operate in peer-to-peer and centralized scenarios) and in terms of context reasoning (it is easy to add new types of sensors or new reasoning engines), but has two limitations. First, it does not support inter-domain discovery of context sources. As a result, applications may need to query many context brokers in many different domains to find the context sources that provide context information about a particular entity (e.g., a user). This puts a serious burden on applications and thus does not scale well. This problem is worsened by the dynamic nature of pervasive computing environments, which means that the set of relevant context source changes frequently (e.g., as a result of roaming). The second limitation is that context brokers are privacy-unaware, which makes them unsuitable for any realistic pervasive computing environment as is. We solve these problems by adding a context agent layer on top of the CMF.

3. Context Agents

A context agent is a discoverable service that represents an entity and maintains references to context sources that can provide context information about that entity. The entities we consider may be mobile and may be physically part of an intelligent environment that is not trusted (e.g., a wall-mounted display at a customer's premises). In this abstract, we focus on context agents for users and devices, but in general we also distinguish places and services (cf. [1, 2]). Observe that a context agent can be realized in various ways, for instance as a web service [1].

A context agent has at least one identity (e.g., bob@domain for a user or building@campus.nl for a place). This is also the main difference between a context agent and a context broker: a context agent is tightly bound to an identity (and to synonyms of that identity), whereas a context broker is unaware of identities and can form a rather arbitrary grouping of context sources (e.g., the context sources available on a particular device or in a particular domain).

We assume that applications use a well-known name resolution infrastructure to resolve an identity into a reference to the entity's context agent. We currently use SIP, but a DNS-resolvable service could be used as well. Actual name resolution mechanisms are however outside the scope of our work.

Given a reference to the proper context agent, an application can *query* that context agent for certain types of context information (e.g., location information). The context agent responds to such a query with a reference to a *proxy context source*, which sits between the application and the context source that can provide the actual context information. The main responsibility of a proxy context source is to enforce the privacy policies set by the owner of the context agent (see Section 4). A proxy context source also helps to reduce the resources that mobile nodes need to host context sources. In this case, a proxy caches context information of the actual context source and performs subscription management (the proxy sends out an event whenever the context changes).

4. Privacy

One of the responsibilities of a context agent is to enforce the privacy of its owner. By *privacy policies* we mean the rules that define when an entity (e.g., a user) whishes to release context information about itself to applications or other entities. An example of a privacy policy is "if I'm not at work, then only my family members can see my location".

The policies in our system allow users to specify the conditions under which context information may be released and to whom (e.g., based on the time of day), which types of context information can be released, and what the 'quality' of that information should be (e.g., location information in the form of GPS coordinates, a street name, or GSM cell IDs).

Context agents enforce privacy policies in two steps. First, a context agent checks its privacy policies against queries from applications for certain types of

context information (see Section 3). For that purpose, the application has to present some proof that indicates that it is authorized to access this information (we will use SAML for that, which will allow us to integrate the CMF with identity management solutions).
If the application is authorized, the context agent creates a proxy context source (see Section 3) that will enforce the entity's privacy policies. The context agent configures the proxy to make sure that it returns context information at a suitable quality level (accuracy, timeliness) [5] when the application calls the proxy.

Figure 1 illustrates this behavior in case a client application on Alice's machine wants to get Bob's location. To accomplish this, the application sends a query message to Bob's context agent. Bob's context agent knows that Bob owns a smart phone and therefore forwards this requests to the context agent of the smart phone ("CA(smartPhone)"). The smart phone's agent checks whether Alice is allowed to get Bob's location, creates a proxy for the location context source, and returns the reference to the proxy back to the application. Alice's application can subsequently retrieve/subscribe to context information using this proxy. Observe that the ownership relationship between the two context agents is pre-configured using domain configuration management tools.

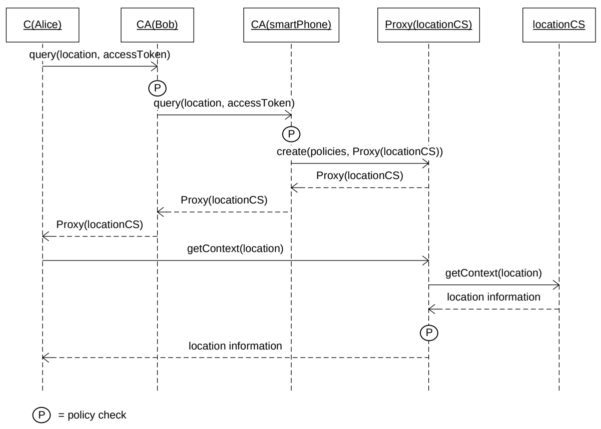

Figure 1. Policy enforcement, single domain.

The P's in Figure 1 mark the places where Bob's policies are enforced. The final policy check in the proxy may change the location information returned to the application to enforce the privacy policies of Bob's mobile device (e.g., return a street name instead of the GPS coordinates provided by the context source).

The scenario in Figure 1 focuses on single domain aspects (the context sources are part of the same network as the context agent). In the next section, we will illustrate our roaming concepts.

5. Privacy-aware Roaming

Pervasive computing environments typically enable users to make use of services, devices, and information existing in their local environment (e.g., in a smart room). The objective of our system is to enhance the information that describes an entity's context with context information that the local environment has gathered about that entity (e.g., using temperature and location context sources), in particular when the entity resides in a foreign domain. An important requirement to accomplish this is that our system enforces both the entity's privacy policies as well as those of the foreign domain.

In our approach, the context agent of an entity runs in that entity's home domain (e.g., a residential network or an operator's network) and continues to be responsible for enforcing the entity's privacy policies. It also continues to be the single point of access for context information about the entity, even when the entity has roamed to a foreign domain.

To be able to publish context information gathered by a foreign domain, an entity's context agent links to a *Temporary Context Agent* (TCA). A TCA represents an entity when it visits a foreign domain and enforces the privacy policies of that domain. A TCA runs in the foreign domain and links to context sources in that domain that can provide context information about the entity. Observe that our framework requires trusted relationships between context sources and context agents, which means that mobile nodes cannot directly access context sources in foreign domains.

Figure 2 shows an example in which Bob resides in a foreign domain F. The context agent of Bob's smart phone resides in Bob's home domain and links to a TCA in domain F. This TCA represents the smart phone in F as long as Bob resides in F. As we will see below, the context agent uses the TCA to provide information about the ambient light level around Bob to Alice's client application.

Figure 2 also shows the main components of our system. When Bob's smart phone enters an intelligent environment in F, the Context Agent Discovery Client (CADC) on the phone discovers the Context Agent Manager (CAM) of F (1). A CAM is a component that configures all context agents in a particular domain, including the TCAs. The CADC requests the CAM to create a TCA for the smart phone (2), after which the CAM configures the new TCA with a few context sources that are part of F (3). The CAM then returns a reference to the TCA to the CADC (4).

The CADC subsequently links the TCA with its own context agent in the home domain (5). This association will make it possible for clients to obtain context information about Bob's smart phone gathered by foreign domain F. From a security perspective, the CAM will give the CADC temporary credentials that allow a component with those credentials to access the TCA created for Bob's smart phone. The CADC will register these credentials with the smart phone's context agent in Bob's home domain (5) so that the context agent can use the TCA.

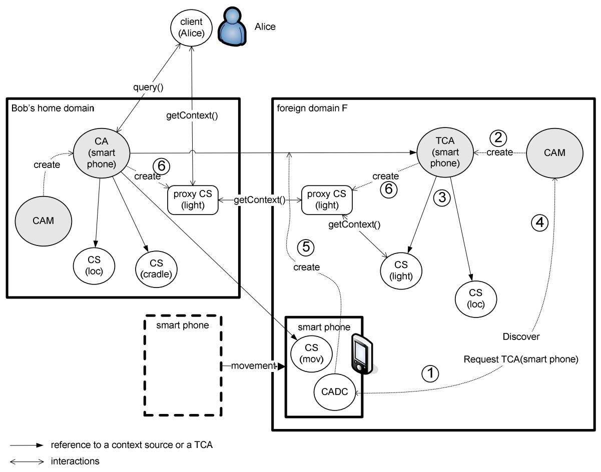

Figure 2. Policy enforcement, inter-domain.

The client application on Alice's machine obtains information about the ambient light level at Bob's current location by sending a query to the context agent of Bob's smart phone (see Section 3). This context agent forwards that request to the TCA in F, which creates (6) a proxy context source (see Section 3). The proxy enforces the privacy policies of its owner, domain F. The context agent of the smart phone creates another proxy, which enforces Bob's privacy policies. The information that Alice's client requests will flow from the light context source in F back to the client through both proxies, thus ensuring that both Bob's and F's policies will be enforced.

From a privacy perspective, it is important to note that the identity of Bob is not revealed to F. The 'only' requirement that we have is that Bob needs to have access to F's network. Bob may use a completely different identity (e.g., using a scratch card) to access this network, and remain relatively private in his current environment.

6. Summary

In this extended abstract, we described a context management infrastructure that supports two layers: a context agent layer that gives applications and services easy-to-use access to context information and a context source layer that provides features for interpretation and inference of context information. We also showed how this infrastructure obtains contextual information and that it preserves privacy in scenarios that involve unrestricted mobility in combination with ubiquitous networking. Our framework is a natural extension of service platforms for home, enterprise, operator, and hotspot environments.

The work we described in this extended abstract is ongoing. We are currently implementing it on top of the CMF.

Acknowledgements

This work is part of the Freeband AWARENESS project. Freeband is cosponsored by the Dutch government under contract BSIK 03025. In particular we thank our colleagues Remco Poortinga, Andrew Tokmakoff and Niels Snoeck for the fruitful discussions on the topics presented in this paper.

References

[1] P. Debaty and D. Caswell, "Uniform Web presence architecture for people, places, and things", *IEEE Personal Communications*, Aug. 2001, pp. 46-51

[2] A. Dey, D. Salber, and G. Abowd, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications", *Special issue on context-aware computing; Human-Computer Interaction (HCI) Journal*, 2001, pp. 97-166

[3] C. Hesselman, A. Tokmakoff, P. Pawar, and S. Iacob, "Discovery and Composition of Services for Context-Aware Systems", *1st European Conference on Smart Sensing and Context (EuroSCC'06)*, Enschede, The Netherlands, October 2006

[4] H. van Kranenburg, M. S. Bargh, S. Iacob, A. Peddemors, "A Context Management Framework for Supporting Context-Aware Distributed Applications", *IEEE Communications Magazine*, Aug. 2006

[5] K. Sheikh, M. Wegdam, M. van Sinderen, "Enforcement of Dynamic Privacy Policies in Distributed Context-Aware Systems", *Adjunct Proceedings of the 4th International Conference on Pervasive Computing*, Dublin, Ireland, May 2006

[6] M.J. van Sinderen, A.T. van Halteren, M. Wegdam, H. Meeuwissen, E.H. Eertink, "Supporting Context-Aware Mobile Applications: An Infrastructure Approach", *IEEE Communications Magazine*, September 2006