Controlled Disclosure of Context Information across Ubiquitous Computing Domains

Cristian Hesselman, Henk Eertink, Martin Wibbels
Telematica Instituut
The Netherlands
first.lastName@telin.nl

Kamran Sheikh
University of Twente
The Netherlands
k.sheikh@ewi.utwente.nl

Andrew Tokmakoff
Philips Research
The Netherlands
andrew.tokmakoff @philips.com

Abstract

One of the challenges in ubiquitous computing is that of mobility, which typically requires interaction between intelligent environments in different domains of administration. We present a highly distributed and collaborative system that enables context-aware applications to obtain context information about mobile entities (users or devices) independent of the domain that produces this information. The added value of the system is that it enriches the amount of available context information about these entities in a way that is transparent for applications. In addition, the system shares context information across domains in a controlled manner by taking privacy policies into account, both of the mobile entity as well as of the domains it visits. We discuss the system's architecture, its implementation, and the way we deployed it.

1. Introduction

A critical component in ubiquitous computing is a system that enables applications to obtain (enriched) context information (e.g., [6],[7],[15]) about "entities", typically users, places, and devices. Examples of context information include the activity, mood, and heart rate of a user, as well as a device's system level state (e.g., CPU usage).

A known problem of most existing systems is that their operation is constrained to a single administrative domain [1]. As a result, applications cannot get context information generated by sensors in remote domains. For example, when a user is in a hotel that tracks the location of its guests, this information will be unavailable to that user's personal applications as well to applications of others (e.g., the user's buddies). In such cases, these applications will at best only be able to obtain context information provided by the sensors of the user's personal device (e.g., GSM cell IDs).

In this paper, we address this problem and present a novel context management system that (1) enables applications to obtain context information on roaming entities from remote domains and (2) at the same time enforces privacy policies regarding the release of context information, both of the roaming entity as well as of the remote domain. The latter is a crucial requirement for the deployment of ubiquitous computing systems in any realistic scenario [5].

Our work focuses on interactions between relatively small domains (e.g., homes and office environments) that are not necessarily federated through roaming agreements. This is unlike the more telecom-oriented approach discussed in [18]. The work described in this paper is an extension and elaboration of [11].

In this paper we first illustrate our ambition with a mobility scenario (Section 2). We then discuss the architecture of our system, first concentrating on intra-domain aspects (Section 3) and then on inter-domain context sharing (Section 4). Next, we present the implementation of our system and describe how we deployed it (Section 5). Finally, we compare our work to existing systems and present our conclusions and future work in sections 6 and 7.

2. Scenario

Figure 1 shows a scenario in which a mobile user visits a remote domain. The goal of the scenario is to introduce the roles of the involved domains, a typical context-aware application, and the policies that control the exchange of context information.

Alice@CompanyA is visiting John in his office @CompanyB. The third person in the scenario is Bob, a co-worker of Alice. He is in his office at Company A and is unaware of Alice's appointment with John.

Building B is able to detect information about users, objects and places within building B. This information is made available via local services, which we call

context sources. In the example of Figure 1, building B contains a location context source that tracks the location of the people in building B (e.g., using Bluetooth or WLAN sensors) and an activity context service that determines what activity they are currently engaged in. The activity source may infer the activity of a person by combining information from several other sources, for instance by combining Alice's location and the appointments in John's electronic calendar.

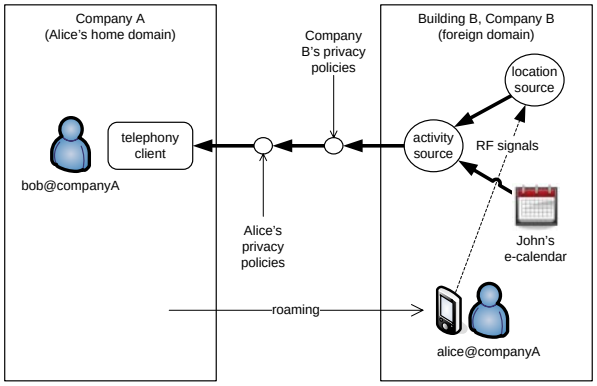

Figure 1. Scenario.

Now suppose Bob opens his context-aware telephony application to call Alice and invite her to lunch. In the scenario, the application indicates that Alice is in a meeting until 11am, which is information that originates from the activity source in building B (foreign domain). This context source knows that Alice is in a meeting with John at Company B until 11am, but only sends part of this information to Bob as a result of two policies: First, Company B's privacy policies say that it only releases activity information to the outside world when the information does not pertain to employees of Company B. Hence, Alice's activity information will not include a reference to John. Second, Alice's policies could indicate that co-workers like Bob should not see her current location. As a result, the information about Alice's current activities will be downgraded again before being passed to Bob's telephony application, i.e. to "John is in a meeting until 11am".

The main question we address in this paper is how we can share context information like Alice's activity across administrative domains, while at the same time enforcing the privacy policies of the roaming entity and those of the visited (or "foreign") domains.

3. Intra-domain Context Management

We designed a distributed system called the Context Management Framework (CMF), which consists of several services that together collect, share, interpret, and manage context information about entities. The two key services of the CMF are context sources (Section 3.1) and context agents (Section 3.2). In Section 4, we consider how the CMF shares context information across domains. Context sources and agents have been reported on before in [11] and [15].

3.1 Context Sources

A context source is a service that computes or senses context information and makes this available to applications through a well-defined interface, both synchronously and asynchronously. Context sources may wrap physical sensors (e.g., a Bluetooth dongle or a temperature sensor) or entire sensor networks. They can also aggregate context information from other sources or reason about it to infer new context information. Context sources use a context ontology to describe the types of context information they can provide and the quality levels they support (our quality of context mechanism is outlined in [20]).

The example of Figure 1 contains three context sources operated by the foreign domain (Company B): a location context source, a context source that provides calendar information and a context source that infers activity information from the other two.

Context sources have a trust relationship with the other context sources in their domain (e.g., by imprinting with a shared secret, or using certificates). Context sinks use a stateful discovery mechanism to find context sources, the core component of which is a so-called "context broker". There is at least one context broker per domain, but usually all devices that support the CMF run a context broker and the context brokers exchange information within their domain using a stateless discovery protocol (our current implementation supports SLP [10] and WS-Discovery [19]). We refer to [15] for more details on the interaction between context sources and context brokers.

3.2 Context Agents

A context agent is a discoverable service that is bound to a specific entity (e.g., a user such as Alice) and provides a one-stop shop for applications to obtain context information about that entity. As we will see in Section 4, this includes information gathered by context sources in foreign domains.

In general, a context agent keeps track of the set of context sources that can currently provide context information about the entity that the agent represents (the agent's set of registered context sources). A

domain will therefore publish a context agent for each entity (e.g., devices, users, and places) about which it wants to provide context information.

In our work, each context agent has an identity of the form entityID@domainID (e.g., alice@companyA). Applications such as Bob's telephony application locate a context agent by resolving its identity to a network address, for instance using a name resolution infrastructure. The CMF supports both DNS and SIP based name resolution.

Since a context agent is the single point of access for context information about a particular entity, it is also a natural place for enforcing that entity's privacy policies (cf. [6]). Privacy policies are rules that specify who is allowed to get what context information in what quality in what situation. To enforce such policies, the CMF follows the Policy Core Information Model [16], which consists of Policy Decision Points (PDPs) and a Policy Enforcement Points (PEPs). A PDP is responsible for evaluating policies (privacy policies in our case), while a PEP requests a policy decision from a PDP and enforces the decision that the PDP makes.

The PEPs in our architecture are context agents and proxy context sources. A context agent controls which types of context information an application is allowed to get, while a proxy context source subsequently ensures that the application gets this information at the appropriate Quality of Context (QoC) [2][20]. To accomplish this, a proxy context source wraps the source that provides the "real" context information. Our privacy policies determine the maximum QoC that a proxy passes to the requester of certain context information (e.g., Bob in Figure 1) is allowed to get. Examples of QoC indicators are precision [20], freshness and probability of correctness.

We assume that context agent and proxy context sources have their own logical PDP, but we do not make any assumptions on how they should be realized (e.g., as truly separate PDPs or as a single domain-wide PDP for multiple agents and sources).

A PDP communicates the result of its policy evaluations to a PEP in the form of policy decisions. This includes an "obligations" part [8], which specifies the maximum level of QoC. A policy decision depends on three inputs: (1) the identity of the party requesting the context information (e.g., bob@companyA), (2) the identity of the entity whose context information the requestor wants to obtain (e.g., alice@companyA), and (3) the type of context information being requested (e.g., activity information).

Each domain contains a *Context Agent Manager* (CAM), which is responsible for creating, configuring, and destroying context agents in that domain. A CAM configures a new context agent with the privacy policies of the entity that the agent represents and obtains these policies from a policy repository in its local domain. A CAM also registers a set of 'well-known' context sources with each new agent. These are sources in the CAM's local domain, typically building-specific aggregators of context information. At a later stage, a context agent may use locally available context brokers (Section 3.1) to dynamically locate additional context sources in the CAM's local domain. A CAM must be discoverable through service discovery protocols like SLP or WS-Discovery.

4. Inter-domain Context Sharing

For inter-domain context sharing, we focus on "friendly" foreign domains that (1) allow visitors to use the foreign domain's network and (2) that are willing to share at least part of the information they collect about visitors with the outside world.

Our prime mechanism for supporting inter-domain context sharing is the so-called "temporary context agent" (TCA). A TCA represents an entity when it resides in a foreign domain and forms the entry point for getting context information from the foreign domain about that entity. A TCA links to the entity's context agent in the home domain to make the foreign context information available to applications and enforces the foreign domain's privacy policies (with the assistance of proxy context sources in the foreign domain). The CAM of the foreign domain manages the life-cycle of TCAs, which are softstate components. The latter means that a CAM will destroy TCAs automatically if they are not kept alive. This frees up resources and therefore benefits the scalability of the entire system. It may also help in foreign domains deleting information they have collected about visiting entities, which may be required from a (legal) privacy perspective. As the foreign domain is not likely to know the identity of its visitors, we instantiate one or more new TCAs for each visitor. No authentication is required. In practice, a TCA will be created for each network interface belonging to the user that obtains access to networking resources at the visiting domain.

In this section we describe two protocols: one for setting up a secure association between a TCA and the context agent in the home domain (Section 4.1) and the protocol that subsequently makes the context information from the foreign domain available to applications through the context agent (Section 4.2).

4.1 Establishing Secure Associations

To enable applications to obtain context information from a foreign domain, we will need to establish a

connection between a TCA and the context agent of the corresponding entity in the home domain. In our system, it is the mobile user's personal device that controls this process. The reason is that it typically knows when it is in a foreign domain, plus that it has a secure association with its own context agent. A TCA will be created by the foreign (visiting) domain for all network interfaces of the personal devices of the user that are connected to access points in the foreign environment. For instance, the user's WiFi MAC address, or bluetooth id. This, therefore, does not require any explicit tagging by foreign host, and it will allow the guest to use pre-configured home-credentials on his personal device to link the foreign TCA to its own Context Agent.

To establish these secure connections, we introduce two additional components: Authentication and Authorization Services and Context Agent Discovery Clients.

An *Authentication and Authorization Service* (AAS) is responsible for authenticating and authorizing visitors for particular services in the domain for which the AAS is authorative. The AAS grants these rights by means of access tokens. AAS clients (in our scenario: the personal device of the visitor) need to present these tokens to the services in the AAS' domain in order for those services to execute the client's requests. AAS clients discover the AAS of a certain domain's by means of a generic service discovery protocols like SLP or WS-Discovery. Our AAS service is configured to provide zero-authentication guest-certificates for nodes that have obtained network access, like our visitors. Local users will be granted more permissions, typically. This means that cross-domain authentication mechanisms are not needed.

The *Context Agent Discovery Client* (CADC) controls the establishment of secure associations between a context agent and a TCA. A CADC runs on a personal mobile device (e.g., on Alice's smart phone) and goes through the following four steps:

1) Upon entering a foreign domain, the CADC obtains an IP address and discovers the foreign domain's AAS and its CAM;
2) The CADC obtains an access token from the AAS and requests the CAM in the foreign domain to create a TCA for the roaming entity. The request message includes the token the CADC received from the AAS. The CAM checks with its local AAS if the token is valid. If it is, it retrieves the privacy policies it requires from the policy repository, creates the requested TCA, and configures the TCA with the foreign domain's privacy policies;
3) The CADC obtains a reference to its TCA from the CAM and registers it with the entity's context agent in the home domain. The registration message includes the CADC's access token in the home domain (which was issued previously by the AAS in the home domain) as well as the access token that the CADC received from the AAS in the foreign domain. The CADC encrypts all messages between the user's personal device and the home context agent (we use SSL for this purpose).
4) The context agent in the home domain establishes a secure association with the TCA, presenting the credentials the CADC received from the AAS in the foreign domain (token B in Figure 2).

Figure 2 illustrates this using the example of Figure 1.

An alternative approach would be that the TCA registers itself with the entity's agent in the home domain. The disadvantage is that this would require the CADC to expose the entity's identity to the foreign domain, which may not be desirable from a privacy perspective.

The CADC is also responsible for keeping the association between its context agent and the TCAs alive. It accomplishes this by regularly refreshing (1) the access token and the TCA it requested in the foreign domain and (2) the registration of the TCA with the entity's context agent. We expect these refresh intervals to be rather large (currently set to 30 minutes), which minimizes the impact on the mobile device (e.g., in terms of used battery power). An alternative is that the entity's context agent refreshes the association with a TCA by regularly polling that TCA.

The association between a context agent and a TCA disappears automatically if the TCA disappears. The CADC may however also explicitly remove such an

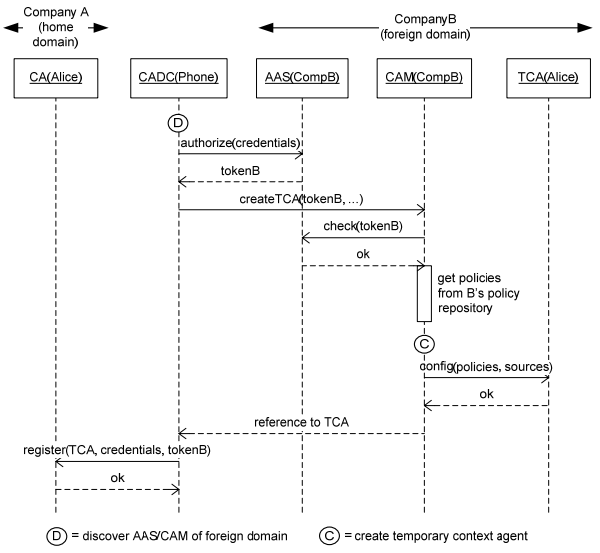

Figure 2. Establishing a link with a TCA.

association by deregistering the TCA at the context agent. The corresponding message contains the access token for the home domain as well as the one from the foreign domain.

4.2 Obtaining Context Information

After having established a secure association, CMF applications such as Bob's telephony client can obtain (foreign) context information from a context agent. These interactions follow three steps procedure, which is similar to typical discovery protocols:

1) *Locate an entity's context agent.* A client application uses a name resolution infrastructure to map the identity of an entity onto the network address of the corresponding context agent (see Section 3.2);
2) *Query the context agent.* A client queries a context agent by sending a query message to it. The message contains the type of context information the client is looking for, the identity of the requesting entity (e.g., Bob@CompanyA), and a token that represents that requestor's rights (e.g., using SAML [17]). The context agent validates the request at its PDP. If the subject of the query (e.g., Alice) is in a foreign domain and if the PDP indicates that the client is allowed to get the requested type of context information, the context agent forwards the query to the TCA in the foreign domain using the secure connection of Section 4.1. The TCA then creates a proxy context source in the foreign domain, links it to the foreign context source, configures the proxy with the foreign domain's privacy policies, and returns the reference to the proxy to the context agent in the home domain. This agent creates another proxy in the home domain and configures it with the privacy policies of the roaming entity. It then links the proxy to its peer in the foreign domain, and passes the references to the proxy in the home domain to the client application. If the entity is in its home domain, the context agent will not link to a TCA and the proxies in the home domain will link to context sources in that same domain.
3) *Query the proxy context source.* A client uses the reference it received from the context agent in the home domain to query the proxy context source and obtain the actual context information. The proxy forwards the query to its peer in the foreign domain, which gets the actual context information from the context source for which it acts as a proxy and returns the context information back to the client along the same path. The proxy context source in the foreign domain interacts with its local PDP to enforce the privacy policies of the foreign domain, which may for instance result in the proxy lowering the quality of the context information. The proxy in the home domain also interacts with its local PDP and may modify the context information it receives from the proxy in the foreign domain once more before passing it to the application. The obligation part of the PDPs' policy decisions specify the maximum QoC.

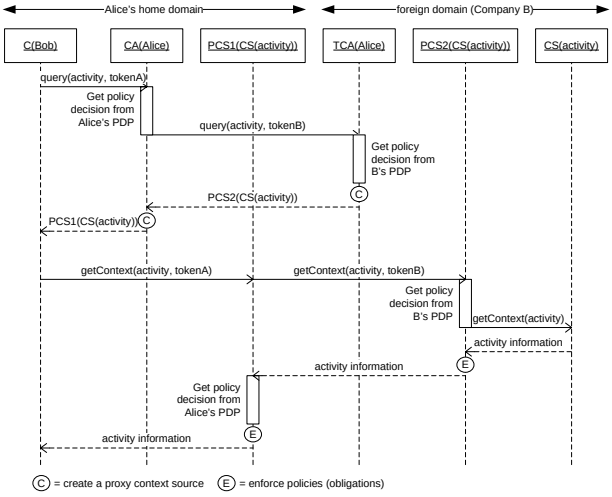

Figure 3. Inter-domain context sharing

Figure 3 illustrates this behavior for Bob's client and Alice's context agent (example in section 2). To simplify the figure, we did not detail the interactions with the PDPs of Alice and the foreign domain. The token that Bob's client application passed to Alice's context agent (tokenA) proves Bob's authenticity and is typically issued by a trusted third party. Observe that token B is a result of the establishment of the secure connection with the TCA (see Section 4.1).

The usage of two sets of privacy policies (one of the roaming entity and one from the foreign domain) may result in certain context information never reaching the context-aware application. This for instance happens when the policies of the visiting domain lower the QoC below the level required by the policies of the roaming user and is therefore not returned to the requesting application. As the policy rules will not be exchanged (privacy), this is impossible to prevent.

If the foreign domain's security policies do not allow access to the TCA, our system also supports a fall-back scenario where the CADC entity instantiates a context source proxy on the personal device that acts as a second bridge. This does, of course, consume additional resources on the personal device.

This system also assumes that the quality of the information provided by the TCA is sufficient. This could very well be spoofed by un-compliant intelligent environments. This is another level of trust-

management that is not covered by our system, although it is possible to switch the 'automatic context sensing' feature off.

5. Implementation and Deployment

We implemented the CMF as Java components. They run in a distributed run-time environment that allows them to abstract away from the underlying operating system (Windows 2000/XP, Windows Mobile, Linux), RPC mechanisms (web services, XML-RPC), local service discovery infrastructures (WS-Discovery, centralized registry), and global name resolution schemes (SIP, DNS). Only a few context sources are operating system or hardware dependent because they acquire context in an OS/hardware-specific way. Examples include keyboard activity, schedule information and system power status.

Context sources use an OWL ontology to describe the types of information they produce. We use Jena [13] to generate and parse the RDF and to process SPARQL sent to context sources and context agents.

We use the OASIS standard XACML (eXtensible Access Control Markup Language) [8] to encode the privacy policies that our system uses and use the PDP implementation provided by SUN to evaluate those policies. We extended the PDP implementation so that policies can specify a maximum allowed QoC, in particular regarding the precision of activity information. If a certain request triggers the evaluation of multiple policies, the XACML PDP chooses the highest allowable QoC as the result. More details can be found in

Figure 4 illustrates how we deployed the CMF. We put a CMF-enabled PC in our office (the home

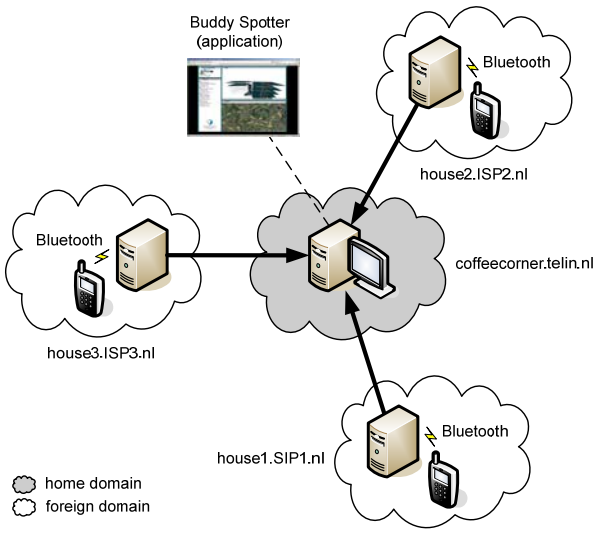

Figure 4. Inter-domain deployment of the CMF.

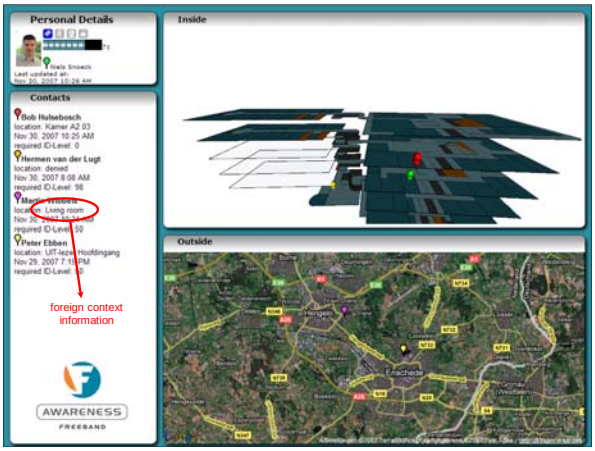

Figure 5. Screenshot of Buddy Spotter application.

domain) and in the houses of three different employees (foreign domains). The PCs in the three houses are equipped with Bluetooth dongles and contain a context source that uses the Bluetooth RF signal to determine if an employee is in his house or not. The PC in the home domain hosts three context agents, one for each of the three employees. The PCs in the foreign domains run a TCA when the employee is in his house and feed the extra context information ("at home" or "away") back to the employee's context agent in the home domain.

We make the additional context information available through an instant messaging-like application called the Buddy Spotter. The Buddy Spotter is accessible from a wall-mounted display in our office's coffee corner and keeps track of the locations of several employees, including the three with the PCs in their houses. For these employees, the Buddy Spotter also indicates if they are at home, which is information it obtains from the three context agents running on the office PC. Without our system, this information would have been unavailable without inter-domain context sharing. Figure 5 shows a screen shot of the Buddy Spotter application and marks the foreign location information ("living room") available on user Martin Wibbels.

6. Related Work

The work that comes closest to ours is the Vade system of José et al. [12]. Their context managers are very similar to our context agents and also allow applications to obtain foreign context information. The main difference is that our system explicitly supports configurable privacy policies, also for foreign domains. Another difference is that José et al. seem to rely on telecom operators to track a device's location and use

that information to determine whether a mobile device is in a particular foreign domain or not. Although they also identify our mobile-controlled approach, they have not detailed it.

Several other projects, like e.g. in-Context, have developed a system that essentially has a per-group server (e.g. [[8]]. This is fundamentally different from our approach, in which each entity has its own service, and sharing and privacy-management does not require a centralized service at all.

Project DAIDALOS has developed a system in which foreign domains push the context information they collect about visiting users to so-called "home managers" [18][21]. A home manager is a database (that resides in a user's home domain) with context information about a particular user, including foreign context information. The similarity with our approach is that context agents also provide a single point of contact for context information. The main difference is that DAIDALOS targets an environment that consists of large trusted domains that are federated through service level agreements (cf. traditional telco operators). As a result, DAIDALOS users inform a foreign domain of the URL of their home manager, which the foreign domain subsequently uses to push context information to the home manager. In our system, visiting users typically deal with untrusted foreign domains and therefore usually do not publish their identity. As a result, it is the user's personal device that establishes a link with the foreign domain and our home domains pull for foreign context information. Another important difference is that the DAIDALOS system does not consider the privacy policies of users and foreign domains.

While not the main contribution of their paper, Chen, Finin, and Joshi [3] outline how their COBRA system could be used to provide foreign context information to applications. Unlike in our system, they make foreign domains responsible for enforcing the privacy policies of visiting users. The advantage of their approach is that it takes the home domain out of the loop for policy enforcement, which reduces the load on the home domain and possibly also the number of traversed network hops. The downside is that roaming users have to (1) inform every foreign domain they visit of their privacy policies, and (2) that they need to trust these domains to correctly enforce those policies and (3) that applications will need to rebind to the foreign sources of information. This is not necessary in our approach, but malicious foreign domains can still manipulate the context information they collect about a visiting entity. Another limitation of the COBRA system is that they do not consider the privacy policies of foreign domains, which we do.

Like our system, the ACAI system of Khedr and Karmouch [14] also enables applications to obtain foreign context information. The main difference with our work is that they do not consider the privacy policies of roaming entities and foreign domains. Their architecture furthermore assumes that each domain runs an agent platform. In our architecture, we do not make this assumption and allow each domain to use its own technology for intra-domain discovery and communications. Finally, the ACAI system uses SIP for inter-domain communications and for location updates, which is one of the protocols we could have used to realize these interactions in our system (between a CADC and a context agent as well as between a context agent and its TCA).

Chin et al. [4] and Gu et al. [9] enable context-aware applications to get foreign context information by means of a self-organizing peer-to-peer overlay that interconnects the context brokers (called "discovery gateways") of different administrative domains (which they call "spaces"). The difference with our work is that they rely on the peer-to-peer infrastructure to discover context sources in foreign domains and do not leverage the current location (domain) of a roaming entity. There also seems to be no explicit mechanism to determine in which domain a roaming entity currently resides, which makes it unclear how they associate foreign context sources with that entity. They also not consider privacy aspects.

7. Conclusions

In order for the vision of ubiquitous, context-aware computing to become a reality, we will need to be able to share context information across different domains of administration. This will enable context-aware applications to use context sources in foreign domains to obtain additional context information about roaming entities such as users and devices, which will ultimately improve these applications' operation. A crucial requirement, however, is that the underlying system enforces the privacy policies of mobile users and foreign domains, which determine who is allowed to get what context information under which circumstances.

The key contribution of our system is that it realizes both of these requirements. In contrast to existing systems, we do not require pre-established security relations between domains, but dynamically establish these relations based on the physical presence of a roaming entity in a foreign domain. Based on current literature, we believe this is novel work. Another benefit of our work is that applications that use our system will only need to interact with a single context

agent, independent of the domain that provides the contextual information about that specific entity. We have implemented all concepts described in this paper and also deployed the CMF on a small scale.

Future work includes the validation and usability check of the policy language, in particular regarding its expressiveness, the types of rules it should support, automation of context-information adaptation (due to policy enforcement) and the empowerment how users can be empowered to easily manage their privacy policies. We will also investigate privacy policies that depend on context information and will use the system for different applications (healthcare domain, professional services) in order to validate the application programming model of the CMF.

Acknowledgments. This work has been conducted within the projects Freeband AWARENESS (co-sponsored by the Dutch government under contract BSIK 03025) and IST Amigo (partially funded by the European Commission under contract IST 004182). Remco Poortinga and Niels Snoeck contributed to the topics presented in this paper. Maarten Wegdam reviewed the draft version of this paper.

8. References

[1] M. Blackstock, R. Lea, and C. Krasic, "Toward Wide Area Interaction with Ubiquitous Computing Environments", 1st European Conference on Smart Sensing and Context, the Netherlands, October 2006

[2] T. Buchholz, A. Küpper, and M. Schiffers, "Quality of context: What it is and why we need it", Workshop of the HP OpenView University Association (HPOVUA 2003), Geneva, 2003

[3] H. Chen, T. Finin, and A. Joshi, "Using OWL in a Pervasive Computing Broker", Workshop on Ontologies in Open Agent Systems (AAMAS 2003), July 2003

[4] C.-Y. Chin, D. Zhang, M. Gurusamy, "Orion: P2P-based Inter-Space Context Discovery Platform", 2nd Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous'05), San Diego, USA, July 2005

[5] N. Davies and H.-W. Gellersen, "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems", IEEE Pervasive Computing, pp. 26-35, January 2002

[6] P. Debaty and D. Caswell, "Uniform Web presence architecture for people, places, and things", IEEE Personal Communications, Volume 8, Issue 4, Aug 2001, pp. 46-51

[7] 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, Volume 16 (2-4), 2001, pp. 97-166

[8] Dorn, C. Schall, D. Dustdar, S., "Granular Context in Collaborative Mobile Environments", Springer LNCS 4278, pp 1904-1913, Springer Verlag, 2006

[9] T. Gu, E. Tan, H. Keng Pung, D. Zhang "A Peer-to-Peer Architecture for Context Lookup", 2nd International Conference on Mobile and Ubiquitous Systems (MobiQuitous'05), San Diego, California, July 2005

[10] E. Guttman, C. Perkins, J. Veizades, M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999

[11] C. Hesselman, H. Eertink, and M. Wibbels, "Privacy-aware Context Discovery for Next Generation Mobile Services", 3rd SAINT2007 Workshop on Next Generation Service Platforms for Future Mobile Systems (SPMS 2007), Hiroshima, Japan, January 2007

[12] R. José, F. Meneses, A. Moreira, "Integrated Context Management for Multi-domain Pervasive Environments", First International Workshop on Managing Context Information in Mobile and Pervasive Environments (MCMP-05), Ayia Napa, Cyprus, May 2005

[13] HP Labs, "Jena – A Semantic Web Framework for Java", http://jena.sourceforge.net/, October 2005

[14] M. Khedr and A. Karmouch "'ACAI: Agent-Based Context-aware Infrastructure for Spontaneous Applications", Journal of Network & Computer Applications, Volume 28, Issue 1, pp. 19-44, 2005

[15] H. van Kranenburg, M. S. Bargh, S. Iacob, A. Peddemors, "A Context Management Framework for Supporting Context-Aware Distributed Applications", IEEE Communications Magazine, August 2006, pp. 67-74.

[16] B. Moore, et al., IETF RFC3060 Policy Core Information Model—Version 1 Specification, February 2001.

[17] Ragouzis, et al. (eds.), Security Assertion Markup Language (SAML) V2.0 Technical Overview, http://www.oasis-open.org/committees/documents.php?wg_abbrev=security, Oct 2006.

[18] I. Roussaki, M. Strimpakou, C. Pils, N. Kalatzis, M. Neubauer, C. Hauser, and M. Anagnostou, "Privacy-Aware Modelling and Distribution of Context Information in Pervasive Service Provision", IEEE International Conference on Pervasive Services (ICPS 2006), Lyon, France 2006, pp. 150-160

[19] J. Schlimmer (ed), "Web Services Dynamic Discovery (WS-Discovery)", Microsoft Corporation, 2005, http://msdn2.microsoft.com/en-us/library/bb706924.aspx

[20] K. Sheikh, M. Wegdam, M. Sinderen, "Quality-of-Context and its use for Protecting Privacy in Context Aware Systems", Journal of Software (JSW), Vol.3, Issue 3, pp 83-93, March 2008.

[21] M. Strimpakou, I. Roussaki, C. Pils, M.Angermann, P. Robertson, M. Anagnostou, "Context Modelling and Management in Ambient-aware Pervasive Environments", International Workshaop on Location- and Context-Awareness (LoCa 2005), Munich, Germany, 2005