The Architecture Design of a Cross-Domain Context Management System

Bin Guo, Lin Sun, Daqing Zhang
Institut Telecom SudParis
9, rue Charles Fourier, 91011 Evry Cedex, France
Email: {bin.guo, lin.sun, daqing.zhang}@it-sudparis.eu

Abstract—A new context management framework, which supports the interaction between different domain context managers, is proposed. Two different context producer-consumer patterns are defined to support context-aware applications. To deal with the mobile entity problem in cross-domain context sharing, a transparent query mechanism that enables applications to obtain context information about mobile entities from remote domains is presented. The capability of our framework is demonstrated through a user scenario.

Keywords—Cross-Domain; Pervasive Computing; Architecture; Context Management; Remote Context Querying

I. INTRODUCTION

The context manager acts as middleware between context-aware applications (or context consumers) and context producers, which can collect contexts from various context producers and provide query mechanisms for applications to selectively access to the contexts they need.

In recent years, different context managers have been developed to meet specific domain requirements (e.g., smart homes, smart hospitals). These context managers are designed to store contexts produced in a single environment and deliver them to applications running locally. However, very few of them consider about context exchange between different domains. The scenario originated from the Feel@Home project gives an example on this.

The Dupont family is composed of the father David, the mother Barbara, and their kid Tony. It’s 10:00 AM, Barbara is having a meeting at one of her partner’s office (called Office-M), and David is strolling with Tony in the street. At this time, the postman arrives to their house bringing a postal parcel from Barbara’s parents living in Lyon. The home context manager (HCM) detects that the doorbell is ringing and publish this context to the Smart-Gate application (SGA). SGA queries HCM and finds nobody is at home. After checking Barbara’s current status from remote domain context managers (including the one she is currently in, e.g., Office-M, and others where she is a registered user, e.g., her own office), the home server finds out that Barbara is in a meeting and David is available to answer the doorbell. It then sends a picture of the postman taken to David to inform this situation. David recognizes the postman and agrees to answer the doorbell, a connection between David’s smart phone and the intercom installed on the gate is built. The postman talks with David directly and they agree on a new time for resending the parcel.

The above scenario leads to a few issues to be addressed in cross-domain context management. Here we focus on the following two problems.

- Update of the context manager (CM) entry point for mobile entities when they roam around different domains (e.g., when Barbara moves from home to her office, a “handover” from the home CM to the office CM is needed).
- Effective routing methods that ensure remote context queries can be directed to the right context manager. For example, when Barbara is having a meeting in Office-M, the query of her location should be directed to the CM of Office-M; however, for some other contexts, like her calendar info, is managed by the CM of her own office, and a query about this should be routed there instead. Therefore a routing mechanism is required to fulfill this task.

In contrast to “peer-to-peer routing” and “home-broker based redirecting” solutions provided by previous studies that either do not consider mobile entity problem or only partly solve the routing issue (see Section 6), this paper presents a global context management architecture to address the above issues. The main benefit of it is that it provides a unified method for managing the CM entry point update of mobile entities (called “mobile entity registration”) and a transparent routing mechanism that enables applications to obtain context info about mobile entities from the right domain. Our work is based on the requirements from a running Europe project – Feel@Home (https://rd-projet.feelathome.rd.francetelecom.com).

The rest of this paper is organized as follows. In Section 2 we describe the Feel@Home project and analyze its requirements on cross-domain context interaction. Based on the requirements, a new context management framework is proposed in Section 4. In Section 5, we explain the cross-domain mobile entity registration and context query mechanism used in our framework, followed by a scenario analysis to demonstrate this method. Section 6 compares our work with other related work. Finally, we summarize our work and present several issues to be addressed.

II. THE FEEL@HOME PROJECT

The Feel@Home project, coordinated by France Telecom and Telefonica, aims at resource sharing (including contexts and multimedia resources) and remote control among smart environments. The principle for context management in it is that contexts generated in different Feel@Home enabled environments should be managed by their local context
managers. In the current stage, three different environments – home, office, and outdoor – are considered, and three context managers are built accordingly.

- **Home context manager** manages contexts generated within a home environment, such as the status of a house, user location in a house (e.g., in the kitchen), user activity in a house (e.g., cooking, watching TV).

- **Office context manager** manages contexts generated within an office environment, such as user location in the office, and personal (e.g., drinking coffee) or group activities (e.g., having a meeting).

- **Mobile context manager** manages contexts generated from outdoor environments (e.g., on the street, in a car, on the train), which is mainly collected by a user’s smart phone (equipped with sensors, such as accelerometers and GPS), such as network status (e.g., 3G or Wi-Fi), user geo-location (e.g., in a park, near the supermarket), user activities (e.g., walking, running, travelling in a car).

Based on the design principle that contexts produced in a domain should be managed by the local context manager, two requirements for context-aware service providing are identified.

- (1) **Two context producer-consumer patterns**: There are general two types of context consumers in Feel@Home. **Indoor applications** running in home/office domain and **mobile applications** running on a mobile device in the mobile domain. In consideration of different locations of context producers and context consumers within a multi-context-manager environment, two context-aware application design patterns are derived:

  - Intra-domain context producer-consumer pattern: In this pattern, the application running in a certain domain consumes the contexts produced by the same domain. For example, in the scenario described in the introduction, the Smart-Gate application running at home has to check the availability of the out-of-home family members. But as a person like Barbara may present in different environments, it is not possible for the application to specify a target remote context manager to fetch her status context. One way to solve the mobile-entity problem is to broadcast the query request to several possible target domains and compare the timestamp of retrieved results, but because the human often enters a guest domain that is unknown to applications (like ‘Office-M’ in the scenario), this solution can only handle a few situations. Therefore, for a remote query that is hard to specify the target domain, one feasible way is to provide a transparent mechanism that can route the query to right context manager.

- (2) **Remote context query mechanism**: In the cross-domain pattern, an application from a domain needs to obtain contexts from remote domains. There are general two cases for remote context query:

  - **Explicit query**: For the contexts that are produced and maintained by a single domain, such as lighting-level of a room and current group activity in an office, the application can specify where it can be obtained.

  - **Mobile entity query**: For the contexts that may exist in several domains, the application, in most cases, cannot specify where the context can be obtained. It mainly happens when querying contexts about mobile entities like a human, because he roams among different domains and his context information is maintained by distinct context managers. It should be noted that in Feel@Home, we assume that a user can not only be served by his “registered domains” (e.g., his home/office), but, at least partly, served by his friendly “guest domains” (e.g., his friend’s home, his partner’s office). Since part of information about the human is overlapping among these domains, such as his location, sometimes it is hard for a context consumer to specify where the freshest information about the human can be obtained. For example, in the described scenario, the Smart-Gate application running at home has to check the availability of the out-of-home family members. But as a person like Barbara may present in different environments, it is not possible for the application to specify a target remote context manager to fetch her status context. One way to solve the mobile-entity problem is to broadcast the query request to several possible target domains and compare the timestamp of retrieved results, but because the human often enters a guest domain that is unknown to applications (like ‘Office-M’ in the scenario), this solution can only handle a few situations. Therefore, for a remote query that is hard to specify the target domain, one feasible way is to provide a transparent mechanism that can route the query to right context manager.

### III. SYSTEM DESIGN

Based on the requirements presented in last section, we design the Feel@Home context management framework, which supports both intra-domain and cross-domain context producer-consumer patterns and builds a global routing scheme to facilitate remote context query.

#### A. Feel@Home Framework

As illustrated in Fig. 1, the Feel@Home framework is broadly divided into three parts: **remote queries from applications**, **global administration server (GAS)**, and **domain context managers (DCM)**.

- (1) **Remote queries from applications**. It refers to applications that try to retrieve context information from remote domains. Two different remote query types are identified previously and they should provide different information in the query statement.

  For an explicit query, the query statement should include four parameters, **Domain_ID** (the domain of the target entity; managed by the GAS), **Entity_ID** (e.g., a light; managed by the DCM), **Context_Name** (e.g., ‘Status’) and **Requester Info** (it may be a domain id for an indoor application or a user id for a mobile app). An example that queries the status of a light in a remote domain is shown in Table I (see Example 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Example 1</th>
<th>Ex. 2</th>
<th>Ex. 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain/User ID</td>
<td>d00001</td>
<td>u00001</td>
<td>u00001</td>
</tr>
<tr>
<td>Entity ID</td>
<td>light01</td>
<td>–</td>
<td>notebook01</td>
</tr>
<tr>
<td>Context Name</td>
<td>Status</td>
<td>Activity</td>
<td>Status</td>
</tr>
<tr>
<td>Requester Info</td>
<td>u00001</td>
<td>d00001</td>
<td>d00002</td>
</tr>
</tbody>
</table>

For a mobile entity query, the query statement includes three or four parameters, **User ID**, **Entity_ID** (optional), **Context_Name** and **Requester Info**. In this kind of query, the application do not have to specify an explicit domain for obtaining the context, instead, it only gives the id of a mobile
entity. For example, when querying the context information about a user, the query statement should provide \textit{User ID}; while when querying the context about a user’s belongings, the query statement should include both \textit{User ID} and \textit{Entity ID} (the id of the belonging). Two examples are given in Table I, where Ex.2 queries the activity of a user, and Ex.3 queries the status of a user’s notebook (a belonging).

Table I. ACCESS ENTRY TABLE (A SAMPLE)

<table>
<thead>
<tr>
<th>Entity ID</th>
<th>Entity Name</th>
<th>Access Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>u00001</td>
<td>John</td>
<td>143.168.10.2/Access</td>
</tr>
<tr>
<td>u00002</td>
<td>John’s Mother</td>
<td>143.167.10.3/Access</td>
</tr>
<tr>
<td>u00003</td>
<td>Alice</td>
<td>143.165.11.5/Access</td>
</tr>
<tr>
<td>d00001</td>
<td>John’s Home</td>
<td>156.155.14.2/Access</td>
</tr>
<tr>
<td>d00002</td>
<td>John’s Office</td>
<td>150.2.17.60/Access</td>
</tr>
</tbody>
</table>

Context Entry Manager (CEM) is a component that deals with updated requests to CED (updating AET table when the user enters a new domain). We clarify this process in detail in Section 4.1.

Context Entry Engine (CEE) is the core module that routes remote queries to the right context manager. For an explicit query request, CEE queries AET to retrieve the entry point of the target domain context manager. For a mobile entity query request, a transparent solution is provided. It firstly redirects the query to the domain the user is currently in. If there is no result, it then sends the query to the registered domains of the user by querying URT (some contexts about the user may not be managed by the domain the user is currently in, for example, his biomedical info is not stored in his friend’s context manager). We describe this in Section 4.2.

Access control (AC) is another important issue in cross-domain context exchange. In the current design, there are global access control (GAC) in GAS side and local access control (LAC) in domain side. Since the access control function is out of the scope of this paper, we just elaborate the common functions of them: GAC is in charge of preventing the unwanted visitors to a domain or to a user; LAC is used for managing the user access to certain context information (e.g., which contexts, whose contexts, in what situation, to what extent can be accessed by authorized consumers).

(3) Domain Context managers (DCM). DCM manages contexts produced in its domain and deals with context consumption by local/remote applications. We describe its inner architecture in next subsection.

B. Domain Context Manager Architecture

The internal architecture of a domain context manager is shown in Fig. 2, which is based on our previous work [1, 2]. It consists of the following components.

\textit{Context Wrappers} transform the obtained raw data from various sensing sources into context markups and sends it to the context aggregator. By gathering contexts from context wrappers, the context aggregator will trigger JENA operations to store and infer contexts.

The JENA component is based on the Jena Semantic Web package (http://jena.sourceforge.net/), which integrates several APIs to query and modify the context knowledge base (CKB) at the programming level. Jena also provides an inference engine (the context reasoner) that can infer high-level contexts from low-level ones. We leverage the Semantic Web language – OWL [3] to represent contexts in CKB.

When a user enters a domain, the Registration Centre of this domain will interact with this user to determine whether he is a guest or a registered user and if he wants to be served by this domain (see details in Section 4.1).

The SOAP layer contains APIs for dealing with querying and modifying requests from local or remote applications.
The main task of it is to (1) analyze SOAP requests and translate them into the formal format (e.g., Jena query language) that can be processed by JENA, (2) encapsulate the query result from JENA and send it back to the requester. With the support of the Publish Engine (based on a publish/subscribe engine like ActiveMQ, refer to "http://activemq.apache.org/"), a subscription API that supports context subscription by local applications is also provided (Feel@Home does not consider remote subscription in its current stage).

As shown in Fig. 2, for local applications, they can directly send a query to the Query API at the SOAP layer. The JENA component will execute the query and send back the result to the application directly. This illustrates the working process of the intra-domain context producer-consumer pattern. For the cross-domain context exchange, the process is more difficult, and we will describe it in next section.

IV. CROSS-DOMAIN CONTEXT QUERY

As presented in Section 2, there are two types of remote queries in terms of whether the context consumer can clearly specify the target context managers. We provide a unified, global scheme to deal with this issue. The whole scheme consists of two main parts: mobile entity registration and global context query mechanism.

A. Mobile Entity Registration

The mobile entity registration function supports user-entry-point update when a user roams from domain A to domain B. Upon entering domain B, the user will update his entry point maintained by GAS from A to B. This function enables applications to obtain the freshest context about the user.

We divide the registration process into three stages: (1) domain-entering event detection, (2) authorization, and (3) entry point update. The whole process is controlled by the user’s mobile phone, because it knows when it changes domains (the Feel@Home project allows smooth network connection change, for instance, from 3G to local Wi-Fi) and it is a trusted agent for the user as well as GAS (to avoid phishing registration).

In the domain-entering event detection stage, the user agent (UA) running on a personal mobile phone discovers the domain registration service, and a session is established between UA and the domain registration center (DRC). The UA sends a request to the user and asks whether he would like to be served by this domain. The session stops if the user doesn’t want to be served. Otherwise, the registration process enters its next stage.

In the authorization stage, UA will send the user’s id to DRC. DRC checks the user id by querying CKB and determines whether the user is already a registered user or just a guest user of this domain.

- **If the user is a registered user.** DRC updates the related information about the user (e.g., his location) to CKB;
- **If the user is a guest user.** DRC generates a temporary account (using the user’s id) for him (in CKB).

In the entry point update stage, UA obtains the reference address (i.e., domain ID) to the domain context manager from DRC and updates the user’s presence information to GAS.

Through the above three stages, a mobile user can maintain his newest entry point information in the GAS, which can be acquired by authorized remote consumers. Corresponding to the two diverse authorization processes (a registered user or a guest user) mentioned above, when the user leaves a space, there are also two cases:

- If he is a registered user, some dynamic contexts related to the user (e.g., his location, his activity) will be set to none;
- If he is a guest user, his temporary account and all related contexts will be deleted.

B. Global Context Query Mechanism

The global context query mechanism provides a unified way for cross-domain context query. Both the two query types identified in Section 2, explicit query and mobile entity query, are supported by this mechanism. The query mechanism is implemented through the following four stages.

In the access control stage, the application sends a remote query to GAS. The global access control (GAC) component of GAS examines whether the application (according to the requester info) has the right to access the target domain (for explicit queries) or to obtain the information about the target user (for mobile entity queries).

If the query request is allowed, we enter the access entry acquisition stage. In this stage, the context entry engine (CEE) of GAS will query the access entry table for the entry point of the target domain or user. If the target is a user, CEE will get the entry point of the domain context manager where the user currently resides.

In the entry-point based query stage, CEE forwards the query to the domain context manager according to the obtained entry point. The local access control (LAC) component of DCM analyzes the query request and checks if it can be executed according to user privacy setting in LAC (certain contexts cannot be obtained by remote queries in terms of privacy consideration). If the query request is allowed, execute the query and send back the result to CEE.
V. SCENARIO ANALYSIS

Based on the cross-domain context management framework we present, in this section, we analyze how it support the scenario described in the introduction. The whole operation process is illustrated by a UML sequence diagram shown in Fig. 4.

![Sequence Diagram](Image)

Figure 4. Feel@Home Scenario Analysis.

When the postman arrives, the home context manager (HCM) acquires this context through the ringing doorbell. Since this context has been subscribed by the local Smart-Gate application (SGA), it will publish it to this application. By querying HCM and learns that nobody is at home, SGA sends remote queries to GAS to check the current activity of the out-of-home family members and to learn who is available to answer the doorbell. First remote query is about Barbara. CEE at GAS firstly forwards the query to the Office-M’s context manager according to the record from the entry point table (Barbara is currently there). However, because Office-M only provide location context for remote queries, it does not provide any activity information about Barbara. CEE then redirects the query to the registered context managers of Barbara, including her office and the mobile context manager, and sends the final results to SGA. According to the electric calendar information from Barbara’s office, SGA finds out that Barbara is having a meeting in another place and thus she is not available. Another remote query is sent to fetch David’s status. By the entry-point-based query, SGA learns that David is walking in a park and he is available to answer the doorbell. Finally, a video-based connection is established between David and the postman.

VI. RELATED WORK

Over the last decade, a lot of researchers have been working on context management architecture for context-aware systems. The pioneering work of Context Toolkit presents an object-oriented architecture for rapid prototyping of context-aware applications [4]. In the CoBrA project [5], Chen et al. propose an agent-oriented infrastructure for semantic context representation, knowledge sharing and privacy control. Want et al.'s Semantic Space system exploits the Semantic Web technologies to support explicit representation, expressive querying and flexible reasoning of contexts in smart spaces [2]. A common problem of these systems is that their operations are constrained to a single domain. As a result, applications cannot obtain context information from remote context managers.

Little attention has been paid to cross-domain context exchange. Gu et al. [6] enable context-aware applications to get foreign context information by means of a self-organizing peer-to-peer overlay that interconnects the context managers of different smart spaces. The difference with our work is that they focus on context source discovery in remote context mangers and do not contain a solution to mobile entity problem.

The CMF system of Hesselman et al [7] leverages mobile entity problem. In their approach, each user belongs to a “home” domain (e.g., his company) and all other domains are called foreign domains. When a user enters a foreign domain, his mobile detects this and triggers a connection between his home domain and the foreign domain. The applications running at the home domain can then obtain information from the foreign domain. This solution has several drawbacks. First, the assumption is not reasonable, because a user may have several home or registered domains, including his home and his working office. According to the proposed solution, not all registered domains can obtain freshest context information about the user from the remote domain he resides. Second, it does not consider the distributed storage nature of the contexts to mobile entities. Beyond the home domain and the foreign domain the user is currently in, some information about the user may be stored in another registered domain of him, which cannot be routed to by their solution. Comparing with CMF, our system
provides a global routing mechanism that supports, all authorized applications, no matter where they locates, can on one hand fetch the freshest context info about a mobile entity from his newest registered entry-point, and on the other hand, obtain the context from several registered domains of the user when the entry-point-based query fails.

VII. CONCLUSION

This paper reports our initial effort on cross-domain context management. Two context producer-consumer patterns, intra-domain and cross-domain, are identified and supported by the Feel@Home context management framework we proposed. Our system further addresses the mobile entity related context acquisition problem, and provides a global routing mechanism to support transparently, effectively obtain context information about mobile entities from remote domains. Future work includes implementation and evaluation the performance of the framework as well as integration of robust access control mechanisms.

REFERENCES