iCROSS: Towards a Scalable Infrastructure for Cross-Domain Context Management

Bin Guo[1], Daqing Zhang[1, 2], Lin Sun[2], Zhiwen Yu[1], Xingshe Zhou[1]

[1] Northwestern Polytechnical University, P.R. China
[2] Institut TELECOM SudParis, France

{guob, zhiwenyu, zhouxs}@nwpu.edu.cn, {daqing.zhang, lin.sun}@it-sudparis.eu

Abstract

With the development of pervasive computing techniques, the world will be filled with interconnected context-aware domains (e.g., homes, offices, hospitals, etc.). While previous studies focused solely on the management of contexts produced in a single domain, in this paper we discuss the challenges to be addressed for cross-domain context management. By analyzing the requirements from several scenarios, we identity two different context producer-consumer patterns in multi-domain environments. Furthermore, to deal with the mobile entity problem raised in cross-domain context sharing, a transparent query mechanism that enables applications to obtain context information about mobile entities from remote domains is proposed. Two prototype applications - smart home and community services in a smart campus - have been developed to demonstrate the key features and usefulness of cross-domain context management. Initial experiments have also been conducted to evaluate the performance of our system.

Keywords: Pervasive Computing, Cross-domain Context Management, Context Awareness, Query Routing Mechanism

1. Introduction

Context-aware computing refers to the idea that applications can adapt their behaviors according to the contexts collected (e.g., location of use, the collection of nearby people, accessible devices, work schedules, software usage) in which they are run. The contexts can be sensed or extracted by various physical/software sensors in cyber-physical spaces. In a general context-aware system, the context manager often acts as middleware between context-aware applications (or context consumers) and sensors (or context producers), collecting contexts from various context producers and providing query mechanisms for applications to selectively access the contexts they need.

In recent years, different context managers have been developed to meet the requirements of specific domains, such as homes [1, 2], offices [3], hospitals [4], supermarkets [5], classrooms [6] and vehicles [7]. These context managers are designed to store contexts produced in a smart domain (e.g., a smart home) and deliver them to applications running locally (e.g., an eldercare app). With the development and prevalence of pervasive computing techniques, we can imagine that the real world will be filled with various smart domains in the near future. Human beings are mobile in nature. Each day, they travel across different smart domains and leave their digital footprints (or contexts) in them [8]. One significant problem raised here is how the contexts should be managed when the pervasive environment is extended from single to multiple domains. For example, how should the contexts be stored and queried? Should they be controlled in a *centralized* manner, or in a *self-organized* manner? Due to privacy and proprietary issues, it is not practical to assume the usage of a cloud-like server that gathers and manages all the contexts generated from different administrative domains. It is more practical to store and control the contexts generated from a particular domain by the local context manager, which we call the "*Local-Production, Local-Management*" (LPLM) principle. Under this principle, each smart domain is equipped with its own context manager to store and control the access of the contexts it produces. The following scenario from the EU Feel@Home project [9] gives an example of applying this principle.

The Smart Home Scenario. *The Dupont family is composed of the father, David, the mother, Barbara, and their son, Tony. It is 10:00 AM, David is having a meeting in a business partner's office (called Office-P domain), and Barbara is walking the dog in a park (i.e., in the mobile domain). At this moment, the postman arrives at their house, bringing a parcel from Barbara's parents living in Lyon. Having detected that the doorbell is ringing, the home domain context manager (home DCM) sends this context to its subscriber − Smart-Monitor application (SMA). SMA queries the home DCM and learns that nobody is at home. After querying David's current status from remote DCMs (including the one he is located in – Office-P, and others that he registers, e.g., his own office – Office-D), the home server learns that David is in a meeting (acquired from*

Office-D DCM) and Barbara might be available (acquired from the outdoor/mobile DCM) to answer the doorbell. It then sends a picture of the postman taken by the video intercom (installed at the entrance) to Barbara. She recognizes the postman and agrees to "answer" the doorbell. A connection between Barbara's mobile phone and the video intercom is thus built. The postman talks with Barbara directly and they make a new time for redelivering the parcel.

The above scenario introduces an interesting issue to be addressed for context management in multi-domain environments: *due to the mobility of some entities (e.g., people, devices), which domain context manager (DCM) should an application query if it wants to obtain the context information on these mobile entities from remote domains?* In the simplest case, we can obtain a person's context from the DCM that the user is currently in (e.g., acquiring Barbara's 'walking' activity from the mobile DCM), but the problem is how to determine in which domain the person locates. Things become more complicated sometimes because the target context may not be simply found from the user-situated domain. For instance, in the above scenario, David is within a foreign domain (e.g., *Office-P*) that does not provide activity tracking service for visitors (or unregistered users). His activity information, however, can be obtained from the agenda stored in the DCM of his office (*Office-D*). In short, it is often difficult for an application to specify which DCM to query, due to the "mobility" of entities (we call it the *mobile entity problem*) in multi-domain pervasive environments. Effective routing methods that can direct remote context queries to the right DCM are thus crucial.

A number of systems have been developed to manage contexts in pervasive environments. However, most of them are constrained to a single smart domain. Very few of them consider the storage and acquisition of contextual information generated from remote domains. Previous studies provide "peer-to-peer routing" and "home-broker based redirecting" solutions, but they either do not consider the mobile entity problem or only partially solve the routing issue (see Section 2). This paper focuses on the interactions between smart domains and presents a global context management system called iCROSS to address the above issues. The main contribution is that it provides a unified method to manage the DCM entry point update of mobile entities (called "*mobile entity registration*") and a transparent routing mechanism that enables applications to obtain contextual information about mobile entities from the right domain. We have developed several prototype applications to demonstrate the key features and validate the performance of iCROSS.

The rest of this paper is organized as follows. Section 2 reviews related work on context management in smart domains. In Section 3, we describe the Feel@Home project and analyze its requirements on cross-domain context interaction. Based on the requirements, we present the iCROSS system infrastructure in Section 4. In Section 5, we explain the cross-domain mobile entity registration and context query mechanism used in our system. The implementation of two prototype applications and an initial evaluation of our system are presented in Section 6. We conclude our paper in Section 7.

2. Related Work

Over the last decade, many researchers have been working on context management infrastructure for context-aware systems. The pioneering work of Context Toolkit presents an object-oriented architecture for rapid prototyping of context-aware applications [10]. In the CoBrA project [11], Chen et al. propose an agent-oriented infrastructure for semantic context representation, knowledge sharing and privacy control. Wang 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 [1]. One common feature of these studies is that they are all designed to store and manage the contexts produced in a smart domain (e.g., a smart home). To facilitate the access of contexts required by local applications, they employ informal or formal methods to represent the contexts generated by heterogeneous devices in the domain and provide a unified interface for context query. However, the problem shared by them is that their operations are constrained to collecting and accessing contexts from a single domain. As a result, applications cannot obtain context information from remote context managers. A survey on context management is recently reported in [12], which reviews and discusses a set of techniques and issues relevant to context modeling, reasoning, storage, and query. It, however, does not cover the context management problem in multi-domain environments. In their work [13], Wibisono et al. have proposed a Dempster-Shafer based approach to manage the contexts in Mobile Ad Hoc Network (MANET) environments, where each mobile phone has a context manager, and can retrieve contexts from other mobile phones. Our work is founded on a different assumption: we adopt the “LPLM” principle leveraged by CoBrA[11] and Semantic Space [1], where the contexts produced in one domain are collected and maintained by the centralized domain context server running in that domain. Context inconsistency is another important issue as a context manager, and distinct methods have been proposed to deal with it [14, 15]. In this paper we focus on the cross-domain context query issue. The context inconsistency problem in multi-domain environments will be formalized and tackled in our future work.

Very few studies have been done on cross-domain context management. In their work [16], Gu et al. propose a peer-to-peer mechanism that enables context-aware applications to retrieve foreign context information from remote smart spaces. The difference with our work is that they focus on context source discovery in remote context mangers and do not provide a solution to mobile entity problem. Similar to our system, the Vade system [17] developed by Jose et al. also relies on telecom operators to track a user’s location and use this information to determine whether the mobile device is in a particular foreign domain. The main contribution of their work is that it allows local applications running in the foreign domain to obtain contexts about a visitor and, on the other hand, enables applications running on the visitor’s mobile phone to retrieve public contexts shared by that domain. Nevertheless, it does not provide a solution that allows other domains to obtain context about a mobile entity.

The CMF system developed by Hesselman et al [18] comes closer to our work, which also deals with the 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 phone 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. A similar approach was used by Roussaki et al. in [19]. There are several drawbacks of the solution they propose. First, the assumption is not reasonable, because a user may have several “home” or “registered” domains, such as his home or office. According to the proposed solution, not all registered domains can obtain recent contextual information about the user from the remote domain he is in. Second, it does not consider the distributed storage nature of the contexts for 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 his, which cannot be routed to by their solution. In contrast with CMF, our system provides a global routing mechanism that supports all authorized applications, independently of where they are located. It can, on the one hand, fetch the most recent contextual information about a mobile entity from his newest registered entry-point, and on the other hand, obtain the context from several of the user's registered domains when the entry-point-based query fails.

3. Requirements from the Feel@Home Project

The Feel@Home project, coordinated by France Telecom and Telefonica, addresses mobile social connection, resource sharing (including contexts and multimedia resources), and remote control in multi-domain pervasive environments. Contexts generated in different Feel@Home enabled domains/environments should be managed by their local context managers (i.e., the LPLM principle described in the introduction). In the current stage, five different domains− *home*, *office*, *business*, *classroom*, and *mobile* − are considered, and five context managers are built accordingly.

- The *home DCM* manages contexts generated within a home environment, such as the status of a house, user location in a house (e.g., in the kitchen), human activity in a house (e.g., cooking, watching TV).
- The *office DCM* manages contexts generated within an office environment, such as personal (e.g., drinking coffee) and group activities (e.g., having a meeting), work schedules.
- The *business DCM* manages contexts generated within a commercial space (e.g., a hotel, a public entertainment place), such as user location and social activities (e.g., watching a movie).
- The *classroom DCM* manages contexts generated within a school classroom, such as device status (e.g., the projector is working), room status (e.g., full or free seats available).
- The *mobile DCM* manages contexts generated from outdoor environments (e.g., on the street, in a car, on the train), which are collected by user-carried mobile phones (equipped with sensors, such as accelerometers, GPS, Bluetooth), such as network

connection (e.g., 3G or Wi-Fi), user geo-location (e.g., in a park, near the supermarket), user activities (e.g., walking, running, meeting friends). It also stores the software usage contexts (e.g., video viewing, phone calls, SMS messaging, plan-to-do list) on the mobile phone.

To illustrate better the motivation and identify the requirements for cross-domain context management, beyond the "*Smart Home*" scenario presented in the introduction, here we describe three other scenarios from the project.

The Community Scenario. *Mobile social network builds an ever-connecting community to facilitate the communication and interaction among peers. One day, Barbara is doing shopping in the downtown area (DA). At around 5 pm, Barbara starts planning for her dinner. She wants to invite some friends together for the dinner if they are available in the vicinity. A request is then sent to the social activity agent (SAA) running on her mobile phone, with the semantic description of this event:*

<time-due, activity-type, location> = <6:30 pm, Dinner, NearMe>

Before the deadline of the planned event, SAA queries the status of Barbara's friends every ten minutes and sends an invitation once a match happens. Alice, a friend of Barbara, becomes the first person invited. She enters DA to buy something in a shopping mall after the request is posted 20 minutes. SAA detects this by querying Alice's mobile DCM (Alice's mobile phone is equipped with GPS). More surprisingly, Sally, who works in another city, becomes the second person being invited. She happens to have a project meeting in a hotel of DA. Sally is a common friend of Barbara and Alice, but she has not seen them for several years. Though Sally's mobile phone is not equipped with GPS, the hotel she situates reports her location information (through Wi-Fi connection). From the business DCM of the hotel, SAA learns that Sally's meeting closes at 6:15 and sends an invitation after that time. Finally, the three have a good dinner together in a restaurant of DA.

The Smart Campus Scenario. *In university campuses, students often face the problems of finding partners to do sports in a certain free time slot, searching if there are free seats in a classroom, finding if someone can help him solve a technical issue, etc. These problems can be solved in a sensor-enhanced smart campus (e.g., sensor-equipped mobile phones, classrooms with WiFi/Bluetooth access points). For example, if Tony has a problem about Java programming and he wants to find a friend to help* *him, he can send a query to the smart campus agent (SCA) running on his mobile phone. The SCA agent can retrieve the activity context of his friends from the relevant DCMs (classroom DCMs, mobile DCMs, etc.) in the campus. Afterwards, he finds that his friend Bob is studying in a classroom (queried from the classroom DCM) that is near to him, he then heads for the classroom for help.*

The Entertainment Recommendation Scenario. *On Sunday, Tony wants to look up what entertainment his friends are enjoying and plan to enjoy. He sends this request to the recommendation agent (RA) running on his mobile phone. RA queries the most recent and scheduled multimedia-related activities of his friends. Bob is watching the "National Geographic Channel" at home. John is having dinner in a KFC (a business domain), but his "want-to-enjoy" list kept in his mobile phone (the mobile DCM) indicates that he wants to see a movie titled "Tomorrow" recently. Interestingly, another friend of Tony, Tom, is watching the "Tomorrow" movie in a cinema (queried from the business DCM). RA collects all the information and displays it on Tony's mobile phone. Tony learns that the "Tomorrow" movie might be funny and decides to watch it the next day, maybe with John.*

Based on all the four scenarios described, we identify two requirements for context-aware application development in multi-domain pervasive environments.

(1) Two context producer-consumer patterns: There are generally two types of context consumers in Feel@Home, *indoor applications* running in home/office/business domain and *mobile applications* running on a mobile device in the mobile domain. When considering the different possible locations of context producers and context consumers within a multiple DCM 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 "Smart Home" scenario, the Smart-Monitor application running at home queries the home DCM to learn if anyone is at home.
- *Cross-domain context producer-consumer pattern*: In this pattern, contexts produced in a domain can be consumed by applications of remote domains. For example, in the "Smart Home" scenario, the Smart-Monitor application can retrieve information from remote context managers (e.g., *Office-D* DCM, mobile DCM) to

determine which family member is available to “answer” the doorbell. In the “Community” and “Smart Campus” scenario, applications can query the location or status of Barbara or Tony’s friends from remote domains (e.g., mobile DCM, classroom DCM, hotel DCM). The mobile agent collects multimedia-related activities of Tony’s friends from other domains (e.g., friend-home DCM, KFC DCM, mobile DCM) in the “Entertainment Recommendation” scenario.

(2) Remote context query mechanism: As described above, for the cross-domain context producer-consumer pattern, an application running in a domain often needs to obtain contexts from remote domains. In terms of the complexity of accessibility, we categorize remote context queries into two types:

- *Explicit query*. For the contexts that are produced and maintained by a single domain, such as lighting-level information of a room and ongoing group activity information 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 they roam 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 also, at least partially, by his friendly “*guest or foreign domains*” (e.g., his friend’s home, his partner’s office, movie theatres, hotels). Since part of the information about the person, such as location, is overlapping amongst these domains, sometimes it is hard for a context consumer to specify where the most recent information about the person can be obtained. For example, in the “Smart Home” scenario, the Smart-Monitor application running at home has to check the availability of the out-of-home family members. But given that a person like David may be present in different environments, it is not possible for the application to specify the target remote DCM to fetch his status context. Similarly, in the “Entertainment Recommendation” scenario, a person may enjoy entertainments in different domains (e.g., watching TV at home, seeing a movie trailer on his mobile phone) within a day. 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. However, because the human often enters a foreign domain that is unknown to applications (like '*Office-P*' in the scenario), this solution can only handle a few situations. Therefore, for a remote query where it is hard to specify the target domain, one must provide a transparent mechanism that can route the query to the right DCM.

4. The iCROSS System

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

4.1. System Infrastructure

As illustrated in Fig. 1, the iCROSS infrastructure is broadly divided into three parts: *remote queries from applications*, *global administration server (GAS)*, and *domain context managers (DCMs)*.

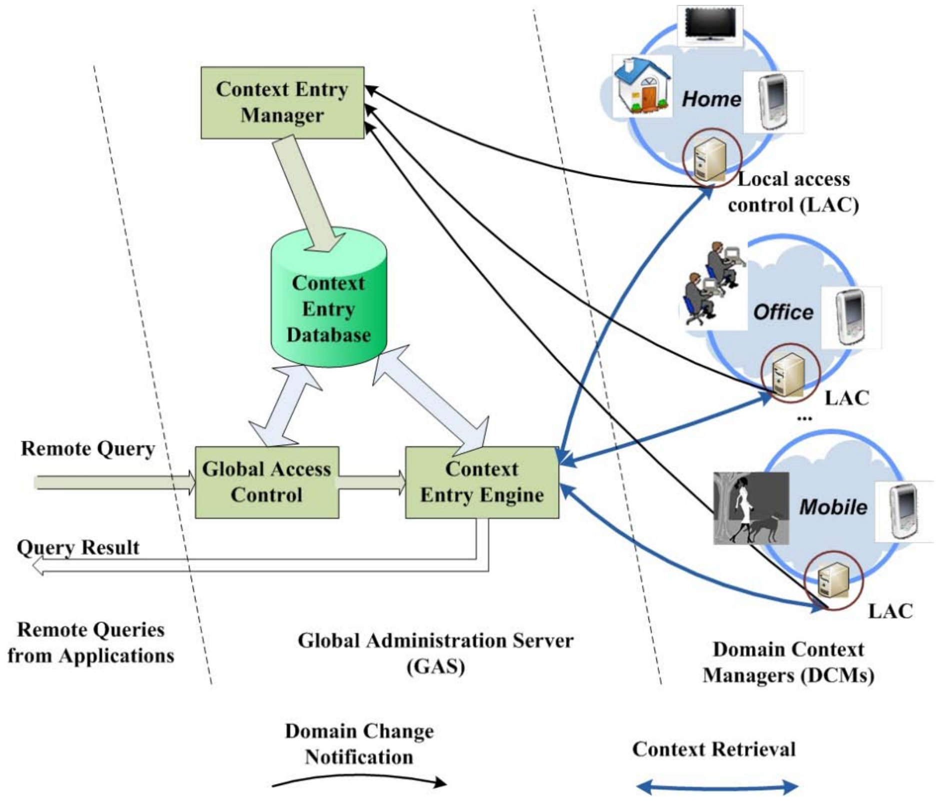

Fig. 1: The iCROSS system infrastructure

(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 application). An example that queries the status of a light in a remote domain is shown in Table 1 (Example 1).

Table 1: Query statement

Parameter	*Example 1*	*Example 2*	*Example 3*
Domain/User ID	d00001	u00001	u00001
Entity ID	light01	–	notebook01
Context Name	Status	Activity	Status
Requester Info	u00001	d00001	d00002

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 does not have to specify an explicit domain to obtain the context, instead, it only gives the id of a mobile entity. For example, when querying the context information of a user, the query statement should provide *User_ID*; while when querying the context about a user's belongings, the query statement should include both *User_ID* and *Entity_ID* (the id of the belonging). Two examples are given in Table 1, where Example 2 queries the activity of a user, and Example 3 queries the status of a user's notebook (a belonging).

(2) Global Administration Server (GAS). GAS is a global routing service running in a big service provider (like a telecommunication company) which redirects remote queries from applications to the correct DCMs. It consists of the following four components.

Context Entry Database (CED) is a global entry control database that manages two tables. One is the *access entry table (AET)*, which stores the entry point of the DCM indexed by the entity id (id of a domain or a user). We give a sample of this in Table 2. A user-id-indexed entry-point record indicates the current domain (registered or guest domain) where the user resides. The *user registration table (URT)* stores the entry point of all registered domains of a user.

Table 2: Access entry table (a sample)

Entity ID	***Entity Name***	***Access Point***
u00001	John	143.168.10.2/Access
u00002	John's Mother	143.167.10.3/Access
u00003	Alice	143.165.11.5/Access
d00001	John's Home	156.155.14.2/Access
d00002	John's Office	150.2.17.60/Access

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 specify the details of this in Section 5.1.

Context Entry Engine (CEE) is the core module that routes remote queries to the right DCM. For an explicit query request, CEE queries AET to retrieve the entry point of the target DCM. For a mobile entity query request, the challenge is how to reduce invalid DCM visits while maintain a reasonable query response time. A transparent query routing mechanism is proposed, the details of which is presented in Section 5.2.

Access control (AC) is another important issue in cross-domain context exchange. In the current design, there is *global access control (GAC)* on the GAS side and *local access control (LAC)* on the domain side. Since the access/privacy control function is outside of the scope of this paper, we will only discuss their common functions: GAC is in charge of preventing 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 they 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 the next subsection.

4.2. Domain Context Manager

The internal architecture of a domain context manager is shown in Fig. 2, which is based on our previous work [1, 20]. It consists of the following components.

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 [21] to represent contexts in CKB.

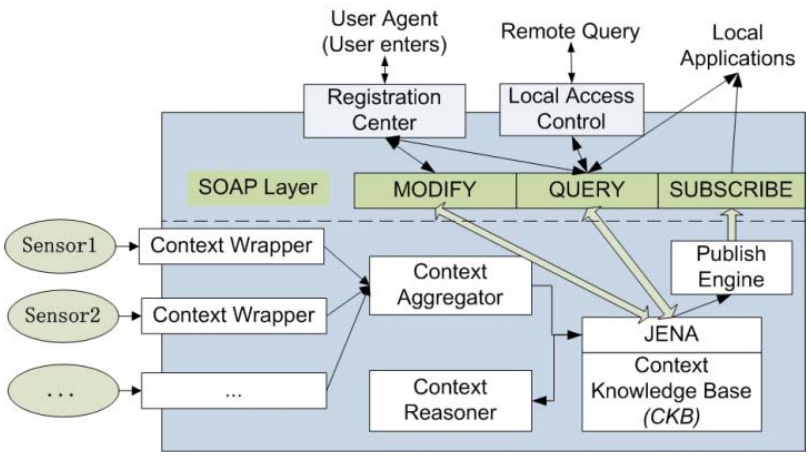

Fig. 2: Context manager inner architecture

When a user enters a domain, the *Registration Centre* of this domain will interact with this user to determine whether he/she is a guest or a registered user and if he/she 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 [22] for details), a subscription API that supports context subscription by local applications is also provided (our system does not consider remote subscription in the 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 complex, and we will describe it in the next section.

5. Cross-Domain Context Query

As presented in Section 3, there are two types of remote queries depending on whether the context consumer can clearly specify the target context manager. 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*.

5.1. Mobile Entity Registration

The mobile entity registration function supports user-entry-point updates when a user moves 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 most recent context of 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).

(1) 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/she would like to be served by this domain. The session stops if the user chooses not to be served. Otherwise, the registration process enters its next stage.

(2) 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).

(3) 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. When the user leaves a space, there are two cases:

- If he is a registered user, some out-of-date contexts about the user (e.g., location, activity) will be sent to "none";
- If he is a guest user, his temporary account and all related contexts will be removed.

5.2. Global Context Query Mechanism

The global context query mechanism provides a unified way for cross-domain context queries. Both *explicit query* and *mobile entity query* are supported by this mechanism. As illustrated in Fig. 3, the query mechanism is implemented through the following four stages.

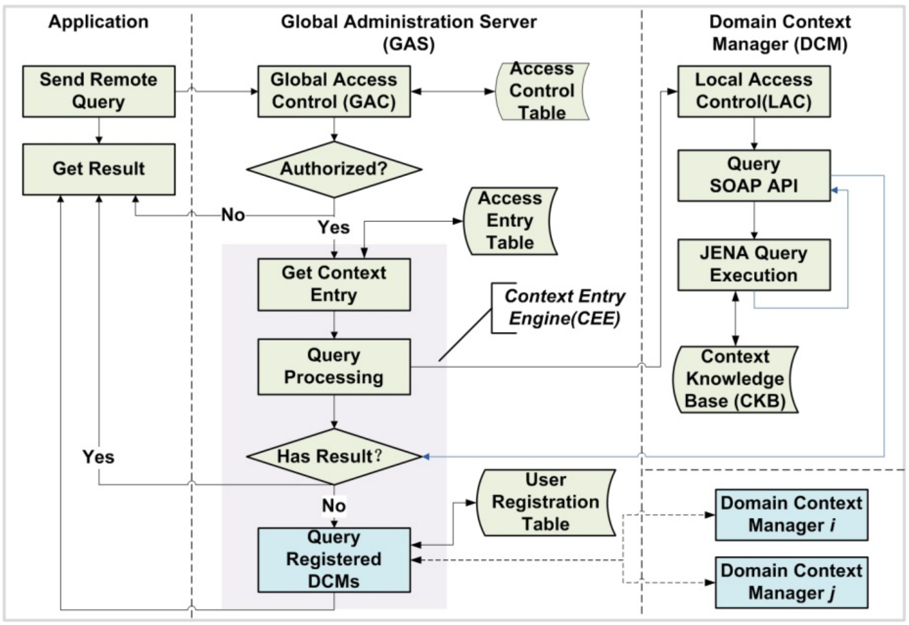

Fig. 3: Global context query process

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 information) 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 forwarding* stage, CEE forwards the query to the domain context manager according to the obtained entry point (we call it the entry domain context manager or, simply, EDCM). The local access control (LAC) component of DCM analyzes the query request and checks if it can be executed according to user privacy settings in LAC (certain contexts cannot be obtained by remote queries due to privacy considerations). If the query request is allowed, the query is executed and the result sent back to CEE. CEE analyzes the result and performs the following actions:

- If the result is not null, it sends the result to the application and the query session stops;
- If the result is null and the query is a mobile entity query (i.e., the first parameter is a user id), CEE will do further queries from the registered domains of the user. This is because some contexts (such as his biomedical information) about the user may not be stored by the domain he is currently in (e.g., his friend’s home), but are stored in several registered domains of his (e.g., his home or company).

In the *RDCM based query forwarding* stage, CEE will search the user registration table to retrieve other registered domains of the user, and forward the query simultaneously to all the registered DCMs (RDCMs) for the target context. We call it the “simultaneous-multicast”. Finally, CEE sends the query result to the query publisher (e.g., an application or a service) and the query session stops.

The mechanism mentioned above provides a unified way for cross-domain context query. This query process is transparent to mobile entity queries. Furthermore, the mechanism proposed allows applications to fetch the most recent context information about a mobile entity, as well as ensuring that we can find the correct storage place of the context when it is not managed by the domain the user is currently in. It should be noted that the term “context” used in this paper refers to “fresh” while not “historical” information about an entity. Fresh information appears merely in one domain, while historical context can be reserved in several different domains. For example, if we want to query “what entertainments does Tony enjoy on Sunday” in the “Entertainment Recommendation” scenario, we should multi-cast the query to the EDCM and all RDCMs at the same time. That’s because multimedia-related activities can happen at different domains (e.g., at home, on the mobile phone, in a cinema, and so on) within a long period. In other words, if we make a small variation to the above algorithm, combining the *entry-point based* and *RDCM based* query forwarding stage (to a unified multi-cast forwarding stage), our algorithm can also deal with queries about historical contexts.

6. Scenario Implementation and Evaluation

Having described the iCROSS infrastructure, in this section, we describe how it supports cross-domain context query in our ongoing projects. Specifically, we present the implementation of two scenarios described earlier in this paper. Initial experiments have also been conducted to measure the performance of our system.

6.1. The Smart Home Prototype

To demonstrate the key features of iCROSS, we have implemented a prototype of the Smart Home scenario (as described in the introduction). The design of it is shown in Fig. 4. In the prototype, we have the GAS server and the home DCM located in our lab, and three DCM clients (including two office DCMs and a mobile DCM) located at students’ home. Each DCM maintains the contexts generated locally, and stores them in the context knowledge base (CKB, see Section 4.2 for details). The CKB is developed based on the *SS-ONT* ontology model we designed for smart environments [23]. In the initial implementation, each CKB in a DCM consists of around 1000 OWL triples. For the purpose of demo, some of the contexts identified in the scenario are pre-specified in the context ontology, such as the doorbell ringing context, human activity context (e.g., Barbara is walking the dog, David is having a meeting), and so on. The whole working process of the scenario is illustrated through a UML sequence diagram shown in Fig. 5.

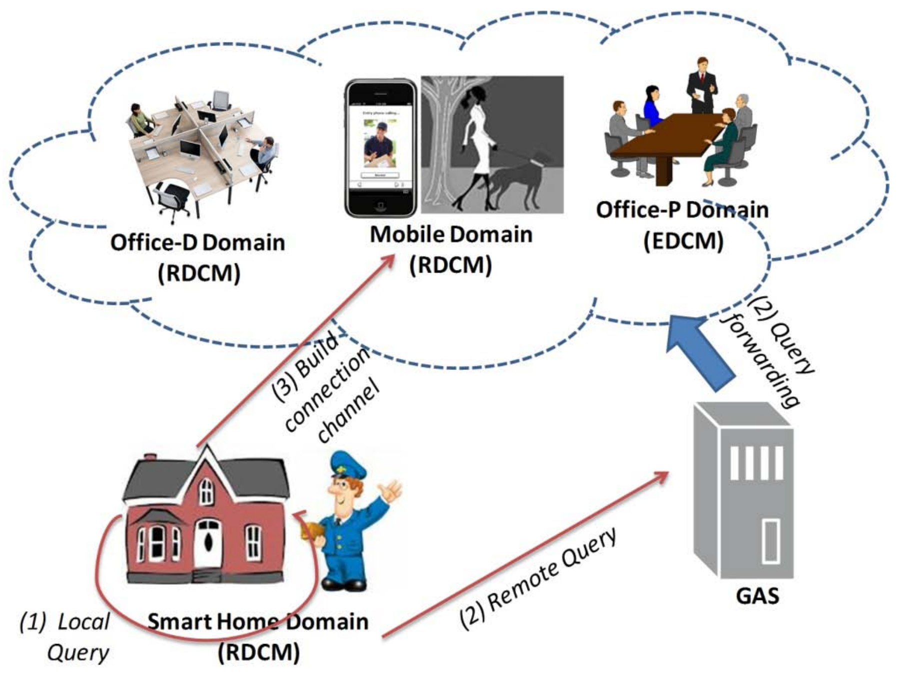

Fig. 4: The design of the smart home scenario

When the postman arrives, the home DCM acquires this context through the ringing doorbell. Since this context has been subscribed by the local Smart-Monitor application (SMA), it will publish it to this application. By querying the home DCM and learns that nobody is at home, SMA sends remote queries to GAS to check the current activity of out-of-home family members, which can be used to learn who is available to answer the doorbell. The first remote query is about David. CEE (see Section 4.1) at GAS firstly forwards the query to the *Office-P* DCM according to the record from the entry point table (David is currently there). However, because *Office-P* only provides location context for remote queries, it does not provide any activity information about David. CEE then redirects the query to the registered context managers of David, including his office (*Office-D*) and the mobile DCM, and sends the final results to SMA. According to the electric calendar information from David's office, SMA finds out that he is having a meeting in another place and thus is not available. Another remote query is sent to fetch Barbara's status. By the entry-point-based query, SMA learns that Barbara is walking in a park and she is available to answer the doorbell. Finally, a text-based communication channel is established between Barbara and the postman (in the current stage, we did not build the video-based connection).

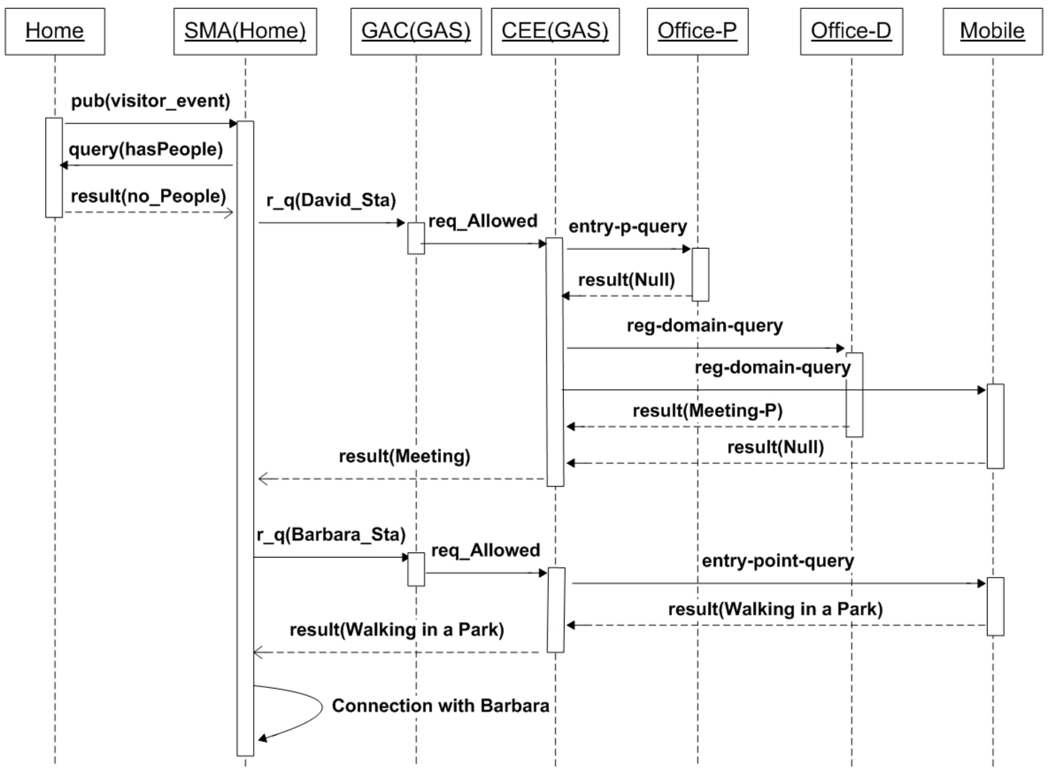

Fig. 5: The smart home scenario analysis

We have also conducted experiments to validate the performance of iCROSS based on this prototype environment, as described later in Section 6.3.

6.2. Community Services in a Smart Campus

To benefit student life and social connection in university campuses, we have designed and implemented the Smart Campus prototype (see Section 3). The prototype includes several community services, one of which is *Where2Study*. The main purpose of *Where2Study* is to find a suitable place to study by using Wi-Fi positioning and mobile social networking techniques [24, 25]. It not only supports students to query the status and locate their friends in the university campus (left of Fig. 6), but also shows the status of the classrooms (full or free seats available), as shown in the right of Fig. 6. In the prototype, each classroom can have a domain context manager, which manages the contexts generated in it, such as number of free seats, the status of devices in it, people in the classroom, and so on. For a student *S1*, the classroom he/she resides can be viewed as a foreign or guest domain. It becomes the entry DCM (EDCM) of *S1*, which is registered in the GAS of the smart campus. Another student *S2*, can remotely query *S1*'s location if they are friends. At the same time, the status of the space *S1* situated in can also be obtained through cross-domain context query. In the prototype, the mobile client user interface shown in Fig. 6 is developed on the Samsung i909 Android platform.

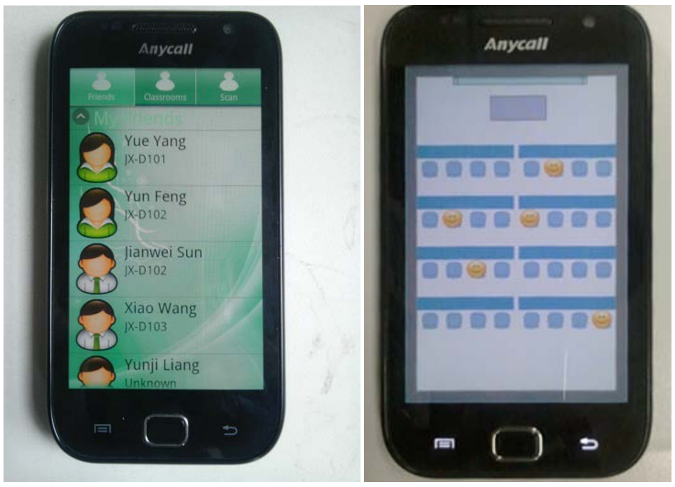

Fig. 6: Screenshots of the Where2Study application

As shown above, the key feature of this application is the capability to browse the status (e.g., location, activity) of human friends within different domains. This allows users to reach out its individual space and be aware of their social ties in a campus-wide environment, which will enhance social interaction and domain context sharing (e.g., the status of a classroom). In addition, when a user encounters a problem during study, he or she could turn to their friends for discussion according to their location obtained by remote queries.

6.3. Initial Evaluation

As a cross-domain query system, query latency is an important parameter to validate the performance of iCROSS. We have conducted experiments to measure the basic “one-round” query latency, i.e., *the time latency between a query is transmitted from the GAS to a DCM and the query result is sent back to the GAS*. Technically speaking, it comprises of the *data-transmission time* and *local-context-query time*, which is measured by the following way.

- First, we measured the data-transmission (a query packet with the size of 100 Bytes) time from the client-domain node to the server node, and obtained the average “one-way” data transmission time (0.1s) after fifty runs.

- Second, we tested the local-context-query time on a DCM (the context ontology size is around 1000 OWL triples), and figured out the mean (0.3s) after fifty runs.
- The average one-round query latency is thus 0.5s ($0.1\mathrm{s} * 2 + 0.3\mathrm{s}$).

As described in Section 5, the cross-domain query mechanism we proposed can be viewed as a two-stage query routing method. The method first forwards an incoming query Q to EDCM; if there is no matched query result, it transmits Q to all the remaining RDCMs simultaneously. Therefore, our query mechanism is at most a "two-round" query process. In other words, if Q can be resolved in the first stage/round, the query time should be 0.5s; otherwise, it doubles to 1.0s. The number of query rounds, however, is affected by the probability that the target DCM (where the target context is kept) is the same as the entry DCM (EDCM). Assume the *probability parameter* is P, if P is 80%, it means that the target DCM will choose from EDCM with 80% probability, and will choose from all the other DCMs with 20% probability. We thus have the expected value for the query time (Qtime) given by:

$$\mathrm{E[Qtime]} = 0.5 * \mathrm{P} + (1 - \mathrm{P}) * 1.0 \qquad (1)$$

It can be transformed to:

$$\mathrm{E[Qtime]} = 1 - 0.5 * \mathrm{P} \qquad (2)$$

Formula (2) reveals that Qtime decreases when P increases, and when P increases to 100%, we have the minimum Qtime− 0.5s. It is because that when P increases, there is higher probability that the target context is accompanying the user (i.e., in the EDCM), and thus decreases the time cost on the second-round routing (to RDCMs). It should also be noted that the query time can be affected by the network condition and the size of the ontology [20], however, we consider that a 1-2 seconds delay is tolerable by most context-aware applications.

7. Conclusion

This paper reports our early effort on cross-domain context management. Two context producer-consumer patterns, *intra-domain* and *cross-domain*, are identified and supported by our proposed iCROSS context management infrastructure. Our system further addresses the mobile entity related context acquisition problem, and provides a global routing mechanism to support the effective acquisition of context information about mobile entities from remote domains in a transparent manner. We have also built two scenarios to demonstrate the features and test the performance of iCROSS.

Our work on cross-domain context management is ongoing. Future work includes the integration of robust access/privacy control mechanisms and the combination of advanced query processing algorithms leveraged by distributed systems. We will also exploit the programming paradigm for context-aware app development in multi-domain environments. A programming platform that facilitates user collaboration among different smart domains has been reported in [23]. Context inconsistency and uncertainty issues will evolve differently in multi-domain environments. As mentioned in Section 2, it will become another direction of our future research.

References

[1] X.H. Wang, D.Q. Zhang, J.S. Dong, C.Y. Chin, S. Hettiarachchi. Semantic Space: An infrastructure for smart spaces. *IEEE Pervasive Computing*, Vol. 3 No. 3, 2004, pp.32-39.

[2] B. Guo, R. Fujimura, D.Q. Zhang, M. Imai. Design-in-Play: Improving the Variability of Indoor Pervasive Games. *Multimedia Tools and Applications*, Vol. 59 No. 1, 2012, pp. 259-277.

[3] N. Streitz. Smart Artefacts as Affordances for Awareness in Distributed Teams. *The Disappearing Computer*, Springer, 2007, pp. 3-29.

[4] D. Sanchez, M. Tentori, J. Favela. Activity recognition for the smart hospital. *IEEE Intelligent Systems*, 23, 2008, pp. 50–57.

[5] Javier Bajo et al., SHOMAS: Intelligent guidance and suggestions in shopping centres. *Applied Soft Computing*, Vol 9 No. 2, 2009, pp. 851-862.

[6] Y. Shi, W. Xie, G. Xu, R. Shi, E. Chen, Y. Mao, F. Liu. The Smart Classroom: Merging Technologies for Seamless Tele-education. *IEEE Pervasive Computing*, 2003, pp. 47-55.

[7] Z. Wu, Q. Wu, H. Cheng, G. Pan, M. Zhao, J. Sun. ScudWare: A semantic and adaptive middleware platform for smart vehicle space. *IEEE Trans. on Intelligent Transportation Systems*, Vol. 8 No.1, 2007, pp. 121-132.

[8] D.Q. Zhang, B. Guo, Z.W. Yu. Social and Community Intelligence. *IEEE Computer*, Vol. 44 No. 7, 2011, pp 22-29.

[9] The EU FeelAtHome Project, http://www.celtic-initiative.org/Projects/Celtic-projects/Call5/FEEL@HOME/feelhome-default.asp.

[10] D. Salber, A.K. Dey, G.D. Abowd. The Context Toolkit: Aiding the development of context-enabled applications. In: Proc. of CHI'99, 1999, pp. 434-441.

[11] H. Chen, T. Finin, A. Joshi, F. Perich, D. Chakraborty, L. Kagal. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, Vol.19 No. 5, 2004, pp. 69–79.

[12] Bettinia, C., Brdiczkab, O., Henricksenc, K., Indulskad, J., Nicklase, D., Ranganathanf, A., Riboni, D.: A survey of Context Modelling and Reasoning Techniques. *Pervasive and Mobile Computing*, Vol. 6 No.2, 2010, pp. 161-180.

[13] W. Wibisono, S. Ling, A. Zaslavsky. Collaborative context management framework for mobile ad hoc network environments, In: Proc of the 2010 ACM Symposium on Applied Computing (SAC-10), 2010, pp. 558-562.

[14] C. Chen, C. Ye, H. Jacobsen. Hybrid Context Inconsistency Resolution for Context-Aware Services. In: Proc. 9th IEEE Conf. on Pervasive Computing and Communications (PerCom 2011), 2011, pp. 10-19.

[15] V. Degeler, A. Lazovik. Interpretation of Inconsistencies via Context Consistency Diagrams. In: Proc. 9th IEEE Conf. on Pervasive Computing and Communications (PerCom 2011), 2011.

[16] 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, 2005.

[17] R. José, F. Meneses, A. Moreira. Integrated Context Management for Multi-domain Pervasive Environments. In: Prof. of the First International Workshop on Managing Context Information in Mobile and Pervasive Environments, 2005.

[18] C. Hesselman et al. Controlled Disclosure of Context Information across Ubiquitous Computing Domains. In: Proc. of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 2008, pp. 98-105.
[19] I. Roussaki, M. Strimpakou, C. Pils, N. Kalatzis, M. Neubauer, C. Hauser, M. Anagnostou. Privacy-Aware Modelling and Distribution of Context Information in Pervasive Service Provision. IEEE International Conference on Pervasive Services (ICPS2006), 2006, pp. 150-160.
[20] B. Guo, S. Satake, M. Imai. Lowering the barriers to participation in the development of human-artifact interaction systems. *International Journal of Semantic Computing (IJSC),* Vol. 4 No. 2, 2008.
[21] Web Ontology Language, http://www.w3.org/2004/OWL/.
[22] ActiveMQ, http://activemq.apache.org/.
[23] B. Guo, D.Q. Zhang, M. Imai. Towards a Cooperative Programming Framework for Context-Aware Applications. *Personal and Ubiquitous Computing*, Vol. 15 No. 3, 2010, pp. 221-233.
[24] Q. Yang, S.J. Pan, V. W. Zheng. Estimating location using Wi-Fi. *IEEE Intelligent Systems,* Vol. 23 No. 1, 2008, pp.8-13.
[25] Z. Yu, Y. Liang, B. Xu, Y. Yang, B. Guo. Towards a Smart Campus with Mobile Social Networking. The 2011 IEEE International Conference on Internet of Things (iThings 2011), Dalian, China, 2011.