Context and Location-Aware Encryption for Pervasive Computing Environments

Jalal Al-Muhtadi[*], Raquel Hill[†], Roy Campbell[‡], M. Dennis Mickunas[‡]
[*]*King Saud University, Saudi Arabia*
jalal@ccis.ksu.edu.sa
[†]*Indiana University, USA*
rahill@cs.indiana.edu
[‡]*University of Illinois at Urbana-Champaign, USA*
{rhc, mickunas}@cs.uiuc.edu

Abstract

Pervasive computing promises to revolutionize computing, empower mobile users, and enhance mobility, customizability and adaptability of computing environments. Intrinsic to the notion of such environments is the capturing of location and context information. Context awareness and validation enables significant functionality to pervasive computing applications, users, resources and the ways they interact. Much of this functionality depends on validating context information and using it for granting access to data or resources. In this paper we propose an encryption and access control framework that uses both context and identity to determine whether an entity or a group of entities may access protected services, data, devices, and other resources. We assume that the resources are context-sensitive, thus requiring the requesting entity to be under a specific context before it is able to access the resource or decrypt the information. Our approach is unique in the way that we decouple context from identity, which adds extra security, facilitates value-added services, and enables efficient key management for group communication.

1. Introduction

Pervasive Computing advocates the creation of "active spaces," where the computational infrastructure is integrated with the physical surroundings. One key feature of an active space is its ability to track all people and equipment within the environment and capture contextual information. This capability inspires researchers to investigate ways in which location information can be best utilized. Examples of such applications include location-based value-added services or "follow-me" applications and video streams. Location awareness is currently an active research area [1, 2]. However, only few research efforts have focused on providing location-aware security services. Some researchers investigated different notions of location-based access control [3].

We believe that location awareness can provide traditional security mechanisms with greater flexibility and expressive power. Our goal is to devise and experiment with mechanisms that enable real-time capturing and fusion of location data, while enforcing appropriate security policies.

In pervasive computing environments, contextual changes may trigger a change in a user's access permissions. Therefore, one specific challenge to providing location and context-aware security services is the efficient integration of contextual triggers into the authorization mechanism. Prior work [4] defines an authorization framework for integrating these contextual triggers. In this work, we extend the use of context to create location-based security mechanisms for providing confidentiality and restricting access. Another prior work [5] establishes location-based keys for static nodes. Each node is given its location-based key and its geographic location is used as its identity when communicating with adjacent nodes. In contrast we use a node's location to authorize access to a resource. We assume that a user or node's location is dynamic. Administrative nodes within an active space use the location-based keys to perform cryptographic functions on the behalf of the user. The key generation and authentication mechanisms that are described in [5] can be used to generate location-based keys for administrative nodes within active spaces.

In this paper, we present a framework for providing location-based encryption. We extend this framework and illustrate how it can be used to provide efficient key management for secure group communication. Throughout this paper, we define context as "any information that can be used to characterize the situation of an entity." [6]. We consider location information to be part of the context. In this paper we focus specifi-

cally on location information and argue that the same ideas can be generalized for other context information.

The remainder of this paper is organized as follows. Section 2 describes our experimental environment. Section 3 gives an overview of the proposed system and how it is integrated with two key components in our pervasive computing environment. Section 4 describes some enhancements and optimizations to the proposed system. Section 5 describes the implementation briefly. Section 6 provides some evaluations. Finally, Section 7 concludes.

2. Experimental Environment

To build a prototype pervasive computing environment, we developed Gaia [7]. Gaia is a generic computational environment that integrates physical spaces and computational infrastructures into an active space. Gaia is a meta-OS that utilizes a layered architecture as illustrated in Figure 1. The low-level layer consists of the basic functionality, which includes component management, resource discovery, security, file management, context service, location service and other core services necessary for any general-purpose distributed computing environment. We refer to this layer as Gaia's kernel, as it resembles a kernel in a traditional OS. Building applications using only the Gaia low-level services is tedious. The application layer of Gaia provides a framework that automates the common patterns for creating and managing typical pervasive computing applications and their components.

A key issue in pervasive computing is to provide a method to obtain accurate locations of objects and people, in order to provide greater customizability and seamless interactions. We designed our encryption and access control framework with the hope of using Ubisense location technology [8] to obtain accurate 3D location information. Ubisense is a unidirectional UWB (Ultra Wide Band) location platform that uses a bidirectional TDMA (Time Division Multiple Access) control channel. RFID tags transmit UWB signals to networked readers and are located using "angle of arrival" and "difference of arrival" techniques. Ubisense has an accuracy of 6 inches with 95% confidence.

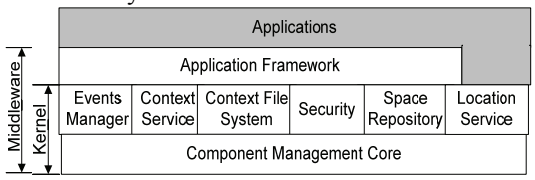

Figure 1 – Gaia Layered Architecture

The Ubisense tag has two programmable LED displays and two programmable buttons. At the time of this writing, Ubisense is in the experimentation phase and only a prototype of the system is available for testing. We assume that the Ubisense system is able to provide unforgeable location data. To achieve this, it is possible to employ techniques similar to those used by Denning et al. [9] to build a system that uses GPS to provide unforgeable location signatures from a remote device, by using the signal instabilities added by the US Department of Defense to prevent spoofing. Alternatively, it is possible to build tamper-resistant Ubisense tags that are assigned digital certificates. The bidirectional TDMA channel can be used to perform a challenge-response in order to authenticate the tag. Information transmitted by the tags can be digitally signed. Ubisense readers can then check the validity of the data to prevent the forging of location information. Authentication techniques and tamper-resistant tags are needed to prevent spoofing.

Although Ubisense provides relatively accurate location information, it is not currently available in all active spaces. Moreover, it is possible that other accurate location technologies are deployed in the future. Therefore, to be independent of the location technology being used, we rely on the MiddleWhere location infrastructure [10]. MiddleWhere provides a middleware layer that stores and updates location information in real-time. It aggregates location information from various location technologies and provides a uniform interface for obtaining location information. Additionally, it includes information about the accuracy of the location sensing devices.

3. System Overview

In order to enable the functionality and scenarios described earlier, we incorporate support for the location-based security protocols into two core services in Gaia. The Gaia Context File System [11], and the Event Manager [12], which implements Gaia's Publish/Subscribe system.

3.1. Gaia Context File System (*CFS*)

Context-awareness is a key issue in pervasive computing, and many of the core services that a traditional operating system provides need to be enhanced with context-awareness in order to reap the full benefits of pervasive computing. Gaia provides a file system that is context-aware [13]. Context is associated with files and directories and is used to limit the scope of available data to what is important for the current task, aggregate related material and trigger data type conversions, therefore simplifying the tasks of application developers and users of the system. For example, a teaching assistant can develop a multimedia presentation using material from a laptop and home and office computers. When the assistant enters the active space, the context file system can aggregate the material into a folder that is mounted conveniently within the active space, allowing the assistant to activate a Gaia session to begin the presentation. If the context is not right, this aggregation of data does not take place, and thus, the session is not available. The original context file

system does not employ cryptographic techniques to prevent attackers from accessing the material directly from the machines providing the storage space. It was initially created to provide a method for organizing files, so that the relevant material can be seen and aggregated properly, while hiding the irrelevant material. In this paper, we extend the file system, so that the material is stored in an encrypted fashion, and can be aggregated and decrypted only when the requestor entity is at the correct location or under the correct context.

Encrypting files mitigates the need for complicated access control mechanisms or reference monitors that mediate every attempted access by a user. This advantage is even more crucial in a pervasive computing environment, where it is common to have different pieces of data stored on a plethora of different devices with various capabilities and processing powers. These can include lightweight devices such as PDAs and smartphones. In such a setting, it is infeasible to implement sophisticated access control checks on these devices. In our approach we extend the notion of encrypting files further by introducing location-based encryption, while preserving backward compatibility.

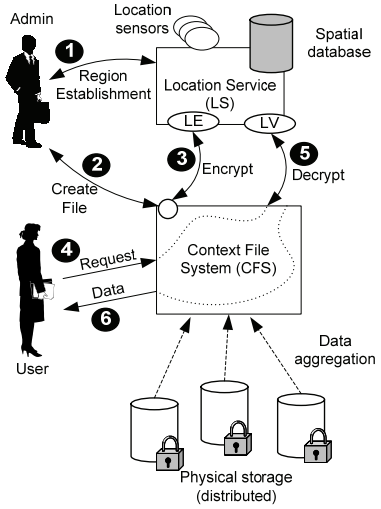

Figure 2: Incorporating Location-Encryption into Gaia's Context File System

The location-based encryption add-on for the Gaia context file system is illustrated in Figure 2. For simplicity, we assume a scenario in which an administrator or an authorized entity creates a spatial region and corresponding files, so that only users located within that region can read these files. We achieve this by performing location-based encryption on the files. Spatial regions in our system are represented in a hierarchical format. The representation could be symbolic, e.g., *SC/3/room-3216*, (i.e., room 3216, 3rd floor, SC building) or coordinate-based, e.g., *SC/3/(45,12), (45,40), (65,40), (65,12)* which represents a specific region in the 3rd floor in building SC. There are six steps involved in setting up and utilizing location-based encryption. In step 1, the administrator sets up the spatial region(s) in which data access is authorized. The following three messages are exchanged:

$$AD \rightarrow LS : \{D = (CREATE\_REGION, LS, R, N_{AD}, ID(AD)), < H(D) >_{PK_{AD}^{-1}}\}_{PK_{LS}}$$

'AD' is the administrator in Figure 2. 'LS' is the location service. 'R' is the region for the location-based encryption N_{AD} is a nonce generated by the admin. *ID(AD)* is the identity certificate of the admin. *H(D)* is a hash of the message *D* and it is used to verify message integrity. The message *D* is signed by the administrator and encrypted with *LS* public key. This step will result in the generation of a location key K_R for region 'R'. This key is only known to the location service; thus, no one else is able to decrypt. Currently, we are using a secure one-way function that converts region boundaries into a secret key. Identical regions generate the same key. The location service replies with the following messages.

$$LS \rightarrow AD : \{D = (OK, R_{ID}, CFS, N_{AD}, N_{LS}, K_{LS/AD}), < H(D) >_{PK_{LS}^{-1}}\}_{PK_{AD}}$$

Where R_{ID} is a region ID that uniquely identifies region R. $K_{LS/AD}$ is a session key shared between the LS and the admin for further communication. Finally the admin acknowledges:

$$AD \rightarrow LS : \{D = (ACK, LS, N_{LS}), < H(D) >_{K_{LS/AD}}\}_{K_{LS/AD}}$$

At this point, a secure session is established between the administrator and the LS. This secure session can optionally be used to establish more regions without the need to perform any more expensive asymmetric cryptography. Once the regions are set up, the administrator establishes a secure connection with the context file system (*CFS*) in a manner similar to what was described above. This would result in the generation of the session key $K_{AD/CFS}$. To create location-encrypted files, (step 2), the data is sent to the *CFS*. The file creation message indicates that this is a location-encrypted file, and provides an ID for the target region. Upon receiving this request, the data is passed to a Location Encryptor (*LE*) (step 3) which is a component spawned by the LS. The *LE* verifies that the target region is established and that the requestor is authorized. If successful, the *LE* is able to retrieve the secret key (K_R), which corresponds to the target region (*R*). The data is then encrypted using K_R, and the result is sent back to the *CFS*: $LE \rightarrow CFS : \{OK, N_{CFS}, \{DATA\}_{K_R}\}_{K_{LE/CFS}}$

Where $\{DATA\}_{KR}$ is the data encrypted with the location key of region 'R'. $K_{LE/CFS}$ is the session key for the communication channel between *LE* and *CFS*. The files system can store this anywhere in the distributed storage. Note that only the *LS* can decrypt this data because it is the only component that knows K_R.

In step 4 in Figure 2, a user makes a request to access a file. If the target file is a regular file, then the

CFS provides the same functionality as before. If the target file is location-encrypted, then the request is intercepted at the client side, and the identity certificate of the caller is injected. In this case, the attributes of the target file will indicate that the file is location-encrypted with region R_{ID}. The *CFS* sends a request to another component spawned by the *LS* called the Location Verifier (*LV*). The *LV* verifies that the location of the requestor is within the region *R* with probability *p* or higher. If successful, the data is decrypted (step 5), and sent back to the user (step 6), otherwise, the *LV* throws an exception, and, in turn, the *CFS* throws an exception back to the requestor. The value *p* is dependent on the application or the usage scenario. This value utilizes the MiddleWhere infrastructure's capability of attaching a confidence value of the location reading, which takes into account the accuracy and the number of matching location readings (in case of more than one location reading is detected).

Depending on the application and the scenario at hand, we identify two useful modes for writing location-bound files. (1) *Read-only*, where only an authorized entity can create and modify location-based files, while other authorized users are only allowed to read the files, if they are located in region R. This mode is illustrated in Figure 2 and described above. The Read-only mode is useful for a class examination situation where only students who are physically present in the lecture hall can access the exam file. However, they cannot create other files. (2) *Location-bound write permission,* where authorized users are allowed to write to the file only when they are located within the region *R*. In such a situation, we proceed as described earlier, except that a user may request to modify a file, in which case, the location of the user needs to be verified before the file is location-encrypted.

In many scenarios, we observe that restricting access based on location only is not enough. For example, in the class exam scenario, the exam data should be made available for the *class students* who are physically present at the lecture hall. Thus, in addition to validating location, the users or entities need to be members in a specific role (i.e. student). For this reason, we introduce the notion of a multi-layer encryption. In this case, the data is encapsulated within two layers of encryption as depicted in Figure 3. Users are authenticated, and if they are members in a certain role (or have a certain attribute) they are given a group key (K_g). In addition, the users need to be within the target region. If users are located within region R, then the *CFS* will be able to peel off the first layer of encryption. Then the user receives the data over a secure session between him and the *CFS*. The data in this case will be wrapped with a single layer of encryption that requires K_g for decryption (i.e. the user must be a member in the group *g*). Only then the user can decrypt the data. In other words, the data is stored in *CFS* distributed storage as: $\{\{ data \}_{Kg}\}_{KR}$

If the user is located in the correct location, then the encryption with K_R is peeled off by the LS, and $\{data\}_{Kg}$ is sent back to the requestor, which can be decrypted if the requestor is a member of the group *g*.

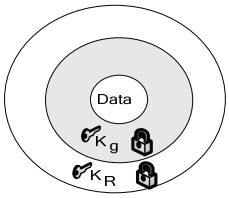

Figure 3: Multi-layer Encryption

3.2. Gaia's Publish/Subscribe System

Intrinsic to the notion of active spaces is the ability to deploy an efficient technique for dispersing data and information relevant to the space to the devices, applications and services that reside in the space. In Gaia, the underlying communication between the different entities is facilitated by a communication abstraction called an 'event channel.' These event channels are managed by the Event Manager Service and are implemented as publish/subscribe channels [12, 14]. Publish/subscribe systems offer completely asynchronous, decoupled message transmission. Publishers send messages to a channel without explicit knowledge of details of who may be subscribed to the channel. By subscribing to a channel, a subscriber is essentially registering for notifications of new messages. Rather than polling the state of device's resources or the state of a sensor, interested clients register to be notified when the state of that resource or sensor changes.

Some resources in an active space send sensitive information, like user tracking data or login audits, etc. Therefore, in previous work, a secure communication infrastructure that provides guarantees about subscriber authorization, data integrity, and confidentiality in an active space was developed [4]. Since active spaces are physically bound, there is an additional requirement to limit access to the channels based on physical location. Devices moving out of range should not be able to receive the data sent on these location-bounded channels. Moreover, unauthorized entities should not be able to transmit data on these channels or receive unencrypted data from these channels. While the secure version of the event service [4] can provide encrypted channels to prevent unauthorized access, if a subscriber leaves the current location, expensive re-keying needs to take place to prevent unauthorized access.

We briefly describe how the extended context-aware publish/subscribe works (Figure 4). The enhanced version is fully backward compatible, as publishers can still create regular unsecured channels, or role-based secure channels. However, in the new system a publisher (*P*) can choose to create a publish/subscribe channel that is location-bound. An Event Broker (*EB*)

is initialized for each publish/subscribe channel to facilitate the creation and management of a group key and to verify that subscribers are authorized according to their credentials, role names, or attributes. Based on this, the group key is generated and dispersed securely to the authorized members. This secure dispersion can take place using an off-line communication mechanism, or by authenticating members and establishing a temporary secure session in a fashion similar to that described in Section 3. *P* also sets up a region *R*, where all subscribers must reside to be able to receive the events. *P* establishes a secure session with EB. *P* pushes events into the channel by sending them over the secure session to *EB*. Depending on the requirements of *P*, the *EB* could encrypt the data with a group key K_g in order to limit access to this data to entities that belong to a particular group. By performing the group key encryption at the *EB*, *P* does not need to deal with issues pertaining to group management and re-keying. Furthermore, note that *P* may actually be a lightweight device, e.g., a smart door lock mechanism that sends data to authorized devices in the room about the card ID of the person trying to enter this room. Next, as described earlier for the context file system, events are passed by the Location Encryptor (*LE*) component of the *LS*, which encrypts the event data with the secret region key K_R. Afterwards, the data in the form of $\{\{ data \}_{Kg} \}_{KR}$ is sent over the actual event channel (Figure 4).

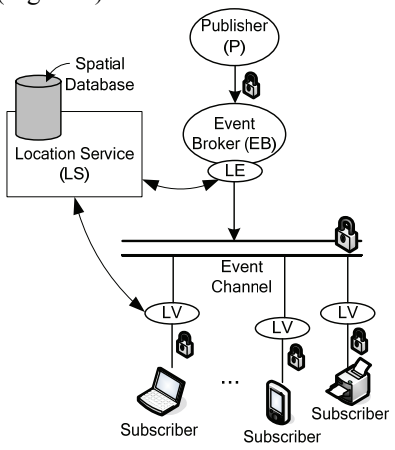

Figure 4: Publish/Subscribe System with Location

On the subscriber side, each subscriber is authenticated to ensure that it belongs to the authorized group. If the authentication is successful, the subscriber is provided with the group key K_g. On behalf of every subscriber, a Location Verifier (*LV*) is spawned, which intercepts the event data sent to the subscriber, and attempts to location-decrypt it, only if the corresponding subscriber is located within the target region *R* with a specific confidence value.

Note the significance of the two layers of encryption; a device that is not authorized to receive the information cannot decrypt the data if it attempts to eavesdrop on the event channel even if it was located within the region *R*. A device that is authorized, but is not located in the target region *R* cannot peel the outer layer encryption. This location-based publish/subscribe system along with other traditional secure group communication mechanisms are vulnerable to collusion-based attacks (e.g., a subscriber leaking info to a non-subscriber).

4. System Enhancements

4.1. Addressing the overhead of Re-Keying

Prior work [4] explores the use of a group key management strategy for re-keying in pervasive computing environments. Given this strategy, the management entity defines permission levels based on attributes, e.g., groups, roles, security clearance, etc. It stores a single key for each defined level. When the permissions of any entity associated with specific permissions levels changes, only the keys for the affected levels are revoked. The overhead of this re-keying includes the generation of a new group key plus the distribution of this key to all members of the group or permissions level. The distribution process as defined by Lee et al [4] requires that each member of the group re-authenticate itself before receiving the new key. While re-authenticating each entity ensures the correctness of the authorization mechanism, it increases the overhead for the management entity. Our use of multi-layer encryption reduces the overhead of revocation by eliminating the need for entities within a group to re-authenticate when a contextual trigger changes the permissions for the group. In [4] the departure of a group member would initiate key revocation and the re-authentication of the remaining group members. Given our multi-layer encryption scheme, the location encryption layer is peeled only after an entity's location has been verified. Therefore, a group member who leaves the active space is unable to decrypt the data with its group key because the location encryption layer is present. Thus, re-keying of the remaining group members is not necessary.

4.2. Distributing the Load

The location service generates keys for specific regions, verifies the location of entities, and encrypts and decrypts data per region attributes. We distribute the load of the location service among multiple location servers within an active space. We assume that an active space encompasses one room, multiple rooms, multiple floors or an entire building. When an entity specifies a region from which its protected resource can be accessed, a location server is selected to manage the requests for this resource. The current load and load history of the location servers are used to select a server with the appropriate load.

5. Implementation

We utilize CORBA [15] for implementing the various components in the system, including the *LV*, *LE*, and *EB*. CORBA provides primitives for discovering, managing, and communicating among distributed objects. For lightweight devices, we use the Universal Interoperable Core (UIC), which provides a lightweight, efficient implementation of basic CORBA services [16], which enable commodity devices to participate in pervasive computing. The secure publish-subscribe channels are implemented above CORBA event channels [14].

6. Evaluation

We evaluated a simple scenario in which we used Bluetooth discovery as a location detection mechanism. We defined a simple spatial region that approximately covers the Bluetooth base station's range. We used AES 128-bit and 256-bit to perform the multi-layer encryption. Cryptographic performance for different devices is illustrated in Table 1. Figure 5 illustrates the latencies measured when creating a location-aware event channel, where packets are encrypted with two layers of encryption, one for group membership and the other for location information. The two layers use AES 128-bit. Note that the increase in latency is minimal when the number of subscribers increases.

Table 1: Cryptographic performances

Device	AES 128-bit performance	AES 256-bit performance
Pentium™ 4 processor @ 1.7 GHz, Windows™ XP PC	61.01 MB/s	48.23 MB/s
HP Pocket PC H5550, Intel® PXA250 400MHz processor	23.61 MB/s	10.84 MB/s
Treo 600, Palm OS, Arm processor @144 MHz	5.76 MB/s	0.452 MB/s
Onhand PC watch, 16-bit processor @ 3.67 MHz	0.362 **KB/s**	[too slow]

7. Conclusion

We believe that location awareness can enrich traditional security mechanisms with greater flexibility and expressiveness power and enable a variety of security services. In this paper, we define location-based encryption and apply it to two Gaia services, the publish/subscribe event system and context file system. In addition to enabling context-aware access control, our location-aware encryption mechanism eliminates the need for re-keying when changes in an entity's location trigger a change in the permissions for its corresponding role. We feel that location information can be used to provide many other security services for pervasive computing. We envision location being used in conjunction with identity and group membership to provide finer-grain access control services, location-based encryption services and efficient key management for group communication.

8. References

[1] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," presented at INFOCOM, 2000.

[2] E. Gabber and A. Wool, "How to Prove Where You Are: Tracking the Location of Customer Equipment.," presented at 5th ACM conference on Computer and Communications Security, 1998.

[3] P. Tao, A. Rudys, A. M. Ladd, and D. S. Wallach, "Wireless LAN Location-Sensing for Security Applications," presented at WiSE 2003.

[4] A. Lee, J. Boyer, C. Drexelius, P. Naldurg, R. Hill, and R. Campbell, "Supporting Dynamically Changing Authorizations in Pervasive Communication Systems," presented at 2nd International Conference on Security in Pervasive Computing, 2005.

[5] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Securing Sensor Networks with Location-Based Keys," presented at IEEE Wireless Communications and Networking Conference (WCNC 2005), New Orleans, LA, 2005.

[6] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness," presented at Workshop on the what, who, where, when and how of context-awareness at CHI, 2000.

[7] M. Roman, C. K. Hess, R. Cerqueira, R. H. Campbell, and K. Narhstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," *IEEE Pervasive Computing Magazine*, vol. 1, pp. 74-83, 2002.

[8] UbiSense, "Local position system and sentient computing." http://www.ubisense.net/.

[9] D. E. Denning and P. F. MacDoran, "Location-Based Authentication: Grounding Cyberspace for Better Security," *In Computer Fraud & Security.*, 1996.

[10] A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. Campbell, and M. D. Mickunas, "MiddleWhere: A Middleware for Location Awareness in Ubiquitous Computing Applications," presented at 5th International Middleware Conference (Middleware 2004) (accepted), 2004.

[11] C. K. Hess, "A Context File System for Ubiquitous Computing Environments," University of Illinois at Urbana-Champaign, Urbana-Champaign, CS Technical Report UIUCDCS-R-2002-2285 UILU-ENG-2002-1729, July 2002 2002.

[12] B. Borthakur, "Distributed and Persistent Event System For Active Spaces," in *Master Thesis in Computer Science*. Urbana-Champaign: University of Illinois at Urbana-Champaign, 2002, pp. 67.

[13] C. Hess and R. H. Campbell, "A Context-Aware Data Management System for Ubiquitous Computing Applications," presented at International Conference in Distributed Computing Systems (ICDCS 2003), Providence, Rhode Island, 2003.

[14] OMG, "CORBA Event Service Specification," Specification March 2001.

[15] OMG, "CORBA, Architecture and Specification," Common Object Request Broker Architecture (CORBA) 1998.

[16] M. Roman, A. Singhai, D. Carvalho, C. Hess, and R. H. Campbell, "Integrating PDAs into Distributed Systems: 2K and PalmORB," International Symposium on Handheld and Ubiquitous Computing (HUC'99), Karlsruhe, Germany, 1999.

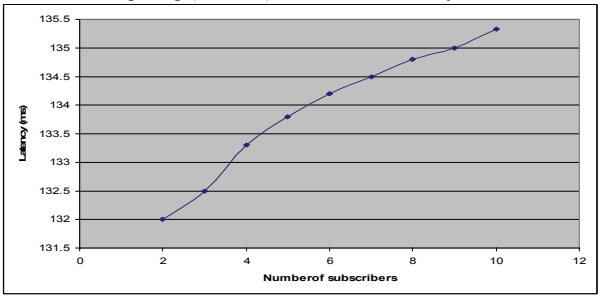

Figure 5: Latency in a location-aware event channel