Implementation of a Scalable Context-Aware Computing System

Tamer Nadeem[1], Adel Youssef, Suman Banerjee, Moustafa Youssef,

Sulabh Agarwal, Kevin Kamel, Andrzej Kochut, Christopher Kommareddy,

Pankaj Thakkar, Bao Trinh, A. Udaya Shankar, and Ashok Agrawala

MIND Lab, UMIACS and Department of Computer Science

University of Maryland, College Park, MD 20742, USA

nadeem, adel, suman, moustafa, sulabh, kamelkev, kochut,

kcr, thakkar, bao, shankar, agrawala@cs.umd.edu

Abstract. Context-aware computing involves the automatic tailoring of information and services based on the current location of the user. In this paper, we describe our experience in implementing Rover, a system that enables location-based services, as well as the traditional time-aware, user-aware and device-aware services. To achieve system scalability to very large client sets, Rover servers are implemented in an "action-based" concurrent software architecture that enables fine-grained application-specific scheduling of tasks. We have demonstrated its feasibility through implementations for both outdoor and indoor environments on multiple platforms. In this paper, we focus on different implementation issues that arose in the deployment of this system, including location determination, management of spatial data, security of client data and our analysis and measurements to estimate system scalability.

Keywords: ubiquitous, context-aware, location-based, operational-law.

1 Introduction

Context-aware computing requires the automatic tailoring of information and services based on the current context of the user. The *context* of the user typically consists of a set of user-specific parameters including his location, the characteristics of the access device and interface, and the interests of the user, usually represented in an user profile. The different technology components needed to realize context-aware computing are present today, powered by the increasing capabilities of mobile personal computing devices and the increasing deployment of wireless connectivity (IEEE 802.11 wireless LANs [9], Bluetooth [1], Infra-red [2], Cellular services, etc.)

What has hindered its ubiquitous deployment is the lack of system-wide integration of these components in a manner that scales with large user populations. In our prior work [17], we had described an architecture of such a context-aware computing system, called *Rover*, and had discussed its applications, devices, and users. In this paper, we focus on our implementation experience of this system for both indoor and outdoor environments.

Rover enables services with characteristics: a) Location-aware, in addition to the more traditional notions of time-aware, user-aware, and device-aware. b) Available via a variety of wireless access technologies and devices. c) Scales to a very large client population through fine-resolution application-specific scheduling of resources at the servers and the network.

In the next section we present an overview of the Rover system. In Section 3 we describe the our implementation of the system. We explain the functionality of the system in Section 4. In Section 5, we briefly comment on our experiments and analysis to test system scalability. We describe some related projects in Section 6 and finally conclude in Section 7.

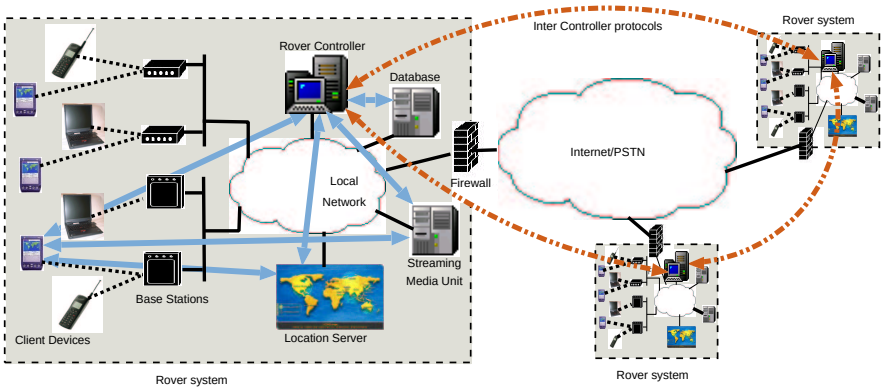

Fig. 1. Physical architecture of the Rover System

2 Overview of Rover

A Rover system, depicted in Figure 1, consists of the following entities:

- *End-users* of the system. Rover maintains a *user profile* for each end-user, that defines specific interests of the user and is used to customize the content served.
- *Rover-clients* are the client devices through which users interact with Rover. Rover maintains a *device profile* for each device, identifying its capabilities and thus, the functionality available at that device.
- *Wireless access infrastructure* provides wireless connectivity to the Rover clients.
- *Servers* implement and manage the various services provided to the end-users. The server system consists of the following set of devices:
  - Rover controller: is the "brain" of the Rover system. It provides and manages the different services requested by the Rover clients. It schedules and filters the content sent to the clients based on user and device profiles and their current locations.
  - Location server: is a dedicated unit responsible for managing the client device location services within the Rover system.

- Media manager: coordinates direct audio communication between different Rover clients.
- Rover database: stores all content delivered to the Rover clients. It also serves as the stable store for the state of the users and clients that is maintained by the Rover controller.
- Authentication server and security manager[1]: authenticates users before connecting them to the Rover controller. Currently, we are using login-password technique for authentication. However, we can use any authentication mode (e.g. digital certificates, credit cards, etc.).
- Logger: interacts with all the Rover server devices and receives log messages from their instrumentation modules.

System achieves scalable performance using a new *fine-grained real-time application-specific scheduling* called the *Action Model* (described in [17]). In this model, scheduling occurs in atomic units called actions. An action is a small piece of code that has no intervening I/O operations. The Action model avoids the overhead of thread context switches and allows more efficient scheduling of execution tasks.

A Rover system represents a single domain of administrative control that is managed and moderated by its Rover controller. A large domain can be partitioned into multiple administrative domains each with its own Rover system, much like the existing Domain Name System [14]. For this multi-Rover system, we define protocols that allow interaction

[1] The Authentication server is not shown in Figure 1.

between the domains to ease user roaming. In the next section we describe our prototype of the Rover system that has been implemented in the University of Maryland campus.

3 System Implementation

In our prototype implementation all system components were developed on Linux-based platforms in C++ and they can easily be ported to any other operating system as well.

Rover Controller

Rover controller is the core part of the Rover system, scalability and reliability are the most important factors required in such system. It is implemented using the action model. Each independent transaction of the Rover controller with the clients is called a *server operation*. A server operation consists of a sequence (or more precisely, a partial order) of actions interleaved by asynchronous I/O events.

Rover Database

The database is implemented using a standard SQL language (MySQL[2]). It consists of two different components — the relational data and the spatial data. All attribute-based data, e.g. user and client states, are stored in the relational component. All spatial information, e.g. floor plan of a building, are stores in the spatial component. Vector-based image is used to represent spatial data.

[2] Any other relational database could replace MySQL since we do not use any particular feature of it.

We make use of spatial indexes (quad-trees) for efficient manipulation of the spatial data. Rover spatial space, including all spatial object, is partitioned into two dimensional areas of interest called *interest spaces*. We model the spatial dependencies, such as accessibility and visibility, between interest space objects by a *connectivity graph*. An edge in the connectivity graph models the accessibility properties between two objects, modeled as the edge's nodes. Rover spatial space is modeled as a hierarchical structure of interest spaces.

Rover Clients

The Rover clients are implemented on the Compaq IPAQ devices running the familiar distribution of Linux for handheld devices. The user interface is implemented using GTk++ (Figure 2).

Media Manager

Media manager enables Rover clients to establish direct audio communication channels, independent of the Rover server system. We have defined a protocol called *Click & Connect* that allows users to easily initiate voice communication with each other. The protocol is so named because an user can directly click on the icon representing another user to initiate voice communication. The Media Manager arbitrates the handshaking process between two clients[3] to provide a degree of organization.

Wireless Infrastructure

[3] The current implementation version limit the connection between two clients only. We plan to extend the implementation in the next Rover version to support ulti-user chat.

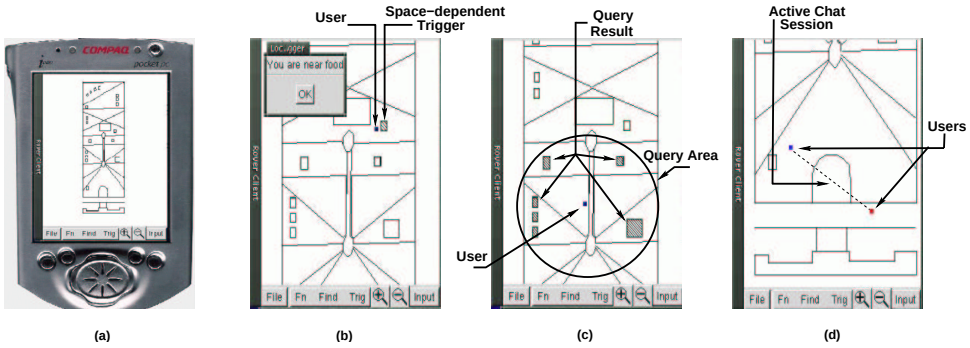

Fig. 2. Rover client screen shots taken from a demonstration at the McKeldin mall of the University of Maryland campus. (a) Rover client running the client software showing the mall map. (b) A notification to the client about a nearby food stall. The user associated with the client had previously set a trigger notification request when he is close to a food stall. (c) The user had issued a query operation about the sites of interest in his vicinity. On receiving the response from the Rover system, the client has highlighted the relevant sites. (d) An active chat session between this user and another user is marked as a dotted line connecting both users.

University of Maryland has a IEEE 802.11 wireless network widely deployed across the campus, which we used for all wireless communication to and from the Rover clients. We also deployed some additional base stations in the campus to increase the reachability of the Rover system in areas not covered by the university wireless network.

Location Server

The location server is responsible for storing and managing user locations in the Rover system. The system is designed to work in both indoors and outdoors environments. We have experimented with RF-based systems that infer the location of a device based on the signal strength of received RF signals of IEEE 802.11 wireless LAN frames.

In our RF-based techniques, the user location of a client is obtained without the use of any additional hardware. It thus provides more ubiquitous coverage in campus-like environments

that already have a rich wireless LAN coverage for data transport. This can be contrasted to alternative Infra-red tag-based systems [18, 11, 3] or ultra-sonic emitter and receiver based systems [16] in which additional devices need to be attached to the infrastructure as well as the clients. We have developed different RF-based technique in the context of the Rover system. Techniques are categorized into:

- *Radio-map Techniques:* Work in 2 phases: an offline phase and a location determination phase. During the offline phase, the signal strengths received from the access points, at selected locations in the area of interest, are gathered as vectors and tabulated over the area. During the location determination phase, the vector of samples received from each access point is compared to the radio-map and the "best" match is returned as the estimated user location. We used two methods to calculate the best match:
  - K-Nearest Neighbors (KNN): A distance function is defined to measure the distance between any two data vectors. The nearest K vectors to the vector of samples, received from each access point at the location determination phase, are calculated. Then from the K vectors a vote is conducted to estimate the best user location.
  - Probabilistic Clustering-based: Baye's theorem is used to estimate the probability of each location within the radio-map. Then the most probable location, using the vector of samples, is reported as the estimated user location. Refer to [21] for more details.
- *Model-based Techniques:* The relation between the signal strength received from an access point and the distance to this access point is captured by some function (model). By using three or more access points, the user location is estimated. Two methods are used:

- Minimum Triangulation: Given each access point i located at coordinates x_i, y_i, z_i, the distance between the receiver and the access point (d_i) is modeled as:

$$d_i^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = \frac{k}{v_i}$$

where x, y, z are receiver's coordinates, v_i is the strength of the received signal, and k is a constant. Due to problems in signal propagation like reflection and multi-pathes, we model the problem as finding a solution to the minimization problem:

$$\min f(x, y, z, k) = \min \sum_{i=1}^{n} (v_i d_i^2 - k)^2$$

By solving the derivatives equations of f numerically, we can get the estimation coordination of the receiver.

- Curve Fitting: The received signal power is modeled as:

$$PL(d)[dB] = A + B \log(d)$$

$$A = PL(d_0)[dB] - 10n \log(d_0), \; and \; B = 10n$$

where PL is the received power at certain position in decibels, d is the three dimensional path length between the transmitter and the receiver, d_0 is a reference distance, and n represents the path loss exponent. We estimate the A and B parameters for each access points using curve fitting techniques. Given the vector of samples, we can estimate how far the receiver is from each access point to estimate his location.

For indoor environments, we found that radio-map based techniques achieve better accuracy than model-based techniques. This is because the relation between the signal strength and distance in indoor environments is complicated by to the multi-path effect and other phenomena which are difficult to capture by simple models. On the other hand, model-based techniques have the advantage of not depending on the calibration process required

to build the radio-map. This advantage favors model-based techniques in outdoor environments, where the relation between signal strength and distance can be captured by simple functions and the coverage area is large making building the radio-map a time consuming process.

Security Manager

The Rover system inherits the vulnerability of the IEEE 802.11 wireless communication system such as *unauthorized access*, *non confidential*, and *non integrity*. Therefore, we have implemented different security mechanisms to protect client interactions.

- *Client-Server Security:* A session based security protocol, Secure Sockets Layer (SSL) is used to secure the communication channel between the clients and the server. Users are pre-registered with the Rover system and the user-assigned password is used to authenticate the user once the secure channel has been set up between user's client device and the Rover server system.
- *Chat (peer-peer) Security:* The audio chats are real-time communication and hence require a light-weight solution. For the direct audio chat sessions we employ the Data Encryption Standard (DES), which is an inexpensive symmetric key cryptographic system. The session key is transferred using the SSL-protected secure channel.

4 System Functionality

The Rover system provides different capabilities to the users, which can be categorized as follows:

– *System Admin Operations* are available only to the authorized system administrator. These set of operations are — register new user/device, update user/device attributes and de-register user/device. The administrator is also able to query the Rover server system about the state and information specific to any and all Rover clients in the system.
– *User Access Operations* are the basic set operations that every user avails to access the Rover system. They include the user login and user logout operations.
– *Trigger Operations* allow users to set context-specific alerts. The triggers are activated based on user interests and depend on current time and/or location of the user. An user can enable triggers by specifying the relevant time or space-dependent condition. When the trigger condition is satisfied the Rover server system sends appropriate notification to the particular user (Figure 2(b)).
– *Query Operations* allow users to acquire information about different aspects of the system and the environment. For example, an user can request information about all or a subset of all active users in the system. Figure 2(c) shows a client screen shot in response to a client query on sites of interest in its vicinity.
– *Location Update Operation* inform the server system about the client's location using.
– *Audio Chat Operations* enable direct audio communication between clients. Audio chat between clients is initiated with the coordination of the Media Manager. Once an audio chat is initiated, the clients interact directly with each other without intervention of the Rover server system. Figure 2(d), shows the display at a client that is involved in an audio chat with another client. The dashed line indicates an active chat session between the clients [4].

[4] Each user is marked on the Rover client using a different color.

5 System Performance

To asses the performance and scalability of the Rover System we take two approaches: a) *Active Monitoring* where we instrumented the controller to collect different performance statistics (e.g. queue lengths for each component, the response time for each operation, etc.). b) *Passive Monitoring* which is described in this paper.

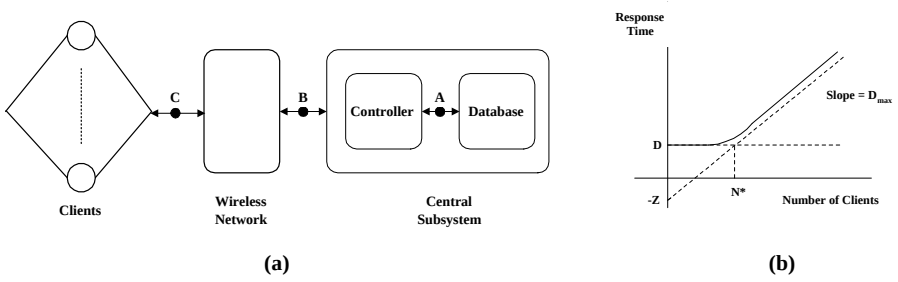

Fig. 3. Passive monitoring analysis: (a) Performance model for passive monitoring. (b) Typical asymptotic bounds.

5.1 Analysis using Passive Monitoring

In this approach, no instrumentation code is introduced in the server system. Instead, we use a client load generator to stress test the server and observe two different metrics — the response time obtained by individual clients and the number of clients that can simultaneously served by the system without significantly impacting the performance of the clients.

We model the Rover system as a single-server multi-client system as shown in Figure 3(a). The Rover System is modeled as a central subsystem consisting of two devices, the Rover controller and the Rover database, and N terminal subsystems. Each of the N terminals is a client of the Rover System and perform the cycle of issuing a request, waiting for the response and processing the response (think time). Wireless network models the communi-

cation channel between the server and the clients. Since we are interested in assessing the performance of the Rover system we do not explore the affects of communication channel in this paper.

Using a technique called *operational law* [10] to analyze such systems, it can be shown that the response time observed by clients increase marginally with increasing the number of clients up to a critical client population. Let D denotes the time required by the system to process a single client operation, and $D_{\max}$ denotes the time required at the bottleneck server of a multi-server system. For the single server model, $D_{\max} = D$. If N^* indicates the critical number of clients that the system can support without impacting the response time for the clients and Z the *think time* used by the clients between operations, then operational analysis suggests that:

$$N^* = \frac{D + Z}{D_{\max}} \tag{1}$$

The graphical representation of N^* is shown in Figure 3(b).

5.2 Experiment Configuration

The central subsystem runs on Pentium IV 1.5 GHz desktop machine with 256 MB of RAM running the Linux OS with kernel version 2.4.7. A second machine is used to behave as a set of clients (client loader). The client loader runs on a Pentium III 800 MHz laptop with 128 MB of RAM and running a Linux OS with kernel version 2.4.2. The client machine uses 802.11b wireless network to connect to the network.

The response time for each operation were collected as observed at the database, the controller and the client (points A, B, C respectively in Figure 3(b)). Instead of collect-

ing response time for each of the system operations, we experimented with three different operations representing three different categories:

1. *GetAllLoginUsers:* Gets the position of all users who are logged into the system. This operation is controller intensive and does not involve the database.
2. *VectorMap:* Gets the vector map of an area. This operation is computationally intensive at the database side.
3. *Locate:* Locates the object containing the given point. This operation involves the database though it is not very computationally intensive at either the database or the controller sides.

5.3 Results and Discussion

In this section we show the results obtained for each of the above operations.

GetAllLoginUsers

Figure 4(a) shows the response time at the controller plotted against the number of clients in the system for different think times ($Z = 100ms, 200ms, 300ms$). The total service demand time, D of the controller is observed to be around 300 microseconds[5]. For only one device in the system $D_{max} = D$. Using Equation 1 where $D = 300$ microseconds and a think time $Z = 200$ milliseconds, we get N^* to be approximately 667 requests. Hence the server can support 667 requests without any significant delays. In an actual deployment, the think time

[5] This is the response time for only one client in the system is equal to the total service demand time for the database device as there is no queuing in the system with one client.

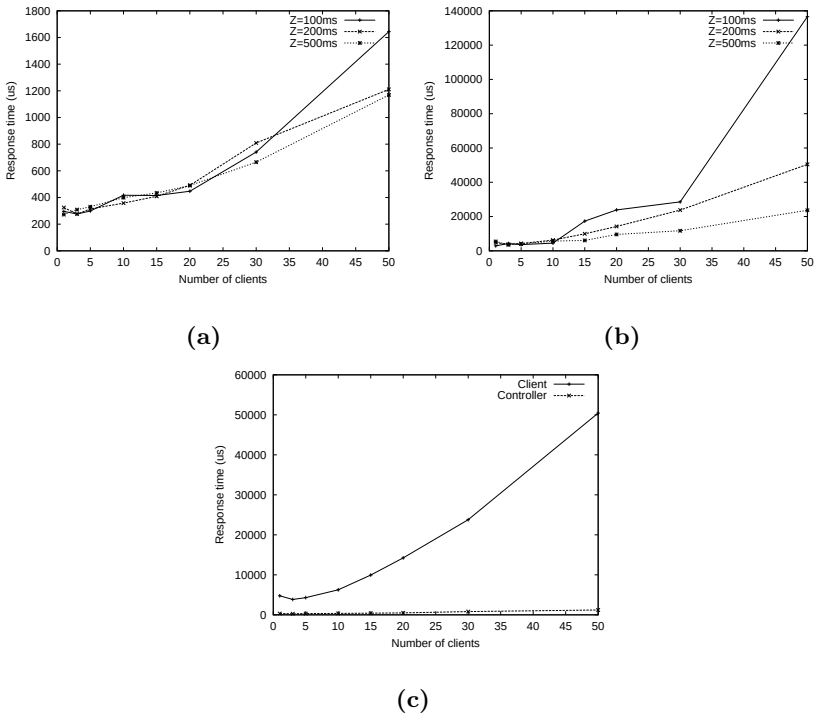

Fig. 4. *GetAllLoginUsers* operation: (a) Controller response time, (b) Client response time, and (c) Response time when Z=200ms.

would be of the order of 10's of seconds and that would give an even higher value of N^* (of the order of thousands of clients). Figure 4(b) shows the response time for the client side.

Figure 4(c) shows the response time behavior of both the controller and the client when the think time is Z = 200 milliseconds. As the graph shows the controller graph stays almost horizontal as the number of clients are increased which shows the controller can handle a large number of clients. On the other hand, the client graph grows with the number of clients. This can either be due to the effect of the wireless hop involved or the processing involved at the OS level. The performance of the 802.11b wireless network has not been taken into account and is left for future work.

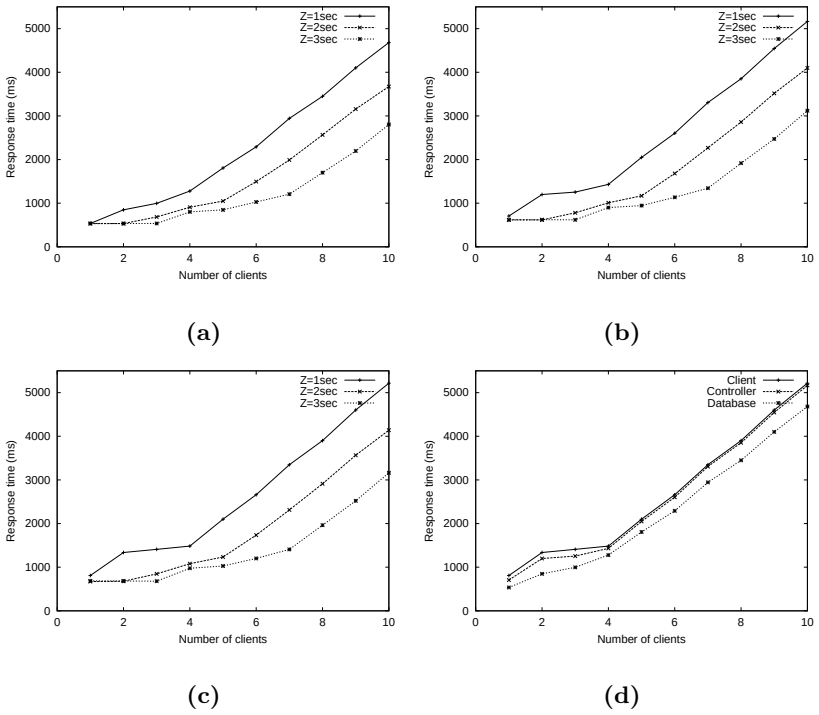

Fig. 5. *VectorMap* operation: (a) Database response time, (b) Controller response time, (c) Client response time, and (d) Response time when Z=1s.

VectorMap

Figure 5 shows the response time at the different Rover components. For the database the total service demand time is observed to be around 0.5 seconds. Using equation 1, we can predict the knee-point to be at $N^* = 3$ with a think time $Z = 1$ second. We should note that the *VectorMap* operation is an infrequent operation and has been used only to assess the performance of the system in the extreme case. In an actual deployment, the duration between subsequent *VectorMap* operation requests (Z) would be in the order of minutes.

Figure 5(d) shows the response time observed at the database, the controller, and the client when the think time is $Z = 1$ second. The difference in the database response and the controller response could be explained by the fact that at the controller all the data is

touched and a copy is created for debugging purpose. Once this debugging code is taken out the controller graph should follow the database graph very closely.

Locate

Similar to the analysis of the previous operations, we show the response time at the database, the server and the client for a think time of 200 milliseconds in Figure 6. With database service demand time (D) is 200 microseconds and the think time (Z) is 200 milliseconds, we get N^* to be approximated 1000 requests the database can handle without any significant delays. The difference between the database graph and the controller graph can be explained by the same reasoning as given in the analysis of *VectorMap* request.

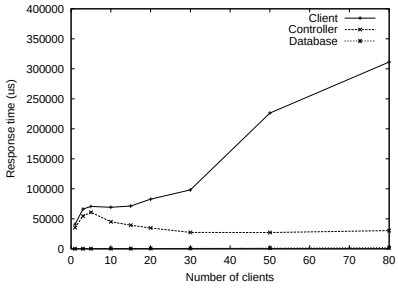

Fig. 6. Response time of *Locate* (Z=200ms)

6 Related Work

There are several ongoing research efforts in the area of context-aware applications. The Active Badge system [18] developed at Olivetti Research is considered to be one of the earliest context-aware applications. Using infrared based special badges, the system was able to find out location of the current people in each room. A personal shopping assistant was designed

at AT&T [4] to help the customer navigation through a store using infrared technology enabled special badges. Georgia Tech's conference assistant [7] was designed to assist conference attendees in choosing specific presentations to attend based on their profile. AT&T sentient computing system [13] location based system is based on ultrasonic measurements. The Cyberguide [3] project is a context-aware tourist guide prototype based on infrared technology. Several other tourist guide systems have been developed as GUIDE system [5]and CMU's smart sight [20]. Other personal assistant projects include: office assistant [19], ComMotion [12] project, the Rome project [8], CyperMinder [6] and HP's Cooltown [15]. All the context-aware applications described in this section were developed as a framework to assist individual users. Rover, *in contrast*, defines a system framework in which such applications can be built. It allows direct interaction between the users and as well as between the users and the environment in a scalable manner. Rover system architecture enables easy instantiation of new applications into the system by appropriate definition of server operations and client interactions.

7 Conclusions and Future Work

Rover is currently available as a deployable system using specific technologies, both indoors and outdoors. Our final goal is to provide a completely integrated system that operates under different technologies, and allows a seamless experience of location-aware computing to clients as they move through the system. With this in mind, we are continuing our work in a number of different directions. We are experimenting with a wide range of client devices, specially the ones with limited capabilities. We are also experimenting with other alternative

wireless access technologies including a Bluetooth-based LAN. We are also working on the design and implementation of a multi-Rover system.

We believe that Rover Technology will greatly enhance the user experience for many different context-aware applications in different environments. Our initial experience indicate that our designed system scales to large user populations and the benefits of the system increase with increase in the number of users.

References

1. http://www.bluetooth.com.
2. http://www.irda.org.
3. G.D. Abowd, C.G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide. *Wireless Networks*, 3(5), October 1997.
4. A. Asthana, M. Cravatts, and P. Krzyzanowski. An indoor wireless system for personalized shopping assistance. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, California, December 1994.
5. K. et al. Cheverst. Experiences of developing and deploying a context-aware tourist guide: The lancaster guide project. In *Proc. 6th Ann. Int'l Conf. MobileComputing and Networking (Mobicom 00)*, New York, 2000.
6. A.K. Dey and G.D. Abowd. Cybreminder: A context-aware system for supporting reminders. In *Proceedings of Second International Symposium on Handheld and Ubiquitous Computing (HUC 2000)*, Bristol, UK, September 2000.
7. A.K. Dey, M. Futakawa, D. Salber, and G.D. Abowd. The conference assistant: Combining context-awareness with wearable computing. In *Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99)*, San Francisco, California, October 1999.
8. A.C. Huang, B.C. Ling, S. Ponnekanti, and A. Fox. Pervasive computing: What is it good for? In *Proceedings of the ACM International Workshop on Data Engineering for Wireless and Mobile Access*, Seattle, Washington, August 1999.

9. IEEE. Wireless LAN medium access control (MAC) and physical layer (PHY) specification, Standard 802.11, 1999.
10. R. Jain. *The Art of Computer Systems Performance Analysis*, chapter 4. John Wiley and Sons, New York, 1991.
11. S. Long, R. Kooper, G.D. Abowd, and C.G. Atkeson. Rapid prototyping of mobile context-aware applications: the cyberguide case study. In *Proceedings of the Second Annual International Conference on Mobile Computing and Networking*, White Plains, New York, November 1996.
12. N. Marmasse and C. Schmandt. Location-aware information delivery with commotion. In *Proceedings of Second International Symposium on Handheld and Ubiquitous Computing, HUC 2000*, Bristol, UK, September 2000.
13. Steve Hodges Joe Newman Pete Steggles Andy Ward Andy Hopper Mike Addlesee, Rupert Curwen. Implementing a sentient computing system. *IEEE Computer Magazine*, 34(8), 2001.
14. P. Mockapetris. Domain names - implementation and specification, RFC 1035, November 1987.
15. S. Pradhan, C. Brignone, J.H. Cui, A. McReynolds, and M.T. Smith. Websigns: Hyperlinking physical locations to the web. *IEEE Computer*, 34(8), 2001.
16. N.B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of ACM Mobicom*, August 2000.
17. A. Udaya Shankar Ashok Agrawala et al S. Banerjee, Tamer Nadeem. Rover technology: Enabling scalable location-aware computing. *IEEE Computer*, 2002.
18. R. Want, A. Hopper, V. Falco, and J. Gibbons. The active badge location system. In *ACM Transactions on Information Systems*, January 1992.
19. H. Yan and T. Selker. Context-aware office assistant. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, New Orleans, LA, January 2000.
20. J. Yang, W. Yang, M. Denecke, and A. Waibel. Smart sight: a tourist assistant system. In *3rd International Symposium on Wearable Computers*, San Francisco, California, October 1999.
21. M. Youssef, A. Agrawala, and A. Shankar. Wlan location determination via clustering and probability distributions. *IEEE International Conference on Pervasive Computing and Communications (PerCom) 2003*, March 2003.