

A General Architecture in Support of Interactive, Multimedia, Location-Based Mobile Applications

Sean J. Barbeau, Miguel A. Labrador, Philip L. Winters, Rafael Pérez, and Nevine Labib Georggi,
University of South Florida

ABSTRACT

The widespread use of cellular telephones and the availability of user-location information are facilitating the development of new personalized, location-based applications. However, as of today, most of these applications are unidirectional and text-based where the user subscribes and the system sends a text message when appropriate. This article describes a modular and general architecture that supports the development of interactive, multimedia, location-based applications, providing an extra level of service to the users. The flexibility of the architecture is demonstrated by presenting the Wireless Safety Security System (Wi-Via) and other potential applications.

INTRODUCTION

Advances in end-system devices, information technology, mobile cellular networks, and the Internet are allowing innovative applications and services that may have been unattainable only a few years ago. The proliferation of powerful and handy mobile devices makes these innovations more convenient than ever. For example, personal digital assistants (PDAs) and cellular phones can now be used to carry not only voice but also data traffic, such as text messages, pictures, and video clips from anywhere at anytime. PDAs and cell phones now emulate computers, with enhanced graphical user interfaces, integrated Global Positioning Systems (GPSs), wireless data connectivity, efficient batteries, powerful central processing units (CPU), and expanded storage capabilities. Because of advances in communication protocols, databases, and software development environments, these end-system devices connected to wireless cellular networks can interact with many hosts and servers via the Internet. Similarly, hardware-independent programming languages allow the development of applications that can run on any of these devices and exchange information to

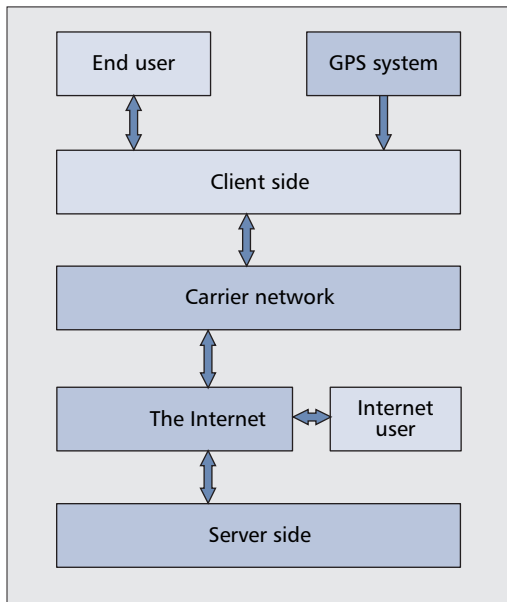
and from other clients, servers, and specialized databases. This generalized concept facilitates transportability of developed software across different devices and networks, which is a necessity for the rapidly advancing market of wireless communications.

One of the key technological advances for the development of location-based applications is the use and availability of positioning systems. GPS chips are now included in many devices to analyze satellite signals and determine the user's location¹ with high accuracy. For cellular devices that either do not have integrated GPS chips or are located in areas where the GPS signal is not strong enough, other positioning alternatives have been developed. In the United States, cellular positioning technology is being pushed forward by the Federal Communications Commission (FCC) mandate that requires cellular carriers to provide user position information for emergency calls (E911) [1].

The availability of user-location information makes a big difference between today's applications and the ones considered in this article. For example, Advanced Traveler Information Systems (ATIS) provide subscribed users with traffic alerts about specific locations of interest via text messages. As a result, subscribers may occasionally receive traffic alerts about congested highways in Tampa while on a business trip in San Francisco. This is because most current applications work on a "push" basis; the system sends data to the user but the user does not have the capability to send information back to the system. The possibility of sending user-location information can change the effectiveness of these applications dramatically, as personalized information can be provided to users where and when it is actually needed. Another limitation of today's applications is the lack of both interactive and multimedia support. For example, Amber Alerts (missing child alerts) are of limited use when only a written description of the child is provided to mobile phone users.

This article describes a general architecture

¹ Although location and position normally refer to different accuracy levels, in this article we use these terms interchangeably.



■ **Figure 1.** High-level system architecture.

capable of supporting new location-based, personalized, interactive, multimedia applications using regular cellular telephones and networks.

The rest of the article is organized as follows. Similar architectures and the proposed one are described in the following section. Then the main components of the architecture and the interactions among its modules are described in detail. The location application program interface (API) for mobile phone software and the location-based services API for Web applications are explained next. Finally, the flexibility of the architecture is demonstrated by presenting the Wireless Safety Security System (Wi-Via) and other potential applications.

RELATED WORK AND OVERVIEW OF SYSTEM ARCHITECTURE

Much literature exists on architectures for location services and positioning mechanisms. Location services consist of network architectures and protocols meant to provide location data. The Mobile Location Protocol (MLP) [2], the Wireless Application Protocol (WAP) [3], and the Parlay/OSA architecture [4] belong to this category. On the other hand, positioning mechanisms are methods and techniques used to calculate the user's location. Positioning mechanisms, such as those based on proximity, lateration, angulation, and scene analysis are utilized in indoor, outdoor, satellite, and cellular positioning systems. These architectures, protocols, and mechanisms are out of the scope of this article and the interested reader is referred to [5–7] and the references therein.

This article discusses a different type of architecture, a higher-layer architecture used to develop interactive, multimedia, personalized, and location-based applications. As such, of interest are location-based middleware, such as the Common Object Request Broker Architecture (CORBA) and the Java-based Remote

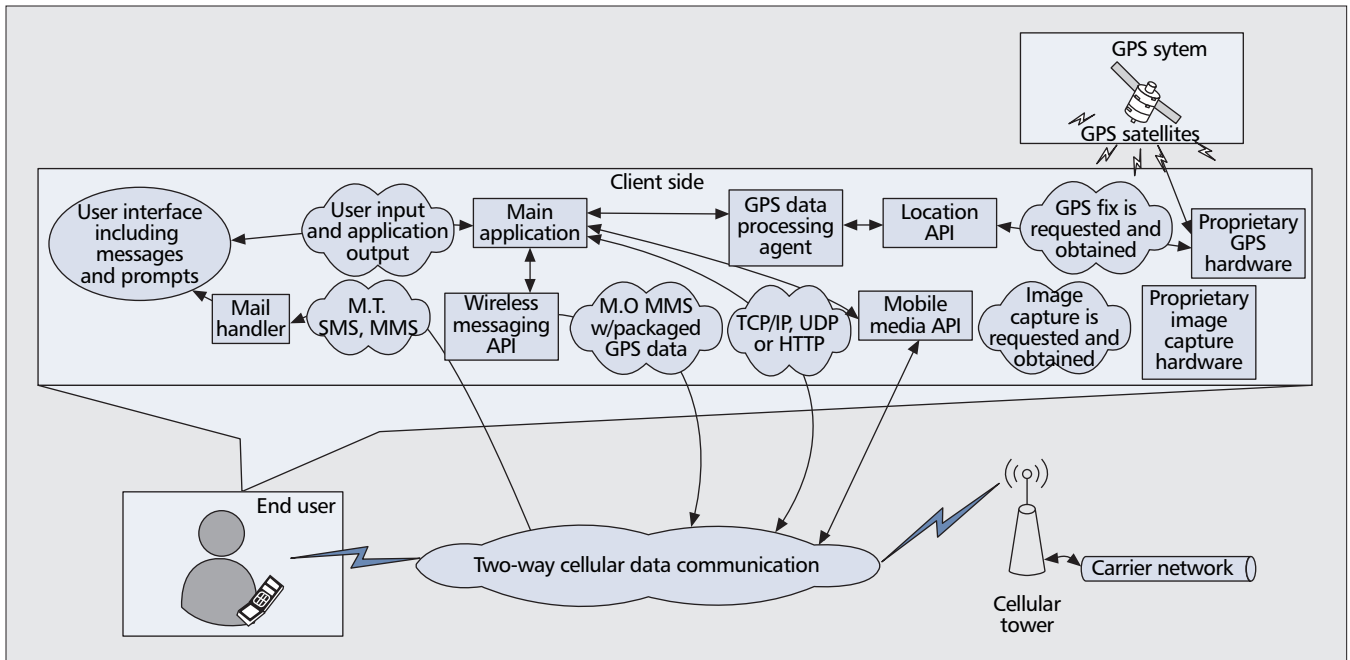
Method Invocation (RMI) [8], which provide a series of APIs that hide low-level details from the application. In terms of architectures, the location-based services (LBS) supply chain presented in [6] is a reference model similar to our high-level architecture; however, it identifies the reference points between modules and does not describe the modules in detail. The work presented in [9] is also similar but deals with push-based applications, where users need to subscribe to take advantage of LBS. Further, it is based on the Session Initiation Protocol (SIP) [10] and utilizes the Parlay/OSA architecture. A list of available platforms for location-based services is included in [11], where the authors introduce their own middleware and service-provisioning platform. Nonetheless, none of these platforms integrates all the features and services provided in our architecture.

Figure 1 shows a high-level view of the proposed system architecture. It consists of four main components: the Client Side, consisting of the end user, the end user device, and the GPS system; the Carrier Network; the Internet and its users; and the Server Side (database and application servers).

In this high-level system architecture, the Client Side is a cellular telephone or PDA with multimedia capabilities and embedded software that provides the user interface and obtains the user-location information. The end user devices are all assumed to be customers of a cellular service provider; therefore, the second component is the carrier's cellular network. The cellular network also can provide user-location information through network-based positioning systems. The Internet, the third component, is used as a transport network to send the application data from the cellular network to the server implementing the service and vice versa. The Server Side consists of servers and databases that run the applications and store the users' data. Depending on the type of service provided by the application, access to different databases and servers may be needed. Our architecture opens up the possibility for third-party application developers to offer new and innovative services to the mobile community at large.

The architecture supports the exchange of interactive multimedia messages. For example, one user responding to an Amber Alert could send a picture or video of the suspect to a 911-like center with the location data embedded in the message. Upon receiving the message, the operator could either send a message back to the user or establish a voice conversation using a Voice over IP (VoIP) application. The architecture also supports automatic or on-demand messaging. In the case of on-demand messages, the user requests the service only when needed, using the application's user interface. In the case of automatic messaging, the system includes some extra capabilities to determine if it is appropriate to send a message to the user. In this case the system will not send a traffic alert about Tampa while the user is physically located in San Francisco. Therefore, using the architecture described above, a location-based, personalized, interactive, and multimedia service is possible.

Depending on the type of service provided by the application, access to different databases and servers may be needed. Our architecture opens up the possibility for third-party application developers to offer new and innovative services to the mobile community at large.



■ Figure 2. The client side.

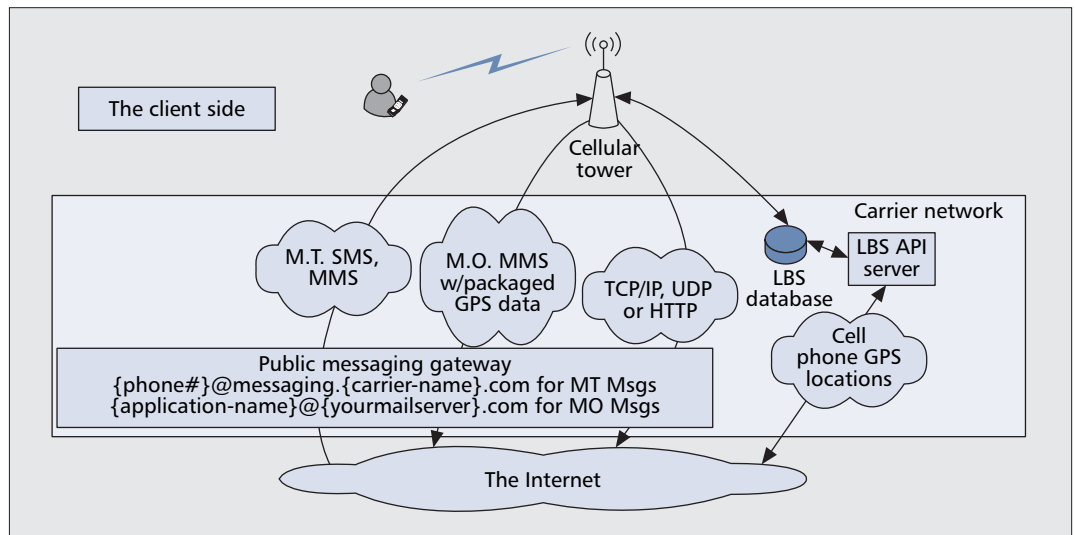
DESCRIPTION OF SYSTEM COMPONENTS

This section describes each of the four components of the system architecture in more detail, including the interactions between them.

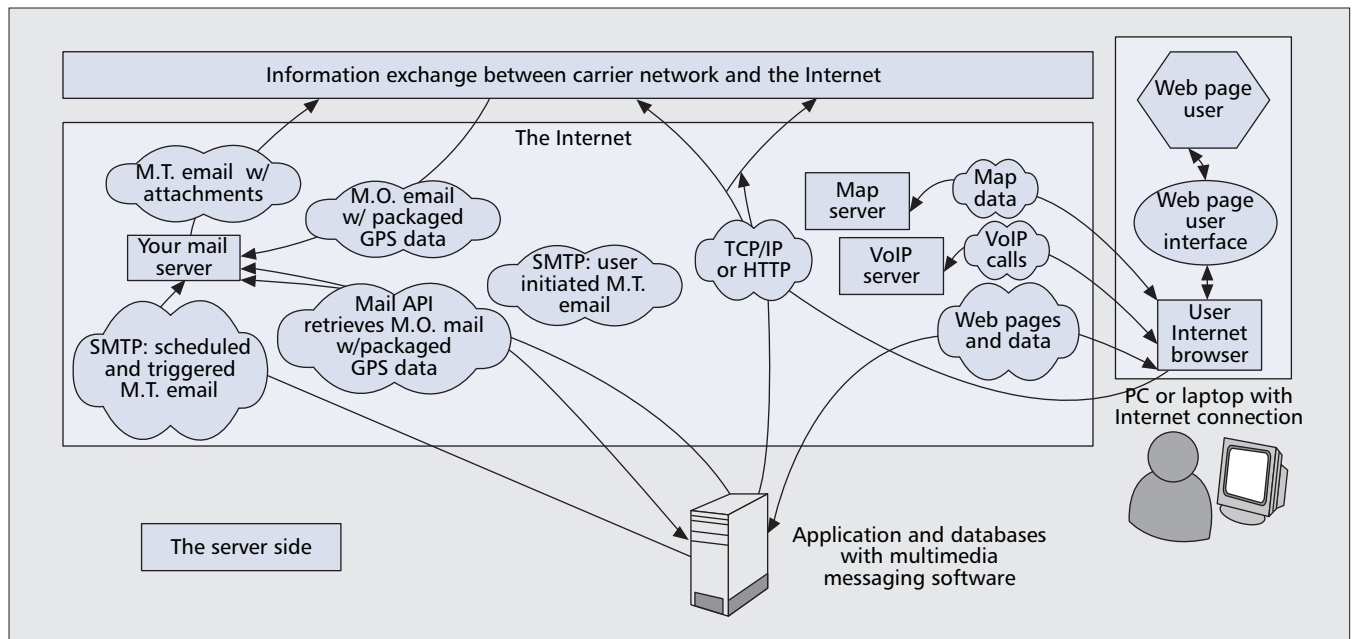
THE CLIENT SIDE

The Client Side, depicted in Fig. 2, includes the end user, the GPS system and the end user device. The **Main Application** is the software program that implements the desired service in the mobile device. This module interfaces with all other modules in the Client Side and selects inputs and outputs from other modules as necessary. The Client Side software also includes a Graphical User Interface (GUI) so that the user can interact with the system.

The application must be developed using a software development environment that specifically addresses the limited resources of cellular telephones, PDAs, and embedded devices. Furthermore, it should be able to dynamically modify the runtime environment according to the capabilities of the end user device. The application development environment should also provide general as well as specific APIs. General APIs are very useful for common tasks, such as developing user interfaces and establishing HTTP and regular TCP/UDP network connections. Specific APIs provide additional functionality and automate several important tasks. For example, the **Location API** provides a means to request and obtain GPS coordinates from the GPS hardware or cellular network, regardless of the implementation of the positioning technology. Other important APIs uti-



■ Figure 3. The carrier network.



■ Figure 4. *The Internet.*

lized in the client application are the **Mobile Media API** for displaying and recording audio, video, and images, and the **Wireless Messaging API** for transmitting mobile originating (MO) messages (messages sent from cell phone to server) to the wireless cellular network utilizing the multimedia messaging service (MMS) and the short messaging service (SMS). On the other hand, mobile terminating (MT) messages coming from the cellular network are handled by the **Mail Handler** module embedded in the phone's operating system, which shows the incoming message on the telephone's screen and provides the user with an onboard mail management tool.

THE CARRIER NETWORK

The Carrier Network, shown in Fig. 3, is the transport network that interfaces directly with the Client Side and the Internet. Our architecture utilizes two main modules from the carrier network: the public messaging gateway (PMG) and the LBS API.

The PMG handles the MMS and SMS services. This gateway channels messages to and from the Internet and the user's telephone. MO messages are addressed to {application-name}@{youremailserver}.com and are automatically translated from the MMS format to emails at the gateway. MT messages are addressed to {phone#}@messaging.{carrier-name}.com and are translated from emails to the SMS or MMS format at the gateway. Since mobile-terminating SMS and MMS capabilities differ according to the carrier, different results are displayed when sending messages to different public gateways. For example, when sending video as an attachment, the network may convert a 10 sec .3GP format video file into a four-frame animated .GIF. Cellular carriers can manipulate these messages as they wish in order to save bandwidth or storage space.

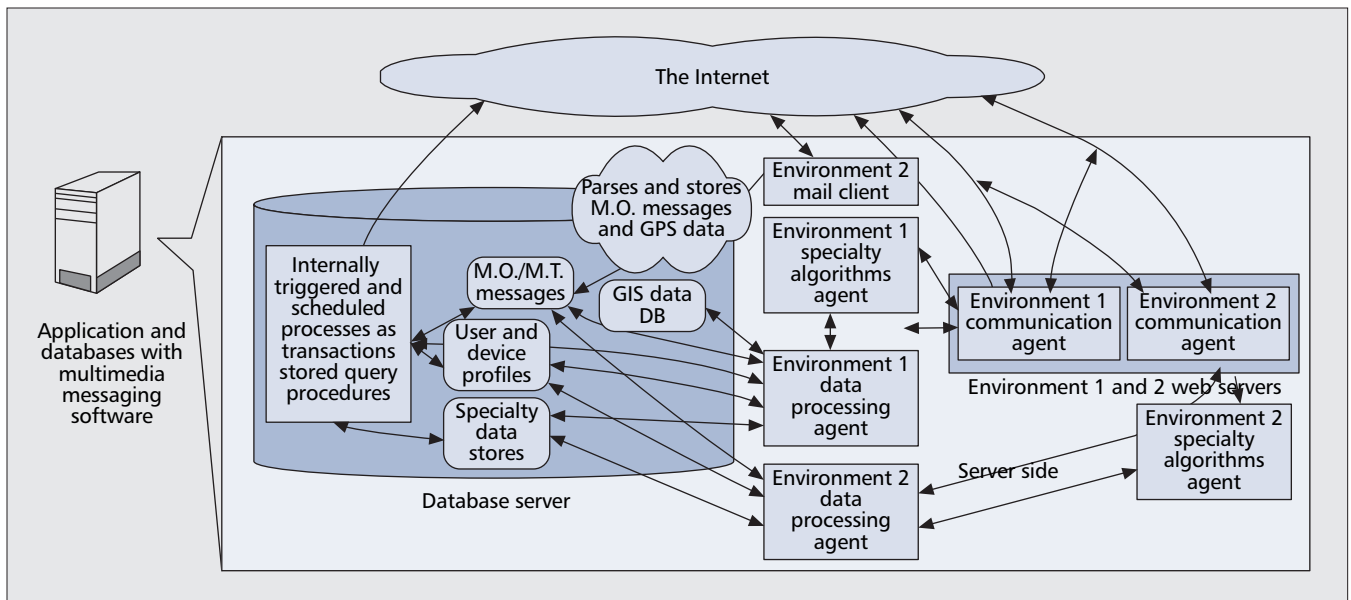
To obtain the user's location, the Client Side

application utilizes the Location API, which queries the GPS chip in the user's telephone first and the cellular network later if GPS signals (or GPS chip) are unavailable. As an alternative to device-based software running in the handset, some carriers provide an LBS API Web service whereby a Web application can query the LBS API and obtain the user's location. From the location-based application point of view, there are trade-offs between using the Location API or the LBS API, which are later discussed.

THE INTERNET

The Internet is the transport network that allows users to interact with the applications anywhere at anytime. This component, shown in Fig. 4, consists of three modules. The first module is an **Email Server** to handle the delivery of incoming and outgoing messages. MO messages travel from the carrier's Public Messaging Gateway over the Internet and are delivered to the Email Server. MO messages are then retrieved by a **Mail Client API** located on the server side, where they are further processed. MT messages are sent over the Internet from the server to the Email Server using the Simple Mail Transfer Protocol (SMTP). The Email Server then sends the messages via the Internet to the Public Messaging Gateway.

The second module is a **Map Server** to process the user's location and display it on a geographical map viewed on a Web page. There are publicly available map servers that can be utilized for this and that provide well-defined, albeit proprietary, APIs. The third module is the Voice over IP (VoIP) Server. This server belongs to any VoIP service provider that allows interactive phone conversations to be initiated from a Web page API to any phone over the Internet. These last two modules are utilized by Web applications as required, according to the services provided. For instance, the Web application utilized by the 911 operator of the Wi-Via



■ Figure 5. The server side.

system, uses the Map and VoIP servers to display current users' locations on a street map and make interactive voice calls, if necessary.

THE SERVER SIDE

The Server Side, shown in Fig. 5, is the most complicated component of the architecture. The architecture includes a **Database Server** used to store MO and MT messages, user and device profiles, specialty data for specific applications such as bus stops locations, bus routes, subscription lists, and a Geographical Information System (GIS) database used for comparing user-location information with geographic maps. In addition, a **Transactional** module is used to schedule tasks and trigger functions from within the database server. For example, it instructs the Data Processing Agent (DPA) module how and when data needs to be stored/accessed, and triggers the Specialty Algorithms Agent (SAA) module when its function is requested. The DPAs handle how data are stored and accessed in the appropriate databases while the SAAs run algorithms that utilize data supplied by the DPAs and return a response. For example, the DPA could provide the SAA with the user position and GIS map data from the database, and the SAA would utilize this information to find the evacuation zone in which the user is currently located. Finally, the transactional module also has the ability to generate automated SMTP emails when necessary. Again, its main functionality is to schedule tasks and trigger functions within the database server.

Another part of the Server Side component is the **Communication Agent**. Here, two replicated communication agents are included to increase efficiency and ease the development of applications. It may be that certain tasks can be accomplished in a considerably easier way in one particular environment than in others. For example, one environment may make direct communication between the server applications and the client easier to implement. Similarly, a different

environment may be preferred for handling complex GIS processing and Web service communication via XML. Therefore, two Web servers are included in the architecture. As a direct consequence, the architecture includes data processing and specialty algorithm modules for each environment.

SOFTWARE DEVELOPER METHODS TO OBTAIN LOCATION DATA

While different technologies are used to calculate the user's location, software developers should not be subject to differentiating one type of technology from another. This section discusses in more detail the Location API for cell phones, as well as the more recently developed LBS API that can be used to obtain a cell phone's location from a Web application.

THE LOCATION API

To obtain GPS data from the phone, the developer simply needs to call a function in the Location API. The API handles the low-level interface with the end-system device hardware and returns the requested GPS data to the application, regardless of the type of technology or method of implementation that actually calculates the user's location. Some carriers' phones utilize a hybrid positioning method whereby data are supplied by both the handset and the network. When the developer places the function call to the Location API, a criterion must be defined for the desired accuracy of the GPS fix as well as a timeout value of how long the function should search for a fix before it returns. Assuming that this is a request for a first fix (a cold start), the phone will try to obtain and utilize information from the cellular network that can lessen the time-to-first-fix (TTFF). If the GPS hardware cannot obtain a fix that meets the provided accuracy criteria within a given timeout period, then the Location API will provide the

known location of a nearby cellular tower to the application along with a note that a high accuracy GPS fix was not attainable. If a GPS fix can be obtained, it will be returned to the user along with the estimated accuracy of the GPS fix and the number of available satellites. Subsequent (“hot”) fixes will not contact the network to obtain information, since the GPS location of the phone is already known and should return new fixes rapidly unless there is a problem with the GPS signal. If the application does not query the Location API for an extended amount of time, the GPS hardware will have to begin from the “cold start” state again.

Developing software that reacts properly to each of these scenarios while adjusting the criteria for accuracy and timeout values accordingly before the next GPS fix is requested will greatly enhance the execution and performance of the location-based application. In the detailed architecture diagram of the Client Side shown in Fig. 2, the GPS Data Processing Agent performs these tasks using two finite state machines that obtain the most accurate position in the least amount of time. This increases the efficiency and performance of the interaction between the Main Application and the Location API, and therefore, the application performance. Since the input values to the Location API (accuracy and timeout value) alter the behavior of the Location API, these two values, along with the frequency of function calls to the API (controlled by a third state machine), can be changed to produce the desired behavior of the application. This optimizes the use of the GPS hardware and saves battery power based on the specific needs of the current main application, which are functions beyond the capabilities of the Location API. For example, if a tracking application is being used and the user moves into an area that cannot receive a GPS signal, the state machine will recognize that GPS fixes are not being obtained and will start to query the Location API less frequently. Once the user gets back into coverage, the state machine will resume querying the Location API at a normal pace.

THE LBS API FOR WEB APPLICATIONS

As an alternative to using the Location API, a Web application API has been developed by some carriers and is currently being tested with select commercial customers. The LBS API is a Web service that can be queried by any Web application to obtain a phone’s location without having to install any third-party software on the cell phone. The LBS API simply aggregates location information for many cell phones that are connected to the network and makes data available to external Web applications. The LBS API draws from a repository, referred to as the **LBS Database** (Fig. 3), that contains location information obtained by whatever location method is utilized. When queried by a Web application, the LBS API returns the location of the phone in an Extensible Markup Language (XML) format to the Web server that initiated the request if the LBS Database has current knowledge of the requested phone’s location. If the phone’s location is unknown,

the LBS API will attempt to obtain and return the phone’s location within a specified accuracy and timeout period. The company running the Web application must have gone through a registration process and obtained the proper permissions from both the carrier and the end user before any information regarding a phone’s location is returned.

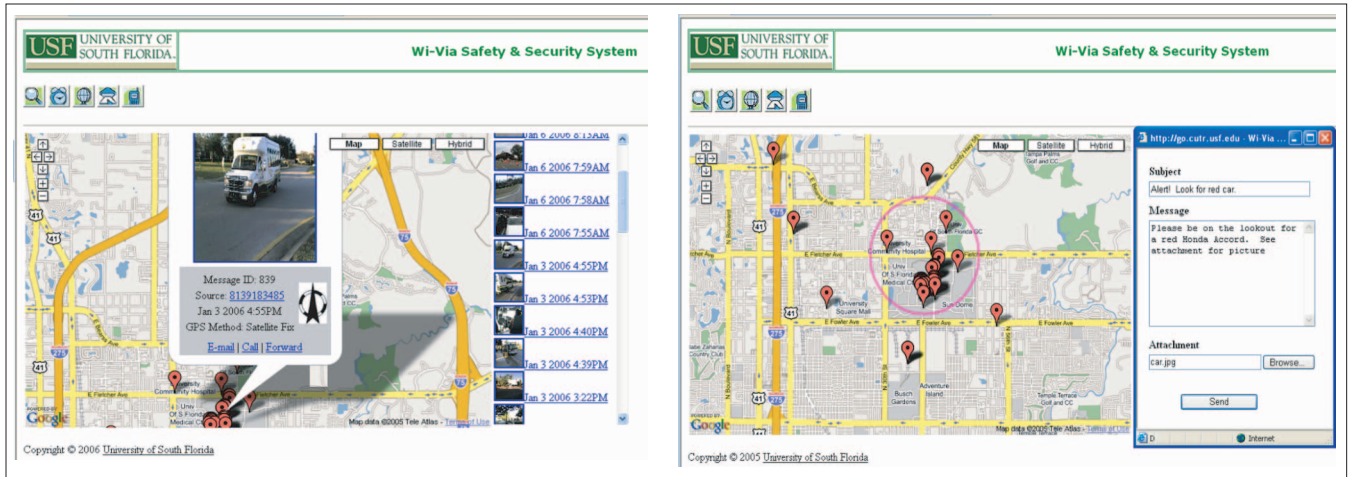
Utilizing a Web application with the LBS API to obtain the location of a phone has advantages and disadvantages over utilizing the Location API. The most obvious advantage is that no software must be configured, distributed, installed, and maintained on the end user devices. Also, the LBS API contributes to an easier design of the Client Side application. However, because the LBS API is a Web service controlled by a carrier, restrictions are placed on the number and frequency of requests made by a Web application for a specific phone, which makes emergency access of this API difficult. Because of these differences, the developer must consider the nature of the application being developed before choosing the Location API or the LBS API. For high-accuracy tracking applications that require location information every few seconds (such as real-time vehicle navigation software), the client-side Location API must be utilized. However, for some location-enabled subscription message services that occasionally need to obtain a zip-code-level estimate of a phone’s location in a single request, a Web application utilizing the LBS API may be preferable.

THE WIRELESS SAFETY SECURITY (WI-VIA) SYSTEM

With Wi-Via, cellular telephone users become additional sets of “eyes and ears” for law enforcement, emergency dispatchers, and first responders, as these users can report incidents by sending text messages, pictures, and/or video clips to a Web-accessible system in real time. For example, the application can be used by Neighborhood Watch groups to report incidents and by citizens to respond to Amber Alerts. Since the location of the user is known to the system, the emergency official can immediately see where the user is geographically located. Additionally, if more users report a suspicious person as he or she moves, the central station’s agent will be able to track the person in real-time.

The bidirectional capabilities of the system can be utilized to send personalized or group messages back to each reporting citizen. Figure 6 shows two screen shots of the Wi-Via Web interface at the control or “dispatch” station. The incoming messages are shown on a street map where the exact location information is provided. The official at the control station sees the picture or the video clip and can send a message including multimedia attachments back to the sender by clicking the link that contains the user’s telephone number. This two-way messaging system can also be used as a “reverse” 911 dispatch center. The dispatcher can select an area on the map and send a message to all sub-

Developing software that reacts properly to each of these scenarios while adjusting the criteria for accuracy and timeout values accordingly before the next GPS fix is requested will greatly enhance the execution and performance of the location-based application.



■ Figure 6. Two screen shots of the Wi-Via Web interface.

scribers that are currently located within the selected geographic area (Fig. 6). This aspect of the system could be used to give citizens specific instructions based on their current location. Citizens in potentially harmful locations can be directed by emergency officials to move in a specific direction, while those outside the dangerous area can be directed to stay where they are or to move to a different location.

The Web-based application developed for the Wi-Via system has several other capabilities. For example, the dispatcher can use VoIP technology to call any local, long distance, or international cell phone number by just a click of the mouse. In addition, the application is designed to perform real-time tracking, conference calls, zoom in and zoom out, and others. Finally, since no special hardware or software is required to view the main dispatch Web page, Wi-Via offers a flexible and portable solution in case of emergencies or natural disasters where physical access to a normal centralized dispatch center could be restricted. Any computer with Internet access can be utilized to view the Web page. This flexibility and reliance on standard communication methods ensures that if any method of data communication is possible, the Wi-Via system remains accessible and the continuity of emergency operations remains intact.

OTHER APPLICATIONS

Many other applications can be developed utilizing the infrastructure described in this article. For example, the Evacuation Zone Finder (EZ-Finder) is an application that allows mobile phone users to know their evacuation zone in case of emergency. This application can be easily enhanced to provide users with additional information such as evacuation routes and available shelters and directions from the user's location.

Another application, the Travel Assistance Device (TAD), is currently being developed to help individuals with mental or cognitive disabilities utilize public transportation. Knowing the user's route, the system can track whether the user is on the correct bus and traveling in the right direction. TAD can also show travel

instructors the real-time status of people with disabilities who are under their care on a Web page. The architecture described in this article can be used in a countless number of applications.

CONCLUSION

User-location information along with advances in information technology and communications are facilitating the development of many personalized, multimedia, interactive, community-oriented applications that will add convenience and safety to people's everyday lives. This article has described a modular and general architecture utilized to support these types of applications. The potential of the architecture is demonstrated by the Wi-Via application as well as several others.

ACKNOWLEDGMENTS

The work presented in this article has been partially funded by the University Consortium/Center for Intermodal Transportation Safety and Security (UCITSS), the U.S. Department of Transportation, and the National Science Foundation under grant numbers FTA-FL-26-7102-01, 2117760900 and 2117770900, and 0453463, respectively. The authors would like to acknowledge the work of the REU students Pierre Rosado and Alfredo Pérez, and Sasha Dos Santos, who participated in the programming aspects of this project.

REFERENCES

- [1] Federal Communication Commission 911 services web site, accessed August 24, 2006, <http://www.fcc.gov/911/enhanced/>
- [2] The Open Mobile Alliance (OMA), "The Mobile Location Protocol," <http://www.openmobilealliance.org>
- [3] WAP Forum, WAP-165, "WAP Push Architectural Overview," Nov. 1999.
- [4] The Parlay Group, <http://www.parlay.org>
- [5] J. Hightower and G. Borriello, "A Survey and Taxonomy of Location Systems for Ubiquitous Computing," *IEEE Computer*, vol. 34, no. 8, 2001, pp. 57-66.
- [6] A. Kupper, *Location-Based Services: Fundamentals and Operation*, New York: Wiley, 2005.
- [7] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*, John Wiley, 2005.
- [8] Sun Microsystems, "Java Remote Method Invocation Specification," 1998.

- [9] G. Pospischil, J. Stadler, and I. Miladinovic, "A Location-based Push Architecture using SIP," *4th Int'l. Symp. Wireless Pers. Multimedia Commun.*, Sept. 2001.
- [10] M. Handley et al., "SIP: Session Initiation Protocol," IETF RFC 2543, Mar. 1999.
- [11] M. Spanoudakis et al., "Extensible Platform for Location Based Services Provisioning," *Proc. 4th Int'l. Conf. Web Info. Systems Engineering Wksp. (WISEW)*, 2004.

BIOGRAPHIES

SEAN J. BARBEAU [M] (barbeau@cutr.usf.edu) is a research associate at the Center for Urban Transportation Research and a part-time Ph.D. student in the Department of Computer Science and Engineering at the University of South Florida. His research interests include intelligent mobile and ubiquitous systems, including location-aware distributed systems utilizing mobile phones.

NEVINE LABIB GEORGGI (georggi@cutr.usf.edu) works at the Center for Urban Transportation Research at the University of South Florida where she investigates the impacts of employer-based policies and programs on the transportation system. Other areas of interest and expertise over the past 18 years include using ITS applications to collect travel data, enhancing the transit rider experience, preparing ITS planning and integration guides, alcohol-related safety research, travel behavior, travel-related statistical analysis, trip generation models, transportation survey design and analysis, project development and environmental studies, transportation resource information center development, and specialized library development. She holds an M.S.C.E. from the University of South Florida (2000) and a B.S. in civil engineering, from Cairo University (1984).

MIGUEL A. LABRADOR [SM] (labrador@cse.usf.edu) received his M.S. in telecommunications and Ph.D. degree in information science with concentration in telecommunications from the University of Pittsburgh, in 1994 and 2000 respectively. Before joining the University of South Florida as an assistant professor in the Department of Computer Science and Engineering in 2001, he worked at Telcordia Technologies, Inc. in the Broadband Networking Group of the Professional Services Business Unit. His research interests are in design and performance evaluation of computer networks and communication protocols for wired, wireless, and optical networks. He has served as Technical Program

Committee member of many IEEE conferences and is currently member of the Editorial Board of *Computer Communications* (Elsevier Science). He is the former Secretary of the IEEE Technical Committee on Computer Communications (TCCC) and the Chair of the IEEE VTC 2003 Transport Layer Protocols over Wireless Networks Symposium.

RAFAEL A. PEREZ (perez@cse.usf.edu) is professor of computer science and engineering, and dean of academics of the College of Engineering at the University of South Florida. He received M.S. and Ph.D. degrees in electrical engineering from the University of Pittsburgh in 1967 and 1973, respectively. Before joining the University of South Florida as an assistant professor in the Department of Computer Science and Engineering in 1983, he worked as a project manager with Westinghouse International Company. His research interests are in artificial intelligence, neural networks, and genetic algorithms. He has also served as coordinator for the IEEE Computer Society Latin America Distinguished Visitor's Program, program evaluator for the Computing Accreditation Commission of ABET, and mentor for McNair Scholar's Program for Underrepresented Minorities.

PHILIP L. WINTERS (winters@cutr.usf.edu) joined the Center for Urban Transportation Research (CUTR) at the University of South Florida as TDM program director in 1993. He has more than 25 years of experience with transportation demand management (TDM) research, planning, operations, training, and evaluation. Prior to joining CUTR, he worked for two and a half years in corporate relocation and TDM consulting, and 10 years directing a regional nonprofit TDM program in Virginia. Among his program's recent accomplishments are the development of the *Transportation Management Association Handbook*, *Vanpool Pricing and Financing Guide*, and *Worksite Trip Reduction Model and Manual*. He manages the National TDM and Telework Clearinghouse, and co-hosts the "Learn More. Travel Less" Netconference series in partnership with the Association for Commuter Transportation (ACT), which has connected as many as 150 attendees in 15 locations at one time. He also created the TRANSP-TDM listserv, which has nearly 1000 subscribers. He is member of the Transportation Research Board's Committee on Transportation Demand, past editor of ACT's *TDM Review* for five years, and member of the Institute of Transportation Engineers' Transportation Planning Council Executive Committee.

User-location information along with advances in information technology and communications are facilitating the development of many personalized, multimedia, interactive, community-oriented applications that will add convenience and safety to people's everyday lives.