Event-Driven Messaging Services Over Integrated Cellular and Wireless Sensor Networks: Prototyping Experiences of a Visitor System

Yu-Chee Tseng, *Senior Member*, Ting-Yu Lin, Yen-Ku Liu, and Bing-Rong Lin

***Abstract*—As one of the killer applications, instant messaging has become a simple yet efficient tool for peer-to-peer communications in data networks. In telephone networks, short message services are gaining more popularity as well. However, this kind of services typically operates in their own respective networks and is triggered by fairly simple events (such as pushing a "send" button or prescheduling the transmission at later time). A promising direction is to trigger instant messages by environmental information from the physical world. In light of this, this paper proposes to establish an event-driven messaging service over a cellular-and-sensor-integrated network. We have prototyped a system which adopts Global System for Mobile Communication (GSM) as the cellular network, and Bluetooth technology as the sensor network. The latter is to realize a Bluetooth surveillance network with location-sensing capabilities to be deployed within an office building area. While using other technologies is possible, GSM and Bluetooth are two dominating technologies in telephone and data networks. So the proposed technology is immediately feasible, given the fact that many handsets are already Bluetooth-enabled.**

Through this combination, we demonstrate a *visitor system (VS)* that offers several attractive features/services for visitors arriving at an office. First, messaging services in VS are driven by preconfigured events which can be collected from the Bluetooth surveillance network. Simple events might be a person entering/leaving a space, while complicated events might be a compound logic statement involving multiple users and multiple locations in VS. Second, we believe that the proposed system justifies the potential of cross-network applications and services. Third, the proposed architecture takes a modular approach by dividing the system into several subsystems according to their functionality. Logically dispatching jobs is the key to future extensions and further value-added services. The system architecture and implementation details are reported. Performance analyses are presented to model the detection latency of a Bluetooth sensor network.

***Index Terms*—Bluetooth, context awareness, Global System for Mobile Communication (GSM), instant message, mobile computing, sensor network, short message service (SMS), wireless communication.**

Manuscript received February 1, 2004; revised December 1, 2004. The work of Y.-C. Tseng was supported in part by the NSC Program for Promoting Academic Excellence of Universities under Grant 93-2752-E-007-001-PAE, in part by Computer and Communications Research Laboratories, ITRI, Taiwan, in part by Intel, Inc., and in part by the Institute for Information Industry and MOEA, R.O.C., under the Handheld Device Embedded System Software Technology Development Project and the Communications Software Technology Project.

Y.-C. Tseng, Y.-K. Liu, and B.-R. Lin are with the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsinchu 30010, Taiwan (e-mail: yctseng@csie.nctu.edu.tw; yklau@csie.nctu.edu.tw; linnbiro@csie.nctu.edu.tw).

T.-Y. Lin is with the Computer and Communications Research Laboratory, Industrial Technology Research Institute, Hsinchu 310, Taiwan, R.O.C. (e-mail: tylin@csie.nctu.edu.tw).

Digital Object Identifier 10.1109/JSAC.2005.845623

I. Introduction

PEER-TO-PEER messaging has become a common way of communications in people's daily lives. In mobile phone systems, such as Global System for Mobile Communication (GSM) and NTT DoCoMo i-mode, short message services are gaining more revenues year-by-year according to the Internet service provider (ISP) statistics [13], [14]. For IP-based data networks, instant messaging software, such as ICQ and MSN [8], [10], is also popular for nowadays Internet users. In addition, an interesting project called Hubbub [15], which aims at providing mobile users with impromptu information exchange services, is recently reported. However, this kind of services typically operates in their own respective networks and is triggered by fairly simple events (such as pushing a "send" button or prescheduling the transmission at later time).

On the other hand, with the rapid development of wireless data networks, the potential service range of instant messages can be further extended, if cross-network solutions can be provided. In this work, we adopt Bluetooth [12], which is one of the dominating short-range wireless radio systems, as our platform. Bluetooth is characterized by small size and low cost, and has drawn much attention from both academia and business worlds [4], [7], [9], [18], [20]. It is also available immediately, given the fact that many laptops, PDAs, and handsets are already Bluetooth-enabled. Main applications of Bluetooths nowadays include cable replacement and simple voice/data exchange.

Motivated by the above observations, this paper proposes an *event-driven messaging service* over an integrated GSM-IP-Bluetooth network. We use Bluetooth as a sensor network for surveillance and location-tracking purposes in an indoor environment. GSM is used to support instant messaging services. These networks are integrated under Internet protocol (IP) networks. Targeting at providing visitor-friendly services in an office environment, our system is referred to as the visitor system (VS). VS offers several attractive features/services. First, messaging services in VS are driven by preconfigured events which can be collected from the Bluetooth sensor network. Simple events might be a person entering/leaving a space, while complicated events might be a compound logic statement involving multiple users and multiple locations in VS. Through logic operators such as "AND," "OR," and "INVERSE," we can provide new types of event-driven instant messaging services with novel applications. Second, we believe that the proposed system justifies the potential of cross-network applications and services. In addition, integrated services across different

networks release people from a used-to-be-confined service area. Migration between networks is transparent to users in VS. Third, the proposed architecture takes a modular approach by dividing the system into several subsystems according to their functionality. Logically dispatching jobs is the key to future extensions and further value-added services. This feature provides application programmers with the flexibility to customize their own visitor systems. The deployment cost of VS should be low because it takes advantage of existing cellular networks and embedded Bluetooth devices that are already built in many portable devices. Only some servers are needed in the backbone networks.

The rest of this paper is organized as follows. In Section II, we review several existing messaging services and sensing systems. Section III discusses our VS in more details. Section IV addresses the performance aspects and possible extensions of VS. Finally, Section V concludes the paper.

II. Background and Related Works

Sections II-A and B review existing messaging systems and location-aware sensor networks, respectively. A new opportunity is to seamlessly bridge a messaging system and a location-sensing platform to provide a cross-network service. In Section II-C, we will review some existing cross-network architectures, and point out the main features that distinguish our VS from those works.

A. Instant Messaging Services

We present messaging services in telecommunication systems first, followed by those in IP-based networks.

- Short message service (SMS) in GSM [13]: As part of the GSM Phase 1 standard, SMS is a store-and-forward service providing text-based messages originated from and sent to mobile phones. Currently, the SMS center is in its second-generation, which is capable of handling more messages in a more efficient and reliable way. Furthermore, national SMS internetworking and international SMS roaming allow mobile users to communicate short messages freely between different operators. As GSM is evolving to encompass high-speed packet data services (e.g., GPRS), SMS is expected to support longer messages. In addition, potential multimedia applications are under development. For example, the emerging multimedia messaging service (MMS) supports combinations of text, audio, picture, and video.
- NTT DoCoMo i-mode/FOMA [14]: The i-mode platform developed by Japan NTT DoCoMo provides a convenient interface for content providers over mobile phone telecommunication systems. This mobile service allows users to access more than 75,000 Internet sites, as well as specialized services such as e-mail, online shopping and banking, ticket reservations, and restaurant advice. Charges in i-mode are based on the volume of data transmitted, rather than the amount of time remaining connected. With the success of i-mode, in October 2001, NTT DoCoMo further launched the FOMA services based on wideband-code-division multiple-access (CDMA) third-generation (3G) networks. FOMA supports high-speed data transmissions from 64 to 384 kb/s, thus enabling more attractive high-speed services.
- ICQ [10]: Originated from the pronunciation of "I seek you", ICQ enables Internet users with capabilities of sending instant messages, online chatting, as well as files exchanges. Besides peer-to-peer instant messaging, ICQ also allows users to conduct group conferences or participate in online games. Furthermore, another ICQ-based software, ICQphone, allows users to enjoy voice communications over PCs/IP phones via IP telephony technologies.
- MSN messenger [8]: Similar to ICQ, the MSN messenger (or. NET messenger) delivered by Microsoft also creates an environment for users to pleasantly enjoy instant message, chat, file transfer, and video conference. An MSN messenger client program will connect to an MSN messenger server over the Internet. The client then exchanges data with other clients through the server, which keeps track of users' current points of Internet attachment. The instant messages are forwarded by the server, and processed by the destined client itself. MSN messenger was not bundled with Windows operating systems until Windows XP, in which the messaging service is incorporated in a program known as Windows Messenger.

B. Wireless Sensor Networks (WSNs)

The WSN has many applications in disaster rescue and environment monitoring applications. Reference [2] discusses important design issues of WSN. Advances in microsensor and communication technologies have made it possible to manufacture small and cost-effective WSNs [17], [22]. Several attractive applications of WSNs have been reported [3], [19]. Below, we review several important systems.

- Active badge [27]: The active badge system provides a means for locating individuals within a building by determining the locations of active badges worn by them. An active badge is a small device that can transmit a unique infrared signal periodically. Important places within a building such as offices and corridors can be equipped with networked sensors (i.e., badge readers) to detect these badges. The locations of badges (and, hence, their wearers) are then sent from the sensors to a server on the backbone.
- RFID [11]: Radio frequency identification (RFID) technology has been used for years in transportation applications (rail car tracking, toll collection, and vehicular access control) and inventory management and monitoring. A typical RFID system consists of some tags, readers, and processing servers on the backbone network. A reader can transmit radio waves to trigger antennas of tags. A tag then can convert the signal into DC voltage to charge itself and in turn emit radio signals with identification information for readers to interpret. As a result, the sizes of tags can remain very small due to this batteryless design. Readers then can pass the collected data on to the backbone servers to exploit further applications.

- Berkeley Smart Dust [28]: The Smart Dust mote developed by U.C. Berkeley contains a microcontroller. Periodically the microcontroller gets readings from sensors, which can measure different physical or chemical stimuli such as temperature, ambient light, vibration, acceleration, and air pressure, and processes the data. It also occasionally turns on its optical receiver to see if anyone is trying to communicate with it. In response to a message or upon its own initiative, the microcontroller can use its corner cube retroreflector or laser to communication with other motes or base stations. A smart dust is designed to be only a few cubic millimeters in volume, with sensing, communicating, and self-powering capabilities. Given the limited available storage, one major challenge is to incorporate power-saving designs in all functional parts.
- MIT cricket [23]: The cricket indoor location system is developed by MIT. The system is based on TinyOS and supports continuous object tracking. Following the time difference of arrival (TDoA) model, the cricket compass uses a combination of radio frequency (RF) and ultrasound technologies to determine the position and orientation of a device. A program called listener is used to analyze beacons from objects. A maximum-likelihood estimator is used to correlate a sequence of location information.

C. Existing Cross-Network Services

In Sections II-A and II-B, we have reviewed several instant messaging and sensor networking technologies. We observe that combining these two technologies would be very powerful to provide many novel applications. The sensor networks can detect and report environment-related information and events. Then through instant messaging systems, these events can be sent to the outside world for processing immediately. These events may trigger human or application programs to take actions, which may be further fed back into the sensor networks. Cross-network services may be an important trend in the near future. Several works have addressed integration wireless local area network (WLAN) and 3GPP/CDMA2000/GPRS [1], [5], [24]. As far as we know, no much work has addressed the integration of instant messaging and sensor networking technologies as our work does.

III. Event-Driven Messaging Services: A Visitor System

We have prototyped a VS by combining a Bluetooth-based sensor network and the GSM SMS instant messaging service. Our VS targets at providing novel and friendly services for visitors coming to a building, such as a company headquarters. The devices at visitors' hands can be a Bluetooth-enabled laptop/palmtop or a Bluetooth-enabled GSM WAP-compatible handset. We deploy Bluetooths in important locations within a building, which are connected to a backbone wireline network, to form a sensor network. We use the Bluetooth-based sensor network to identify the locations of visitors/employees in the building, as well as to provide data services. Messaging services are delivered to visitors/employees via either Bluetooth or GSM SMS.

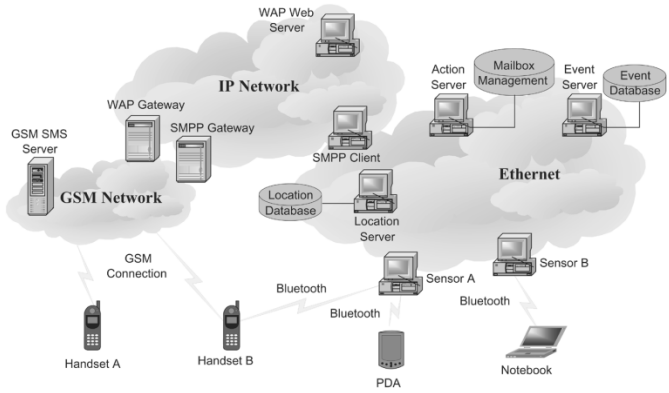


Fig. 1. Visitor system architecture.

GSM SMS has become a pervasive telecommunication data service, and has triggered many interesting applications. On the other hand, Bluetooth is characterized by low price, low-power consumption, and short-range communication, making it a potential choice for supporting positioning in an indoor, office environment. Nowadays, many mobile devices are already equipped with Bluetooths. As a result, we adopt Bluetooth and GSM as our prototyping platform. Through the implementation, we have developed several interesting location-based "event-driven" services.

A. System Architecture

The architecture of our VS is illustrated in Fig. 1, which includes user equipments, sensors, backbone networks, and several servers. Next, we detail each component separately.

In our VS, user equipment could be laptops, palmtops, or GSM handsets with WAP capability. Since we adopt Bluetooth as the sensing technology, all participant mobile terminals should be Bluetooth-enabled. One exception is a single GSM handset that is only WAP-enabled, in which case the user can only receive our instant messageing services, but can not be detected while traveling in our VS coverage area. Every mobile terminal registered in VS is assigned a unique logical ID. We classify user equipment into two categories: *program-loadable*, such as laptops and PDAs, and *program-unloadable*, such as mobile phones. For program-loadable terminals, a client program will be installed such that users can configure their own events and query databases. For program-unloadable terminals, a WAP interface is provided in VS so that users can still enjoy most functionalities provided to program-loadable devices.

A Bluetooth-based sensor network is deployed at important venues in an office building. These sensors are responsible for location sensing. At every predefined 30 s, each sensor will perform the "inquiry" procedure to discover newly arriving devices. The inquiry interval is in fact adjustable (we will further discuss this parameter in Section IV). Once a new device is discovered, the sensor will obtain its Bluetooth medium access control (MAC) address and logical ID, and notify the location server to establish a MAC-ID-location mapping in its database. Sensors also keep MAC-ID mapping at their local databases. Users' subsequent communications are done via logical IDs. Note that in addition to user location tracking, sensors are also responsible for relaying information between users and servers,

which includes event configuration with the event server and instant message exchange with the action server.

The backbone network of VS is an integrated IP data network over the Bluetooth sensor network and GSM cellular network. The information exchange from network to network is transparent to end-users. In this way, communications are no longer confined to the limited office area. In particular, with the convenient and robust SMS service supported by GSM, we successfully extend the service area of VS. Two gateways, WAP and short message peer to peer (SMPP), are used to bridge these networks, which will be elaborated later on.

Our VS consists of a number of servers to support event-driven messaging services and a variety of functions. A modular system architecture is adopted in our design, so as to facilitate future extensions and improve flexibilities for third-party application developments. Note that those logically separated components can actually be executed on the same machine through multiple process threads. Next, we describe servers in VS in more details.

- **Location server**: The location server maintains the user-location mapping in a database. Whenever a sensor detects a newly joining device, it will register the device ID with the location server. There is a two-way handshaking between the location server and sensors. Only after the registration is completed can the corresponding mapping be committed to its local database. Once a device is detected absent by a sensor in a prespecified interval, the sensor will notify the location server to deregister that device. Other servers, such as event server or action server, will query the location server to obtain such user-location mapping information.
- **Event server**: In VS, actions are triggered by events. An event could be an instant message request, time event, location event, or their combinations. An instant message request is simply a message to be delivered to a terminal and it will be processed immediately. In VS, there are two basic events: *time event* and *location event* (detailed definitions to be presented in Section III-B). A user can configure a basic event or a compound event with the event server. The event server will parse users' requests into basic events and store them into its event database. Time events will be stored locally, while location events will be delivered to the Location server. Time events are triggered by a local clock at the event server. Location events are stored at the location server and triggered by users' movements. Once a location event becomes true, a signal will be sent from the location server to the event server. The event server will then check its event database and, on a user event becoming true, trigger the action server to take proper actions. Once the action request is sent to the action server, the corresponding event record will be removed from the event database.
- **Action server**: The responsibility of the action server is to really deliver the services. Basically, the Action server is triggered by commands from the event server. Upon receiving an action instruction, the action server will first issue a query to the location server to retrieve the receiver's location information. According to the receiver's profile, if it is recognized as a GSM handset, the service will be forwarded onto the GSM network via several assistant servers and delivered to the destined mobile phone. On the other hand, for services destined for a laptop/palmtop currently located in the VS coverage area, the action server acts as a gateway to relay the services to the receiver.

  To increase reliability, we associate a mailbox with each device ID. According to the classification of *mailbox-based schemes* in [6], our mailbox is stationary with "push" message delivery. However, we do not synchronize between mobile terminal migration and mailbox forwarding. Instead, we adopt a retransmission strategy. Before a message is successfully delivered, it is buffered locally at the action server. Each buffered message will be retransmitted at predefined time intervals. To avoid wasting too much resource on retransmissions, the retransmission interval is doubled each time a failure is experienced. We set up a lifetime of 2 hours for each message. Once the lifetime expires, the corresponding message will be deleted from the buffer.
- **SMPP client**: The short message peer to peer (SMPP) protocol is an open industry standard messaging protocol designed to simplify integration of data applications with wireless mobile networks such as GSM, time-division multiple-access (TDMA), and CDMA. The current system that we are using is SMPP v3.3 (the newest version is v5.0, dated February 2003). In order to provide messaging services to GSM handsets, we developed a SMPP v3.3 client program. To transmit, we need to specify the SMPP Gateway Host Name, Port Number, User Name, Password, and Telephone Number in the URL text. The SMPP client is responsible for creating a SMPP connection to SMPP gateway. The message content is attached at the end of the URL text, which will be formatted by the SMS protocol and forwarded to the GSM SMS server for delivery.
- **WAP web server**: WAP provides an interface for mobile handsets to access the Internet. GSM handset users can configure their requests/services through our WAP server. As a result, we set up a WAP web page to accept users' configuration request. This web page is developed using wireless markup language (WML) and WML script. Any WAP-compatible GSM handsets can connect to the WAP web server and enjoy available VS services. On receipt of a request, the WAP server will contact the event server, which is responsible for processing the request.
- **Other assistant servers**: In the prototyped system, we utilized several existing interfaces available in GSM telecommunication system. Those assistant servers include SMPP Gateway, WAP Gateway, and GSM SMS server.

B. Events and Actions

One of the main features that distinguishes our VS from other messaging systems is its event-driven and location-aware

service capability. The concept of events is useful when determining whether to take an action or not. Furthermore, a combination of multiple basic events may support more complicated novel applications. For example, a general manager may want to be notified that a meeting is ready only when the scheduled time is up and all other meeting members have shown up in the meeting room. A staff may want to be notified if he/she is not in the meeting room within 3 min before the meeting time.

In light of this, services in VS are accomplished by event-driven actions, which can be formulated by a number of rules, each with the following format:

$$\langle EvntService\rangle = On\langle EvntVal\rangle Do\langle Action\rangle.$$

In a VS, there are two types of basic events.

- **L**ocation event: This simply includes someone entering or leaving an area.
- **T**ime event: Four types of temporal concept can be specified: absolute time, relative time, time interval, and periodical time. For example, we can specify "10 a.m. on 7/9/2003," "20 min. after ID leaves sensor X," "from 10:00 a.m. to 10:15 a.m.," and "every 5 min."

More complicated events can be combined from basic events through logical operators, such as "AND," "OR," and "NOT."

We summarize the definition of events in the following EBNF-like recursive grammar:

$$\begin{aligned}
\langle EvntVal\rangle &= \langle SubEvntVal\rangle AND\langle EvntVal\rangle \mid \\
&\quad \langle SubEvntVal\rangle OR\langle EvntVal\rangle \mid \\
&\quad \langle SubEvntVal\rangle \\
\langle SubEvntVal\rangle &= (\langle EvntVal\rangle) \mid NOT\langle SubEvntVal\rangle \\
&\quad \langle SnglEvntVal\rangle \\
\langle SnglEvntVal\rangle &= \langle LocEvnt\rangle \mid \langle TimeEvnt\rangle \\
\langle LocEvnt\rangle &= \langle ID\rangle\langle Rel\rangle\langle Sensor_X\rangle \\
\langle Rel\rangle &= ENTER \mid LEAVE \\
\langle TimeEvnt\rangle &= \langle min/hr/dat/mon/yr\rangle \mid \\
&\quad \langle TimeOfEvnt(LocEvnt) + min/hr\rangle \mid \\
&\quad \langle min/hr/dat/mon/yr, \\
&\qquad min/hr/dat/mon/yr\rangle \mid \\
&\quad \langle min/hr/dat/mon/yr, period\rangle.
\end{aligned}$$

Note that $TimeOfEvnt(LocEvnt)$ returns the actual time when a particular location event happens.

Actions in a VS are all message transmission commands. The service types include unicast, multicast, geocast, and broadcast. The receiver ID (i.e., target of the transmission) can be a user, a group of users, or users in a particular space. The mapping between service type and receiver ID is illustrated in Table I.

TABLE I
SERVICE TYPES AND CORRESPONDING RECEIVER IDS IN VS

	Receiver ID	Service Type
1	Single user	Unicast
2	Group	Multicast
3	Sensor	Geocast
4	ALL	Broadcast

C. Service Operations and Message Flows

Based on the above definition of event-driven actions, we next discuss the operations of a VS and some real message flows. We will explain through two examples in this section.

The first example demonstrates how one is reminded by an instant message when a meeting is ready. Suppose that manager Mike calls for a meeting involving staffs Alice and Bob at 10:00 a.m. However, Mike does not want to be kept in waiting if Alice and Bob is late. So Mike uses a notebook installed with the VS client program and sets up an event service: "On (time $= \langle 10:00, 10:10\rangle$) AND (Alice ENTER Sensor X) AND (Bob ENTER Sensor X) Do Unicast(Mike, "staff meeting is ready")".

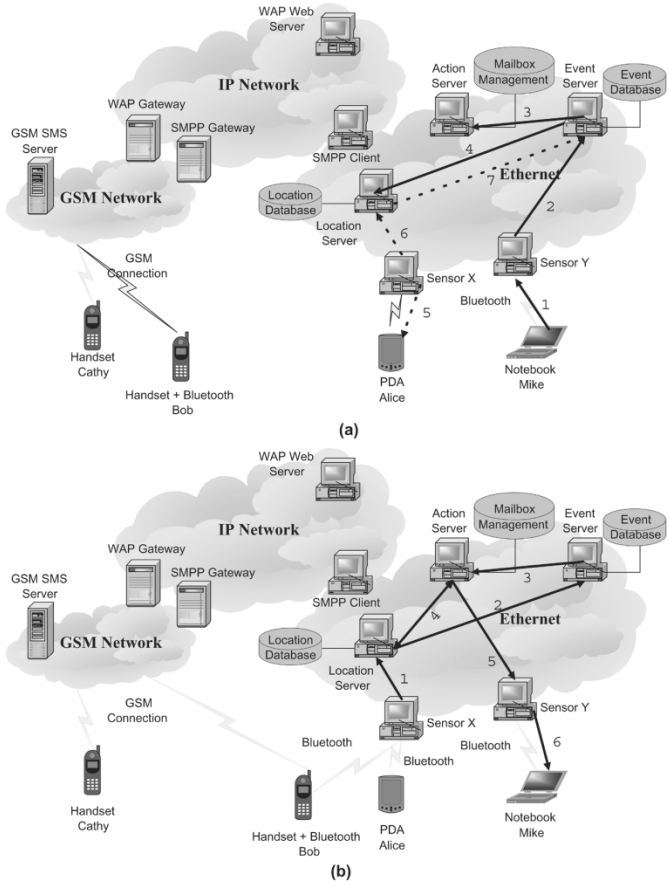


Fig. 2. Message flows for a meeting-reminding instant message service.

Suppose that Mike is now in sensor Y. We illustrate the message flow in Fig. 2(a). At first, Mike configures the above event-driven action through sensor Y. In this case, since Mike uses a notebook with a Bluetooth, the Bluetooth is used as a data network to connect to the event server. Through our VS client program, sensor Y will relay the request. On receiving the request, the event server parses the request, which includes a time event and two location events together with a unicast action. The time event will be stored locally at the event server, while the two location events will be forwarded to the location server. The action server will also be contacted to register the unicast action together with an index to the action. Note that

the index will be stored at the event server, for which to trigger the corresponding action later on. Next, suppose that Alice, who has a PDA installed with VS, arrives at sensor X first. Then, such an event will be captured by sensor X, which will conduct a location update with the location server. (Note that this update action is independent of the content of the location server; it is conducted whenever a sensor finds a device entering or leaving its coverage.) On receiving the update of Alice's location, the location server checks its local database and finds that such an event should be signaled to the event server.

Later on, as shown in Fig. 2(b), as Bob, who has a Bluetooth-enabled handset, arrives at the meeting room. Such an entering event will also be captured by sensor X and triggered to the location server, and then the event server. Depending on whether the time event has become true or not, the Event server may trigger the unicast action immediately or at later time. Once all events are satisfied, the action server will be triggered by the proper action index. The action server then queries Mike's current location and sends the reminder to Mike. On receiving an acknowledgment message from Mike, all related information at these servers will be cleaned.

Next, suppose that another staff Cathy needs Mike to sign a document but finds that Mike is still in the meeting. Suppose that Cathy does not want Mike to be disturbed, but wants to send herself a message three minutes after Mike leaves the meeting room. Therefore, Cathy may submit a request: "On (time = TimeOfEvnt(Mike LEAVE Sensor X) + 3) Do Unicast(Cathy, "Mike is available now")."

In Fig. 3(a), we show the message flow for Cathy's request through a WAP-enabled handset. In this case, the handset does not need to be Bluetooth-enabled. The configuration procedure will be done through our WAP server. Note that in our implementation, we regard the cellular network as a single special sensor in our VS network. So the WAP server will go through our SMPP client to submit the event-driven action to the event server. In this way, the event server has a consistent view on users in VS, and all handset users are regarded as under a special "GSM sensor." Fig. 3(b) shows the rest of the message flow after Mike leaves the meeting room. Note that since Cathy is recognized as resident in the special "GSM sensor" network, the SMPP client will be contacted to provide the instant message service. The SMPP client will then contact the SMPP Gateway, which will connect to the GSM SMS server, who actually delivers the short message.

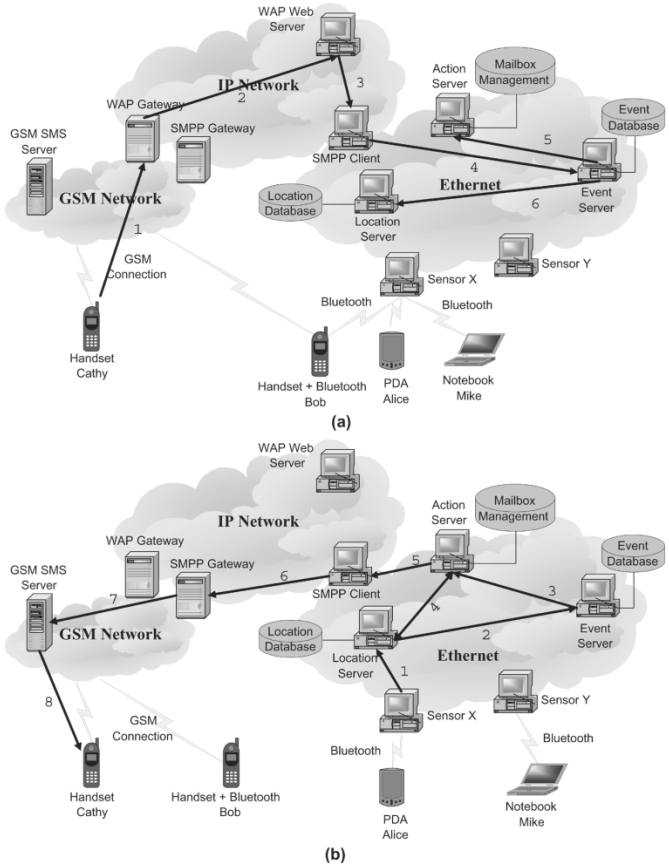


Fig. 3. Message flows for a relative-time instant message service.

D. Implementation Details and User Interfaces

In this section, we demonstrate the configuration interfaces of VS. We first explain the interfaces for laptops/palmtops, followed by the interactive WAP web pages established for handsets.

Fig. 4(a) is the main user interface of the VS client software for program-loadable devices. On the right-hand side, the asterisk marks the current location of the user. The user can send an instant message by selecting "Instant Msg" as the event type. After filling in receiver ID and outgoing message, clicking the OK button will submit the request. The user can configure a basic event as follows. To set up a basic time event, the event type field should be declared first, followed by filling a parameter in the time/location field with one of the following formats: @mm/dd/yy/hh:xx (for absolute time), +xx{m,s,h} (for relative time, xx minutes/seconds/hours after the current time), and #xx{m,s,h} $\sim$ yy{m,s,h} (for time interval, every xx time units, lasting for yy time units). To set up a basic location event, we first declare the event type field followed by entering desired sensor with the format: @SensorID. Then, target ID field should be declared by filling in either a single device or a group of devices.

To configure a compound event, one should click on the "event expr" in the main window. Then, the event expression window, as shown in Fig. 4(b) will pop up. Up to eight basic events (numbered A–H) can be defined. To define a basic event, the "set event" button should be clicked, which will bring up the interface as shown in Fig. 4(c). The setup procedure is similar to what is discussed earlier. After configuring all basic events, the user can enter a logic statement in the field logical expression in Fig. 4(b).

We also provide online Help through the main window. In addition, buttons USER LIST and set group facilitate users to check users currently visible by VS and to set up a group ID for multicast services. The Attachment option, though not discussed yet, supports simple file transfers between users. At the current stage, VS only allows text-based messaging services due to the limited bandwidth and MAC restrictions imposed by

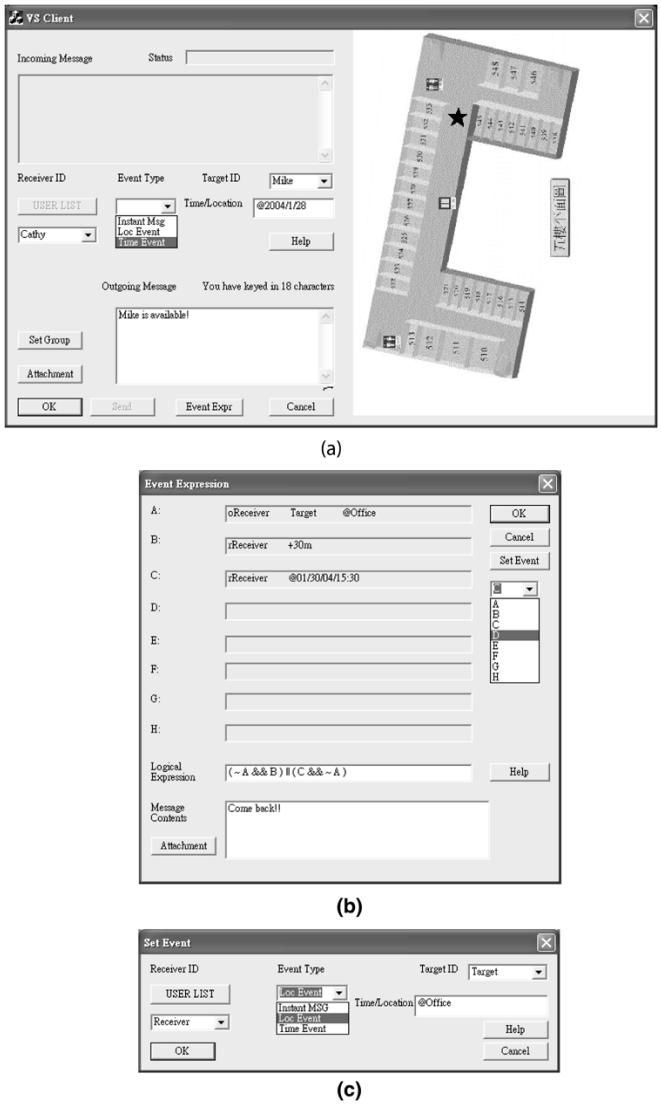


Fig. 4. Interfaces for laptop/palmtop terminals in VS.

Bluetooth. Multimedia applications are being considered and will be directed to our future work.

The WAP interface is designed following the similar philosophy as discussed above, except that the setup procedure is decomposed into several small WAP pages. We are unable to provide all details due to space limitation. Fig. 5 shows some WAP web pages for handset configurations.

Our current implementation is based on IBM X21/X24 notebooks with Bluetooth USB dongles (with Broadcom chipsets and firmware specification v1.1b), HP iPAQ H3630 with Bluetooth CF cards, and Sony Ericsson T68i handsets. Windows XP SP1 is used, and the server/client programming environment is Visual C++ 6.0. The number of active slaves per piconet is limited to 7. Although theoretically a lot of slaves can be supported in one piconet, since the current system does not support the *park* mode, only seven visitors can be tracked by one Bluetooth sensor. Users are required to register before they can use our system. We consider security and authentication as independent issues, so currently everyone can register into VS. Devices without going through the registration procedure will not be tracked by our system.

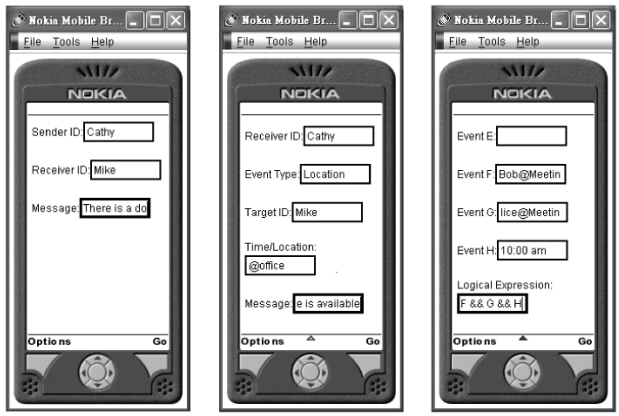


Fig. 5. Interactive WAP web page interfaces for handsets in VS. (a) Instant message. (b) Single event configuration. (c) Hybrid event configuration.

IV. Performance Evaluations and Discussions

Our VS relies on Bluetooth sensors to track the movement of objects. Below, we propose a model to evaluate the sensing capability of a given Bluetooth network and discuss some possible extensions of VS.

A. Sensing and Detecting Capability

In a VS, sensor nodes are required to perform Bluetooth inquiry periodically. We experiment on different inquiry intervals from 10 s to 1 min. Intuitively, the larger the inquiry interval, the longer the connection latency will be. However, a short inquiry interval means that sensors will spend significant time discovering devices and thus messages buffered at sensors may experience delays or even dropping. Due to its special inquiry frequency hopping rules, Bluetooth suggests that each inquiry should last for at least 10 s. Otherwise, the "inquiry" from a sensor and the "inquiry scan" from a device are likely to miss each other. Our experimental results suggest that an inquiry interval of 25–30 s performs better, resulting in an average connection time of about 15 s. Several works have indicated that Bluetooth suffers from long discovery time. (The Bluetooth v1.2 is claimed to have better connection efficiency. However, the firmware update is not available to us at the time of implementation. A lot of works have tried to improve the inquiry speed of a single Bluetooth piconet [16], [21], [25], [26], [29], but this is regarded as an independent issue in our work.)

Our VS is built upon a multipiconet environment. Below, we propose an analytical model to evaluate the sensing capability of such an environment. We are given a sensing field A with a number of arbitrarily deployed sensors. Each Bluetooth sensor has a circular sensing range, and sensors may have overlapping sensing ranges. Sensors are not synchronized in time (in terms of inquiry timing). Given any user appearing in any location in A, we would like to analyze the average latency L such that the user can be detected by any of the sensors after it appears.

1) Analytical Results: Let A_i be the area in A that is covered by exactly i sensors $1 \leq i \leq n$, where n is the total number of sensors. (We assume that every point in A is covered by at least

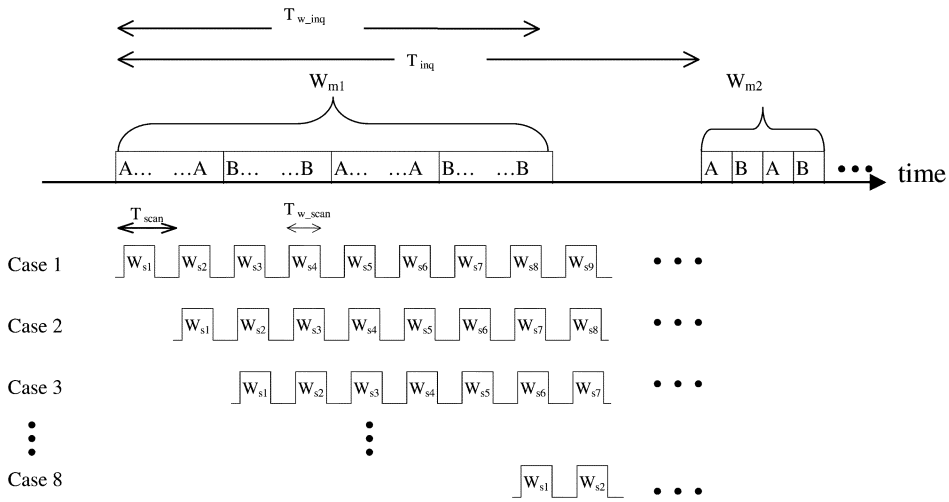


Fig. 6. Eight possible cases for the slave to start its inquiry scan in the analysis of D.

one sensor; otherwise, analyzing the average detection latency is meaningless.) Also, let L_i be the latency such that a user is detected by any piconet/sensor after it appears in a point in A_i and starts it first inquiry scan window. It is easy to see that

$$L = \frac{A_1}{A} \times L_1 + \frac{A_2}{A} \times L_2 + \cdots + \frac{A_n}{A} \times L_n.$$

Below, we show how to derive L_i. The derivation is based [16], where a probabilistic model is given to analyze the inquiry latency between a master–slave pair. Here, we further extend the result to multiple-sensor-covered scenarios.

In the following analysis, let $T_{\text{inq}}, T_{w\_\text{inq}}, T_{\text{scan}}$, and T_{w_scan} be the lengths of one inquiry interval, one inquiry window, one inquiry scan interval, and one inquiry scan window, respectively. We follow the recommendation of Bluetooth that T_{w_inq} is one half of a sequence of 256 A/B trains [12]. Therefore, a slave's inquiry scan has two chances to match with the frequencies on which the master sends ID packets. First, we analyze the *expected inquiry latency*, denoted by D, for a slave to be found by a master if the former performs inquiry scan during the latter's inquiry window (note that this is a special case of inquiry). Let $W_{s1}, W_{s2}, \ldots$, be the sequence of inquiry scan windows of the slave, and $W_{m1}, W_{m2}, \ldots$, that of inquiry windows of the master. We consider all possible positions where the first window W_{s1} appears in the first window W_{m1}. This is illustrated in Fig. 6. Basically, there are eight cases to be considered. In the following, D_j, $1 \le j \le 8$, is the detection delay when W_{s1} appears in the jth $(1/8)$ window of W_{m1}

$$D_1 = \frac{1}{32} \times T_{\text{scan}} + \frac{1}{32} \times (4 \times T_{\text{scan}}) + \frac{14}{32} \times (2 \times T_{\text{scan}}) \quad (1)$$

$$D_2 = \frac{1}{32} \times (3 \times T_{\text{scan}}) + \frac{15}{32} \times T_{\text{scan}} \quad (2)$$

$$D_3 = \frac{1}{32} \times T_{\text{scan}} + \frac{1}{32} \times (4 \times T_{\text{scan}}) + \frac{14}{32} \times (2 \times T_{\text{scan}}) \quad (3)$$

$$D_4 = \frac{1}{32} \times (3 \times T_{\text{scan}}) + \frac{15}{32} \times T_{\text{scan}} \quad (4)$$

$$D_5 = \frac{1}{32} \times T_{\text{scan}} + \frac{1}{32} \times \left(T_{\text{inq}} - 4 \times T_{\text{scan}} + D_1^{(1)}\right) + \frac{14}{32} \times (2 \times T_{\text{scan}}) \quad (5)$$

$$D_6 = \frac{1}{32} \times (T_{\text{inq}} - 5 \times T_{\text{scan}} + D_1) + \frac{15}{32} \times T_{\text{scan}} \quad (6)$$

$$D_7 = \frac{1}{32} \times T_{\text{inq\_scan}} + \frac{15}{32} \times (T_{\text{inq}} - 6 \times_{\text{scan}} + D_1) \quad (7)$$

$$D_8 = \frac{1}{2} \times (T_{\text{inq}} - 7 \times T_{\text{scan}} + D_1). \quad (8)$$

To understand the meanings of D_j, we refer to the discussion of [16]. A slave will repeatedly scan all frequencies of train A in 16 consecutive inquiry scan windows, followed by all frequencies of train B in 16 consecutive inquiry scan windows. For example, for the calculation of D_1, there are 32 possibilities, depending on the frequency in W_{s1}, where the slave waits for the master's ID packets. These possibilities can be classified into four subcases, as illustrated in Fig. 7. It is even possible that the slave may have to wait until the next inquiry window due to frequency mismatch in the current inquiry window (e.g., D_5, D_6, D_7, and D_8 are such cases). So we get the expected delay of D as follows:

$$D = \frac{1}{8} \sum_{j=1}^{8} D_j. \quad (9)$$

Next, we extend the derivation for areas covered by α, $1 \le \alpha \le n$ sensors (i.e., areas of A_α). The situation is more complicated because the inquiry windows of the α masters may or may not overlap with each other in a T_{inq} interval. In the following analysis, for simplicity, we assume that the time axis is divided into slots such that each slot is of length T_{scan}, and T_{inq} is a multiple of T_{scan}. Also, we assume that each sensor's inquiry window starts at the beginning of a slot. Such simplification allows us to analyze the discovery latency on a discrete time axis.

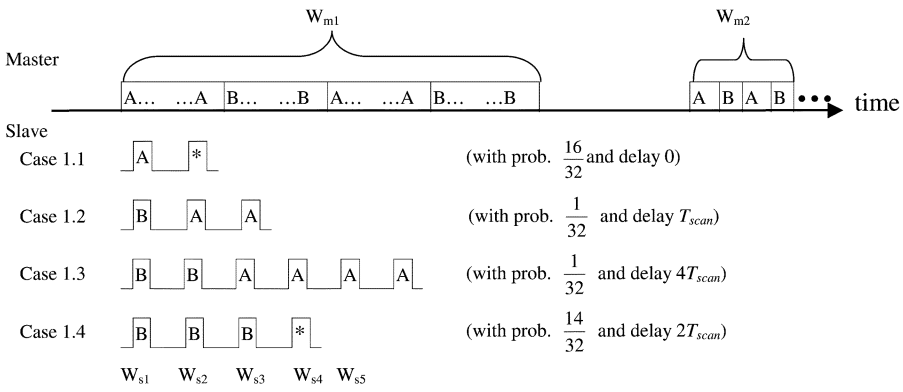


Fig. 7. Calculation of D_1. There are four subcases for frequency-matching between the master and the slave. A "$*$" means a "Don't Care" frequency, because a matching has already appeared in the previous inquiry scan window.

Without loss of generality, let the slave start its first inquiry scan window at time 0, and let T_i be the starting time of the ith sensor's first *complete* inquiry window[1] after time 0, $0 \le T_i < T_{\text{inq}}$, $i = 1, 2, \ldots, \alpha$. We first analyze the probability that the first frequency-matching appears between the slave and the ith sensor at the end of the slave's kth inquiry scan window, denoted by $P_1(k, T_i)$, $1 \le k$. Note that although the value of k appears to have no bound, from the above analysis of D, we see that $P_1(k, T_i) = 0$ for all $k > (T_{\text{inq}}/T_{\text{scan}}) + 5$. This is because in the design of Bluetooth's A/B trains, after a slave tries five trains to receive an ID packet from a master in the same inquiry window, a frequency-matching is guaranteed to appear (cases 1, 2, 3, and 4 in Fig. 6 are examples). So we only need to derive $P_1(k, T_i)$ for $k \le (T_{\text{inq}}/T_{\text{scan}}) + 5$. Next, we separate the discussion into two cases.

Case 1) $0 \le T_i \le T_{\text{inq}} - T_{w\_\text{inq}}$. In this case, the first frequency-matching must appear within the first five slots after time T_i. This is illustrated in Fig. 8(a). So we have

$$P_1(k, T_i) = \begin{cases} 0, & k = 1, 2, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \ge \frac{T_i}{T_{\text{scan}}} + 6. \end{cases} \tag{10}$$

The above probabilities can be derived from the formulation of D_1.

Case 2) $T_{\text{inq}} - T_{w\_\text{inq}} < T_i < T_{\text{inq}}$. In this case, although the first complete inquiry window of the sensor appears after the first slot, the tail of the master's previous inquiry window should overlap with the first few inquiry scan windows of the slave. An example is illustrated in Fig. 8(b). The first frequency-matching may appear in the tail of the previous inquiry window or the current window. The probabilities can be derived from the formulation of $D_2, D_3, \ldots, D_8$

$$P_1(k, T_i = T_{\text{inq}} - 7 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ 0, & k = 2, 3, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \left(1 - \frac{16}{32}\right) \times \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \left(1 - \frac{16}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \left(1 - \frac{16}{32}\right) \times \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ \left(1 - \frac{16}{32}\right) \times 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \left(1 - \frac{16}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \ge \frac{T_i}{T_{\text{scan}}} + 6 \end{cases} \tag{11}$$

$$P_1(k, T_i = T_{\text{inq}} - 6 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{1}{32}, & k = 2 \\ 0, & k = 3, 4, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \left(1 - \frac{17}{32}\right) \times \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \left(1 - \frac{17}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \left(1 - \frac{17}{32}\right) \times \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ \left(1 - \frac{17}{32}\right) \times 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \left(1 - \frac{17}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \ge \frac{T_i}{T_{\text{scan}}} + 6 \end{cases} \tag{12}$$

$$P_1(k, T_i = T_{\text{inq}} - 5 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{15}{32}, & k = 2 \\ 0, & k = 3, 4, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \left(1 - \frac{31}{32}\right) \times \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \left(1 - \frac{31}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \left(1 - \frac{31}{32}\right) \times \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ \left(1 - \frac{31}{32}\right) \times 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \left(1 - \frac{31}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \ge \frac{T_i}{T_{\text{scan}}} + 6 \end{cases} \tag{13}$$

[1]Note that the first *complete* inquiry window does not mean the first frequency-matching has to appear after this point. Refer to the discussion of Case 2) for more details.

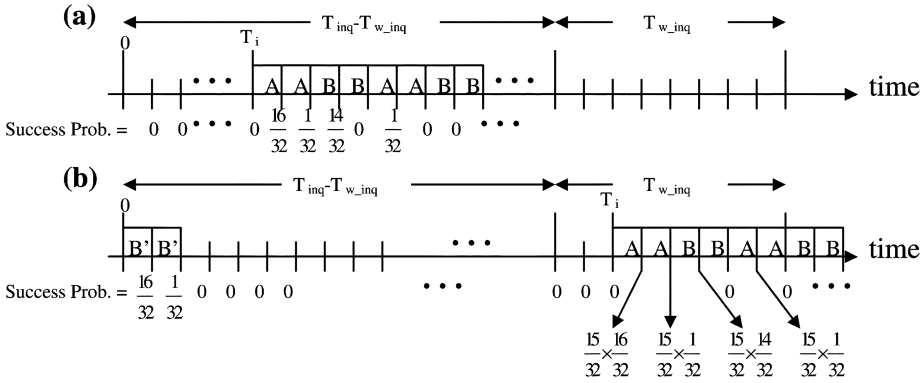


Fig. 8. Analysis of P_1. (a) Example of Case 1). (b) Example of (12) of Case 2).

$$P_1(k, T_i = T_{\text{inq}} - 4 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{1}{32}, & k = 2 \\ \frac{14}{32}, & k = 3 \\ 0, & k = 4, 5, \ldots, \frac{T_i}{T_{\text{scan}}} \\ \left(1 - \frac{31}{32}\right) \times \frac{16}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 1 \\ \left(1 - \frac{31}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 2 \\ \left(1 - \frac{31}{32}\right) \times \frac{14}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 3 \\ \left(1 - \frac{31}{32}\right) \times 0, & k = \frac{T_i}{T_{\text{scan}}} + 4 \\ \left(1 - \frac{31}{32}\right) \times \frac{1}{32}, & k = \frac{T_i}{T_{\text{scan}}} + 5 \\ 0, & k \geq \frac{T_i}{T_{\text{scan}}} + 6 \end{cases} \tag{14}$$

$$P_1(k, T_i = T_{\text{inq}} - 3 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{15}{32}, & k = 2 \\ 0, & k = 3 \\ \frac{1}{32}, & k = 4 \\ 0, & k \geq 5 \end{cases} \tag{15}$$

$$P_1(k, T_i = T_{\text{inq}} - 2 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{1}{32}, & k = 2 \\ \frac{14}{32}, & k = 3 \\ 0, & k = 4 \\ \frac{1}{32}, & k = 5 \\ 0, & k \geq 6 \end{cases} \tag{16}$$

$$P_1(k, T_i = T_{\text{inq}} - 1 \times T_{\text{scan}}) = \begin{cases} \frac{16}{32}, & k = 1 \\ \frac{15}{32}, & k = 2 \\ 0, & k = 3 \\ \frac{1}{32}, & k = 4 \\ 0, & k \geq 5 \end{cases}. \tag{17}$$

The example in Fig. 8(b) represents the calculation of (12). The sensor starts performing its current inquiry procedure at time $T_{\text{inq}} - 6 \times T_{\text{scan}}$ so the tail of its previous inquiry window will overlap with the slave's first and second inquiry scan windows. At the first slot, the frequency-matching will happen only if the slave listens on one of the 16 frequencies in train B, so the success probability for $k = 1$ is $(16/32)$. At the second slot, the first frequency-matching will happen only if the slave listens on the last frequency in train A' at the first slot and on the first frequency in train B' at the second slot. So the success probability for $k = 2$ is $(1/32)$. If the slave did not receive any ID packet during the first two slots, the next inquiry window will appear at time $T_i = T_{\text{inq}} - 6 \times T_{\text{scan}}$. Note that the frequencies in trains A and B have no relation with those in trains A' and B'. So the success probability for $k = (T_i/T_{\text{scan}}) + 1$ is conditional to the failure in the first two slots [with a probability = $1 - (17/32)$]. Since there are 16 frequencies in train A, the success probability for $k = (T_i/T_{\text{scan}}) + 1$ is $(1 - (17/32)) \times (16/32)$. The rest of the derivation of (12) is similar.

Next, we define $P_2(j, k, T_1, T_2, \ldots, T_\alpha)$ to be the probability that: 1) no frequency-matching appears in the first k slots between the slave and sensors numbered from 1 to $(j-1)$; 2) no frequency-matching appears in the first $(k-1)$ slots between the slave and sensors numbered from $(j+1)$ to α; and 3) the first frequency-matching appears between the slave and the jth sensor in the kth slot. We derive

$$\begin{aligned} P_2(j, k, T_1, T_2, \ldots, T_\alpha) = & \prod_{i=1}^{j-1} \left(1 - \sum_{n=1}^{k} P_1(k, T_i)\right) \\ & \times \prod_{i=j+1}^{\alpha} \left(1 - \sum_{n=1}^{k-1} P_1(k, T_i)\right) \\ & \times P_1(k, T_j). \end{aligned} \tag{18}$$

Note that the first, second, and the last terms in (18) are the probabilities for the afore mentioned 1), 2), and 3), respectively. It follows that the probability that the first frequency-matching will appear in the kth slot is

$$P_3(k, T_1, T_2, \ldots, T_\alpha) = \sum_{i=1}^{\alpha} P_2(i, k, T_1, T_2, \ldots, T_\alpha). \tag{19}$$

Finally, we can derive the expected inquiry latency for the areas in A_α

$$\begin{aligned} L_\alpha = & \left(\frac{T_{\text{scan}}}{T_{\text{inq}}}\right)^{\alpha} \sum_{i_1=0}^{(T_{\text{inq}}/T_{\text{scan}})-1} \sum_{i_2=0}^{(T_{\text{inq}}/T_{\text{scan}})-1} \cdots \\ & \sum_{i_\alpha=0}^{(T_{\text{inq}}/T_{\text{scan}})-1} \sum_{k=1}^{(T_{\text{inq}}/T_{\text{scan}})+5} \\ & \times (P_3(k, i_1 \times T_{\text{scan}}, i_2 \times T_{\text{scan}}, \ldots, i_\alpha \times T_{\text{scan}}) \\ & \times (k-1) \times T_{\text{scan}}). \end{aligned} \tag{20}$$

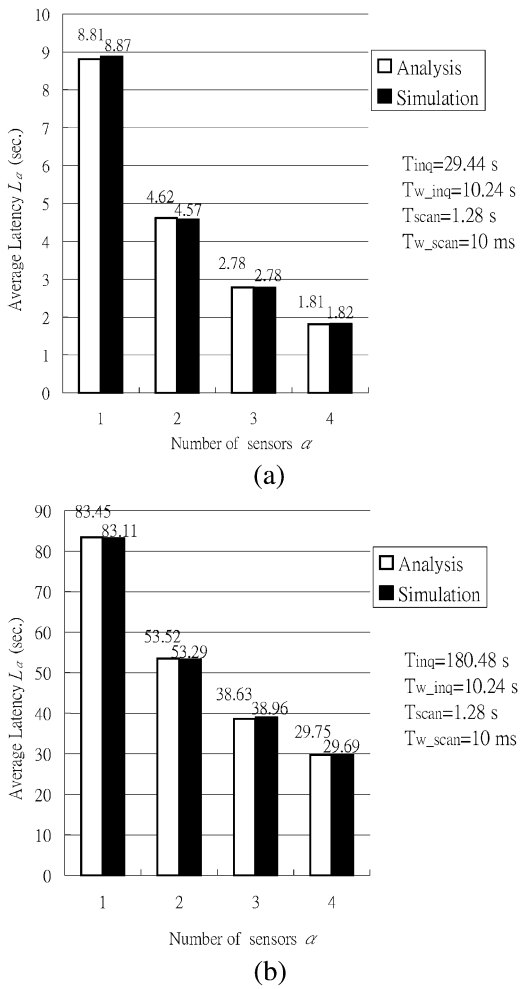


Fig. 9. Average latency $L\alpha$ versus number of sensors α with (a) $T_{\rm inq} = 29.44$ s and (b) $T_{\rm inq} = 180.48$ s.

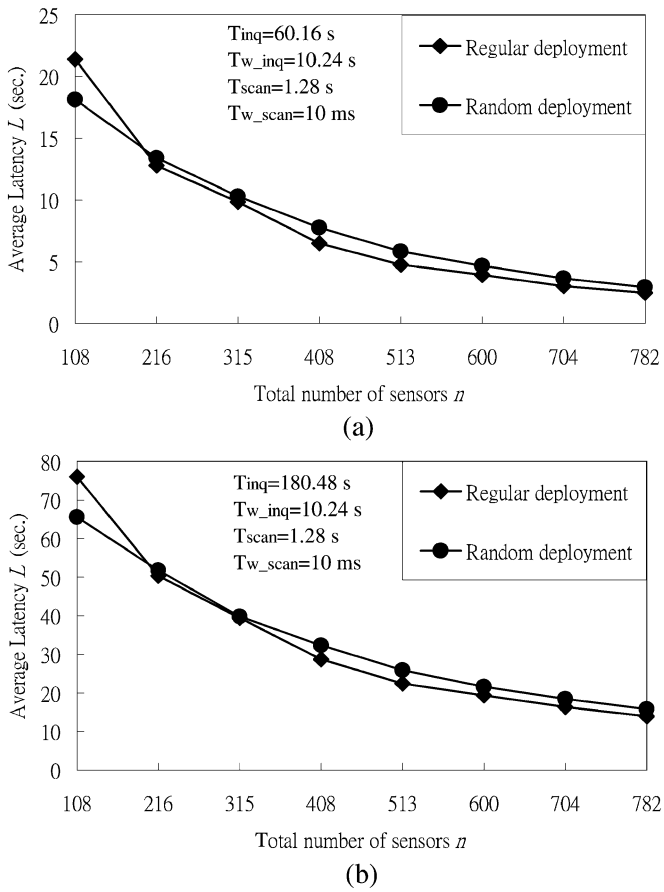


Fig. 10. Average latency L versus total number of sensors n for regularly and randomly deployed networks. (a) $T_{\rm inq} = 60.16$ s. (b) $T_{\rm inq} = 180.48$ s.

Note that in (20), the first α $\sum$'s enumerate all combinations of $T_1, T_2, \ldots, T_\alpha$, and the last $\sum$ is to account for all nonzero probabilities for P_3(So k is bounded by $(T_{\rm inq}/T_{\rm scan}) + 5$).

2) Simulation Results: We have also conducted some simulations to verify our analytical results. We first evaluate the discovery latency L_α when an object appears in an area covered by α sensors. In our simulation, sensors do not synchronize their clocks. Fig. 9(a) and (b) shows the latency when $T_{\rm inq} = 29.44$ s and $T_{\rm inq} = 180.48$ s, respectively, with $T_{w\_{\rm inq}} = 10.24$ s, $T_{\rm scan} = 1.28$ s, and $T_{w\_{\rm scan}} = 10$ ms (note that according to our analysis, $T_{\rm inq}$ is a multiple of $T_{\rm scan}$). As can be seen, our analytical results are very close to simulation results.

We have also simulated the network-level latency. A sensing field of size 500×500 is simulated. The radius of each Bluetooth coverage area is 30 units. We vary the number of sensors n and observe the detection latency L of the sensing field. Sensors are deployed either in a regular manner or in a random manner. In the former case, the typical triangular deployment is adopted, where each three nearby sensors form a triangle of the same side length. We adjust the side length to adjust the value of n. Generally, as n increases, the latency L will decrease, which is reasonable. The regular deployment has a lower latency than the random deployment for both cases of $T_{\rm inq}$. One exception is for small values of n, where the random deployments have lower latency. This is because in random deployments, some areas could be uncovered (for example, when $n = 108$, 216, 315, and 408, the average ratios of uncovered areas are 31.51%, 10.42%, 3.81%, and 1.55%, respectively). Since such areas are not taken into account, what has shown in Fig. 10 is based on unfair comparison.

Finally, we observe the impact of $T_{\rm inq}$. In Fig. 11, we vary the value of $T_{\rm inq}$ (which is a multiple of $T_{\rm scan} = 1.28$ s). There are 408 sensors deployed in a regular or random manner. We observe the average latency L and the ratio of time that a sensor spends on inquiry (excluding the inquiry time, a sensor can do data communications.) As can be seen, a larger $T_{\rm inq}$ will increase the latency L linearly, but the inquiry overhead is reduced. So one should properly tune $T_{\rm inq}$ to account for both detection latency and the ratio of time that sensors can spend on data communications to support other activities of VS.

B. User Profile Management

Currently, VS recognizes senders/receivers by device IDs. However, one disadvantage is that device ID provides little information on the ownership of the user. It is also possible that a user owns multiple devices. The problem is that the system is unaware of which mobile device the user is carrying. One solution is to maintain a user profile, which registers all devices he/she

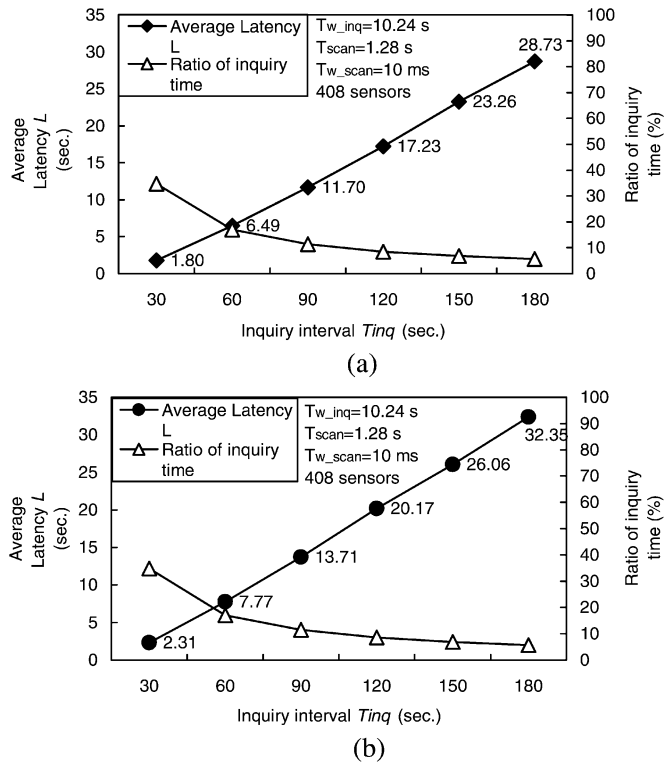


Fig. 11. Average latency L and ratio of inquiry time versus inquiry interval T_{inq} with 408 sensors. (a) Regular deployment. (b) Random deployment.

owns. Once we have such information, multicast could be a solution to this problem. With user profiles, several user-centric applications are possible.

C. More Value-Added Services

Limited by Bluetooth and GSM wireless bandwidth, the current VS supports mainly text-based services. We have experimented some graphics files, such as maps, and the transmission performance is acceptable. However, supporting multimedia streaming services requires much more works. With the popularity of GSM MMS services, one extension step is to implement MMS transmissions in VS. This will require modifications in the action server.

V. Conclusion

In this paper, we have reported how we prototyped an event-driven instant messaging application over integrated telecomm and datacomm networks. The proposed visitor system in its current stage exercises both GSM and Bluetooth technologies. With the recent release of Bluetooth v1.2, which emphasizes on faster connection and better interference mitigation, we expect to improve the performance of our system. Our VS aims to provide human-centric services. As we have predicted, unifying novel applications across heterogeneous networks is an attractive trend. From the prototyping experiences, we have shown how to separate logical servers (such as event, location, and action servers) for event-driven messaging services, and the importance of an open interface for third-party application development. Our module-based system design allows software programmers to configure their own systems by just adding or modifying components without changing the base architecture. This flexibility is critical when implementing a real system. Our VS relies on Bluetooth as sensors to track the movement of objects. As far as we know, there is no prior work which addresses the device discovery delay in an environment with multiple Bluetooth masters. Novel analyses have been presented to address this problem, and our simulation results do verify the accuracy of the analyses.

References

[1] K. Ahmavaara, H. Haverinen, and R. Pichna, "Interworking architecture between 3GPP and WLAN systems," *IEEE Commun. Mag.*, vol. 41, no. 11, pp. 74–80, Nov. 2003.
[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–114, Aug. 2002.
[3] A. Baptista, T. Leen, Y. Zhang, A. Chawla, D. Maier, W. C. Feng, W. C. Feng, J. Walpole, C. Silva, and J. Freire, "Environmental observation and forecasting systems: Vision, challenges and successes of a prototype," in *Proc. Int. Annu. Conf. Society Environ. Inf. Sci. (ISEIS)*, 2003.
[4] J. Beutel, O. Kasten, M. Ringwald, F. Siegemund, and L. Thiele, "Bluetooth smart nodes for mobile ad hoc networks," TIK Rep. 67, 2001.
[5] M. M. Buddhikot, G. Chandranmenon, S. Han, Y.-W. Lee, S. Miller, and L. Salgarelli, "Design and implementation of a WLAN/CDMA2000 interworking architecture," *IEEE Commun. Mag.*, vol. 41, no. 11, pp. 90–100, Nov. 2003.
[6] J. Cao, X. Feng, and S. K. Das, "Mailbox-based scheme for mobile agent communications," *IEEE Comput.*, vol. 35, no. 9, pp. 54–60, 2002.
[7] J. Haartsen, W. Allen, and J. Inouye, "Bluetooth: Vision, goals, and architecture," *Mobile Comput. Commun. Rev.*, vol. 1, no. 2, pp. 38–45, 1998.
[8] MSN Messenger. [Online]. Available: http://messenger.msn.com/
[9] (2001) MIT Blueware Project. [Online]. Available: http://nms.lcs.mit.edu/projects/blueware
[10] ICQ Homepage. [Online]. Available: http://web.icq.com/
[11] AIM RFID Technology. [Online]. Available: http://www.aimglobal.org/technologies/rfid/
[12] (2003) Bluetooth SIG Specification v1.2. [Online]. Available: http://www.bluetooth.com
[13] GSM Short Message Service. [Online]. Available: http://www.gsmworld.com/technology/sms/index.shtml
[14] NTT DoCoMo I-Mode. [Online]. Available: http://www.nttdocomo.com/corebiz/imode/global/
[15] E. Isaacs, A. Walendowski, and D. Ranganathan, "Mobile instant messaging through hubbub," *Commun. ACM*, vol. 45, no. 9, pp. 68–72, Sep. 2002.
[16] J.-R. Jiang, Y.-C. Tseng, and B.-R. Linn, "A mechanism for quick Bluetooth device discovery," presented at the Mobile Computing Workshop, Taiwan, 2004, . .
[17] J. Kahn, R. Katz, and K. Pister, "Mobile networking for smart dust," *Mobile Comput. Netw.*, pp. 483–492, 1999.
[18] O. Kasten and M. Langheinrich, "First experiences with Bluetooth in the smart-its distributed sensor network," in *Proc. Workshop Ubiquitous Comput. Commun.*, Oct. 2001.
[19] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proc. 1st ACM Int. Workshop Wireless Sensor Netw. Applicat.*, 2002, pp. 88–97.
[20] N. Milanovic, A. Radovanovic, B. Cukanovic, A. Beric, N. Tesovic, and G. Marinkovic, "Bluetooth ad hoc sensor network," Univ. Belgrade, Belgrade, Yugoslavia, Tech. Rep., 2002.
[21] P. Murphy, E. Welsh, and J. P. Frantz, "Using Bluetooth for short-term ad hoc connections between moving vehicles: A feasibility study," in *Proc. IEEE Int. Conf. Veh. Technol. Conf. (VTC)*, May 2002, pp. 414–418.
[22] G. Pottie and W. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, pp. 51–58, May 2000.
[23] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proc. ACM/IEEE Int. Conf. Mobile Comput. Netw. (MobiCom)*, Aug. 2000, pp. 32–43.
[24] A. K. Salkintzis, C. Fors, and R. Pazhyannur, "WLAN-GPRS integration for next-generation mobile data networks," *IEEE Wireless Commun.*, pp. 112–124, Oct. 2002.
[25] T. Salonidis, P. Bhagwat, and L. Tassiulas, "Proximity awareness and fast connection establishment in bluetooth," in *Proc. Int. Conf. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, 2000, pp. 141–142.

[26] F. Siegemund and M. Rohs, "Rendezvous layer protocols for bluetooth-enabled smart devices," in *Proc. Int. Conf. Arch. Comput. Syst.*, 2002, pp. 256–273.
[27] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Trans. Inf. Syst. (TOIS)*, vol. 10, no. 1, pp. 91–102, Jan. 1992.
[28] B. Warneke, M. Last, B. Liebowitz, and K. S. Pister, "Smart dust: Communicating with a cubic-millimeter computer," *IEEE Comput.*, pp. 2–9, Jan. 2001.
[29] R. Woodings, D. Joos, T. Clifton, and C. D. Knutson, "Rapid heterogeneous connection establishment: Accelerating Bluetooth inquiry using IrDA," in *Proc. IEEE Int. Conf. Wireless Commun. Netw. Conf. (WCNC)*, 2002, pp. 342–349.

Yu-Chee Tseng (S'99–M'95–SM'03) received the B.S. degree in computer science from the National Taiwan University, Taipei, Taiwan, in 1985, the M.S. degree in computer science from the National Tsing-Hua University, Hsin-Chu, Taiwan, in 1987, and the Ph.D. degree in computer and information science from Ohio State University, Columbus, in January 1994.

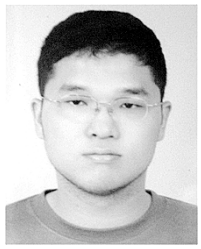
He worked for D-LINK, Inc., as an Engineer in 1990. He was an Associate Professor at the Chung-Hua University (1994–1996) and at the National Central University, Taoyuan, Taiwan, (1996–1999), and a Full Professor at the National Central University (1999–2000). Since 2000, he has been a Full Professor at the Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsinchu, Taiwan. His research interests include mobile computing, wireless communication, network security, and parallel and distributed computing.

Dr. Tseng is a member of the Association for Computing Machinery (ACM). He is a two-time recipient of the Outstanding Research Award, National Science Council, R.O.C., from 2001 to 2002, and 2003 to 2005, and a recipient of the Best Paper Award at the International Conference on Parallel Processing in 2003. Several of his papers have been chosen as Selected/Distinguished Papers in international conferences. He has guided students to participate in several national programming contests and received several awards. He served as a Program Chair in the Wireless Networks and Mobile Computing Workshop, 2000 and 2001, as a Vice Program Chair in the International Conference on Distributed Computing Systems (ICDCS 2004), and as a Vice Program Chair in the IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2004). He served as an Associate Editor for *The Computer Journal*, as a Guest Editor for *ACM Wireless Networks* (Special Issue on Advances in Mobile and Wireless Systems), as a Guest Editor for the IEEE TRANSACTIONS ON COMPUTERS (Special Issue on Wireless Internet), as a Guest Editor for the *Journal of Internet Technology* (Special Issue on Wireless Internet: Applications and Systems), as a Guest Editor for *Wireless Communications and Mobile Computing* (Special Issue on Research in Ad Hoc Networking, Smart Sensing, and Pervasive Computing), as an Editor for the *Journal of Information Science and Engineering*, as a Guest Editor for *Telecommunication Systems* (Special Issue on Wireless Sensor Networks), and as a Guest Editor for the *Journal of Information Science and Engineering* (Special Issue on Mobile Computing).

Ting-Yu Lin received the Ph.D. degree in computer science and information engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in December 2002.

She is currently a Software Engineer at the Computer and Communications Research Laboratory, Industrial Technology Research Institute. Her research interests include wireless communication, mobile computing, WLANs/WPANs, sensor networking, and energy conservation designs (software-level).

Dr. Lin was the recipient of the Phi Tau Phi Scholastic Honor Award.



Yen-Ku Liu received the B.S. and M.S. degrees in computer science from the National Chiao-Tung University, Hsinchu, Taiwan, in June 2002 and April 2004, respectively.

His research interests include wireless networks, sensor networks, and Bluetooth technology.

Bing-Rong Lin received the B.S. and M.S. degrees in computer science from the National Chiao-Tung University, Hsinchu, Taiwan, in June 2002 and June 2004, respectively.

His research interests include wireless networks, sensor networks, and Bluetooth technology.