IEEE pervasive COMPUTING
MOBILE AND UBIQUITOUS SYSTEMS
www.computer.org/pervasive

# Using Smart Phones to Access Site-Specific Services

*Eleanor Toye, Richard Sharp, Anil Madhavapeddy, and David Scott*

Vol. 4, No. 2
April–June 2005

# Using Smart Phones to Access Site-Specific Services

*The Mobile Service Toolkit is a client-server framework for developing mobile services that interact with users' smart phones. Two prototypes that employ the MST are described: a virtual queuing service and a vacation browsing service.*

**Eleanor Toye and Richard Sharp**
*Intel Research Cambridge*

**Anil Madhavapeddy**
*University of Cambridge*

**David Scott**
*Fraser Research*

Our work investigates how smart phones can augment *site-specific services*—that is, electronic services or applications that reside in a specific location. Site-specific services already exist in the form of ticket machines, electronic information kiosks, interactive product catalogues, and so on. However, integrating users' smart phones into these interactions can enhance service functionality while reducing deployment costs.

Our approach offers several benefits. By using personal information stored on smart phones, site-specific services can automatically tailor their actions to suit a particular user. Additionally, users can store information resulting from an interaction with a service on their smart phones for future reference (for example, downloading a map). The service could later push dynamic information updates to the phone via text and MMS (multimedia messaging service). Finally, users no longer have to queue to access a single, shared site-specific service (for example, a ticket machine). Instead, multiple users can connect their smart phones to the service simultaneously and access it concurrently.

Our research led us to develop the Mobile Service Toolkit: a client-server software framework supporting the development of site-specific services that exploit interaction with smart phones. We'll discuss MST and present case studies of two site-specific services implemented using it.

## Mobile Service Toolkit

The MST client software (called the Mobile Service Explorer or MSE) runs on users' smart phones. We implemented the MST server software, which runs on a device providing a site-specific service, on Linux and Windows.

Service providers run MST servers on Bluetooth-enabled PCs located in their store, restaurant, cinema, and so on. Users in close proximity to these services can connect and access them via their MSE-enabled smart phones.

The MSE performs three primary functions:

- *Connection establishment*. We use machine-readable visual tags to establish connections to site-specific services. The MSE contains image-processing software that decodes tags. Users establish connections by pointing the smart phone's embedded camera at a visual tag and pressing the Select button.
- *Personal information management*. The smart phone contains a repository of personal information. MST servers can request personal information from the MSE to provide personalized services. The MSE chooses whether or not to supply the requested information in accordance with the user's privacy policy.
- *General-purpose data entry and display*. The MST server can push user interfaces to smart phones and supports both thin-client functionality (similar to Virtual Network Computing[1]) and WAP (Wireless Application Protocol)-like user interface controls.

Figure 1 shows the architecture of a system


Published by the IEEE CS and IEEE ComSoc ■ 1536-1268/05/$20.00 © 2005 IEEE

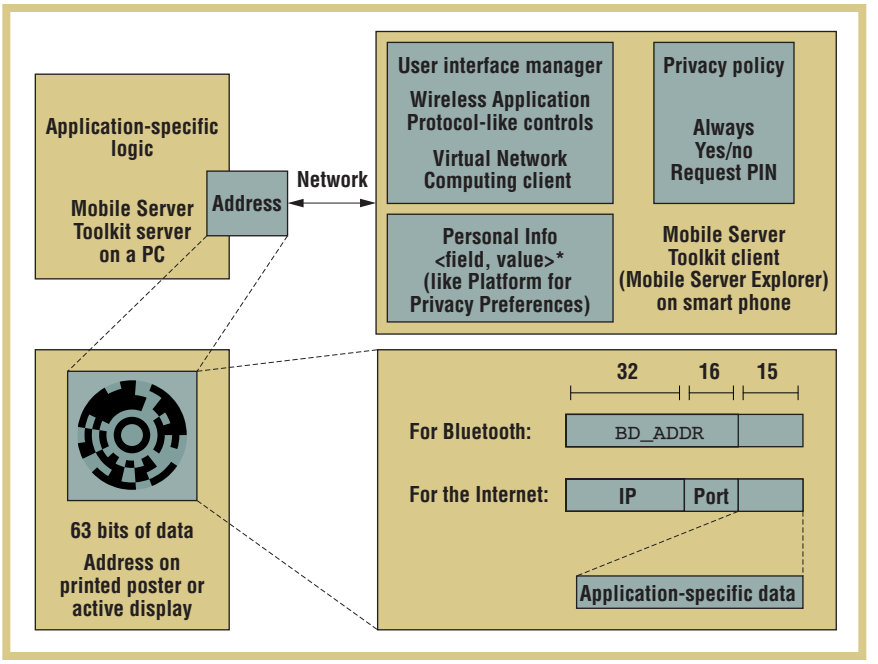


Figure 1. The architecture of systems created with the Mobile Service Toolkit.

built using the MST. The components running inside the MSE on the smart phone deal with user interface control and personal information management. We represent the MST server's address as a visual tag on a poster or active display. The box in the bottom-right corner shows two possible encoding schemes—for a Bluetooth and an Internet connection.

## Networking and connection establishment

Connecting a phone to a site-specific service requires wireless networking technology. Various choices exist, ranging from operator-provided data services such as GPRS (General Packet Radio Service) to short-range, point-to-point solutions such as IrDA and Bluetooth. Although these technologies can transfer information between smart phones and site-specific services, we based the MST's current version on Bluetooth. We chose Bluetooth because it's free (unlike GPRS, for example), low latency, and supported by numerous existing smart phones as well as other electronic devices such as PCs, laptops, digital cameras, and GPS devices.

Irrespective of the networking protocol, the greatest technical challenge is connection establishment. Users need to know when they can connect to site-specific services, and they need to connect quickly and efficiently. Because we use Bluetooth, we can easily determine an MST server's device address by using standard Bluetooth device discovery. However, this raises several usability issues. First, this device discovery is slow in practice (in a realistic environment, it's not uncommon for device discovery to take over 30 seconds). Second, in public places, Bluetooth device discovery might find an overabundance of devices, including, for example, the mobile phones and PDAs of passersby.

In previous work, we showed that you can use visual tags, which encode Bluetooth device addresses, to bypass device discovery.[2] We integrated support for this technique in the MSE. From a user's perspective, tag-based connection establishment proceeds as follows: the MSE software turns the phone's display into a viewfinder—a live video feed continually capturing frames from the phone's embedded camera. Whenever a tag is visible in the viewfinder, it's highlighted with a red crosshair on the phone's display. A user clicks on a tag by focusing on it and pressing the Select button on their phone's keypad. (For the remainder of this article, we assume that our users' smart phones have cameras. However, the MSE also supports standard Bluetooth device discovery, which you can use on smart phones without cameras.)

Visual tags for Bluetooth connection establishment are an order of magnitude faster than traditional device discovery. Our experiments showed that visual tags let you set up a connection in 1.7s on average (with a small standard deviation of 0.47).[2] This compares to an average of 17.12s for device discovery (with a much larger standard deviation of 8.6). Visual tags are also distinctive, recognizable symbols (see the example in Figure 1). So, in addition to encoding machine-readable data, they can advertise the existence of mobile services to users.

In our current scheme, our visual tags encode a 48-bit `BD_ADDR` (Bluetooth device address) and also have room for 15 bits of application-specific data. As soon as a user's phone connects to a service, the MSE transmits these application-specific data bits to the MST server. This lets us have multiple tags that all initiate a connection to the same MST server. Based on the value of the application-specific data bits in each tag, the server can then provide different functionality to users depending on which tag they clicked.

Although the MSE currently only supports Bluetooth, using a tag to establish a connection scales to other protocols. For example (see Figure 1), we could use tags to establish Internet connections by partitioning our 63 bits of data to include a 32-bit IP address, a 16-bit TCP v4 port number, and some application-specific data. More generally, we could support multiple protocols simultaneously by assigning, say, 4 bits of the tag as a protocol selection field. This selection field would define the way the client interprets the rest of the tag data.

Figure 2 shows a typical MST deploy-

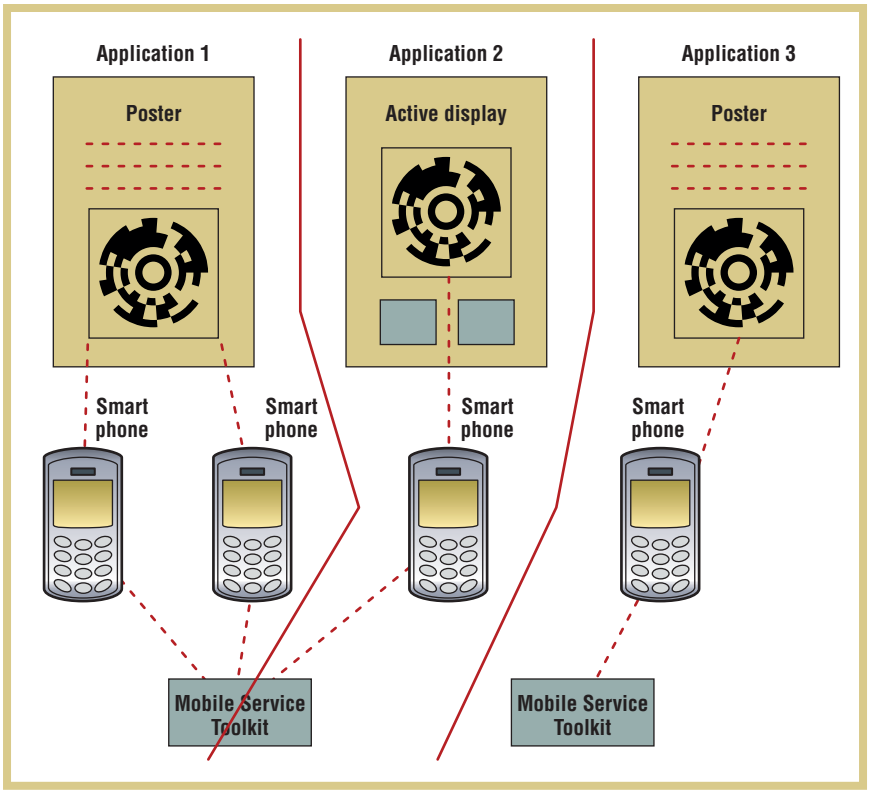


Figure 2. Hidden MST servers let users interact with personalized site-specific services—here, posters and active displays.

ment involving multiple services, servers, smart phones, posters, and displays. We segmented the diagram into three portions representing three site-specific services. Two applications (1 and 2) simultaneously share the MST server on the left. Additionally, two smart phones are simultaneously using application 1.

## Personalizing site-specific services

The simplest way to support personalization with a smart phone is for the phone to declare its identity to the service (for example, a handset ID number). Assuming that each smart phone has one primary user, the phone's unique identity enables a service to react differently for each user. Furthermore, combined with a server-side persistent state, user identity also lets a service adapt according to how a user has previously interacted with it. For example, consider using a smart phone to control an interactive game running on a display in a shop window. Whenever users return to the shop, their identities (provided by their phones) lets the game greet them and resume where they left off.

Personalization has potential for more than simple user identification. The MSE enables the smart phone to house a repository of personal information, including name, email address, home address, and mobile phone number. So, instead of filling out a form to register for a store card, you could simply authorize your MSE-enabled phone to supply the required information to the store's site-specific service.

The MSE stores personal information as a set of field name and value pairs. Several XML schemas for representing personal information already exist; we based the personal information fields we included in our initial MST implementation on the Platform for Privacy Preferences (P3P) specification.[3] By using a simple request-response protocol, site-specific services can obtain the personal information. The request packet, generated by our MST server software, includes the field name of the personal information required.

### Privacy of personal information

Some readers might wonder whether the potential privacy risks would deter people from storing personal information on their phones. However, the general public has already demonstrated its willingness to put large quantities of personal information on smart phones: the average user stores hundreds of personal text messages and contacts (including phone numbers, addresses, email, and so on). To alleviate privacy concerns, the MSE ensures that individuals always have awareness of and control over the transfer of their personal information.

The MSE lets users protect their personal information with a customizable three-level privacy policy. For each piece of personal information, a user can specify whether to

- always disclose it automatically,
- ask for a Yes or No permission to disclose it, or
- ask for a PIN before disclosing it. (If required, the user enters their PIN into the phone's keypad.)

We also provide an option that lets users turn all personal information-sharing on or off by entering a PIN. This lets users lend their phones to their friends (for example) without losing control over their personal information.

### Configuring a personal information repository

Users can configure their personal information repositories by filling in a standard electronic form. We provide two ways of doing this:

- A phone-based application lets users enter and edit their personal information. This application resembles (in look and feel) the address book software typically supplied with smart phones.

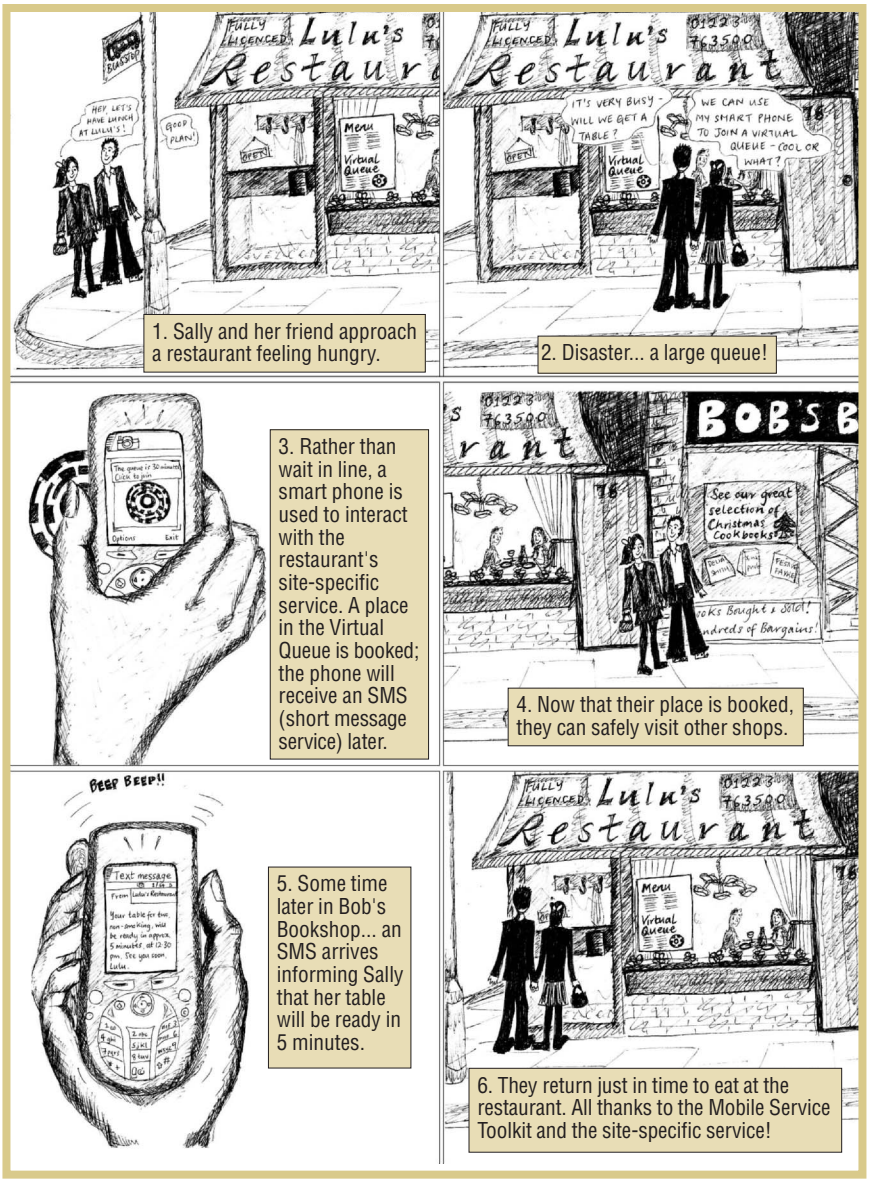


**Figure 3. Using the Virtual Queuing Service at a restaurant.**

- For those who prefer to avoid entering data on the small phone keypad, we offer a PC-based application. Users can enter and edit their personal information on their PCs and then transfer this to their smart phones over Bluetooth.

Both the phone-based and PC-based applications let users configure their privacy policies and set their PINs.

## Data entry and display devices

Site-specific services can push user interfaces to smart phones. The MST offers user interface controls such as text-entry fields, alert dialogues, confirmation dialogues, and selection lists. The MST uses a markup language similar to WML (Wireless Markup Language) to export these controls to phones and return user responses.

As well as WML-like user interface controls, the MST also provides thin-client functionality. The MSE:

- allows an MST server to push arbitrary graphics to the phone's display and
- transmits all keypress events from the phone's keypad back to the MST server in real time.

Combining WAP-like user interface controls and thin-client functionality lets designers trade off ease of implementation against customization. Opting for the WAP-like standard user interface control library makes services quick and easy to develop. However, when designers need fine-grained control over the user interface parameters, they can rely on the thin-client approach. If they take the thin-client approach, service designers must explicitly deal with the way the interface is laid out on different-sized phone displays.

## Case studies

We've implemented two realistic case studies using the MST. The case studies demonstrate how you can construct powerful applications by combining the MSE's three features (personalization, user interface control, and connection establishment). We implemented these applications on the Nokia 3650 smart phone. The Nokia 3650 runs the Symbian OS and supports programs written in either C++ or Java. The phone has 3.4 Mbytes of internal dynamic memory and supports up to 128 Mbytes of external flash memory. The phone uses a 32-bit RISC CPU based on the ARM9 series running at 104 MHz. The MST server software and auxiliary server-side application code are written in Objective Caml and run on PCs; the MSE is written in C++ and runs directly on the phone.

### Virtual Queuing Service

Many real-life scenarios require queue management. Our Virtual Queuing Service lets users reserve a place in a queue and receive updates on their queue position via their smart phones.

For example, take a busy restaurant where customers have to wait for available tables (see Figure 3). A poster in the window of the restaurant describes the Virtual Queuing Service and displays a visual tag. After reading the poster, our


Figure 4. Using a touch screen to request information from the Vacation Browsing Service; users can submit information automatically through their smart phones.

user, Sally, can click on the tag using her MSE-enabled phone to establish a Bluetooth connection with the service. As soon as the phone connects with the service, her phone displays a message containing the current queuing time and asks whether she'd like to join the queue. Next, the service pushes a text field to the phone, prompting Sally to enter her party size. Finally, the Virtual Queuing Service queries her repository of personal information, requesting her smart phone's phone number. (Depending on Sally's privacy policy, this data transfer may occur automatically or may require explicit confirmation or even PIN entry.) At this point, the server terminates the connection. Five minutes before Sally's table is ready, the Virtual Queuing Service sends her a text message telling her to return to the restaurant.

Numerous restaurants have already implemented solutions such as this using custom hardware devices that flash when a customer's table is ready. In such restaurants, customers are typically given one of these notification devices when they arrive and proceed to wait in the bar until it's activated. Our Virtual Queuing Service provides the same functionality but at significantly reduced cost to the restaurant owner (no hardware to supply) and with increased customer convenience (nothing extra to carry).

### Interactive advertising

This case study aims to demonstrate how you can integrate smart phones with existing site-specific services. Consider a touch screen deployed in a coffee shop, which lets users browse videos of a travel agent's vacation destinations. A number of services like this exist today, advertising a range of products. By bringing smart phones into the interaction, we make it easier for potential customers to follow up on information they see on the touch screen.

We augment interactive advertising services with visual tags that users can click on to receive further information (see a graphical representation in Figure 4). Our current implementation of the Vacation Browsing Service has three conventional touch screen buttons labeled "email me info" (about this destination), "send brochure" (to a home address), and "call me" (to discuss the vacation details).

As soon as users see a vacation package that interests them, they press one of the buttons on the screen. The service then offers them the option of either entering their personal information (email address, home address, or phone number) via the touch screen or clicking on a tag and transferring it automatically from their smart phone's MSE software.

To understand what goes on "under the hood" when a user clicks on a visual tag, Figure 5 shows a typical network message sequence chart. First, a user clicks on a tag and establishes a Bluetooth connection. After the phone connects with the service, it transmits the application-specific data bits read from the tag to the MST server. The server responds by requesting the appropriate personal infor-

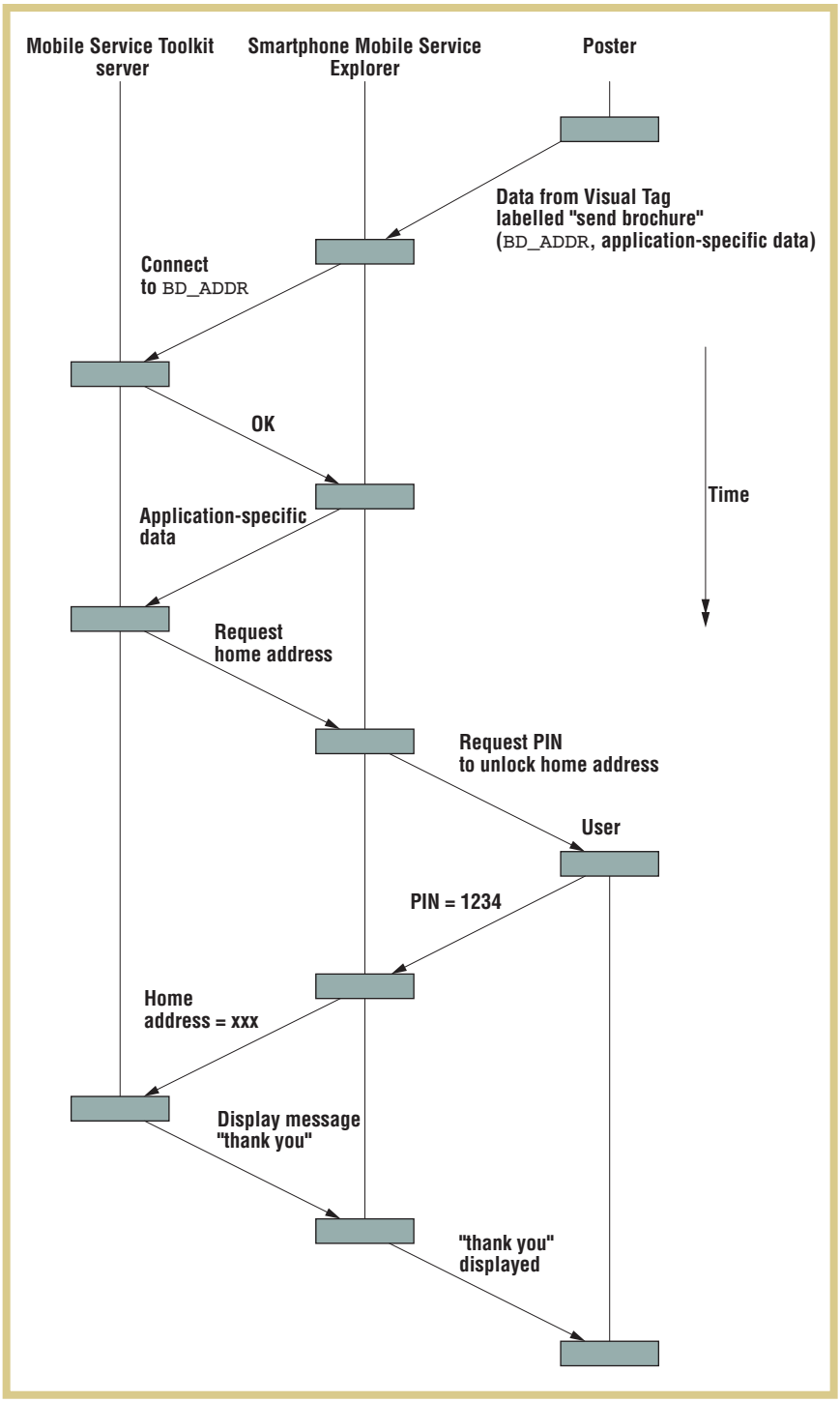


Figure 5. A network message sequence chart showing messages in the Vacation Browser Service when a user selects the "send brochure" option.

mation. (The value of the application-specific data bits determines the personal information requested.) If the user's privacy policy authorizes it, the smart phone can send the appropriate personal information back to the MST server.

## Related work

We're not the first to use out-of-band signaling to bypass Bluetooth device discovery. Eric Hall and his colleagues propose using RFID tags to "wake up" a sleeping Bluetooth radio and transmit a device's BD_ADDR.[4] Hall also reports significant power savings by turning off his Bluetooth radio and using an RFID-based rendezvous to wake it up again. Our implementation achieves the same time benefits as Hall and, because our interactions are user-initiated, we can also achieve similar power savings by only turning the Bluetooth radio on when the MSE is executed.

Our system offers other advantages over Hall's. First, we rely entirely on commodity hardware, so our work is immediately applicable. You don't need to augment existing devices with RFID readers. (Although some manufacturers are incorporating RFID readers into mobile devices, no such products exist in the consumer market space. Current offerings target specific industrial applications such as ruggedized phones for field engineers.) Instead, you could immediately deploy the MSE on the camera phones that have already shipped globally. Second, our work considers both Bluetooth device discovery and service selection whereas Hall considers only the former.

The Google Toolbar has a feature called AutoFill (http://toolbar.google.com/autofill_help.html) that automatically fills in values on Web forms using personal data stored locally on a user's PC. The AutoFill feature handles the set of data items named in the ECML (Electronic Commerce Modeling Language) specification [5] such as username, address, phone number, and credit card details. AutoFill aims primarily to speed up filling forms on the Web and, unlike our system, doesn't let the user create a privacy policy. Our approach differs primarily from AutoFill in the MSE's inherent mobility.

Liviu Iftode and his colleagues proposed a system architecture that would let users supply personal information electronically in various situations and

## the AUTHORS


**Eleanor Toye** is a researcher with Intel Research Cambridge, UK. Her research interests are in interaction design and usability; her recent work includes educational applications of RFID technology and studies of domestic DVD and video use. She received her PhD in experimental psychology from the University of Cambridge. Contact her at Intel Research Cambridge, 15 JJ Thomson Ave., Cambridge, CB3 0FD, UK; eleanor@recoil.org.

**Richard Sharp** is a senior researcher at Intel Research, Cambridge, UK. His research interests include ubiquitous computing, security, programming language design, and compiler implementation. His recent work includes research projects on investigating programming languages for multicore network processors and improving the security of complex Web applications. He received his PhD in computer science from the University of Cambridge Computer Laboratory. Contact him at Intel Research Cambridge, 15 JJ Thomson Ave., Cambridge, CB3 0FD, UK; richard.sharp@intel.com.

**Anil Madhavapeddy** is a PhD student at the University of Cambridge Computer Laboratory in the Systems Research Group and a developer on the secure OpenBSD operating system. His research interests include networked systems security and ubiquitous computing. He received his BEng in information systems engineering from Imperial College London. Contact him at Computer Laboratory, Univ. of Cambridge, 15 JJ Thomson Ave., Cambridge, CB3 0FD, UK; avsm2@cl.cam.ac.uk.

**David Scott** is a member of technical staff at Fraser Research in Princeton, NJ. His research interests include computer security, ubiquitous computing, and networking. He received his PhD in engineering from the University of Cambridge. Contact him at Fraser Research, 182 Nassau St., Ste. 301, Princeton, NJ 08542; djs@fraserresearch.org.


locations.[6] Although they highlight issues surrounding the privacy of personal information, they've not yet proposed concrete architectural solutions.

Lauri Aalto and her colleagues developed a push-based location-aware advertising system using Bluetooth and WAP.[7] This lets users receive advertisements on their smart phones based on their proximity to particular shops. By contrast, our model is entirely user-initiated.

A defining characteristic of mobile commerce is the ability to transmit payments using mobile phones, and many competing mobile payment systems exist today.[8] These systems use various schemes—from requesting authorization codes over SMS to adding new payment hardware to phone handsets. Using the MSE and a camera-equipped smart phone requires no SMS messages or extra hardware; a user can just point and click on a visual tag and authorize the transmission of their credit card number. These existing mobile payment schemes are early examples of smart-phone-based personalization. Our MST generalizes this technique, encompassing types of personal information other than payment details.

We've prototyped several site-specific services based on our new Mobile Service Toolkit framework. In addition to the ones described here, we've developed a stock querying service that let's you check whether a store's product is in stock as well as a flight information service that sends you text messages with information about your flight. Installing a site-specific service is very cheap; at a minimum, all you'll need is a single Bluetooth-enabled PC (possibly placed somewhere out of sight) and some printed posters advertising the service. Using the screen and keypad of a user's smart phone to interact with the service makes this low installation and maintenance cost possible. Additionally, the MST can augment existing site-specific services such as the touch screen-based application in our Vacation Browsing Service scenario.

In the future, we plan to build more prototype applications and deploy and evaluate them in realistic environments, soliciting feedback from real users. We also plan to extend the MST with support for more connection-establishment techniques and wireless networking technologies. For example, we plan to support both NFC (Near Field Communication)[9] and RFID-based connection establishment when these technologies become available in the consumer marketplace. We also intend to allow the MSE to connect the MST servers via GPRS, Wi-Fi, and IrDA in addition to Bluetooth. P

## REFERENCES


1. T. Richardson et al., "Virtual Network Computing," *IEEE Internet Computing*, vol. 2, no. 1, 1998, pp. 33–38.
2. D. Scott et al., "Using Visual Tags to Bypass Bluetooth Device Discovery," *ACM Mobile Computing and Communication Rev.*, Jan. 2005, pp. 41–53.
3. L. Cranor et al., *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*, World Wide Web Consortium (W3C) Recommendation, 16 Apr. 2002; www.w3.org/TR/2002/REC-P3P-20020416.
4. E.S. Hall, D.K. Vawdrey, and C.D. Knutson, "RF Rendez-Blue: Reducing Power and Inquiry Costs in Bluetooth-Enabled Mobile Systems," *Proc. 11th Int'l Conf. Computer Communications and Networks* (CCN 2002), IEEE Press, 2002, pp. 640–645.
5. D. Eastlake and T. Goldstein, *ECML v1.1: Field Specifications for E-Commerce*, RFC 3106, Apr. 2001; www.faqs.org/rfcs/rfc3106.html.
6. L. Iftode et al., "Smart Phone: An Embedded System for Universal Interactions," *Proc. 10th Int'l Workshop Future Trends in Distributed Computing Systems* (FTDCS 04), 2004, CS Press, pp. 88–94.
7. L. Aalto et al., "Bluetooth and WAP Push-Based Location-Aware Mobile Advertising System," *Proc. 2nd Int'l Conf. Mobile Systems, Applications, and Services* (MobiSys 04), ACM Press, 2004, pp. 49–58.
8. S. Wallage, "The Far East Mobile Payment Race," 27 Nov. 2003, www.thefeature.com/article?articleid=100241.
9. *Near Field Communication*, white paper, ECMA Int'l, 2004, www.ecma-international.org/activities/Communications/2004tg19-001.pdf.