Audio Networking: The Forgotten Wireless Technology

Audio networking can leverage existing smart phone components to transmit data reliably with increased simplicity and less power than more high-profile wireless communication protocols such as Bluetooth or infrared.

Anil Madhavapeddy, David Scott, and Alastair Tse
University of Cambridge

Richard Sharp
Intel Research Cambridge

Modern computing and communication devices offer a wide range of wireless communication protocols to transmit data, such as infrared, Bluetooth, and Wi-Fi. However, one technology has fallen off the radar in recent years, even though it's more ubiquitous with lower power requirements. *Audio networking* uses audible sounds as a low-bandwidth data channel and can enhance, for example, smart phone[1] usability. In that domain, audio networking offers both short-range communication with nearby devices as well as longer-range data transfer by introducing audio data packets into ongoing telephone conversations. Moreover, applying audio networking to smart phone applications doesn't change the fundamental devices people use to communicate. Instead, it can exploit the existing programmable interfaces that modern smart phones offer to augment modes of communication that people already know and use.

In this article, we'll review various modulation schemes we've worked with previously, covering how to transfer data to nearby smart phones as well as usability and security issues. We'll consider audio networking as a mechanism for introducing data packets into ongoing mobile phone calls. We'll also discuss some real-world problems reported with telephone conferencing and apply audio-networking techniques to them in a case study application.

Audio networking modulation

At an abstract level, audio networking is a simple concept: transmitters modulate data in a suitable fashion and play the resulting audio data using the host machine's speakers. (For more information on audio networking, see the related sidebar.) Receivers listen (via microphones) and demodulate incoming signals to recover the transmitted information.

However, choosing from the plethora of audio data modulation schemes is more difficult. Each has different characteristics—bit rate, transmission range, and what the data sounds like to humans. The latter characteristic, although as important as the first two, isn't one that designers of traditional wireless networking technologies have to face.

In previous work,[2] we experimented with four audio data transmission schemes:

- dual-tone multifrequency (DTMF),[3] used in touch-tone phones;
- on-off keying (OOK)[4] modulated over a variety of audible carrier frequencies;
- inaudible data transmission; and
- melodic data transmission.

Here, we briefly summarize our results, giving an overview of various audio modulation schemes and the transmission speeds they facilitate.

Conventional modulation schemes

Using standard sound hardware (internal laptop speakers and microphones and a pair of US$10 Harman/kardon HK 206 desktop speak-

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

Related Work in Audio Networking

Transmitting data over the phone network is clearly not a new idea in itself; modems have done this for decades. Additionally, our approach doesn't focus on achieving maximum bandwidth from this channel, but by multiplexing data into an existing audio channel, we focus on addressing spontaneous interaction between users.

Others have explored audio networking in the context of device-to-device and human-to-device communication.[1,2] Our contribution is to explore how to integrate audio networking with the telephone network to exploit preexisting, popular lines of communication that have evolved over the past century. Mobile phones have always had audio capability, but only with the rise of programmable smart phones have we been able to take advantage of it for data transmission.

Our telephone-conferencing application is inspired by the AT&T Laboratories' Broadband Phone (www.uk.research.att.com/bphone), which is a VoIP (voice over IP) phone with a large display and thin-client access to a suite of communications-orientated PC applications. As many workers already have a Broadband Phone's constituent parts (a PC, telephone, and Internet access) on their desks, we aim to provide users with the Broadband Phone experience using this existing hardware.

Researchers have explored computer and telephone convergence by installing applications on over 7,000 PCs integrated with a custom corporate PBX (private branch exchange).[3] They reported an "overwhelmingly positive" reaction to their enhanced telephony prototype, with 94 percent of the initial users recommending the system to their colleagues. Clearly, user demand exists for the more effective control over telephony that modern programmable smart phones can provide.

Other researchers have commented on modern telephony interfaces' complexity.[4] Today's myriad wireless networking protocols only increase this complexity, and our work on exploiting existing phone audio interfaces aims to give users simpler alternatives to wireless radio protocols.

REFERENCES

1. V. Gerasimov and W. Bender, "Things That Talk: Using Sound for Device-to-Device and Device-to-Human Communication," *IBM Systems J.*, vol. 39, nos. 3–4, 2000, pp. 530–546.
2. A. Nakayama et al., "Rich Communication with Audio-Controlled Network Robot. Proposal of 'Audio-MotionMedia,'" *Proc. 11th IEEE Int'l Workshop Robot and Human Interactive Communication*, IEEE Press, 2002, pp. 548–553.
3. J. Cadiz et al., "Exploring PC-Telephone Convergence with the Enhanced Telephony Prototype," *Proc. 2004 Conf. Human Factors in Computing Systems*, ACM Press, 2004, pp. 215–222.
4. N. Griffeth, "Making a Simple Interface Complex: Interactions among Telephone Features," *Proc. Conf. Companion on Human Factors in Computing Systems*, ACM Press, 1996, pp. 244–245.

ers), we achieved a bit rate of 20 bps using DTMF across a three-meter distance with a 0.006 percent symbol error rate (that is, receivers decoded 0.006 percent of DTMF tones incorrectly). We found that our OOK coding scheme worked better in short-range transmission (less than 1 meter) because you don't have to worry about pulse reflections at low amplitudes. Using OOK modulated over a 10-kHz carrier, we achieved a bit rate of 251 bit/s across 30 cm with a bit error rate of 4.4×10^{-5}.

Inaudible data transmission

We implemented a variant of OOK modulated over a 21.2-kHz carrier. We chose this frequency because it's greater than the maximum frequency of human hearing and less than the Nyquist limit of standard sound cards (which operate at 44.1 kHz).

Our experiments with inaudible data transmission involved broadcasting using a bit rate of 8 bps across 3.4 meters in an office with two occupants. While 8 bps is undoubtedly a low transmission rate, it suffices for broadcasting small identifiers (for example, room-grained location beacons[2,5]).

The experimental setup consisted of a Dell desktop with onboard sound (Intel AC97 Audio Codec) transmitting data through Harman/kardon HK 206 speakers. The receiver was a Sony Vaio PCG-Z600LEK laptop. The office's occupants generated ambient noise by continuing with their normal tasks (which included typing, bursts of conversation, walking around the room, moving objects and papers, answering the telephone, and drinking coffee). Under these conditions, 95 percent of the 172 16-bit identifiers transmitted were received correctly.

Melodic data transmission

The audible property of audio networking is of course a double-edged sword—it makes users aware of data transmission between their devices, but it can also become aggravating to hear it for long data transmissions. To make short-range data transmission more pleasant for users, we designed a simple melodic data transmission scheme. Although unlikely to receive critical acclaim, these "amusing little ditties" nevertheless sound surprisingly pleasant.

Our melodic data transmission scheme assumes the existence of a set of four carrier frequencies. Playing a tone at one of these carrier frequencies for a prespecified duration signals a two-bit value. Notably, instead of activating two of these frequencies simultaneously as touch-tone phones using DTMF do, we only activate one carrier frequency at a time.

At any given time, we choose four notes from the C major (Ionian) scale as our carrier frequencies. We constructed a melody by varying the carrier frequencies over time according to a pre-

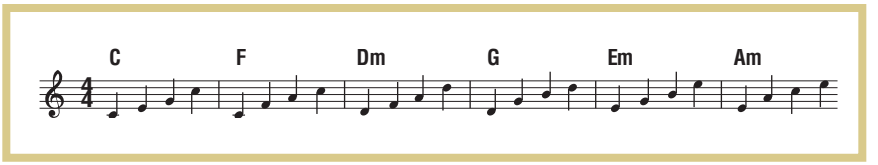


Figure 1. Notes corresponding to carrier frequencies for the "baroque" theme's first six bars.

defined sequence previously known to both the transmitter and the receiver. We employed a musician to choose carrier frequencies that, irrespective of the data being transmitted, result in pleasant-sounding melodies. We call a particular sequence of carrier frequencies a *theme*. Figure 1 shows an example of a theme that follows a chord sequence commonly found in baroque music. Each bar contains a group of four carrier frequencies. The *n*th carrier frequency (that is, the *n*th crotchet) of each bar encodes the two-bit number *n*. Our current implementation changes carrier frequencies every eight notes (that is, every 16 bits). We've made available for download audio files that encode data packets in various themes (see http://anil.recoil.org/projects/telephony.html).

We prototyped our melodic data transmission scheme by encoding data as monophonic mobile phone ringtones. (We achieved 83 bps, the fastest that the mobile phone could play ringtones, with an upper limit of 0.001 percent bit-error rate, with the phone held 2 cm from the microphone.) However, we programmed our mobile phones to play the ringtones at seven notes per second (a transmission rate of 14 bps). We observed that although both receiver and phones are capable of faster transmission, increasing the ringtones' tempo beyond eight or nine notes per second makes them less pleasant-sounding because the ear doesn't have enough time to pick out the melody. Because we wish to emphasize the data encoding's melodic nature, we see 14 bps as the upper limit of this encoding technique.

Secure, short-range audio networking

Spontaneous interaction between mobile users includes swapping contact details such as telephone numbers, home addresses, or Internet URLs. In modern smart phones, the most common protocol used for this is the Bluetooth OBEX (*ob*ject *ex*change) protocol.[6] From a user perspective, this procedure has several steps that make it time-consuming and error prone:

- Users must perform a Bluetooth device discovery to locate the other smart phone (which takes up to 10 seconds in a noise-free environment and, in practice, can take much longer).
- The user must identify the recipient from the discovered list of phone names (which, unless they've been changed from the default, typically consist of a list of generic manufacturer names).
- The phone must establish a Bluetooth connection and transmit the small amount of contact information.

Audio networking offers a dramatically simpler model. First, a smart phone encodes the desired contact information into an audible message, then you hold the two phones close together, allowing the data to transmit at a low amplitude. This eliminates the complexity of Bluetooth device discovery, instead using close physical proximity to form a low-bandwidth data channel and also to act as a simple authentication mechanism (because an attacker without sophisticated listening equipment would have to get very close to listen to the transmission).

This simplicity also affects the security model presented to smart phone users—rather than requiring them to understand Bluetooth's intricacies, we exploit the data's audible nature. By adjusting the output volume, two parties can ensure a suitable level of privacy for their information exchange.

To exchange higher-bandwidth data, you can still use audio as a local channel to bootstrap encrypted Bluetooth connections. Currently, to create an encrypted Bluetooth connection, both mobile phones must perform device discovery[6] and also *pair* the two devices by entering a matching PIN number into both phones. The Bluetooth protocol uses this PIN to generate a session key for the encryption, and the PIN's randomness decides the resulting encryption's strength. Unfortunately, smart phones depend on users to generate the PIN, typically leading to choices such as "1111," "1234," or other predictable sequences that brute-force dictionary attacks can easily decode.

Audio networking provides a solution to creating a spontaneous yet secure Bluetooth connection between smart phones. A smart phone encodes its Bluetooth *BD_ADDR* (a unique network address that every Bluetooth device has) and an automatically generated random PIN as a short audio packet. As before, by holding the two phones close to each other, the transmitting phone can send the (BD_ADDR, PIN) packet as low-amplitude audio.

This type of conversation faces two main threats:

- The attacker could somehow learn the session PIN and decode the rest of the conversation.
- The attacker could somehow fool the receiving phone into hearing a PIN that the attacker has transmitted (also known as a *man-in-the-middle attack*).

When using audio networking as we've described, the first threat would require the attacker to possess sophisticated listening equipment to detect the low-volume audio between the two mobile phones (in addition to Bluetooth-sniffing capability). Moreover, the directional speakers and microphones typically found in smart

TABLE 1
Frequencies used for encoding 4-bit values for inline data transmission across telephone lines.

	1,209 Hz	1,336 Hz	1,477 Hz	1,633 Hz
697 Hz	1	2	3	4
770 Hz	5	6	7	8
852 Hz	9	10	11	12
941 Hz	13	14	15	16

phones don't broadcast audio widely.

For the second attack, if the attacker generated a noise loud enough to convince the listening phone to use the fake information, both of the real users would hear it and become suspicious. Audio alone has this property; in a purely Bluetooth exchange, users wouldn't be able to detect a high-gain Bluetooth transmission by themselves.

Inserting audio data into calls

Because mobile phone audio channels are band-limited to only 3 kHz, we adopt a data transmission scheme that uses frequencies similar to DTMF (see table 1). Touch-tone phones already use these frequencies for functions such as dialing numbers, so they're guaranteed to have good frequency response across most phone channels. Although DTMF mandates minimum intertonal lengths of at least 40 ms, we discovered that 10 ms still achieves robust signal decoding. Each DTMF tone encodes 4 bits of data, resulting in a data transmission rate of 40 bps—sufficient to robustly transmit small chunks of information such as GPS coordinates in a few seconds.

We managed to keep our telephone data-transmission mechanism surprisingly simple. In the spirit of distraction-free computing,[7] we deliberately avoid requiring a MAC (media access control) layer for our audible packets—unlike, for example, Ethernet, which senses channel availability and randomly backs off when a collision occurs. Instead, we rely on the social-flow control between call participants, allowing them to manually schedule audio data transmission as part of their normal conversation. Consequently, call participants are never surprised by the sudden interjection of data packets during a conversation.

Figure 2 illustrates a snippet of telephone conversation between two hypothetical users, Alice and Bob. Alice phones Bob to find out where he is. Bob tells her his location ("the Castle pub"), but Alice isn't sure exactly where this is. To resolve the problem, Bob informs Alice that he's about to transmit his GPS coordinates to Alice, waits for her consent, selects his smart phone's audio transmission application, and sends the burst of data to her. Alice's smart phone intercepts this audio transmission, decodes it, and launches the GPS application to tell her how far the received location is from her current position. They then continue the phone call and agree to meet.

Alternatively, Bob could have read out the GPS coordinates to Alice, who would type them into her smart phone manually—a tedious and error-prone process. Or, Bob could have encoded the data into an SMS (short message service) message and sent it out-of-band to Alice. However, most mobile phone networks don't guarantee the timely delivery of SMS messages, and Alice might have waited a long time for the small amount of data she required. Additionally, they could have logged into an Internet-based instant-messaging application over GPRS (general packet radio service) and used that to exchange the data. However, this requires Alice to contact Bob twice—once using his mobile phone number and again using his IM address. Audio networking offers the advantage of allowing both voice and data to coexist on the same channel, so that you can send data such as Bob's GPS coordinates during the conversation.

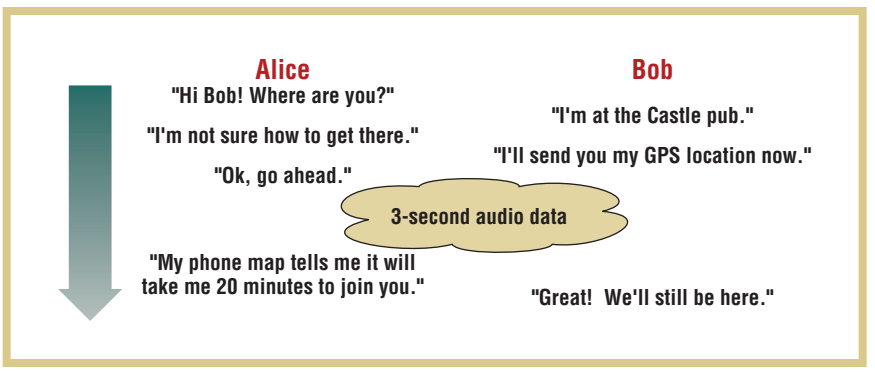


Figure 2. An example of social-flow control scheduling audio data transmission as part of an ongoing mobile phone conversation.

Internet rendezvous

Every user of a mobile phone network has a unique telephone number assigned to them that lets any other phone in the world address them. This wide deployment of phone numbers makes it the most ubiquitous naming system on the planet, with the International Telecommunication Union reporting the existence of over 1.5 billion mobile phone subscribers by mid-2004 (see www.theregister.co.uk/2004/12/10/itu_telecoms_report_2004). On a social level, many tools have emerged to assist with associating users with their telephone numbers, ranging from paper-

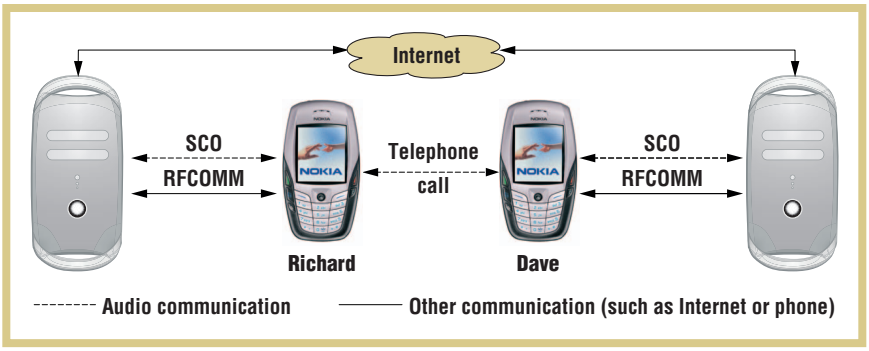


Figure 3. Architecture of smart phone and PC/laptop integration via Bluetooth.

based (but regularly updated) phone books to mobile phones' electronic personal address books.

Many smart phone owners regularly use laptops or desktop PCs in their day-to-day activities. These PCs increasingly have high-bandwidth connectivity to the Internet via broadband or leased-line connections. However, locating other Internet users can still be difficult due to the increasing deployment of Network Address Translation (NAT) devices, firewalls, and other protections. Services such as IM allow online chat but have fragmented naming such that members of competing services can't talk to each other directly without using proxy applications.

We developed the concept of an *Internet rendezvous* service, which leverages audio networking to combine the advantages of the telephone network (global, end-to-end addressing scheme) and the Internet (high-bandwidth multiservice data transmission). Figure 3 illustrates the architecture to perform this integration. We pair smart phones to the users' PCs via Bluetooth, with the PC acting as a headset via the Bluetooth Hands-Free Profile (www.bluetooth.org/spec). Using HFP, the PC can intercept the voice data going to and from the smart phone via an isochronous SCO (Synchronous Connection-Oriented)[6] link and control (for example, dial numbers) using the RFCOMM (the Bluetooth equivalent of a serial connection, complete with flow-control emulation) control connection. In our prototype software implementation, we perform the signal processing of audio on the PC rather than on the smart phone. The software, called Bluegate, runs on Linux with the open-source Bluez Bluetooth stack (www.bluez.org). It performs the appropriate pairing with the smart phone and continuously scans voice packets for audio data packets encoded using the fast DTMF method described earlier. It can also encode fast DTMF tones for transmission to the smart phone.

Although this system works well as a research prototype, many PC-based Bluetooth stacks do not yet reliably support SCO channels. We expect that this situation will improve with time.

Enhanced telephone conferencing

Most modern mobile phone networks support telephone conferences. Teleconference participants often find it difficult to converse naturally because of the lack of social cues and body language found in face-to-face conversation. We implemented a freely available telephone-conferencing application (download this at http://anil.recoil.org/projects/telephony.html), which demonstrates the Internet rendezvous capability of audio networking. We assume that participants in the teleconference also have access to a high-bandwidth desktop PC (such as at work) or a laptop (such as at a Wi-Fi hotspot). Participants without access to such equipment can still take part in the teleconference using their mobile phone, but they won't have access to the extra information provided by our telephone-conferencing software.

Our application supplies two main features that conventional telephone conferences lack: a list of all participants and a real-time display of who's currently talking. (User studies showed that providing these two features significantly improved participants' experience of telephone conferencing.[8]) The only channel the participants share is the actual audio connection via the telephone network.

Suppose that Richard is the moderator of a telephone conference. Once Anil and Dave have dialed into the audio conference bridge (in the usual manner), Richard clicks on his PC's Create Conference button. A brief one-second burst of audio data is broadcast to the telephone conference. This audio data encodes the IP address of an Internet-messaging service. Anil and Dave's PCs intercept this audio transmission, and their PC-based conferencing software initiates a connection to the encoded IP address (see figure 4 for an example of how the screen now looks). When Anil speaks, his PC sends a message to the Internet service indicating this, and the listening-conferencing applications (on Richard and Dave's computers) update their screen to reflect the new status. A history window displays recent speaker information, giving participants context about the flow of conversation over the conference.

We use audio networking only to automatically establish a common Internet server. After that, all communication between the software happens over the network, minimizing the disruption to conversation caused by audio packets. Most modern teleconference bridges emit an audible sound when a participant joins or leaves—we simply augment this existing sound to encode data as well. Our current implementation of the Internet-messaging services uses the IRC (Internet Relay Chat) protocol, with a new channel created per telephone conference. You could use other IM protocols such as AIM (www.aim.com), Jabber (jabber.org), or SILC (Secure Internet Live Conferencing, s.ilcnet.org) to equally good effect.

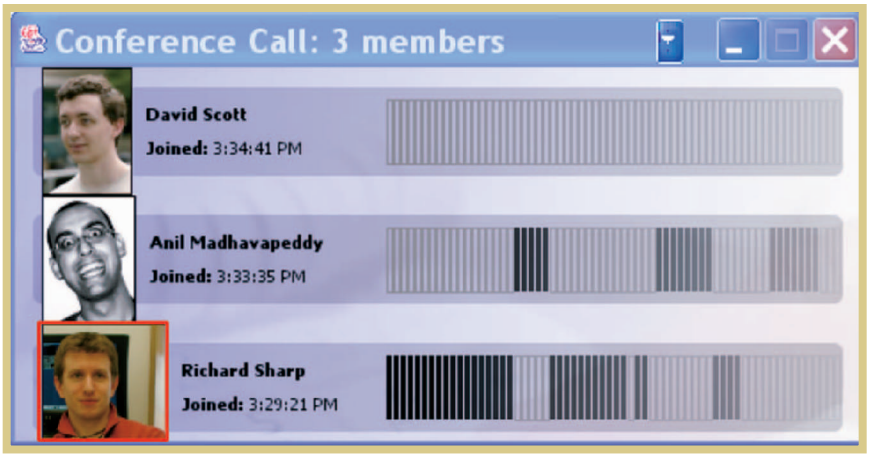


Figure 4. Screenshot of a telephone-conferencing prototype during a conference. Richard is currently talking (indicated by the red square around his face). Anil has recently interjected a few words, and David is simply listening in.

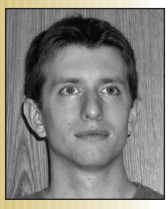
Smart phones currently rely on complex, power-hungry protocols such as Bluetooth to solve user requirements for the spontaneous transfer of small amounts of data. We've demonstrated that you can use audio to deliver many of the same benefits with a dramatic increase in simplicity and ease of use.

People already make simultaneous use of telephone calls and Internet data transfers on an ad hoc basis—for example, sharing a URL by reading it out over the phone. By automatically inserting data into telephone conversations, we hide URLs (and other control data required to bootstrap Internet connections) from users. We see this as a conversational analog to the way hypertext hides URLs from readers, delivering similar benefits. Our telephone-conferencing application goes further and broadcasts information to provide participants with extra visual cues and context about the call.

We're currently porting our research prototype implementations to an application that you can install on Symbian Series 60 phones (for example, Nokia 6600 and 7610) and use on any mobile phone network. We plan to solicit feedback and opinions from the large installed user base of smart phone users to continue to develop the concepts presented in this article.

the AUTHORS

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.



Richard Sharp is a senior researcher at Intel Research. His research interests include ubiquitous computing, security, programming language design, and compiler implementation. His recent research projects include 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.

David Scott is a member of the technical staff at Fraser Research. His research interests include computer security, ubiquitous computing, and networking. Previously at the AT&T Laboratories in Cambridge, he built part of a high-performance CORBA middleware platform. 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.

Alastair Tse is a PhD student at the University of Cambridge Computer Laboratory in the Digital Technology Group. His research interests include wireless network protocols, location systems, and ubiquitous computing. He received his BEng in computer engineering from the University of New South Wales, Sydney, Australia. Contact him at the Digital Technology Group, Univ. of Cambridge, 15 JJ Thomson Ave., Cambridge, CB3 0FD, UK; acnt2@cam.ac.uk.

REFERENCES

1. *IEEE Pervasive Computing*, special issue on the smart phone, vol. 4, no. 2, 2005.
2. A. Madhavapeddy, D. Scott, and R. Sharp, "Context Aware Computing With Sound," *Proc. 5th Int'l Conf. Ubiquitous Computing* (UbiComp 03), LNCS 2864, Springer-Verlag, 2003, pp. 315–332.
3. *Recommendation Q.23: Technical Features of Push Button Telephone Sets*, Int'l Telecommunications Union, 1989.
4. V. Gerasimov and W. Bender, "Things That Talk: Using Sound for Device-to-Device and Device-to-Human Communication," *IBM Systems J.*, vol. 39, nos. 3–4, 2000, pp. 530–546.
5. G. Borriello et al., "Walrus: Wireless Acoustic Location with Room-Level Resolution Using Ultrasound," *Proc. 3rd Int'l Conf. Mobile Systems, Applications, and Services* (MobiSys 05), ACM Press, 2005, pp. 191–203.
6. D. Gratton, *Bluetooth Profiles*, Prentice Hall PTR, 2003.
7. D. Garlan et al., "Project Aura: Toward Distraction-Free Pervasive Computing," *IEEE Pervasive Computing*, vol. 1, no. 2, 2002, pp. 22–31.
8. R. Colburn et al., *Graphical Enhancements for Voice Only Conference Calls*, tech. report MSR-TR-2001-95, Microsoft Research, 2001.