Accepted for publication at the Spontaneity Workshop of Ubicomp 2002

Automated Selection of the *Active Device* in Interactive Multi-Device Smart Spaces

Khomkrit Kaowthumrong, John Lebsack, Richard Han
University of Colorado at Boulder
Contact: rhan@cs.colorado.edu

ABSTRACT

Pervasive computing offers the vision of seamless interaction by users with pervasive smart spaces filled with many wireless and wired devices, including monitors, speakers, printers, kiosks, soda machines, appliances, toys, sensors, and networked software services. To achieve seamless interaction, we must first address what we term as the *active device resolution* problem: when a user enters a smart room filled with *N* devices, which of the *N* devices is the "active device" that the user intends to interact with immediately? This paper focuses on the particular interaction mode introduced by remote control PDA's and cell phones. Manual selection of the intended device, either by pointing-and-clicking, gesturing or selecting a device from a software menu, is either imprecise or decidedly intrusive. Automated selection of the active device, using context clues and user history, is a step towards the vision of more seamless interaction. In this paper, we study the historical behavior of a small sample of users and apply various prediction algorithms to user history to automatically select the next active device that a user intends to interact with. We show that the accuracy of prediction can vary depending upon the algorithm, e.g. up to 90% using 3rd order Markov prediction, and upon the length of training.

Keywords

active device resolution, smart space, remote control, wireless, PDA, service discovery, middleware.

1 INTRODUCTION

Ubiquitous computing offers the vision of a physical environment that is fundamentally more responsive to people. Users will be able to benefit from the "smartness" of rooms and public spaces to seamlessly interact with a wide variety of internetworked wireless and wired devices, e.g. printers, monitors, speakers, wearable computers, appliances, kiosks, toys, sensors, other wireless personal digital assistants and other video-enabled mobile phones [Weiser]. Emerging technologies such as wireless pico-cell Bluetooth networks, wireless Ethernet, third generation high speed cellular systems, low power high speed microprocessors, and matchbox-sized gigabyte disks are already hastening the arrival of a world in which smart spaces are truly pervasive.

One of the first tasks faced by a user who enters a smart room populated with many devices, e.g. a smart living room filled with multiple components of a home entertainment system, is to determine which of these devices is the active device that the user wants to interact with first. We call this the *active device resolution* problem, as shown in Figure 1. This paper focuses on the variant of active device resolution in which the user knows the active device but the smart room does not and therefore must determine the active device. Typical modes of interaction for a user to manually indicate the active device include pointing and clicking a device-specific remote control, speaking the name of the intended device to a voice recognition system, walking up to the device and pushing the appropriate buttons, or even gesturing towards the active device. This paper concerns itself with the common mode of interaction introduced by remote control wireless PDA's and cellular phones, which are already nearly ubiquitous and which we anticipate will soon acquire more general-purpose functions capable of interacting with and remotely controlling other devices in the environment [Bluetooth].

Conventional approaches for manually specifying the active device in a multi-device environment using a handheld wireless PDA or other remote control-like device are cumbersome and far from seamless. The brute force approach of one remote control per device is certainly cumbersome, and often leaves the living room's coffee table strewn with many remotes. So-called "universal" remote controls ideally coalesce multiple remote controls into a single device, but in practice are often impossibly challenging to the typical user to program. X.10 remote controls are a step towards general-purpose wireless PDA's capable of controlling any device, from home entertainment components to light switches to home security systems [X10]. Again, X.10 remote controls suffer in terms of their difficulty to program and inflexible user interface.

While the convenience of coalescing multiple controlling devices into a single handheld wireless PDA or cellphone with a flexible UI is compelling, a single controlling device introduces new challenges in terms of active device resolution. In ubiquitous computing scenarios, this single controlling device is often relied upon to speak one or more service discovery protocols, , e.g. Sun's Jini [Jini], Microsoft's Universal Plug-and-Play [UPNP], Bluetooth's service discovery [Bluetooth], the Service Location Protocol (SLP) [Guttman], and Salutation [Pascoe]. The controlling device discovers the list of devices and services in a smart room, and presents this list or menu to the user for manual selection of the active device. Menu-based manual selection is certainly intrusive, slow and far from seamless. Moreover, if the devices or services are not clearly named, or if there is ambiguity when two similar devices are in the same room, e.g. two lights, printers, or displays, then menu-based selection becomes even more cumbersome.

Directional pointing and clicking of a single controller towards the active device may unintentionally contact several other devices, particularly if the devices share the same manufacturer or same proximity. Orientation support can help reduce *N*, the number of devices in a remote control's field of vision [Priyantha2001], but does not eliminate the problem of multiple services and devices within the range of the remote control. This device ambiguity problem is exacerbated if the wireless medium is omnidirectional RF instead of directional IR, as would be the case for Bluetooth-based communication, allowing even more devices to be contacted. Further, the control command sent by the user may be generic, with no evident destination. If the user with a wireless PDA is presented with a generic UI with generic Volume, Play, and Power buttons, then simply pressing the play button does not identify whether the user wishes to play the DVD or the recorded clip in the Tivo player, while pressing to increase the volume fails to discriminate between the television's volume or the CD player's volume. Pressing "Power on" may affect the room's lighting and climate in addition to the components of the home entertainment system.

Automated selection of the active device in a multi-device environment addresses each of these issues and creates a more seamless interaction with a smart room. By relying on the history of user commands and interaction patterns, a smart infrastructure can better determine the active device that the user intends to communicate with. Generic commands can be routed to the likeliest destination, while device ambiguity can be minimized and the illusion of point-and-click can be maintained. Anticipating the active device can also help determine which user interface (UI) to automatically select and prefetch before the user manipulates their PDA. From the user's perspective, it is far more convenient to expect a smart room to automatically discern the user's intent and thereby resolve the active device to be controlled, than to manually select the active device from a menu.

Figure 1. The active device resolution problem: Given *N* devices capable of being controlled, which is the "active" device that the user intends to interact with? A smart room may use the user's historical behavior to infer the active device.

In the remainder of the paper, we explain the various prediction algorithms that we considered in Section 2, describe our experimental setup in Section 3, and discuss the accuracy statistics obtained by applying the prediction algorithms in Section 4.

2 PREDICTION ALGORITHMS

Our approach is to consider the user's past history of device accesses in order to determine the active device that a user next wishes to communicate with. Among the context clues that we consider relevant are the identity of the user, the time of a device interaction, the action being initiated (e.g. Play, Power, Volume), and the past history of device accesses. For example, for a user who has just entered a smart room, such parameters as the time of day and location are available to help narrow the list of devices to one likeliest candidate device – the predicted active device - available for instant interaction.

In the following, we outline our Markovian approach, though we also considered Bayesian and other standard algorithms to predict the active device based on past user behavior.

Markov analysis observes a sequence of events, and develops a probabilistic model to show the dependency of certain events following other events. Our hypothesis was that human behavior is marked by highly correlated behavior, where one event is often followed by a sequence of dependent events, e.g. nearly every time after a user turns on the TV the same user then turns on the DVD. As a result, we felt that a Markov process would be a useful tool for modeling each user's behavior.

For this analysis we made the assumption, which is necessary for Markov models, that the current state depends only on a finite history of previous states. We found that it was not necessary to apply the more complex Hidden Markov Model since none of the state in our problem domain is hidden. All user choices are entirely visible, in contrast to a domain where some user choices can only be inferred from other evidence, as in the scenario where the observer is hidden in another room and can only infer the user's selections of TV, stereo or DVD by listening to the sound through the wall and trying to guess what device is making those sounds. The Hidden Markov Model would be required to make predictions if the users choices are not visible to the observer and could only be inferred from other evidence.

A Markov process is characterized as follows. The state c_k at time k is one of a finite number in the range {1,..,M}. Under the assumption that the process runs only from time 0 to time n and that the initial and final states are known, the state sequence is then represented by a finite vector $\mathbf{C}=(c_0,...,c_n)$. Let $P(c_k \mid c_0,c_1,...,c_{k-1})$ denote the probability (chance of occurrence) of the state c_k at time k conditioned on all states up to time k-1. The process is a **first-order** Markov if the probability of being in state c_k at time k, given all states up to time k-1, depends **only** on the previous state, i.e. c_{k-1} at time k-1, i.e. $P(c_k \mid c_0,c_1,...,c_{k-1}) = P(c_k \mid c_{k-1})$. For an *n*th-order Markov process, $P(c_k \mid c_0,c_1,...,c_{k-1}) = P(c_k \mid c_{k-n},...,c_k)$.

Figure 3 illustrates how we applied first-order Markov modeling to the access patterns of *N*=4 devices. Each state in the model corresponds to a device access, e.g. TV, DVD, stereo, and alarm. The transition probabilities are constructed from the sample statistical traces we collected of user behavior, i.e. for each given state, we calculated the percentage of accesses found in the trace to the other states. For example, P_{TD} is the sample statistic equal to the number of times the DVD was accessed after the TV. *Given a current state, then a Markovian prediction algorithm would choose as the next state the one that maximizes the state transition probability from the current state.*

While our first order Markov model simply used the previous device accessed to predict the next active device to be accessed, we also studied second and third order Markov models, e.g. what is the likelihood that the next access will be the TV given that the past two access were alarm and DVD? Since these models were constructed on a per-user basis, then what we term as our first order Markov is technically a second-order Markov conditioned both on the user identity and the previous user action.

3 IMPLEMENTATION

In order to collect samples of user behavior and thereby establish a user history upon which prediction of the next

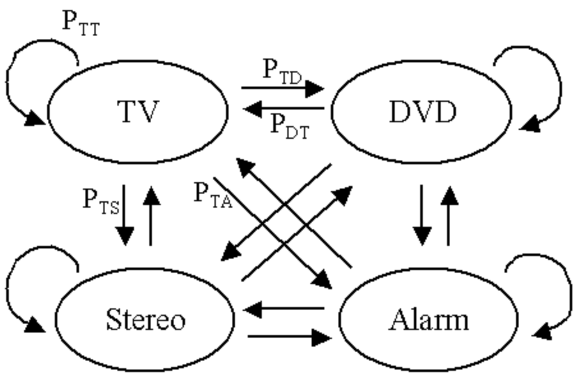

Figure 2. Per-user first order Markov with four states, corresponding to four remotely controlled devices. The arrows represent state transition probabilities.

active device could be performed, we designed and implemented a software control system consisting of a wireless PDA and wired Web infrastructure. The UI to control a device such as a VCR is shown in Figure 3 and was downloaded as a Web interface onto the wireless handheld PDA. When the user pressed a button such as Play, this command would be relayed over the wireless link to an awaiting Web server, capable of translating the command into an X10 command for communication with the VCR. At the same time, the Web server would also log the user action.

3.1 Interface Design

The layout of the interface is critical in the effectiveness of the console. The layout is divided into three main regions, the shortcut bar, the prediction bar and the actual device console. Figure 3 shows the consistent feel of all the buttons in the interface.

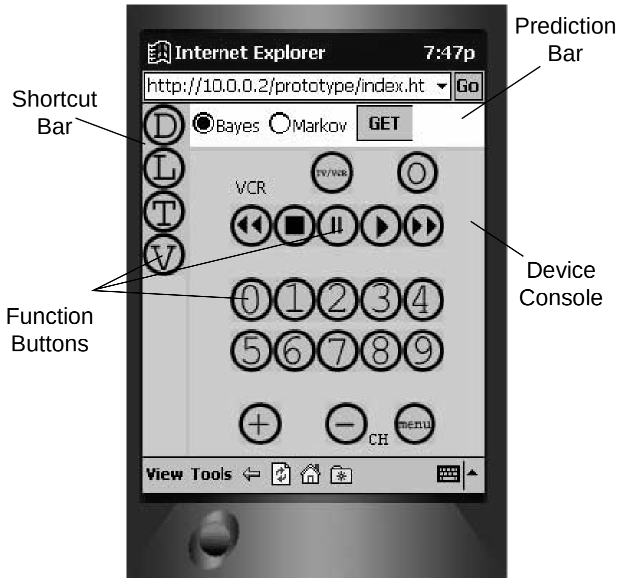

Figure 3. Example VCR interface

The shortcut bar acts as way for users to manually select the interface in the device console window. The icons in the bar are currently letters that signify the device that can be controlled. The system logs all such device selections, to allow for training of the system. The GET button provides a mechanism to maintain the illusion of point-and-click, downloading the UI of the predicted active device after the GET button is clicked. During this type of automated selection, the shortcuts bar provides a fallback option and a valuable feedback mechanism in case the predicted device is not the actual device desired by the user. During our initial collection of data, we did not enable the GET feature.

3.2 System Architecture

Figure 4 illustrates the following components of the system architecture: Client – Ipaq 3650 with Wi-Fi PCMCIA Card; Web Server – Hosts the consoles and the basic AI algorithms; X10 translator module – Translates commands received via TCP socket to X10 and transmits them over the power line; X10 enabled devices – devices capable of responding to X10 signals.

The client uses the built-in Web browser to access the main console of the whole system. All basic command functions that the remote client is directed at the Web server, through the use of PERL web pages viewed on the client as plain HTML forms. No processing is done on the client, except for HTML parsing. The Web server provides the Web interface and the intelligence for console determination algorithms. The Web server also acts as an intermediary for the client in directing and executing remote commands on the applications. The server uses PERL scripts to open a TCP connection with the remote application and issues the command via TCP sockets. Once the translator module receives the command via the TCP socket the commands are translated into the corresponding X10 command and transmitted over the power line to the appropriate device.[X10]

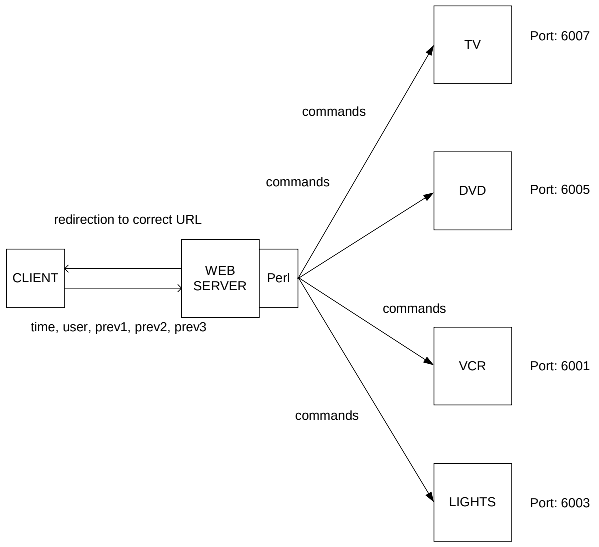

Figure 4. The system architecture of the prototype. This diagram depicts the connections made between the various modules of the system.

The machine learning aspect of the console is coded as a black box that can be interchanged easily. The current algorithms implemented are Naïve Bayes and Markov. These algorithms were the easiest to implement and provide real-time computation of the next possible console. Both are written in PERL, with Markov having also a C++ counterpart. These modules can be replaced simply by changing the function call.

4 EXPERIMENTAL RESULTS

Traces were collected over the course of five weeks in two homes. We deployed the system of Figure 4 into the homes of both User A and User B and automatically logged their device selections. User events recorded the following information on the log: day of the week, time of day, device being controlled, and the action (on/off). The logs also showed the identity of each user.

In our prediction tests, we chose the first N samples of each five week trace as a training set, and then measured the accuracy of the algorithm on the first N samples to the actual device access record. Our initial findings suggest that prediction is feasible, and can achieve an accuracy of up to 85% depending on the length of the training set and the algorithm used. However, before we begin with the analysis of any of the algorithms presented in this paper, we need a baseline for comparison. Figure 5 depicts the distribution of device accesses for User B. The most common device chosen in all the training sets was the TV. Hence, a useful baseline algorithm to compare against would be a most-common device (MCD) algorithm that chose the device with the largest percentage of the distribution. In our study, an MCD algorithm that picked the TV for every test sample would be right 75 percent of the time for User B. Hence, any of our other prediction algorithms considered here should at least perform better than 75 percent correct for User B.

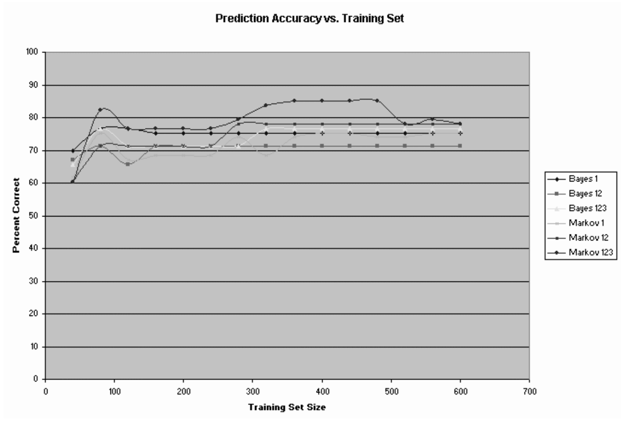

Figure 6. Predicting the next active device from User A (Markov and Bayes).

Figure 6 illustrates the prediction accuracies of both the Markov and Bayes algorithms applied to User A's trace. Both result in roughly the same prediction accuracies for User A, roughly between 70-80% for User A. However, the Markov 3rd order prediction algorithm, that is looking back three events in time, performed the best, achieving prediction accuracies of 85%. The intuition is that, as the algorithm is given more information, the algorithm is able to form a better prediction of the next active device. This is not strictly monotonic but more statistical in nature, since the accuracy can slightly decrease. Our Markovian implementation has the added advantage of real-time operation. For User B's trace in Figure 7, we see that Markovian prediction is able to achieve close to 90% accuracy, besting the baseline MCD algorithm by over 15 %. As a result, a sophisticated machine learning algorithm applied to predict the next active device does indeed bring a benefit in outperforming simpler heuristic approaches.

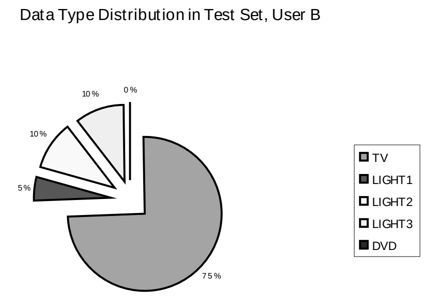

Figure 5. Distribution of User B's device accesses.

The shapes of the graphs also reveal that, as more training is performed, the accuracy of both Markovian and Bayesian techniques increases. Moreover, the techniques appear to plateau, after which time further training is not helpful.

5 RELATED WORK

Prior work on remote control of applications via a PDA has focused on controlling a video conferencing application [Hodes99a] as well as providing an XML framework for controlling lights and stereo components [Hodes99b]. In CMU's Pebbles project, a PDA is used to control a single PC's screen and various PC applications, including PowerPoint as well as a Web browser [Myers98, Myers2000]. Cellular phones have been used as remote controls to purchase sodas from vending machines in Nordic countries using the Jalda payment standard developed [Jalda]. The challenges introduced by more general ubiquitous computing environments populated with *N* devices are only beginning to be addressed [Hodes99b, Han2000]. Stanford's multibrowsing project [Johnson2001] investigates user interaction with multiple output displays. Similarly, multi-device user interfaces have been proposed in which a user can "pick" an object from a PDA and "drop" it on a PC screen or digital whiteboard using middleware [Rekimoto98]. Other work has focused on partitioning of a dual-device user interface between a wireless PDA remote control and a single other device, namely an interactive TV [Robertson96]. Early work on intelligent environments includes a study of predicting a user's movement within a smart home using neural networks [Mozer98]. This study focused on triggering lighting based on the user's location and past pattern of behavior in turning on and off lights. Our study extends this work to focus on predicting more general kinds of user activity. Georgia Tech has built an "Aware Home" to assist those who may be impaired with Alzheimer's [Fox2001]. Microsoft's EasyLiving project has created an intelligent living room implements a follow-me application that tracks user motion using stereo cameras in order to keep a user interface in front of a user for easy access [EasyLiving].

6 SUMMARY

In this paper, we addressed the active device resolution problem encountered in multi-device ubiquitous computing environments. We collected real world traces of user behavior and applied several predictive algorithms, including Markov. Our initial findings suggest that prediction of the next active is feasible, and can achieve an accuracy of 70-90% depending on the length of the training set and the algorithm used.

7 REFERENCES

[Bluetooth] Bluetooth Service Discovery, http://www.bluetooth.com.

[Brumitt2000] B. Brumitt, B. Meyers, J. Krumm, A. Kern, S. Shafer, "EasyLiving: Technologies For Intelligent Environments," Handheld and Ubiquitous Computing, September 2000. www.research.microsoft.com/easyliving.

[Chan98] J. Chan, S. Zhou, A. Seneviratne, "A QoS Adaptive Mobility Prediction Scheme For Wireless Networks," IEEE GlobeCom, 1998.

[Fox2001] C. Fox, "Genarians: the Pioneers of Pervasive Computing Aren't Getting Any Younger," *Wired*, November. 2001, pp. 187-193.

[Guttman] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2. Request for Comments 2608, Internet Engineering Task Force, June 1999.

[Han2000b] R. Han, V. Perret, M. Naghshineh, "WebSplitter: A Unified XML Framework For Multi-Device Collaborative Web Browsing", *ACM Conference on Computer Supported Cooperative Work (CSCW)*, Dec. 2000, pp. 221-230.

[Hodes99a] T. Hodes, R.Katz, "A Document-based Framework for Internet Application Control," *2nd USENIX Symposium on Internet Technologies and Systems (USITS)*, 1999, pp. 59-70.

[Hodes99b] T. Hodes, M. Newman, S. McCanne, R. Katz, J. Landay, "Shared Remote Control of a Video Conferencing Application: Motivation, Design, and Implementation," *SPIE Multimedia Computing and Networking (MMCN), Proc. SPIE*, vol. 3654, 1998 (conf. held Jan 1999), pp. 17-28.

[Jalda] Jalda Payment Standard for the Fixed and Mobile Internet, http://jalda.com.

[Jini] Sun's Jini architecture, http://www.jini.org

[Johnson2000] B. Johanson, S. R. Ponnekanti, C. Sengupta, A.Fox., Technical Note in *UBICOMP 2001*, Sept. 2001.

[Mozer98] M. Mozer, "The Neural Network House: An Environment that Adapts to its Inhabitants", *Intelligent Environments, Papers from the AAAI Spring Symposium*, March 23-25, 1998, Technical Report SS-98-02, AAAI Press.

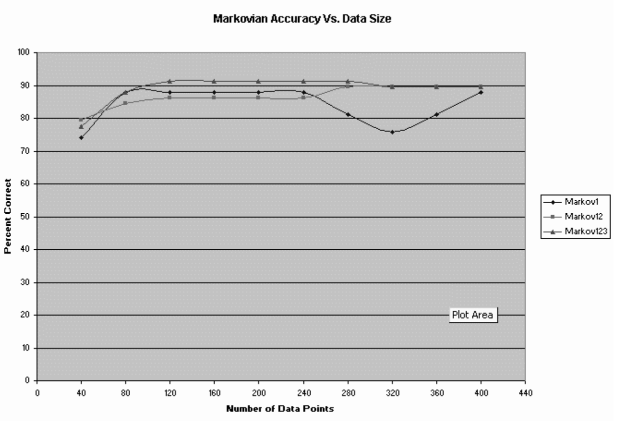

Figure 7. Predicting the next active device from User B (Markov only).

[Myers98] B. Myers, H. Stiel, and R. Gargiulo. "Collaboration Using Multiple PDAs Connected to a PC." *Proceedings CSCW'98: ACM Conference on Computer-Supported Cooperative Work*, November 14-18, 1998, Seattle, WA. pp. 285-294. See also *http://www.cs.cmu.edu/~pebbles/*

[Myers2000] B. Myers, "Using Multiple Devices Simultaneously for Display and Control," *IEEE Personal Communications Magazine*, October 2000, pp. 62-65.

[Pascoe] R. Pascoe, "Building Networks on the Fly," *IEEE Spectrum*, March 2001, pp. 61-65.

[Priyantha2001] N. Priyantha, A. Miu, H. Balakrishnan, S. Teller, "The Cricket Compass for Context-Aware Applications", *ACM MobiCom*, 2001.

[Rekimoto98] J. Rekimoto, "A Multiple Device Approach for Supporting Whiteboard-based Interactions," *Human Factors in Computing Systems (CHI)* 1998, pp.344-351.

[Robertson96] S. Roobertson, C. Wharton, C. Ashworth, M. Franzke, "Dual Device User Interface Design: PDA's and Interactive Television," *SIG CHI* 1996, pp. 79-86.

[Stanfill86] C. Stanfill, and D. Waltz, "Toward Memory-Based Reasoning," Communications of the ACM 29(12), 1986, , pp. 1213-1228.

[UPNP] Universal Plug and Play Service Discovery, http://www.upnp.org/

[X10] X10 Remote Control, http://www.x10.com

[Weiser] M. Weiser, "Some Computer Science Problems in Ubiquitous Computing," *Communications of the ACM*, July 1993, pp. 75-83. (reprinted as "Ubiquitous Computing". Nikkei Electronics; December 6, 1993; pp. 137-143.).