Wearable Computing Meets Ubiquitous Computing: Reaping the best of both worlds

Bradley J. Rhodes, Nelson Minar and Josh Weaver
MIT Media Lab
20 Ames St., Cambridge MA 02139
{rhodes | nelson | joshw}@media.mit.edu

Abstract

This paper describes what we see as fundamental difficulties in both the pure ubiquitous computing and pure wearable computing paradigms when applied to context-aware applications. In particular, ubiquitous computing and smart room systems tend to have difficulties with privacy and personalization, while wearable systems have trouble with localized information, localized resource control, and resource management between multiple people. These difficulties are discussed, and a peer-to-peer network of wearable and ubiquitous computing components is proposed as a solution. This solution is demonstrated through several implemented applications.

1 Introduction

Ubiquitous computing and wearable computing have been posed as polar opposites even though they are often applied in very similar applications. Here we first outline the advantages and disadvantages of each and propose that the two perspectives have complementary problems. We then attempt to demonstrate that the failing of both ubiquitous and wearable computing can be alleviated by the development of systems that properly mix the two. This concept is demonstrated by the description of several applications that have been developed using Hive, a distributed agent architecture that supports peer-to-peer messaging for creating decentralized systems

1.1 Ubiquitous Computing

When Mark Weiser coined the phrase "ubiquitous computing" in 1988 he envisioned computers embedded in walls, in tabletops, and in everyday objects. In ubiquitous computing, a person might interact with hundreds of computers at a time, each invisibly embedded in the environment and wirelessly communicating with each other [1]. Closely related to the ubiquitous computing vision is the more centralized idea of smart rooms, where a room might contain multiple sensors that keep track of the comings and goings of the people around [2].

Many applications have been demonstrated in ubiquitous computing and smart rooms. Some of these have concentrated on intelligent configuration of an environment based on who is in the room. For example, air conditioners and lights might automatically turn off when no one is in the room, or blinds may open and close depending on natural light levels in the room [3]. Other applications have implemented what is called *proximate selection* interfaces, where objects that are nearby are automatically easier to select. For example, a print command might automatically default to the nearest printer [4]. In a similar vein is the presentation of contextual information, where information or annotations about a particular location or object are automatically displayed to a person when she enters an area. Finally, systems have been created that watch a user's location and actions and store that information in an automatic diary [5].

1.2 Problems with ubiquitous computing

In the purest form of ubiquitous computing, all computation is contained in the environment rather than on the person. This extreme has several problems.

- **Privacy issues**: Probably the most important problem is that ubiquitous computing environments pose serious privacy risks. By watching everything a user does these systems have the potential to leak all our actions, preferences, and locations to others unknown to us, now or in the future. Unfortunately it seems to be a truism that the most useful information is also the most personal. For example, several experiments at Xerox PARC, EuroPARC, and the Olivetti Research Center used active badge systems to support location-based information and collaboration. In these systems, participants wear badges that broadcast their location to sensors positioned in each room [6]. The researchers suggested a combination of both technical and social mechanisms to help address this problem. However, as Foner points out [7], sometimes good security and a strong corporate privacy policy is not enough to protect a person's privacy. Central databases are a prime target for subpoena, and the more places that sensitive information resides the more potential places there are to compromise that information. Finally, there are always situations where someone should not trust the environment to keep her information secure. One case is in the customer-provider relationship, where we already have seen a large interest in logging customer profiles and buying habits to increase sales. Another case is when entering a hostile environment. For example, if a businessman is entering a competitors company to negotiate a contract, he probably would not like all his personal profile information to automatically be uploaded to their system where it might be viewed to gain an unfair advantage.

- **Difficulty with personalized information**: The second problem is that it is often difficult to maintain personalization of ubiquitous computing system. In the worst case, every time a new person joins a work-group or community her personal profile needs to be added to every device. Even if all the devices and environments on a campus share a personal profile database, profiles need to be updated every time she moves to a new site.

1.3 Wearable computing to the rescue (?)

The wearable perspective suggests that instead of putting sensors and cameras in the room, put them on the person. In the purest form, the wearable user would do all detection and sensing on her body, requiring no environmental infrastructure at all.

Wearables offer a solution to most of the problems mentioned above. Because the wearable always travels with the wearer, personal profiles never need to be transfered to a new environment. And because a wearable might stay with a user for many years her profile can automatically evolve over time. Furthermore, wearable computers are an inherently more private location for sensitive data because it never needs to leave the person. If the data is never transmitted it becomes much harder to intercept, and the distribution of personal profiles across several wearables (possibly owned by many entities, each with a vested interest in keeping his own data private) makes them a less convenient target for compromise, subpoena, or strong-arm tactics.

Of course, one might still infer a person's location by the fact that a room's resources are being controlled by a particular network address. Traffic analysis is a serious threat to privacy, and the TCP/IP protocols have no features for anonymity. At the application level, a possible solution is bunching together many requests in a scheme such as Crowds [8]. At the network level, Chaum Mixes such as the Onion Routing system [9] show promise. Of course, the physicality of wearable applications may make traffic analysis unnecessary if an eavesdropper is at the end resource itself. For example, if my office light receives a request to light up, it is not hard to surmise that I am at work no matter how anonymous the request was. However, since this requires an eavesdropper at the end resource (i.e. at my office light) there is still no central point of attack. Finally, sometimes privacy leaks are inherent in the application itself. For example, an application that shows where a person is on a map has no choice but to reveal that information; that's its job. Our goal is not to maintain total privacy, but rather to design a system whereby personal data is distributed on a need-to-know basis.

Many wearable systems have been demonstrated that act very similar to smart rooms and ubiquitous computing. For example, wearables have been used to create proximate selection interfaces for room control [10] as well as personalized room controllers for the disabled [11]. Wearable systems have also been used to help create automatic diaries of a user's state over time, both for health monitoring applications and video diaries [12][13]. Finally, many applications exist that present context-based information such as tour-guides [14][15] and general notes related to a user's context [16] [17]. In these systems location is sensed on the wearable either by GPS (for outdoors) or indoors by location beacons. The location beacons are essentially the same as the active badges used in ubiquitous computing, except that instead of being worn by a person to broadcast his identity they are placed in rooms to broadcast locations [18][10]. Similar systems have used paper labels that are recognized through machine vision to recognize a location or object [19]

Table 1: Features provided by Ubicomp vs. Wearables

Feature	Ubicomp	Wearables
Privacy		X
Personalization		X
Localized information	X	
Localized control	X	
Resource Management	X	

[20], while still others recognize objects or locations without any tagging or infrastructure at all [21][22].

1.4 Problems with wearable computing

Wearable systems are well suited to providing privacy and personalizations, but they tend to lack in other areas:

- **Localized information**: Just as smart rooms are ill-suited for personalized information, wearable computer systems have trouble maintaining localized information. For example, if information about a single location gets updated then every wearable needs to be given the new information. Furthermore, is it often difficult for a wearable system to sense information beyond the user's local area.

- **Localized control**: If a wearable is used to control a resource off the persons body, such as a stereo, big screen display, or air conditioner, it is often much easier to design the system with the resource-specific drivers in the device itself. When low-level control is left to the wearable it tends to produce higher demands on the wearable's CPU and wireless network and necessitates that the wearable have code to control each kind of device that might be discovered.

- **Resource management**: Wearables are also not well suited to managing resources among several people. When more than one wearable user wants to use the same stereo, for example, often it is desirable to have a more intelligent system than simply allowing the last request to take precedence.

1.5 Having your cake and eating it too

In this paper we argue that by properly combining wearable computing and ubiquitous computing, a system can have the advantages of both. This synthesis is demonstrated by several applications that have been developed using Hive, a distributed agent architecture that links programs running on wearable computers, desktop computers, and "things that think" [23]. While the details of Hive are beyond the scope of this paper, a brief description is in order. For a more thorough discussion, see the citation [24].

2 Hive

Hive is a distributed agents platform, a decentralized system for building applications by networking local resources. The key abstraction in Hive is the software agent: applications are built out of an ecology of multiple agents interacting over the network.

From a programmer's perspective, each Hive agent is a distributed Java object and an execution thread. However, Hive agents also implement the following properties:

- **Agents are autonomous**: Agents can be sent into a system and entrusted to carry out goals without direct human micro-management.

- **Agents are proactive**: Because agents have their own threads, they can act independent of other running agents. They encapsulate computational activity.

- **Agents are self-describing**: an ontology of agent capabilities can be used to describe and discover available services. Hive agent descriptions consist of both a syntactic description (represented by the Java class of the agent) and a semantic description written in the Resource Description Format (RDF). RDF is in turn encoded in the eXtensible Markup Language (XML).

- **Agents can interact**: Agents can work together to complete a task. Hive agents can communicate both through an asynchronous event / subscriber mode and through Java RMI (Remote Method Invocation). Agent communication is completely peer-to-peer, so an agent might both send and receive at different times.

- **Agents can be mobile**: Agents can move from one physical device to another.

Along with agents, the Hive architecture defines "shadows." Shadows are the low-level drivers for the physical resources of a particular object. For security, only local agents can access a particular shadow.

All remote agents that wish to access local resources must go through local agents. Finally, Hive defines the concept of a "cell" which encapsulates a group of agents. The cell is essentially a Java process that implements an environment within which the agents operate. Generally there will be one cell per local object or wearable computer, though this is not a hard and fast rule. Agents are free to communicate and move between cells. Hive also provides a graphical interface to view and manipulate agent interactions. The interface is itself just one more Hive agent that receives and displays agent events.

2.1 Agent Discovery

In mobile applications such as those described above it is particularly useful to be able to identify what resources are available in a given area, or with a given set of criteria. Hive supports the discovery of new agents through two kinds of lookups. Agents can be queried both based on their syntactic description and on their semantic description. Semantic descriptions include information such as a resource's owner or the room it lives in.

Using this infrastructure, several agents have been created to make resource finding simple for wearable computer users. For example, a "resource finder agent" has been created that receives location events and produces sets of agents whose semantic description matches that location. These agent sets can then be winnowed further by other resource finder agents that are looking for specific services like stereo equipment or large-screen monitors. For more information on agent discovery in Hive, see the cited thesis [25]. To bootstrap the initial list of cells, a resource finder agent contacts a known master cell-listing agent. The creation of new agents are announced to subscribing agents in the form of events, or a special cell-managing agent can be queried.

2.2 Hive with wearables

In our system, each wearable computer is simply another Hive cell in a decentralized Hive network. The peer-to-peer relationships of Hive are a primary difference between our model and the client-server model used by Hodes [10] and by Kortuem [26]. We have found this more symmetric architecture especially useful when implementing applications such as the "Where's Brad?" agent described below, where the wearable user is not the primary recipient or user of information. Sometimes the wearable is the interface to an external service, sometimes the wearable is a service provider for an agent in the environment (or on another wearable), and sometimes the wearable and environment are interacting.

Hive itself is a Java program. For the wearable side, it runs on the Lizzy wearable computers developed by Thad Starner [27], with the Digital Roamabout wireless system for network connectivity. To gather location information, several rooms in the media lab have been equipped with the "locust" location beacons [28]. These beacons are based on a 4MHz PIC 16C84 micro-controller and measure about 1" by 3". The original locusts communicated via infrared, but our current ones have been modified by Alex Loffler of British Telecom to broadcast via RF. This makes it possible to cover an area with fewer beacons, and obviates the need to place them in line-of-site of a wearable. The range varies depending on other equipment in a room, but one or two beacons will usually cover a small room adequately.

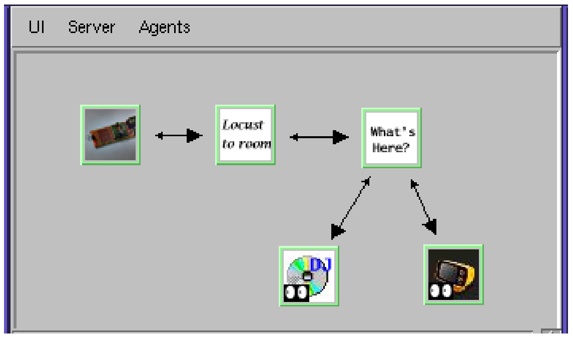

Figure 1: A screen-shot of the Hive user interface, showing connections between several agents. In this configuration, events flow from the locust shadow (far left), to an agent that converts the beacon to a location, to a resource finder agent. Resources for a given location are then passed to a DJ-finder agent and a display-finder agent.

3 Applications

The applications listed below have been implemented and are currently running on the Lizzy wearable and in the lab. The applications range from the useful to the whimsical, and have been chosen to display a range of requirements, including privacy, personalization, localized resources, and scarce resource management. The applications are listed starting with those that emphasize the wearable. The list continues through applications that use both wearable and the environment, and ends with those that primarily emphasize the environment. Next to the name of each application, the salient features demonstrated are listed in parentheses.

3.1 Automatic diary (privacy)

One of the simplest applications is an automatic event logger. As a user walks between rooms, the shadow in charge of receiving RF locust beacons automatically sends the new location to its subscribers on the wearable's Hive cell. One of these agents simply writes the current location to disk. Whenever the wearable user types in a note or idea, that note is automatically timestamped and tagged with the location in which that note took place [16]. Unlike ubiquitous computing automatic diaries (e.g. [5]), the user's location never leaves the wearable.

In this agent, all computation occurs on the wearable. No computation occurs in the environment, except in the transmitting location beacons.

3.2 Proximate selection of a big-screen display (privacy, localized resources)

Sometimes a wearable user wants to project a display on a large screen for others to view. With this application, typing a single chord on the chording keyboard automatically puts the current Emacs buffer onto the nearest screen, where it can be edited by multiple people.

When the user enters a new room a resource finder agent automatically looks for any display agents within the room. If one is found, the display agent is queried for its X-windows screen name, which is then written to a file. The screen name is then used by XEmacs to display the file.

In this application most computation still occurs on the wearable. However, the screens that are available for a given location are maintained in that physical location. If a display name changes or a new one is added, the only information that needs to be updated is in the room itself.

3.3 Theme music (privacy, personalization, localized resources)

One of the original inspirations for this work was that the primary author wanted to have theme music played whenever he entered his office. We have therefore implemented a room automation system in the form of a theme-music agent. Whenever a wearable user enters a room, this agent tries to find a DJ agent that runs on a computer hooked up to a stereo for that room. If it finds one, and if the DJ isn't currently playing music, it sends the URL of a an MP3 file containing the user's personal theme music (for example, the "Inspector Gadget" theme). This music is then played, usually to the annoyance of everyone else in the room.

This is the first agent described that actually performs negotiation with a localized resource. The wearable keeps track of private information (the user's location) and personalized information (the theme music). At the same time, the DJ agent maintains resource information like whether music is already being played. It also maintains final control over the local hardware (the stereo), as will be seen in the next agent description.

3.4 DJ (privacy, personalization, localized resources, resource management)

As is implied by the theme-music agent, people might not want their DJ agent to play music whenever someone enters the room, or only want certain kinds of music played, or want to make sure that no one hogs the DJ with his own taste in music. For this reason, the DJ agent implements resource management policies to insure fairness. In the default case, a DJ takes requests sequentially and plays one request for each agent that has a song. Thus people's requests will be played in a round-robin fashion, one request per person.

It is difficult to do resource management in a completely decentralized, wearable-only system, and so it is extremely convenient to let the DJ itself do the management with the resource it controls. At the same time, the DJ needn't know who is in the room, nor need it keep profiles of favorite songs for people who might visit the room. This information is kept on individual wearables. When a DJ-finder agent finds a DJ agent it automatically sends a play-list of song URLs.

3.5 Remembrance Agent (privacy, personalization, localized information)

An earlier version of an entirely wearable-based remembrance agent (RA) has been described in previous publications [16]. In the old version a user's location and other contextual information was used as a query into her personal note files to proactively bring up information that may be useful in her current environment. This version worked well for making suggestions from personal data because all the database was stored on the wearable itself, where it was easily personalized and kept private. However, for the reasons discussed earlier it was difficult to maintain information about the environment in the database. In the combination ubiquitous / wearable computing system, users can integrate personal and localized data seamlessly. An early version of this combined system was used with an augmented reality system for the Media Lab's 10th anniversary open house in 1995 [19].

As an example, imagine a wearable that acts as a tour guide at a museum. As the visitor moves through various exhibits, extra information about each site is presented on her wearable. Because this information is location specific, it is more easily maintained if it resides in the museum databases and is only sent to a visitor's wearable she enter the exhibit area. However, other information is personal: the fourth-grader would like to know how this exhibit is related to the worksheets he did last week, while the anthropologist would like to know how an exhibit relates to her dissertation. In the new system, resource agents associated with an exhibit provide varying levels of detailed description to the RA. These descriptions can be presented to the visitor as exhibit-specific information, but can also be used as more info for the query into the personal remembrance agent. In this way, the museum can update exhibit information in its database, while visitors can keep personal data locally on their own wearable computers, and both can be presented in a context-sensitive manner.

In this application, information and computation are shared equally between the wearable and the environment. Personal information is still kept private on on the wearable, as is the visitor's location. Localized information is kept in the environment where it can be updated easily.

3.6 Context-aware alert filtering (privacy, personalization, localized information)

The message filtering agent receives alerts, communications, and other information that might interest a wearable user and filters based on the wearer's context and personal profile. For example, if the wearer is at the media lab, messages relating to the lab appear on his or her head-mount display. These messages include requests to play doom death-matches and automatically generated messages announcing food has been left in the lab kitchen. When outside the lab, these messages are not displayed. The agent also filters based on a personal profile. For example, announcements of incoming emails can be filtered to only show personal mail or mail mentioning certain keywords. Negative keyword lists can also be used, for example, to filter out announcements of Indian food in particular.

This application also shares computation and information between the environment and the wearable. Personal profiles are kept on the wearable where they can be updated easily, transported to new environments, and where they can be kept private. Contextual filtering is also done at the wearable so the contextual cues being used (e.g. location) can be kept private. However, triggered information like the arrival of food are processed in the environment since they are outside the sensor range of the wearable.

3.7 "Where's brad?" agent (privacy, personalization, localized resources)

While the agents described so far are services mainly for the wearable user, the architecture makes it easy for the wearable to act as a service provider as well. The "Where's brad?" agent runs on a computer anywhere at the media-lab and produces a map that shows the current location of a wearable user. This agent uses the same resource discovery tools as do the agents on the wearable, except now instead of finding agents associated with a location they find agents associated with a particular person. The amount of disclosure is controlled at the wearable end. For example, the wearable can decide to only allow certain agents to subscribe to location information, or can only reveal certain info to certain people. This privacy policy might not simply be a hierarchy of increasing access. For example, a user might want a policy where coworkers will his location if he is within the workplace, but will only see a "not here" if he is outside. Friends and family may see more detailed information outside of work, but only see an "at work" otherwise.

This agent primarily emphasizes information being displayed in the environment, for people other than the wearable user. However, because a person's location is provided by agents running on his own wearable, he can still maintain fine-grained control over sensitive information.

4 Related Work

While we have our own take on implementation, the applications described in this paper were deliberately chosen to be similar in function to common applications found in ubiquitous and wearable computing. These systems have been described earlier in the paper. In part because of the difficulties cited earlier, others have also designed hybrid wearable / ubiquitous computing systems. Of particular note are Kortuem [26] and Hodes [10] who both describe wearable systems that discover room resources and automatically download interfaces to control them. However, both of these systems follow a client-server model, which makes them less suited for applications where control rests more in the environment and less on the wearable. This work is also related to toolkits for building context-aware applications such as the context-aware toolkit developed at Georgia Tech [29].

The Hive infrastructure is also closely related to several several distributed agent architectures, the closest being Sun's Jini system [30]. Both Hive and Jini are Java-based agent architectures, both rely on RMI distributed objects and mobile code, and both systems represent devices and capabilities on the network, proxying if necessary. However, there are some design differences that make Hive more flexible than Jini for these particular kinds of applications. One is that Jini does not have the conceptual split between shadows (trusted controllers of local resources) and agents (possibly untrusted, possibly mobile code that communicates with shadows and other agents). This split gives Hive a useful abstraction barrier similar to device drivers in an operating system. Hive also has a more location-dependent model of computation. In Hive, an agent's cell can indicate where the agent is physically, what resources it has access to, etc. Jini focuses mostly on services; the actual place a service is hosted on is not a major part of the Jini model. For a more detailed comparison between Hive, Jini, and other agent architectures, see the citation [24].

5 Future Work

The interfaces to almost all the applications described are implicit rather than explicit; the user cannot choose an action other than the default action in a given environment. For many applications, a combination of explicit and implicit interfaces is necessary. For example, the user might need fine-control over a slide projector, or want to specify a different play-list for a DJ agent. Negotiating interface requirements is still an open research issue, especially when different users may have different interface hardware.

Another open research topic is making service discovery scalable. Currently agents are filtered using semantic and syntactic descriptions, but the initial set of Hive cells to query for potentially useful agents is provided by a central server. This works fine for our small-scale system, but will not scale to hundreds of cells in the world. Models include hierarchical structures as used for Domain Name Service (DNS) [31], or location beacons that include a bootstrapping address for downloading service information [10]. However, both methods constrain service discovery along a constrained and pre-determined dimension such as service name or physical location. They do not lend themselves to free-form service discovery along several potentially shifting dimensions.

6 Conclusions

We have presented what we see as fundamental difficulties in both pure ubiquitous and pure wearable computing paradigms. In particular, ubiquitous computing and smart room systems tend to have difficulties with privacy and personalization, while wearable systems have trouble with localized information, localized resource control, and resource management between multiple people. By implementing several applications that span this space of requirements we have tried to show how a peer-to-peer network of wearable and ubiquitous computing components, with proper information flow, can help solve these difficulties.

Acknowledgements

Special thanks go to the rest of the Hive team: Matthew Gray, Oliver Roup, Raffi Krikorian, and to Pattie Maes. Thanks also go to our sponsors at Merrill Lynch and British Telecom, to Thad Starner, Rob Poor, and Alex Loffler for their work on the Locust room beacons, and to Chris Wren for his food-alert system.

This paper is dedicated to the memory of Mark Weiser.

References

[1] M. Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, vol. 36, no. 7, pp. 75–84, July 1993.

[2] A. Pentland, "Smart rooms," *Scientific American*, April 1996.

[3] S. Elrod, G. Hall, R. Costanza, M. Dixon, and J. Des Rivieres, "Responsive office environments," *Communications of the ACM*, vol. 36, no. 7, 1993.

[4] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December 1994, pp. 85–90, IEEE Computer Society.

[5] M. Lamming, P. Brown, K. Carter, M. Eldridge, M. Flynn, and G. Louie, "The design of a human memory prosthesis," *The Computer Journal*, vol. 37, no. 3, pp. 153–163, 1994.

[6] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Trans. on Info. Sys.*, vol. 10, no. 1, pp. 91–102, January 1992.

[7] L. Foner, *Political Artifacts and Personal Privacy: The Yenta Multi-Agent Distributed Matchmaking System*, Ph.D. thesis, MIT Media Laboratory, 1999.

[8] M. Reiter and A. Rubin, "Anonymous web transations with crowds," *Communications of the ACM*, vol. 42, no. 2, pp. 32–48, February 1999.

[9] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, February 1999.

[10] T. Hodes, R. Katz, E. Servan-Schreiber, and L. Rowe, "Composable ad-hoc mobile services for universal interaction," in *Third ACM/IEEE International Conference on Mobile Computing*, Budapest, Hangary, September 1997.

[11] D. Ross and J. Sanford, "Wearable computer as a remote interface for people with disabilities," in *First International Symposium on Wearable Computers*, Cambridge, MA, 1997, pp. 161–162, IEEE Computer Society.

[12] J. Healey and R. Picard, "Startlecam: A cybernetic wearable camera," in *Second International Symposium on Wearable Computers*, Pittsburgh, PA, October 1998, pp. 42–49, IEEE Computer Society.

[13] S. Mann, "Smart clothing: The wearable computer and wearcam," *Personal Technologies*, vol. 1, no. 1, pp. 21–27, 1997.

[14] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: A mobile context-aware tour guide," *ACM Wireless Networks*, pp. 3:421–433, 1997.

[15] S. Feiner, B. MacIntyre, and T. Höllerer, "A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment," in *First International Symposium on Wearable Computers*, Cambridge, MA, October 1997, pp. 74–81, IEEE Computer Society.

[16] B. Rhodes, "The wearable remembrance agent: A system for augmented memory," *Personal Technologies Journal Special Issue on Wearable Computing*, vol. 1, no. 4, pp. 1:218–224, 1997.

[17] J. Pascoe, "Adding generic contextual capabilities to wearable computers," in *Second International Symposium on Wearable Computers*, Pittsburgh, PA, October 1998, pp. 92–99, IEEE Computer Society.

[18] J. Ioannidis, D. Duchamp, and G. Maguire, "Ip-based protocols for mobile internetworking," in *ACM SIGCOMM Symposium on Communications, Architecture, and Protocols.* 1991, pp. 235–245, ACM Press.

[19] Thad Starner, Steve Mann, Bradley Rhodes, Jeffrey Levine, Jennifer Healey, Dana Kirsch, Rosalind W. Picard, and Alex Pentland, "Augmented reality through wearable computing," *Presence*, vol. 6(4), 1997.

[20] J. Rekimoto and K. Nagao, "The world through the computer: Computer augmented interaction with real world environments," in *The 8th Annual ACM Symposium on User Interface Software and Technology.* November 1995, pp. 23–36, ACM Press.

[21] T. Jebara, B. Schiele, N. Oliver, and A. Pentland, "DyPERS: Dynamic Personal Enhanced Reality System," in *1998 Image Understanding Workshop*, Monterrey, CA, November 1998.

[22] T. Starner, B. Schiele, and A. Pentland, "Visual contextual awareness in wearable computing," in *Second International Symposium on Wearable Computers*, Pittsburgh, PA, 1998, pp. 50–57, IEEE Computer Society.

[23] Neil Gershenfeld, *When Things Start to Think*, Henry Holt & Company, 1999, ISBN: 0805058745 http://www.media.mit.edu/physics/publications/books/ba/.

[24] Nelson Minar, Matthew Gray, Oliver Roup, and and Pattie Maes Raffi Krikorian, "Hive: Distributed agents for networking things," Submitted to ASA/MA '99. http://hive.media.mit.edu/, 1999.

[25] M. Gray, "Infrastructure for an intelligent kitchen," M.S. thesis, MIT Media Lab, 20 Ames St, Cambridge MA 02139, May 1999.

[26] G. Kortuem, Z Segall, and M Bauer, "Context-aware, adaptive wearable computers as remote interfaces to 'intelligent' environments," in *Second International Symposium on Wearable Computers*, Pittsburgh, PA, October 1998, pp. 58–65, IEEE Computer Society.

[27] T. Starner, "Lizzy: MIT's wearable computer design 2.0.5," http://wearables.www. media.mit.edu/projects/wearables/lizzy/, 1997.

[28] T. Starner, D. Kircsh, and S. Assefa, "The locust swarm: An enviromentally-powered, networkless location and messaging system," in *First International Symposium on Wearable Computers*, Cambridge, MA, 1997, pp. 169–170, IEEE Computer Society, http://wearables.www.media.mit.edu/projects/wearables/locust/.

[29] Daniel Salber, Anind Dey, and Gregory Abowd, "The context toolkit: Aiding the development of context-enabled applications," in *CHI '99*. May 15-20 1999, ACM Press.

[30] Ken Arnold, Ann Wollrath, Bryan O'Sullivan, Robert Scheifler, and Jim Waldo, *The Jini Specification*, Addison-Wesley, 1999.

[31] P.V. Mockapetris and K. Dunlap, "Development of the domain name system," in *ACM SIGCOMM '88*, August 1988.