The Mobile People Architecture

Guido Appenzeller Kevin Lai Petros Maniatis Mema Roussopoulos
Edward Swierk Xinhua Zhao Mary Baker
{appenz, laik, maniatis, mema, eswierk, zhao, mgbaker}@cs.stanford.edu
http://mosquitonet.stanford.edu

Techical Report: CSL-TR-99-777
January 1999

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305-9040

Abstract

People are the outsiders in the current communications revolution. Computer hosts, pager terminals, and telephones are addressable entities throughout the Internet and telephony systems. Human beings, however, still need application-specific tricks to be identified, like email addresses, telephone numbers, and ICQ IDs. The key challenge today is to find people and communicate with them personally, as opposed to communicating merely with their possibly inaccessible machines—cell phones that are turned off, or PCs on faraway desktops.

We introduce the *Mobile People Architecture*, designed to meet this challenge. The main goal of this effort is to put the *person*, rather than the devices that the person uses, at the endpoints of a communication session. This architecture introduces the concept of *routing between people*. To that effect, we define the *Personal Proxy*, which has a dual role: as a *Tracking Agent*, the proxy maintains the list of devices or applications through which a person is currently accessible; as a *Dispatcher*, the proxy directs communications and uses *Application Drivers* to massage communication bits into a format that the recipient can see immediately. It does all this while protecting the location privacy of the recipient from the message sender. Finally, we substantiate our architecture with ideas about a future prototype that allows the easy integration of new application protocols.

I. Introduction

One of the defining trends of the 1990s has been the explosive growth of the Internet. A growing number of people have Internet access at work, at home, and on the road. Meanwhile, other types of networks, such as cell phones and pager networks, are proliferating rapidly. In the next decade, more and more people will expect ubiquitous network access—the ability to communicate with anyone, anywhere. These trends present us with a number of challenges:

Enabling ubiquitous reachability. Most people will continue to use a variety of network-enabled devices and applications to communicate with others. The notion of a one-size-fits-all communication device is just as misguided as a universal network link or operating system. Basic tradeoffs like weight, speed, and ease of use will not vanish anytime soon; in the meantime, people will use different devices and applications at different times. Our ideal of ubiquitous network access cannot be achieved unless people can be reached regardless of the communication devices or applications they choose to use.

Maintaining location privacy. Enabling ubiquitous network access unfortunately makes privacy issues even more urgent than they are now. A system that keeps track of how a person is reachable and distributes that information without limits could be used to deduce the person's location and compromise his privacy. Ideally, people should be able to receive messages anywhere, without revealing their whereabouts to the entire world.

Thwarting "spam." Receiving unwanted messages is another type of invasion of privacy. Many messaging applications have no way to deliver messages unintrusively. For example, most telephones can either ring or not ring when a call arrives, instead of ringing for some callers and taking a message for others, or ringing during the day and taking a message at night. Users should be able to have all their incoming communications prioritized and filtered on their behalf.

Converting among protocols. Not all application-layer communication protocols can be used by all devices. For example, most phones are not capable of receiving email. Optimally, communications would be converted automatically from the sender's preferred type to the recipient's preferred type.

We have designed the Mobile People Architecture (MPA) to address each of these challenges. In Section II, we describe how MPA fits into the big picture of networking. In Section III, we give an overview of MPA. In Section IV, we describe the design of MPA by giving detailed descriptions of four different usage scenarios. In Section V, we describe the functions of the Personal Proxy, the key component of the MPA system, which tracks the mobile person and handles communications on his behalf. In Section VI, we describe related work, and in Section VII we state our conclusions.

II. The Role of MPA in the Network Layer Model

In this section, we describe how MPA fits into the overall picture of networking and argue that MPA, or something like it, is a logical extension of the current model of networking.

Networking systems are traditionally organized using a layering model composed of Application, Transport/Network, and Link layers (Figure 1). This model is useful in clearly defining the responsibilities and restrictions of software that exists at each level.

For a layer to be fully implemented, it needs a naming scheme, a way to resolve those names, and a way to route communications. The *Name Types* column of Figure 1 shows the naming scheme that Internet email uses at each layer. Some examples of names are shown in the *Packet Headers* column. These naming schemes usually mandate that the names are unique and change infrequently. In addition, each layer in the figure has a protocol to map its names to lower layer names (the *Name Lookup* column in Figure 1). This mapping facilitates routing a communication to its destination.

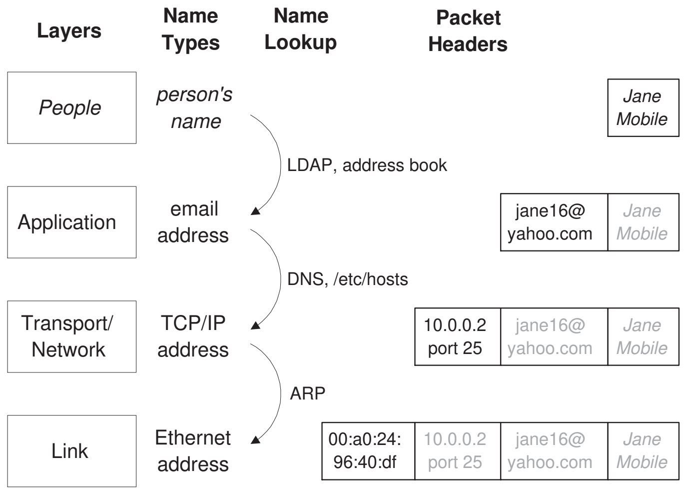

Fig. 1. The Layering Model. We show the traditional networking layers, extended with the *People* layer. *Name Types* shows examples of the kinds of names used at each layer. *Name Lookup* shows some methods of mapping names from each layer to names in the next lower layer. *Packet Headers* shows examples of actual names at each layer, and their relative locations in a typical email packet.

There is one key problem with the traditional layering model: it does not explicitly include people. It seems odd that a communication model would not model people when probably the most important communication is from one person to another.

To model the full process of personal communication, we need to extend the model to include people (the new People layer is shown in Figure 1). Although the layer is new in the model, it is not new in reality. As a result, it is currently implemented in an ad hoc, non-unified way. People are not always named in a unique way, although a name or nickname is often unique among those with whom a person communicates frequently. These names (e.g., Jane Mobile) are resolved into application-specific names (e.g., `jane16@yahoo.com`) using a directory service (e.g., LDAP [WKH97]), an address book, or simply from a person's memory. By directing messages to application-specific addresses, it is the sender who controls their ultimate destination rather than the recipient.

As a result, messaging applications (and therefore their users) have difficulty delivering messages to people who move from one application-specific address to another. For example, if Jane Mobile's email address changes because she travels between home and work, Dan Sender's mail client (and therefore Dan) cannot reliably send email to her. Even worse, if Jane is temporarily unavailable by email, but is reachable by phone, Dan cannot communicate with her until she is available by email. The problem is that Dan cannot identify Jane in a way that is independent of how she is reachable.

The solution is to create a unified implementation for the People layer. Such an implementation needs to name people, map people's names to application-specific addresses, and route communications between people (which we refer to as *people-level routing*). Although the first two functions are partially implemented today, no implementation exists for a people-level router.

The role of a people-level router is similar to that of an IP router: it takes communication from a variety of interfaces and directs it out one or more interfaces, based on the recipient's preferences and on characteristics of the communication itself. The closest current approximation is a human

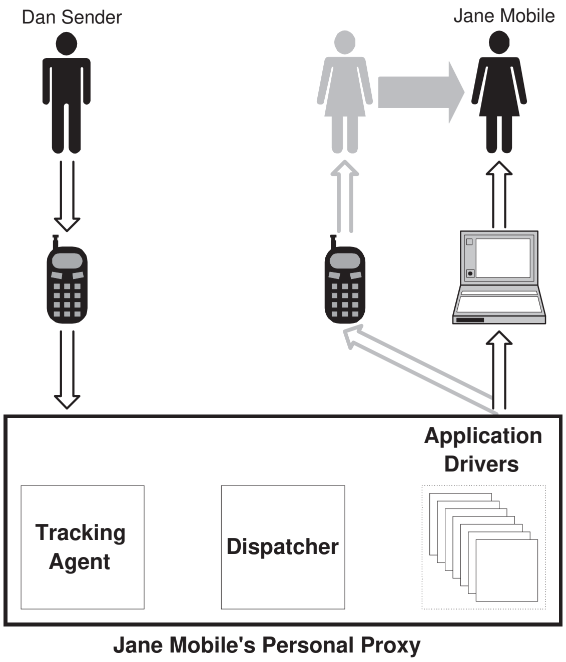

Fig. 2. A Typical Usage Scenario. Dan Sender uses his cell phone to place a call to Jane Mobile. The call is redirected to Jane Mobile's laptop through her Personal Proxy. The gray figure and arrow indicate Jane was recently accessible through her cell phone, but is now accessible through her laptop only.

assistant who answers Jane's phone, reads her email and forwards her messages by calling, emailing, or paging her. Aside from wasting the assistant's time, this implementation would have difficulty forwarding real-time communication (e.g., forwarding an IP telephony call to her cell phone).

The people-level router is a necessary component of any implementation of the People layer. The People layer is a logical extension of the traditional layering model which is the basis of current networking architectures. Therefore, the people-level router is also a logical extension of traditional networking architectures. The MPA implementation of the people-level router is the Personal Proxy, which we describe in greater detail in Section V.

III. Architecture Overview

The main goal of MPA is to route communication to a mobile person, independently of his location or the communication applications he is currently using. This people-level routing uses an addressing scheme that uniquely identifies people. In MPA, these addresses are called Personal Online IDs (POIDs). The architecture does not depend on how POIDs are maintained or how people retrieve the POIDs of other people.

Figure 2 shows a typical usage scenario in which Dan Sender wants to initiate communication with Jane Mobile. If Dan's communication application (which could be anything ranging from email to a fax machine) supports MPA, then it uses Jane's POID to direct communication to her Personal Proxy. If Dan's application is not MPA-aware or if a POID naming scheme is not widely deployed, then an alternate scheme is used (see Section IV).

The Personal Proxy is the heart of MPA and consists of three components: the Tracking Agent, the Dispatcher, and a set of Application Drivers. We briefly describe their functions here, and give more detailed descriptions in Section V.

The Tracking Agent in Jane's Personal Proxy is responsible for keeping track of her as she moves from an application on one device to another application (possibly on another device). For example, in Figure 2, Jane has switched her cell phone off and is now accessible only via email on her laptop. The Tracking Agent makes this information available to the Dispatcher in her Personal Proxy.

The Dispatcher processes any communication that arrives at the Personal Proxy. Using Jane's accessibility information and her preferences, the Dispatcher directs the communication to the appropriate application. In some cases, the Dispatcher may call upon an Application Driver to convert the communication into a form understandable by the receiving application. In Figure 2, Dan Sender calls Jane on her cell phone. Since she is accessible only via email, an Application Driver converts the voice message into an email message with an embedded sound file. This sound file is then forwarded to Jane's laptop. An Application Driver could also enforce user-specified restrictions (e.g., to block spam), or convert intrusive forms of communication into less intrusive ones (e.g., a phone call into voicemail).

IV. Design

In this section we will describe the design of the Mobile People Architecture by outlining in detail four different usage scenarios between Dan Sender and Jane Mobile. In these scenarios, Dan initiates communication with Jane. We assume the existence of a Personal Online ID (POID) system.

A. Scenarios for MPA-aware Applications

In the following two scenarios we assume that all applications are MPA-aware:

Figure 3(a) shows a scenario in which Jane wants privacy; she does not want to reveal her location to anyone. She also wants to receive communication from Dan, regardless of what application he is using. To achieve these goals, Jane has her Personal Proxy receive communication on her behalf and forward it to her. The Personal Proxy acts as an enhanced online analog to the human assistant referred to in Section II. We show below how the Personal Proxy achieves the goals of privacy and application-independent communication.

Dan enters Jane's POID into his communications application. If Dan is going to communicate with Jane, he knows her POID, just like he knows her real name. The application sends a query with Jane's POID to a Directory Service (DS) such as LDAP. Based on the POID and the application type, the Directory Service returns the relevant *Proxy Application-Specific Address* (PASA) of Jane's Personal Proxy (e.g., `jane@janemobile.nom` for email or 555-1000 for telephony). For each type of application that Jane uses, her proxy has a corresponding PASA. This allows the Personal Proxy to intercept and redirect all communication to Jane's applications which are at undisclosed Application-Specific Addresses (ASA). Some examples are `jane16@yahoo.com` or 123-4567.

Dan's application initiates communication with Jane's Personal Proxy at the returned PASA. Her proxy determines which of her applications should receive the communication. If necessary, it also converts the communication into a different format and then forwards it to Jane's application. Note that at no point is Dan or his application aware of the redirection; this ensures Jane's location privacy.

Figure 3(b) shows a scenario in which Jane does not care to conceal her location. Here, the Personal Proxy does not participate in the communication between Dan's and Jane's applications. Instead, the Personal Proxy updates the Directory Service with the ASAs of Jane's currently available applications. In the figure, we refer to this as *Tracking Info.*

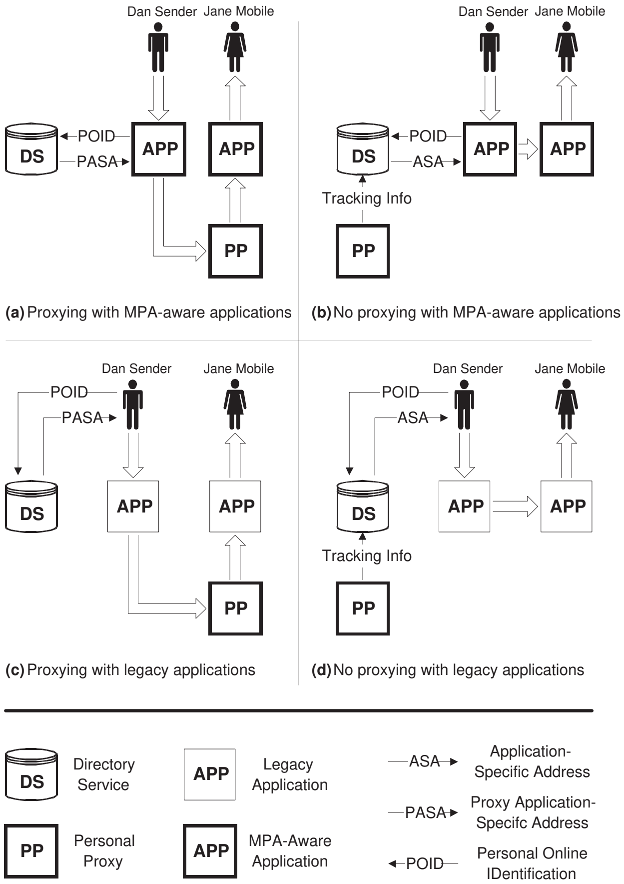

Fig. 3. Four Usage Scenarios of MPA. Wide arrows indicate the transfer of communication data, and thin arrows indicate the transfer of control data.

Dan enters Jane's POID into his communications application. The application sends a query with Jane's POID to a Directory Service (DS) such as LDAP. Based on the POID and the application type, the Directory Service returns Jane's current ASAs.

Dan's application initiates communication directly with Jane's application using the returned ASA. While this scenario is more efficient than the first scenario, it does not offer the same privacy and application-independent communication benefits.

B. Scenarios for Legacy Applications

In the following two scenarios we assume that no applications are MPA-aware. We illustrate that MPA is flexible enough to support legacy applications.

Figure 3(c) shows a scenario in which Jane desires privacy and application-independent communication. Since Dan's application does not recognize POIDs, Dan must manually query the Directory Service to obtain Jane's PASA. Dan feeds the PASA into his application. The application sends the communication using the PASA as a destination address. The communication is routed to the Personal Proxy. As before, the Personal Proxy determines which of Jane's applications should receive the communication. If necessary, it converts the communication and then forwards it to Jane's application using that application's ASA.

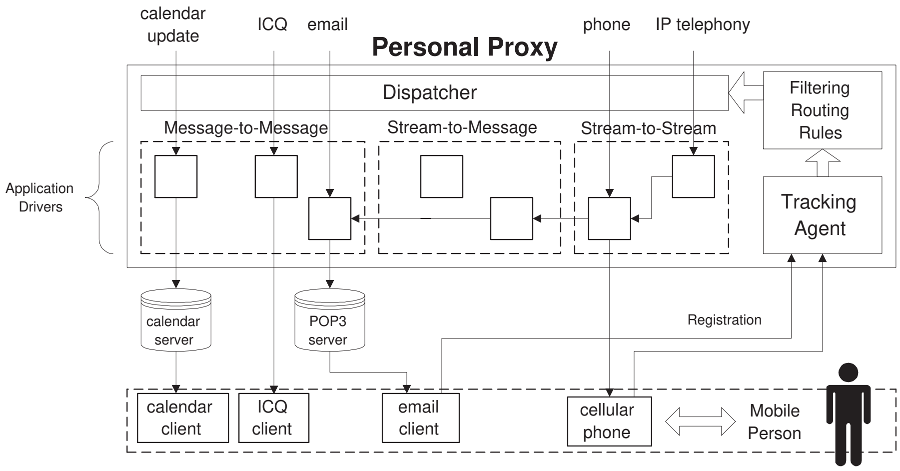

Fig. 4. The Personal Proxy Design. Various communication types are shown entering the Personal Proxy through the top. Arrows indicate possible routes through the Dispatcher and the Application Drivers to the mobile person's applications.

The scenario in which Jane does not care to hide her location from Dan, is similar to the second scenario in the previous subsection. The only difference is that Dan must query the Directory Service to obtain the ASA of Jane's available application. We illustrate the scenario in Figure 3(d).

C. Sender Privacy

In the scenarios above we emphasize the receiving person's preferences for privacy. MPA is flexible enough to support location privacy for the sender as well as the receiver. If a sending person has requested location privacy, all of his communication must travel through his Personal Proxy. If his application is MPA-aware, this is straightforward: he configures the application so that it always sends communication to the PASA of his Personal Proxy. However, if the application is not MPA-aware, we need to perform some kind of application-level encapsulation. That is, the user must incorporate the recipient's POID within the application's data and must set the application's destination address to be the PASA of his Personal Proxy. When the Personal Proxy receives the communication, it must use the receiver's POID to obtain from the directory service the appropriate ASA (or PASA if the receiver has also requested privacy) to use when forwarding the communication.

V. Personal Proxy Design

The Personal Proxy performs a number of key functions in the MPA system: it keeps track of the mobile person's whereabouts; accepts incoming communications on the person's behalf; converts or filters communication data; and delivers communications to the correct Application-Specific Address.

The general design of the Personal Proxy is shown in Figure 4. The Tracking Agent keeps track of the mobile person's whereabouts. Meanwhile, communications arrive through a variety of application-specific protocols, shown at the top of the diagram. The Dispatcher uses the Filtering Routing Rules (derived from the person's preferences) and his current location (obtained from the Tracking Agent) to determine which Application Drivers should be invoked to convert or filter the communication. The Dispatcher then sends the communication to the correct application.

In the following sections we discuss the components of the Personal Proxy in more detail.

A. Tracking Agent

A.1 Tracking

The Tracking Agent keeps track of the applications through which a mobile person is most likely to be accessible at a particular time. The person conveys this information by registering applications with the Tracking Agent. A registration does not guarantee that the person will be accessible through this application; it merely indicates that he is likely to be reached at the registered application.

A.2 Registration

An application can be registered in a variety of ways; the method used depends on the application type and user preferences. The registration can be manual, automatic, polled, or based on some user-specified profile:

- *Manual registration* requires the mobile user to perform some task to indicate that he is likely to be accessible through an application. He might enter his username and password into a secure web page or dial a particular phone number and enter a personal code. The user might provide an estimate of how long he expects to use the application, or might perform another manual task to deregister the application.
- *Automatic registration* relieves the user from any manual task; instead, the application or operating system senses a user's presence and automatically registers with the Tracking Agent. For example, a device might assume that a user is present when he turns on the device, or when it detects that the user's "smart badge" is within range. This is just a hint that the user is present; it is the responsibility of automatic registration mechanisms to maximize the probability that this information is accurate while still being user-friendly. This automatic type of registration requires new software on the device.
- *Polled registration* requires no effort from either the user or the application. The Tracking Agent periodically polls each of a user's applications or devices to detect if the application is running or the device is turned on. For example, polling might be done by pinging the device or by sending a message to the application and waiting for an acknowledgement. Polled registration is not practical for certain devices, such as one-way pagers.
- *User-specified profiles* allow users to specify *a priori* which applications they are likely to be using in the future. A user might have a profile that indicates the days and times he is likely to use each application. The user can modify the profile as often as desired. Although this option does not provide dynamic detection of active applications, as the previous methods do, it is simple to implement and may be the only feasible option for receive-only devices like one-way pagers.

B. Dispatcher

The Dispatcher receives incoming communications and decides whether the data need to be processed through an Application Driver. It bases this decision on preferences set by the recipient (e.g., "Send all ICQ communications to my pager") and information from the Tracking Agent about the recipient's location. If the message needs to be converted, the Dispatcher attempts to find the Application Drivers necessary to convert the communication to the appropriate application-specific protocol.

To make the Personal Proxy easily extensible, the Dispatcher is not able to make decisions based on application-specific properties of the communication. Actions such as "Throw out all emails that contain Java objects" have to be made by Application Drivers, which are described in the next section.

C. Application Drivers

A vital goal of MPA is to allow for the easy integration of new applications. To achieve this goal, we limit application-specific knowledge to small, modular blocks called Application Drivers. All other parts of MPA have to know only what types of application-specific protocols there are, not the details of each protocol.

Ideally, a driver should be able to convert any communication to any other format; however, in practice it is doubtful whether this is practical or desirable. In some situations the best a driver can do is to embed one format within another (e.g., embed a voicemail message as a sound file within an email message). Generally we can distinguish four types of drivers:

- *Stream-to-stream drivers* transform one live stream into another. This is currently most likely to be of interest for voice-based applications. Since these drivers must perform complex operations under stringent real-time constraints, they are the most difficult to design.
- *Stream-to-message drivers* convert a live stream into a message. If a user is contacted with an interactive streaming format but is only reachable by non-interactive message types, the Personal Proxy can use this type of driver. This is similar to the functionality of a voicemail system.
- *Message-to-message drivers* convert one message format to another, a process which need not be performed in real time.
- *Filter drivers* enable one of the key features of MPA: enabling users to direct incoming communications based on application-specific properties, such as keywords.

It is important to note that while the Application Drivers are shown in Figure 4 as being completely contained within the Personal Proxy, a driver can be built using a client-server model. Most of the real work could be accomplished by a server on a different machine; the local driver would be a simple stub that communicates with the server. The Berkeley NINJA Project [GWBC99] demonstrates the potential of such a system (see Section VI).

Message storage, performed in the figure by the *calendar server* and the *POP3 server*, is outside the scope of the Personal Proxy. In our design, we leverage the support for message storage already provided by applications.

VI. Related Work

Several projects and products are related to our work on MPA. This is a very good indication of the growing interest in supporting convenient and instant communication with people on the move. While these other efforts share goals with our project, they do not provide a coherent end-to-end model that integrates people with the communications hierarchy. In our model, *people* are the ultimate endpoints.

The AT&T Easy Reach 500 Service [ATT] and the ever-popular instant messaging schemes, such as ICQ [ICQ] and AOL's Instant Messenger [Ame], clearly reflect people's desire to stay connected. The 500 Service is somewhat primitive. It does not track the owner of the 500 number; instead, it calls a predetermined list of numbers in turn, until somebody answers. The instant messaging services use proprietary naming schemes, thus hindering interoperability.

The GSM cellular telephony system, and the Personal Mobile Telecommunications option of the Japanese cellular telephony system (PDC) support personal mobility within their respective networks by separating the subscriber's identity from the device he uses. However, it is unlikely that people will use only one type of network in the future. Therefore, these systems cannot provide the full personal mobility support that MPA aims to provide.

Mobile IP [Per96] enables a mobile host to be addressed by a well-known, static IP address and to receive communication regardless of its current point of attachment to the Internet. However, it provides only host mobility and only within the Internet.

Iceberg [JBK98] aims at integrating cellular telephony networks with the Internet. It shares with MPA the view that people will continue to use multiple devices for communication. However, Iceberg approaches the problem primarily at the network layer, rather than the people layer. Moreover, it does not provide location privacy. We believe location privacy is a key goal when supporting personal mobility.

A related project is NINJA [GWBC99], which focuses on providing an infrastructure for the construction of flexible and adaptable services in a clustered environment. This infrastructure could provide a solid foundation for new, pluggable Application Drivers in our Personal Proxy.

The Presence Information Protocol [ADM98] (PIP) and the IDentity Infrastructure Protocol [FM98] (IDIP) provide some support for personal online identities and tracking of people. Both allow people to advertise dynamic information about their online presence and to exchange instant messages with each other. IDIP goes a step further, by permitting more specific negotiation of multimedia communication formats. Neither of the two approaches addresses location privacy.

VII. Conclusions

People-centric communication is the next big step for mobile computing. Whereas existing mechanisms have addressed mobility in the network, none has fully addressed the issue of providing mobility support to *people*, who are the ultimate and most important endpoints of communication.

We propose an architecture called the Mobile People Architecture (MPA), which provides support for instant and convenient communication between people, as they move from place to place and make use of multiple heterogeneous communication devices, including laptops, PDAs, or cellular phones. MPA makes it possible for people to protect their location privacy and for application designers to facilitate the deployment of their applications within this framework. We identify the key components within this architecture and their corresponding functionalities. Finally, we illustrate a potential prototype design of the system.

People cannot be the outsiders in the communication landscape any longer. We firmly believe that with the help of the mobile computing research community, this challenge will be met.

VIII. Acknowledgements

We would like to thank Ichiro Okajima, of NTT DoCoMo, for providing us with valuable questions and suggestions during our MPA discussions, and Trevor Holt of Nortel Networks for bringing to our attention related work on PIP and IDIP.

This research has been supported by a gift from NTT Mobile Communications Network, Inc. (NTT DoCoMo), a grant from the Keio Research Institute at SFC, Keio University and the Information-technology Promotion Agency in Japan, and a grant from the Okawa Foundation.

References

[ADM98] S. Aggarwal, M. Day, and G. Mohr. Presence Information Protocol Requirements, August 1998. http://search.ietf.org/internet-drafts/draft-aggarwal-pip-reqts-00.txt.

[Ame] America Online, Inc. AOL Instant Messenger. http://www.aol.com/aim/.

[ATT] AT&T Easy Reach^{SM} 500 Service. http://www.att.com/easyreach500/.

[FM98] S. Fujimoto and D. Marvit. The IDentity Infrastructure Protocol, August 1998. http://search.ietf.org/internet-drafts/draft-fujimoto-idip-00.txt.

[GWBC99] S. D. Gribble, M. Welsh, E. A. Brewer, and D. Culler. The MultiSpace: an Evolutionary Platform for Infrastructure Services. In *Proceedings of the Usenix Annual Technical Conference*, June 1999.

[ICQ] ICQ, Inc. How to Use ICQ. http://www.icq.com/icqtour/quicktour.html.

[JBK98] A. D. Joseph, B. R. Badrinath, and R. H. Katz. The Case for Services over Cascaded Networks. In *Proceedings of WOMMOM '98*, October 1998.

[Per96] Charles Perkins. IP Mobility Support - RFC2000, 1996.

[WKH97] M. Wahl, S. Kille, and T. Howes. Lightweight Directory Access Protocol (v3) - RFC 2251, 1997.