iMobile ME – A Lightweight Mobile Service Platform for Peer-to-Peer Mobile Computing

Yih-Farn Chen, Huale Huang, Bin Wei
AT&T Labs – Research, USA
chen@research.att.com, huale@research.att.com, bw@research.att.com

Ming-Feng Chen
National Chiao-Tung University, Taiwan
mfchen@ieee.org

Herman Rao
Far Eastone Telecommunications Co., Ltd., Taiwan
hrao@fareastone.com.tw

Abstract: As mobile devices become increasingly more powerful in storage, computation power, and communication capabilities, we anticipate an emerging need for a mobile device to access information or services on other mobile devices. A mobile device, however, may still be limited by its physical size, battery power, and intermittent communication capabilities. To facilitate information exchanges among these mobile devices, we propose a lightweight service platform on each mobile device and a network-based, always connected proxy that routes requests and responses among these devices. The lightweight platform adopts the notions of *devlets* and *infolets* in iMobile Standard Edition (SE), a proxy-based mobile service platform, to provide communication and information access interfaces on each mobile device. We call this lightweight platform iMobile Micro Edition (ME). ME devlets allow the local user and remote devices to communicate with the ME dispatcher through various communication protocols. ME infolets provide access to resources available on the mobile device. The ME dispatcher arbitrates communications among the front-end devlets and the backend infolets. To handle intermittent connections and varying bandwidths, each devlet or infolet with remote access is extended with an inbox queue that accumulates incoming messages and an outbox queue that accumulates outgoing messages - until a

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35616-7_23
W. Cellary et al. (eds.), *Internet Technologies, Applications and Societal Impact*

synchronization request with the network-based proxy (iMobile SE) is activated. The iMobile SE proxy synchronizes its message queues with those of mobile devices that attempt to communicate with each other. The collaboration of iMobile ME platforms and an SE proxy provides a lightweight infrastructure that enables new peer-to-peer mobile applications to be developed quickly for various mobile devices.

Key words: mobile computing, peer-to-peer, proxy, middleware

1. INTRODUCTION

Recent advances in hardware and software technologies have created mobile devices with computation, storage, and communication capabilities that rival a typical desktop PC except for its limited size, battery power, and intermittent communications. More over, many resources available to a mobile device may not be readily available on any networked servers. These include its location information, locally captured media files, and its exposure to surrounding resources, such as thermometers or X10 cameras that are wirelessly connected. This prompts us to ask whether the Peer-to-Peer (P2P) computing paradigm, such as the model adopted by the (once) hugely popular Napster service [1], will also enable mobile devices to exchange information with each other directly. Figure 1 shows a scenario with four mobile users located in San Diego, San Antonio, New York, and Paris, France, sharing resources on one another's mobile device. For example, the mobile device user (*chen*) in Paris with an iPAQ and local wireless LAN access (possibly provided by a coffee shop) may want to access a particular image captured by his friend (*wei*), a mobile user with a Palm device on a CDPD network. *Chen* may also be interested in the location of another user, who happens to be in San Antonio, and the address book entry of a Paris friend that he knows the other mobile user in San Diego has. None of these contents are stored on any network-based server. Since none of these mobile users know the IP addresses of other mobile users, they would rely on an always-on, network-based server (iMobile router) to help them locate each other. We consider such peer-to-peer mobile computing a paradigm shift from the traditional client-server computing and it has profound implications for the design of future mobile applications and the network architecture.

This paper describes an approach to creating such an infrastructure for P2P mobile computing by extending the same set of abstractions in our iMobile service platform [2][3][4] to mobile devices. Figure 2 shows a

logical architecture view of the iMobile Standard Edition (SE). iMobile SE is a proxy-based mobile service platform designed to provide personalized mobile services. iMobile provides a modular architecture that supports accesses from various mobile devices to various information spaces. iMobile achieves modularity through three key abstractions shown in Figure 2: *devlets* to interface with devices, *infolets* to access information, and *applets* to implement application logic. The proxy hosts all devlets, infolets, and applets.

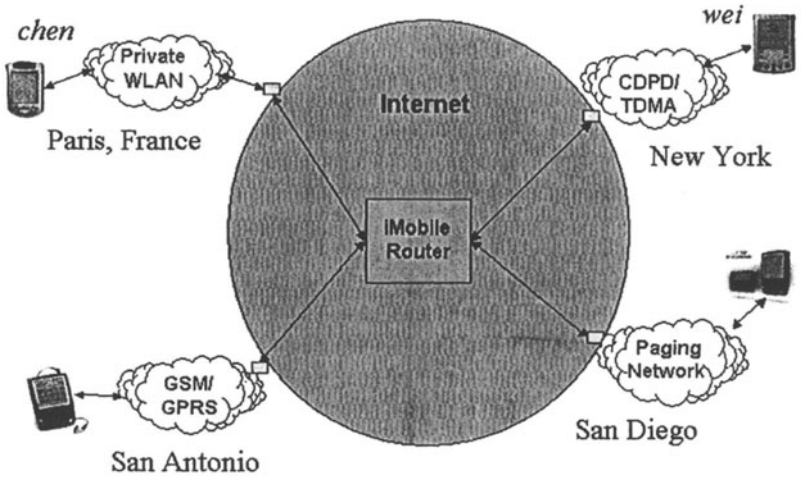

Figure 1. Peer-to-Peer Mobile Computing

Devlets represent protocol adaptors for different devices that communicate with iMobile. Such an adaptor converts protocol-specific messages that a device understands to command strings that the iMobile platform can interpret. It delivers results formulated in a MIME type appropriate for that device. For example, the Instant Messaging (IM) devlet on iMobile is an IM client that listens to service requests from other IM clients. This devlet receives service requests as instant messages and returns responses as instant messages. iMobile supports devlet interfaces to AIM, SMS (short message service), WAP, Email, Telnet, HTTP, etc.

Each service request is sent to a command dispatcher hosted on the proxy. The dispatcher authenticates the users, validates the command, and decides which applets or infolets to invoke to service the request. It also

renders the returned information suitable for the target device by invoking the proper transcoder provided by each infolet based on corresponding user and device profiles.

Infolets are responsible for obtaining information from various data sources or content providers. An infolet creates an abstract view of an information space using an access protocol appropriate for that space. iMobile supports several infolets for accessing corporate databases, email servers, directories, a CORBA[5] infolet to access CORBA services, an X10 [6] infolet to control home network devices, and several infolets based on the HTTP protocol for language translation, location-aware services, etc. Each infolet also hosts a transcoder that translates the raw content (XML or plain text) into various forms.

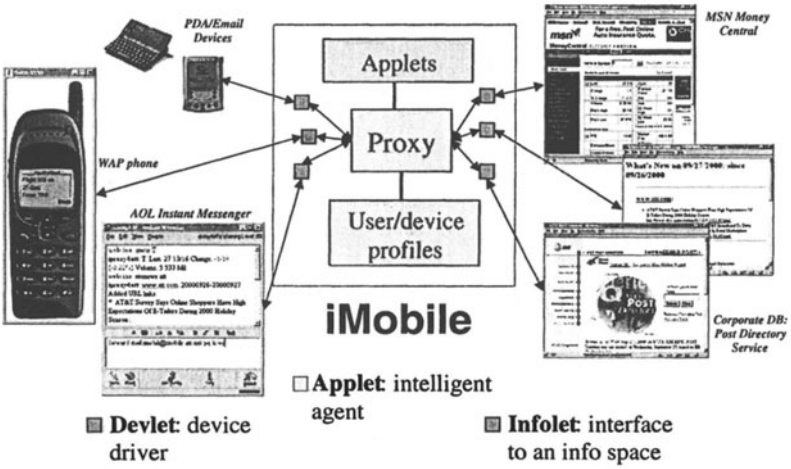

Figure 2. Original iMobile Architecture (Standard Edition)

An iMobile applet implements some application logic by post-processing information obtained by the various infolets. An example applet that delivers local weather report might use both the location infolet and either a text-based or video-based weather infolet.

As mobile devices have become increasingly more powerful, we decided to experiment with placing a scaled-down version of the iMobile SE server on a mobile device and we call it iMobile Micro Edition (ME). We reduce the role of iMobile SE to a simple router that allows mobile devices running iMobile ME to locate and exchange resources with each other. This paper describes the architecture of iMobile ME and how it collaborates with iMobile SE to support peer-to-peer mobile computing. The rest of the paper is organized as follows. Section 2 describes the basic architecture of iMobile

ME. Section 3 uses several scenarios to describe network synchronization through ME. Section 4 describes a particular implementation of iMobile ME. Section 5 discusses related work and several design issues. Section 6 gives a summary of the ME work.

2. ARCHITECTURE OF IMOBILE MICRO EDITION

Figure 3 shows a sample architecture diagram of iMobile Micro Edition (ME). ME devlets provide communication interfaces to the ME dispatcher (msgnet), for both the local device user and remote users. ME infolets provide access to local resources - those on the device or accessible thru the device's communication ports. The ME engine arbitrates communications among the front-end devlets and the backend infolets. As a mobile device may be disconnected or have connections with varying bandwidth, it's difficult to guarantee a long lasting session between two interacting mobile devices. Therefore, each remotely accessible devlet/infolet is extended with an inbox queue that accumulates incoming requests/responses and an outbox queue that accumulates outgoing responses/request. iMobile ME relies on an iMobile standard edition server (SE) on the network acting as a simple router to locate another mobile device. An SE server is also enhanced with an inbox and outbox that allows ME devices to store and forward their requests/responses through message queue synchronization.

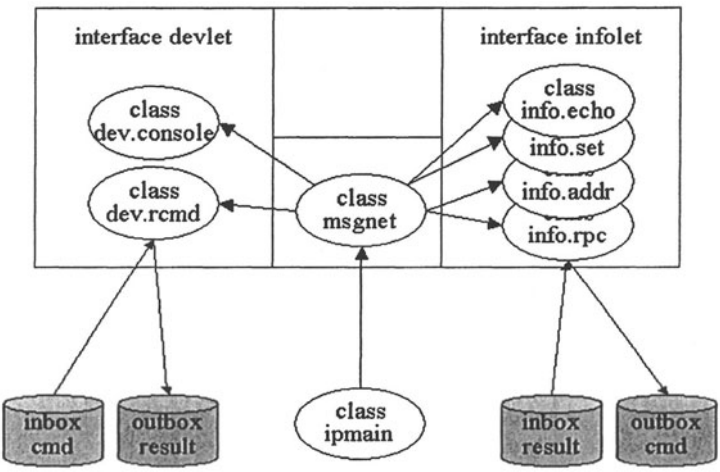

Figure 3. Architecutre of iMobile Micro Edition

Each devlet or infolet is implemented as a Java class. Each iMobile ME typically has at least two basic devlets:

- *console* : It allows the mobile user to directly interact with ME through a command console
- *rcmd* : It allows ME to receive service requests from remote mobile devices through its associated inbox queue. If a mobile device (such as iPAQ) is powerful enough to run a web server, we can also provide an HTTP devlet.

We also show four example infolets in Figure 2:

- *echo* : It simply echoes the string sent to it. This infolet is useful for testing the connection and for measuring the lower bound of round-trip delays from one mobile device to the other, independent of any specific infolet implementation.
- *set* : It sets system parameters (such as SE synchronization host and the name of the local mobile device) of the iMobile ME environment.
- *addr* : It exposes the local address book entries, which can be found on most mobile devices. Similarly, if a mobile device has a built-in location determination technology or a way to obtain its surrounding temperature (probably through bluetooth to a nearby sensor), we can implement *location* or *temperature* infolets to expose the local environment to remote mobile devices.
- *rpc* : This is the essential infolet for a mobile device to send service requests to a remote device. It exposes the accessible infolets of all reachable mobile devices to the local mobile device.

Figure 4 shows how a mobile user can use the *console* devlet to access a local address book entry thru the *addr* infolet of a typical ME device. The large console window shows responses returned from the console devlet. The single line that starts with "<cmd:>" allows the mobile user to access infolets through a command line interface. The *Exec* button executes the command line and places a request in the outbox if it deals with a remote service. The *Queue* button examines the inbox and outbox queues. The *Sync* button performs queue synchronization with the iMobile router when the mobile device is connected to the network. The *Exit* button allows the user to leave the ME system.

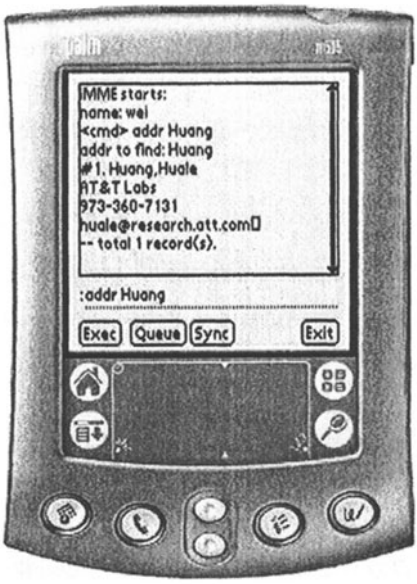

Figure 4. Accessing the addr infolet from the console devlet

3. ME QUEUE SYNCHRONIZATION

We use two scenarios to demonstrate how two ME devices communicate with each other: the first scenario (Figure 5) shows how ME responds to a remote service request (*echo*); the second scenario (Figure 6) shows how ME sends a service request to a remote mobile device.

3.1 Scenario 1: Remote Access of a Local Infolet

In Figure 5, the inbox of the *rcmd* devlet received a service request at its inbox. The request was routed to its *echo* infolet through the *msgnet* class. The result is then returned to the outbox of the *rcmd* devlet. If the device is disconnected at this point, the result will be queued in the outbox queue until connection is restored. If the device is connected, a synchronization operation will enable the request to be sent to either the remote device directly or through a network-based router. If the remote device is disconnected at that time, an iMobile router will enable the request to be

stored and forwarded later when the remote device performs a synchronization operation.

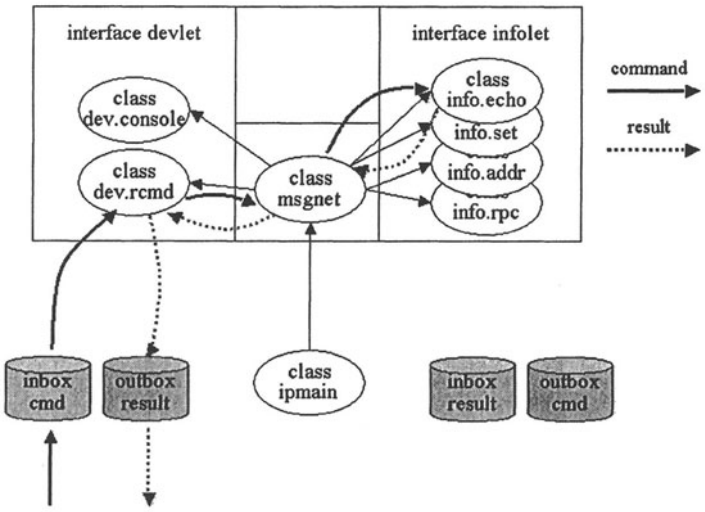

Figure 5. Scenario 1 - Remote Access of a Local Infolet

3.2 Scenario 2: Access of a Remote Infolet

In Figure 6, a mobile user uses the console infolet to access the rpc infolet, which allows the user to access remote infolets. The request is first queued in the outbox of the infolet until the network connection becomes available and until the user activates the synchronization operation. When the result becomes available and arrives at the inbox (through another queue synchronization), then it's delivered to the console.

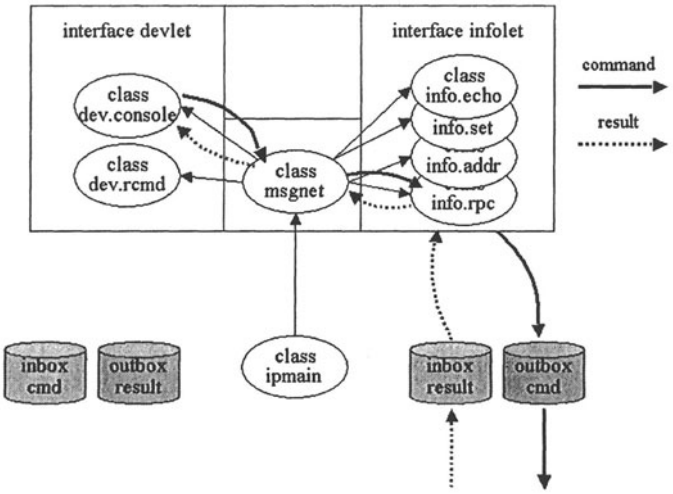

Figure 6. Scenario 2 - Access of a Remote Infolet

Figure 7 shows the complete picture of how an ME system performs queue synchronization with a network-based iMobile platform (SE). As mobile devices may become disconnected occasionally, the always-connected network-based platform serves as a simple router that also stores and forwards requests and results.

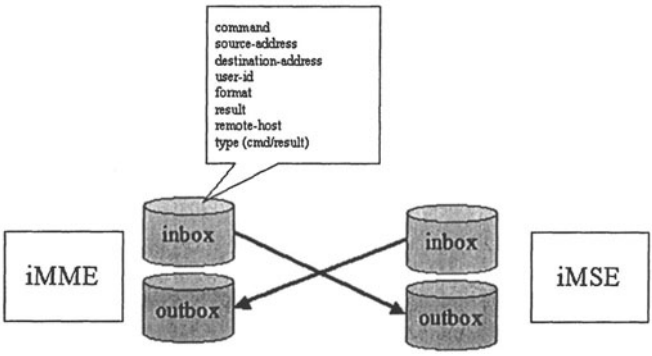

Figure 7. ME Queue Synchronization

4. IMPLEMENTATION

We have implemented iMobile ME as described in Section 3 on both Palm and iPAQ devices. The Palm implementation is based on the CLDC (Connected Limited Device Configuration) of J2ME [7] and KVM (Kilo Virtual Machine), a Java virtual machine for small, resource-constrained devices. It runs on both Palm III and Palm V devices. The following scenario demonstrates in further detail how this implementation allows a mobile device **chen**, to access the address book entry of "*huang*" on the other mobile device **wei**, with the help of an iMobile router and queue synchronizations.

4.1. Send command from ME:chen to SE (iMobile router)

From the console devlet of **chen**, the user first sets the synchronization host (iMobile router) and then issues a request to access the remote address book entry of Huang using the *rpc* devlet:

```
<cmd> set synchost=imse.research.att.com:8080 //router
host:port number
<cmd> set syncurl=/bin/httpsync.cgi //synchronization
program on router
<cmd> rpc wei addr Huang
```

The request will be stored in the outbox until the user issues a synchronization request:

```
sync message
send> addr Huang
sync OK.
```

At that point, the request is delivered to the iMobile router. The request includes attributes such as command, source-address, destination-address, format, result, remote-host, and type (cmd/result).

4.2. Retrieve command from SE to ME:wei - execute, and send the results back to SE

Note that the recipient device (**wei**) may not be connected and it will have to explicitly send a synchronization request to the iMobile router to receive the remote command and then send the result back to the iMobile router.

```
sync message
rcmd> addr Huang
send> addr Huang
sync OK.
```

At this point, the original sending device (**chen**) may be disconnected, so the result may have to be queued in the outbox of the iMobile router.

4.3. Send the result from SE to ME:chen

When the sending device (**chen**) becomes connected and initiates a synchronization operation, the result in the outbox of the iMobile router is then retrieved and sent back to **chen**.

```
sync message
[Remote answer for chen]
Sender: console:self (rcmd:console:self@chen)
addr to find: Huang
AT&T Labs: 973-360-7131
huang@research.att.com
--- total: 1 record(s)
sync OK.
```

Figure 8 shows two screen shots that capture these interactions between two ME devices: **chen** and **wei**. Note that the same mechanism can be used by **chen** to access the location information of **wei** if a location infolet is available on **wei**, regardless of what location determination technology is used on **wei**'s device.

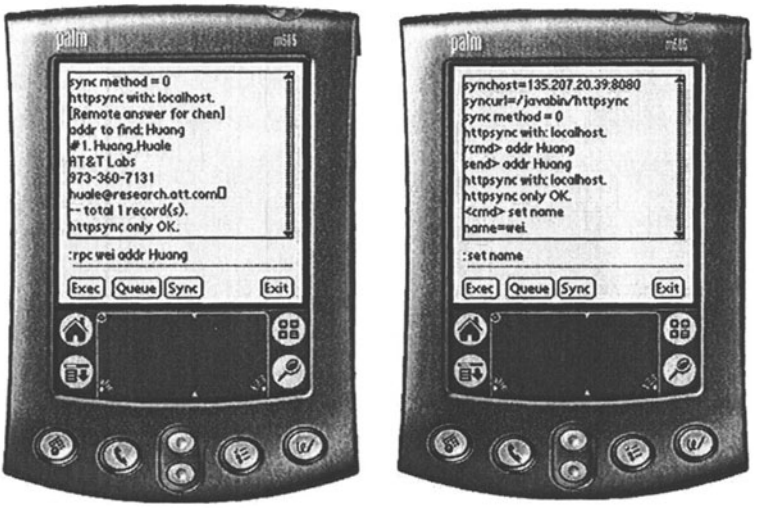

Figure 8. Interactions between two ME devices: the device chen accesses the address book entry of Huang on the device wei

The above implementation does not address several important issues: naming, security, and scalability. These issues will be addressed as mobile devices become more powerful and as we migrate the iMobile router implementation from the iMobile standard edition (SE) to the iMobile enterprise edition (EE) [4]. iMobile EE differs from iMobile SE in several ways:

- *Service Profile Database:* iMobile EE provides an extensible data model that describes all devices, users, resources, permissions, groups, etc. and their relationships. The database can be extended to give each device a unique name in the form of *user.deviceid* (*such* as *chen.ipaq).* It can also cache the list of infolets available on each device and specify which other users have the privileges to query this info.
- *Gateways:* iMobile EE replaces each devlet with a gateway that interacts with the iMobile authentication or a corporate authentication service (such as RADIUS [8] or Windows domain authentication) to verify a user before communication with an iMobile server is allowed. An HTTPS gateway also allows encrypted communication between the iMobile router and mobile devices that support HTTPS-enabled browser clients.

- *Replicated iMobile Servers:* iMobile EE employs Java Message Service (JMS) [9] as application messaging middleware to decouple iMobile gateways from iMobile servers, which host all the infolets. As a result, more iMobile servers can be allocated to help route service requests as more mobile devices engage in peer-to-peer mobile computing.

5. RELATED WORK AND DISCUSSIONS

Typical P2P systems differ from traditional client-server computing by allowing every node to act as both a client and a server. Each node participates in the P2P network by sharing its resources with others [10]. iMobile ME shares the same spirit, with a particular focus on providing a simple infrastructure that involves a network-based application router and a simple device-independent platform on each mobile device. This infrastructure allows communications and resource sharing among mobile devices through their message queue synchronizations with the iMobile router (Figure 1).

Our P2P infrastructure is similar to Napster [1] in that it also relies on a third-party, the iMobile router, to locate another mobile device, but it differs from Napster in several important ways:

- *Communication mechanisms*: While Napster focuses on desktop users with internet connections, iMobile ME, by inheriting the *devlet* abstraction from iMobile SE, aims to support multiple communication protocols, including HTTP, SMS, choice of Instant Messaging protocols (AIM or Jabber[11]), etc.
- *Resource access*: While Napster focuses on MP3 or WMA music files, iMobile ME, by inheriting the *infolet* abstraction from iMobile SE, aims to support multiple forms of resources such as location information, camera views, and other resources that can be available in the local environment through specialized interfaces.
- *Queue synchronization:* While most Napster users remain connected during the file transfer, the nature of intermittent communication capabilities of mobile devices requires that we provide a request/response queue synchronization mechanism between ME devices and the network-based iMobile router. Requests or responses are not lost even if the receiving devices are not readily available.

- *Service Discovery:* Unlike Napster, the iMobile router does not store an index of all infolets available on all mobile devices. As mentioned previously, an extension of the user and device profiles maintained by iMobile SE or EE would allow a mobile device to query the capabilities and infolets of other mobile devices. On the other hand, this approach requires that the iMobile router maintain an up-to-date database. We currently provide a primitive *help* infolet on each mobile device that lists the infolets available on that device.

Unlike Napster, Gnutella [12] is another P2P system that does not require any central server or database. Each Gnutella peer uses a constrained broadcast mechanism to send packets it receives to all of its peers with a "time-to-live" parameter set on each packet. We are currently adding a group communication mechanism to iMobile ME so that it can also have ad-hoc P2P networking in addition to the current infrastructure-based P2P networking. The group communication module in each iMobile ME instance will allow it to find nearby ME devices and start communicating with each other without relying on any network-based router.

The JXTA [13] project from Sun Microsystems provides a network programming and computing platform to support peer-to-peer computing. It defines several protocols upon which JXTA technology-based services and applications can be developed. This is a centralized architecture in the sense that all applications have to initially connect to the JXTA server. Most JXTA applications today are still targeting peer-to-peer desktop devices over the Internet. JXTA for J2ME [14] is a new project that focuses on the support for resource sharing among mobile devices. Due to the resource constraints on small devices, JXTA replays or proxies are employed to act on behalf of JXTA for J2ME peers at the network edge to provide interoperability with other JXTA peers and protocols.

Note that the queue synchronization mechanism provided by iMobile ME is different from systems such as CompanionLink [15], and XTNDConnect Server [16] which allow mobile devices to access enterprise databases or servers through a network synchronization mechanism. These systems are mainly designed for data synchronization, with very little support for communication among mobile devices for the purpose of sharing resources.

Recently, SyncML [17] is becoming a common language for synchronizing all devices and applications over any network. SyncML leverages Extensible Markup Language (XML), making it platform independent. This is an important step towards standardizing data exchanges among mobile devices. Since the iMobile ME architecture is designed to support multiple communication protocols on mobile devices, we plan to add a SyncML devlet to support this XML-based data exchange mechanism.

There are also research interests to provide a platform in order to use handheld devices and desktops together in a seamless way. The Pebbles project at CMU [18] explores how small handheld devices can serve as a useful adjunct to the "fixed" computers. The CANS project at NYU [19] aims to provide a user with seamless, ubiquitous access to a service irrespective of the user's end device and location. The vision of these projects is to use handheld devices as extensions of desktop devices. iMobile ME, on the other hand, provides information access and exchange facilities among mobile devices.

The Berkeley Ninja project developed an interesting approach of secure service discovery based on Public Key Cryptography and a capability manager to avoid relying on a central server to verify a user's access rights during service invocations [20]. As mobile devices become more powerful with adequate support for the Java Cryptography Architecture (JCA), such as JCSI (Java Crypto and Security Implementation) ME SSL for the CDC configuration and PersonalJava [21], we plan to explore the possibility of building a similar security infrastructure for iMobile ME.

6. SUMMARY

iMobile ME provides a uniform architecture on mobile devices to allow these devices to both communicate with and access resources from each other. The devlets manage the communication interfaces to each mobile device, while the infolets manage device-specific accesses to local resources. Devlets that allow remote communications are equipped with an inbox for accumulating incoming requests and an outbox for storing outgoing results. Requests and results are stored in the inboxes and outboxes until a queue synchronization operation is performed. iMobile ME employs a network-based platform (such as iMobile SE) to act as a router to help locate other mobile devices. It also allows requests and results to be stored on the iMobile router until the receiving mobile devices perform a synchronization operation.

As mobile devices become more powerful in communication, computation, and storage capabilities, we believe that the traditional client-server computing structure will be challenged and a lot of computations will be shifted to the mobile devices themselves. iMobile ME aims to provide an

infrastructure to facilitate peer-to-peer mobile computing and to address growing needs in this area.

REFERENCES

[1] Naspter Inc., http://www.napster.com .
[2] H. Rao, Y. Chen, D. Chang, M. Chen, "iMobile: A Proxy-based Platform for Mobile Services", The First ACM Workshop on Wireless Mobile Internet (WMI 2001), Rome, July 2001.
[3] Yih-Farn Chen, Huale Huang, Rittwik Jana, Sam John, Serban Jora, Amy Reibman, Bin Wei, "Personalized Multimedia Services Using a Mobile Service Platform", Proceedings of the IEEE Wireless Communications Networking Conference, Florida, March 17-21, 2002.
[4] Yih-Farn Chen, Huale Huang, Rittwik Jana, Trevor Jim, Radhakrishnan Muthumanickam, Sam John, Serban Jora, Bin Wei, Matti Hiltunen, "iMobile EE - An Enterprise Mobile Service Platform", to appear in ACM Journal on Wireless Networks.
[5] CORBA: Common Object Request Broker Architecture, http://www.corba.org
[6] X-10.org, http://www.x10.org
[7] Sun Microsystems, J2ME, http://java.sun.com/j2me/ .
[8] Remote Authentication Dial In User Service (RADIUS), http://www.ietf.org/rfc/rfc2138.txt .
[9] Sun Microsystems, "Java Message Service API", http://java.sun.com/products/jms/ .
[10] Karl Aberer and Manfred Hauswirth, "Peer-to-peer information systems: concepts and models, state of the art, and future systems", 18th International Conference on Data Engineering, San Jose, Feb. 2002 .
[11] Jabber, http://www.jabber.org.
[12] Gnutella, http://www.guntella.com .
[13] Jxta.org,, "Project JXTA: An Open, Innovative Collaboration", white paper, http://www.jxta.org/project/www/docs/OpenInnovative.pdf , April 2001.
[14] Akhil Arora, Carl Haywood, Kuldip Singh Pabla, "JXTA for J2ME – Extending the Reach of Wireless with JXTA Technology", white paper, http://www.jxta.org/project/www/docs/JXTA4J2ME.pdf , March 2002.
[15] CompanionLink, http://www.companionlink.com .
[16] Extended Systems, http://www.extendedsystems.com .
[17] SyncML, http://www.syncml.org .
[18] The Pittsburgh Pebbles PDA Project, http://www.cs.cmu.edu/~pebbles .
[19]Xiaodong Fu, Weisong Shi, Anatoly Akkerman, and Vijay Karamcheti, "CANS: Composable, Adaptive Network Services Infrastructure", USENIX Symposium on Internet Technologies and Systems (USITS), March 2001.
[20] Steven E. Czerwinski, Ben Y. Zhao, Todd D. Hodes, Anthony D. Joseph, Randy H. Katz, "An Architecture for a Secure Service Discovery Service", Proceedings of The Fifth ACM/IEEE International Conference on Mobile Computing (MobiCom '99), Seattle, WA, August 1999, pp. 24-35.
[21] Wedgetail Communications, Java Crypto and Security Implementation (JCSI) Micro Edition, http://www.wedgetail.com/jcsi/microedition/ .