PDAgent: A Platform for Developing and Deploying Mobile Agent-enabled Applications for Wireless Devices

Jiannong Cao, Daniel C.K. Tse, and Alvin T.S. Chan
Internet and Mobile Computing Lab
Department of Computing
Hong Kong Polytechnic University
Hung Hom, KLW, Hong Kong

Abstract

Mobile agents (MAs) can support mobile computing by carrying out tasks autonomously for a mobile user temporarily disconnected from the network. In this paper we describe PDAgent, a lightweight and highly portable platform for developing and deploying mobile agent-enabled applications on wireless handheld devices. Our approach offers the following advantages: (i) It does not require installing MA server on handheld devices and supports the adoption of any kind of mobile agent system at network hosts, while at the same time allow mobile users to have control of mobile agent activities; (ii) It supports dynamic downloading of MA-enabled applications which can be made aware of and adaptive to the user's current context, and deploying the application to perform various tasks on behalf of the user over the wired network; (iii) It requires the minimum network connectivity and cost for users to deploy their applications. PDAgent provides APIs which allow developers to build new mobile applications on top of PDAgent. Example applications have been developed for PDAgent. We will report on and discuss the results of experimental performance evaluations of PDAgent.

Key words: Mobile agent, mobile computing, mobile network access, wireless devices

1. Introduction

Wireless handheld devices, such as cellular phones and PDAs, are becoming increasingly popular. In the past few years, many mobile computing applications have been developed for handheld devices including m-commerce, mobile office, and mobile service/information access applications. However, these applications running on mobile devices in a wireless network still suffer from limited computing, battery power and storage capability, low bandwidth, greater latency, as well as having high network connection costs.

Mobile agent technology has been proposed as one way to overcome the limitations of connectivity, latency and bandwidth [5, 4, 7]. Mobile agents can move through the network of sites to search for, filter, and process information they need to accomplish their tasks [10, 18]. They can also cooperate with each other by sharing and exchanging information and partial results, and collectively making decisions [1, 2]. Mobile agents have also been used to develop middleware for mobile computing [15].

In this paper, we focus on mobile agent support of mobile computing by carrying out tasks for a mobile user temporarily disconnected from the network. The mobile user can dispatch a mobile agent to perform these tasks autonomously over the network. Once the agent is dispatched, the user can disconnect from the network. Later on the user reconnects to the network to collect the result returned by the mobile agent. To provide this support requires a suitable MA platform and tools for mobile users to deploy mobile agents and manage their execution from wireless handheld devices.

We describe a general approach and a lightweight and highly portable platform, called *PDAgent,* for developing and deploying mobile agent-enabled applications from wireless handheld devices. Although MA platforms have been developed for small devices [11, 12, 16], our approach has a number of advantages:

(i) Most of the existing approaches to deploy mobile agents on wireless devices are bound to one specific mobile agent platform. PDAgent, however, does not require the installation of any MA server on the wireless handheld devices and supports the adaptation of any kind of mobile agent system at the network host as the underlying runtime platform. This feature makes mobile applications highly portable across handheld devices and networks with different MA servers.

(ii) It provides mobile users reliable and efficient mobile computing services in a low-bandwidth mobile network environment, with the minimum network latency and cost. In our approach, the user downloads the MA code for an application and then deploys the application by passing the code with appropriate parameters to a proxy gateway. Then the user can disconnect from the network. The gateway serves as a middle-tier to manage the mobile agent dispatched from the mobile user. It provides a more

Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)
0190-3918/04 $20.00 © 2004 **IEEE**

powerful processing capability that can reduce the CPU loading within the wireless devices. It also helps to provide a reliable network connection to the Internet so that when the user goes offline, it can communicate and retract the mobile agent. For interoperability, both the agent code and the parameters are encoded with XML.

(iii) It supports dynamic downloading and deployment of MA-enabled mobile applications, which can be made aware of and adaptive to the current user's context.

PDAgent's APIs provide function primitives for programming mobile agent-enabled activities, such as downloading mobile agent code, dispatching mobile agent, disposing a mobile agent, and managing the mobile agent operations. Using the APIs, the programmer can develop various new applications on the PDAgent platform.

The rest of this paper is organized as follows. Section 2 describes related work, comparing various approaches for mobile users to access information and run applications in wired networks. We also review research on mobile agent platforms adapted to handheld devices and compare them with the design of PDAgent. Section 3 presents the design of the PDAgent platform. Chapter 4 describes the implementation and performance evaluation of PDAgent with sample applications. Chapter 5 concludes the paper with the discussion of our future work.

2. Related work

In the past several years, researchers have been seeking efficient solutions to the problem of providing reliable Internet service to users with wireless devices. Various approaches have been proposed. The major ones as illustrated in Figure 1, including the client-server-model, client-agent server model, and mobile agent server model.

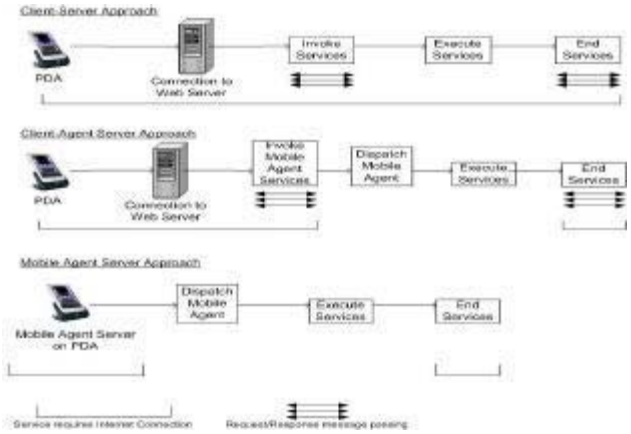

Figure 1. Approaches to mobile Internet Computing

The *Client-Server* approach makes use of the traditional Client-Server model, in which a mobile client communicates with the web-server to invoke Internet services. In this approach, the mobile user has to keep the connection with the wired network until the service is completed and the result is obtained [6].

The *Client-Agent Server* approach makes use of an Agent Server, which is a combination of a web server and a mobile agent server. The mobile user does not need to maintain the Internet. Instead, the mobile user only needs to submit the service request to the server and can then disconnect from the network. The agent server will determine and launch a mobile agent to execute the requested network services without having any interaction or communication with the mobile client. The mobile agent will later bring back the execution result and temporarily store it at the agent server. Afterwards, the mobile client can collect the execution result by reconnecting to the agent server [17]. This approach has a limitation that a mobile user is provided with only MA-based applications which must have been installed on the agent server.

The *Mobile Agent Server* approach makes use of a Mobile Agent Server installed on the mobile client device. Mobile users can program mobile agents and store them on their handheld devices. No middle-tier server is needed to launch the mobile agent application in order to access network services, assuming network sites are capable of hosting the specific forms of mobile agents. All the preprocessing can be done on the handheld device without being connected to the Internet. The connection is needed only when the mobile client wants to dispatch or retract the mobile agent [11, 12, 16].

Compared with the client-server and client-agent approaches, the Mobile Agent Server approach provides a better way for mobile access to Internet services because it allows the mobile user to store and launch mobile agent-enabled applications, and reduces network connection time and the amount of message-passing between clients with servers. Reducing network connection time also decreases the possibility of network errors. However, developing and installing a mobile agent platform on wireless handheld devices is not an easy job; there are limitations on storage capacity and processing power. Another major problem with this approach is the difficulty of achieving interoperability between the proprietary platform on the handheld device and the mobile agent servers located in network sites.

In this paper we propose an alternative approach called the *Agent-Proxy-Server* approach, as illustrated in Figure 2. In this approach, no mobile agent server needs to be installed on the handheld device. Instead, it can be adapted to use any kind of mobile agent server supported by hosts in the target service network (e.g., either Intranet or a VPN over the Internet). A gateway acts as the middle-tier service bridge. The code for mobile agent-enabled applications are downloaded and stored in the handheld devices. The mobile user can launch the application by deploying the code with necessary

parameters, both are encoded in XML for interoperability, to the gateway, which accepts and interprets the mobile agent code, wraps it into a mobile agent in a form supported by the network sites, and dispatches the mobile agent on behave of the mobile user. The handheld device acts as remote server controller, with the capability to perform all the mobile agent functions and to administer the mobile agent server to manage the mobile agent operations.

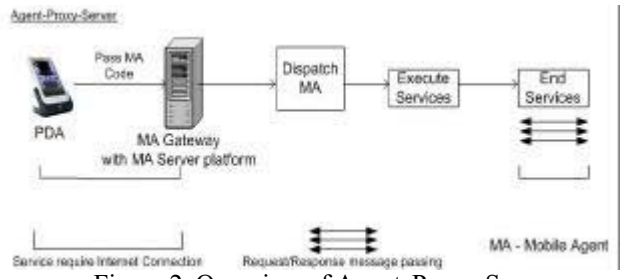

Figure 2. Overview of Agent-Proxy-Server Approaches

Our approach has the advantages of the previous approaches while at the same time solves both problems of resource limitations and dependency on any specific proprietary mobile agent system. More importantly, it provides a potential to address MA interoperability problem by facilitating a standard MA code format (e.g., specified using XML) which can be understood and interpreted by gateways and different MA servers. In addition, by providing a mechanism for downloading and deployment of MA-enabled applications, it supports the development of context-aware and adaptive mobile applications. MA programs can be designed in a way that can be parameterized, either manually or automatically, to reflect the current user's context. The mobile user can pre- or dynamically download the MA code to be used in a particular context, and further customize the code by specifying appropriate parameters, before launch the application. This approach has been advocated recently by researchers because it is believed that applications may have valuable information that could enable the middleware system on the wired network to deliver services more efficiently [3]. We observe that for most mobile applications, the MA code is of a size ranging from 1KB to 8KB, and can be compressed before download into the wireless device. Therefore, it should not cause big problems in terms of storage capacity and network latency.

3. Design of PDAgent

Based on the proposed Agent-Proxy-Server approach described in the last section, we have developed a lightweight and highly portable platform, called PDAgent. Figure 3 shows the operating environment of PDAgent. There are several important issues to consider in the design of PDAgent:

- *Mobility* – the mobile user can access network services in different locations from all wireless devices that support the PDAgent platform.
- *Being lightweight* – The PDAgent platform should be able to run on resource-constrained devices.
- *Adaptability* – the PDAgent platform should support mobile agents that can run on the different kinds of mobile agent servers located in network sites.
- *Network Connectivity* – Wireless handheld devices are not permanently connected to the network and can often be disconnected for long periods of time. Even if the wireless device is connected, the connection often suffers with low bandwidth and high latency. The PDAgent platform should provide mobile users with reliable and efficient network access minimizing network connection. It should also minimize the size of information packages transmitted during the running of applications when interacting with network services.

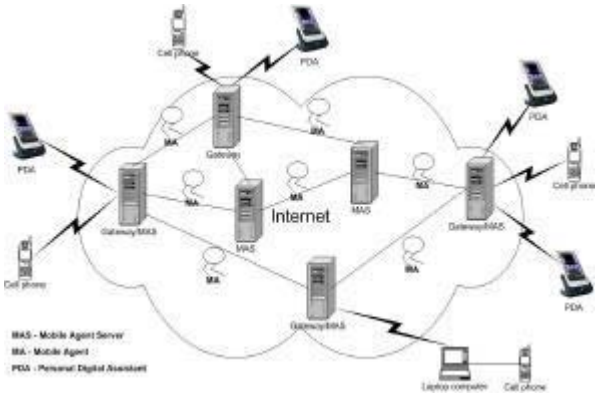

Figure 3. System environment of PDAgent

The overall system structure of PDAgent consists of three main parts: a *PDAgent Platform* on wireless devices, the *Gateway,* and the *Mobile Agent Server*. The Mobile Agent Server can be any kind of Mobile Agent Server located on a high-end desktop in a network site. Figure 4 shows the architectures of the PDAgent system and the Gateway.

The PDAgent Platform is the main component, which provides functions to mobile users for downloading of mobile agent code, internal mobile agent administration, and agent launching and management. The components of the PDAgent Platform are separated into UI and System API. UI is the interface via which mobile users submit service requirements and administer mobile agent activities both internally and externally. System API separates the a background process running inside the PDAgent platform that facilitates tasks such as dispatching mobile agents, disposing mobile agent and managing internal Database.

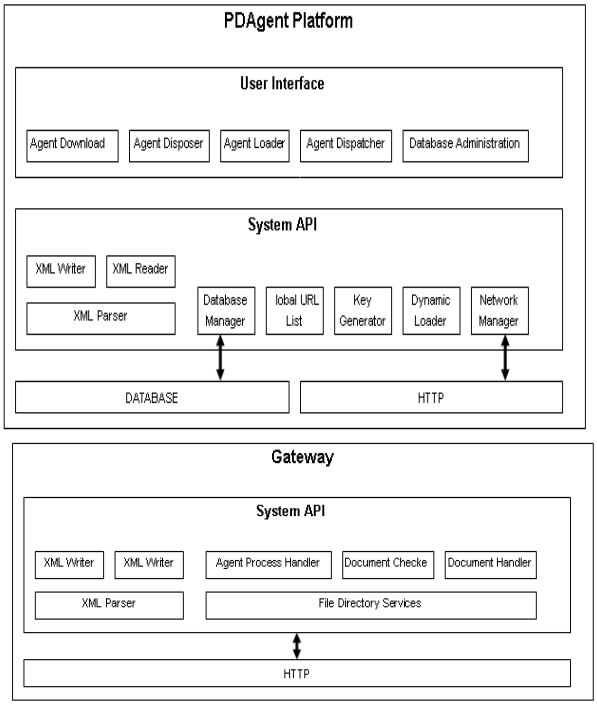

Figure 4. System Architecture of PDAgent Platform and Gateway

Gateway is a middle-tier communication and operation bridge. It is responsible for transferring information between the mobile agent platform located in wireless devices and the mobile agent server located in Gateway.

To support interoperability, XML is used to encode the MA code and the parameter/result data transferred between the PDAgent platform on handheld devices and the Gateway. The PDAgent platform also offers secure design by encrypting MA code and data during wireless transmission. As described in the next section, PDAgent implementation uses an opened source lightweight XML API called 8 for XML conversion and interpretation. The size of kXML [8] API is very small. PDAgent collects client request and converts it into XML document by using kXML API. The XML document is compressed within the wireless devices before being transferred to the gateway. This minimizes the size of the transferred packet and thus reduces the transmission time. Using simple text compression algorithms, the compression process requires only small amount of CPU time.

In the remaining part of this section, we will describe the main functions that PDAgent provides for acquiring services, and for providing reliable services and security.

3.1. Service Subscription

A mobile user can subscribe network services to downloading the service mobile agent code from the nearest trusted Gateway (see Figure 5). Upon downloading, the PDAgent platform will store the mobile agent code into its database on the wireless device. Each MA code downloaded will be assigned a unique id by the platform for the purpose of authorization in later execution. Once the service agent code is present in PDAgent's database, the subscription is no longer needed.

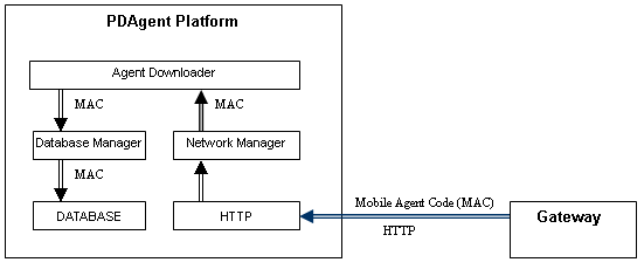

Figure 5. System Architecture of service subscription

3.2. Service Execution

Figure 6 shows the interactions that take place between system components when invoking a service. To deploy an application, the application interface is loaded within the wireless device. The mobile user enters service parameters using the application interface without being connected to the network. The Agent Dispatcher will collect the agent code and parameters, generate a unique key from the assigned code id, encode them into a XML document, and pass it on as a single package, called "Packed Information" (PI), to the Network Manager. The Network Manager is responsible of uploading the packed information to the Gateway through a HTTP connection.

When the Gateway receives the packed information, it will pass the information to the Agent Dispatch Handler. The Agent Dispatch Handler will separate this information into modules and pass them on to the XML Writer, Agent Handler, and Document Handler for further extraction. The XML Writer will read the xml document and parse all the user requirement parameters. The Agent Creator will generate mobile agent classes from the information if the supplied unique key is valid. The Document Creator will create different files from information for the Mobile Agent Server to collect later on. Once the document and classes are ready, the File Directory will allocate a space for storing these document and classes, and then it will signal the Mobile Agent Server to create a mobile agent from the user's uploaded classes and dispatch it to the Internet.

3.3. Service Result Collection

In our current design, the mobile agent will return to the Gateway where it is dispatched after the service execution is completed. The result it brings back will be wrapped in XML format. The mobile user can download the XML document from the Gateway by reconnecting to the Internet. The PDAgent platform on the wireless handheld device will extract the XML document and

display it on the user interface.

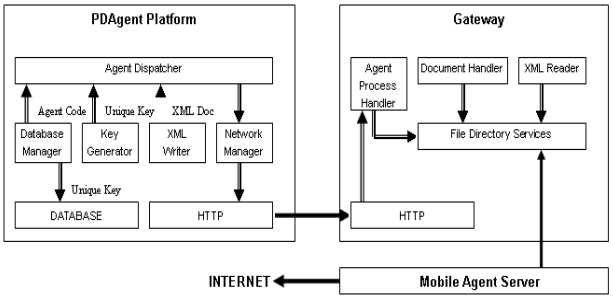

Figure 6. System Architecture of service execution

3.4. Information Security Management

One of the important issues that PDAgent addresses is that of providing secure information. Although implementing a comprehensive security service is beyond the scope of this paper, we have attempted to handle as much as possible security issues in developing the PDAgent platform.

First, the design of PDAgent advocates the provision of MA service code for mobile users to download from trusted gateways. This, to certain degree, will solve the traditional security problem of mobile agent. Second, data submission in PDAgent is secured since it is done without connecting to the Internet. However, the mobile user is required to send packed information to the Gateway via a HTTP connection during code dispatching and result collection. At this point, data may be stolen and/or altered. PDAgent provides security control which is based mainly on the existing Internet security model, using "Asymmetric Key Encryption" to identify the PDAgent user and encrypt the data during the dispatching process (see Figure 7). PDAgent will encrypt user's information using a public key [14] and then wrap it into XML format, generating the so called Packed Information. Once the Gateway receives the Packed Information, it will use MD5 [14] to verify whether the Packed Information is valid if so. It will then extract the mobile agent code and the mobile client's requirements by using its private key to decrypt the Packed Information.

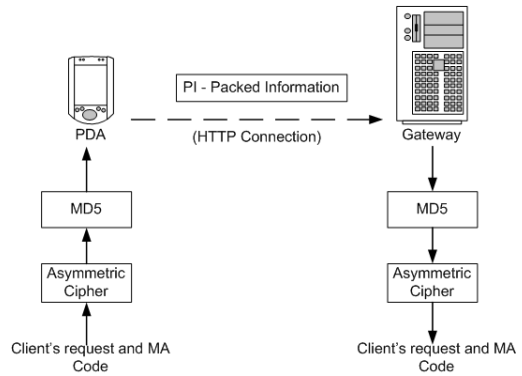

Figure 7. PDAgent's Security Model

3.5. High Performance Service Management

Reducing network connection time and network error is an important issue in designing the PDAgent platform. In order to provide high performance services, PDAgent executes a service mobile agent on behalf of the mobile user, thereby making use of the Gateway to reduce the network connection time. Furthermore, to ensure that communication between the wireless device and the Gateway will not affect overall operational performance, PDAgent will find the nearest Gateway for sending the Packed Information.

Initially, PDAgent will download a list of gateway addresses from the central server. This list will be used until the Round Trip Time (RTT) from the nearest gateway found in the list exceeds the pre-defined threshold. In this case, the PDAgent will request for a new address list from central server or through one the gateways.

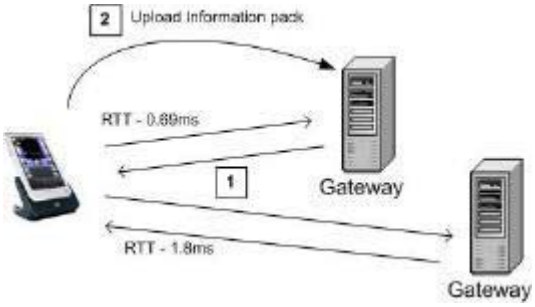

Figure 8. PDAgent's High Performance Service Management Model

As shown in Figure 8, the PDAgent platform will find the nearest Gateway by sending 1-bit data to all the gateways on the address list and calculating which Gateway takes the shortest Round Trip Time (RTT). The PDAgent platform will send the Packed Information to the Gateway with the shortest RTT. Assuming that the network bandwidth and the size of Packed Information is the same, this can minimize data transfer time, with the minimum transfer time being dependent on the shortest RTT from the wireless device to the Gateway.

3.6. Mobile Agent Management

The mobile user can manage the mobile agent from the PDAgent's platform on the wireless device. The concept is the same as managing mobile agent activity from a mobile agent server located on a high-end desktop. For example, the mobile user can invoke functions to clone an agent, retract an agent, dispatch an agent, and view agent status.

PDAgent provides a set of APIs that help application developers to build their own mobile applications. The API contains primitives for dispatching mobile agents, monitoring mobile agent activities, retracting mobile

agents from the Internet, and disposing mobile agents. In addition to these core functions, PDAgent APIs also provide functions for internal system management and network management. System management allows the programmer to write code for generating unique keys, and reading and writing XML documents. Network management is responsible of managing all the activities that require wireless network connections from wireless devices to gateways, such as downloading mobile agent code and upload packed information.

Because PDAgent supports any kind of mobile agent system as the underlying operating environment on the gateways and on network hosts, the developer can use development tools provided by the underlying mobile agent system (e.g., Aglets, Voyager etc) to build their applications.

4. Implementation and evaluation of PDAgent

A prototype of PDAgent has been implemented. As with the design, the implementation is in three main parts: *PDAgent platform* on wireless devices, *Gateway,* and *Internal mobile agent manager*.

The PDAgent platform was implemented using Java CLDC / MIDP [20] in the J2ME technology [19]. J2ME is a lightweight VM supporting Java applications on wireless devices. It addresses the problems of storage capacity and low processing power on wireless devices. For XML encoding and parsing, kXML [8], a light-weigh XML API package for J2ME technology, is used. The kXML API supports DOM, DTD and generic XML parsing functions. The database located inside the PDAgent platform on the handheld devices was implemented using J2ME's Record Management System (RMS). RMS is a persistent storage mechanism modeled from a simple record-oriented database. Figure 10 shows some screen captures for PDAgent's main functions. The PDAgent platform is very lightweight. To store the PDAgent platform together with the kXML package within the wireless devices requires only 120KB storage space.

(a) (b) (c)

Figure 9 a. Platform Main Screen; b. Mobile Agent Management; c. Internal Database Management

A web server is installed inside Gateway for request/response though a HTTP connection. We used Tomcat as the Gateway web server because Tomcat is well known and widely used in most business sectors. The Gateway's functional component was implemented using Java Servlet.

IBM Aglets [9], a well known Java-based mobile agent system, is used as the Mobile Agent Server. It is worth noting that any mobile agent system can be used. We chose IBM Aglets because it is stable and easy to use.

To experiment and test the performance of PDAgent, we have developed several example applications, for example, Food Search Engine, E-Banking etc. The results of the experiments described here have been collected from running an "e-banking" application, in which a mobile client makes transaction requests from one bank site to another bank site.

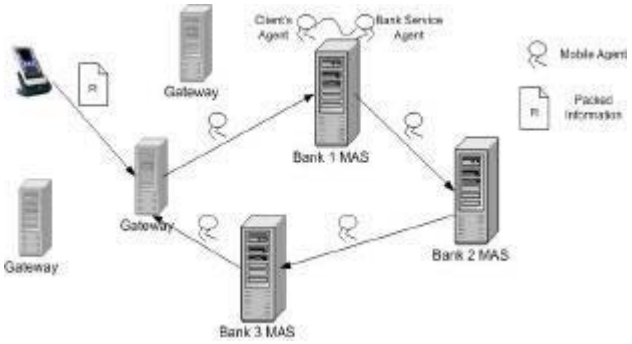

Figure 10. Runtime environment of E-Banking application

Figure 10 illustrate the runtime environment of the e-Banking application. The mobile user submits the transaction information (Figure 11b) from the wireless device without connecting to the Internet. PDAgent Platform collects client transaction information and mobile agent code wrapped as Packed Information (PI), and sends it to the nearest Gateway. The platform will also display the dispatched agent ID (Figure 11c). The Gateway will dispatch mobile agent to execute the service on behalf of the mobile client. In this application, there is a Mobile Agent Server (MAS) with a Service Agent within each bank. When the client's agent arrived at each bank, it will execute the transaction by communicating with the Service Agent. If the transaction is completed, the Service Agent will return transaction details to the client's agent, which will dispatch itself to other banks to continue the transaction execution. At last, the client's agent will return to Gateway and create a XML document containing all the transaction details. Later on, the mobile user can view the transaction result (Figure 11d) by downloading the XML document from the Gateway.

Performance is evaluated by running the same application and comparing the performance with other two approaches: a client-server and a web-based

approach – accessing Internet services through a web browser on a high-end desktop. Performance metrics used are the Internet connection time and the variances of network latency.

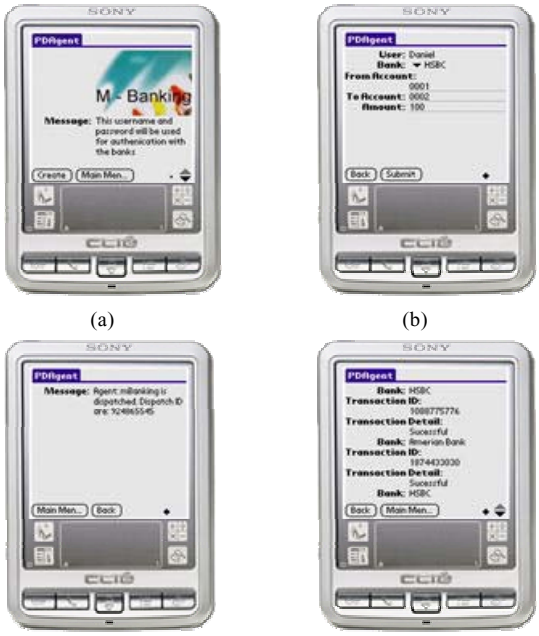

Figure 11 a. E-Banking Login Screen; b. Submit Transaction Information,; c. Dispatched Mobile Agent ID; d. Obtain Transaction Results

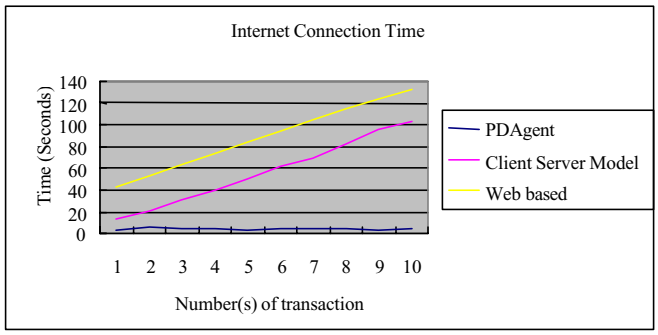

Figure 12. Internet connection times: three different approaches

We can see from Figure 12 that increasing the number of transactions submitted affects the Internet connection time for both the client-server and the web-based approaches. The Internet connection time increases roughly in a constant manner against the increasing number of transactions. This is because the mobile client has to maintain the network connection – beginning when the request for Internet services is made and until the service is completed. In contrast, PDAgent's Internet connection time is not affected by any increase in the number of transactions. This is because the user's requirement for all the transactions is submitted without the need to connect to the network. PDAgent requires an Internet connection only when sending the PI and collecting the result documents from the Gateway. As a result, internet connection time is less than the other two approaches no matter how many transactions it makes.

An experiment was also conducted to compare the variance of latency in wireless network access and its effect on the Internet connection times of both the PDAgent approach and the client-server approach. Four test runs were conducted to determine the variance in network latency. The results are expressed in terms of the effect of variances on transaction completion times. It was observed that network latency in server request/response is the dominant factor affecting transaction completion times, assuming that the time for submitting a transaction is the same for every single trial with same number of transactions. The transaction complete time is calculated as follows:

- *Client-server-platform* - time for submitting transaction information (offline) + time for requesting server (online) + time for obtaining the server response (online)
- *PDAgent* - time for sending "Packed information" (online) + time for downloading result (online)

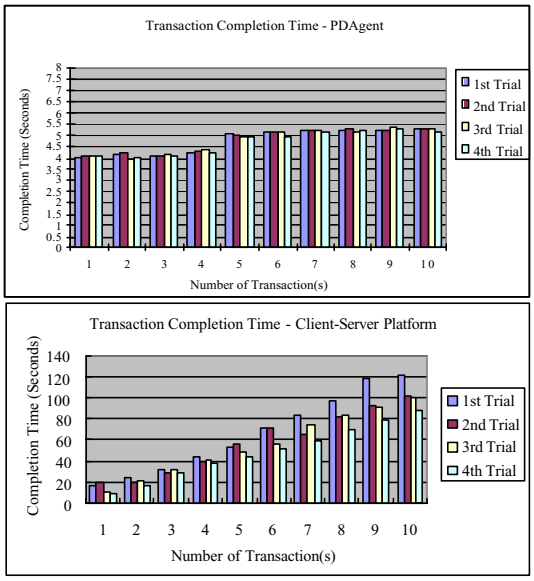

Figure 13. PDAgent and Client-Server Platform: Transaction completion times

As we can observe from Figure 13, variances in network latency in the client-server approach are greater than for PDAgent. Ideally, the completion time should be roughly the same for different trials if the network latency is stable. However, we found that as the number of transaction increased, the completion time of the client-server approach became more unstable.

The completion time of PDAgent was little affected by

network latency, which remained quite stable for all of the trials. This is because PDAgent uses a network connection only when it sends PI and downloads the result documents from the Gateway, while the client-server approach requires that many network connections be made between the client and the server before the Internet service can be completed.

5. Conclusions

In this paper, we have described a lightweight and highly portable platform, called PDAgent, for developing and deploying mobile agent-enabled applications from wireless devices. It has several advantages over existing solutions in providing support for the development of highly portable, efficient, and reliable mobile applications.

PDAgent provides efficient and reliable network services to mobile users with handheld devices without requiring a large amount of resources. Yet, it has low network latency and low network connectivity costs. This is achieved by providing a lightweight platform for running mobile agents that can execute services autonomously on the Internet on behalf of the user. The mobile user can go offline after submitting the execution plan to the Mobile Agent, thereby reducing network connection times. PDAgent also reduces the use of resources within wireless devices by 1) using the Gateway as a middle-tier for deploying and managing mobile agent to the Internet, and 2) compressing the agent code before storing it in the device's database. PDAgent is platform independent. It supports the adoption of different mobile agent systems located in high-end desktops on Internet sites. It is implemented using JAVA and allows developers to build PDAgent applications without being bound to a specific type of mobile devices. Performance of PDAgent has been evaluated and compared with other approaches. The results show that the PDAgent is not only lightweight but very efficient in terms of network connectivity and latency.

The work described in this paper is a part of a larger project in developing a middleware for mobile agent-enabled mobile applications. In our future work, we will further enhance the functionality and performance of the PDAgent platform as well as developing more practical applications, including m-commerce and mobile workflow management. Further experiments for performance evaluation will also be carried out.

6. Acknowledgment

This research is partially supported by the Hong Kong Polytechnic University under PolyU ICRG grant G-YD63 and grant A-PD54.

7. References

[1] J. Cao, X. Feng, J. Lu, and S. K. Das, "Mailbox-based Scheme for Mobile Agent Communications", IEEE Computer, Vol. 35, No. 9, Sept. 2002. pp. 54-60.

[2] J. Cao, X. Wang, J. Wu, "A Mobile Agent Enabled Fully Distributed Mutual Exclusion Algorithm", Proc. 6th IEEE International Conference on Mobile Agents (MA'02), Oct. 2002, Barcelona, Spain. Lecture Notes in Computer Science, Vol. 2535, pp.138-153.

[3] L. Capra, W. Emmerish, and C. Mascolo, "CARISMA: Context-Aware Reflective Middleware System for Mobile Applications", to appear in IEEE Transactions on Software Engineering, Nov. 2003.

[4] P. Dasgupta, N. Narasimhan, L.E. Moser, and P.M. Melliar-Smith, "MAgNET: Mobile Agents for Networked Electronic Trading", IEEE Trans. On Knowledge and Data Engineering, Vol. 11, No. 4, July/Aug. 1999. pp. 509-525.

[5] L. Hagen, M. Breugst, and T. Magedanz, "Impact of Mobile Agent Technology on Mobile Communications System Evolution", IEEE Personal Comm., Aug. 1998. pp.56-69.

[6] J. Jing, A. Helal, and A. Elmagarmid, "Client-Server Computing in Mobile Environments", ACM Computing Surveys, Vol. 31, No. 2, June 1999.

[7] D. Kotz and R.S. Gray, "Mobile Agents and the Future of the Internet", ACM Operating Systems Review, 33(3), Aug. 1999. pp.7-13.

[8] kXML, http://kxml.org

[9] D.B. Lange and M. Oshima, "Programming And Deploying Java Mobile Agents With Aglets", Addison Wesley, 1998.

[10] D. B. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents", Communication of the ACM, Vol. 42, No. 3, March 1999. pp. 88-89.

[11] LEAP – Lightweight Extensible Agent Platform 5th Framework Project Contract: IST-1999-10211 http://leap.crm-paris.com/index.html

[12] Qusay H.Mahmoud, "MobiAgent: A Mobile Agent-based Approach to Wireless Information Systems". Proceedings of the 3rd International Bi-Conference Workshop on Agent-Oriented Information Systems, Montreal, Canada. May, 2001.

[13] C. Mascolo, L. Capra, and W. Emmerich, "Mobile Computing Middleware", Lecture Notes in Computer Science, Vol. 2497. 2002, pp.20-58.

[14] Internet RFC/STD/FYI/BCP Archives – RFC1321 http://www.faqs.org/rfcs/rfc1321.html

[15] P. Bellavista, A. Corradi, and C. Stefanelli, "Mobile Agent Middleware for Mobile Computing", IEEE Computer, March 2001. pp.73-81.

[16] Patik Mihailescu, Walter Binder and Elizabeth A. Kendall, "MAE: A Mobile Agent Platform for Building Wireless M-Commerce Applications", Proc. 8th ECOOP Workwhop on Mobile Object Systems, Málaga, Spain, June 2002.

[17] S. Papastavrou, G. Samaras and E. Pitoura, "Mobile Agents for World Wide Web Distributed Database Access", IEEE Transactions on Knowledge and Data Engineering, 12(5), pp 802-820, 2000

[18] V. A. Pham and A. Karmouch, "Mobile Software Agents: An Overview", IEEE Communications Magazine, July 1998. pp. 26-37.

[19] Java 2 Platform Micro Edition (J2ME) Technology for Creating Mobile Devices White Paper. May 19, 2000.

[20] Mobile Information Device Profile (JSR-37) JCP Specification Java 2 Platform, Micro Edition, 1.0a.