Ubiquitous Mobile Agent System in a P2P-Network

Faruk Bagci, Jan Petzold, Wolfgang Trumler, and Theo Ungerer

University of Augsburg
Institute of Computer Science
Eichleitnerstr. 30, 86159 Augsburg, Germany
{Bagci, Petzold, Trumler, Ungerer}@Informatik.Uni-Augsburg.DE

Abstract. The vision of ubiquitous computing projects a multitude of computers surrounding people. The ubiquitous environment is to support people in their everyday life in an inconspicuous and unobtrusive way. This requires that information of the person and her preferences, liking, and habits are available in the ubiquitous system. Since mobility is in the nature of people the system has to ensure that the information moves with the persons. One possibility is to store a mobile virtual reflection of the person in the ubiquitous environment. This reflection should follow people in the virtual world. The paradigm of the mobile agents ideally matches here to represent mobile objects by mobile virtual objects. This paper introduces a ubiquitous mobile agent system called UbiMAS that is used to accompany and lead persons in an office building with smart doorplates. The agent system is based on an ubiquitous middleware which uses a JXTA peer-to-peer network as communication infrastructure.

1 Introduction

The next generation of computers will be embossed by ubiquitous systems. The computer will disappear behind daily things and will support people in their everyday life. For this it is inevitable to realize the environment using sensors. The situation that arises by realizing and interpreting environmental information is called context. On basis of this context knowledge the ubiquitous system should dynamically adapt to the environment. Moreover the ubiquitous environment should know person's profiles, preferences, liking, and habits.

Since people often change their location the personal information have to be send to the current location. There are two possibilities: either the user carries the relevant information on devices or wearables or the ubiquitous system takes care of storing and sending the information. We want to adopt the second method: the mobile person is always accompanied by a mobile virtual object in the ubiquitous environment.

The paradigm of mobile agents ideally fits here. The mobile agent thereby constitutes a virtual reflection of the user and carries personal information on which basis he can perform different services. The mobile agent interacts with the ubiquitous environment and can request data from sensors in the near surrounding. In this paper we introduce an ubiquitous mobile agent system called UbiMAS that is based on an ubiquitous middleware and uses JXTA [7] peer-to-peer network as communication platform. The agent system is applied in an office building where each door is equipped with an electronic and intelligent doorplate. Mobile agents are used to guide visitors to the searched office.

The next section describes first mobile agents in general and their requirements on ubiquitous systems. Then the ubiquitous mobile agent system UbiMAS on top of the underlying middleware is explained. Section four introduces the evaluation results. The paper ends up with the conclusion.

2 Mobile Agents

Agents in general can be defined as software units with certain autonomy. They perform services by order of an user or other agents. Mobile agents have an additional property: they can autonomously migrate, i.e. they can transfer program code, data and continuation pointer to a remote computer and resume with the program execution. Beside this physical mobility mobile agents have the possibility to communicate with each other in order to exchange information.

Mobile agents are used in electronic markets, for information procurement and internet search, distributed computing and most recently in ubiquitous systems. We will next examine which requirements a mobile agent system has to fulfill to be used in ubiquitous systems.

2.1 Requirements for Ubiquitous Mobile Agent Systems and Related Work

Ubiquitous systems are usually decentralized and distributed systems. User's mobility shouldn't be restricted under no circumstances. Mobile communication media and ad-hoc networking play an important role. A requirement for mobile agent systems is thus the use of both wired and wireless media for communication and migration.

The agent system node organization should be dynamically managed. Since ubiquitous nodes can be very different relating to software and hardware an ubiquitous agent system has to be platform independent.

Also important is that agents should be light-weighted to work not only on powerful PCs or workstations but also on battery-operated and memory-restricted PDAs and wearable computers.

The mobile agent should use different access mechanisms for information procurement, like accessing sensor data as well as using different communication protocols.

One more requirement is context-sensitivity, i.e. agents should recognize contexts and react accordingly. In order to recognize contexts on strange nodes it is recommended to use an uniform information description.

Mobile agents usually carry user- or service-specific information necessary for performing their tasks. This information have to be protected against intrusion to secure user's privacy. Thus security is another important requirement.

Mobile agent systems for the internet like Grasshoppers from IKV GmbH [5] and IBM's Aglet system have very heavy-weighted agents, because the focus of these systems lays on security. Therefore these systems are not suited for ubiquitous applications. In distributed systems light-weighted agents are used to perform distributed tasks. The MESSENGERS agent system [2] is developed at the university of California in Irvine on basis of the WAVE system ([9]). MESSENGERS works on LAN clusters but has the disadvantage that it is limited to maximum of 64 computers. This limitation was compensated by JMessengers [4]. The basic idea of MESSENGERS were ported to Java achieving platform independence. Our experiences with such systems show that they are centralized and rigid concerning the node infrastructure and starting of new agents, i.e. they are less suitable for ubiquitous systems. The FLASH system [6] at the university of Luebeck is another mobile agent system for distributed computing. FLASH is extended for ubiquitous systems and is used for person tracking in department stores [8].

3 UbiMAS - Ubiquitous Mobile Agent System

In order to develop an ubiquitous mobile agent system which fulfills the requirements we need at first a middleware managing the interface and communication issues of the ubiquitous system. This section introduces first a sample application used for evaluation purposes and then the ubiquitous middleware.

3.1 Sample Application

The aim of our application is to transfer simple doorplates into the vision of Smart Doorplates within an office building. The doorplates are able to display current situational information about the office owner, to act instead of the office owner in case of absence, and to direct visitors to the current location of the office owner based on a location-tracking system. The office owner is able to control the information that is displayed on his doorplate. [1] describes different scenarios of doorplate applications in detail.

A first prototype of the Smart Doorplates is installed in our lab. The smallest unit is a Doorplate on a Compaq iPAQ running Windows CE 3.0 and JVM J9 from IBM [3]. As person identification and tracking system we use RFID-tags carried by the users and RFID-readers placed at each door.

3.2 Ubiquitous Middleware

Ubiquitous systems consists of different kinds of devices and services which can communicate over different communication mechanisms. The aim of an ubiquitous middleware is to abstract interfaces and to provide possibilities for an uniform communication. Our current Ubiquitous Middleware [10] is based on an event-driven approach: a peer-to-peer system that decouples the Service Layer from the various networking techniques.

In order to build a non centralized application structure we decided to choose an architecture based on a peer-to-peer communication layer. The peer-to-peer middleware is JXTA implemented in Java. Messages are send over so-called unidirectional, bidirectional or propagate pipes. Unidirectional pipes can send a message in only one direction whereas over bidirectional pipes messages can be send and received by the peers. Propagate pipes have the property that the messages are send to all peers listening to this pipe, i.e. a multicast communication.

The tracking system consisting of RFID-readers at special locations is used to become aware of persons entering and leaving a room which is reflected by a change in the location context.

Figure 1 shows the architecture of the ubiquitous middleware. The figure illustrates the different layers used by the application. The lowest level shows the transport mechanisms used for the messages in the peer-to-peer network. We currently use TCP/IP based communications over Ethernet for the PCs and Bluetooth and WLAN for the iPAQs.

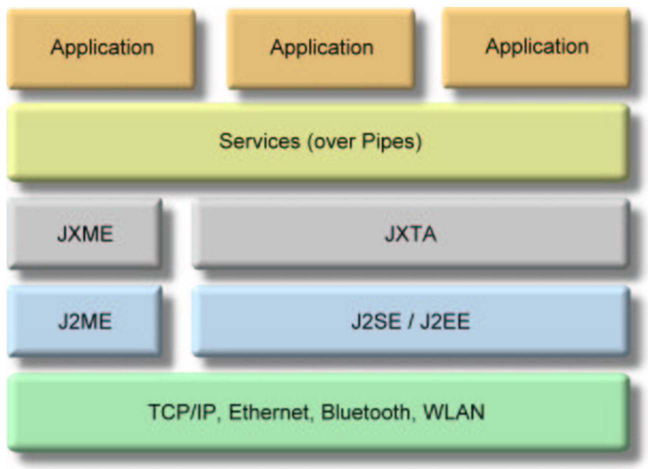

Fig. 1. Middleware Architecture

The second and third level build up the JXTA peer-to-peer network. There is a noticeable difference concerning the JXTA implementations for Java 2 Standard Edition (J2SE) and Java 2 Micro Edition (J2ME). The JXTA J2ME implementation (JXME) consists of only three classes. To hide the implementation dependent parts of JXTA we introduced a Service Layer. The Service Layer contains all the services used by the application.

3.3 UbiMAS Architecture

UbiMAS is a mobile agent system which runs as a service on top of the ubiquitous middleware beside other services like the location tracking service. All services are communicating over propagate pipes. Also the migration of agents is realized over propagate pipes. Figure 2 shows the UbiMAS integration and architecture.

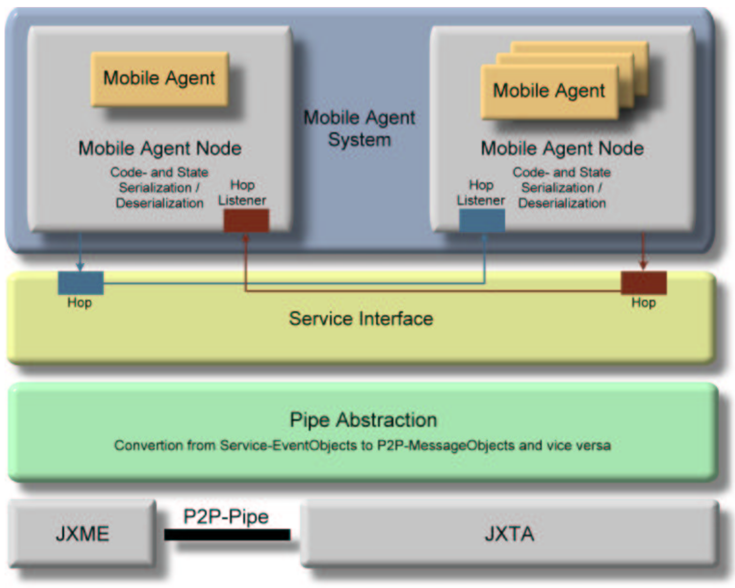
Mobile Agent
Mobile Agent Node
Code- and State Serialization / Deserialization
Hop Listener
Mobile Agent System
Mobile Agent
Mobile Agent Node
Hop Listener
Code- and State Serialization / Deserialization
Hop
Service Interface
Hop
Pipe Abstraction
Convertion from Service-EventObjects to P2P-MessageObjects and vice versa
JXME
P2P-Pipe
JXTA

Fig. 2. UbiMAS Architecture

The JXTA implementation for J2ME and J2SE differ in the methods used to read elements out of the messages and write elements into them. We use a Pipe Abstraction Layer to standardize this access. Furthermore this layer transforms the message elements into Service Message Objects which are sent and received by services.

The Service Communication Layer is used to pass the service messages to the Pipe Abstraction and to inform registered services in case of arriving messages delivered from the Pipe Abstraction. Because of the event driven communication mechanism a service must register itself by the Service Communication Layer to receive messages.

The Agent Node registers itself at the peer-to-peer network and notifies interests on special messages of other services if desired. In order to accompany a person an agent needs information on person's location. These information can be obtained by registering to location service messages of the location tracking system. So all location changes are transferred to the Agent Nodes where the agents reside. In this manner all sensor information can be accessed by the mobile agents if needed.

An Agent Node can manage several agents. It is possible to feed agents into the network from all nodes. If an agent wants to migrate to another node he calls a method and gives the destination as a parameter. Then the Agent Node serializes the agent into a byte stream, packs it into a pipe message, and sends it over the peer network. The message is delivered to the destination peer. Receiving this message the Agent Node at the other end deserializes the agent and checks if the local system knows the received agent class. If the class is missing, the node requests the class from the source node.

4 Evaluation

In our Smart Doorplate application the agents have to hop at least as fast as a human walks from one office door to the next. The evaluation shall show which configurations are suitable by choosing between PDAs or PCs with touch panels as doorplates and different communication technologies. UbiMAS works on PCs and on smaller devices like PDAs. For evaluation issues we developed different test scenarios with different devices and tested the communication over the peer-to-peer network using different communication media.

In the first test scenario we used PCs as peers and measured the ping time (PT) over Ethernet, i.e. the time an agent needs to jump from a node to another and back. We increased the size of data carried by the mobile agent. Then we decided to measure the PT using PC and PDA as devices running UbiMAS. The communication was realized once over Bluetooth and then over wireless LAN (WLAN). As last scenario we used PDAs as UbiMAS nodes and let agents move from one PDA on another. Also here we measured the PT over Bluetooth and WLAN.

Since migration of agents for the first time triggers sending the agent class to the new location we performed the five scenarios two times: with class transfer and without class transfer. Figure 3 illustrates the PT in seconds over the size of data carried by the agent with regards to a first agent hop triggering a class transfer. Figure 4 shows the PT in seconds regarding agent hops without class transfer.

Increasing the size results in increased PTs as expected. Over Bluetooth we got the worst results. Agent hops over Bluetooth are from a size of 100 KBytes unacceptable low in speed. Also for lower sizes it needs 2-10 seconds over Bluetooth. Besides the low communication speed of Bluetooth a further reason is that J2ME currently needs a relay station that send messages to the PDAs sequentially instead of using multicast. Agent hops over LAN and WLAN provide both very good results in all device combinations. For agents lower than 100 KBytes the values lie on range of milliseconds.

In order to evaluate the network load of the peer-to-peer system we performed another test where we built up a network of 12 peers. Each peer was a PC with the same hardware and software configuration. On each peer there was one agent

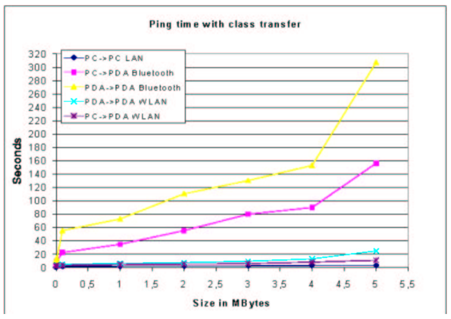

Fig. 3.

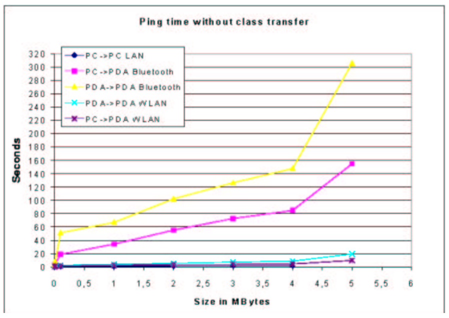

Fig. 4.

node which started a number of agents. The agents chose randomly another node and performed a hop to it and back again. We measured the average ping time of the mobile agents increasing the number of agents in the network. Since the peer-to-peer system sends peer messages as UDP packets there is no guarantee that the messages will be received by the destination node. So we measured also the agent loss by increasing the agent number. Figure 5 illustrates the PT results and figure 6 the agent loss.

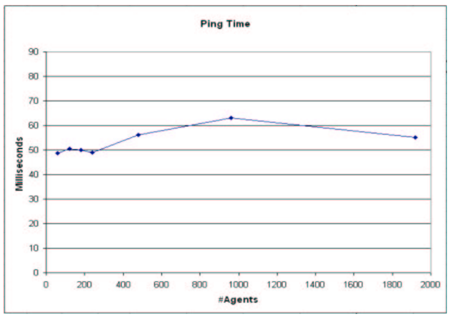

Fig. 5.

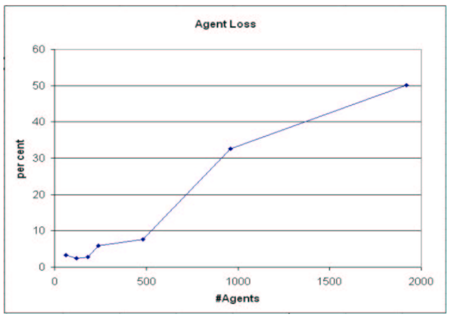

Fig. 6.

The results show that by increased agent traffic the loss of agents increases dramatically. The number of agents has minimal influences to the ping time. In order to get reliable message transport it only needs to use TCP as transmission protocol. The JXTA team is working on extensions for reliable protocols, so this will not be a problem in the next future. The evaluations show that the UbiMAS agent system is suitable for our Smart Doorplate application provided that LAN or WLAN connected PCs or PDAs are used as doorplates.

5 Conclusion

Ubiquitous systems support people in their everyday life by performing services and using devices independently in behalf of the user. For this it needs to reflect personal information and context information of the environment into a virtual structure. Mobile agents can react by means of this reflection.

We described here an ubiquitous agent system (UbiMAS) on basis of an event driven ubiquitous middleware. UbiMAS uses a peer-to-peer network for the migration of agents. Since sensor information is send in the same way over pipes agents can register for sensor messages and receive desired information. This means that agents are becoming context sensitive.

UbiMAS is fast enough for delivering messages over LAN and WLAN. Bluetooth is for heavyweighted agents unsuitable, but for smaller sizes still acceptable. For increasing number of messages sent over the peer-to-peer network a reliable transmission protocol is needed. The JXTA project is working on extensions for this purpose.

Our own future work focuses on agent transfer algorithms after establishing a larger Smart Doorplate scenario within the institute building by combining several floor-based peer-to-peer networks. Mobile agents will be used to accompany people through the building and to search for absent office owners over different peer-to-peer networks.

References

1. Faruk Bagci, Jan Petzold, Wolfgang Trumler, and Theo Ungerer. Smart doorplate. In *The First International Conference on Appliance Design (1AD), Bristol, UK*, May 2003.
2. Lubomir F. Bic, Munehiro Fukuda, and Michael B. Dillencourt. Distributed computing using autonomous objects. In *IEEE Computer*, pages 55–61, August 1996.
3. IBM Corporation. IBM Web-Sphere Studio Device Developer. http://www.ibm.com/software/pervasive/products/wsdd/, October 2002.
4. M. Gmelin, J. Kreuzinger, M. Pfeffer, and Th. Ungerer. Agent-based distributed computing with jmessengers. In *I2CS Innovative Internet Computing Systems*, pages 131–145, Feb. 2001.
5. Grasshopper. http://www.grasshopper.de, 2003.
6. C. Grewe, W. Oberler, and H. Pals. Programmierumgebung fuer mobile agenten in hererogenen workstation-clustern. In *Workshops zur Architektur von Rechensystemen ARCS*, pages 163–172. VDE-Verlag, 1999.
7. Project JXTA. http://www.jxta.org, November 2002.
8. Holger Pals, Andreas C. Dring, and Erik Maehle. Usage pattern and their requirement of bluetooth equipped gadgets in everyday life. In *Fachtagung der GI/ITG, Arbeitsplatzcomputer APC*, pages 11–18. VDE-Verlag, 2001.
9. Peter S. Sapaty and Peter M. Borst. An overview of the wave language and system for distributed processing of open networks. In *Tech. Report.* University of Surrey, UK, 1994.
10. Wolfgang Trumler, Faruk Bagci, Jan Petzold, and Theo Ungerer. Smart doorplates - toward an autonomic computing system. In *5th Annual International Workshop on Active Middleware Services, Seattle, WA*, June 2003.