AgentServices: Agent Services Framework for Handheld Devices

Ali Muhammad[1], Aslam Muhammad[2], Martinez-Enriquez A. M.[3]

[1,2]Department of CS & E., UET., Lahore, Pakistan ([1]ali@kics.edu.pk, [2]maslam@uet.edu.pk)

[3]Department of CS, CINVESTAV-IPN, Mexico (ammartin@cinvestav.mx)

***Abstract*– Computers and computing have become versatile and ubiquitous in the form of handheld devices. This computation power can be utilized in agent services to solve users' daily life problems like hotel reservation. We propose an agent oriented services framework (AgentServices) using JADE as an agent system. AgentServices can be seen as a colony of multi mobile agents running on handheld devices. It creates and consumes services, exchange code with other agents and allows streaming of services offered by mobile agents. We harvest mobile agent technology in providing the user with comfort and ease to carry daily activities using just a handheld device. As case studies, we use the hotel reservation service support and a disaster management service.**

Keywords: Mobile Agents, Service Agents, Agent Communication and Assistant Agent.

I. INTRODUCTION

Computer applications have become ubiquitous in the form of smart phones, PDA's, and laptops. The computing power of a typical handheld device is many times greater than the most powerful desktop computer which existed almost two decades ago. The computing power of these handheld devices is bound to increase at an exponential rate due to Moore's law [9]. Cellular phone is the most common form of handheld devices that are widely available.

In the future, the use of desktop computers would become obsolete as ubiquitous computing will come to dominate [8]. Therefore, agents would increase as well as people depend more on handheld device for tasks they used to do on desktop computers. Intelligent agents are the future trend of the Internet [7]. But, before we move forward, we briefly state what an agent is. Though there is known controversy on agent definition, but all agree that agents are capable of autonomous action in an environment where they reside [2]. Discussion about what an agent is and how it differs from programs is given in [1].

As computer become more and more smart with higher computation, Internet data rates, the trend towards handheld computing would become more manifested in PDA, mobile phones, smart watches and any electronic device (paper, pen or clothes) [9]. However, security is a serious concern for mobile agents, as none fully understands the influence of mobile agents. In fact, mobile agent's implementations have been quite few with limited success [10].

Some examples of daily life problems that can be tackled by agents and our proposed framework include showing only the required support to users, automatic room reservation in a hotel in the vicinity of the area, knowing in advance about failures of some ATM machine before making an effort to use it and finally streaming of live traffic video of a remote location.

Thus, we propose a colony of collaborative mobile agents running on handheld devices, named Agent-Services. The agents are able to communicate each other and establish negotiation based on users requirements, giving rise to a mobile ad hoc network [18]. An agent has two layers: - reactive that is responsible for resources streaming data; and - logical layer where decisions take place. The main contributions of this paper are: i) the design and implementation of an environment for agent oriented services, ii) the streaming of input/output devices data using agent messages, and iii) the support for full agent code migration. Our solution can be seen as an intelligent assistant for dynamically changing computational services, wherever user goes, whatever he needs [5].

The rest of this paper is organized as follows. Section 2 briefly reviews some of the existing attempts and prototypes for providing services by agents. Section 3 explains our approach in abstract terms and then gradually proceeds to details such as agent communication, mechanism, security, among others. Section 4 and 5 presents the case study of "Hotel reservation" and a disaster management service called "Register It" to demonstrate the practical aspect of research. Finally, Section 6 concludes and gives glimpse of future work.

II. RELATED WORK

There have been number of systems providing agent services and systems based on agents that provide versatile services to users.

EasyLife [3] provides location aware agent based services to facilitate users with daily needs. Three services for which a working prototype has been developed, is weather, restaurant, and shopping. JADE is used for implementation, although there is a concept of agent but there is no ability to socially interact and negotiation with other agents.

OKEANOS [4] is a distributed service based middleware for agents that allow access to services and communication using KQML [1]. Agent behavior is specified by symbolic logic rules and facts. OKEANOS provides a high layer interface for symbolic applications.

Our approach differs from OKEANOS as it provides agent mobility and ability to provide streaming.

Siri [25] is a personal intelligent assistant on IPhone that allows users to make verbal requests. Siri processes these requests and acts upon like scheduling appointments, buying movie tickets, and making dinner reservations. The Siri project uses the Net to locate services and reply to user request. Siri does not involve social interaction with other agents to satisfy the user request.

Microsoft Automated Service Agent [23] is an application where users can interact in natural-language. Useful and accurate answers in real time are provided. This approach is limited since it can only give predefined answers which does not involve human. So, users may formulate questions that are not considered in the database, so any answer can be returned.

AgentServices differs from these existing approaches in providing the ability to socially interact and negotiate with other agents and the ability to create services on the fly. Also in most agent systems there is rarely an emphasis or estimation of security risks

III. AGENT SERVICES

AgentServices colony follows standards of Foundation for Intelligent Agents (FIPA) specifications [17]. Agents can stream data over FIPA ACL messages. Java Agent Development Framework (JADE) [16] is an open source, fully FIPA complaint middleware for multi-agent systems, a tiny version of JADE called JADE leap is used for implementing and testing the AgentServices system as the leap version can run on handheld devices with J2ME support. In addition, negotiate and exchange of code is possible in AgentServices.

Agents temporary establish a network connection, exchange messages, and disconnect. Mobile nodes working in this way are called mobile ad hoc network (MANETS) [18]. The transmission mechanism is transparent for agents. In abstract terms, each agent percepts from the environment and performs an action. The perception either comes from the I/O of the handheld device or though a message received. So the agent environment is a set containing two elements.

$$E=\{IO, M\}$$

Where IO: Input/output device and M: message.
The see function of the agent takes the set E as input and generate perception as output (see Figure 1).

$$See: E \rightarrow Per$$

Finally, the action or output of the agent is also a set.

$$Action: Per \rightarrow Ac$$

Where Per means perception and Ac means Action.

Java ME compatible [26] handheld device is the basic component of AgentServices, required for running. Communication facility based on TCP/IP is provided for working in harmony.

As we can observe in Fig 2, a handheld device has two running mobile agents in the agent container system. The container manages the agents and provides them services for freely in/out movement from the handheld device system.

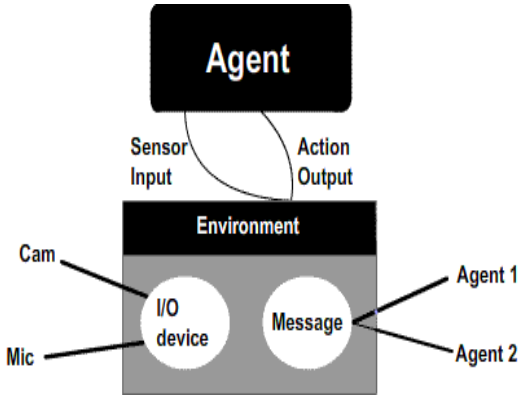

Fig. 1. Agent taking input from the environment and making changes to the environment by means of actions.

Agents are designed to be peer to peer flexible, i.e. depending on the situation, they can either act as: i) server agent or ii) client agent.

Client agent is a thin agent and runs on the handheld device; the basic purpose of the client agent is to locate services and exchange messages or stream data with the server agents. Server agents are agents which provide a specific service to the client agents and also process and negotiate with the client agents. In the subsequent section, we discuss how agents represent beliefs and how rules can be specified for making decisions.

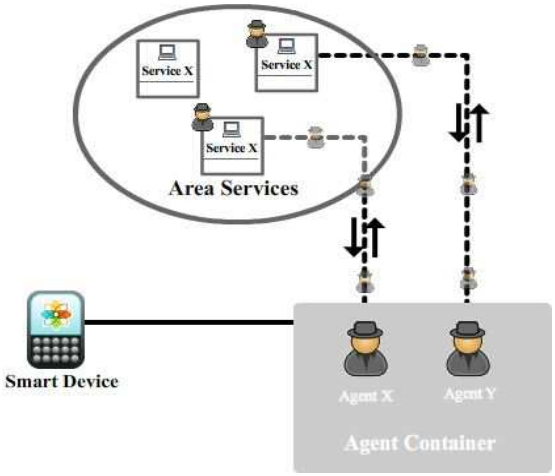

Fig.2. AgentServices basic architecture

A. Knowledge Base

The knowledge base contains a set of beliefs in the form of facts and rules of the agent. In AgentServices, facts and rules are specified using JESS [15], a popular rules engine for JAVA. JESS is used to reason and execute rules taking into account the facts. A rule in JESS has the pattern:

IF <condition> THEN <action> -ELSE <action> statement.

JESS has high interoperability since rules and facts can be read and sent to other agents using XML. For instance, when a

user needs cash from ATM, the agent is required to find an ATM near, the following xml schema for a rule is used:

```
(defrule NeedsCash
   "If  cash is needed from ATM, deliver the cash."
   (ATM-has-cash)
   (ATM-in-order)
   => (deliver-cash))
```

In JESS, the input is the set of facts and the output is the modification of these facts. Some of these output values are associated with actions that agents can perform. These values must be prefixed by the character asterisk.

For example, when rule output is "*alert ("Shopping successful")". This output is automatically parsed by the system and the handheld device launches an alert message on the screen. Few other actions that can be performed are given below in Table 1.

Table 1. Special output values

Message (X,Y)	X Send a message to the agent named Y
Stream (D,Y)	Stream a device D to the agent named Y

More discussion on stream message is given in Section 3.4.

B. Agent Registration

In AgentServices, services can be created on the fly after a first time registration with the Agent Management system of the JADE which resides on the main container [16]. Only server agents need to register as they are the one providing a service. Client agents can polymorph to a server agent by requesting the application to create a new service. AgentServices interface encapsulate the details of registering with the main container of the JADE system. It knows the location and commands to interact with the JADE system. Once the agent has been registered with the JADE main container, other client agents can discover and exchange messages with this agent.

The only distinction between server and client agent is that the server agent offers a service and is registered with the JADE. The communication mechanism between client and server agent remains the same.

C. Agent Communication

FIPA ACL [17] is used for exchange of message. ACL is similar to Knowledge Query and Manipulation Language (KQML) [22]. ACL has 21 performative of speech acts.

The content part of FIPA ACL is used to send content in Darpa Markup language (DAML) [19], developed and funded by DARPA. It is based on XML and RDF. The DAML is an effort towards the semantic Web which is yet to be articulated.

DAML also contains a basic inference mechanism for reasoning from the available data. An example of the DAML encoded FIPA ACL inform message is given in [19].

User of client agent builds a profile specifying to the agent what he is looking for. The profile must conform to the XML schema of the service. There is one profile associated with each service. Each profile defines ontology of the domain that is expected by the service. When a profile for a particular service doesn't exist, it can be loaded dynamically from the service by a FIPA ACL 'request' message with content 'Service'. The data in the profile is used during communication with the service and is used to state user preferences.

D. Agent Streaming

The logical layer is responsible for agent communication, negotiation, and smooth transportation of agents among different systems, whereas reactive layer is used for streaming data of a particular device. Streaming was implemented using a UDP protocol [13] which is less reliable but more suitable and faster for streaming voice and video over the network. There are basically two ways in which this streaming can be feasible.

1) Direct streaming from one to one agent. It is faster and simple to implement, since it only supports one to one streaming between two agents, it lacks of flexibility.
2) Indirect streaming from one agent to another. Data passes through the main container, then main container forwards data to relevant agent. It has the advantage of streaming data to a large number of agents.

Streaming is initiated by FIPA ACL 'request' message with content 'action: stream'. The agent waits for response. The destination agent checks if the device is available and the agent who is requesting has good reputation. The destination agent may reply with 'refuse' or 'agree' message. When the destination agent replies 'agree', a final 'request' message can be formulated and the content "action: stream-confirm" is sent to confirm the stream. After streams confirm message has been received streaming of the device data starts over the network using UDP protocol.

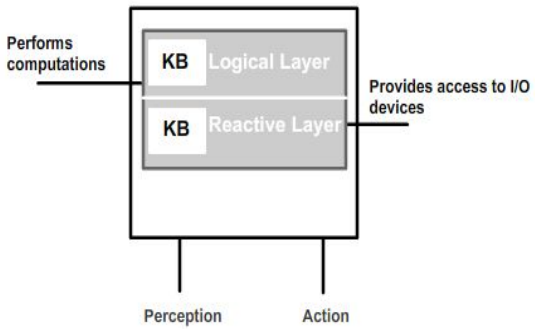

Fig. 3. AgentServices Layers

Next we discuss how agents can discover other services using JADE as an agent toolkit.

E. Agent Discovery

According to FIPA, an agent may discover services of other agents by making use of Directory Facilitator agent (DF) [16] and agent management system, which provides a unique identifier (name) to each agent. For instance, Agent X inquires DF agent about agent services near at most 5km. Once a service has been chosen, the source and service agents, begins normal communication with each other as shown Figure 4.

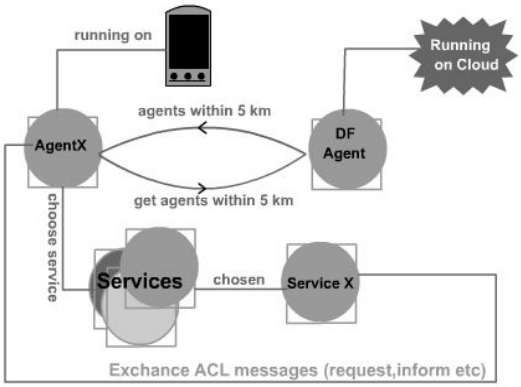

Fig. 4. Agent Services discovery

F. Agent Migration

A very core concept in mobile agents is code mobility as discussed in [6, 7]. Our agent migration is based on a modified form of [20] which proposes agent migration using ACL message based on FIPA agent migration protocol. The goal of transporting agents embedded in FIPA ACL message is very clear as it is independent of agent platform and particular programming language. In our implementation, the entire agent migration is transparent to the platform, and main container does not know whether the agent has been transported.

This works by sending the compact version of the code or executable program like a Java jar file embedded in the content of FIPA ACL request message. The mechanism for the agent migration request is given below.

- First the agent must request other agent to whom it wants to migrate or execute some code by using the FIPA ACL request message with 'Transfer' action.
- When the other agent receives the request message, it decides to accept (Agree) or reject (Refuse) the migration by sending respective message.
- If the agent agrees for the migration, the immediate next message is the code of the agent. The code is using the request performative with action 'code'.
- The receiver agent informs the sender agent using 'inform' performative of ACL, when the agent has successfully been migrated.

Now, we define the semantics for the agent migration in content message. To specify the action in the content message we make use of simple XML message. For instance, to specify the action transfer we can use 'action: transfer'.

The request must also specify an additional parameter 'language' or non-FIPA parameter 'extension' to inform the destination agent about the run time environment that agent is expecting.

Both extension and language can be specified in the agent migration request. However, at least one is compulsory. For instance, when agent is sending a flash file, a complete request message may look like below:

```
(request
  :sender (agent-identifier :name i)
  :receiver (set (agent-identifier :name j))
  :content (Action:transfer)
  :x-extension swf)
```

Swf is an extension for flash files, and swf files are in binary, therefore, the content of the message is in binary. To denote that an attribute is not a standard FIPA, it is prefixed with x-, like x-extension.

Such a dynamic transport of the agent is associated with risks of malware and fraud [11], so agent security is a critical issue which is the topic of next section.

G. Agent Security

There are security risks associated with mobile agents such as malicious code execution, altering the critical files on the file system or sending unwanted messages by using the victim network. Some issues are discussed in detail in [12]. Agents can be put into two categories based on security model of IBM aglet framework [14].

- Trusted agents
- Untrusted agents

The agent security manager checks whether the agent should be allowed file system and other resources and lies entirely on the host. Mobile host may ask and evaluate the response from other known agents to determine whether to allow/disallow access to remote agent.

Each agent is ranked from 0 to M-1. Higher value indicates more trust. The trust value varies from 0 to 10, the maximum is 10. Magnitude is directly proportional to the number of trusting agents. When trust level is 4, means there are four agents which trust the agent we are trying to communicate.

On the other hand, the host agent can use the centralized approach to security. In centralized approach, all the agents refer to a central agent (server) passing to it the identifier of the agent for which it wants to know the rating. The next section presents a real world problem and how our system can be used to solve problems.

IV. USE CASE: HOTEL RESERVATION

This section discusses a real world problem in which a user wandering in a foreign city needs to find a cheap hotel to live or find an Indian food restaurant. It is assumed that hotels and restaurants have created and registered their services at AgentServices system.

The users are unaware of visited city, but the services of hotels or restaurants are needed to achieve goal. Now they have two options.

1) Wander around streets and continuously ask people around the area for services.
2) Use Internet with specific keywords to find services.

Unfortunately, both methods have their cons. The first approach is tiring and doesn't always lead good results for instance; user may obtain misguiding, wrong or unverified information instead.

The second approach lies on continuously searching for required service, websites maybe outdated and in our experience many cheap hotel and restaurant services don't

even have a Web page. Even if a web page exists, it may not guide user towards what exactly is looking for.

The solution is to find all services related to hotel/restaurant in the area. This can be done by obtaining a list of all the services on the mobile device by opening the MobileServices application. This application will connect to our server passing it down user profile and location information. The user profile will contain a search string such as hotel for the service and the maximum distance which user has to travel to use that service. This feature requires GPS on the agent device. The result will be a list of services matching the user profile and user location information. The user profile specifies what the user is looking for and is given below in XML for the hotel service. A typical hotel schema is shown in Fig 5.

```
<xs:element name="reservation">
  <xs:complexType>
   <xs:element name=”From” type="xs:date”/>
   <xs:element name=”To type="xs:date”/>
 <xs:choice>
     <xs:element name="room"/>
     <xs:element name=”room+car"/>
   </xs:choice>
  </xs:complexType>
</xs:element>
```

Fig. 5. XML schema for hotel service

Since purpose of this schema is to get hotel reservation information, therefore the top element is “Reservation” which is of complex type and contains other attributes. The “choice” is a special element which only allows one attribute to be selected from its child attributes. In this case, user can either select single room or a room and car package.

This XML schema is loaded by AgentServices on first interaction with the service. AgentServices provides an interface to fill these schema data and sends the information back to hotel service.

The hotel service agent receives the message and parse it using JESS and check for any rules fired. If some rules are fired a reply in DAML ACL format is sent to client agent named “Agent1”.

```
<acldaml:Inform >
<acldaml:sender>Serivce1 </acldaml:sender>
<acldaml:receiver>Agent1 </acldaml:receiver>
<acldaml: language value = FIPA-DAML />
<acldaml: content>
<fipa:availabe>Yes </fipa:available>
<fipa:pricing>
<fipa:standard>
<fipa:id>1</fipa:id>
<fipa:price>10>
<fipa:unit>$</unit>
<fipa:facilities>
<fipa:airconditioner>no</fipa:airconditioner>
<fipa:tv>yes</fipa:tv>
</fipa:facilities>
<fipa:simulation> Yes </fipa:simulation>
</fipa:standard>
<fipa:luxury>
<fipa:id>2</fipa:id>
<fipa:price>20>
<fipa:unit>$</unit>
<fipa:facilities>
<fipa:airconditioner>yes</fipa:airconditioner>
<fipa:tv>yes</fipa:tv>
</fipa:facilities>
</fipa:standard>
<fipa:simulation> Yes </fipa:simulation>
 </fipa:pricing>
</acldaml:content>
 value = FIPA-DAML />
<acldaml:content-length value = 12 />
<acldaml:ontology value =Hotel Ontology />
</acldaml:inform >
```

Fig. 6. Sample Reply from hotel service

The reply message in XML schema (Fig.6) from hotel services indicates rooms available based on user requirements are ‘standard’ and ‘luxury’. The price for a standard room with ‘TV’ facility is 10$ and the price for luxury room with ‘TV’ and ‘AC’ is 20$. They are identified by a unique id. User can select one of these options. The message also states that simulation is available for both of these room types. The simulation means ‘agent 1’ can request the simulation from the hotel service which is ‘service 1’.The simulation may include sending a photo, video or a program which allows user to experience and become more familiar with the object of interest in this case a room before materializing their final decision. The simulation can be requested from the service by sending the ‘request’ message to the service by specifying the unique id of the resource for which simulation is request.

Once the service agent receives this message, it parses and adds the message to its KB and checks fired rules. When a simulation rules is fired, the simulation in binary ‘jar’ format of JAVA or any other binary format is sent to ‘agent 1’. A typical simulation message may look like that shown in Fig. 7.

```
<acldaml:execute >
<acldaml:sender>Service1 </acldaml:sender>
<acldaml:receiver>Agent 1</acldaml:receiver>
<acldaml: language value = FIPA-DAML />
<acldaml: content>
<fipa:simulation>
<fipa:id>2</fipa:id>
<fipa:binary>
[binary data]
</fipa:binary>
</fipa:simulation>
</acldaml: content>
</acldaml:execute>
```

Fig.7. Simulation message

Once the message is received, the agent ‘agent1’ compares the Id of the execute message to check whether it has requested the simulation for such Id, ‘agent1’ executes the binary code if and only if

1) Id matches with beliefs and knowledge base of the agent.

2) The service is reliable and no known complain is available for it.
3) The agent still has a desire to obtain the simulation.

Statement 1 is easy to check, for 2 we have discussed two ways to check the reliability of the agent that is either ask fellow agent about the service reputation or ask a centralized server about the service reputation.

The statement 3 is also not hard to check, when immediately the service agent replies and within a specified time. It is very likely that agent still has a desire to see the simulation. On the other hand if agent has already seen the simulation of some other service or user explicitly cancels the reservation, the agent may not have a desire anymore to see the simulation.

V. USE CASE: DISASTER MANAGEMENT SERVICE (REGISTER-IT)

Natural disasters like earthquake, floods and tornados have become extremely common due to changing climate, human activities and global warming. On 11th March 2011 Japan had its worst crisis in 65 years in the form of 8.9 magnitude earthquake followed by tsunami. Although natural disasters can't be prevented, they can be managed using technology such as intelligent agents. In these natural disasters; it is very important and crucial for families of the victims to swiftly know about the victim's safety and situation of the area affected by natural disaster.

One way for government to manage the situation is to create a website with instructions and a guide to what to do in the scene and air information about the affected areas and people. This approach is rather time consuming, and it is often the case that victims who have been affected by the natural disaster like earthquake may not have access to the TV due to breakage or no power. The probability that they still have access to their handheld device like cell phone is most likely. Although the complete elaboration of this use case is outside the scope of the paper, we briefly state how we can tackle this problem using AgentServices.

Government of the affected area can immediately create a number of agent services. The core purpose of some services is to guide the users and allow them to get the crucial information like emergency numbers.

One of the services named "Register Me" can help a victim identify themselves and register their name, location and problem with the centralized database. Additionally they can also upload a picture or brief video of the situation to the database. The information can then be used for data mining, providng up-to-date news of the affected area and population.

A typical scenario is a user affected by the natural disaster like earthquake locates the emergency services created on the fly by the government and registers with the service. The XML schema for "Register Me" service is given below.

```
<xs:element name="Register Me Service">
 <xs:complexType>
 <xs:element name=”Name:” type="xs:String”/>
 <xs:element name=”Location:”type="xs:String”/>
 <xs:element name="Problem:" type="xs:String"/>
 <xs:choice name=”Picture,Video,Choice”>
  <xs:element name="picture"/>
  <xs:element name=”Video "/>
  <xs:element name="picture+video"/>
   </xs:choice>
</xs:complexType>
</xs:element>
```

Fig 8: XML schema for "register me service".

This XML schema allows users to share their name, location and problem with the service. Also it gives them choice to select picture, video or both for uploading to the service database. This information goes to the service agent who registers the information in its database and checks for any rules fired. If the user has selected for instance a picture element, the following rule can be fired and evaluated using JESS.

IF <message.picture=true> THEN <ask For Picture>.

When 'ask for picture' rule is fired the server agent can send a simulation request message to the client giving it ability to upload a picture to the server database. At this point, client receives the simulation request which is interpreted again by the JESS rule engine and an interface is provided to upload the picture. Complete mechanism and code for this mechanism is being intentionally not given due to the space requirement. At this point user of the client agent can either upload the picture or cancel the request. If many victims use this service, it can greatly aid government and families of the victims to plan things and thus manage the disaster using the technology.

VI. CONCLUSION

The main contribution of our approach is a dynamic creation and consumption of agent services, and the ability to transfer and execute remote code, as well as the streaming of devices data. The significance focus on security was explained. The two use cases demonstrate few problems that can be tackled using our approach out of many versatile problems that this framework is capable of solving. AgentServices may be viewed as a form of services oriented architecture in which intelligent agents provide useful information services. This framework is built on top of FIPA ACL and DAML which are quite popular.

Future work on this approach includes more feasibility and security of the proposed framework.

REFERENCES

[1] Franklin, S., Graesser, A. 'Is it an agent, or just a program?' Proceedings Third International Workshop on Agent Theories, Architectures and Languages, Budapest,Hungary, 193-206 (1996)

[2] M. Wooldridge, 'An Introduction to Multiagent Systems', Wiley, New York, (2002)

[3] J. M. Shen, M. J. O'Grady and G. M. P. O'Hare EasyLife: 'A Location-Aware Service Oriented Mobile Information System' Knowledge-Based Intelligent Information and Engineering

Systems Lecture Notes in Computer Science, Volume 5177/2008, 229-236 (2008)
[4] R. D. Schimakat, W. Blochinger, and C. Sinz, 'A service-based agent framework for distributed symbolic computation,' in Proc. 8th Int. Conf. High Performance Computing Networking, Amsterdam, The Nether-lands, pp. 644–656 (2000)
[5] Martin Griss, Reed Letsinger, Dick Cowan, Craig Sayers, Michael VanHilst, and Robert Kessler., 'CoolAgent: Intelligent Digital Assistants for Mobile Professionals - Phase 1 Retrospective', HP Laboratories Report HPL-2002-55(R), (Jul 2002)
[6] D. Kotz, R. Gray, and D. Rus 'Future directions for mobile-agent research'. IEEE Distrib. Syst. [Online] Available: http://dsonline.computer.org/0208/f/kot_print.htm (2002, Aug).
[7] D. Kotz and R. S. Gray, 'Mobile agents and the future of the internet' IEEE Trans. Automr. Contc, vol. AC-28, pp. 1081-1090, (Dec 1983).
[8] M.Weiser. Hot topic: 'Ubiquitous computing'. IEEE Computer, 26(10):71–72, (Oct 1993)
[9] Mattern, F.: 'Wireless future: Ubiquitous' Congress 2004, Munich, Germany. (2004)
[10] D. Milojicic. Trend Wars: 'Mobile agent applications'. A review article in IEEE Concurrency, pp. 80-90 (July-Sep 1999)
[11] Greenberg,M.S., Byington, J.C., Harper, D.G.:'Mobile agents and security'. IEEE Commun. Mag. 36(7), 76–85 (1998)
[12] Borselius, N 'Mobile agent security', Electronics and Communication Engineering Journal, IEE Press, Vol. 14, No. 5, pp 211-218.(2002)
[13] J. Postel, RFC 768: 'User Datagram protocol', (August 28, 1980)
[14] D.B. Lange, 'Java Aglet Application Programming Interface' white paper, 2nd draft, IBM Tokyo Research Laboratory (1997)
[15] Ernest Friedman-Hill The Rule Engine for the JavaTM Platform http://www.jessrules.com/ (1995)
[16] Bellifemine, F., Poggi, A., Rimassa, G.: 'Jade, A FIPA-compliant Agent Framework'.4th International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology, (1999)
[17] Foundation for Intelligent Physical Agents. Specifications. Available from http://www.fipa.org (1997).
[18] Jinbae Park, Hyunsang Youn, Eunseok Lee 'A Mobile Agent Platform for Supporting Ad-hoc Network Environment' (2007)
[19] Youyong Zou, Tim Finin, Yun Peng, Anupam Joshi, and Scott Cost 'Agent Communication in DAML World'. In Innovative Concepts for Agent-Based Systems Lecture Notes in Computer Science, Volume 2564/2003, 347-354 (2003)
[20] Ametller, S. Robles, and J. Borrell, 'Agent Migration over FIPA ACL Messages', in Mobile Agents for Telecommunication Applications (MATA), ser. Lecture Notes in Computer Science, vol. 2881. Springer Verlag, pp. 210-219. (Oct 2003)
[21] P. O'Brien and R. Nicol, 'FIPA—Towards a Standard for Software Agents' BT Technology J., vol. 16, no. 3, pp. 51-59, (July 1998).
[22] T. Finin, Y. Labrou, and J. Mayfield. 'KQML as an agent communication language'. In J. Bradshaw, editor, Software Agents. MIT Press, Cambridge, (1997).
[23] Microsoft White Paper Comparing Microsoft Automated Service agent (ASA) solutions to browser and search based self service solutions [2007]
[24] S. Liu, P. Küngas, M. Matskin, 'Agent-Based Web Service Composition with JADE and JXTA', Proceedings of the 2006 International Conference on Semantic Web and Web Services, SWWS 2006, Las Vegas, USA, June 26-29, (2006).
[25] Siri- Your Virtual Personal Assistant http://siri.com/.
[26] J. W. Muchow. Core J2ME Technology and MIDP. Prentice Hall, (2001).