Communication and migration of an embeddable mobile agent platform supporting runtime code mobility

Mohamed BAHAJ
Department of Mathematics and Computer Science,
Université Hassan 1er, FSTS, LABO LITEN
Settat, Morocco

Khaoula ADDAKIRI
Department of Mathematics and Computer Science,
Université Hassan 1er, FSTS, LABO LITEN
Settat, Morocco

Noreddine GHERABI
Department of Mathematics and Computer Science,
Université Hassan 1er, FSTS, LABO LITEN
Settat, Morocco

***Abstract*—In this paper we present the design and the implementation of Mobile-C, an IEEE Foundation for Intelligent Physical Agents (FIPA) compliant agent platform for mobile C/C++ agents. Such compliance ensures the interoperability between a Mobile-C agent and other agents from heterogeneous FIPA compliant mobile agent platforms. Also, the Mobile-C library was designed to support synchronization in order to protect shared resources and provide a way of deterministically timing the execution of mobile agents and threads. The new contribution of this work is to combine the mechanisms of agent migration and their synchronization.**

Keywords- Mobile agent; Mobile agent platform;Agent communication.

I. INTRODUCTION

Mobile agent is a design program with a persistent identity which migrates in the network and communicates with its environment and other agent [1]. It has been applied to a variety of distributed applications, such as manufacturing [2-4], electronic commerce [5-7], network management [8, 9], health care [10], and entertainment [11]. During the execution, mobile agents can be dynamically created and sent to the destination systems to perform tasks with the up-to-date code. The mobility allows mobile agent to migrate from one host to another in the network and provides a several applications with flexibility and adaptability that are both able to satisfy the requirement and condition in a distributed environment.

The importance of Mobile agent technology in the design of distributed applications on the web has led the OMG (Object Management Group) to define the specifications MASIF (Mobile Agent System Interoperability Facility) for interoperability between different systems to mobile agents. Another effort is made by FIPA (Foundation for Intelligent Physical Agents) to specify the architecture and also the semantics of communication between mobile agents.

The majority of mobile agent platforms in use are Java-oriented. Multiple mobile agent platforms supporting Java mobile agent code include Mole [12], Aglets [13], Concordia [14], JADE [15], and Agents [16]. Using a standard language like the mobile agent code language that provides both high-level and low-level functionalities is a good choice to treat with the large number of distributed applications. The choice of C/C++ is a proper for a mobile agent code language because it's provides powerful functions in terms of memory access. Furthermore, C is a language which can easily interface with a variety of low-level hardware devices. Ara [17, 18] and TACOMA [19] are two mobile agent platforms supporting C mobile agent code, while Ara also supports C++. Mobile agent code is compiled as byte code [20] and machine code [21] for execution in both Ara and TACOMA, respectively.

Mobile-C [22-25] was originally developed as a stand-alone, IEEE FIPA compliant mobile agent platform to accommodate applications where low-level hardware is involved and embedded systems [26]. Most of the systems are written in C/C++; Mobile-C chose C/C++ as the mobile agent language because C has an advantage for easy interfacing with control programs and underlying hardware. Additionally, Mobile-C integrated an embeddable C/C++ interpreter, Ch [27-29], as the Agent Execution Engine (AEE) in order to run the mobile agent code. The migration of mobile agent in Mobile-C is achieved through FIPA agent communication language (ACL) messages. Using FIPA ACL messages for agent migration in FIPA compliant agent systems simplifies agent platform, reduces development effort and easily achieves inter-platform migration through well-designed communication mechanisms provided in the agent platform. Messages for agent communication and migration are presented in FIPA ACL and encoded in XML. Also, the Mobile-C library was designed to support synchronization [26] in order to protect shared resources and provide a way of deterministically timing the execution of mobile agents and threads.

In this paper we present the Mobile-C library that can embed Mobile-C into any C/C++ programs to facilitate the design of mobile agent-based applications, also the possibility

to combine the migration of the mobile agent over the network and the synchronization mechanism existing in Mobile-C. Mobile agents are an application that can control the agent platform, its modules and other mobile agents, as well as smoothly interface with a variety of low-level hardware devices. Using FIPA ACL messages for agent migration in FIPA compliant agent systems simplifies agent platform since both agent communication and migration can be achieved through the same communication mechanism provided in the agent platform. Flexible synchronization mechanisms have been added for execution and interaction of several mobile agents. This paper proposes a new approach by combining two concept migration and synchronization supports in Mobile-C.

The remainder of the article is structured as follows. In section 2, we present the concept and the properties of mobile agent. Section 3 introduces the architecture of Mobile-C. Section 3 presents the migration of mobile agent over the network from multiple hosts. Section 4 describes the program structure and implementation of the component of agency. Section 5 gives an example of a mobile agent that migrates from hosts via mobile agent messages and illustrates the synchronization support in Mobile-C.

II. MOBILE AGENTS

An agent is defined as "person who's acting on behalf of other people" [30]. In the context of computer science, mobile agent is considered as an entity that moves from one machine to another in the network to perform certain tasks on behalf of the user [31].

Mobile agents have the following properties which distinguish them from other programs [32]:

- **Adaptability** - Mobility of agent required to learn about user's behavior and adapt it to suit the user. Indeed, to evolve adequately the differences between heterogeneous systems, the agent must be able to adopt the changes during the execution.
- **Autonomy**- Mobile agent must be able to make his own decision to be performed to achieve the user's tasks, also he must be able to migrate from one machine to another in the network and execute the user's tasks.
- **Communication** - Mobile agent must have the ability to communicate with others agents of the system in order to exchange information and benefit from the knowledge and expertise of other agents.
- **Mobility**- Mobile agent has the ability to move from one host to another, either by moving the agent's code or by serializing both code and state to allow the agent to continue the execution in a new context.
- **Persistence** - A persistent agent it will be able to retain the knowledge and state over extended period of time to be accessed later on. Once the mobile agent is set up, it is not dependent on system that has been initiated and it is automatically recovered when the agent is terminated or when it is flushed from memory to the database.

III. THE ARCHITECTURE OF MOBILE-C

The system of mobile-C is shown in figure1. Agencies are the major building blocks of the system and abode in each node of a network system in order to support Stationary Agents (SA) and Mobile Agents (MA) at runtime. They serve for locating and messaging agents, moving mobile agents, collecting knowledge about other agents and providing several places where the agent can be run. The core of an agency provides local service for agents and proxies remote agencies. The principle of an agency and their functionalities can be summarized as follows [33]:

- Agent Management system (AMS): The AMS manages the life cycle of agents in the system. It relates the creation, authentication, registration, deletion,execution, migration and persistence of agents. AMS is also responsible for receiving and dispatching mobile agent's .Each agent must register with an AMS in order to get a valid AID.
- Agent Communication Channel (ACC): The ACC forwardsmessages between local and remote entities. The interaction and coordination of mobile agents and host systems can be performed through agent communication language (ACL).
- Agent Security Manager (ASM): The ASM is responsible for protection of access for platform and infrastructure.
- Directory Facilitator: DF serves yellow page services. Agents in the system can register their services with DF for providing to the community. They can also look up required services with DF.
- Agent Execution Engine (AEE): AEE serves as the workhorsefor mobile agents. Mobile agents must reside inside an engine to execute. AEE has to be platform independent in order to support a mobile agent executingin a heterogeneous network.

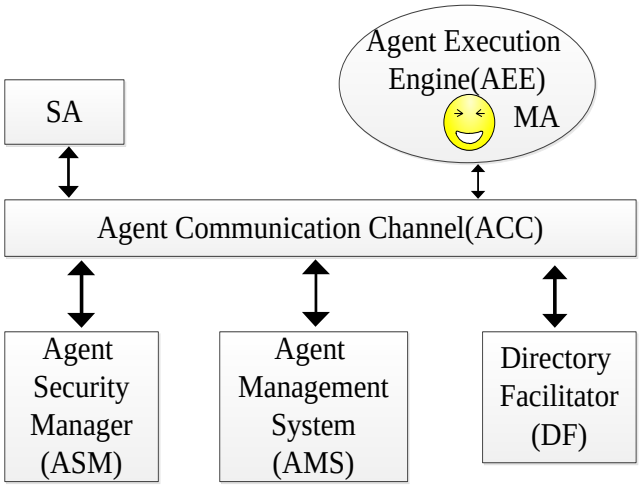

Figure 1. The system architecture of agencies in Mobile-C.

IV. MOBILE AGENT MIGRATION

Mobile agent is a software agent who is able to migrate from one host to another over the network and resume the

execution in the new host. The migration and the execution of mobile agents are supported by a mobile agent system. In previous studies, Chen et al. have developed a mobile agent system called Mobile-C.The Mobile-C supports weak migration. The task of a mobile agent can be divided into several subtasks whichcan be executed in different hosts and listed in a list of tasks as shown in figure 2. The task list can be modified by adding new subtasks and new conditions. Changing dynamically the task list improves the flexibility of a mobile agent. Thus, once we start the execution of a subtask in a host, the mobile agent cannot move until the end of execution.

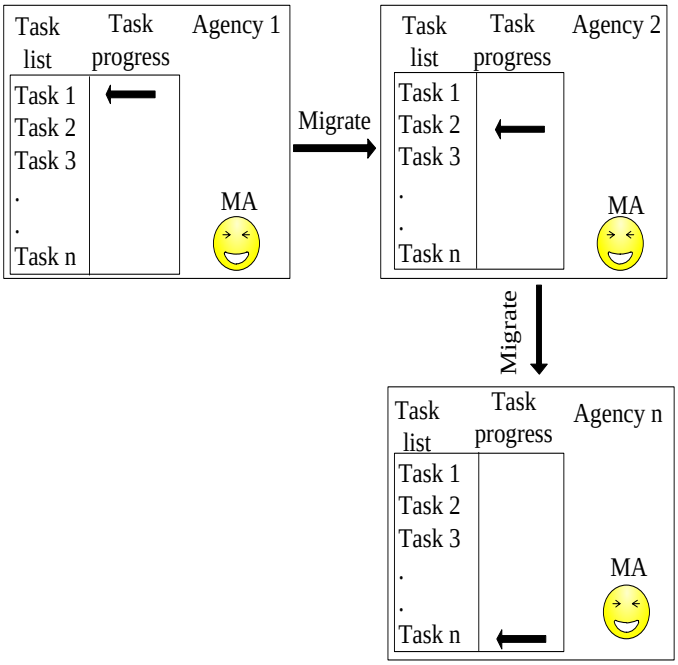

Figure 2. Agent migration based on a task list and a task progress pointer.

Mobile agent migration is achieved through ACL mobile agent messages encoded to XML, which convey mobile agents as the content of a message. Mobile agent message containsthe data state and the code of an agent. The data state of mobile agent include general information about mobile agent as agent name, agent owner and agent home , also the tasks that mobile agent will performed on destination hosts. The data state and code will be wrapping up into an ACL message and transmitted to a remote host trough Agent Communication Channel. Mobile agent migration based on ACL messages is simple and effective for agent migration in FIPA compliant systems because these systems have mandatory mechanisms for message communication, transmission and procession.

V. THE PROGRAM STRUCTURE AND IMPLEMENTATION OF COMPONENT OF AGENCY

An agency is a principle program running in each node of the network [23]. When the execution of an agency is started, the system is initialized and threads are created for all of the components in the agent platform.

After the initialization of the system, the agency waits for defined events. When the agency receives a request to run a mobile agent, it creates a new thread and embeds an Embeddable C/C++ Interpreter – Ch into the thread for executing mobile agent code. After the mobile agent migrates to the other hosts, this thread is terminated automatically (figure 3). If the agency receives a system termination request, the execution of agent platform and the system itself will be finished. In the current implementation, each mobile agent runs in an embeddable Ch inside its own thread.

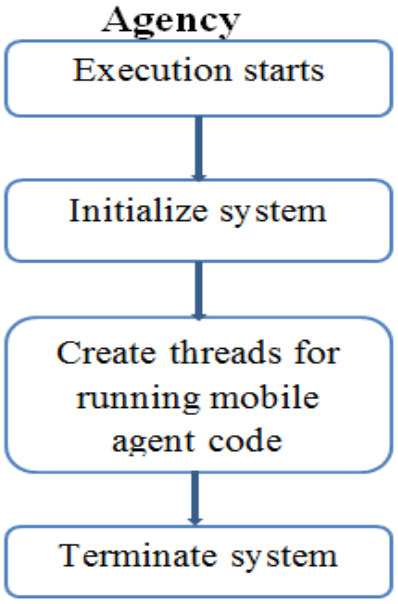

Figure 3. The program structure of an agency

According to the FIPA specifications, each agency should provide mechanisms to receive and send messages. This requirement is satisfied by three components: listening thread, connecting thread and ACC processing thread as shown in Figure 4. The listening thread serves to listen for client connections. When a new connection client is accepted, it will be added to the connection list. Also, the connecting thread is responsible for making connections with other hosts. The ACC processing thread processes the lists of client connections and requests for connecting remote hosts. The ACC facilitates remote agent to agent communication and remote agent platform to agent platform communication via ACL messages. Remote horizontal communication in Mobile-C is implemented on top of TCP/IP and the transport protocol uses HTTP (HyperText Transfer Protocol).

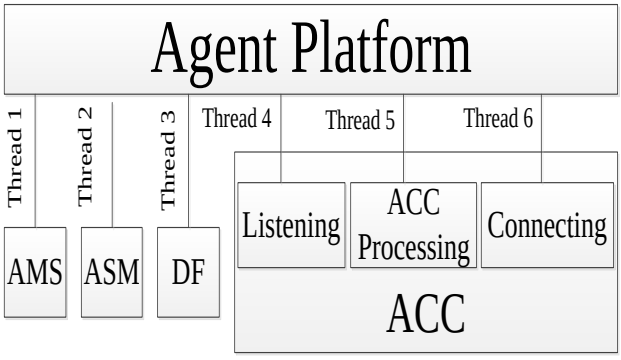

Figure 4. The multi-thread implementation of an agent platform

VI. SYNCHRONIZATION SUPPORT IN MOBILE-C

Amongproperties of mobile agents is the ability to immigrate to perform tasks that exist in the remote host. The purpose of this example is to use an embeddable mobile agent system to protect shared resources by used the synchronization and combines it with the migration of mobile agents from hosts.

The Mobile-C library allows synchronization via mutex. The mutex is a program that allows multiple threads to share the same resource, but not simultaneously.

The example below demonstrates the capabilityof a Mobile-C mutex to protect a resource that may be shared between two or more agents in several hosts. As shown in program 1, a mobile agent is transferred by an agency in the host fsts1 visits remote host fsts2 then host fsts3 .The mobile agent message is represented in Extensible Markup Language (XML), it contains information of the mobile agent and tasks that will be performed on destination hosts. The general information of a mobile agent contains:agent name, agent owner, and the home of the agent. The task information for example the statement <TASK task="2" num="0"> shows that this mobile agent has two tasks to perform and no task has been done yet.The DATA element overall information about the number of element, the name of the task's return variable, the completeness of the task and the host to perform the task.The sub-element DATA ELEMENT contains the return data from the execution task and the sub-element AGENT_CODE contains a C program which will be executed in remote host.

```
<NAME>mobileagent</NAME>
<OWNER>fsts</OWNER>
<HOME>fsts1.fsts.ac.ma:5125 </HOME>
<TASK task= "2" num= "0">
<DATA number_of_elements ="0" name = "results_fsts2"
complete = "0" server = "fsts2.fsts.ac.ma:5138">
<DATA_ELEMENT></ DATA_ELEMENT >
<AGENT_CODE>
Mobile agent code on fsts2
</AGENT_CODE>
</DATA>
<DATA number_of_elements ="0" name = "results_fsts3"
complete = "0" server = "fsts3.fsts.ac.ma:5135">
<DATA_ELEMENT></ DATA_ELEMENT >
<AGENT_CODE>
Mobile agent code on fsts3
</AGENT_CODE>
</DATA>
</TASK>
```

Program 1: The content of the mobile agent message from the host fsts1 to fsts2 and to host fsts3

As shown in Program2, the mobile agent 1" MA1" initialize a mutex with an ID 55 via the function mc_SyncInit() and defines two functions, SetN1() and GetN1() in the host fsts2 . After visiting this host, the mobile agent 1 "MA1" visits the host fsts3 and defines also two functions, SetN2() and GetN2().The result obtained from the host fsts2 is sent to the host fsts3 and the return data will be included in the sub-element DATA_ELEMENT.

```
<NAME>MA1</NAME>
<OWNER>fsts</OWNER>
<HOME> fsts1.fsts.ac.ma:5125 </HOME>
<TASK task= "2" num= "0">
<DATA number_of_elements ="0" name = "results_fsts2"
complete = "0" server = "fsts2.fsts.ac.ma:5138">
<DATA_ELEMENT></ DATA_ELEMENT >
<AGENT_CODE>
<![CDATA[
int N1;
int main () {
intmutex_id=55;
mc_SyncInit(mutex_id);
return 0;
}
void SetN1(inti){
N1 +=i;
if(N1 > 1000){
N1=0;
}
}
intGetN(){
return N
}
]]>
</AGENT_CODE>
</DATA>
<DATA number_of_elements ="0" name = "results_fsts3"
complete = "0" server = "fsts3.fsts.ac.ma:5135">
<DATA_ELEMENT></ DATA_ELEMENT >
<AGENT_CODE>
<![CDATA[
int N2;
int main () {
intmutex_id=55;
mc_SyncInit(mutex_id);
return 0;
}
void SetN2(inti){
N2*=i;
if(N2 > 1000){
N2=0;
}
}
intGetN(){
return N2
}
]]>
</AGENT_CODE>
</DATA>
</TASK>
```

Program2 .A mobile agent that contains a global variable and defines function to access the global variables

As shown in Program 3, the task of the mobile agent 2"MA2" is to perform the operation setting the variable. The operation includes locking the mutexthrough the function mc_MutexLock(), setting the global variable by calling

SetN1() from the host fsts2 and SetN2()from the host fsts3via the function mc_CallAgentFunc(), and unlocking the mutex via the function mc_MutexUnlock().

```
<NAME>MA2</NAME>
<OWNER>fsts</OWNER>
<HOME> fsts1.fsts.ac.ma:5125 </HOME>
<TASK task= "2" num= "0">
<DATA number_of_elements ="0" name = "results_fsts2"
complete = "0" server = "fsts2.fsts.ac.ma:5138">
<DATA_ELEMENT></ DATA_ELEMENT >
<AGENT_CODE>
<![CDATA[
#include <stdio.h>
int main (){
MCAgent_t agent;
inti= 0,mutex_id =55 ,retval;
agent= mc_FindAgentByName("MA1");
wile(1){
mc_MutexLock(mutex_id);
mc_CallAgentFunc(agent,"SetN1",NULL,i);
printf("N1:%d\n",retval);
mc_MutexUnlock(mutex_id)
i++;
if(i==20) {
i=0;
      }
}
return 0;
]]>
</AGENT_CODE>
</DATA>
<DATA number_of_elements ="0" name = "results_fsts3"
complete = "0" server = "fsts3.fsts.ac.ma:5135">
<DATA_ELEMENT></ DATA_ELEMENT >
<AGENT_CODE>
<![CDATA[
#include <stdio.h>
int main (){
MCAgent_t agent;
inti= 0,mutex_id =55 ,retval;
agent= mc_FindAgentByName("MA1");
wile(1){
mc_MutexLock(mutex_id);
mc_CallAgentFunc(agent,"SetN2",NULL,i);
printf("N2:%d\n",retval);
mc_MutexUnlock(mutex_id)
i++;
if(i==20) {
i=0;
      }
}
return 0;
]]>
</AGENT_CODE>
</DATA>
</TASK>
```

Program3. A mobile agent that sets a variable

Likewise, as shown in Program 4, the task of the mobile agent 3“MA3” is locks the mutex, get the global variable, and unlocks the mutex from both the host fsts2 and fsts3

```
<NAME>MA3</NAME>
<OWNER>fsts</OWNER>
<HOME> fsts1.fsts.ac.ma:5125 </HOME>
<TASK task= "2" num= "0">
<DATA number_of_elements ="0" name = "results_fsts2"
complete = "0" server = "fsts2.fsts.ac.ma:5138">
<DATA_ELEMENT></ DATA_ELEMENT >
<AGENT_CODE>
<![CDATA[
#include <stdio.h>
int main (){
MCAgent_t agent;
inti ,mutex_id =55 ,retval;
agent= mc_FindAgentByName("MA1");
mc_MutexLock(mutex_id);
mc_CallAgentFunc(agent,"GetN",&retval,NULL);
printf("N1:%d\n",retval);
mc_MutexUnlock(mutex_id)
return 0;
]]>
</AGENT_CODE>
</AGENT_CODE></DATA>
<DATA number_of_elements ="0" name = "results_fsts3"
complete = "0" server = "fsts3.fsts.ac.ma:5135">
<DATA_ELEMENT></ DATA_ELEMENT >
<AGENT_CODE>
<![CDATA[
#include <stdio.h>
int main (){
MCAgent_t agent;
inti ,mutex_id =55 ,retval;
agent= mc_FindAgentByName("MA1");
mc_MutexLock(mutex_id);
mc_CallAgentFunc(agent,"GetN",&retval,NULL);
printf("N2:%d\n",retval);
mc_MutexUnlock(mutex_id)
return 0;
]]>
</AGENT_CODE>
</DATA>
</TASK>
```

Program4. A mobile agent that gets a variable from another agent.

The results of the mobile agent 1”MA1”, mobile agent 2 “MA2” and mobile agent3 “Ma3” obtained from both the host fsts2 and fts3 are send back to the home agency fsts.

VII. CONCLUSION

In this work we present the design and implementation of an IEEE FIPA compliant agent platform, Mobile-C. Mobile-C integrates an embeddable C/C++ interpreter—Ch—into the platform as a mobile agent execution engine in order to support mobile agent. The migration of mobile agent is achieved through ACL messages. Mobile agents, including both its data state and code, are transported to a remote agent platform via

ACL messages which is encoded in XML, and the execution of mobile agents is resumed by the task progress pointer. The Mobile-C library supports the synchronization among mobile agents and threads because the synchronization functions protect shared resources and provide a way of deterministically timing the execution of mobile agents and the migration to a remote host.

In our future work, this framework will be tested and extended in various types of industrial applications like e-commerce and network management.

ACKNOWLEDGMENT

The authors thank the referees for valuable constructive comments and suggestions which lead to a significant improvement of this paper.

REFERENCES

[1] D.Chess,C.Harrison, A.Kershenbaum,"Mobile Agentes: Are They a Good Idea?", IBM ResearchReport, 1998 [Online] Available : http://www.research.ibm.com/iagentes/paps/mobile -idea.ps.

[2] W. Shen, D. Xue, D.H. Norrie, An agent-based manufacturing enterprise infrastructure for distributed integrated intelligent manufacturing systems, in: Proceedings of the 3rd International Conference on the Practical Applications of Agents and Multi-Agent Systems (PAAM-98), London, UK, 1998, pp. 533–548.

[3] H. Wada, S. Okada, An autonomous agent approach for manufacturing execution control systems, Integrated Computer-Aided Engineering 9 (3) (2002) 251–262.

[4] H.V.D. Parunak, A.D. Baker, S.J. Clark, The AARIA agent architecture: from manufacturing requirements to agent-based system design, Integrated Computer-Aided Engineering 8 (1) (2001) 45–58.

[5] M. Yokoo, S. Fujita, Trends of internet auctions and agent-mediated web commerce, New Generation Computing 19 (4) (2001) 369–388.

[6] T. Sandholm, eMediator: a next generation electronic commerce server, Computational Intelligence 18 (4) (2002) 656–676.

[7] S.P.M. Choi, J. Liu, S. Chan, A genetic agent-based negotiation system, Computer Networks: The International Journal of Computer and Telecommunications Networking 37 (2) (2001) 195–204.

[8] W.E. Chen, C. Hu, A mobile agent-based active network architecture for intelligent network control, Information Sciences 141 (1–2) (2002) 3–35.

[9] L. Chou, K. Shen, K. Tang, C. Kao, Implementation of mobile-agent-based network management systems for national broadband experimental networks in Taiwan, Holonic and Multi-Agent Systems for Manufacturing (Lecture Notes in Computer Science) 2744 (2003) 280–289.

[10] J. Huang, N.R. Jennings, J. Fox, Agent-based approach to health care management, Applied Artificial Intelligence 9 (4) (1995) 401–420.

[11] I. Noda, P. Stone, The RoboCup soccer server and CMUnited clients: implemented infrastructure for MAS research, Autonomous Agents and Multi-Agent Systems 7 (1–2) (2003) 101–120.

[12] K.Straber, J.Baumann and F.Hohl.. Mole - A Java Based Mobile Agent System. Institute for Parallel and Distributed Computer Systems, University of Stuttgart,1997

[13] D. Lange, M.Oshima. Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley: MA, 1998.

[14] D.Wong, N.Paciorek, T.Walsh, J.DiCelie, M.Young, B.Peet. Concordia: An infrastructure for collaborating mobile agents. Proceedings of the First International Workshop on Mobile Agents (MA'97) (Lecture Notes in Computer Science, vol. 1219). Springer: Berlin, 1997; 86–97.

[15] F.Bellifemine, G.Caire, A.Poggi, G.Rimassa.JADE: A software framework for developing multi-agent applications.Lessons learned. Information and Software Technology 2008; 50(1–2):10–21.

[16] R.Gray, G.Cybenko, D.Kotz,R.Peterson, D.Rus. D'Agents: Applications and performance of a mobile-agent system. Software—Practice and Experience 2002; 32(6):543–573.

[17] H.Peine. Run-time support for mobile code. PhD Dissertation, Department of Computer Science, University of Kaiserslautern, Germany, 2002.

[18] H.Peine .Application and programming experience with the Ara mobile agent system. Software—Practice and Experience 2002; 32(6):515–541.

[19] D.ohnansen, K.Lauvset, R.V.Renesse, F.B. Schneider, N.P. Sudmann, K. Jacobsen. A TACOMA retrospective.Software—Practice and Experience 2002; 32(6):605–619.

[20] MACE—Mobile agent code environment. Available at: http://wwwagss.informatik.uni-kl.de/Projekte/Ara/mace.html [last modified 10 August 2004].

[21] N.P.Sudmann,D.Johansen. Adding mobility to non-mobile web robots. Proceedings of the IEEE ICDCS00 Workshop on Knowledge Discovery and Data Mining in the World-wide Web, Taipei, Taiwan, 2000; 73–79.

[22] B.Chen, H.H.Cheng. A run-time support environment for mobile agents. Proceedings of ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications, No. DETC2005-85389, Long Beach, CA, September 2005.

[23] B.Chen,H.H.Cheng,J.Palen. Mobile-C: A mobile agent platform for mobile C/C++ agents. Software—Practice and Experience 2006; 36(15):1711–1733.

[24] B.Chen, D.Linz, H.H.Cheng. XML-based agent communication, migration and computation in mobile agent systems. Journal of Systems and Software 2008; 81(8):1364–1376.

[25] Mobile-C: A multi-agent platform for mobile C/C++ code. Available at: http://www.mobilec.org.

[26] Y.C. Chou, D. Ko, H. H. Cheng, An embeddable mobile agent platform supporting runtime code mobility,interaction and coordination of mobile agents and host systems, Information and Software Technology 52 (2010) 185–196

[27] H.H.Cheng. Scientific computing in the Ch programming language. Scientific Programming 1993; 2(3):49–75

[28] H.H.Cheng.Ch: A C/C++ interpreter for script computing. C/C++ User's Journal 2006; 24(1):6–12.

[29] Ch—An embeddable C/C++ interpreter. Available at: http://www.softintegration.com .

[30] P.M .Reddy, Mobile Agents Intelligent Assistants on the Internet, July 2002

[31] A. Kaur , S.Kaur, Role of Mobile Agents In Mobile Computing, Proceedings of National Conference on Challenges & Opportunities in Information Technology (COIT-2007) RIMT-IET

[32] M. L. Griss, Software Agents as Next Generation Software Components, Chapter 36 in Component-Based Software Engineering: Putting the Pieces Together, Edited by George T. Heineman, Ph.D. & William Councill, M.S., J.D., May 2001,Addison-Wesley

[33] B. Chen, H. H. Cheng, J.Palen, Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems, Transportation Research Part C 17 (2009) 1–10