Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 81 (2008) 1364–1376
The Journal of Systems and Software
www.elsevier.com/locate/jss

XML-based agent communication, migration and computation in mobile agent systems

Bo Chen [a,1], David D. Linz [b], Harry H. Cheng [b,*]

[a] Department of Mechanical Engineering-Engineering Mechanics, Michigan Technological University, Houghton, MI 49931, United States
[b] Integration Engineering Laboratory, Department of Mechanical and Aeronautical Engineering, University of California, Davis, CA 95616, United States

Received 17 November 2006; received in revised form 24 October 2007; accepted 27 October 2007
Available online 12 November 2007

Abstract

This article presents the research work that exploits using XML (Extensible Markup Language) to represent different types of information in mobile agent systems, including agent communication messages, mobile agent messages, and other system information. The goal of the research is to build a programmable information base in mobile agent systems through XML representations. The research not only studies using XML in binary agent system space such as representing agent communication messages and mobile agent messages, but also explores interpretive XML data processing to avoid the need of an interface layer between script mobile agents and system data represented in XML. These XML-based information representations have been implemented in Mobile-C, a FIPA (The Foundation for Intelligent Physical Agents) compliant mobile agent platform. Mobile-C uses FIPA ACL (Agent Communication Language) messages for both inter-agent communication and inter-platform migration. 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 system. The ability of interpretive XML data processing allows mobile agents in Mobile-C directly accessing XML data information without the need of an extra interface layer.

Keywords: Mobile agents; Agent communication; Mobility; XML

1. Introduction

Mobile agent technology is promising for many applications (Lange and Oshima, 1999), where mobile agents roam over the network accessing distributed resources and cooperating with other agents or non-agent components during the course of performing certain tasks. Agent collaboration is important for complex tasks. The intelligence of an agent system is not only reflected by the expertise of individual agents but also exhibited by the collaborative behavior of a group of agents. Cooperation and collaboration of agents require agents to communicate and interact with each other. The communication and interaction could be intra-platform or inter-platform. In a heterogeneous network, an agent may also need to cooperate with agents developed for different platforms or written in other languages. Furthermore, a group of cooperating agents and their interacting pattern are usually dynamic and are unknown at the design stage. As a result, an open and flexible communication and interaction model is needed to ensure the interoperability among mobile agents and mobile agent systems.

Agent communication and interaction are achieved through agent communication mechanisms provided by agent systems. Agent communication mechanisms vary considerably from one agent system to another agent system. Commonly used communication approaches include events, remote method invocation, and message passing. For the remote method invocation, it is usually language dependent and is commonly used in Java-based agent

* Corresponding author. Tel.: +1 530 752 5020; fax: +1 530 752 4158.
E-mail addresses: bochen@mtu.edu (B. Chen), hhcheng@ucdavis.edu (H.H. Cheng).
[1] Tel.: +1 906 487 3537; fax: +1 906 487 2822.

0164-1212/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2007.10.026

Nomenclature

XML	Extensible Markup Language	DF	Directory Facilitator
FIPA	The Foundation for Intelligent Physical Agents	AEE	Agent Execution Engine
ACL	Agent Communication Language	ACC	Agent Communication Channel
SQL	Structured Query Language	ODBC	Open Database Connectivity
DTD	Document Type Definition	HTTP	Hypertext Transfer Protocol
AMS	Agent Management System	SDK	Software Development Kit
ASM	Agent Security Manager	API	Application Programming Interface

systems. Many agent systems employ message passing for complicated agent communication scenarios. To promote the interoperation of agents and agent systems across heterogeneous agent platforms, FIPA (The Foundation for Intelligent Physical Agents) (FIPA) has been working on specifications that range from agent platform architecture to support inter-agent communication, communication languages and content languages for expressing exchanging messages, and interaction protocols that expand the scope from single messages to complete transactions. Similar to agent communication mechanisms, agent mobility mechanisms that support agent migration vary in different agent systems. Although the agent migration through standard agent messages has been proposed (Ametller et al., 2003), most systems use customized mechanisms to transport mobile agents. These customized agent migration mechanisms are usually sophisticated and separated with the communication mechanisms.

A mobile agent system has different types of information, including communication messages, mobile agent messages, and system data. Using a unified information representation will definitely simplify agent systems and reduce development effort. XML (Extensible Markup Language) is a standard for building tag-based structured documents/data. It is a markup language for representing and exchanging data over the Internet. A number of researchers had proposed to use XML for encoding ACL (Agent Communication Language) messages (Grosof and Labrou, 1999; Griss, 2001; Leung and Li, 2004), representing documents, such as business documents (Glushko et al., 1999; Sedbrook, 2001), course content (Madjarov et al., 2004), ontology (De Meo et al., 2003), and system parameters (Mathieu and Verrons, 2003), and performing agent coordination (Cabri et al., 2001). These researches mainly explore using compiled programs to process XML data. This article presents an exploration of using XML to represent different types of information in a mobile agent system. The system not only uses XML to represent agent communication messages and mobile agent messages, and processes these XML messages in binary agent system space, but also allows mobile agents to process XML data interpretively to avoid the need of an interface layer between script mobile agents and system data represented in XML. The interpretive XML data processing is achieved through an interpretive software package Ch XML (Wang and Cheng, 2006) based on Gnome libxml2 (Libxml2) and Oracle XDK (Oracle XML Developer's Kit) for C/C++ (Oracle XML Developer's Kit).

XML-based agent communication, migration, and computation have been implemented in Mobile-C (Chen et al., 2004; Chen, 2005; Chen et al., 2006; Mobile-C), a mobile agent system that is compliant to IEEE FIPA agent standards. Mobile-C uses IEEE FIPA ACL messages for inter-agent communication and inter-platform mobile agent migration. Compliance with a dominant international agent standard greatly enhances the openness and interoperability of Mobile-C. Encoding ACL messages in XML further increases the flexibility of agent communication and migration. Mobile agents in Mobile-C are migrated via FIPA ACL messages. 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.

The rest of the article is organized as follows. Section 2 highlights advantages of using XML to encode communication and mobile agent messages and represent other types of data in mobile agent systems. Section 3 shows two types of messages in Mobile-C, agent communication messages and mobile agent messages. Section 4 presents the system architecture and major components of the Mobile-C, a mobile agent system implementing these XML representations. Section 5 explains how mobile agent messages are processed and the life cycle of a mobile agent is managed in an agent platform. Section 6 gives an example of a mobile agent that migrates via mobile agent messages and processes XML data interpretively in remote hosts. Section 7 compares processing times for a mobile agent to process raw data that are stored in data files, XML files, and SQL (Structured Query Language) databases. Finally, we draw conclusions in Section 8.

2. Good reasons of exploiting XML in agent systems

Using XML to encode ACL messages and represent other types of information in agent systems offers a number of advantages.

- Comparing with other encoding formats, such as Lisp-like encoding and plain ASCII (The American Standard Code for Information Interchange) text, XML is easy to generate, parse, edit, and translate because of its strict and extremely consistent syntax, and a rich set of standard XML tools available for these purposes.
- XML is a self-describing file format to store structured information. It allows application developers to define their own document structure and vocabularies for describing different format of information. The modification of ACL syntax won't result in re-writing the parser as long as such changes are reflected in the ACL DTD (Document Type Definition).
- XML documents and parsing standards are open. The complete specification for XML is available at W3C's Web page (World Wide Web). The openness of XML makes XML document interchangeable, satisfying the interoperability of a system in which heterogeneous entities need to interact with each other. XML information can be transmitted by any systems capable of transferring text and move over to systems that have a mechanism to parse and use XML information.
- Many agent systems are internet-related and agents are often applied to Web-based systems. XML is more flexible than HTML (Hypertext Markup Language) and less complex than SGML (Standard Generalized Markup Language) (Overview of SGML Resources) for Web-based applications. Using XML to encode ACL messages and represent different types of data facilitates the practical integration with a variety of Web technology and leverage Web-based tools and infrastructure.
- XML's hierarchical structures are well suited for representing different types of information in the real world, such as documents, database, and objects. The ability of representing different formats of data, including communication messages, agent data, access control policies, and raw data, allows it to form a uniform information base. This information base is programmable and easy to be managed either in files or databases.
- XML is platform independent. It is suitable for use in heterogeneous environments.

3. Messages in Mobile-C

There are two types of messages in Mobile-C. One type is agent communication messages, and another type is mobile agent messages. A sample agent communication message from *agent-a* to *agent-b* requesting the travel time on a highway from city Davis to city Dixon is shown in Fig. 1. The sample message is represented in XML. In Fig. 1, the sender and intended recipient of the message are identified by their agent-identifiers. For the sample message, the sender and receiver agent names are *agent-a@duck.engr.ucdavis.edu* and *agent-b@dragon.engr.ucdavis.edu*, respectively. The sender and receiver agent addresses are http://duck.engr.ucdavis.edu:5120 and http://dragon.engr.ucdavis.edu:5120, respectively. The ontology attribute denotes the ontology(s) used to give a

```
<?xml version="1.0"?>
<!-- ACLMessage -->
<!DOCTYPE gafmessage SYSTEM "Mobile-C.dtd">
<gafmessage type = "Request">
    <sender>
        <agent-identifier>
            <name>agent-a@duck.engr.ucdavis.edu</name>
            <address>
                 <url>http://duck.engr.ucdavis.edu:5120</url>
            </address>
        </agent-identifier>
    </sender>
    <receiver>
        <agent-identifier>
            <name>agent-b@dragon.engr.ucdavis.edu</name>
            <address>
                <url>http://dragon.engr.ucdavis.edu:5120</url>
            </address>
        </agent-identifier>
    </receiver>
    <content ontology="TDMS">
        <expression>
            <datatype>Travel Time</datatype>
            <origin>Davis</origin>
            <destination>Dixon</destination>
        </expression>
    </content>
    <protocol>Request-Interaction-Protocol</protocol>
</gafmessage>
```

Fig. 1. An ACL message represented in XML.

```
Message type: MOBILE_AGENT

Sender: Agent ID

Receiver: Agent ID

Content:
  name
  owner
  home
  tasks:
     number of tasks
     task progress pointer
     task1:
       mobile agent data
       return data
       mobile agent code
     task2:
       ⋮
     taskn:
```

Fig. 2. A mobile agent message.

meaning to the symbols in the content expression. The requested action is specified within the expression tags. The message requests the travel time from the origin Davis to the destination Dixon. The interaction protocol is Request-Interaction-Protocol.

Mobile-C supports weak migration in the current implementation. The task of a mobile agent is divided into multiple subtasks similar to the approach presented in Chong et al. (1999). These subtasks can be executed in different hosts and are listed in a task list. New subtasks can be added into the list and the task list can be modified dynamically based on the new conditions. The ability to dynamically change the task list significantly improves the flexibility of a mobile agent. However, once a subtask is started to execute in a certain host, the mobile agent cannot move until the termination of the execution. Mobile agent migration is based on mobile agent messages. A mobile agent message contains general information of a mobile agent and tasks that the mobile agent is going to perform on destination hosts as shown in Fig. 2. The general information of a mobile agent includes agent name, agent owner, and the home agency that creates the mobile agent. Task information includes number of tasks, a task progress pointer, description of each task, and the code for each task. During agent migration, intermediate results from previous tasks are also capsulated into agent messages. At the end of migration, all of the results are sent back to the home agency. The examples of mobile agent messages will be presented in Section 6.

4. System architecture and major components of Mobile-C

The system architecture of Mobile-C is a peer-to-peer network architecture as shown in Fig. 3. Agencies are major

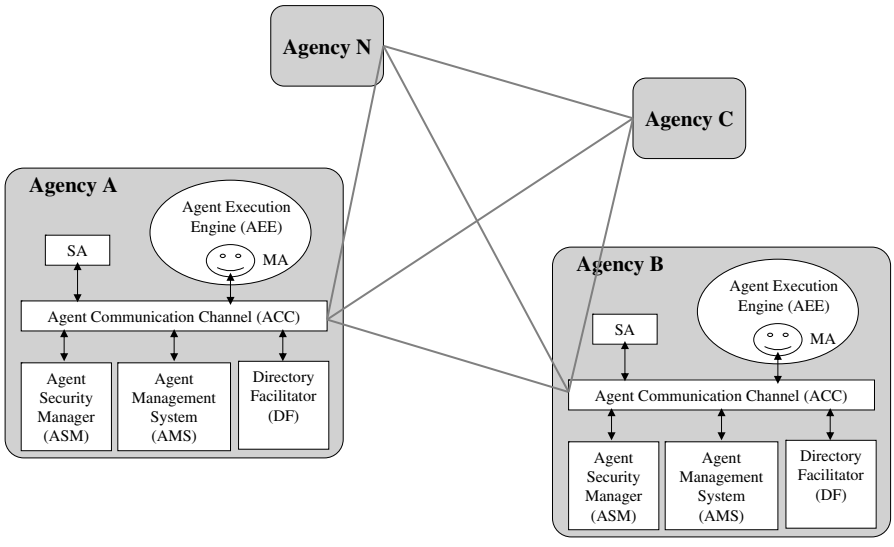

Fig. 3. The system architecture of Mobile-C.

building blocks of the system and reside in each node of the network to support stationary and mobile agents at runtime. They serve as "home bases" for locating and messaging agents, migrating mobile agents, collecting knowledge about the group of agents, and providing an environment in which a mobile agent executes (Griss, 2001). The core of the agency is the agent platform, which provides local service for agents and proxies to access remote agencies. The main components of an agent platform include Agent Management System (AMS), Agent Security Manager (ASM), Directory Facilitator (DF), Agent Execution Engine (AEE), and Agent Communication Channel (ACC). The AMS manages the life cycle of the agents. It controls creation, registration, migration, and persistence of agents. The ASM is responsible for maintaining security policies for the platform and infrastructure, such as communication and transport-level security. The 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. The AEE serves as the execution environment for mobile agents. Mobile agents must reside inside an engine to execute. The ACC routes messages between local and remote entities, realizing messages using an agent communication language.

An agency is a main program running on each node in a network. When the execution of an agency is started, the agency initializes the system and creates threads for all of the components in the agent platform. After initializing system, the agency waits for defined events. When an agency receives a request to run a mobile agent, it creates a new thread and embeds an Embeddable C/C++ Interpreter – Ch (Cheng, 1993; Ch – An Embeddable C/C++ Interpreter) into the thread for executing mobile agent code. After the mobile agent migrates to the other hosts, this thread is terminated automatically. If an agency receives a system termination request, it terminates the execution of agent platform and the system itself. In the current implementation, each mobile agent runs in an embeddable Ch inside its own thread. The multi-thread approach is much more efficient than multi-process approach because start-up and termination of multi-processes and expensive Inter-Process Communication are avoided.

As previously described, an agent platform is the core of an agency. The agent platform is also a multi-thread program. Each component of an agent platform works in an individual thread, which ensures that all of the components can work concurrently. The AMS is responsible for managing agents. All of the registered agents are recorded in an agent list. Each node of the agent list contains information for an agent, such as agent type, agent identifier for a stationary agent, and data state and code for a mobile agent. The AMS is also responsible for accepting and dispatching mobile agents, including creating a mobile agent from a mobile agent message, requesting for executing and suspending the execution of a mobile agent, and generating a mobile agent message from a mobile agent structure. The DF manages all of the registered agent services. To speed up searching a requested service, a Service Hash Table and two types of linked lists are designed to record available agent services.

The AEE is a critical component in a mobile agent system. Mobile-C chooses C/C++ as a mobile agent language and uses an Embeddable C/C++ interpreter – Ch (Cheng, 1993; Ch – An Embeddable C/C++ Interpreter) as an AEE. Using Ch as an embedded runtime scripting environment in a mobile agent system has many advantages over other alternatives. First, all of the C functions can readily be used in mobile agent scripts. Second, if the agent system programs are written in C, the communication (including data sharing) between mobile agents and system programs is much easier to achieve than other scripting languages. Third, as a superset of C, Ch supports many standards and libraries, such as CGI (Common Gateway Interface), XML (Ch XML Package for Libxml2), ODBC (Open Database Connectivity), GTK+ (The GIMP Toolkit), OpenGL (Open Graphics Library) (Chen and Cheng, 2005), and OpenCV (Yu et al., 2004), an open source computer vision library. An increasing number of toolkits and packages, such as Control Systems Toolkit (Yu et al., 2004), and Mechanism Toolkit (Cheng and Trang, 2006) are available for solving complicated engineering problems. Finally, Ch supports most existing platforms, such as Windows, Linux, Solaris, HP-UX, Mac OS, FreeBSD, and QNX. It is suitable for use in a heterogeneous network. Each mobile agent is interpreted inside an interpreter, which is embedded into the system as a separate thread. The agent platform mediates mobile agents to communicate with other agents or access the host system.

The ACC facilitates the remote horizontal communication via ACL messages, such as remote agent-to-agent communication and agent platform to agent platform communication. The ACC consists of three modules that are running in three threads, a listening thread, a connecting thread, and an ACC processing thread. The listening thread is used to listen for client connections, and the connecting thread is responsible for connecting to other hosts. The ACC processing thread processes linked lists of client connections and requests for connecting remote hosts. Remote horizontal communication in Mobile-C is implemented on top of TCP/IP. The transport protocol uses HTTP (Hypertext Transfer Protocol). Messages are asynchronously sent to a recipient agent identified by its global unique identifier. The local agent-to-agent platform information exchange is achieved by Ch SDK (Software Development Kit) and Embedded Ch SDK (Ch – An Embeddable C/C++ Interpreter), an interface between agent platform space (binary C/C++ space) and mobile agent space (Ch space).

Each agency in Mobile-C is expected to be able to communicate with multiple agencies concurrently and act as a server and a client at the same time in the communication. Two data objects, Server Connection List and Client Connection List shown in Fig. 4, are designed to record the client connection requests from other agencies and local requests to connect to remote agencies. These two objects

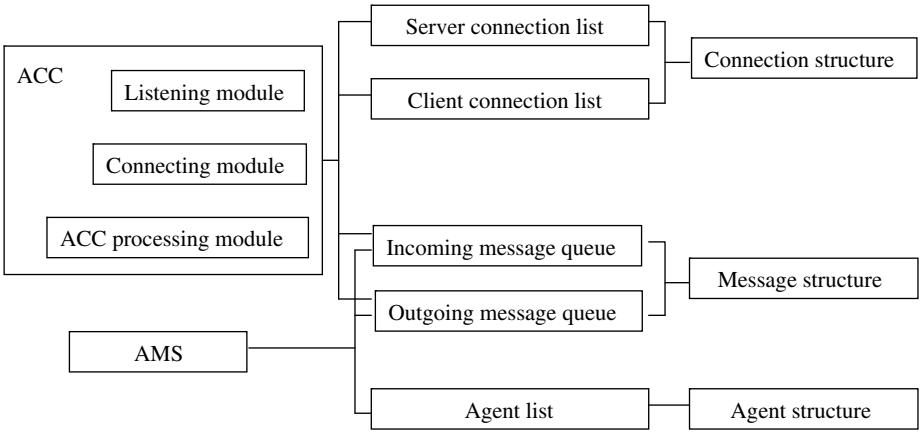

Fig. 4. Data objects related to agent messages.

are managed by Listening Module, Connecting Module, and ACC Processing Module in the ACC object. Each node of these two objects is built on a Connection Structure, which records the socket connection information. There are two threads accessing each of these two lists. Listening thread and ACC processing thread access Server Connection List, while connecting thread and ACC processing thread access Client Connection List. Since each connection list is accessed by two threads, a *mutex* is used to control the accessing of each list. A "*mutex*" is a program object that allows multiple threads to share the same resource (the Server Connection List or the Client Connection List in this application), but not simultaneously. When an agency program is started, a "*mutex*" is created with a unique name for each list. After that, any thread that needs the resource must lock the "*mutex*" from other threads while it is using the resource. The "*mutex*" is set to unlock when the resource is no longer needed or the routine is finished. ACC Processing Module processes Server Connection List and adds the ACL messages into the Incoming Message Queue. Similarly, messages in the Outgoing Message Queue are delivered to the remote agencies by ACC.

A mobile agent message in the Incoming Message Queue is processed by AMS to create a mobile agent structure and initiate a mobile agent. Also, when a mobile agent wants to migrate to a remote agency, AMS wraps mobile agent code and state, constructs a mobile agent message, and adds the mobile agent message to the Outgoing Message Queue.

5. Mobile agent message processing and life cycle management

Mobile agent migration in Mobile-C is accomplished through mobile agent messages. When a mobile agent migrates to a remote host, it goes through three stages as shown in Fig. 5. In the first stage, a received mobile agent message is added into an Incoming Message Queue by ACC. Because the message type is MOBILE_AGENT, AMS creates an instance of mobile agent structure, initializes the mobile agent structure according to the information contained in the mobile agent message, and then adds this newly created mobile agent (represented by a mobile agent structure) to a mobile agent list. A mobile agent structure contains agent's general information, task information, and a task progress pointer. Multiple tasks are represented by an array of task structures. Since Embedded Ch can execute small-size mobile code stored in a buffer, the mobile agent code in a mobile agent message does not have to be saved into a file. This feature significantly speeds up mobile agent execution because creating a file and running the code from a file is time consuming.

In the second stage, a mobile agent physically resides on an agent platform and managed by the agent platform according to the FIPA agent life cycle model (FIPA). When a mobile agent is about to migrate to a remote agency, the mobile agent goes into the third stage. AMS wraps the agent data state, results from previous tasks, and mobile agent code to a XML file and reconstructs a mobile agent message for transmitting. Once a mobile agent message is created, AMS adds this message into the Outgoing Message Queue. Finally, the mobile agent message is sent to a remote agency by ACC.

In Mobile-C, mobile agent data and code are represented as sub-nodes of a mobile agent message as shown in Fig. 2. According to FIPA specifications, mobile agent messages are delivered in a transport-message format. A transport-message consists of a payload and an envelope. The envelope includes the sender and receiver transport-descriptions, and the payload encodes a message. When a mobile agent transport-message is sent to a remote host, i.e., a mobile agent migrates to a remote host, the agent platform on the remote host saves the payload of the transport-message in a memory block. The process of creating a mobile agent structure and saving mobile agent code from the message payload is illustrated in Fig. 6. First, an XML Parser creates an XML Document Tree from an XML in memory document. Second, an XML processing module transforms agent data state node and code node in the XML Document Tree into manageable data. The extracted

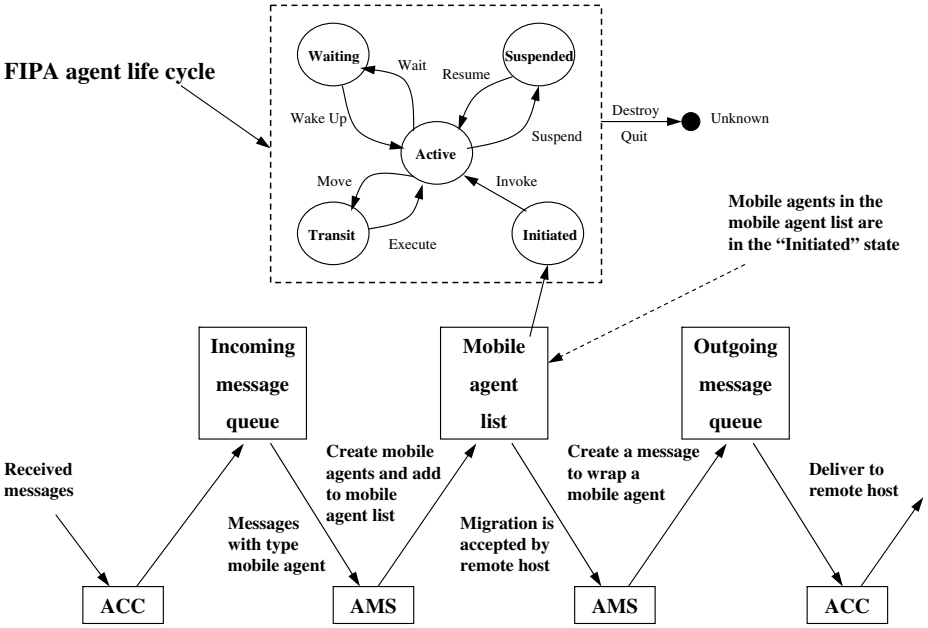

Fig. 5. Mobile agent message processing and life cycle control.

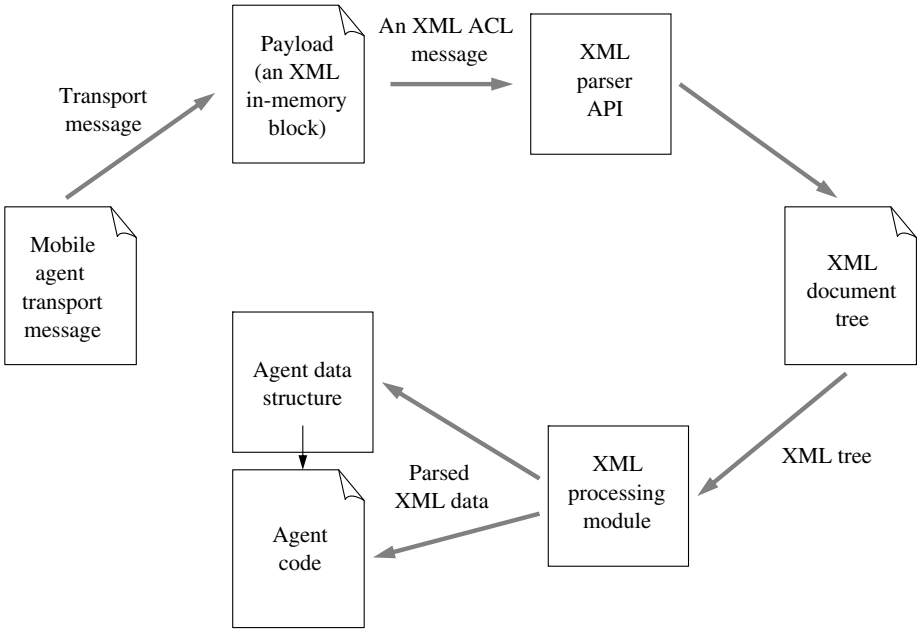

Fig. 6. Parsing mobile agent data and code from a mobile agent message.

agent data are used to create a mobile agent structure. Agent code can either be saved to a buffer or a file based on its size.

We use libxml2 (Libxml2), a C language library, for reading, creating and manipulating XML data. Libxml2 is a XML C parser and toolkit developed for the Gnome project but usable outside of the Gnome platform. It is open source and portable across a large number of platforms. The main reasons that we chose libxml2 are:

- Libxml2 is written in C. It is easy to embed into our system.
- Libxml2 works on Linux/Unix/Windows and a number of other platforms. It is suitable for use in a heterogeneous network system.
- Libxml2 can do DTD validation at parse time, using a parsed document instance, or with an arbitrary DTD.
- Basic support for HTTP and FTP (File Transfer Protocol) client allowing applications to fetch remote resources.
- Libxml2 implements a number of existing standards related to markup languages, such as XPath, XPointer and XInclude. The internal document representation is as close as possible to the DOM (Document Object

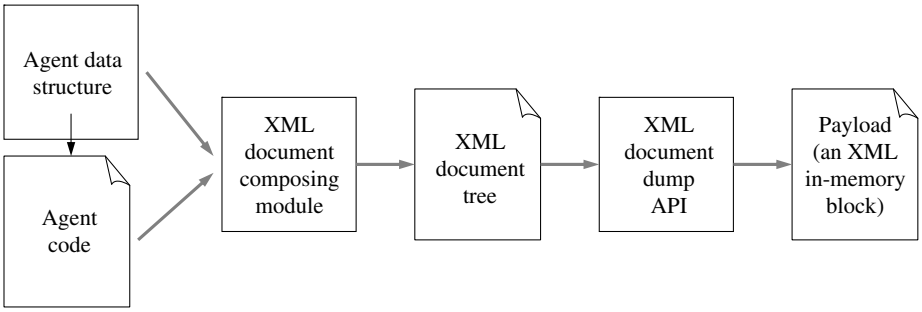

Fig. 7. Composing a mobile agent message from the mobile agent structure.

Model) interfaces. It also has a SAX (Simple API for XML) like interface. The wide range support of XML related standards and interfaces allows systems be extended with additional capabilities for different applications.

Libxml2 has a set of routines and structures for manipulating an XML tree in memory. When a mobile agent transport-message is sent to a new host, the payload of the message that is encoded in XML and contains the data state and code of a mobile agent is stored in memory. In order to retrieve mobile agent data state and code, we use parser API (Application Programming Interface) to parse this XML in memory block and build a corresponding XML document tree. After we create a XML document tree, the XML processing module traverses all the child nodes of the root node in the resulting document tree and extracts mobile agent data state and code for generating a mobile agent structure and saving mobile agent code to a memory block or a file. The implementation of XML processing module is based on libxml2 tree interface for tree manipulation.

The reverse process occurs when a mobile agent is about to migrate to a new host. The XML Document Composing Module creates an XML document tree according to the mobile agent data structure and mobile agent code. And then, the resulting XML document tree is dumped into memory for constructing a mobile agent message as shown in Fig. 7.

6. An example of a mobile agent visiting remote hosts and processing XML data interpretively on remote hosts

One of the advantages of mobile agents is able to migrate to different hosts to perform tasks based on resources available in remote hosts. The purpose of the mobile agent in this simulation example is to access XML vehicle information data files at geographically distributed locations and process XML data interpretively. The XML data files store vehicle information from a laser-based highway vehicle detection system (Cheng et al., 2001; Cheng et al., 2005).

A mobile agent dispatched by an agency in the host *bird1* visits remote host *iel2* and *ch*. Fig. 8 shows part of the mobile agent message sent from host *bird1* to host *iel2*. The mobile agent message contains the general information of the agent, including agent name, agent owner, and home agent platform. The agent tasks are described within the *TASK* tag. The *task* attribute of the *TASK* element specifies how many tasks the mobile agent has. The *num* attribute indicates how many tasks have been completed. The *DATA* element contains agent data, code, and return results related to each task. The sub-element *DATA_ELEMENT* contains the return data

```
<NAME> mobagent </NAME>
<OWNER> IEL </OWNER>
<HOME> bird1.engr.ucdavis.edu:5125 </HOME>
<TASK  task = "2" num = "0">
  <DATA number_of_elements = "0" name = "results_iel2"
        complete = "0" server  = "iel2.engr.ucdavis.edu:5138">
    <DATA_ELEMENT> </DATA_ELEMENT>
    <AGENT_CODE>
      Mobile agent code on iel2.
    </AGENT_CODE>
  </DATA>
  <DATA number_of_elements = "0" name = "results_ch"
        complete = "0" server = "ch.engr.ucdavis.edu:5135">
    <DATA_ELEMENT> </DATA_ELEMENT>
    <AGENT_CODE>
      Mobile agent code on ch.
    </AGENT_CODE>
  </DATA>
</TASK>
```

Fig. 8. The content of the mobile agent message sent from host *bird1* to host *iel2*.

from the task. The sub-element*AGENT_CODE* includes mobile agent code to be executed in a remote host. The mobile agent in this example has two tasks and no task has been completed at this moment. The name of return data array is *results_iel2* for *iel2* host and *results_ch* for *ch* host, respectively. Two hosts that the mobile agent is going to visit are *iel2.engr.ucdavis.edu* and *ch.engr.ucdavis.edu*.

The task of the mobile agent on *iel2* machine is to access an XML hourly vehicle information file listed in Fig. 9, and find the average vehicle speed at this station. The hourly vehicle information file records the number of vehicles passed through the detection station *iel2* within an hour and the average speed of vehicles. Program 1 lists a code fragment of the mobile agent running on the *iel2* machine. Function *parseNode( )* is a typical C XML processing program. It can be executed interpretively without the need of compilation in our system. The function searches each node of the XML hourly vehicle information file and retrieves the value of average speed element. This value is then saved in the array *results_iel2*. Array *results_iel2* is an array in the mobile agent space (Ch space), which is used to store the result obtained by the mobile agent. To save the result into mobile agent structure, the return data stored in the array *results_iel2* is passed to the agent platform space (C/C++ space) by Ch SDK API. The mobile agent also displays the average speed on the *iel2* machine as shown in Fig. 10.

```
<?xml version="1.0"?>
<!DOCTYPE traffic-record SYSTEM "Mobile-C.dtd">

<traffic-record>
  <year>2006</year>
  <month>02</month>
  <day>18</day>
  <start-time>10:00</start-time>
  <number-of-vehicles>1000</number-of-vehicles>
  <average-speed>63</average-speed>
</traffic-record>
```

Fig. 9. The content of an XML traffic information data file.

```
int parseNode(xmlDocPtr doc, xmlNodePtr cur){
  static int i;
  i++;
  while (cur != NULL) {
    if(cur->type == XML_ELEMENT_NODE){
      if(!(xmlStrcmp(cur->name, (const xmlChar *) "average-speed"))) {
        results_iel2[1] = atof(xmlNodeListGetString(doc,
                               cur->xmlChildrenNode, 1));
        printf("The average speed at the iel2 detection station is
                %f.\n", results_iel2[1]);
      }
      parseNode(doc, cur->xmlChildrenNode);
    }
    cur = cur->next;
  }
  i--;
  return 0;
}
```

Program 1. A code fragment of the mobile agent performing task on the *iel2* host.

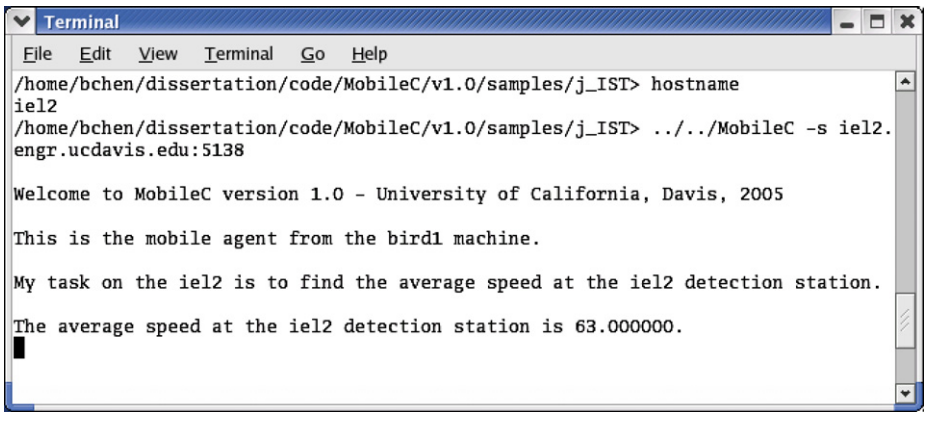

Fig. 10. The output of the mobile agent on the *iel2* host.

```
<NAME> mobagent </NAME>
<OWNER> IEL </OWNER>
<HOME> bird1.engr.ucdavis.edu:5125 </HOME>
<TASK  task = "2" num = "1">
  <DATA number_of_elements = "2" name = "results_iel2"
        complete = "1" server  = "iel2.engr.ucdavis.edu:5138">
    <DATA_ELEMENT> 63.000000 </DATA_ELEMENT>
    <AGENT_CODE>
      Mobile agent code on iel2.
    </AGENT_CODE>
  </DATA>
  <DATA number_of_elements = "0" name = "results_ch"
        complete = "0" server = "ch.engr.ucdavis.edu:5135">
    <DATA_ELEMENT> </DATA_ELEMENT>
    <AGENT_CODE>
      Mobile agent code on ch.
    </AGENT_CODE>
  </DATA>
</TASK>
```

Fig. 11. The content of the mobile agent message from host *iel2* to host *ch*.

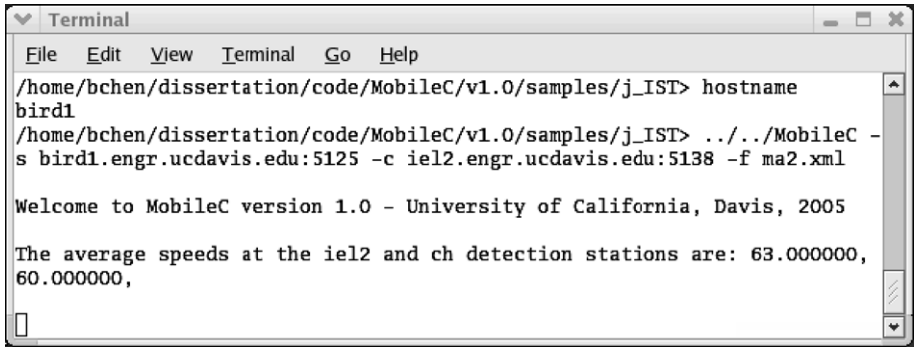

Fig. 12. The results of the mobile agent are sent back to the home agency.

After visiting *iel2* station, the mobile agent visits *ch* station. The result obtained from the *iel2* station is also sent to the *ch* station with the mobile agent message. The content of the mobile agent message sent from host *iel2* to host *ch* is shown in Fig. 11. Since the mobile agent has completed the task on *iel2*, the *num* attribute of *TASK* element has been changed to 1, i.e., one task has been finished. In addition, the return data from the host *iel2*, which is 63.000000, has been included in the *DATA_ELEMENT* element of task 1.

The task of the mobile agent on *ch* machine is the same as that on *iel2* machine excepting processing an XML data file in *ch* machine. The results of the mobile agent obtained from both *iel2* and *ch* stations are sent back to the home agency *bird1* and displayed in a terminal of bird1 machine as shown in Fig. 12.

7. Performance evaluation

A number of researchers have studied mobile agent performance in different aspects. Peine (2002) compared mobile agent approach with an equivalent stationary implementation in a distributed searching application. Johansen (1998) compared mobile agent and client/server implementations in image and video processing applications, and Gray et al. (2002) compared the scalability of these two paradigms. Spyrou et al. (2004) qualitatively and quantitatively analyzed a set of software models built on client/server model or mobile agents for accessing a Web server.

As we described in Section 2, using XML to represent different types of information in mobile agent systems has a number of advantages. However, a high-level XML data representation will inevitably introduce performance overhead. To evaluate the performance overhead of XML representation, a comparison test has been conducted to compare times needed for a mobile agent to process a set of traffic records. Traffic records contain the information of each passed vehicle, including date, time, and vehicle speed. These traffic records are stored in data files, SQL databases, and XML files. Fig. 13 shows traffic records in a database, and Fig. 14 shows traffic records in an XML file.

Fig. 15 shows a comparison chart of the processing time that a mobile agent needs to find the average speed of vehicles passed through a detection station. When traffic data are stored in data files, a mobile agent reads the speed of each vehicle and calculates the average speed using standard C functions. For SQL database format, a mobile agent accesses a vehicle information database through interpretive ODBC, Ch ODBC. The mobile agent first initiates an ODBC driver, then connects to the database, and

Year	Month	Day	Hour	Minute	Second	Speed
2006	03	08	10	00	01	63.35
2006	03	08	10	00	15	65.55
2006	03	08	10	00	31	67.02
2006	03	08	10	00	55	64.87
2006	03	08	10	01	21	66.73

Fig. 13. A traffic record database.

```
<?xml version="1.0"?>
<!DOCTYPE traffic-record SYSTEM "Mobile-C.dtd">

<traffic-record>
  <year>2006</year>
  <month>03</month>
  <day>08</day>
  <vehicle-speed time-passed="10:00:01">63.35</vehicle-speed >
  <vehicle-speed time-passed="10:00:15">65.55</vehicle-speed >
  <vehicle-speed time-passed="10:00:31">67.02</vehicle-speed >
  <vehicle-speed time-passed="10:00:55">64.87</vehicle-speed >
  <vehicle-speed time-passed="10:01:21">66.73</vehicle-speed >
</traffic-record>
```

Fig. 14. A traffic record XML file.

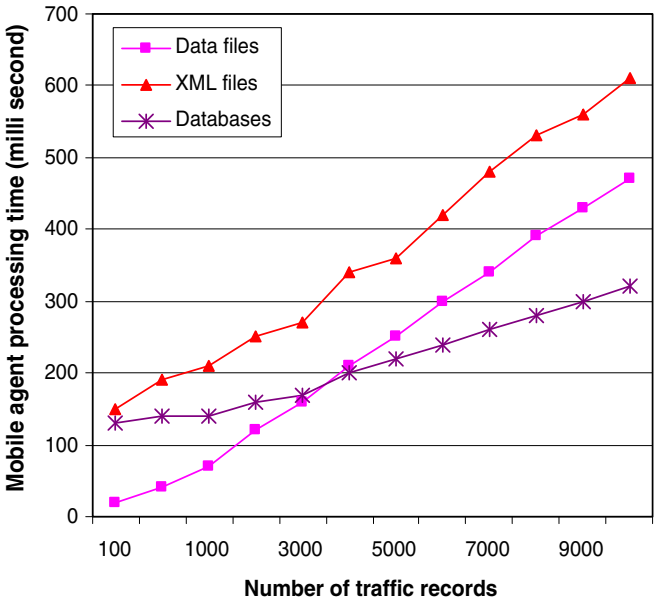

Fig. 15. A comparison of mobile agent processing time.

last performs data queries to get vehicle speeds. When traffic data are stored in XML files, a mobile agent builds an XML tree from the XML file and goes through each node to read vehicle speeds. Comparing with XML files and databases, data file format takes less time when the number of traffic records is small. This is because a mobile agent doesn't need to initiate an ODBC driver and connect to the database or build an XML tree when the traffic data are stored in a data file. However, the mobile agent processing times for data file format increase rapidly when the number of records increases. As a result, database format takes less time than data files when the number of traffic records is large.

Although XML data representation takes longer processing time comparing to regular data representations as shown in Fig. 15, the performance overhead is not significant. Moreover, we can expect a similar performance as SQL databases when using XML databases store XML data information. Most likely, the processing time for a huge set of XML data in an XML database will be less than that for the same size data set stored in a data file.

8. Conclusions

This article presents an XML-based approach for inter-agent communication, inter-platform mobile agent migration, and information representation in mobile agent systems. Our experience shows that using XML to encode different types of messages, including simple agent communication messages and messages containing mobile agents, is simple, convenient, and easy to change. The exploration of using mobile agents to process XML data interpretively opens the door to simplify the architecture of a mobile agent system for building a programmable information base in XML. Since mobile agents are able to read and manipulate XML data directly, other types of information in agent systems, such as raw data and access control policies, can be represented in XML also. The performance evaluation shows that the performance overhead of XML data representation is not significant, and is acceptable for the systems where the benefits of XML are more important than a high-performance. Since the database format presents a good performance for a large size of data source, further exploration of using XML databases to store XML data will be worthwhile.

References

Ametller, J., Robles, S., Borrell, J., 2003. Agent migration over FIPA ACL messages. Mobile Agents for Telecommunication Applications, Proceedings 2881, 210–219.

Cabri, G., Leonardi, L., Zambonelli, F., 2001. XML dataspaces for the coordination of Internet agents. Applied Artificial Intelligence 15, 35–58.

Ch – An Embeddable C/C++ Interpreter. Softintegration, Inc. <http://www.softintegration.com/>.

Chen, B., 2005. Runtime support for code mobility in distributed systems. In: Department of Mechanical and Aeronautical Engineering. Ph.D. Dissertation, University of California, Davis.

Chen, B., Cheng, H.H., 2005. Interpretive OpenGL for computer graphics. Computers & Graphics 29, 331–339.

Chen, B., Cheng, H.H., Palen, J., 2004. Agent-based real-time computing and its applications in traffic detection and management systems. In: the ASME 24th Computers and Information in Engineering Conference. Salt Lake City, Utah, pp. 543–552.

Chen, B., Cheng, H.H., Palen, J., 2006. Mobile-C: a mobile agent platform for mobile C/C++ agents. Software-Practice & Experience 36, 1711–1733.

Cheng, H.H., 1993. Scientific computing in the Ch programming language. Scientific Programming 2, 49–75.

Cheng, H.H., Trang, D.T., 2006. Object-oriented interactive mechanism design and analysis. Engineering with Computers 21 (MAY), 237–246.

Cheng, H.H., Shaw, B.D., Palen, J., Larson, J.E., Hu, X.D., Van Katwyk, K., 2001. A real-time laser-based detection system for measurement of delineations of moving vehicles. IEEE/ASME Transactions on Mechatronics 6, 170–187.

Cheng, H.H., Shaw, B.D., Palen, J., Lin, B., Chen, B., Wang, Z.Q., 2005. Development and field test of a laser-based nonintrusive detection system for identification of vehicles on the highway. IEEE Transactions on Intelligent Transportation Systems 6 (June), 147–155.

Ch ODBC. <http://www.softintegration.com/products/toolkit/odbc//>.

Chong, C., Jiwen, H., Kai, B., Zhongfan, M., 1999. Mobile software agent model and the architecture of JMSAS system. In: The First International Workshop on Mobile Agent for Telecommunication Applications. Ottawa, Canada, pp. 37–52.

Ch XML Package for Libxml2. <http://chlibxml2.sourceforge.net//>.

De Meo, P., Rosaci, D., Sarne, G.M.L., Terracina, G., Ursino, D., 2003. An XML-based adaptive multi-agent system for handling e-commerce activities. Web Services-ICWS-Europe (Lecture Notes in Computer Science), vol. 2853. Springer-Verlag, Berlin, Germany, pp. 152–166.

FIPA. The Foundation for Intelligent Physical Agents. <http://www.fipa.org/>.

FIPA, FIPA Agent Management Specification. <http://www.fipa.org/specs/fipa00023/SC00023K.html/>.

Glushko, R.J., Tenenbaum, J.M., Meltzer, B., 1999. An XML framework for agent-based E-commerce. Communications of the Acm 42 (March), 106–114.

Gray, R.S., Cybenko, G., Kotz, D., Peterson, R.A., Rus, D., 2002. D'Agents: applications and performance of a mobile-agent system. Software-Practice & Experience 32, 543–573.

Griss, M.L., 2001. Software agents as next generation software components. In: Heineman, G.T., Councill, W.T. (Eds.), Component-based Software Engineering: Putting the Pieces Together. Addison-Wesley, Boston.

Grosof, B.N., Labrou, Y., 1999. An Approach to using XML and a Rule-based Content Language with an Agent Communication Language. Technical Report, IBM Research Division.

Johansen, D., 1998. Mobile agent applicability. In: Mobile Agents: Second International Workshop, MA'98 (Lecture Notes in Computer Science), vol. 1477. Springer, Berlin/Heidelberg, pp. 80–98.

Lange, D.B., Oshima, M., 1999. Seven good reasons for mobile agents. Communications of the Acm 42 (March), 88–89.

Leung, E.W.C., Li, Q., 2004. XML-based agent communication in a Distributed Learning Environment. Advances in Web-Based Learning – ICWL 2004 (Lecture Notes in Computer Science), vol. 3143. Springer-Verlag, Berlin, Germany, pp. 136–146.

Libxml2. The XML C parser and toolkit of Gnome. <http://www.xmlsoft.org/index.html/>.

Madjarov, I., Boucelma, O., Betari, A., 2004. An agent- and service-oriented e-Learning platform. In: Advances in Web-Based Learning – ICWL 2004, vol. 3143, pp. 27–34.

Mathieu, P., Verrons, M., 2003. A generic negotiation model for MAS using XML. In: The IEEE International Conference on Systems. Man and Cybernetics, pp. 4262–4267.

Mobile-C, a Multi-Agent Platform for Mobile C/C++ Agents. <http://www.mobilec.org//>.

Oracle XML Developer's Kit. <http://www.oracle.com/technology/tech/xml/xdkhome.html/>.

Overview of SGML Resources. <http://www.w3.org/MarkUp/SGML//>.

Peine, H., 2002. Application and programming experience with the Ara mobile agent system. Software-Practice & Experience 32, 515–541.

Sedbrook, T., 2001. Integrating e-business XML business forms and rule-based agent technologies. Expert Systems 18, 250–260.

Spyrou, C., Samaras, G., Pitoura, E., Evripidou, P., 2004. Mobile agents for wireless computing: the convergence of wireless computational models with mobile-agent technologies. Mobile Networks & Applications 9 (October), 517–528.

Wang, Z.Q., Cheng, H.H., 2006. Portable C/C++ code for portable XML data. IEEE Software 23 (JAN-FEB), 76–81.

World Wide Web. <http://www.w3.org//>.

Yu, Q.C., Cheng, H.H., Cheng, W.W., Zhou, X.D., 2004. Ch OpenCV for interactive open architecture computer vision. Advances in Engineering Software 35 (August–September), 527–536.

Yu, Q.C., Chen, B., Cheng, H.H., 2004. Web-based control system design and analysis – design, implementation, and salient features. IEEE Control Systems Magazine 24 (June), 45–57.

Bo Chen received the B.S. and M.S. degrees in electrical engineering from the Zhejiang Sci-Tech University, China, and the Ph.D. degree in mechanical and aeronautical engineering from the University of California, Davis in 2005.

She is currently an Assistant Professor in the Department of Mechanical Engineering – Engineering Mechanics at Michigan Technological University. Her research interests include mobile agent and multi-agent systems, intelligent distributed control, sensor network, and intelligent transportation systems.

David D. Linz obtained a B.S degree in mechanical engineering from the University of California, Davis in 2006. He worked as an undergraduate research assistant at the Integration Engineering Laboratory at UC Davis from 2004 to 2006. He currently is a graduate student with a research interest in computational and spectral methods for fluid dynamics at the University of Michigan in Ann Arbor.

Harry H. Cheng received the M.S. degree in mathematics and the Ph.D. degree in mechanical engineering from the University of Illinois at Chicago in 1986 and 1989, respectively.

Before joining the faculty at the University of California, Davis in 1992, he worked as a Senior Engineer on information-driven Systems at the Research and Development, United Parcel Service, Inc. from 1989 to 1992. Currently, he is a Professor and Director of the Integration Engineering Laboratory at the University of California, Davis. He is also a member of Graduate Group in Computer Science and Graduate Group in Electrical and Computer Engineering at UC Davis. He is the chief architect of an embeddable C/C++ interpreter Ch for script computing, which is being widely used in both academia and industry. He holds one U.S. patent and has published over 120 papers in refereed journals and conference proceedings. His current research interests include mobile agent systems, intelligent mechatronic and embedded systems, computer-

aided design and manufacturing, robotics, and intelligent transportation systems.

Dr. Cheng received a Research Initiation Award from the National Science Foundation, the Best Paper Award at the IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, the Procter and Gamble Best Paper Award as well as the Waldron Award at the Applied Mechanisms and Robotics Conferences. He received an Outstanding Contribution Award from United Parcel Service, Inc. He participated in revision of the latest C standard called C99 through ANSI X3J11 and ISO S22/WG14 C Standard Committees. He is a Fellow of ASME and a Senior Member of IEEE, IEEE Computer Society, and IEEE Robotics and Automation Society. He is the Chair of the ASME Technical Committee on Mechatronic and Embedded Systems and Applications. He is also the Chair of the Technical Committee on Mechatronic and Embedded Systems in ITS of the IEEE Intelligent Transportation Systems Society. He is the Conference Chair of the 2008 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications. He was the Chair of the Technical Area of Embedded and Ubiquitous Computing and Co-Chair of Technical Area of Computers in Electromechanical Systems in the ASME Division of Computers and Information in Engineering. He served as the General Co-Chair of the 2007ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications, the Program Chair of the 2006 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications.