LOGIC-BASED MOBILE AGENT FRAMEWORK USING WEB TECHNOLOGIES

Shinichi Motomura, Takao Kawamura, Kazunori Sugahara
Tottori University
4–101, Koyama-Minami, Tottori 680–8552, JAPAN
Email: motomura@tottori-u.ac.jp, {kawamura,sugahara}@ike.tottori-u.ac.jp

Keywords: Mobile agent, Logic Programming language, XML-RPC.

Abstract: We have proposed Maglog which is a framework for mobile multi-agent systems. Maglog is based on Prolog, and has the concept of field. A field is an object which can contain a knowledge base. With the concept of field, Maglog provides a simple and unified interface for 1)inter-agent communication, 2)agent migration between computers, and 3)utilization of data and programs on computers. An agent migrates using HTTP as the transport protocol and XML as the encoding format itself. In this paper, we present the implementation of Maglog on Java environment, in detail. Since we have implemented both command-line shell and GUI for Maglog, users can choose them for their needs. In addition, through XML-RPC interface for Maglog which we have also implemented, other systems can easily utilize Maglog. As examples, we outline several applications developed through XML-RPC interface.

1 INTRODUCTION

Multi-agent system is drawing attention as a structural model for many software systems including distributed systems and artificial intelligence systems. In a multi-agent system, a number of autonomous agents cooperates mutually and achieves given tasks. Each agent is generated according to a given task and can have its own situation and operates under the situation. Situation consists of states and procedures where both of them can be dynamically changed in general. Therefore, it becomes necessary for the agent to dynamically hold the states and procedures (hereafter we refer them as a knowledge base). Moreover, when a number of agents are cooperating, it is necessary for them to share knowledge bases and to conduct knowledge communications between agents. In addition, mobility of agents is important in multi-agent system because of not only reducing network latency but simplifying architecture of software systems.

We have proposed Maglog (Kawamura et al., 2003; Kawamura et al., 2004a) which is a framework for mobile multi-agent systems. Maglog is based on Prolog, and has the concept of field. A field is an object which can contain a knowledge base. With the concept of field, Maglog provides a simple and unified interface for 1)inter-agent communication, 2)agent migration between computers, and 3)utilization of data and programs on computers. In this paper, we present the implementation of Maglog on Java environment, in detail.

2 OVERVIEW OF MAGLOG

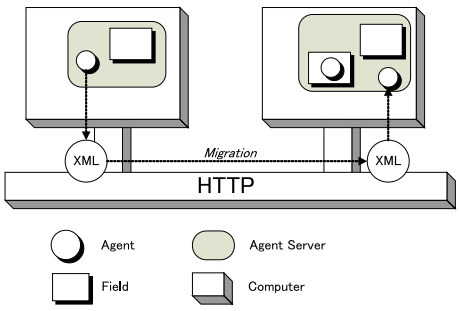

Figure 1: Overview of Maglog.

Figure 1 shows the overview of Maglog. An agent runs as a thread in a process which we call an agent server. Mobile agents of Maglog are written in Prolog. Agent servers have objects which hold Pro-

log clauses. We call them fields. Builtin predicate `in(Goal, Field)` is for evaluation of a goal in a field. An agent can enter a field by this predicate. Entering a field, an agent can utilize data and programs in the field. With the concept of field, Maglog provides a simple and unified interface for 1)inter-agent communication, 2)agent migration between computers, and 3)utilization of data and programs on computers. An agent migrates using HTTP as the transport protocol and XML as the encoding format itself.

2.1 Inter-agent Communication

Agents belonging to the same field can be considered of forming a group. The knowledge within the field is shared by the agents. Moreover, by changing the knowledge within the field, agents can influence the actions of other agents.

An agent can communicate with other agents synchronously or asynchronously by reading/writing Prolog clauses from/into fields. Updating knowledge base in a field can be done by the following predicates.

2.2 Migration

The second arguments of the following predicates can be like `FieldName@HostAddress`.

```
in(Goal, Field)
fasserta(Clause, Field)
fassertz(Clause, Field)
fretract(Clause, Field)
fclause(Head, Body, Field)
```

If a host address specified, the agent will go to the host and access the field.

2.3 Dynamic Change of Behavior

An agent can change its behavior dynamically through entering a field. Figure 2 shows an example. The execution of the goal `print('Hello!')` sends the string "Hello!" to a printer when the agent is in fieldA, on the other hand, the same goal creates a new window containing the string "Hello!" when the agent is in fieldB.

3 IMPLEMENTATION

We have implemented Maglog on Java environment through extending PrologCafé which is a Prolog-to-Java source-to-source translator system (Banbara and Tamura, 1999). Both agents and after-mentioned static fields are translated into *.java files with our Maglog translator and then compiled into *.class files with a Java compiler.

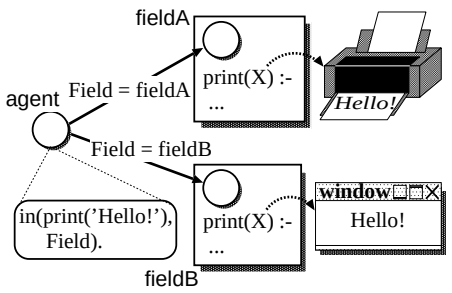

Figure 2: Dynamic Change of agent's behavior through entering a field.

Two functions for agent migration have been implemented: a serialize function for agent and a RPC function. These functions have been implemented by XML-RPC to agent servers. We show the migratation steps below.

1. An agent server encodes a agent to a XML document.
2. The agent server gets a reference to a destination agent server using XML-RPC.
3. The agent server invokes the destination agent server's RPC with the XML document. The RPC uses HTTP as transfer protocol.
4. The invoked agent server decodes the agent from a XML document and continues execution of the agent.
5. The agent server makes up to a XML document from the execution result and returns the XML document.

3.1 Basic Components

In this section, we describe the Java implementation of agent, agent server, and field.

1. Agent

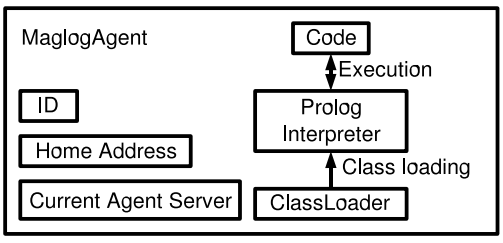

Figure 3: Structure of an agent.

Figure 3 shows the structure of an agent. An agent executes predicates using Prolog Interpreter and

moves between agent servers. An agent has parent-and-child structure and the child agent has the address for returning to the parent agent as HomeAddress.

2. Agent Server

   Figure 4 shows the structure of an agent server. An agent server creates an agent by `createAgent` method. `createAgent` method creates an agent from the agent repository, which contains classes of predicates of agents, and pushes into the agent scheduler. The agent scheduler has threads for agents. When `receiveAgent` method is invoked, the `MaglogAgentRemoteServer` receives a agent via network, decodes it from XML, and pushes into the agent scheduler. When `sendAgent` method is invoked, the MaglogAgentRemoteServer encodes a agent to XML, and sends to a destination agent server via network. These transfer protocols are HTTP.

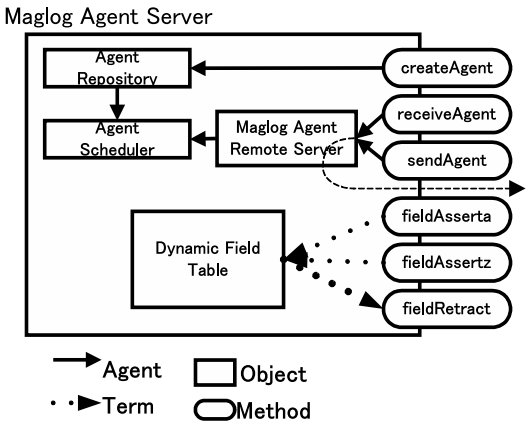

Figure 4: Structure of an agent server.

3. Field

   We have implemented two kinds of field, dynamic field and static field. Static fields are pre-compiled predicates. Therefore agents cannot modify static fields, however executions of predicates in static fields are relatively fast. Executions of predicates in dynamic fields are relatively slow however agents can assert and retract clauses in dynamic fields.

3.2 Web Services Function

Agent Servers have Web Services are implemented using XML-RPC. XML-RPC is a simple remote procedure calling protocol using XML as the encoding format (Winer, 1998). For this function, it becomes easy to use Maglog as a part of application.

Through XML-RPC, other systems can do the following operations.

- create and kill agents.
- create and delete dynamic fields.
- assert and retract clauses in dynamic fields.
- get a list of names of dynamic fields.
- get a list of IDs of agents.

Prolog clauses in return values and arguments of requests are translated to data types of XML-RPC by the agent server.

4 APPLICATIONS

We will outline several applications developed using Maglog.

1. e-Learning System (Kawamura et al., 2004b)

   An P2P-based e-Learning system has been built using Maglog. This e-Learning system has two distinguishing features. Firstly, it is based on P2P architecture for scalability and robustness. Secondly, each content in the system is not only data but an agent so that it can mark user's answers, tell the correct answers, and show some extra information without human instruction. Maglog plays an important role to realize the both features. Figure 5 is a screen-shot of the user interface program. of this e-Learning system, which is developed in Squeak environment. This e-Learning system consists of about 2,000 lines of Maglog code and about 4,000 lines of Squeak code.

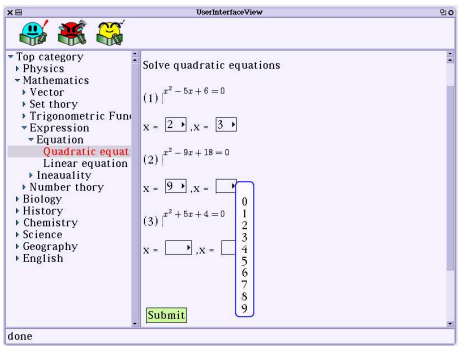

Figure 5: The e-Learning system.

2. Schedule Arrangement System (Kinosita et al., 2003)

   This system, which has been developed using Maglog, establishes and arranges meeting schedule without human negotiations. Once a convener convenes a meeting through the system, an agent moves around the members of the meeting and negotiates with them automatically. This schedule arrangement system consists of about 400 lines of Maglog code and about 4,000 lines of Java code.

3. HECS System (Banbara et al., 2003)

   Maglog is used for HECS (Heterogeneous Constraint Solving) system, which was supported in part by IPA (The Information-technology Promotion Agency) under grant of 2003 Exploratory Software Project, to coordinate distributed solvers each other.

5 RELATED WORKS

There are several mobile agent frameworks realized as a set of class libraries for Java such as Aglets (Lange and Oshima, 1998) and MobileSpaces (Satoh, 2000). The combination of one of them and a Prolog interpreter/compiler written in Java such as NetProlog (de Carvalho et al., 1999) and Jinni (Tarau, 1999) have some similarity to Maglog. The main difference between the combination and Maglog is the class of mobility. Their mobility is so-called weak mobility, in which only its clause database is migrated. On Maglog, all of the execution state including execution stack can be migrated (so-called strong mobility), therefore agents on Maglog can backtrack and unify variables during migration. That makes programs on Maglog simple and understandable.

6 CONCLUSION

In this paper, we presented the implementation of a mobile agent framework Maglog, in detail. With the concept of field, Maglog provides a simple and unified interface for 1)inter-agent communication, 2)agent migration between computers, and 3)utilization of data and programs on computers. And through XML-RPC interface which we have implemented, other systems can easily utilize Maglog. Through XML-RPC interface, several applications have been developed.

REFERENCES

Banbara, M. and Tamura, N. (1999). Translating a linear logic programming language into Java. In M.Carro, I.Dutra, et al., editors, *Proceedings of the ICLP'99 Workshop on Parallelism and Implementation Technology for (Constraint) Logic Programming Languages*, pages 19–39.

Banbara, M., Tamura, N., Inoue, K., Kawamura, T., and Tamaki, H. (2003). Java implementation of a distributed constraint solving system. Exploratory software project, Information-technology Promotion Agency Japan.

de Carvalho, C. L., Pereira, E. C., and da Silva Julia, R. M. (1999). Netprolog: A logic programming system for the java virtual machine. In *Proceedings of the 1st International Conference on Enterprise Information Systems*, pages 591–598. Setubal, Portugal.

Kawamura, T., Kinoshita, S., and Sugahara, K. (2004a). Implementation of a mobile agent framework on java environment. In Gonzalez, T., editor, *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems*, pages 589–593. MIT, Cambridge, USA.

Kawamura, T., Kinoshita, S., and Sugahara, K. (2004b). A mobile agent-based p2p e-learning system. In Gonzalez, T., editor, *Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems*, pages 873–877. MIT, Cambridge, USA.

Kawamura, T., Kinoshita, S., Sugahara, K., and Kuwatani, T. (2003). A logic-based framework for mobile multiagent systems. In Hexmoor, H., editor, *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 754–759. Boston, Massachusetts, USA.

Kinosita, S., Kawamura, T., and Sugahara, K. (2003). Mobile agent based schedule arrangement system. In *Proceedings of the 5th IEEE Hiroshima Student Symposium (HISS)*, pages 205–206.

Lange, D. B. and Oshima, M. (1998). *Programming and Deploying Java Mobile Agents with Aglets*. Addison Wesley.

Satoh, I. (2000). Mobilespaces: A framework for building adaptive distributed applications using a hierarchical mobile agent system. In *Proceedings of IEEE International Conference on Distributed Computing Systems*, pages 161–168. IEEE Press.

Tarau, P. (1999). Inference and computation mobility with jinni. In Apt, K., Marek, V., and Truszczynski, M., editors, *The Logic Programming Paradigm: a 25 Year Perspective*, pages 33–48. Springer.

Winer, D. (1998). Xml-rpc specification. http://xmlrcp.com/spec.