The Communication Infrastructure of the MAPNET Mobile-Agent Platform

Dilyana Staneva, Petya Gacheva

Abstract: *This paper describes the communication infrastructure of the MAPNET platform: our .NET-based implementation of the general mobile agent-based model for organizing distributed computations. The paper highlights key design decisions. It discusses the functions of the MAPNET communication service and their current implementation, focusing on its security mechanisms based on cryptographic keys. Directions for future work are outlined.*

Keywords: *distributed computing, mobile agent-based computing, cryptography, .NET Remoting*

INTRODUCTION

Mobile agent-based computing is an attractive, though not widely accepted model for structuring distributed solutions [3]. The most distinctive feature of this model is the mobile agent: a migrating entity, with the capability to transfer its current state and code to a different network location. Compared to remote communication, migration could reduce network traffic. Furthermore, mobile agents can function independently of their dispatching host and contact it later only to return a small set of results. Relevant application domains for mobile agents are distributed information retrieval, monitoring and filtering.

There are several reasons for the quite limited acceptance of the mobile agent technology. First, it's quite difficult to identify a distributed problem whose solution can be based on mobile agents only, instead of an equivalent or even better "classical" message-passing or Web Services solution. Another major concern is security: how to protect agents and servers from one another. Nevertheless, mobile agent-based computing, being high-level and flexible, can be a useful tool in rapid prototyping. Due to its high level of abstraction and ease of use, it can also be applied as a teaching tool in introducing students to distributed computing.

We've recently developed a .NET-based implementation of the general mobile agent-based model, following the MASIF (Mobile Agent System Interoperability Facilities) specification [4]. The majority of today's mobile-agent platforms are Java-based [2, 3, 6]. Considering the diversity of available platforms, the MASIF specification focuses on their interoperability. It defines two main interfaces: of an agent server (MAFAgentSystem) and of a naming and lookup service (MAFFinder). MASIF does not directly address the communication infrastructure of mobile-agent platforms. It covers security and agent migration aspects, while remote inter-agent communication, though supported by mobile-agent platforms, is not considered mobile-agent specific.

Our implementation is written in C#, in the form of a type library for representing an agent server and mobile agents, using the Microsoft .NET Platform [5] class library and targeting the .NET CLR (Common Language Runtime) (Fig.1).

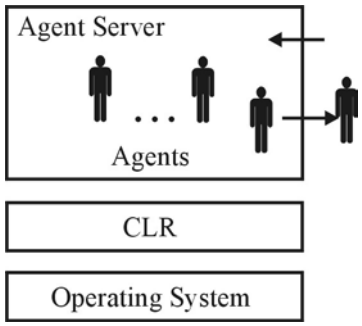

Fig.1 MAPNET components

Each mobile agent is executed in the context of an agent server. The MAPNET server is multi-threaded, with a separate thread for each agent.

The communication service is a core service of the MAPNET server, together with the thread manager and local registration service. It is responsible for all remote interactions: agent migration and remote communication between agent servers and agents.

When developing the communication service, we had to make two key decisions:

- Which of the available components of the .NET Internet technology stack to use?
- How to embed security into the communication infrastructure, making it transparent for the mobile-agent programmer?

A further requirement to the communication service is to support both synchronous and asynchronous interactions.

The .NET stack of Internet technologies is 3-leveled, as illustrated in Fig.2.

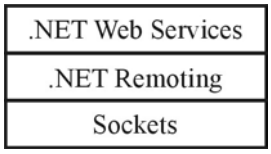

Fig.2 .NET Internet Technologies

In general, higher-level technologies are easier to use, while lower-level offer better performance and flexibility. We consider .NET Remoting the proper level of abstraction and we've built our current implementation on top of it. .NET Web Services are typically hosted by Internet Information Services, and using them as the underlying infrastructure could restrict the hosts of our platform. The only assumption we make is that all communicating parties are based on the .NET CLR. Compared to Web Services, .NET Remoting has broader scope: it provides the infrastructure for using remote objects, not only calling remote methods. .NET Remoting allows the integration of custom security mechanisms.

Some popular mobile-agent platforms implement more than one communication mechanisms. Grasshopper [2], for example, supports IIOP, Java RMI and sockets. Communication can be secured by SSL, but as an add-on to the platform.

REMOTE COMMUNICATION

The MAPNET communication service (Fig. 3) is based on .NET Remoting, which hides lower-level communication and serialization details.

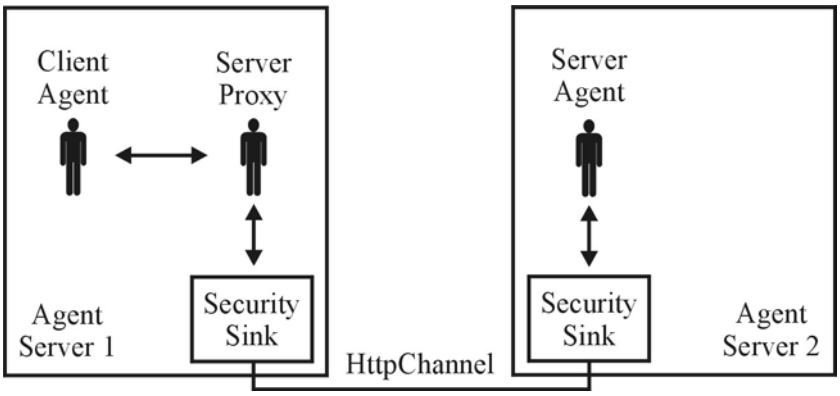

Fig.3 MAPNET communication infrastructure

Two separate parties are distinguished in remote access scenarios: a local client object and a remote server object, which exposes remotely callable methods. Each MAPNET server can be both a client and a server when communicating with other servers. The basic components of the .NET Remoting infrastructure include the following:

- Communication channels: .NET Remoting provides HTTP and TCP channels. SSL is not supported.
- Formatters: Responsible for message serialization and encoding before sending to the channel, and decoding and deserialization after receiving from the channel. By default, the HTTP channel is associated with a SOAP formatter, while the TCP channel uses a binary formatter.
- Proxy: Forwards remote method invocations to the respective server object. The client object interacts entirely with the proxy. After a sequence of configuration and registration steps, remote calls are syntactically expressed like local calls.

The .NET communication channel is organized as a chain of sinks. Each channel sink can read and even manipulate messages. A channel contains at least two sinks: a formatter and a transport sink. The basic idea of integrating security mechanisms into the communication infrastructure is to create and insert custom sinks in the sink chains [1]. We've developed client and server security sinks and plugged them symmetrically between the formatter and transport sinks.

We prefer the more flexible HTTP channel in combination with a binary formatter. Compared to the default SOAP formatter, binary formatting is faster and more compact.

Another distinctive feature of our implementation is that all inter-agent remote communication is mediated by the communication service. To be able to call a method of a remote agent, an agent must first obtain a proxy to that remote agent. Access to the remote agent is provided by the remote communication service. The role of the communication service as a mediator has two main consequences. First, the agent server has greater control over its agents. And second, instead of authenticating agents, we can authenticate agent servers. In our implementation, mobile agents assume the identity of their hosting server. Authentication applies to both migration and remote communications.

The main functions of the MAPNET communication service can be summarized as follows:

- Obtain a proxy to a remote communication service,
- Provide a proxy to a remote object,
- Transmit an agent to another server, and
- Receive a migrating agent.

Suppose, agent **A** wants to call method **M** of agent **B**. Agent **B** currently resides in **locationB**. Agent **A** knows its name: **nameB**. From the developer's perspective, this remote interaction can be expressed like this:

```
ClassB b = (ClassB) this.AgentSystem.GetAgent(locationB, nameB);
result = b.M();
```

This approach hides from the developer the actual details of proxy creation. Agent **A** just calls the `GetAgent()` method of its local server through its `AgentSystem` property.

SECURITY

The MAPNET communication service operates in two modes: secure and non-secure. The security mode is stored as a property of the agent server profile. Communication between MAPNET servers residing on the same host is not secured.

The security support in MAPNET includes:

- Authentication of agent servers based on public/private keys, and
- Encryption of the messages exchanged using "session" keys.

According to the MASIF specification [4], authentication must be transparent for the agent developer, e.g. without entering a username and password. Transparency can be achieved by using cryptographic techniques. Each MAPNET agent server is assigned a pair of private and public keys. The server also maintains a list of IP addresses and public keys of "trusted" agent servers, with which it is willing to communicate.

Secure communication between agent servers **A** and **B** is organized as follows:

- Server **A** issues a request to server **B**.
- If **A** is on **B**'s "trusted" list, **B** generates a "session" key and sends it to **A**.
- Further communication between **A** and **B** is encrypted and decrypted with this "session" key.

We combine two algorithms to secure MAPNET communications: an asymmetric algorithm (RSA) to exchange only the "session" key, and a symmetric one (Rijndael) for the session. Authentication and encryption are performed by the custom security sinks, plugged into the infrastructure of .NET Remoting. Authentication-related information is carried in the message headers and manipulated by the security sinks.

Instead of authenticating each separate remote method call, we prefer to organize "sessions". A session begins on establishing a secure communication link between two agent servers, and expires after a specified timeout.

AGENT MIGRATION

Migration is the most distinctive capability of a mobile agent. From the developer's perspective, migration is expressed by simply calling the `Move()` method of an agent, as shown in the example below:

```
[Serializable]
public class MyAgent : Agent {
   public override void Run() {
      // agent's behavior:
      // local processing first,
      // then attempt to migrate
      try {
         Move(newLocation);
      }
      catch (Exception e) {
         // migration failed
      }
   }
   ...
}
```

A mobile agent in MAPNET is developed as a descendent of the abstract base `Agent` class, which provides the core mobile-agent methods `Move()` and `Run()`. The `Run()` method is executed as the "body" of the mobile agent and has to be overridden to specify the agent's behavior. The mobile agent class must be tagged as `Serializable` to enable its instances' migration.

The basic steps in implementing migration are depicted in Fig.4.

Migration is initiated by the agent itself through calling its **Move()** method, which in turn calls the **MoveAgent()** method of the agent server. The server terminates the agent and delegates the migration task to the communication service. The local communication service converts the current state of the agent into a byte stream (**SerializeAgent()**). Then, it obtains a proxy to the remote communication service and calls asynchronously its **ReceiveAgent()** method, specifying a callback method to be called when migration completes. The remote agent server deserializes the agent and assigns it a new thread.

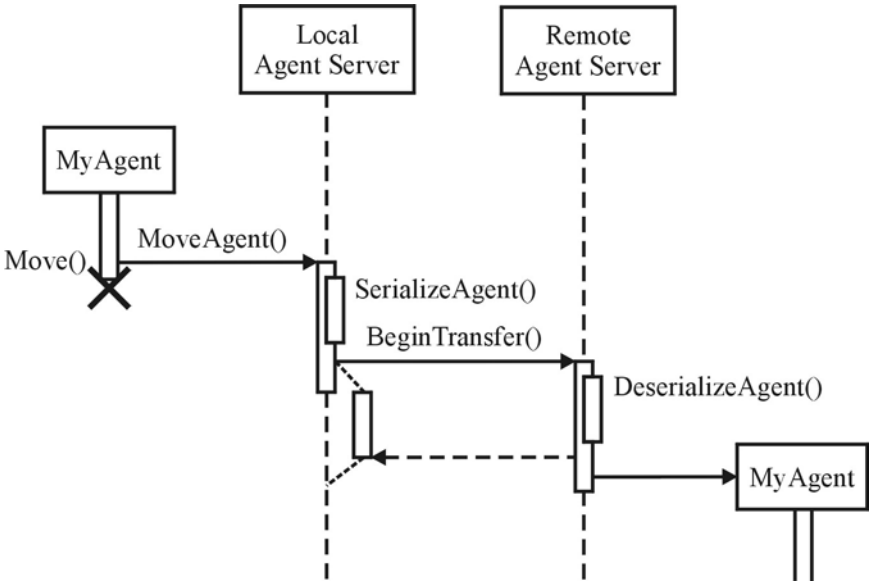

Fig.4 MAPNET communication infrastructure

Migration in MAPNET is "weak": it preserves the agent's state, determined by its serializable fields, including its profile, but not its execution state. Agent-related classes are not automatically transmitted; they are fetched on demand instead. Migration in our current implementation relies on the .NET binary serialization, to achieve type fidelity and compactness.

CONCLUSIONS AND FUTURE WORK

Developing the communication service is one of the most important and challenging tasks in building the MAPNET mobile-agent platform. Security is a major concern for mobile agent-based computing, therefore we had to provide relevant security mechanisms and implement them using the most suitable .NET resources.

We've followed the requirements of the MASIF specification in two aspects: in implementing migration, and, generally, in providing security mechanisms. The actual communication subsystem of MAPNET and the security techniques applied are implementation-specific.

Our current implementation is based on .NET Remoting, serialization and cryptographic algorithms. It provides transparent authentication of agent servers and encryption using symmetric keys. Both synchronous and asynchronous remote invocations are supported. Our implementation provides the mobile-agent specific migration, as well as higher-level remote inter-agent communication, so that the agent developer can select the better suited for his tasks.

The security mechanisms we've integrated in MAPNET could be further extended and improved in several aspects:

- Explore alternative implementations of security, based on standard protocols, like SSL, rather than custom techniques.
- Consider alternatives to .NET Remoting as the underlying infrastructure of mobile-agent interactions.
- Extend authentication to individual agents, not only agent servers.
- Implement authorization. The code-access security support of the .NET Framework could be applied to assign permissions to agents.

The MAPNET platform could be useful for developers working on related problems, as well as for those intending to use a .NET-based mobile-agent platform. Due to its high

level of abstraction and ease of use, the MAPNET platform could be successfully applied as a teaching tool in introducing students to distributed computing in several directions: focusing on key design and implementation issues, experimenting with the platform itself, and building sample distributed solutions on top of it.

REFERENCES

[1] M. Barnett. .NET Remoting Security Solution. In http://msdn.microsoft.com, 2002.

[2] Grasshopper Programmer's Guide. IKV++ GmBH, Informations- und Kommunikationssysteme, 2001. http://www.grasshopper.de.

[3] R. Gray, D. Kotz, G. Cybenko and D. Rus. Mobile agents: Motivations and state-of-the-art systems. Dept. of Computer Science, Dartmouth College, 2000.

[4] MASIF Specification. http://www.omg.org, 1998.

[5] Microsoft .NET Framework. http://msdn.microsoft.com/netframework.

[6] D. Staneva, P. Gacheva. Building distributed applications with Java mobile agents. Proceedings of Next Generation Network Technologies International Workshop, pp. 103-109, Rousse, Bulgaria, 2002.

ABOUT THE AUTHORS

Dilyana Violinova Staneva, Assistant Professor, Ph.D. Computer Science & Engineering Department, Technical University of Varna, Bulgaria. Phone: +359 52 383424, e-mail: dilyana@tu-varna.acad.bg.

Petya Gacheva Ilieva, Worked on the MAPNET platform as a master student at the Computer Science & Engineering Department, Technical University of Varna, Bulgaria.