On Applying Mobile Agents to Network Management

Huaiyu Liu[1], Marian Nodine[2]

[1] Department of Computer Sciences,
University of Texas at Austin,
Austin, TX 78712, USA.
Email: huaiyu@cs.utexas.edu
[2] Telcordia Technologies,
106 E. 6th St., Austin TX 78701
Email: nodine@research.telcordia.com

Abstract. The inscalability and inflexibility problems suffered by the traditional Network Management System(NMS) have been addressed in many research projects. Mobile agent technology is a promising approach to the design of a distributed NMS. In this report, we first investigated the benefits of applying mobile agents to network management. After evaluating several popular mobile agent platforms, we selected the JADE system and upon it, designed a general architecture of a NMS based on mobile agents.

1 Introduction

Network Management is a critical issue in today's rapidly changing environment. Currently, NMSs are based on centralized client/server frameworks. Management agents[1], the servers in the framework that provide a fixed set of operations, are installed in the managed devices. A management center, the client in the framework, routinely polls the managed devices to monitor and control the network.

However, the exponential growth of the Internet is overwhelming management centers. For example, due to the fact that one cannot predict the future operational conditions and situations, it is better off to let management centers to decide the management operations. Therefore, the management agent on each managed device, an integral and permanent part of the managed device's software, is typically kept small, with minimal functionality, and implementing a few mechanisms. Policies of monitoring and controlling networks are left to management centers. This carries the potential of creating too much network traffic between the management center and managed devices as well as placing large processing load on the management centers. Moreover, a management agent often does not provide a single operation that matches a task of a management center exactly. The management center must invoke a sequence of server operations on each of the managed nodes, which brings intermediate data or irrelevant data across the network on each operation. As we can see, such a framework moves inherently distributed operations to a centralized site, and thus suffers problems such as inscalability and inflexibility.

The ever increasing heterogeneous network is also a great challenge to today's network managers. It requires that a network manager have greater knowledge and increased training. Also, the network management tools are becoming more diverse. Besides the NMSs such as HP OpenView, which is mainly based on SNMP, there are many other stand alone tools. Examples of such tools are *Traceroute*, which can trace the route from a node to another

[1] In the SNMP protocol, there are processes running on network devices and performing the network management functions requested by a management center. These processes are called management agents. However, they are different from the agent concept we discuss in this report. In order to distinguish them, we use management agents to refer to the agents defined in SNMP. Agents or mobile agents, discussed in this report, refer to the agent concept commonly accepted by the agent community: an autonomous entity that is able to learn, cooperate and move.

node; *netimer* [12], which estimates bottleneck bandwidth of a path; and *Treno*[11], which measures the single stream bulk transfer capacity over an Internet path, etc. To manage a network successfully, a manager needs to have the expertise to know which tools to use and which data to analyze under different circumstances.

Recently, mobile agents have been proposed [8] as one approach to realize a distributed NMS. Instead of a centralized and usually very large application that encodes the complete intelligence of the system, a number of relatively small systems, mobile agents, cooperate to do the job of network management. A mobile agent is a computational entity that is autonomous and can move between locations in a heterogeneous network. Mobile agents may exchange their viewpoints and provide strategies. When necessary, they can migrate to remote machines to do some local operations or dispatch other agents to perform sub-tasks. Mobile agents also introduce a new software and communication architecture, which is more efficient than the client/server model and allow rapid development of distributed network management applications.

Currently there are many mobile agent platforms available that provide migration mechanisms to enable agents to move from one node to another. There are also many network management, monitoring and diagnosis tools, as mentioned above, each of which performs a certain kind of task. A network manager needs to excute some combination of these tasks to locate, diagnose and solve network problems. Therefore, it is highly desirable to integrate the mobile agent platform with management tools into a system that does not suffer the problem of inscalability and releases a network manager from routine but complicated management tasks. In such a system, some agents encapsulate existing tools that perform monitoring or diagnosis functions. When necessary, they will migrate to other locations to stay close to some other agents or applications with which they are interacting. Agents need to have some intelligence so that they can collaborate together to provide a solution, simulating the process a manager combining tools together to find solutions. Therefore, we are aiming at designing a NMS based on mobile agents, which integrate the mobile agent platforms with the existing management tools, and adding some intelligence to the agents so that they can cooperate together to either find out what causes the problem in the network, or provide a solution.

This report, is a summary of our work at the first stage, in which we researched on whether mobile agents is a feasible solution to network management, evaluated existing mobile agent platforms and finally, and designed a general system architecture.

The rest of this report is organized as follows. In Section 2, we discuss the benefits of using mobile agents. In Section 3, we review related works on different approaches of network management systems. In Section 4, we evaluate several mobile agent platforms and choose one that is close to our purpose. We present a general architecture of a NMS based on mobile agents in Section 5 and conclude in Section 6. In this report, we are targeting at intra-domain network management.

2 Motivation

In general, there are several advantages of using mobile agents [6]. To discuss the benefits of applying mobile agents to network management, we first present two examples.

2.1 Motivating Examples

Example 1: Unreliable links In some cases, a subnet is connected to a network via a satellite or a microwave link. To manage the devices in this subnet, a network manager needs to connect to them through the satellite link and do some operations. If the link becomes disconnected during the operation, then the operation likely will fail and have to be restarted when the connection is available again. Even if the connection is up during the operations, the management data transmitted through the link may cause link congestion, since a satellite link is usually of low bandwidth. The same thing could happen if a manager uses a mobile computer to perform some management tasks.

Example 2: Monitoring an ISP system The second example is about a management system that monitors an ISP network, as illustrated in Figure 1 [1]. There are two key components of an ISP system that supports modem-based dial-in access to the residential subscribers: the Points of Presence (POPs) and a server farm.

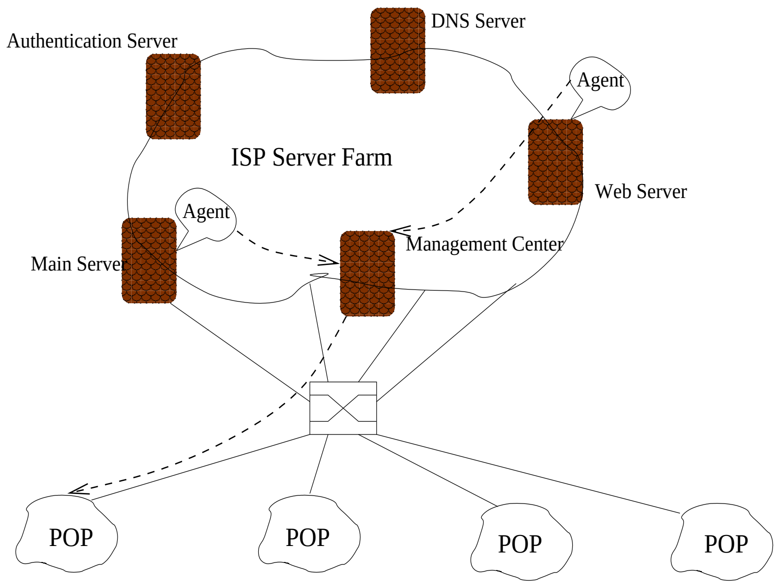

Fig. 1. An ISP monitoring system

– Points of Presence: A site established by an ISP to house equipment through which subscribers are connected to the Internet. It is composed of modems and terminal servers.
– ISP Server Farm: A location that houses servers that support applications such as Web, Email, and News that subscribers access. Infrastructure services such as domain name service (DNS), user authentication service, etc., are also located here.

Typically, the POPs are geographically distributed. To monitor POP equipments, the management center, which is located at the server farm, polls each POP site periodically. This will not only generate a great amount of traffic to the management center, but also suffer great latencies – polling hundreds of sites which may be hundreds of miles away can not be finished very quickly. Also, suppose a link between the server farm and one of the POP site is down, or is highly congested, then querying information of that POP site may fail. Status of that POP site while the link is down will be unavailable. More importantly, problems at the POPs can not be addressed in a timely manner, even when the nature of the problem at the POPs is evident.

Another kind of measurements performed by the ISP monitoring system are service quality measurements. In order to keep track of the subscriber-perceived performance, some monitoring agents[2] are used to make active measurements to assess the availability of services, such as the Web services. However, those agents are placed inside the ISP server farm, close to the servers. As a result, they can not provide an accurate view of subscriber-perceived performance since the impact of networks interconnecting the ISP's POP sites to the server farm is not reflected in the measurement.

[2] They are similar to the concept of agents defined in SNMP.

2.2 Reasons for mobile agents

Mobile agents can help to solve the problems mentioned in the above section.[3]

Disconnected operation By disconnected operation, we mean an operation that need to be carried out even if the connection is down. Since in a wireless network, links are often of low-bandwidth or high-latency, it is common for a mobile computer to disconnect from the network and reconnect some time later. Even in a wired network, connections may go down if some links in between are highly congested. In such circumstances, mobile agents are even more advantageous.

Let us consider example 1 again. To manage devices via an unstable link, for example, the satellite link, a mobile agent, which encapsulates the task, can be dispatched and sent to the subnet in the other side of the link. Then, the agent can perform tasks locally even if the connection is down. Once the link is up again, the agent can report back from the subset what it found and what it did. Also in the same case, the simple ability of agents to migrate and operate at the other side of the link helps avoid extensive use of a low-bandwidth link and reduce latency.

Dynamic deployment As discussed above, a management agent in a managed node normally only provides a fixed set of operations. If the management agent does not provide a single operation that matches a requirement from its management center, either the management center must invoke a sequence of operations, or the management agent needs to be upgraded. The first option brings intermediate data across the network on every operation, potentially wasting network bandwidth. The second option is impractical in most cases, since the network management demands may change over time and one can not always try to match the demands by programming and updating the management agents on each managed node.

Mobile agents can help in such cases – a new operation or an updated program can be implemented as a mobile agent and the mobile agent, in turn, is sent from the management center to the managed nodes. The agent can then execute locally on the managed nodes and return the result to the management center. Such uses of mobile agents are examples of dynamic deployment, where an application dynamically installs software on some remote host. Besides mobile agents, there are some other approaches to dynamic deployment. The drawback of those approaches is that it is often difficult for a dynamically-deployed component to deploy subcomponents, or for the component to re-deploy itself. Mobile agents, on the other hand, can handle both situations easily.

Consider the ISP example described above, to get a measurement that is close to the subscriber-perceived performance, the measurement needs to be made at each POP. However, considering an ISP may have hundreds of POPs, installing and updating the measurement tool at each POP would be a big burden to a network manager. With the help of mobile agents, installing a new tool can be carried out by sending out a mobile agent, which encapsulates the new tool. Similarly, the updating of tools can be accomplished by stopping old agents and dispatching new agents.

Conservation of bandwidth One of the problem of the traditional client/server network management model is it often creates much more network traffic than necessary. To illustrate this, we make the following comparison.

Suppose there are several network servers, such as web servers, that are located in the same subnet. A mangement center is located at another subnet, which is a few hops away

[3] In this report, we assume that mobile agents work at the application layer, since they will be dispatched and managed by network management centers, which run at the application layer. Another reason for the assumption is that the current avaliable mobile agent platforms are also implemented at the application layer.

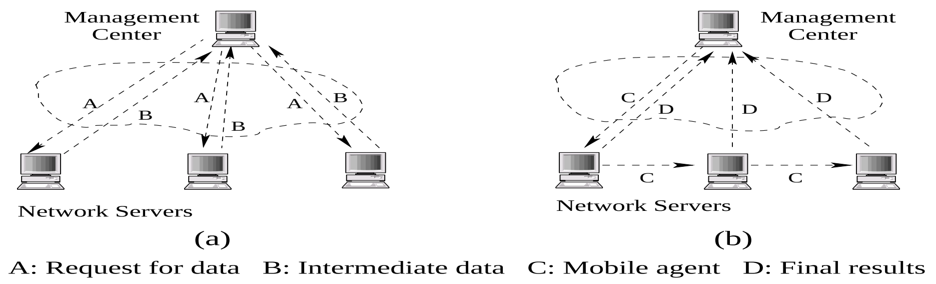

Fig. 2. A comparison of two kinds of NMS.

from those servers. The management center wants to get some statistical data from log files kept at each server. It is undesirable to implement statistic computations on each server, since computations required by the management center are unpredictable and may be different each time. Therefore, the management center has to go to each network server, fetch the log files, which are commonly in the size of hundreds of kilobytes or several megabytes, to perform its desired computation. The management center may be only interested in a small amount of data inside the log files, but nevertheless, the whole files have to travel several hops from the managed nodes(the servers) to the management center. This wastes large amount of network bandwidth.

Mobile agents, on the other hand, do not waste bandwidth unnecessarily. They can migrate from the management center to the managed nodes, perform operations on the managed nodes locally and only send the final results, which is most likely a few percent of a log file, back to the management center, as depicted in Figure 2(b). Typically, a mobile agent is in the size of a few kilobytes. Therefore, by sending small piece of code to a big chunk of data instead of the opposite way, network bandwidth can be saved significantly.

From the above example, we can see that mobile agents not only help with conserving bandwidth, but also can tailor the returned information to meet the current interest of the management center.

3 Related Work

3.1 Smart Packets

Active network [5] is a framework within which users inject programs contained in messages into a network. The programs will be executed at each router or switch they traverse. The goal of active network is to increase the programmability of computer networks and network components. Smart Packets [4], an active network project in BNN, puts active network technology into the network management to make managed nodes programmable. A smart packet consists of a Smart Packet header followed by payload. Based on an IP option in the packet header, the Router Alert option, a router can determine whether a packet is a Smart Packet. If it is, the router will process the datagram content of that packet and execute the program inside the packet.

Smart Packets share some ideas with building a NMS based on mobile agents. They are both aimed at breaking the traditional client/server network management model, distributing management tasks, reducing traffic, and shortening the control loop. However, they are implemented at different layers. Unlike a mobile agent that runs at the application layer, Smart Packets work at the network layer and thus can take advantages of network services. No special migration mechanism is needed in an active network – packets are forwarded by routers. Smart packets are executed by intermediate routers along a path and therefore, can report useful information of each router in that path without extra mechanisms. Working at the network layer also makes a smart packet be able to access MIB information more efficiently.

However, Smart Packets have disadvantages in real applications. First, successful functioning of Active Networks requires support from routers. A router needs to be able to tell whether a packet is an active packet and also needs to install execution environments for active packets. Even without considering security issues, upgrading a router to support active networks is much more difficult than upgrading a host to support mobile agents.

Secondly, as discussed above, smart packets work with routers and switches. However, managing a network not only means the management of routers or switches, but also the management of network servers, such as a web server, and common hosts. The ability of managing the latter is something missing in Smart Packets, since they are implemented in the network layer.

A third concern is, there is a major design decision in Smart Packets that a program sent in a packet must be completely self-contained, which means that the packets cannot be fragmented. This frees routers from the need to keep persistent state for active packets. However, as a result, it constrains the program to be expressed under 1 Kbyte in length. This is a strong requirement. Although BNN develops special programming languages that can encode a program in a much shorter length, it is unlikely that 1 Kbyte is enough for expressing all useful management programs. If we consider the case that a packet not only need to carry a program, but also some data resulting from the processing of the program on routers, or a certificate for authentication purpose, the space left for a program is even more limited.

3.2 Management systems based on static agents

[1] describes an ISP performance and service management system. In this system, there is a diagnostic measurement server(DMS) serving as a host for monitoring agents that make measurement of service qualities at each POP. There are also some monitoring agents operating on the ISP servers to track resource and service usage. Such a system distributes management tasks across the agents. However, all the DMS and the agents reside inside the ISP's server farm, and monitor devices in the server farm as well as in each POP, so this is still a centralized management model. Since the agents are statically installed, they lack the flexibilities discussed in section 2. Moreover, even though called as agents, the agents in [1] are different from the agents discussed in this report.

3.3 Other related works

There are several research projects that develope mobile agent frameworks for distributed network management [2] [3]. These works focus more on building platforms, while we are interested in integrating an existing platform with some tools to build a NMS based on mobile agents. In [9], agents are used to develop a system for QoS management, which is another application of agent technology for network management.

4 Evaluations of Mobile Agent Platforms

Before comparing the existing agent platforms, we will first look at what NMS requires from a mobile agent platform. First of all, security is a big concern in network management systems, since these systems can provide a view into the entire corporate network.

Second, the platform should provide a dispatching mechanism so that whenever necessary, an agent can dispatch some other agents to perform sub-tasks. For example, if a network manager finds that a web-server is not reachable from his machine, then he may initiate an agent, tell the agent about the unreachbility and ask it to find out what could be wrong between his machine and the web-server. The unreachability can be caused by many reasons: one of the links in between may be down; some router/switch along the path may be highly congested; the HTTP socket of the web server may be unreachable; or there may be a problem with the server software. The agent, initiated by the manager, can dispatch several mobile agents, each acting as a different diagnosis tool: a Traceroute agent to find out the nodes in between the two machines; a PING agent to test the livability of each node in between; a SNMP agent to get network statistics using the SNMP protocol; and a client agent to test the performance of the web-server, etc.

A third requirement is support of standard ontology, agent languages and various kinds of message passing mechanisms. These are the standards for a network manager to express management tasks, or for agent communications.

4.1 Standards: FIPA and MASIF

Many mobile agent platforms have been developed in recent years. All these systems have the same general architecture, as shown in Figure 3: a server on each machine accepts incoming agents, starts an appropriate execution environment, loads the agent's state information into the environment and resumes agent execution. Nevertheless, they take different approaches regarding implementation languages, communication protocols and platform functionalities. In order to achieve inter-operability between platforms, two standards of agent technology have been established. One is the Foundations for Intelligent Physical Agents(FIPA) [13] developed by the FIPA organization, the other is the Mobile Agent System Inter-operability Facility (MASIF) [14], developed by the Object Management Group(OMG).

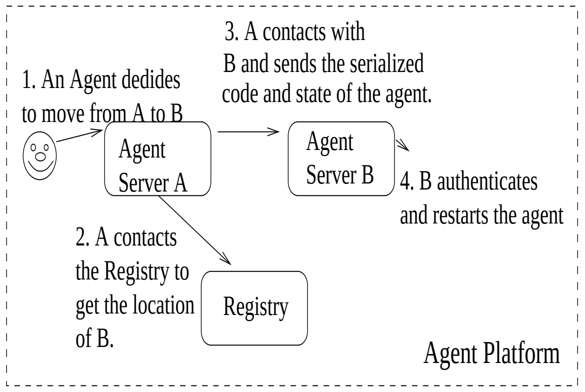

Fig. 3. An general architecture of agent platforms

FIPA and MASIF are different in many aspects [7]. In general, the differences can be characterized by the extent to which they focus on agent mobility and the semantic richness of their communication protocols. The focus of MASIF is mainly on the inter-operability of agent systems to support the mobility of agents. It has no specifications for agent communications. Instead, it relies on the CORBA object communication mechanism to provide communication services, which are in the form of remote procedure calls.

On the contrary, FIPA works on enabling agent inter-operability via standardized agent communication and content languages. Besides the generic communication framework, FIPA also specifies ontology and interaction protocols to support agent interaction in specific application areas. It supports not only syntax-based inter-operability but also semantics-based inter-operability. Agents in FIPA usually has a speech act alike communication language and a predicate logic based language. These features of FIPA will be very useful for complex and dynamic co-operation problems, for example, network management. Although focusing on agent intelligence, most FIPA platforms also support agent mobility.

Our understanding is that MASIF is more like a mobile object-oriented standard, while FIFA is more agent-oriented. Therefore, we concluded that a FIPA platform better fits our approaches to deploying mobile agents in network management.

4.2 Comparison of the current mobile agent platforms

A comparison of some mobile agent systems can be found in [6]. However, only a few of the systems discussed in [6] are FIPA- or MASIF-compliant. Since the general acceptance of mobile agents for network management will depend heavily on standards, we will not consider the platforms that are not standards compliant.

Among the standard compliant platforms, we chose three of them that are under strong technical support, Java Agent DEvelopment Framework(JADE) [15], Grasshopper [16] and Aglets [14]. Aglets is a MASIF compliant platform. JADE is a FIPA platform, and Grasshopper supports both MASIF and FIPA. All the three systems are Java-based systems. We have installed and tried with all the three platforms.

JADE. JADE is a FIPA-compliant platform. It is strongly supported by the JADE group at CSELT, which work with FIFA closely.

JADE is used as the basis for the LEAP kernel. LEAP is an on-going project, which will develop an agent platform that is light-weight and executable on small devices such as PDAs and phones. Therefore, if an application is built upon JADE, it is likely that later on the application can be ported easily to LEAP and running on small devices. Among FIPA platforms, currently only JADE has a micro edition that is extended to small devices.

JADE supports complex agent behaviors. It is very likely that in some cases, an agent must be able to carry out several concurrent tasks in response to different external events. In JADE, tasks of an agent are implemented as different kind of behaviors, such as SimpleBehaviour, CyclicBehaviour and ParallelBehaviour. The platform also provides a scheduler that carries out scheduling policy among all behaviors available in the ready queue. This feature of JADE makes it easier to design an agent and its tasks, and make the management of agents more efficient.

One drawback of JADE is that its current released version does not have security support, which would be unacceptable to network management. However, the source is open, so we could enhance security as needed. Furthermore, as the FIPA security sepcifications envolve, JADE will incoporate them.

Grasshopper. Grasshopper is a mobile agent platform that is built on top of a distributed processing environment. It is compliant to MASIF. It also supports FIPA by providing a FIPA extension as an "add-on" package.

One of the advantages of Grasshopper is that it has a good security support built in, which supports two kinds of security mechanisms:

- External security protects remote interactions. For this purpose, X.509 certificates and the Secure Socket Layer (SSL) protocol are used.
- Internal security protects interfaces of agencies[4] and agents as well as certain agency resources (such as the local file system) from unauthorized access. This access control is achieved by authenticating and authorizing the owner of the accessing agent.

As a MASIF-compliant platform, Grasshopper relies on CORBA to provide some services, such as naming services and communication services.

The FIPA extension of Grasshopper is based on FIPA97. However, FIPA97 is already considered obsolete and replaced by FIPA2000. FIPA2000 has more specifications than FIPA97, such as the FIPA interaction protocols.As a result, Grasshopper does not have agent interaction supports. In Grasshopper, Communications between agents are based on synchronous or asynchronous communication services, which are still like remote procedure calls.

Another concern about Grasshopper is that no open source code is available except the FIPA extension package, which will make it difficult to tailor the platform to meet our requirements.

[4] In Grasshopper, an agency is the actual runtime environment for mobile and stationary agents. The similar concepts in JADE and Aglets are called Agent Container and Aglets server.

As a result, we eliminated Grasshopper from consideration.

IBM's Aglets. IBM's Aglets is a popular agent platforms. It has a good reputation of being easy to install and use. Besides, it also has security support integrated. Because of these reasons, we also investigated Aglets even though it is a MASIF-compliant platform.

In the Aglets system, implementing a mobile agent is clear and simple. When an agent wants to migrate, it calls the *dispatch* method. The Aglets system calls the agent's *onDispatching* method, which performs application-specific cleanup, kills the agent's threads, serializes the agent's code and object state, and sends the code and object state to a new host. On the new host, the system calls the agent's *onArrival* method, and then calls the agent's *run* method to restart agent execution.

Aglets system also has a certain level of security support. It supports intra-domain authentication, agents authorizations and integrity-checked communications.

- Intra-domain authentication
  Aglets servers in the same domain share the same secret. Based on the secret, a server can authenticate an agent that originates from the same domain.
- Authorization
  After being authenticated, an agent will then be granted some access permissions based on its identity, such as permissions to access a file or a socket, send a message or to be loaded dynamically.
- Integrity-checked communications
  The communication between two Aglets servers or two agents are also protected by integrity-checking: a Message Integrity Code (MIC), computed by the value of the message content and the shared secretes are sent along with the content and verified by the receiver.

However, after careful investigation, we do not think Aglets is a satisfying platform despite the above features. Since Aglets is a MASIF platform, it emphasizes supporting mobility of agents as opposed to agent intelligence. The message exchanged between agents are quite simple: a message is composed of two parts, one is a string that identifying the type of the message and the other is the value of some argument. It it left to the agent programmers to define the syntax or semantics of messages. Another concern is that the mechanism supporting agent tasks in Aglets is relatively too simple. Aglets system only provides a *run* method to implement tasks of an agent. Scheduling and management of agent tasks are left to programmers. JADE, on the other hand, supports different agent behaviors(tasks), either sequential or parallel, and provides a scheduler to schedule the ready behaviors.

A summary of the capabilities of these platforms is shown in Table 1.
Finally, we chose JADE as our platform for the following reasons:

- It is a FIPA2000 compliant platform. Although it does not support all FIPA2000 interaction protocols at present, based on our interaction with JADE programmers, who are very accessible, we expect the situation can be improved in the near future, since JADE groups is working with FIPA organization closely. Besides, JADE also has a related micro edition executable on small devices, LEAP.
- Compared with the other two platforms, JADE is better at supporting application-defined content languages and ontologies, which is important to network management applications.
- It supports complex agent behaviors.

4.3 Platform Enhancements

Although JADE is more close to our requirement, it is still not an ideal platform. For instance, the current released version, JADE2.2, does not have security support built in. We

	JADE	Grasshopper	Aglets
Standard	FIPA2000	Supports MASIF and FIPA97	MASIF
Implementation Language	Java	Java	Java
Source code	Open source code	No open source code is available except the FIPA package.	Open source code
Security support	No security support in the current released version. However, an experimental version has been developed to enhance security in JADE.	External security support: X.509 certificates, RMI and plain socket over SSL. Internal security support: Access control	Intra-domain authentication, authorization and integrity-checked communications, implemented by using Java security APIs.
Mobility	Weak mobility[5]	Weak mobility	Weak mobility
Communication mechanisms	Within the same Agent Container: event signaling; Within the same JADE platform but between different Agent Containers: Java RMI; Between different platforms: either CORBA-IIOP or HTTP.	CORBA IIOP, Java RMI or socket connections. Which one to use is dynamically determined by the platform.	Self-defined Agent Transport Protocol and Java RMI.
Message passing mechanisms	FIPA interaction protocols, such as FIPA-Query, FIPA-Contract-Net and FIPA-Request.	Synchronous and asynchronous.	Synchronous and asynchronous.
Agent dispatching mechanisms	(1) Request the platform to create an agent; (2) Use an Agent.doStart() call	Request the platform system to create an agent.	Request the platform to create an agent.
Ontology and agent language support	Yes.	Yes.	No.

Table 1. Comparison of JADE, Grasshopper and Aglets

expect that a new version of JADE will integrate security support[6]. Nevertheless, we need to implement some security mechanism ourself to meet the special security requirements of network management. Another enhancement needed is to implement more FIPA interaction protocols. Protocols such as FIPA-request-when, FIPA-Brokering, FIPA-Recruiting and FIPA-subscribe protocols are useful to network management, but have no implementations in JADE2.2.

Enhancements of the platform are part of our future work.

5 Our approach

5.1 An architecture

In this section, we introduce an architecture of a NMS based on mobile agents. Before discussing about the architecture, we introduce the following concepts:

- Role: A management task is defined as a set of roles. A role can be executed by one instance, or several instances of an agent[7].
- Matchmaker [10]: Responsible for matching roles to agents by specific criteria. A matchmaker is different from a DF defined in FIPA. It matches an abstract role description to some agent instances, instead of being given an AID (ID of an agent, including its name and address) and searching for the corresponding agent.
- Agent factory: A repository of codes of agents.

[5] There are two kinds of migration [6]: (1) Weak mobility, where the system only captures an agent's object state and code before agent migration. (2) Strong mobility, where the system captures an agent's object state, code and control state before migration, allowing an agent to continue execution from the exact point at which it left off.

[6] By private conversations, we were told that the JADE group has already developed an experimental multiuser version, in which a security model is defined.

[7] In this report, an agent instance means a currently running instance of an agent.

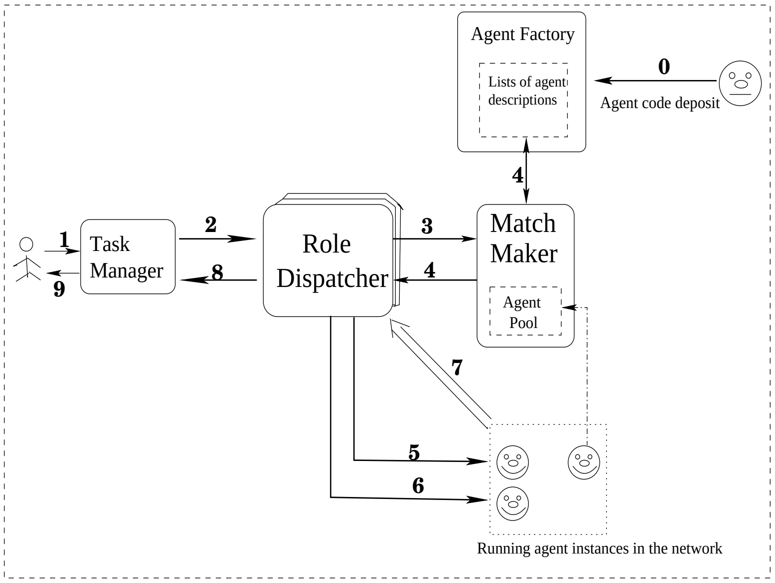

Fig. 4. An architecture of network management based on mobile agents

– Role dispatcher: Resposible for deciding how many agent instances are needed to execute a role and dispatching those agent instances. A role dispatcher is corresponding to one role.

The architecture is shown in Figure 4.

1. A network manager submits his management request to the system. The request, including some necessary input arguments, is expressed according to some ontology.
2. A task manager, receives the request and initiates a management task. The task may be decomposed to a set of nested roles. Then, the task manager invokes several role dispatchers, each representing one role.
3. A role dispatcher decides the number of agent instances needed. It then contacts the matchmaker, sends it the description of the role it represents and the number of agent instances it requires, and ask the matchmaker to match the role to agent instances.
4. The matchmaker does the matching based on the agent pool, in which registrations of agent instances are stored.
   - If a role is matched successfully, the matchmaker will return a list, which contains descriptions of agent instances to the role dispatcher. The name and locations of an agent instance and the protocol, ontology and languages it uses are contained in the list.
   - If no agent instance can be matched to the role, then the matchmaker contacts the agent factory to create instances the appropriate agent. The agent factory returns a description of that agent.
5. If the role is matched successfully to some agent instances, the role dispatcher will negotiate with those agent instances, requesting them to perform actions. An agent instance may or may not accept an assigned action, based on some conditions.
6. The role dispatcher repeated step 4 and 5 until it has a complete set of agent instances.
7. Later on, agent instances who have accepted the assigned jobs return results, including possible failures, to the corresponding role dispatcher. The role dispatcher then collects all the results and return them to the task manager.
8. After processing the results returned from the all the role dispatchers, the task manager present the results to the network manager.

The dash arrow in Figure 4 means that some of the new created agent instances will register themselves with the matchmaker. During the registration, an agent needs to send the matchmaker its name and current location, the service it provides, and the protocol, ontology and language it uses. Not all the new created agent instances will register with the matchmaker. For instance, there is no need for a PING agent instance, which sends out testing packets to a node to test the reachability of that node, to keep running in the system and waiting for requests. Such an instance can be initiated whenever needed.

The step 0 in Figure 4 indicates the process of depositing agent code into the agent factory. In section 3.3, we have discussed about the existing management tools. In order to integrate them with our agent base NMS, we need first to convert these tools into agents. After being developed, the agents(code) need to be deposited in the agent factory. At the same time, the name, service, protocol, ontology and languages of an agent need to be registered with the agent factory.

This architecture is designed upon the JADE platform. Therefore, creation, deletion and migration of an agent are supported by the underlying Agent Management System(AMS). In the architecture, the task manager, matchmaker and role dispatchers can be implemented as agents. Therefore, requests and responses between them can be implemented by using the FIPA interaction protocols.

– The task manager can use FIPA-Request or FIPA-subscribe protocols communicate with role dispatchers.
– Role dispatchers can use FIPA-Query protocol to interact with matchmaker.
– Role dispatchers can use FIPA-Request or FIPA-subscribe protocols request agent instances to execute actions and inform it the results.

5.2 Future work

Since our work is still in the beginning stage, the system architecture shown in Figure 4 is still in the process of evolution. We are seeking good solutions for several problems, which includes:

– How to categorize agents into two groups so that instances of one group of agents, when being created, will register with the matchmaker but instances of the other group of agents will not? By registering with the matchmaker, an agent instance can advertise its services to other agent instances. Then, next time when an instance of the same agent is needed, there is no need to create a new one, but to "reuse" the existing instance. In this way, overhead of creating, deleting and migrating of an agent is avoided. Besides, it is necessary to have some agent running in the system and providing services, for instance, monitoring the status of a part of the network. On the other hand, if some type of agents are used infrequently, for instance, only once per day, then keeping an running instance of that agent in the system is not economic.
– How to organize the agent factory and the agent pool so that matchmaking can be carried out efficiently?
– Ontologies need to be defined in step 0 and step 1. Agent content languages are needed in step 2 to step 11. The languages defined in FIPA, the SL0, SL1 and SL2, are not enough for our purpose. Therefore, we need to define our own content languages.

6 Conclusion

Currently, there are many mobile agent platforms, but few systems that integrate a mobile agent platform and network management tools have been developed. We have researched the feasibility of such systems and concluded that applying mobile agents technology to network management systems has several advantages. Mobile agents can help dealing with the problems of unreliable links, dynamic deployment and bandwidth conservation. Among the standards compliant platforms, we choose the JADE system, since it is more close to our purpose. Based upon JADE, we design a general architecture of a mobile agents based NMS and will continue working on it. Our future work includes enhancing the platform, improving the system design and implementing a prototype system.

References

1. S. Ramanthan, E. Perry. *The Value of a Systematic Approach to Measurement and Analysis: An ISP Case Study*, HPL-98-171, Hewlett-Packard, 1999.
2. G.Susilo, A.Bieszczad, B.Pagurek. *Infrastructure for Advanced Network Management based on Mobile Code*, Proc. of the IEEE/IFIP Network Operation and Management Symposium (NOMS'98), New Orleans, Luisiana, February 1998.
3. H.Ku, G.W.R.Luderer, B.Subbiah. *An Intelligent Mobile Agent Framework for Distributed Network Management*, Globla Telecommunications Conference, 1997.
4. B.Schwartz, A.W.Jackson, W.T.Strayer, W.Zhou, R.D.Rockwell, C.Partridge. *Smart Packets for Active Networks*, OpenArch, March 1999.
5. D.L.Tennenhouse, D.J.Wetherall, *Towards an active network architecture*, ACM Computer Communication Review, vol.26, No.2, April 1996.
6. Robert S. Gray and George Cybenko and David Kotz and Daniela Rus. *Mobile agents: Motivations and State of the Art*, Technical Report TR2000-365, Department of Computer Science, Dartmouth College, 2000.
7. *Agents Technology in Europe*, http://www.infowin.org/ACTS/ANALYSYS/PRODUCTS/THEMATIC/AGENTS/toc.htm.
8. A.Bieszczad, B.Pagurek, T.White, *Mobile Agents for Network Management*, IEEE Communications Survey, http://www.comsoc.org/pubs/surveys, 4th Quarter 1998, Vol 1, No 1.
9. H.de Meer, A.Puliafito, O.Tomarchio, *Management of QoS with Software Agents*, Cybernetics and Systems: An International Journal, 29(5):499-524, July-August 1998.
10. A.Cassandra, D.Chandrasekara and M.Nodine *Capability-based Agent Matchmaking*, Proceedings of the International Conference on Autonomous Agents, June, 2000.
11. M.Mathis and J.Mahdavi, *Diagnosing Internet Congestion with a transport layer performance tool*, Proceedings of the Second Grace Hopper Celebration of Women in Computing Conference, Sept. 1997.
12. K.Lai and M.Baker, *Nettimer: A Tool for Measuring Bottleneck Link Bandwidth*, USENIX Synposium on Internet Topology and Systems, March 2001.
13. The Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org/about/index.html.
14. Mobile Agent System Inter-operability Facility(MASIF), http://www.fokus.gmd.de/research/cc/ecco/masif/.
15. Java Agent DEvelopment Framework, http://sharon.cselt.it/projects/jade/.
16. Grasshopper, http://www.grasshopper.de/index.html.
17. FIPA-OS, http://fipa-os.sourceforge.net/.
18. Aglet Software Developement Kit, http://www.trl.ibm.com/aglets/.