UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
UNIVERSITÉ DU QUÉBEC À MONTRÉAL

THE MOBILE AGENT TECHNOLOGY APPLICATION IN THE DISTRIBUTED DATA SERVICE MODEL

MÉMOIRE
PRÉSENTÉ
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE

PAR
WEIJIA-LU

JUIN 2005

Université du Québec à Chicoutimi

Mémoire présenté à
L'Université du Québec à Chicoutimi
comme exigence partielle
de la maîtrise en informatique

offerte à

l'Université du Québec à Chicoutimi
en vertu d'un protocole d'entente
avec l'Université du Québec à Montréal

par

WEIJIA-LU

THE MOBILE AGENT TECHNOLOGY APPLICATION IN THE DISTRIBUTED DATA SERVICE MODEL

Juin 2005

ABSTRACT

Along with the development of the economy and technology, the network has become more important in people's work and life than before. More and more people take advantage of network to improve the quality of their work and life. The network applications are based on the database visiting, and the database visiting is almost based on the Distributed database.

The Distributed data service can be widely used in the corporation and many important fields. More technology and solutions are proposed, including the Mobile Agent technology. Mobile Agent is a code segment that can migration in the network, carrying the task code, using network and computers' resource, cooperating with other MAs, finishing one or more designated tasks.

Compared with other mobile code, the MA technology can use less network resource, carry less information, and operate with higher efficiency. Using the technology to solve distributed problems is a hot research topic currently.

In this paper, we first review the current research status in the Mobile Agent and distributed data service fields. After that, based on the current issues in the Distributed data service fields, we propose a complete solution using the Mobile Agent technology. Last, with the proposed solution, we construct a complete distributed data service model.

In our solution, the traditional Distributed data-retrieving procedure is investigated and enhanced with the Mobile Agent technology to achieve high efficiency. The Mobile Agent related technology, such as MA communication, MA security, and MA cooperation technology are also used to improve the model's efficiency and reduce the network's cost.

We have used a project management system as a sub-model. Compared with the traditional approaches, the model works more efficiently with larger number of stations and large amount of the data-request. Finally, detailed analyzing of one instance is presented to show the characteristic of the model.

ACKNOWLEDGEMENT

First, I am sincere to thank for my professor Changyun Yu. My paper is accomplished under her supporting and supervising. Her study attitude, scientific attainments, responsibility and conscientious working give me a deep impression, and give me power to accomplish the paper and work in the future.

And then thank very much for professor Paul. He gives me many suggestions, helping to amend my title and finish my paper on time.

Thank for Yuheng Pan, she helps me revise my English.

Thank for my parents. They give me power to finish my paper. Thanks a lot for everyone.

LIST OF ACRONYMS

MA	Mobile Agent
MAE	Mobile Agent Equipment
API	Application Program Interface;
RPC	Remote Procedure Call
UA	User Agent
SA	Server Agent
ACL	Agent Communication Language
KQML	Knowledge Query and Manipulation Language
SSL	Security Socket Layer
UML	Unified Modeling Language;
ATCI	Agent Transport and Communication Interface
ATP	Agent Transfer Protocol
OMG	Object Management Group;
DNS	Domain Name Server

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

PREFACE

The existing research result indicates, distributed data services have to overcome the following obstacles: first, the distributed data service model must be able to divide data access requests rationally based on data distribution among different data storing stations, and send sub-request to different data storing stations; second, in the process of obtaining and collecting data, stations must maintain instant communication to complete data access task.

The Mobile Agent technology has provided a basis for network cooperation and intelligence. The Mobil Agent technology includes concepts for Mobile Agent and Mobile Agent Equipment. The Mobil Agent can be transferred freely among heterogeneous computer networks. The main difference from the traditional distributed computing is that Mobil Agent has mobility and self-organization in nature; Mobile Agent can make mobile decision itself during execution. This allows MA to make the optimum decisions of data operations and data access based real time data fields and actual values. The main advantages are that it can reduce network load, support specialized services for certain fields and requirements, reduce server access time delay, support mobile customers, and achieve higher error tolerance and security. Also, Mobile Agent code is easy to use and program.

The biggest advantage is that it can reduce network load, support customized service for region and specific logics, reduce delaying time during visiting servers, support mobile customers, provide stronger fault-tolerant and security. And, on the premise of constructed platform, the programming mode of Mobile Agent is easy to use and reconstruct.

Mobile Agent Equipment is a carrier and platform for all Mobile Agent operations, such as, movement, communication, and interacts with local or remote program API.

The proposal of Mobile Agent provided a brand-new concept in distributed data service model architecture. Introducing Mobile Agent concept into distributed data service model brings us following advantages:

1. Provide mobile terminal portability.

Because MA is platform independent and has higher abstractness, on the platform that have installed virtual machine environment, each terminal station can interact with the distributed data service network, and apply MA technology without pre-install or pre-download the database engine and the API of Mobile Agent.

2. Achieve higher efficiency

During the execution of distributed data request, it can utilize certain optimized strategy to divide a global data request into multiple sub-requests to many individual stations. These sub-requests can be carried by Mobile Agent code simultaneously, and achieve network resource utilization, and complete the data access requests efficiently.

3. Achieve better flexibility

Because MA can be executed asynchronously, and can migrate in the network according to certain strategy, it can adjust the execution tactics according to the actual data distribution reality.

4. Improve the expansibility

Because MA can detect dynamically the change of the data distribution on the network, and adjust the tactics of the data access and transmitting according to the change, in the process of distributed database access, the service model utilizing MA technology can detect and adapt to the database type and quantity change of individual data site dynamically, and there is no need for human intervene.

5. Improve the transparency

Because MA can take actions self-organize in service terminal, users can let MA perform the related data retrieve work, just like visiting a local centralized database, and needn't to worry about the detail of each site's with distributed data characteristics. Also, MA can adapt the changes of the data site dynamically; the environment change of the network is also transparent to users.

The paper's main research is to apply Mobile Agent technology into the distributed data service region. Specifically, take the full advantage of MA (self-organize, migration, and less network resource demand) into the application of distributed data service; finally, construct a distributed data service model based on MA.

First, in this paper, we discuss the characteristic of Mobile Agent and distributed data model, second, based on the existing research results, we propose some improvement on certain key implementation issues, such as strategies on dividing data request, Mobile Agent communication, migration technology, distributed security etc. These will improve efficiency of the distributed data service. , We apply the aforementioned key improvements to a distributed data service model based on Mobile Agent, and implemented this model to a real project. Finally, we analyzed the result, provided

the characteristic of this model, and proposed the places that can be further improved.

CHAPTER 1

THEORIES FOUNDATION

1.1 Relevant theory of Agent

Agent technology can be traced back to the early stage of the artificial intelligence research, Hewitt first proposed a software model that has self-organize, responses and synchronized execution when he was researching the Concurrent Actor Model in 1977. This is initial software concept of Agent.

Agent is an entity that can finish a specific work independently. Agent belonging to the category of artificial intelligence has imitated the behavior and relationship of the human society to a certain extent. It can offer the corresponding service to other Agent.

The concept of agent has great flexibility and adaptability, is suitable for open, dynamic network environment. [1][27]

1.2 Mobile Agent overview

1.2.1 Definition of Mobile Agent

Along with the more and more advanced application areas using Internet, especially the areas of information search, distributed computing and e-commerce, people hope to obtain the best service using the whole Internet resource. People expect the whole network to become a complete entity, Agent can move freely in the whole network. The concept of Mobile Agent was arisen from this thought. At the

beginning of 1990s, General Magic Company proposed the concept of Mobile Agent firstly in business system "Telescript"; a program code entity that can migrates from a host computer to another host computer independently in different heterogeneous environment of the network and can be interactive with other Agent or resource. This was an extension of original software Agent, besides the basic characteristic of software Agent - -Autonomy, cooperation, initiative, more important, it also has mobility, so it can migrates from a host computer to another host computer in the network independently and execute designated tasks for users. Because Mobile Agent can be migrated freely in different software and hardware network environment, so the new computing mode can reduce the network load of distributed operation, improve efficiency of communication, adapt of network environment changes dynamically. Also, very good security and fault-tolerant ability can be achieved in network.

Mobile Agent can be regarded as the result that the software Agent technology combined with distributed computing technology. It is different from traditional network computing mode in nature. It is also different from the RPC; because MA can migrated from one node to another continuously, and this kind of migration can be initiated per task basis as needed. [2][28]

MA is different from the general process migration. Generally speaking, the general process migration doesn't allow process to choose migration time and migrating target itself. However Mobile Agent can be migrated at any time, and move to any place that it wants to go. MA is also different from Applet of Java, because Applet can only migrate from servers to clients; Mobile Agent can do bidirectional movement [27] between clients and servers.

Though different Mobile Agent system has different system architecture, almost all the Mobile Agent systems include Mobile Agent and Mobile Agent Equipment (MAE) [3-4].

1.2.2 Mobile Agent working method

MA is a code segment that can move freely among nodes in network, and can carry their own state and code from a host computer to target computer to execute corresponding task. Mobile Agent has mobility and independence.

Mobility means: MA can be migrated among nodes freely. It can collect information of each node and execute the designated task. It has less dependency on network transmission, and directly access to server resources to be visited, thus prevents excessive data transmission in network, and reduces the system dependence on network bandwidth.

Independence means: Mobile Agent chooses migration target independently. It does not need a centralized dispatcher, and MAs can take alliance according to each object, execute a specific task together.

There are differences in nature between Mobile Agent and traditional distributed computing. Mobile Agent technology can reduce network load. Traditional distributed system is shown in the following Fig 1; it usually depends on the network communication protocols too much. These protocols require many times interactions during the process of execution of the task, and can cause the network congestion easily.

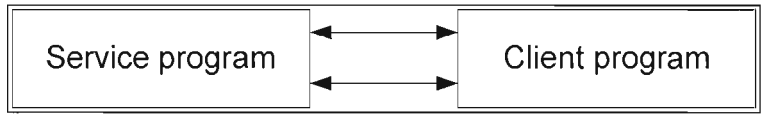


Fig 1 Traditional Distributed Transaction Model

Mobile Agent migrates to the targeted host computer independently, performs local interaction, like the following Fig 2 shows. In addition, when remote host computer needs to do data processing, MA can also avoid too many data transmission in the network. Its concept is send computing to process data in the data station, not transmits raw data to requesting station.

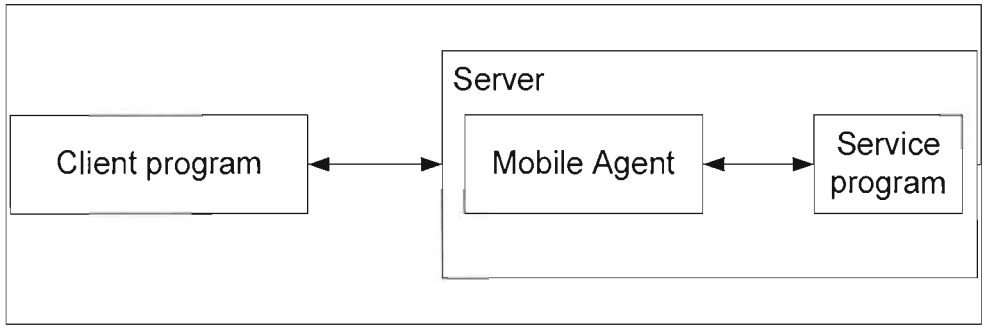


Fig 2 The Interaction Model base on Mobile Agent

From above pictures, we can find out, if a client needs to carry on a large amount of communication with the specific server on the network, the better method is adopting Mobile Agent system, send Agent to the remote server automatically, return after execution of the computing task independently.

The MA interact flow as Fig 3 shows. First of all, the client sends Mobile Agent with the task to server, such as (a) showing. Secondly, when Mobile Agent work in the server, it is unnecessary to keep connection between server and client, Mobile Agent executing task in server independently, like (b) showing, Finally, after finished the task, Mobile Agent sends a request to connect client, establishes

connection, carries the result and return. Such as (c) shows

Based on this method, in the course of computing process, it is unnecessary to maintain the connection between the client and server. Only during a Mobile Agent migrating to a server and return, the client and server need to have connections.

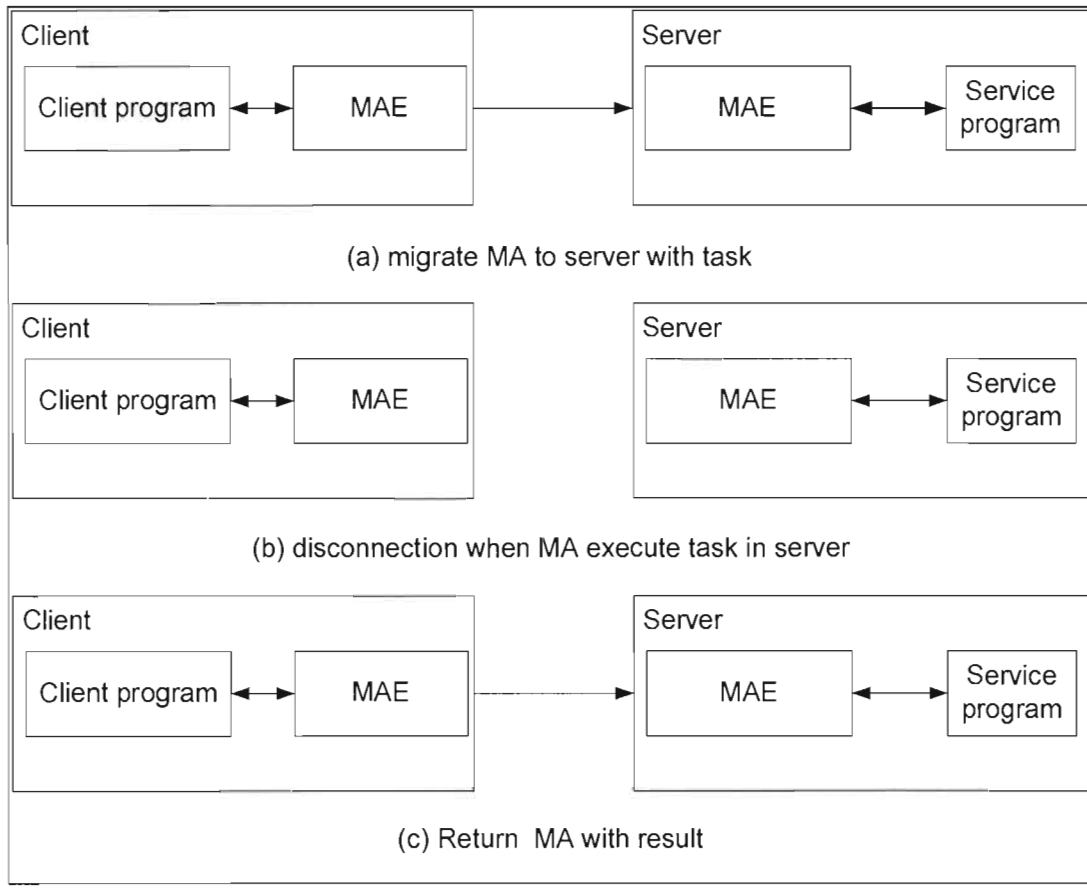


Fig 3 The Interaction procedure of Mobile Agent

From three pictures of above, we can see the flow of Mobile Agent dealing with distributed computing.

Adopt Mobile Agent technology to deal with the distributed task has following advantages:

First, Mobile Agent can reduce the data load on the network.

Through sending the server request Agent to target computer, MA can visit resource of host computer directly. This reduces interact with client computer, and avoids transmission of large amount of data in the network, thus reduces the dependencies on network bandwidth, and shortens communication delay.

Second, Mobile Agent can execute asynchronously independently.

A specific task can be packed into Mobile Agent, and migrate the Mobile Agent that carries the task to target computer in the network, then, close the connection between the host computer and target computer. After that, the MA is working independently on the task that it is assigned, and can operate asynchronously and independently. The source host computer can re-connect with target computer and receive the result of computation in proper time. This is especially useful for mobile devices or mobile users in the future. The current computations on mobile devices depend entirely on real time expensive and fragile network link, it demands to keep connection between mobile device and regular network all the time, this kind of requirement is not ideal neither economical nor technically.

Third, Mobile Agent has stronger adaptability.

Mobile Agent can sense its running environment, and make proper responses to the environmental changes; it can determine to migrate to the target computer dynamically according to the load of the server and network. This helps network load balance.

Fourth, Mobile Agent is easy to achieve parallel processing.

Mobile Agent can dynamically establish many Agents to work in parallel improve efficiency and reduce response delay. Multiple Mobile Agents have unique ability to distribute themselves rationally among the network host computers, and solve a certain specific problem according to certain rules.

Fifth, Mobile Agent has natural different constructing.

The distributed network computer platforms usually have different architectures, and MA is designed to be independent of particular software and hardware environment. It only depends on its running environment, so it is optimum for seamless system integration. [3][4]

1.2.3 Mobile Agent structure model

Fig 4 shows MA architecture, which includes the following interrelated modules:

1. Security Proxy: It is an interact interface for service equipment and external communication. It executes the specific security policy to protect the host computer and arriving MA.

2. Interactive environment module: It provides synchronization protection to communications with MA between local service equipments and external equipment. This module also provides synchronization protection for Multi-Mobile Agent time cooperation.

3. Task handling module: It handles Mobile Agent migration. It regulates task migration according certain rules, and maintains Mobile Agents lifecycle.

4. Knowledge base: It includes an interior state set, instant action database and routing selection tactics. It regulates MA behaviors, including policies of real time change, migration recall etc. It is the guard that ensures that Agents finish their works independently. [6][21][33]

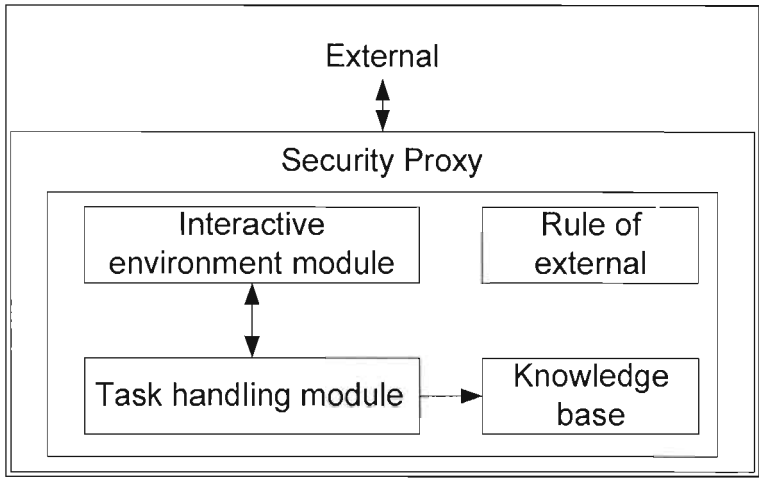


Fig 4 The structure model of MA

From this model, we can see:

- Each Mobile Agent has certain independence, can finish specific tasks independently according to offered constrains.

- Inherent methods of Mobile Agent, such as migration, communication, etc, are encapsulated in the module and interact with running environment instantly.

The corresponding activities of a Mobile Agent are supported by the specific platform—the service equipment of Mobile Agent.

Fig 5 describes Mobile Agent system high-level architecture.

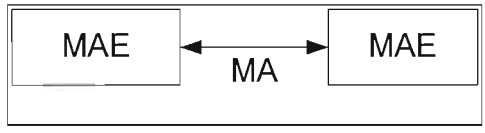


Fig 5 The global structure of the MA

On the high level view, Mobile Agent can carry a task to move among the Mobile Agent service equipments in each host computer of distributed system, after executing the task, carries result back to

the client.

In the architecture of Mobile Agent, Mobile Agent can be subdivided into UA and SA. UA can be moved from one MAE to another MAE. It is executed in MAE, visiting services that MAE offers and communicating with other MA through ACL. The main functionality of a UA is to finish users' designated task; it must perform movement semantics, security and external communication functionalities, etc. SA does not have migration ability, its main functionality is to provide service to local MA or visiting MA, one MAE usually has a lot of SAs, offering different services separately. Because SA can't be moved, and can only be started and managed by the administrator of its MAE, this has guaranteed that SA will not be "evil minded". UA can be local or move to the target site, it can't access system resources directly. It only can visit corresponding resources through the interface of SA after safety verification, thus to avoid bad Agent attack from the host computer. This is the usual security policy adopted in Mobile Agent system. [1-30]

1.2.4 Mobile Agent Equipment

Mobile Agent Equipment is built on top of the operating systems and virtual machines, which offer basic implementation methods for each operation during Mobile Agent's life cycle. It is the platform that Mobile Agent operates. It is also a control platform that a Mobile Agent works and cooperates. Generally speaking, its service configuration should include the following basic services [21-30]:

1. Life cycle service: Control MA's creation, migration, execution on the concrete platform, etc, It

also distributes essential resources to them.

2. Event service: Including Mobile Agent transport protocols and Mobile Agent communication protocols, perform events migrate among Mobile Agents.

3. Application service: Offer the interface for MA to visit local resources. It not only gives permission to allow Mobile Agent to utilize local resource, but also maintain security to avoid unconscious or conscious damages.

4. Directory service: Offer localization information for Mobile Agent and service equipment.

5. Communication service: Provide a transparent communication interface between Mobile Agents or between MAE with Mobile Agent.

Mobile Agent service equipment generally has a communication module, external interface module, lifecycle manager module and local resource API, etc, as Fig 6

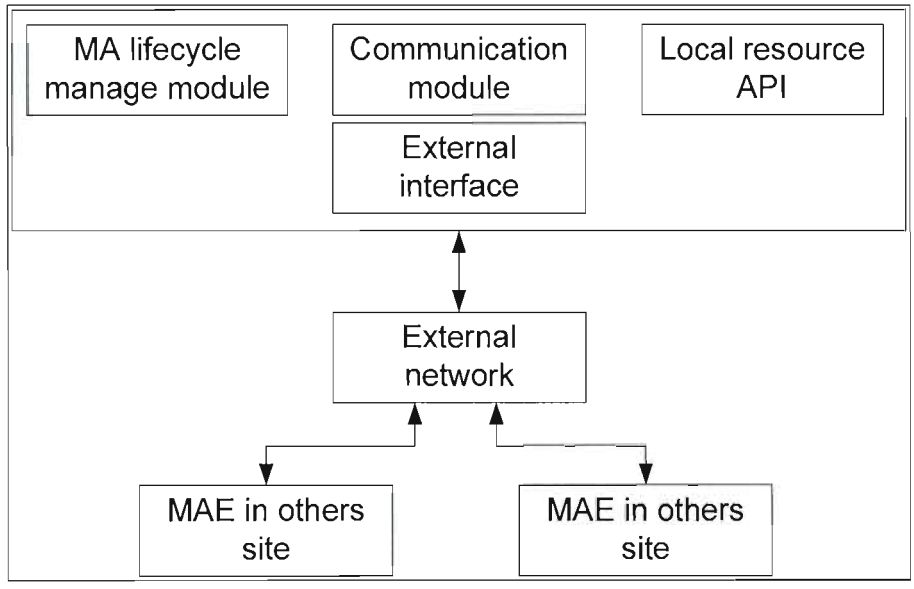


Fig 6 The chart of MAE

Among them, the lifecycle manager module offers lifecycle service for Mobile Agent, offers creation, migration, recall and extract method. The communication module offers directory and communication service, report mutual location and communicate between Mobile Agents. The local resource API offers an interface for Mobile Agent and Mobile Agent to access local resources to accomplish their work. Mobile Agent Equipment interact with others MA in the distributed system through external interface

1.3 Current status of Mobile Agent technology research

1.3.1 The standard of Mobile Agent

For Mobile Agent, there are two kinds of standards are proposed: The purpose of the first one is to promote the interoperability between different MA platforms, the purpose of the second one is to promote the combination of traditional telecom network with the modern computer network, and making progress together. The latter standard has gone through a complicated evolution, and offers a wide space for the MA technology development.

Interact operating standard is mainly MASIF made by OMG. It has defined the functions of inter-operations among MA platforms and defined a public conceptive model; this model includes all abstract concepts that can be found in each MA platform, including MA migration, communication standard, etc.

The representative integration standard is KQML. This standard offers related protocol for MA

communication. It standardizes the communication between MA. It places the function of communication or migrate message on the lower layer. Application related content are placed on the upper layer with some formalized language expression. It defines Meta language for synchronous communication in application layer, such as inquires or answers. [1]

1.3.2 Existing Mobile Agent developing platform

D'Agent (was called Agent TCL) is a Mobile agent system developed by Dartmouth University. It uses RPC, RMI, and other different communication mechanisms. The status information of Mobile Agent can be stored on the physics medium, which makes it easy to recover. On the security part, it uses public key system for identity authentication. It manages the local resource access with static resource management method. It supports duplicate, clone and remote establish etc. [33]

The major characteristic of Voyager is that it uses Voyager ORB as the core; it combines Mobile Agent with distributed computing closely. It only supports Java object communication. It supports both asynchronous and synchronous communication, and supports remote creation, etc. On the security service, it supports SSL communication [33].

From various kinds of platforms we have introduced above, the migration convenience, communicate convenience, security reliability are evaluation criteria to Mobile Agent platform.

In addition, the development cost is a factor that can't be ignored. Aforementioned Mobile Agent platforms are not free. Although they can achieve better performance, this advantage is not decisive, this is the reason I have selected Aglet platform here.

Aglet is a Mobile Agent technology that was developed with pure Java by IBM, and it offers a platform- Aglet Workbench [8] [24].

Aglet development kit can be found from com.ibm.aglet. This kit defines the classification of the basic module of aglet, including Aglet, AgletID, AgletInfo, AgletStub, FutureReply, Message, ReplySet, and the interface: AgletContext, AgletProxy, MessageManager. Next, I describe the kit key components.

1. com.ibm.aglet.event: The event driven mode of Aglet is divided into Clone, Mobility, Persistency, among them Clone event evolves from duplicates aglets, Mobility event evolves from assigns or fetches aglets, etc, persistency event evolves from pause or wake up aglets. This kit has offered the event driven methods.

2. com.ibm.aglet.system: This kit offered the interface of controlling methods.

3. com.ibm.aglet.util: This kit includes common utility methods, such as, AddressBook, AddressChooser, ImageData, etc.

4. com.ibm.agletx.pattems: the methods in this kit can be used for design patterns, such as Master-Slave, Messenger-Receiver, and Notifier-Notification [1] [26].

The following Fig 7 shows systematic frame of Aglet.

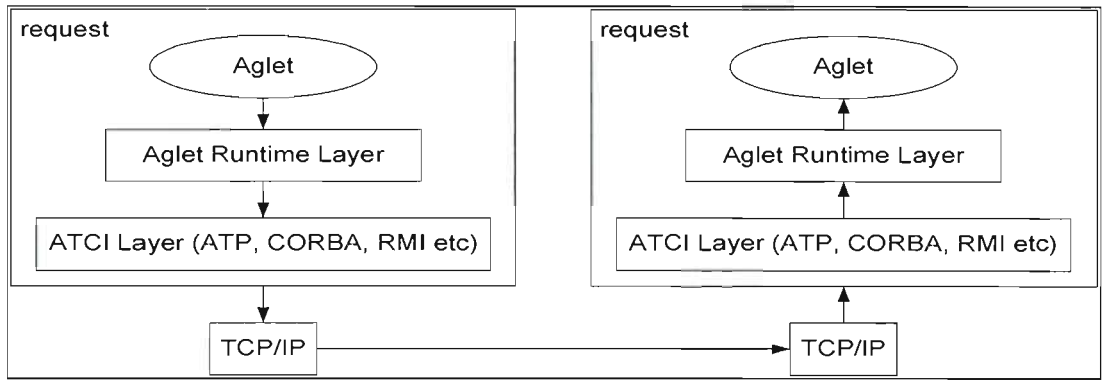


Fig 7 The structure of the Aglet

From Fig 7, we can find out that the execution of Aglet is divided into several stages:

1. When a running Aglet wants to migrate to a remote computer, it should send out a request to Aglet Runtime layer.

2. AgletRuntime layer will encode the Aglet's status information and code into a byte array (serialized). If it is successful, the system will pass the byte array to ATCI (Agent Transport and Communication Interface) layer for processing. This layer offers interface, such as ATP (Agent Transfer Protocol), etc. ATP is a simple protocol in application layer. The following Fig 8 shows that.

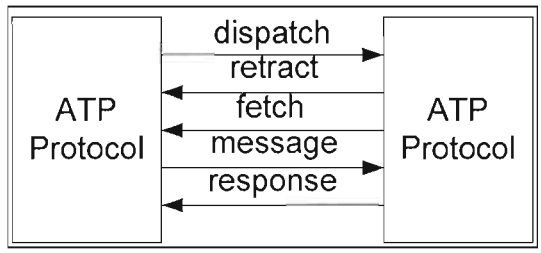


Fig 8 The chart of the ATP Protocol

3. In ATCI layer, system will attach related systematic information to byte stream; and migrate it to remote computer in bit stream.

4. Remote computer use ATP interface that offered by ATCI layer to receive byte array and

system information, then AgletRuntime layer un-serializes the byte array and get the status information and codes of Aglet, Aglet can be executed in remote machine Immediately.

Aglet offers the basic actions in Mobile Agent context: Such as create Aglet, clone Aglet, dispatch Aglet, retract Aglet, deactivate, activate and dispose etc. the following Fig 9 shows that

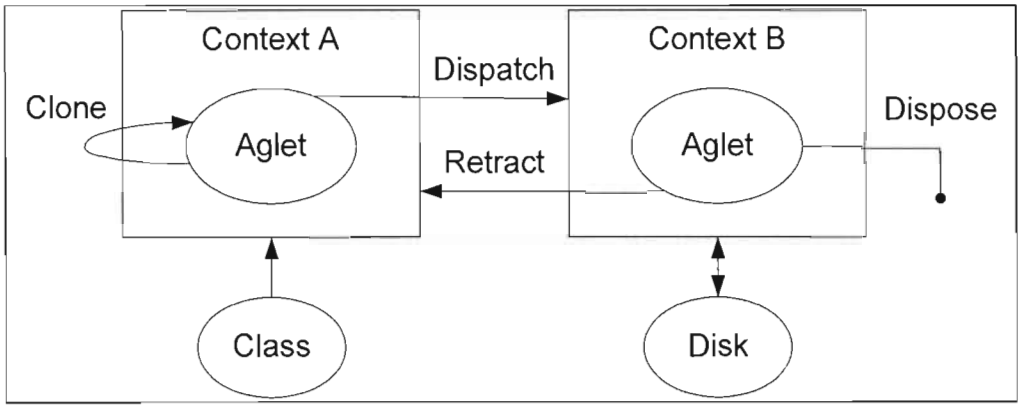


Fig 9 The chart of the Aglet's context environment

The communication of Aglet model is implemented by proxy, the following Fig 10 shows that.

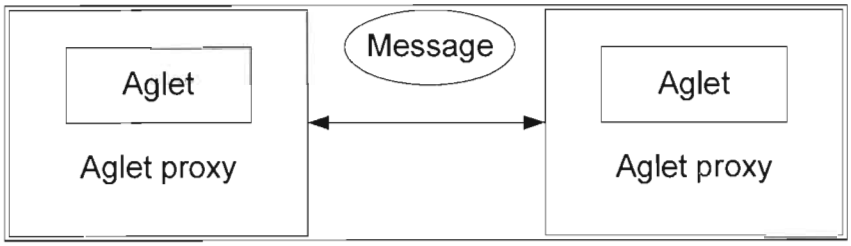


Fig 10 The chart of the Aglet's communication model

I described Aglet developing platform and Aglet system framework. We can find out that the Aglet developing platform and framework is very close to Mobile Agent developing platform. We can use the framework system offered by Aglet to build a distributed data service system based on Mobile Agent [1] [26].

1.4 Chapter summary

This chapter has described some theoretic knowledge of Mobile Agent, Mobile Agent model, methods of performing distributed computing, framework, Mobile Agent service equipment, etc. And, I introduced the develop platform---Aglet.

CHAPTER 2

DISTRIBUTED DATA SERVICE MODEL

2.1 Distributed database technology

The distributed database is a data set whose data belongs to the same system logically, but actually disperses on multiple sites in a computer network. [5][7]

In the distributed data service, there are some problems have to be resolved [30]:

1. Achieve data store location transparency.

2. Divide global data request.

In this paper, I will use Mobile Agent technology to implement a distributed data service model base on the Mobile Agent. In the model, I will use the independence of Mobile Agent to realize location transparency of distributed data servers, use the cooperation and migration convenience of Mobile Agent to move the sub-requests to different sites, reducing the load of the network during the process of data access, improve server efficiency.

2.2 Current status of distributed database technical research

In the distributed data service region, there are many strategies for access optimizations, these theories and methods can reduce the load of the network and improve efficiency of data retrieval.

However, in the process of the data access, there is no change from traditional distributed

computing mode. After a client sends service request to each host computer in the distributed network, the whole flow, including request sending, data obtaining, data retrieval, has to be controlled. There is no independence which can adapt environmental change of the network..

Therefore, while a request that customers send out is executed in the remote computer, the connections between customer and remote computers have to be kept, but this is unnecessary.

In addition, distributed computing cooperation and asynchronies has to be controlled in the procedure, because the control strategy is fixed before implementation, so it can not adapt the change of network, maybe cause the lag.

2.3 Current status on merging Mobile Agent technology and distributed data service

Since the first commercialized Mobile Agent systematic "Telescript" came out in 1994, academia and industrial started to pay more attention to Mobile Agent technology. It has attracted many famous university (such as Dartmouth institute, Cornell university, Stuffgart university), and many companies (such as General Magic, IBM, Mitsubishi, etc), and research institutions (such as DEC research institute, Microsoft research institute in London etc, to research and development.

Because Mobile Agent has more advantages compared with other distributed computing models, many researchers begin to study this, and propose standards and protocols on Mobile Agent systems communication coordination. Many manufacturers released Mobile Agent systems as well, such as Telescript of General Magic Company, Aglet of IBM, etc. [1][7]

However, up to today, Mobile Agent is still in the stage of studying phase, it is only tested in some

applications with preliminary framework. They can only use convenience brought from the Mobile Agent interaction in operation mode, and still need a lot intervention. Some advantages of Mobile Agent, such as independence, intelligent, parallel and synchronous, have not used well in practical applications. So, the existing model really needs some improvement.

In the distributed data service region, there are a lot of visiting and optimization strategies already , they can reduce network load, improve data request efficiency. However, the related advantage of Mobile Agent, for instance intelligent, independence, parallel etc, has not been merged over distributed data service. It only lets Mobile Agent carry the good code to the target site, can't use MA's advantages such as low migration cost, independence, parallel execution of the same task to improve the distributed data service efficiency.

On the basis of existing research, this thesis will propose some improvement, on existing communication, migration, security, etc. And implement a distributed data service model based on this.

2.4 Brief summary

In this chapter, I describe the characteristic and requirement of the distributed data service model, and have discussed the possibility and necessity to merge Mobile Agent into distributed data service model.

Finally, I analyze the current status of merging Mobile Agent technology and distributed data service model. In the following part of this thesis, the proposition of different technologies and solutions are all based on the result in this chapter.

CHAPTER 3

THE PARSING OF INQUIRE SENTENCE

3.1 Global data inquiry parsing theory

The distributed data service system offers global data inquire services for users and application program, the prerequisite of these kinds of inquiries is to resolve these global inquiries transparently [9].

In real operation, user applications only provide logical definitions of these global database inquiries, the physical data are stored in individual physics stations. Therefore, the main process of a distributed inquire is to parse the inquiry sentence of global level into sub inquiries aiming at each physics data site. [10]

In distributed data services, the communication cost of data accesses is a key factor. It is important to obtain data from each distributed physics site with the least cost.

The communication cost can be divided into transmission cost and transmission delay. In terms of transmission cost, the measurement of performance of a strategy is the sum of the transmission cost to and from each individual data site. In terms of transmission delay, the measurement of performance of a strategy is the time between transmission start to end. Shorting transmission delay means improvement of degree of parallelism.

The transmission cost TC and transmission delay TD can be expressed as a linear function of transmission data length X:

$TC(X) = C0 + C1 * X$

$TD(X) = D0 + X/D1$

Here, X represents the amount of data transmission; C0, Cl, D0, Dl are system constants. C0 is the cost of a fixed service charge (such as, connection fee, etc.) for transmissions between two sites, C1 is the unit transmission cost in the network (for example, per bit transmission charge), TC(X) is the cost to transmit X amount of data. D0 is the required time for transmitting data once, D1 is the transmission speed in the network, TD(X) is the time of transmits X amount of data [5].

From above description, we can see that transmission delay is hardware dependent in a great extent. Therefore, reducing transmission cost and localizing data inquires as much as possible are keys for optimization strategy.

3.2 Strategies of optimizing data inquiring based on Mobile Agent technology

For the current status of the distributed database and migration characteristic of Mobile Agent, I will propose the following inquiry optimization strategies.

Here is the strategy of inquiry optimization based on mobile Agent: user input inquiry sentences are parsed according to data location by system. An inquire sentence is divided into one or more sub inquiry tasks base on concrete sites, the application system sends Mobile Agent to each individual site to execute sub-data inquire tasks.

Particularly, we utilize mobility of the Mobile Agent to carry sub tasks to every site respectively. Utilizing the communication module of the target site provided by Mobile Agent Equipment, each

Mobile Agent can interact and cooperate in real time to finish one sub-inquiring task.

3.3 The implementation of transparency in global data inquires

This model offers Distributed data service for application users. It must be transparent to locations of data site. This transparency is provided only for data inquire user applications. For system service implementation, each host computer must have the knowledge of the distribution of global data. For this reason, we have done the following.

In every host computer, we maintain one group of tables. They are used to record how data are stored in others site on the whole distributed system. Table 1 gives the concrete definition

Table 1 The definition of stationinfo

Field name	signification	note
IPAddress	IP address in local	
DomainId	Domain ID	Common key with IPAddress
Port	Port number	

Table 3.1 defines the related information of every machine in the distributed data inquire system. IpAddress and Port are for ip addresses and port numbers to access these machines. In this system, sites can be grouped into regions according to their locations and logically definitions of their data. The column RegionId records the identification number of the region.

On different sites, this table can have different contents. However, this table in different websites must have consistent fields. In fact, user must know these fields in the table before requesting.

In order to reduce inquire cost, we use name stationtable as a name of the table. It describes the

key data distributed on different sites. The stationtable is defined as Table 2

Table 2 The definition of stationtable

Field name	signification	note
IPAddress	IP address in local	Key of correspond field in stationinfo
TableName	Table stored in local	
MainWordValue	Description of table	

The field MainWordValue in stationTable is key field of the table. It stores database table names, called "tableName". If the data that stored in this table has obvious tendentiousness, the information will be saved in this field.

For example, assuming there is a distributed human resource administrative system, its data are distributed into Beijing, Shanghai and Dalian three sites. The data stored in the system are personInfo table with fields including name (name), sex (sex), and workplace (office site). In these three sites, obviously, the site "Beijing" should have workplace filed set as Beijing; this is so-called tendentiousness information.

Here, we use an example to explain; for example, in table stationTable, we see the following value: (202.118.12.34, personInfo, Beijing). That means the table persionInfo stored in IP address 202.118.12.34 (the Beijing site IP) has tendentiousness information as Beijing, this table stores Beijing human resource administrative information mainly.

Utilizing the "tendentiousness information", we can improve the distributed inquire efficiency. However, it does not solve the problem fundamentally. The use of "tendentiousness information" is not

discussed as this model.

3.4 The procedure of parsing Mobile Agent inquiry sentences

3.4.1 Elucidation

This part describes the procedure of parsing inquire based on the Mobile Agent, and it is the core of distributed data service model based on Mobile Agent. The implementation of this procedure is based on the following prerequisites:

First, each site knows the distributed data of other sites in advance. The data distribution information is recorded in stationTable table. The table stationTable stores related table information of all data sites; it is the base knowledge for parsing inquires.

Second, this system use the following unified form to submit sql sentences: "select A.a, B.b, A.c from A, B where A.c = B.c". In this way, during the process of parsing, the system can find out conveniently database location of each field. If there are "and" and "or" operators in "where" sentenes, we use "and' operators first, and then use "or" operators. Certainly, it does not change correctness of data and inquire transparency.

Third, for the type of inquire sentences, such as, "select A.a, B.b from A, B where A.ab = B.ab and A.a> 10 or B.b<5", it has the join information between forms (such as A.ab =B.ab) and constraint information (such as and A.a> 10 or B.b<5). In order to distinguish these two kinds of restraint information, we specially require using two blanks as an interval between two kinds of information.

3.4.2 Procedure of parsing SQL sentences

The target of this procedure is to parse SQL sentence that users submitted. It is divided into following two main steps, first, it parses fields to be inquired, and tables where the fields belong, second, parses field related information after "where" statement, and corresponding restraint conditions joined by "and" and "or" operators.

For the type of sentences in the form of select "A.a, B.b from A, B where A.ab = B.ab and A.a> 10 or B.b<5", first, it finds all fields and corresponding tables form the charater string between the word "select" and "from". The result is saved into "parseInfo" data structure. The data fieldname and tablename in ParseInfo are used for preserving field names to be inquired and tables contain the fields. From information stored in "tablename", we can look at previously defined "stationTable" to find out which sites have has the table "tablename". The IP address information of these sites is stored in ip array.

```
class parseInfo
{
private string[] ip;
private string tablename;
private string fieldname;
//Other methods, such as get and set method
}
```

In addition, to help easy establishing migration target for Mobile Agent, we define the following target data structure.

In target data structure, IP and tablename fields are used to store specific table names and their IP addresses. Certainly, this information can be inquired from stationTable form. However, that requires visiting additional database accesses. The target data structure is generated during the process of producing parseInfo. This will improve efficiency.

```
class target
{
private string ip;
private string tablename;
}
```

The procedure to parse a specific field and its corresponding table is the following: first, we use table field pair "A.a" to know the field to be inquired and table containing the field. Looking up records in stationTable, we find site information where the data is stored. In the same time, we write the information into instantiated parseInfo and target class. Because a table may exist in many data sites, data type of ip field in parseInfo is defined as an array. The target class is the basis for establishing migrate targets after Mobile Agent is created, it stores the mapping information about which sites to go to inquire for tables.

To parse where sub-clauses, we must deal with "A.ab = B.ab and B.bc = C.bc or C.cd = D.cd and

A.a< 5 or B.b> 10" expressions. We use a data structure with "tableRelation" type to record relationship result among tables "where" sub-clauses, such as "A.ab = B.ab", table Information. This data structure is used to describe inter-relationship on tables and relation fields.

The tableRelation data structure is defined as follows:

```
class tableRelation
{
private string[] tablename;
private string relationField;
private string operator;
}
```

TableRelation is used as an array. For example, to the "A.c =B.c and A.b =B.b or C.c = D.d" sentences, "tableRelation1" and "tableRelation2"store two restraint conditions separated by "and" (or "or") operators. The data field "operator" is used to store actual operators, such as "and" or "or".

In tableRelation data structure, tablename is used to store A that in the statement A.ab. relationField is to used to store ab that in the statement A.ab. It defines restraint field among tables.

To non-join expression in "where" sentence, such as "A.a<5 or B.b>10", we save this info into tableInfo data structure in our design. TableInfo data structure is as the following:

```
class tableInfo
{
```

private string tablename;

private string operatorSentence;

private string operator;

}

Above, tablename is used to store table names, such as, A in "and A.a<5", operatorSentence is used to store condition expression, like "a<5" ("A.a<5") in afore-mentioned expression. And operator is used to shows expression operator as "and" or "or", like "and" in afore-mentioned expression.

3.5 Utilizing Mobile Agent to execute distributed inquiring task

The 3.4 Part give final result. It is the prerequisite to utilize Mobile Agent to execute the distributed inquire tasks. The detailed procedure utilizing the result of inquire parsing, sending Mobile Agent to each site to obtain the data, and getting the final result is shown as following.

First, read fields and tables from the resulted parseInfo array, get IP addresses from the resulted target array after parsing SQL sentence.

Second, create migration targets of Mobile Agent using the IP information in target; create sub-task code for concrete sites using information stored in "tablename" in "parseInfo" and "fieldname" field. Here, the sub-task sentence does not include "where" sub-clauses, it will be discussed below, the sentences produced here are similar to the sentences with the following format:

select fieldname from tablename.

Third, using "tableRelation" class, we can find out relation information among lists, their

associated keys, and operators "and" or "or". Using relation information found above, we can add "where" sub-clauses into previously parsed SQL sentences "select fieldname from tablename": adding restraint conditions that are listed in "tablename" of "tableRelation" list; and, in the mean, build the Mobile Agent group needed for communication in this inquiry. Two examples will be used to explain how to add the relation clauses, and build communication methods.

Fourth, using "tableInfoes" class, we can find out restraint conditions about the selection in the concrete tables. Based on these, we modify SQL sentence created in the third step; this will change it into "select fieldname from tablename" + "selection restrain condition of the recorded tablename list".

Next, we use a concrete example to explain the third and the fourth steps.

Example one: assuming list A and list B are stored in two sites TA and TB respectively, we have an inquire sentence "select A.A1, B.B1 from A, B where A.AB =B.AB". In these two sites TA and TB, we will execute inquire sentences "select AB from A" and "select AB from B" respectively. Among them, AB is keys for two related tables. In these two sites, Mobile Agent task put their results into their own temporarily list "tempAB" respectively. Between two sites, migrate temporarily list "tempAB" that include key AB. After one of the sites confirms its intersection set, it returns "tempAB" list to the other site. After that, two sites execute "select A(B)1 from A(B) where A(B).AB = "record in temporarily list" "at same time, and return results.

Example two: we can apply the above procedure recursively, if there are "and" or "or" in where sub-clause in an SQL statement operators. For example, here is the scenario: In sites TA, TB, TC and

TD, there are A, B, C and D four tables separately, the keys joined between these four tables are AB, BC and CD separately, and there is the following inquire sentences on H site: "select * from A, B, C where A.AB =B.AB and B.BC =C.BC or C.CD =D.CD". The concrete execution procedure is:

Firstly execute "select AB from A" on site TA, execute "select AB, BC from B" on site TB, execute "select BC, CD from C" on site TC, execute "select CD from D" on site TD, store the inquire result into four temporarily tables tmpA, tmpB, tmpC and tmpD separately.

Secondly, base on the difference of "and" and "or" operators, access these temporarily lists. Because there is an "or" conjunction relation between table C and D, after migrate temporarily table tmpD on site TD to site TC, execute "select BC from C where C.CD = "every record in tempD" ", merge the BC set got from inquire and the BC set in the tmpC, generate tmpC'. Here, tmpC only has BC records. In the mean time, migrate temporarily table tmpC' to site TB, because there is an "and" relation between table B and C, execute each BC record in "select AB from tmpB where tmpB.BC =the BC record in tmpC' " on site TB, create new tmpB' from AB, and create tmpA' with AB set on site TA.

Finally, every site executes "select * from + list name + where + connection information" according to keys in temporarily table created separately, and return result after finish executing.

Example three: assuming there are record selection information in SQL sentence where sub-clause, and the record selection information is mixed with connection information among tables, such as, table A and table B are stored in sites TA and TB respectively, we have an inquiring sentence "select A.A1, B.B1 from A, B where A.AB=B.AB and A.A1>10 or B.B1<5", the actual procedure is:

First, execute "select AB from A where A.A1>10" sentence on site TA, execute "select AB from B where B.B1<5" sentence on site TB. In both sites, we store inquired result in temporary table tmpA and temB respectively.

Second, exchange temporary tables, so both of the two sites have the combined key set tmpA'(B').

And then, execute "select A.A1 from A where A.AB = "every record in tmpA' " and "A.Al> 10" on site A, and execute "select B.B1 from B where B.AB ="every record in tmpB' " and "B.B1 <5" on site B. At the end, generate result, and return.

From above description, we can find out: if two data tables stored in different two sites, and there is relation between them, the Mobile Agent sent to the two sites need communication, and transmit key set to each other. Meanwhile, they change inquire about the sub task by establishing constrain conditions required, and then execute sub task of inquiring according to the constrain condition. And we can perform above procedure recursively for any complicated situations. Finally, receive the resulted temporary table. The temporary table is the basis of returning data synchronization in the following step.

Finaly, the data return and integration. After finishing execute afore-mentioned steps, Mobile Agents can create the resulted temporary tables in corresponding sites, according to these results, they can generate and return data asynchronously. Using above-mentioned example two to explain, site TA executes every record in the "select * from A where A.AB = tmpA' table". After each site receives their temporary table record, they can send results to inquire initiating host computer. In inquiring initiating

host computer, it uses a temporary table to receive. The temporary fields in the temporary table are fields needed. After receiving these fields, in the host computer, it has all global data already, and can give this result to users.

3.6 Chapter summary

In this chapter, we have used concrete examples, and explained the concept and procedure to utilize Mobile Agent to parse a distributed inquire. In the parsing method of this procedure, it has achieved to divide a global inquiry sentence into sub sentences specific to particular sites according to the distribution condition of actual data, and send sub inquiry tasks to concrete sites.

This parsing strategy utilizes Mobile Agent's migration convenience, mutual cooperating and execution parallelism. It has improved the parallelism during obtaining data from each site. Also, when executing association query, it is achieved by mutual corporation of Mobile Agents. It doesn't need inquire initiating host computer to intervene. This reduces the load of relevant site host computer and network in distributed environment. In addition, Mobile Agent has advantage to deal with distributed computations, it improves the execute efficiency and reduce the cost greatly.

CHAPTER 4

MOBILE AGENT MESSAGE COMMUNICATION MECHANISM BASED ON PROXY AND STORE-SEND

The network environment where Mobile Agent is running is a distributed data processing environment. One of the distributed data service issues in this model must deal with is that every sub task must cooperate with each other to accomplish an inquiring task. Therefore, this model must be able to support message transmission and communication with each other among MAs.

4.1 Overview of basic concept of KQML

KQML (Knowledge Query and Manipulation Language) is a language and protocol that can be used for information and knowledge exchange [11] [12]. In this model, it is used as a language for communication, and exchanged between Mobile Agents.

The core of KQML is a set of expanded Performatives. Utilizing these Performatives, we can create the model [28] of communication among Mobile Agents.

KQML is conceptually layered language. It can be divided into three layers: content layer, message layer and communication layer [1].

Content Layer is actual content that the message carries. The content can be expressed in any kind of representation language. In this layer, the content language translation between their representation and interpretation is done by KQML environment.

Message Layer is the core of KQML, its basic function is to confirm the protocol of migrate message. A sender attach an intension related Performative which is used to demonstrate intension of a message as confirmation, inquire, command or other Performative. Because intension is transparent to KQML, the message layer also includes the parameters available that described intension.

Communication Layer encodes specific information related to both sides of communication parties. It defines the communication format. Therefore, this layer is the foundation of message transmission.

Traditionally, a KQML Performative is also called a message; a typical message format is as follows:

ask-one:

: sender Shanghai

: content(geoloc Chengdu(? long? lat))

: receiverInfoserver

: reply-with location-lax

: language standard_prolog

: ontology geo-model3

Among them, ask-one is the name of a reserved KQML Performative that used for inquire. The words sender, content, in-reply-to， receiver， reply-with language，ontology are reserved KQML Performative parameters. The value of parameter content is an expression, it must follow the grammar

that the parameter language appointed, and the constant among them must be defined in the conceptual relation that parameter ontology defined. Because MAs communicate asynchronously mostly, they use parameter "in-reply-to" and "reply-with" to match out-going inquiries and incoming answers.

Afore-mentioned KQML message can be interpreted as following: The Shanghai airport inquires Infoserver the geographical location of the Chengdu airport. The concrete content of the inquire uses grammar of "standard-prolog" to describe its information, and uses terms in geo-model3 term set. Infoserver uses "reply" Performative to answer the inquiring. In the parameter part, it should have: "in-reply-to location-lax" parameter, it means the answer to the afore-mentioned message [l].

In above paragraphs, we descried KQML's Basic features. This model will adopt a communication mechanism based on KQML, make appropriate definition and improvement on the message layer and communication layer, to improve the efficiency of the communication.

Fig 11 shows Mobile Agent communication system architecture based on KQML.

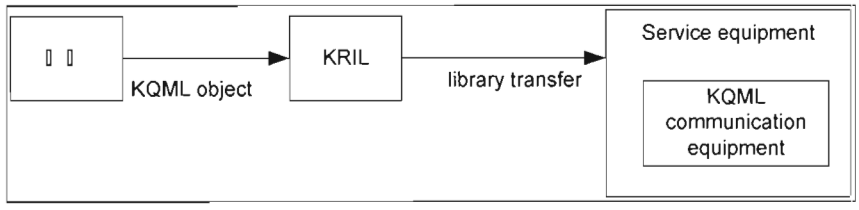


Fig 11 The communication structure of MA based on the KQML

We can find out from Fig 11: The communication system architecture is established around communication function facilities of KQML. In actual implementation, it usually offers KQML Route Interface Language (called KRIL for short), in order to achieve communication among Mobile Agents.

Supported by Aglet developing platform, we can transmit messages directly between Mobile

Agents. However, for security consideration, in this system, Mobile Agents don't communicate directly with each other. Instead, the inter Mobile Agents communication is done through a proxy. It can improve security, and achieve location transparency at low cost. Furthermore it helps to solve the "the communication invalidation " issue mentioned in the following text.

4.2 Issues to be resolved in communication module

In the distributed environment, and the location of Mobile Agents are unknown, how to guarantee the transparency of migrate information among MAs?

MA often faces a so called invalid communication issue in the course of communication [13] [14]. It is caused by random migration of the Mobile Agents who suppose to receive the information. For example, MA A sends message to MA B on Host l, but while message migrating, MA B has move from Host 1 to Host 2, so when message reaches Host 1, it is unable to find the receiver. The invalid communication issue will cause cooperating MA can't got the cooperating status information on time, and cause cooperating failing [21].

In order to solve the invalid communication issue, a MA needs to know the current location that Mobile Agent at any moment. In case that MA is in the state of migration, and unable to receive message, the MA needs to provide message buffering and forwarding mechanism.

4.3 Reference of current research results

This chapter will propose an improved MA position tracking and message buffering and

transmitting method based on KQML and message proxy. This method is built on top of existing Mobile Agent communication method.

There are three kinds of existing methods to solve MA communication problem: Addressing and message transmitting mechanism based on Home Agent; Addressing and message transmitting mechanism based on DNS address tracking and message forwarding according to the routing [13] [14].

The basic concept of addressing and message transmitting based on Home Agent mechanism is: in the distributed system, setup an Agent system on a host computer (Home computer) to record the current host computer address information of all Mobile Agent in this system. It is responsible to receiving, storing and forwarding Mobile Agent messages. The advantage of this kind of message transmits mechanism is that it can achieve faster addressing and messages transmitting. However, the shortcoming is also obvious: If the host computer of the Home Agent has problem, the communication system can be easily broken down [12].

The message transmits mechanism using DNS addressing is: whenever a Mobile Agent migrates to a new region; it registers login information in this region. The register information of these regions uses the tree structured management. Parsing the address with DNS mechanism can achieve tracing of Mobile Agent [12].

Here, upper-level region has all the login information about Mobile Agent of lower-level region. It can avoid the issue of "break down caused by single invalided" in Home agent mechanism. In the mean time, it doesn't need a special Agent server to manage the migrate information of Mobile Agents.

The shortcoming is that system load of upper-level regions can be very heavy, and can easily become a bottleneck of speed and message transmission.

The method of message transmitting by route is: In each region, a MA's login registration information is recorded by every host computer, and each host computer creates a tracing chain according to migration of the MA, migrate message will be forwarded according to the chain [13] [14].

4.4 Solutions

4.4.1 Catalogue Service

In the distributed environment, the communication among MAs is transparent. When an MA can communicate with another MA, it only need to know the target MA identification number, it doesn't have to know the location of its current host computer.

In this model, the communication module of Mobile Agent must be able to locate the position according to the target MA identification number. That is catalogue service, also called position tracing and locating service.

In our communication model, position tracing and locating method is implemented as following:

1. Divide host computers into many regions according to difference of logical function or location. All regions are in parallel relation, and each region has a unique ID (such as 202.118). Every host computer in region will also be assigned a unique ID (such as 76.12); the length of the region ID and host computer ID can be changed according to actual conditions. In a distributed system, the position ID

of a host computer is to combine region ID and host computer ID. One can use mask to separate position ID into region and host computer ID in region.

2. In a region, we choose a host computer in advance, and name it as Route Site. This host is usually selected as a gateway or router that link directly with other region. This host has a Position Table. The position table is used to record each MA's position ID and position information of the current host computer where MA is.

3. When a Mobile Agent is created, the creation host computer will assign a unique ID for it, this ID is generated by combining creation host computer's position ID and a random number. Meanwhile, register the new MA into position table of the region's route site. Because the MA is in the current region, we can let the MA's region ID empty in position table.

4. In case of the MA changed position, if it migrates within current region, when it arrives at target host computer, target host computer will update position table of route site , changing the MA's position to current host computer. Because MA still in this region, and can set region ID to empty in this MA's position table, the MA's position information will be set up as empty as well. If this MA migrates to a host computer of another region, the target host computer will update the position table in both source and target route sites. This MA position in the original position table will be set to target region ID, and MA position in the target position table will be set to position ID of current host computer.

5. According to the information included in the position information table, before a message is migrated, we can find the initial region that the MA is created by its ID. We can further find the MA's

current position information, and find current region and host computer ID from the initial region's route site.

For example, a distributed system is divided into three parts according to its logic function or the region position, the region ID is set as: 202.118, 202.119 and 202.120 respectively. In the region 202.118, create a MA on a host computer with position ID 202.118.76.123. This MA ID will be 202.118.76.123.166.233. Here, the last six numbers are randomly generated by host computer that MA was created. The mask is 0.0.255.255.0.0 which shows that the first six numbers are for region ID of the region that MA was created, the middle six numbers are host computer ID in the creation region, and the last six numbers are randomly generated.

According to the description above, if the MA identification number is 202.118.76.123.166.28, and mask is 0.0.255.255.0.0, we know that this MA is registered in the route site of the region with ID 202.118. After MA is migrated to a new host computer with ID 76.129 in the same region, its updated position table information will become 0.0.76.129. Here, the first six numbers are 0, shows MA is in the same region. The last six numbers shows current host computer ID. If the MA is migrated to 76.10 host computer in region 202.120, route site in the region will update position information in route sites of region 202.118 and 202.120. Change its position information in route table of region 202.118 to 202.120 which shows the MA has migrated to another region, and update position information in route table of region 202.120 to 0.0.76.10.

If another MA wants to communicate with this MA, it finds this MA's region ID (202.118)

according to its ID and mask. It then finds its current position directly or indirectly in position table of that region's route site.

Comparing with the current approaches, it has the following advantages:

1. Avoid point to point localization; it can reduce the scale of question.

2. The route site of every region can manage active Mobile Agent in this region; therefore, it can utilize advantage of "localization principle", and improve catalogue server quality

4.4.2 Storing and transmitting based on message proxy

Mobile Agent has position uncertainty, it brings difficulty for message migrate [15] [16].

In order to solve this problem, this communication module introduces message buffer transmit mechanism based on message proxy.

Introduces message proxy is based on following considerations: The migration of Mobile Agent not only has position uncertainty, also has uncertainty of migration result.

With this, message proxy can stays in original host during the time when Mobile Agent is in migration status, receives messages from others MA, and redirects the messages to target MA in appropriate time.

Message proxy and MA are in one-one relation. It is instantiated with establishment of MAs, follows with MA asynchronously, and released when MA released. In terms of function, a message proxy abstracts the action of receiving, storing, transmitting etc, and achieves message transmission transparency for MA.

Also, a message proxy utilizes a buffer pool to store the messages. In this way, it achieves low coupling of message sent, received and read. The concrete procedure of a message proxy is shown in Fig 12:

1. Together with creating an MA, create a message proxy class to manage message transmit and communication for the MA.

2. In the process of MA migration, its message proxy class will stay in the original host computer, and receive message from others MA's.

3. If the MA migrates successfully, assign a new message proxy for the MA in the new MAE, the previous message proxy will transmit message to the new message proxy.

4. If the MA migrates failed, it returns to the original site. In this case, the previous message proxy stay in original host will continue to be the message proxy for this MA.

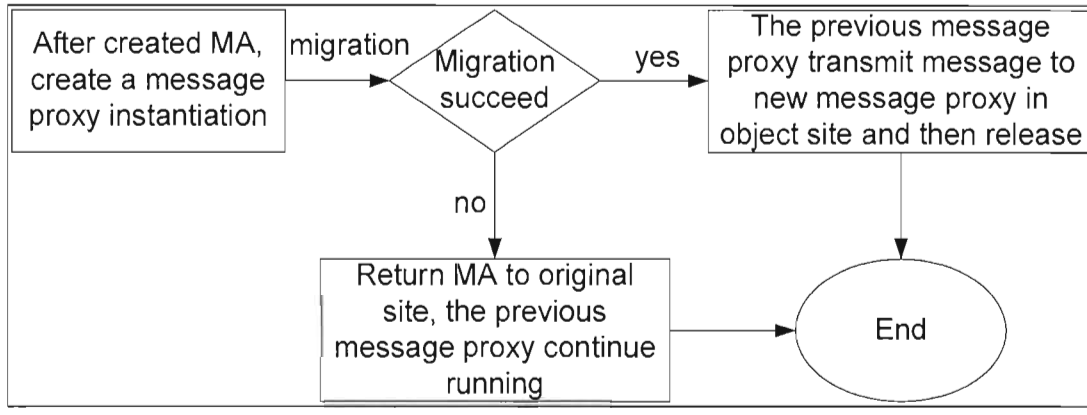


Fig 12 The chart of the procedure of the message store and send

The high level code of a message proxy is simply explained as follows:

/*This interface is used for describing message proxy entity, and message proxy entity is one-one

```
relation with MA.*/
        public interface MAConununicationProxy
        {
        //Used to MA Agent
        private string MAForeignKey;
        //message proxy entity ID
        private string CommEntityId;
        //message class expresses with KQML
        private Message message;
        //the variable of entity's state of message
        private string currentState;
        //Construct function
        public MACommunicationProxy
        (string MAForeignKey,string CommEntityId);
        /*Receiving message from the original address, message stores in the style of KQML with the
format of Vector*/
        public void ReceiveMessage(string sourceAddress, Message message);
        //Send news
        public void SendMessage(string ObjectAddress,Message message);
```

```
//migrate to the new MAE
public void MoveTo(string objectAddress);
//return
public void RollBack(string sourceAddress);
//According to the target MA address, obtaining should the current position of MA
public string getObjectAddress(string MAID);
//The following is set up and obtain private variable, pass over
.....
}
```

The following is a definition of the message:

```
class Message
{
//target address that will be sent
private string objectAddress;
//The following is the information of encapsulation, it can consult above-mentioned KQML
.....
// The following is set up and obtain private variable, pass over
.....
}
```

In this communication system, we specially reserve the words of Table 3 in content. They are used for controlling the communication procedure.

Table 3 The explanation of the reserved words in content

String	signification	explanation
Connect	Connection request	When two message agents communication, they must send this string to connect
Apply_connect	Apply to connect	When message agent receive the connection request, send this string, if it agree to connect
End	Message transmitted completely	After transmit, send this string to disconnect
Busy	Apply, request to delay the connection	If message agent B is migrating, reply " busy" to other message agent A' s connect request. A enters sleep state. When B migrates finished, send " apply_connect" to A, and wake it

The communication between MA is described concretely as follows. For example, entity A communicates with entity B

The actions of entity A and corresponding message proxy are:

1. Entity A invokes position seeking method in the service of catalogues, and gets the current position of the target entity B and corresponding message proxy.

2. Message proxy A sends a communicate request to Message proxy B, sets content of the message as "connect".

3. If the reply is "busy", then message proxy A enters the sleep state to wait until it receives a message "apply_connect" from agent B, and wake up.

4. When message proxy A gets the "apply_connect" reply from message proxy B, it sends a message to message proxy B. Here, the messages class's content is communication content.

5. After all messages are sent, message proxy A will send "end" message to message proxy B.

The actions of entity B and corresponding message proxy:

1. Monitor the network: when it receives a "connect" request from message proxy A, if message proxy B is in the "migration" state, it replies "busy". After MA B finishes migrate, it sends "apply_connect" to message proxy A; If its state is not "migrate", it sends "apply_connect" immediately.

2. When a message is received, it is stored in message proxy B, and wait for transmit command.

3. When entity B receives an "end" message, it acts according to the position of entity B. If they are in the same host computer, transmit the message directly; if entity is succeed in migration, it finishes this migration; if the migration failed, it returns to the original site, and resumes the data and state.

4.5 Implementation of communication module

The communication service module in this mode is shown as Fig 13

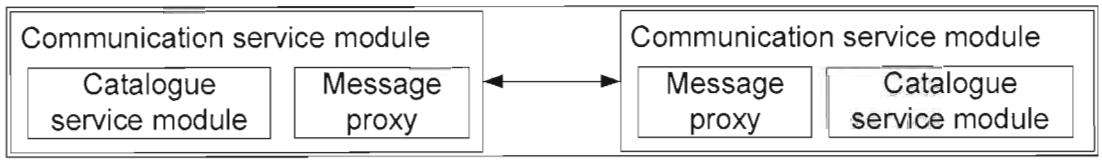


Fig 13 The chart of communication service module

The catalogue service module locates a MA's current position according to ID of the target MA. After confirming the position information of the target MA , store and transmit message by message proxy.

4.6 chapter summary

In this chapter, a communication model of Mobile Agent is provided. In this model we have used KQML as the communication language, and unified communication standard.

At the same time, based on the current research, we have provides catalogue service mechanism of Mobile Agent. Utilizing this mechanism, we can make the best use of "localization principle", and achieve the communication orientation transparency in a low cost.

And, in this model, we adopted store and transmit mechanism based on message proxy, it can effectively solve the communication invalidation problem from Mobile Agent migrate uncertainty.

CHAPTER 5

IMPLEMENTATION AND IMPROVEMENT ON MOBILE AGENT LIFE CYCLE MODEL

5.1 Summary of Mobile Agent life cycle

According to description of Mobile Agent rules, in the life cycle of a Mobile Agent, the basic actions include creation, duplication, migration, deactivation, activation and destroy, etc. These actions are supported via the life cycle management module of Mobile Agent [1] [16].

A Mobile Agent life cycle manage module can initialize Mobile Agents. MA can migrate to other host and continue executing using methods offered by the life cycle model under the direction of its task. Before a MA migration starts, it assigns work and relevant information for the MA, and deactivates its operation in local host [1] [31].

After Mobile Agent is migrated to a new host, it can continue to work, or migrate, until the task is complete and destroyed. The diagrams in Fig 14 show a flow chart of MA life cycle.

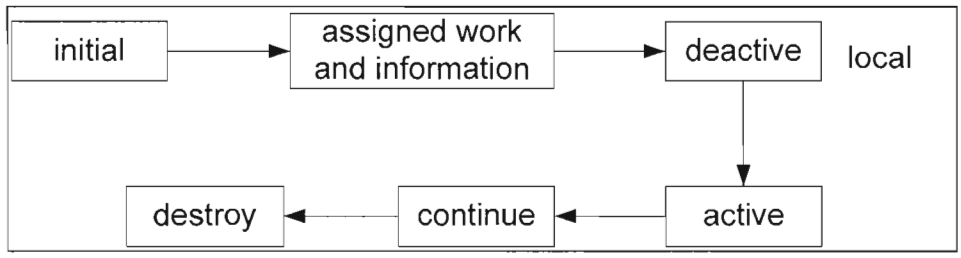


Fig 14 The procedure of the Mobile Agent's life circle

5.2 Implementation of life cycle

In this model, we define a base class based on Mobile Agent object model characteristics. In this base class, basic characteristic of Mobile Agent and common methods are defined abstractly. Application program can inherit this class to implement Mobile Agent various kinds of methods in life cycle. This definition of the class is as following:

```
Abstract public class MobileAgent
{
AgletID sid;            // the only ID of Mobile Agent
String sname;           //the another name of Mobile Agent
Object info;            //Used for storing the information that Agent carries
URL homeURL;        //Show URL of the host computer
Final Object publicKey;       // key that proved for ID
Vector moveObject;    // the information including of the target site that migrate to
//Create Mobile Agent, construct function
Public void initial ();
//Send to the target site
Abstract public void Dispatch (URL sourceURL);
//Deactive Mobile Agent
Abstract public void Deactive ();
```

```
//Active Mobile Agent, change MA state from deacrive to active
Abstract public void Active ();
//Destroying Mobile Agent, destructor
Abstract public void onDisposeQ;
//Followings is for sending messages and receiving messages
Abstract public Boolean receiveMessage(Message msg);
Abstract public Boolem sendMessage(AgletID sourceAgent,Message msg);
//The following is the description of Mobile Agent security part
.....
}
```

The rough flow to utilize this method to implement life cycle of a Mobile Agent is: create a Mobile Agent class using method new, invoke initialization method to initialize, and allocate relevant resources to the Mobile Agent. After a Mobile Agent is created, local host can assign work and relevant data information for it, the task and data information can be recorded in info, and migrate along with the Mobile Agent.

When a Mobile Agent finishes its work in its current host, and needs to migrate to a remote site, the local computer invokes deactive method to stop its execution in local host, and invoke dispatch method to start migrate.

After a MA arrives at a target site, the target site can invokes active method in proper time to

activate the Mobile Agent, and resumes its state and data. After that, the MA can records its current state and data information into info file, and utilizes resources to continue to work on the remote site.

5.3 Implementation and improvement of migration module

Invoking Aglet environment method can achieve MA migration. However, in this model, migration module needs to communicate with security validation module. Therefore, in this model, it is necessary to design a new migration module.

On the basis of studying current research results, in this paper, we proposed some necessary improvement for MA migration module to enable effective interaction with security validation module.

5.3.1 Current research results

For Mobile Agent, there are two kinds of migration approaches: Jump model and known entry point model using itinerary.

In jump model, the system offers a jump performative, it collects all states of current Mobile Agent automatically including breakpoint and resources needed, and sends them to the target host computer. It can also direct the new site to resumes these states once it arrives at the target site. Jump model is very convenient to user, However, its requirement to Mobile Agent platform is also very high [25].

In known entry point model, system sends variables and methods that an Agent is using to target host computer, and let it continue to run from current break point in new site. To Java-oriented system,

the system can get all the target's states automatically according to class and method it offered [1] [16].

In this design, we have selected known entry point model provided by Aglet development environment. It allows a Mobile Agent migrates sequentially according to pre-defined itinerary methods in advance.

According to the arrangement of itinerary, a Mobile Agent migrates to each host computer sequentially, executes its task, and returns result at the end.

5.3.3 Design of Migration module

In this paper, based on current research results, we make necessary improvement to migration subsystem; merge it with security validation module. Fig 15 shows its flow chart.

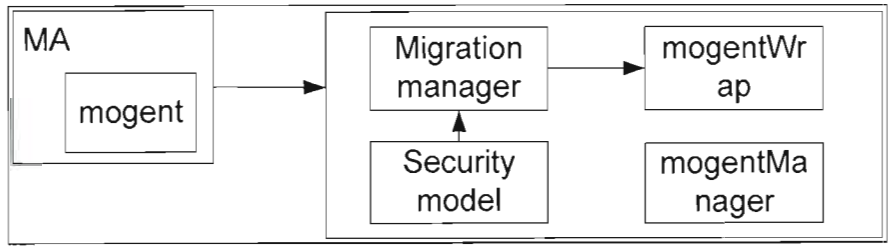


Fig 15 The chart of the migration model

Followings are explanations for each part among them:

Mogent part: Abstract migration characteristics from each Mobile Agent.

MogentWrap part: provide local resources to MAs that have passed security validation.

MigrationManager part: This part is responsible for monitoring Mobile Agent that coming from the network. When a Mobile Agent arrives, it invokes the security policy that stored in security model to validate the Mobile Agent. After validation, it passes the Mobile Agent to mogentWrap for further

processing.

MogentManager part: This module encapsulation Mobile Agent "route" information.

Security model is responsible for security validation. It will be explained in the following chapter.

5.3.4 The workflow of the migration module

The migration model flow is as following Fig 16

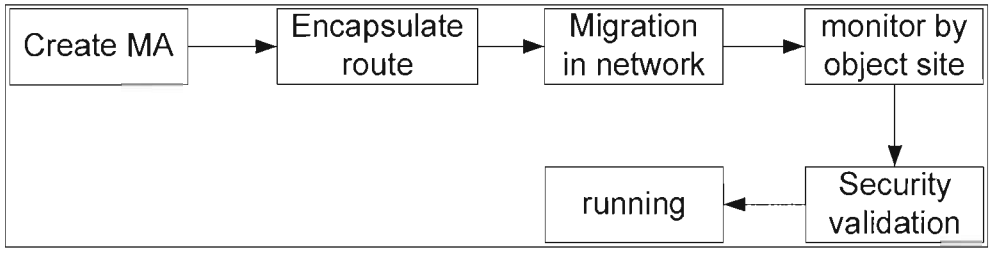


Fig 16 The chart of the procedure of MA's migration

MigrationMmager module monitors Mobile Agent that coming from network; the migration information in MA is described by mogent handle.

When mogent arrives at the host computer, migrationManager starts to parse commands in mogentWrap, and parses the task carried by the MA.

After mogentWrap parses a task carried by a MA, it invokes security model for security validation and identity verification, and opens local resources accordingly.

MogentManager confirms migration routing according to the service equipment's migration instruction.

5.4 Chapter summary

In this chapter, we provided a definition of MA life cycle and implementation specification of every operation in life cycle, and suggested an improved implementation and workflow of the migration module. The migration module after improving can be well integrated with the security module together, and ensure a Mobile Agent service equipment security when it receives the MA.

Use the method provided in this chapter, we can control MA initialization, migration, parsing operation, etc. It can also implement the Mobile Agent whole life cycle procedure.

CHAPTER 6

THE SECURITY IMPLEMENTATION

IN DISTRIBUTED ENVIRONMENT

6.1 Mobile Agent Security requirement in distributed environment

In a distributed computing environment, Mobile Agents and Mobile Agent service equipments are placed in the open network environment, the security concerns are particularly important issues. In a distributed environment, the security issues of this model are mainly on following three parts:

1. Maintain data security during Mobile Agent entity migration [29].

2. Host computers that have Mobile Agent service equipment can be easily damaged and attacked maliciously by Mobile Agent entity. Sensitive information can be stolen or copied after validation with false identity [28] [29].

3. After Mobile Agent entity arrives at a target site, sensitive data in MA can be easily changed or destroyed by host computer intentionally or unintentionally [17] [18].

For aforementioned first issue, to ensure data security of Mobile Agent, we use the traditional security safeguard, such as encrypting, digital signature, etc.

For the second issue, Mobile Agent service equipment can adopt an identity verification mechanism to prevent maliciously invasion by evil Mobile Agents.

And the third issue should be resolved by improving internal structure of MA and add specific

prevention mechanisms.

6.2 Description of current achievements

6.2.1 PKI encryption

PKI (Public Key Infrastructure), it utilizes an asymmetrical encryption algorithm. It offers a good security mechanism needed in network.

The traditional single key encryption uses specific common keys to encrypt and decrypt data, and the keys for encryption and decipherment are same, that is called as symmetrical encrypt algorithms.

Using this encryption method, transmitting encrypted data in the network must send the encryption key to recipients at the same time. The third party may decrypt or change migrated data by intercepting encrypted data and key.

Instead, PKI uses an asymmetrical encryption algorithm, that is, encrypting key is different from decipher key. This can prevent a third party from decrypt data by obtaining the key.

6.2.2 Digital signature

The digital signature is defined in IS07498-2 standard:

> "*Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the unit and protect against forgery e.g., by the recipient.*"[17].

6.2.3 Digital certificate

The digital certificate is also abbreviated as the certificate. It is the core elements of PKI; it is the basis of digital signature technology. It is part of X.509 standard. It can be a proof of entity identification in network; prove the identity legitimacy of one's entity and public key, and matching relationship between an entity and public key. A certificate is a carrier of a public key; it binds a public key and an entity's identity [17].

6.3 General description of security module

In this system, to protect data security during transmission, we use the method of data encryption, and prove the legitimacy of the digital source through the digital certificate. In this security module, it uses RSA (a digital signature algorithm) to ensure that Mobile Agent is not distorted during migration.

For the safe guard needed by service equipment and Mobile Agent, this design provides a security module in Mobile Agent service equipment. It implements information encrypting, digital signature and identity verification. Another important function of this security module is to protect host computers that support the Mobile Agent service equipment to avoid bad Agent from attacking and damaging. The Fig 17 shows its flow chart:

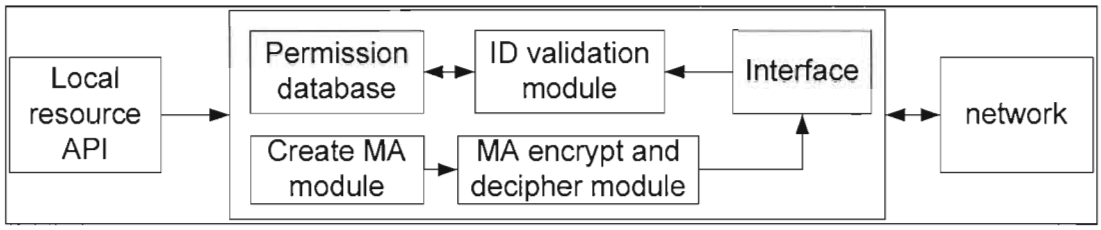


Fig 17 The chart of the Mobile Agent's security model

In this flow chart diagram, the modules to implement security function are explained as follows:

1. MA encrypts and decipher module includes a key used for encrypting. Using this key, it can create digital signature after an MA is created in life cycle module, encrypt information that MA carries, thus it can ensure MA data security during migration in the network. To a MA that is migrated from a remote machine, decipher data with an appointed key, and then, pass it to the identity validation module to verify its identity, and execute corresponding task. In this way, it can guarantee the legitimacy of MA's original source.

2. The identity validation module is responsible for verify the MA identity that come from network, and distinguish from its permission, prevent unverified Mobile Agent from entering to host computer. So, it can ensure service equipment of MA not being invaded by hostile Agent, thus improve the security of the service equipment.

3. In the permission database, a key tuple type data structure stores the relation of digital signature and visiting permissions with API, this relation is also called the security policy.

In this security system, the explanation of security module is as following:

1. The MA creation module is an abstract of MA life cycle module in the MA service equipment.

Among them, there are MA creation, task distribution, and basic data encapsulation that contain basic data and state. Before migration, the information among them waited for MA encrypt module to encrypt protecting.

2. Local resource API is the interface for all kinds of resources. When a MA entity migrates from remote to local, and passes verification of identity, local computer open local resource API for it selectively according its privileges.

3. Migration module. The security module in the migration module is this security module. When MigrationManager in the migration module detects a MA arrives, it use the policy provided in the security module to confirm the identity and privileges of this MA, and allocates corresponding resources for it.

6.4 Security guarantee procedure

In this security guarantee system, the security related procedure is as following:

1. When the system creates Mobile Agent, it allocates task and creates migration route, and encapsulates digital signature of identity and visiting certificate of authorization on remote computer. For MA's state and information, it utilizes the key to encrypt according to RSA algorithm in PKI algorithm.

2. When the host computer detects a Mobile Agent arrives, it verifies MA's digital signature that includes identity of its host computer. If it doesn't pass verification, hang up the Mobile Agent, and do some clean up work, such as logging a record, tracing the IP, etc.

3. After a MA passes its identity verification, it opens API resources in local host according to Mobile Agent certificate of authorization.

4. If the target computer still needs to migrate Mobile Agent to other sites, it changes the original digital signature to a new digital signature that identifies the new host computer,, and then assign a corresponding visiting certificate to substitute old one according to the new Mobile Agent task assigned by the local computer.

6.5 The mechanism of identities verification and security to the service equipment of MA

In order to prevent hostile MA from attacking the service equipment of MA, it needs to add the identity authentication mechanism to visiting MA in the service equipment of MA [32].

In order to prove one's own identity and authority, MA should carry an identity digital signature of the host computer that creates the MA and a visiting authorization certificate of MA privileges.

When a MA arrives a target host computer, the security module in the service equipment confirm it is sent by a legal host computer according to the digital signature, and furthermore awards privileges according to its visiting authorization certificate. In addition, once security guarantee module finds that a visitor's identity fails to verify, it will refuse to provide service for this MA immediately, trace the invader as much as possible, and create a log.

The service equipment of MA reads the visiting authorization certificate that a MA carried, it awards different privileges of open API, according to local visiting policy. The concrete method is: In MA system, to each local API, for instance, the database read, the database write, file access, specific

file modification, all files modification etc, form a local resource visiting policy table, Table 4 shows the following.

Table 4 Local resource visiting policy table

Resource API	Authority state	Time limit
Read database	Authority	20060313
Write database	Non-authority	
Read special file	Authority	20060315
Read all files	Non-authority	

In different host computers in a distributed system, different authorization certificates will be granted on different API, according to their actual conditions.

In different host computers in a distributed system, after a Mobile Agent is created, and its migration target and the route includes many sites, different API access authorization certificates will be granted, and each certificate should include all target sites the MA will visit sequentially.

For example, Mobile Agent is created in host computer A, according to its route, it should migrate to host computer B, and host computer B needs to send this MA to host computer C again. First, host computer A gives authorization certificate "Ca" to host computer B. After it arrives site B, and, before lefts site B, site B will give API visiting authorization certificate "Ca' " of MA for host computer C.

When a service equipment of a host computer receives Mobile Agent from network, it deciphers the digital signature that encoded from this MA's identity. If there is a mismatch, this MA entity is hung up immediately. If identity revivification passes, it continues to read API visiting authorization

certificate, and, open concrete API resources accordingly.

The implementation details of verification API visiting certificate and opening API resources are: Add to a locking variable as parameter into concrete resource access API, for instance, for visitDataBase (String sql, Bool dbvisitLock) API call; dbvisitLock is the lock variable to control visiting database, if the lock value is false, it does not open the API.

After a MA arrives, security guarantee module will create an array of locks according to its API authorization certificate that carried by MA, as following:

```
class lockset
{
private bool dbvisitLock;              //Initial value is true
private bool dbwriteLock;              //Initial value is false
//Corresponding gets and set method.
.....
}
```

The method of verifying identity of visiting Mobile Agent can prevent from non-local MA system sending MA enter into local machine, can also open proper local API according to actual demand of different MA requesting.

6.6 Security measurement for Mobile Agent

6.6.1 The security requirement of Mobile Agent

A mobile Agent is active in the open distributed system environment. Therefore, in the whole life cycle of a MA, it has the following security requirement:

1. During migration, ensure the security of data information.

2. After a MA Arrives at a host computer, prevent the host computer from destroying information carried by MA intentionally and unintentionally [19] [20] [32].

Particularly, the afore-mentioned first security requirement should deal with the following safe threat: during MA migration, a third party may steal the data content, even distort the information among them through communication channel [23] [24].

The second security requirement should face the following threat: The host computer will attack a Mobile Agent intentionally and unintentionally [32].

When MA executes in a target host computer, it equals to reveal its own private information to target host computer, include source computer sensitive information and the data content and current state. If the target host computer crashes while a MA is running, or attacks voluntarily, it is very easy to be stolen, and even distorted to data information carried by Mobile Agent.

6.6.2 Security measurement and implementation

For afore-mentioned first security requirement, the method used in this design is: Use RSA

encryption to protect information of Mobile Agent. And, the encryption key is not migrated together with data. In this way, even if malicious host computer steals the information, because there are not keys, it is still unable to decipher the real content among them. And, based on the theory of digital signature, if a data is distorted after encrypting protecting, it will not match with its digital signature. If MA service equipment discovers mismatched content and sign, it will demand this MA to re-send again, or perform other error handling.

For afore-mentioned data privacy of the second requirement, the traditional method is to define read-only information or information that cannot be serialized to be private or final type to protect the variable.

But, hostile host computer can obtain or distort protected data through revise compile mechanism of virtual machine to attack.

In order to resolve this issue, we provide two implementing schemes in this design:

1. Through the encryption mechanism, set up the internal key code in Mobile Agent to read-only, the MA that arrives at a target host computer after migration has to pass the verification to read the encrypted data content.

However, in real world conditions, target host computer should have authorization to write internal records of a Mobile Agent. For example, after a target site receives one MA, it needs to send this MA to another host computer again, and needs to change the information of migration target, etc. Also, the information should be read-only after it is written, can't be distorted by other host computers.

To cover this situation, the second implementing scheme is proposed as well.

2. Set up a buffer pool especially for writing into in Mobile Agent; write into the buffer pool operations should be verified, in the mean time, encryption and read-write verification must be done to the content in this buffer pool.

As to the first implement scheme, in this system, concrete implementation method is to set up a read-only data structure as following:

```
//This data structure used for storing read-only content
class readOnlyPool
{
//Target to be protected
private Object protectedContent;
//The reader can only access the read-only, i.e., the protected content through this interface
public Vector Read(Object key) {}
/*This method verifies the visit request according to the key that host computer offers. If verification is passed, read operation will be authorized */
private Boolean verify(Object key) {}
}
```

After a MA is created, the MA life cycle module creates a readonlyPool for it to store read-only information with encryption protection at the same time.

Use this method; a hostile host computer can distort data between MA initial creation and migration. Also, it can revise the read-only content only of it knows the encrypting algorithms the key of MA host computer otherwise it will cause mismatch between data content and data signature. However, this is almost impossible, because one of PKI encryption assumption is: only encrypter can hold its encryption key, other entities can not have it.

After the host computer finishes encryption, MAs abandon the private key, and migrate to the target host computer through network. If the target host computer needs to read the content among them, it can only offer the public key to the Read interface for this data structure, after Read method invokes verify method to verify, it could read the protected content. With this, it implements the function of read-only information reading after verification; it also ensures that read-only information can not be distorted.

For afore-mentioned the second safety measurement, the implementation method in this paper is: create a buffer pool that "can only be written once, and then can only be read". Concrete implementation is: create a data structure in MA, its content can only be read after identity verification. And, its data can't be modified or deleted. The concrete definition of the data structure is as follows:

```
class addonlyPool
{
Vector protectedObj;                    //Target to be protected
Object signature;                       //Corresponding signature
```

```
Hashtable content;          //It stores the protection target and corresponding signature key
Object addURL;              //store URL of the adder
Byte[] checkData;           // the byte array that used for detect whether the data is distorted
//Construct function
public addonlyPool
(Vector protectedObj, Object signature, checkData)
//Basic assignment operation
Object signResult(Object o)
{
/*Use the private key to create digital signature for the target, the return is the result after data
signature*/
}
/* Encrypting with the private key that this machine includes, the parameter spread into is the
content to be encrypted */
public void addbySign(Object protectedObj)
{
//The step is as follows:
//Use the private key included in this machine to encrypt protectedObj
//add encrypting result and corresponding signature to one's own data structure and Hashtable
```

```
}
//Verification method, its parameter is a private key
public Boolean verify(privateKey p)
{
//Using the private key to encrypt, its result is assigned to checkData
//Abandon the private key after encrypting
}
}
```

In this class, it includes target to be protected, digital signature of protected target and digital signature operator (owner of the protection target), etc.

CheckData, this attribute is used for detecting whether the data is distorted, its initial value is: PKI (HK), among them, HK is a private key to the host computer, and the result resulted from this expression PKI (HK) is the information flow after encryption.

The target MA host computer can use addbySign method to add the data to addonlyPool. In the process of adding data, it should use the private key of the host computer to sign the target. After the data is added, the value of changed checkData is:

checkData =PKI(checkData + digital signature of added content+ password which is different from other host computers)

After a Mobile Agent returns from the target host computer, it can detect whether the data in

addonlyPool is distorted using verify method. The detection method is: Decipher checkData data, and uses verify method to verify its content. If there is a mismatch that proves that is distorted, abandon it immediately.

The process of Executing the verify method must guarantee that protectedObj recovered by private key must match with its original.

After migration, if the new host computer tries to modify protectedObj that is protected, it can only calculate again using the private key to match it with corresponding signature. However, it will cause checkData value mismatch with original in this way. Because checkData is encapsulated with legal digital signature of protectedObj when calculating checkData, and initial encrypting checkData relies on local computer password, so the hostile host computer is unable to perform distort operations through modifying checkData value.

Using afore-mentioned safe mechanisms, if a hostile host computer distorts the data, it will result in error during data verification, thus leave traces. This kind of method has improved the system security through strengthening the degree of difficulty when the data are distorted and stealing. It can ensure Mobile Agent security during migration and remote migration in the MA service equipment.

6.7 Summary

In this chapter, we analyzed that the system service facility of Mobile Agent and the security threat that Mobile Agent may be met in open network environment, and has proposed corresponding solutions.

This chapter proposed security measurements (criteria) of the service equipment of Mobile Agent, they can reduce the possibility of service equipment attacks by maliciously MA, can also enable the service equipment of Mobile Agent based on different authorities to open local resources selectively.

This chapter proposed a security guarantee mechanism of Mobile Agent, it can reduces the possibility that the hostile host computer attacks other host computers through stealing sensitive information while Mobile Agent visiting, can ensures Mobile Agent security during migration effectively.

CHAPTER 7

THE DISTRIBUTED DATA SERVICE MODEL BASED ON MOBILE AGENT

7.1 overviews

In this chapter, I will use the strategies and methods that provided in the above several chapters, to implement the distributed data service model of Mobile Agent concretely. Before explaining this model, I first give following explanations:

1. In this model, each site can creates and migrates Mobile Agents to other sites. Each site also has a service equipment of Mobile Agent at the same time, and can admit Mobile Agent arriving from other sites and work in local machine. For description convenience, we call the host computer who initiated a request as a client host in one transaction of data service, call the host that creates Mobile Agent as source site or source host computer of MA, call host computer that Mobile Agent need to migrate to as target site or target host computer.

2. A database request can be classified as local request and remote request. However, in the description of thereinafter, I will not focus on local request, only emphasis on remote request and utilize MA to implement distributed data service.

3. This system is platform independent. In order to concentrate attention on the concrete flow and concrete implementation, our experimental platform system runs on Microsoft Windows, instead of

heterogeneous operating systems such as Linux or UNIX, etc. We use Access database instead of more power SQL Server or Oracle with secure and better functionalities. In fact, Java virtual machine mechanism can guarantee platform independent, the database that is connected with database access interface layer is just a simply abstracted database. So, this assumption does not affect the platform and database independence in this model.

4. The security of local database access is not emphasized in this paper's discussion. Therefore, I will not discuss security guarantee of local database access, for instance, daily record, recovery, affairs.

7.2 The general explanation of the model

The distributed data service model is constituted by MA service equipment on each site in distributed system and MA's that are dynamically created by service equipment.

MA service equipment is constructed on Java virtual machine and operating system in host computer of each site. They are independent to each other. Each site in distributed system can submits service requests to distributed system through the service equipment.

The system will parses global requests according to data distributing situation of each site, migrates MA to other sites, submits parsed task, execute relevant request on the site, collects and returns results, passes back to users after combining parsed sub-requests.

In consideration of expansibility and portability, in this model, it only provides a simple and transparent interface for data request module and network link. For inside implementation mechanisms, like, how to create Mobile Agent, how MA migrate, how to communicate, what is distributed inquire

process procedure, and MA service equipment's security mechanisms, are hidden to users.

The model's detailed framework is shown as the following Fig 18

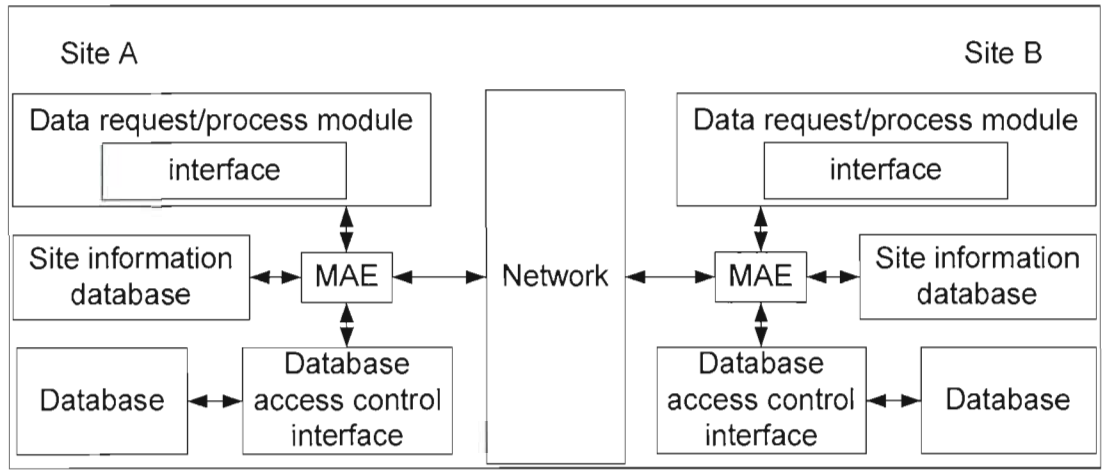


Fig 18 The chart of the distributed data service based on the Mobile Agent

Fig 7.1 has described the whole framework of the model. From top level of point of view, on each physics site of the distributed database, this distributed data service model is based on Mobile Agent. The model is consist of the data request/process module, the interface of the data request/process module with MA service equipment, MA service equipment, information warehouse of network site, database and database access and control interface.

Data request/process module is a functional abstract of application data request operations. Users only need to present inquiries based on the overall conditions to this interface; this module can transparently return users requested data. This module communicates with MA service equipment through an interface; the function of this interface is to hide detail information about data distribution for user.

The site information database stores position information and data distribution information of each site in distributed system, and is used for parsing inquiries and Mobile Agent migration.

Database access control interface is used to hide low level database distribution detail. Through this interface, the application can access database transparently and indistinguishably.

Mobile Agent service equipment is the core of this distributed data service model. It controls and coordinates the creation , communication, migration and working etc for MA. Through the interface, MA service equipment interacts with data request module internally, and interacts with MAE of other sites remotely. Fig 19 shows framework of Mobile Agent service facility.

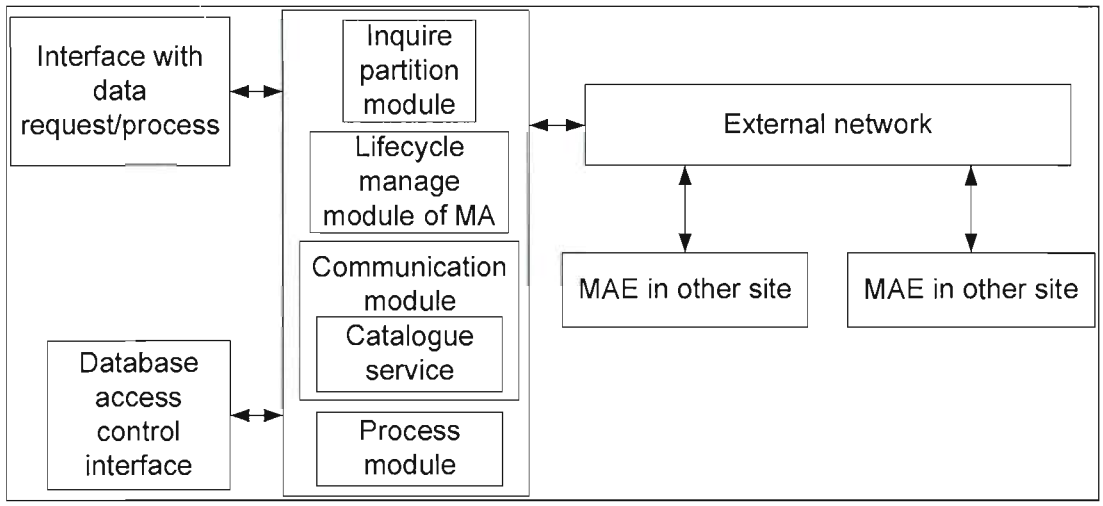


Fig 19 The chart of the MAE

7.3 The definition of each module

7.3.1 Inquire parsing module

In this module, it receives global inquire requests from data request/process module, parse them

into sub-requests, and then send these sub-requests to the lifecycle manage module of Mobile Agent. Mobile Agent life cycle manage module creates and migrates MA to other sites to receive data according to sub-request inquire task.

The third chapter, I have provided the parsing idea and roughly flow, here I present the code to explain detailed parsing steps.

In inquire parse module, the information needs parsed includes: field set of inquire needed, table set that contains the fields, the condition of relation information and record selection of tables after "where".

The first thing needed is to parse required field set. According to agreement, the field set in SQL sentences to be submitted should follow the following format:

select A.a1, B.b1 from …, etc. This sentence begins with keyword "select", and ends with the keyword "from". The procedure can be explained as the following pseudo code:

```
void splitField(String sqlString)
{
//CurrentPos variable shows the position of current code execution
int currentPos =0;
//CdPos expresses the position of the first comma since current parse point
int cdPos;
/*The following two variables shows the beginning of the individual field and end identification
```

position*/

int beginFieldPos;

int endFieldPos;

//The beginning is counted from the blank after keyword "select"

//Set up beginFieldPos according to afore-mentioned methods

While (test the current execution position is not the end position, i.e., the character "f" in keyword "from")

{

/*Set current position as the beginning position of this parsing, if there is a comma next, use it as the end position of current parsing, if there isn't a comma, use the character "f" in keyword "from" as the end position of current parsing.*/

/* Use "." as separator, parse fields and tables that fields belong to. */

/*Save required fields and tables in data structure "parseInfo" respectively.*/

/*Set current position to the end position of this parsing, prepare for next parsing.*/

}

}

The following method can parse the information of each data table in SQL sentence, the initial parse position begins with the first blank after keyword "from", ends with character "w" in keyword "where".

```
void splitTable(String sqlString)
{
//Current process position
int currentPos;
//Position of the current comma
int cdPos;
//Beginning and ending position
int beginFieldPos;
int endFieldPos;
/*The beginning position is counted from the blank after keyword "from"*/
//Set beginFieldPos according to the afore-mentioned methods
While (test the current execution position is not the end position, i.e., the character "w" in keyword "where")
{/*set the beginning position of this parsing as current position, if there is a comma next, set the end position of this parsing as position of the comma, otherwise, set the end position of this parsing as character "w" in keyword "where"*/
/*parse table name, and retrieval its position from stationInfo list and then store data structure parseInfo and target in respectively*/
// Set current position to the ending position of this parsing, prepare for next parsing.
```

}

}

The following method parses the constrain condition of each table in SQL sentence, and store results into the corresponding data structure.

void splitWhere(String sqlString)

{

// Current parsing position

int currentPos;

// Beginning and ending position

int beginFieldPos;

int endFieldPos;

/*parse the expression that connects between tables in where sub-clause*/

/*Set up beginning position to start from the blank after keyword "where"*/

While (the current position is not the position with two blanks)

{

/*set current position as the beginning position of this parsing, if there is a keyword "and" or "or" next, use it as the end position of this parsing, otherwise, set the end position of this parsing as the position with two blanks*/

//Parsing A.AB =B.AB and (or) B.BC =C.BC

/*save the character string before "." into tablename of tableRelation, save the character string after it into relationField, save "or" or "and" into operator. Set current position to ending position of this parsing and prepare for next run.*/

}

//parse the record selection expression in where sub-clause

//Set the beginning position to start as the first effective character after double blanks.

While (the current position is not the end position)

{ /*Set current position as the beginning position of this parsing, if there is a keyword "and" or "or" next, use it as the end of this parsing, otherwise, set the end position of this parsing with the end of this SQL sentence*/

//Parsing statements like A.al>50 and (or) B.bl <20

/*save the table names in front of "." into tablename of tableInfo, save the character strings after that into operatorSentence, save "or" or "and" into operator. Set current position to ending position of this parsing and prepare for next run.*/

}

}

Based on afore-mentioned parsing result, using the distributing situation of data tables in each site that is recorded in stationInfo list, create parseInfo, target, tableRelation and tableInfo, and other related data structures. The parsing result provides basic information of distributing task for MA and

establishing migration target and communication target.

7.3.2 The life cycle manage module of Mobile Agent

In this module, it encapsulates methods of MA creation, migration, releasing, etc. This module uses some of life cycle support methods provided in Aglet to define a method, and encapsulate some common methods according to Mobile Agent's life cycle criteria [1]. Next, we give a detailed analysis of the concrete flow in life cycle using pseudocode.

The pseudocode of initializing MA operation is as following:

```
//Initialize Mobile Agent
public void initMobileAgent()
{
//The flow is described as follows:
/*1. To instantiate Mobile Agent class.
2. Assign information, such as, the identification; create creator's IP, etc. At same time, register this MA into routing list of this region.
3. Create inquire task and communication target, and initialize required data structure.
4. Encrypt task and data to be carried.
5. Create the target of migration, and wait to migrate.
*/
}
```

```
// method for Mobile Agent migration
public void moveto(stationInfo objectStation)
{
//The flow is described as follows:
/*1. To assigns the verification information that the target site needed for this MA.
2. According to task that MA carried, provide API certificate of authorization to the target site for this MA.
3. Invoke the encapsulated migration method in the life cycle module to migrate, and change this MA position information in routing list in this region at the same time.
*/
}
//Release Mobile Agent is equivalent to call destructor of MA
public void deleteAgent()
{
//The flow is described as follows:
/*1. Release resource.
2. Delete MA information in routing list in this region.
*/
}
```

7.3.3 Catalogue service module

The catalogue service module is included in the communication module. This module implement searching and location of MA in whole distributed system using the only identification assigned during the Mobile Agent's creation. The searching and location service is the foundation of Mobile Agent communication.

Please refer chapter 4 (on communication module) for concrete implementation detail. Here, we provide the implementation flow:

1. Location module obtains the identifications of communication target of Mobile Agents.

2. Using the information of creation place included in its identification, we can learn the region and host computer that creates Mobile Agent. Look for the Mobile Agent's current location information in routing list's location registration list in the region that creates the MA. If this MA executes in the same region, it can locates the current position of Mobile Agent directly.

3. If this MA has migrated to another region, we can find its current region using the record in the region position register. This record was saved by the MA. In the region position register, we can locate its position according to the MA's identification.

In fact, according to localization principle, it is a great possibility to locate Mobile Agent in region that the MA is initially created [21]. Find and locate service module in this model can locate Mobile Agent in this region very efficiently.

7.3.4 Communication service module

In chapter four, it mentioned the necessity and implementation method of utilizing message proxy to store and redirect message. The following Fig 20 shows workflow of this communication service module.

According to Fig 20, the workflow of the communication service module is:

1. The communication module obtains a message, encapsulate it with KQML, and then initialize the message entity.

2. Invoke method in the localization module; find the communication host computer according to the target MA's identification.

3. Create communication channel.

4. Send message entity with message to target message entity according to message store and transmit mechanism.

5. Target message entity implement store and transmit according to the position of its corresponding Mobile Agent.

6. Release the channel.

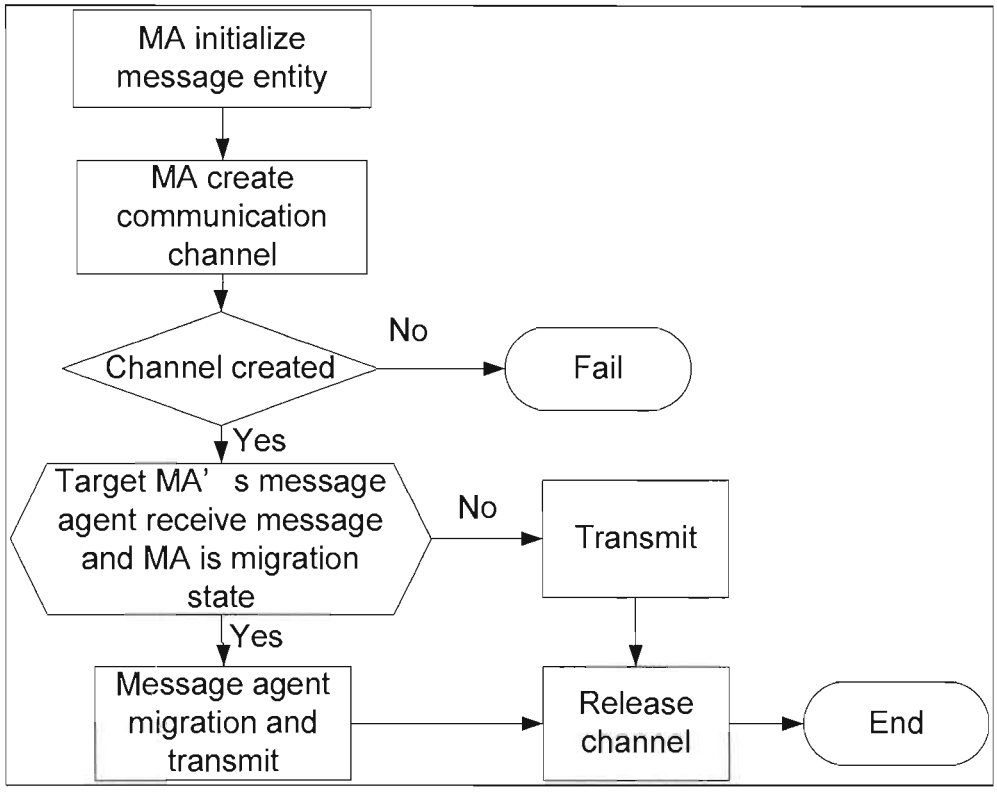


Fig 20 The chart of the work procedure of the communication model

7.3.5 The database access and control module

This module is the interact interface of the MA service equipment and local database resource, as the following Fig 21

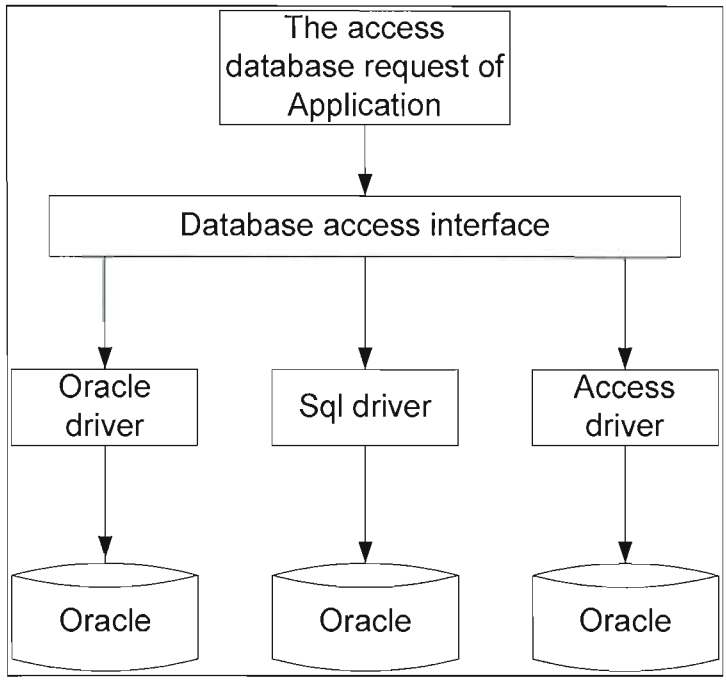


Fig 21 The chart of the DB access-control model

This module uses a high level mode to implement Mobile Agent access to database transparently and independently, the transparency and independence is achieved via database access interface provided by database driver.

According to afore-mentioned frame, database access interface can shield the difference of database access details through database driver.

In addition, for optimizing efficiency of connection, this database access module utilizes a connection pool to allow applications to share one group of cacheable connection object handle.

The key code is as follows:

//According to the name of connection pool of database driver, initialize connection pool

//Utilize the driver to shield the differences of databases

```
DataSource ds = (javax.sql.DataSource)initCtx.lookup(dsName);
/*Here, ds is a handle of the data pool; we can obtain the connection handle of the database through ds*/
//obtain connect handle from connect pool
conn =ds.getConnection();
//we can use connect handle to execute various kinds of databases access operation
```

Utilizing the database access module can shield the difference of databases, and improve the efficiency of database accessing through the connect pool as well.

7.3.6 The explanation of security guarantee Module design

The implementation techniques and process flow of security guarantee are discussed before. Here, we conclude the implementation of security guarantee from a point of view of MA and MA service design.

Among them, the security guarantee criteria of MA include:

1. The service equipment assigns a unique identification and digital signature during creation of Mobile Agents, this signature is used to verify source legitimacy of MA.

2. Encrypt information MA carried, use an asymmetric encrypt method; and send the key and cryptograph at the different time.

3. Once a MA arrives at a target site, before opening critical information access to the MA, it needs to present a password that the target MA equipment provided. According to result of this password

verification, it will be granted various authorizations to read and write critical information respectively.

4. Before a Mobile Agent migrates to a new destination, sign digitally to sensitive information, and migrate with encrypted data only.

When target site modifies sensitive information, it needs to modify the cryptograph properly. After the Mobile Agent returns, the service equipment in the original site will use the encrypted data to judge whether some critical information is distorted or over-written by unauthorized third party.

The security guarantees for MA service equipment criteria are:

1. It monitors MA arrivals from network. When a MA arrives, it needs to show its digital signature, and judge whether it is dispatched by service equipment by this system, and according to its certificate to assign it authorizations to use local API properly.

2. It opens API authorization properly according to the policy of its certificate.

We ensure MA information security in the process of network transmission through digital signature; ensure MA service equipment security through the identity verification. Guarantee Mobile Agent security while working in the service equipment through the proper encryption to Mobile Agent.

7.4 Implementation of platform independence

Because it is hard to predict the target host computer's concrete information arrived by MA, such as, the access methods of the operating system and database type. So, in this model, various kinds of methods offered must be the platform independent.

From the point view of development environment, the platform independence of JAVA language

and database independence of JDBC are the foundation of achieve platform independence. And, in the design of this paper, the communication, interact and migration operations between MAs use ATP protocol or those standard interface. The protocol and criterion used have offered the method guarantee for achievement of the platform independence.

Except convenient factor of the system and environment offered, in this design, the independence of the platform is also concretely reflected in two following respects:

First, positioning this model as a middle ware, define the interface its data request/process module and network layer according to unified protocol and criteria, and separate its detailed operation logic and implementation.

Second, add database management module in database access layer, this module shields database, and implements database access transparency.

7.5 Workflow of this model

The following Fig 22 shows workflow of this model:

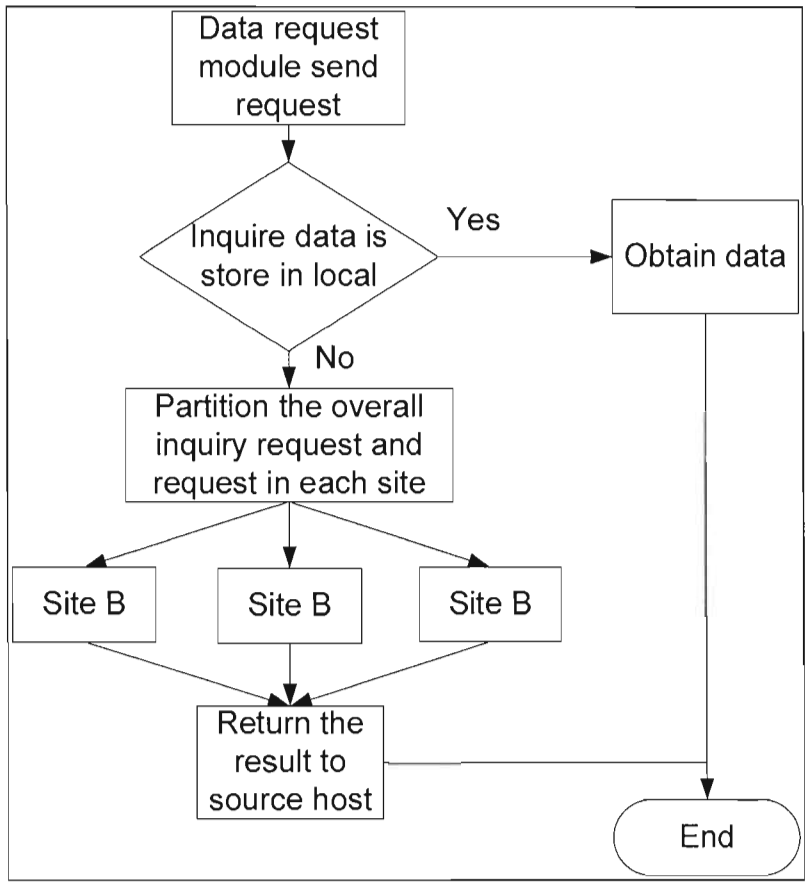


Fig 22 The chart of the procedure of the query using the model

1. The data request module sends out the overall inquiry requests. If a data inquire request can be completed in local site, it obtains the data from the local database, and the task is over. If the local databases are not sufficient to finish the inquiring request, it continues the following steps; starts to distribute data parse flow.

2. Inquire parsing module in the Mobile Agent service equipment parse and parse inquire according to the afore-mentioned inquiry parsing algorithm. Combining the data list stored in local site with system site distribution situation, parses the overall inquiry request into sub-inquiry requests

specific to local or remote sites.

3. MA life cycle management module creates concrete MA according to the inquire request specific a target site, and initialize.

4. MA migrates to the target host computer on network, through interface with external module of MA service equipment on the target host computer, after the validation and authority of identity, it is allowed parsing and executes the task carried by the MA in the target host computer.

5. The target host computer execute task carried by MA. At same time, according to actual conditions, inquire request code calls corresponding API to access database through database access and control module, according to inquire sentences obtain result, and create the result set.

6. After MA obtains the result, the Mobile Agent communicates with source host computer, returns the inquired result set to the source host computer.

7. After Source host gathers all results set returned by the MA, submits final result through the interface of data request module and MA service equipment.

7.6 Summary

This chapter is described on the basis of individual implementation technology in this model, and detailed explanation of the frame, workflow and key modules in this model.

In this chapter, we utilized different modules based on MA to "assemble" a distributed data service model. Certainly, this "assemble" is not piling up on the code level, but a seamless link on module and frame level.

In the following chapters, we will present an actual application of distributed data service model based on MA, and application result and general analysis of the model provided in this chapter.

CHAPTER 8

THE APPLICATION OF THIS MODEL

8.1 The application in project management system

This distributed data inquiring module is used in a project management system. In this system, the application of this model is illustrated in Fig 23

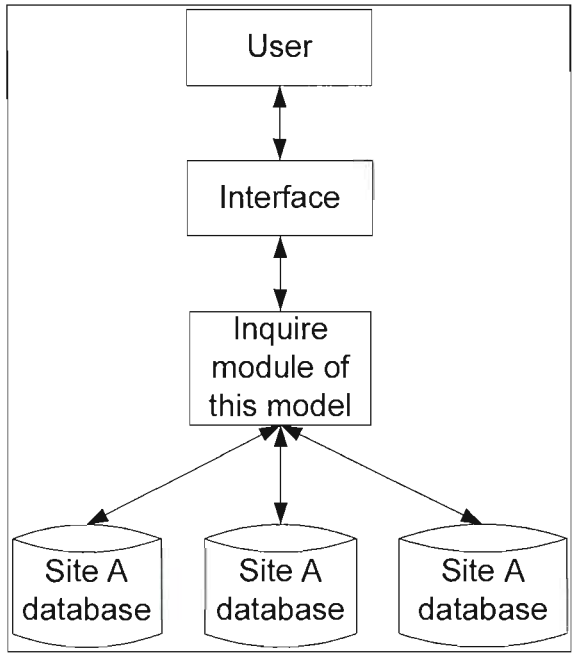


Fig 23 The chart of the usage of the model in the on-line shopping system

In this system, the process of users' submitting inquiries and getting data is the following:

1. Users submit global inquiries to request module interface.

2. The inquiry module sends inquiry task to other sites in the system according to data distribution condition in the distributed database.

3. In the whole distributed system, the inquiry task is accomplished through cooperation and communication among the Mobile Agents.

4. Return the final integrated results to users.

From the above Fig. 8.1, we can see, in the system, this inquiry module is used in the inquiry module of each site. All project database sites form a distributed data service environment. Among this distributed environment, each site finishes its inquiry through the migration and cooperation of Mobile Agent.

8.2 The figuration of systematic surrounding

Before using this system, we should simply configure the information of each site using the interface offered by this system.

Through the interface in Fig 24, administrators can modify or delete project information of each site.

Please select the record that need be changed

column name	num	project name	address	current cost	finish time	schedule(%)
○	1	traffic control system	202.115.17.133	70000	2006/08/07	60
○	2	traffic control system	202.115.17.149	3000	2006/09/13	90
◉	3	traffic control system	202.115.17.121	35000	2006/12/24	70

OK CANCEL

Fig 24 The chart of the interface of setting the information

8.3 Data extracting flow

This part will use real world examples to demonstrate the process of distributed data service which touches the parsing of SQL sentence, communication between Mobile Agents, MA and MA service Equipments' security criteria and data integration.

8.3.1 A simple problem

8.3.1.1 Description of a simple problem

In this project management system, there are many sites, assuming they are sites A, B, C, in different places. The database tables on each sites is all named as project with fields name, current cost, finish time(time) and schedule etc.

The user M (data applicant) inquires from site D, asking for project's information with name "traffic control system". Because there are project tables on all three sites, the data needed may be in these three sites. Therefore, Mobile Agents must be dispatched to afore-mentioned three sites to complete the inquiry task jointly.

8.3.1.2 The concrete procedure

1. On site D, application program sends inquire requests to data request/process module. The corresponding options that the user needs to fill in are shown in Fig 25. Press the "inquire" button, and then enter the next step to execute the concrete inquiry. Through "SQL Parser" option, the result of

analyzed SQL sentence will be displayed.

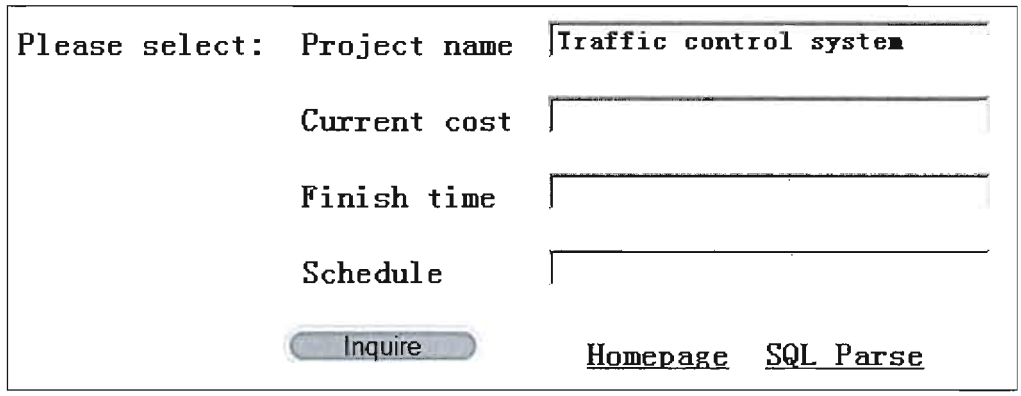


Fig 25 The chart of the user setting the inquire interface

After the system accepts users' submission, it creates a global SQL sentence:

select * from goods where name = "traffic control system"

After the system receives the inquiry, it extracts local data first to see if the data is available on local host. Here, we will not discuss local inquiry in detail, and emphasis on the procedure of acquiring remote data utilizing MA.

1 . Using stationIp and tablename fields in stationTable table, the usable information can be searched out as the following Table 5

Table 5 The distribution of Table

stationIp	Tablename
Ip:202.118.76.125(A)	Project
Ip:202.118.76.123(B)	Project
Ip:202.118.76.124(C)	Project

2. The table shows that the table project is in above three host computers with three IP addresses,

(Note: Here, the IP address data are specially treated). Therefore, the distributed data service system needs to inquire the data in all afore-mentioned three IP.

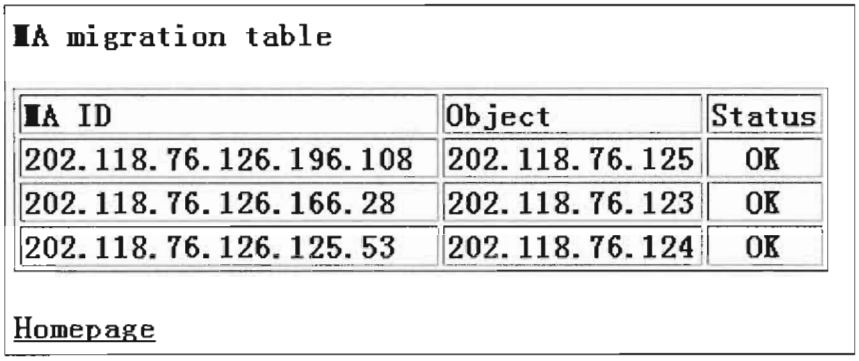
3. Because joining among multi-tables are not involved, three Mobile Agents can be dispatched directly to the three sites to execute the inquiry. Three Mobile Agents don't need to communicate each other, and the tasks they carry are all same, i.e., select * from project where name ="traffic control system". The result of analyzing SQL sentence is showed in Fig 26. Object indicates the MA's ID number, and SQLsentence records actual SQL sentence carried by the Mobile Agent.

Object	SQL sentence
202.118.76.126.196.108	select * from project where name ="traffic control system"
202.118.76.126.166.28	select * from project where name ="traffic control system"
202.118.76.126.125.53	select * from project where name ="traffic control system"

Homepage

Fig 26 The chart of the parser result

4. The MA life cycle module in the MA service equipment creates three Mobile Agents according to the information in stationTable and encapsules the information such as source address, migrating target address and carrying tasks etc. After security setup is finished, and when Status is OK, the migration command is sent out. The MA migration target list is showed in Fig 27, among which MAID is MA's identification number, Object is MA migration target and Status expresses this Mobile Agent is in the state of ready to migrate.


MA migration table

MA ID	Object	Status
202.118.76.126.196.108	202.118.76.125	OK
202.118.76.126.166.28	202.118.76.123	OK
202.118.76.126.125.53	202.118.76.124	OK

Homepage

Fig 27 The chart of the MA migration list

5. After three Mobile Agents reaches the destination separately, they login to target site, and go through identity verification.

6. The host computers of three target sites grant authorization database access according to the authorization policy carried by them after identity verification.

7. The Mobile Agent executes the task carried themselves; access database, perform inquire, and write result into temporary tables through database access control module in this way, three temporary tables are created separately in three sites.

8. Three sites send their temporary tables back to site D respectively.

9. After site D receives data, it integrates three tables and submits to the data request/process module. The project member M receives the final result, which is showed in Fig 28 as following.

Project name	current cost	Finish time	Schedule
traffic control system	70000	2006/08/07	60
traffic control system	3000	2006/09/13	90
traffic control system	35000	2006/12/24	70

Homepage The first page Next Previous The last page

Fig 28 The chart of the inquire result

10. This inquiry is completed. The system releases the communication channel and the Mobile Agent.

The communication between Mobile Agents is not involved in this simple inquiry flow. It will be described in detail in more complicated cases below.

8.3.2 Complicated problem

8.3.2.1 Description of a complicated problem

In this system of shopping on network, there are many sites, supposed site A, B, C, are in different geographical places. Site A has project table (defined as 8.3.1.1) and company table, site B has project table, and site C has company table.

The key field of location is defined in the following table 6. This table is mainly used for describing the information of the project company.

Table 6 The description of Table location

Field name	Definition	note
Name	Project name	Key with name of Project table
Company	Descript member of project company	

The user G (data applicant) submits inquiry to inquire project and company information with cost more than $50000 or schedule smaller than 50%. The submitted SQL sentence is:

select * from goods, location where project.name =company.name and project.cost > 50000 or project.schedule < 50.

8.3.2.2 The concrete flow

1. Initial inquiry request is sent out from site D. The interface is similar to that in Table 7. In this interface, the corresponding items are filled in and press the "inquire" button, and then enter next steps. Here, we need to demonstrate that the goods price and price quantity input frames supports fuzzy logic inquiry. If we want to inquire "the cost equals "50000", we input "50000" in the goods price frame directly. Here, if we like to inquire "cost is higher than "50000", we can input "> 50000" in the text frame, the system will determine this according to the first character. If the first character is a number, the operator is "=". If it is not number, the character before the first number is assumed to be the operator.

2. Based on stationIp and tablename fields in stationTable table, retrieve out useful information as

shown in following Table 7. The information gives the distribution condition of the data:

Table 7 The distribution of Table

stationIp	Tablename
Ip:202.118.76.125(A)	Project, Company
Ip:202.118.76.123(B)	Project
Ip:202.118.76.124(C)	Company

The afore-mentioned information shows: the site with IP 202.118.76.125 has project and company tables. The site with IP 202.118.76.123 has project table, and the site with IP 202.118.76.124 has company table.

3. According to sationTable table and inquiry sentence submitted by the customer, the global oriented inquiry sentence is parsed. For MA to migrate to site A, its inquiry task is as the following:

select name from project,company where project.name=company.name and select * from project,company where project.name = company.name and project.cost>50000 or project.schedule<50.

The first inquiry sentence inquires the foreign key from two tables. The requiring result is saved into a temporary table tempPK(A) and it is used as the basis to communicate between MA B and MA C.(Note: Here tempPK(A) means the name of the temporary table is tempPK, and it is created in site A. We will use the same convention in the following text.)

For MA B to migrate to site B, there is only project table in site B, so the inquiry sentence is:

select name from project and select * from project where project.cost> 50000 or project.schedule <

50

Similarly, the result returned from the first inquiry sentence is saved into tempPK(B) and it is used as the basis of communicating with other MA's.

For MA C to migrate to site C, because there is only company table in site C, the inquiry task is:

select name from company

The result is saved into tempPK(C) and it is also the basis of communication with other MA.

4. After each Mobile Agent reaches its target site, executes assigned inquiry task concurrently, and communicates with each other during the process of data inquiry to finish the same data acquiring task.

The concrete steps are as following:

1. MA A communicates with MA B and transmits the temporary table tempPK(A) used to store foreign key (goods name) to MA B.

2. After the site receives tempPK(A) from site A, perform a join operation on tempPK(A) table and tempPK(B) table , and save the result into tempPK(B).

3. Mobile Agent B communicates with MA C and transmits the temporary table tempPK(B) used as store foreign key to MA C.

4. After MA C receives tempPK (B), perform a join opereation with tempPK(C) , and save the result into tempPK(C).

5. The site C sends tempPK(C) to site A and B.

6. The following inquiry task is executed synchronized in site A, B and C.

The inquiry task executed in site A is expressed as the following pseudocode:

```
for(i=0; i< number of records in tempPK(A); i++)
{
/*select * from project, company where project.name =company.name  and project.name= name in tempPK[i]  and project.cost>50000 or project.schedule <50. */
 }
```

The inquiry task executed in site B is expressed as the following pseudocode:

```
for (i=0; i< number of records in tempPK(B); i++)
 {
/*select * from project where project.name = name in tempPK[i]  and project.cost> 50000 or project.schedule < 50 */
 }
```

The inquiry task executed in site C is expressed as the following pseudocode:

```
for (i =0; i< number of records in tempPK(C); i++)
{
/*select * from company where company.name = name in tempPK[i] and project.cost> 50000 or project.schedule < 50 */
}
```

7. After three sites finish executing the inquiry task, they separately transmit their result set to site

D.

During this process of executing the inquiry, communication between MA is needed. As aforementioned steps one. The communication flow between MA A and MA B is showed as follows:

1. MA A creates KQML performative through the message proxy and send connection request:

ask-one:

: sender (ID of MA A)

: content(connect(?apply_connect?busy))

: receiver (ID of MA B)

: reply-with is Connect

It means that service equipment that hosting MA B is the receiving server, MA B is message receiver, and the content of the message is "connect" to request a connection. The valid responses are "agree to connect (apply_connect)" or "can not connect (busy)" and the identifier of the response message should be "isConnct".

2. If connection is agreed, the message proxy B should send a response KQML performative to MA B as following:

ask-one:

: sender (ID of MA B)

: content(apply_connect)

: receiver (ID of MA A)

: reply-with requestConnect

Its meaning is that MA B agrees to connect.

3. After MA A receives the response message "apply_connect", it sends the following performative through the proxy, and begins to transmit the result set.

ask-one:

: sender (ID of MA A)

: content(beginSend)

: receiver (ID of MA B)

: reply-with beginSend

4. After the result set is finished transmitting, MA A sends disconnect message though KQML.

ask-one:

: sender (ID of MA A)

: content(disconnect)

: Receiver (ID of MA B)

: reply-with disconnect

5. In the aforementioned step 4, the running status of the security module in this system is involved.

First, the visiting MA must show the encryption key. After passing the verification, grant its

authorization according to API access certificate it carries. In afore-mentioned interface, this visiting MA should have the authorization to read file, read and write database.

6. In the afore-mentioned step 4, after three sites finish executing their inquiry task, they separately submit their results sets to site H. The site H submits the integrated result to the customer. The result that the customer obtains is showed in Fig 29

Project name	current cost	Finish time	Schedule	company
traffic control system	70000	2006/08/07	60	Amy Liu, Peter Yao, Terry, Luc
Project management system	10000	2007/07/01	30	Alice Liu, Harry Pan, Ramdas Kumaresan
RFID entrance management system	65000	2006/12/20	70	Sanjit K.Mitra, George Symos, Ivan Selesnick

Homepage The first page Next Previous The last page

Fig 29 The chart of the inquire result

7. This inquiry is complete; the system releases communication channels and Mobile Agent etc.

Here is database class view:

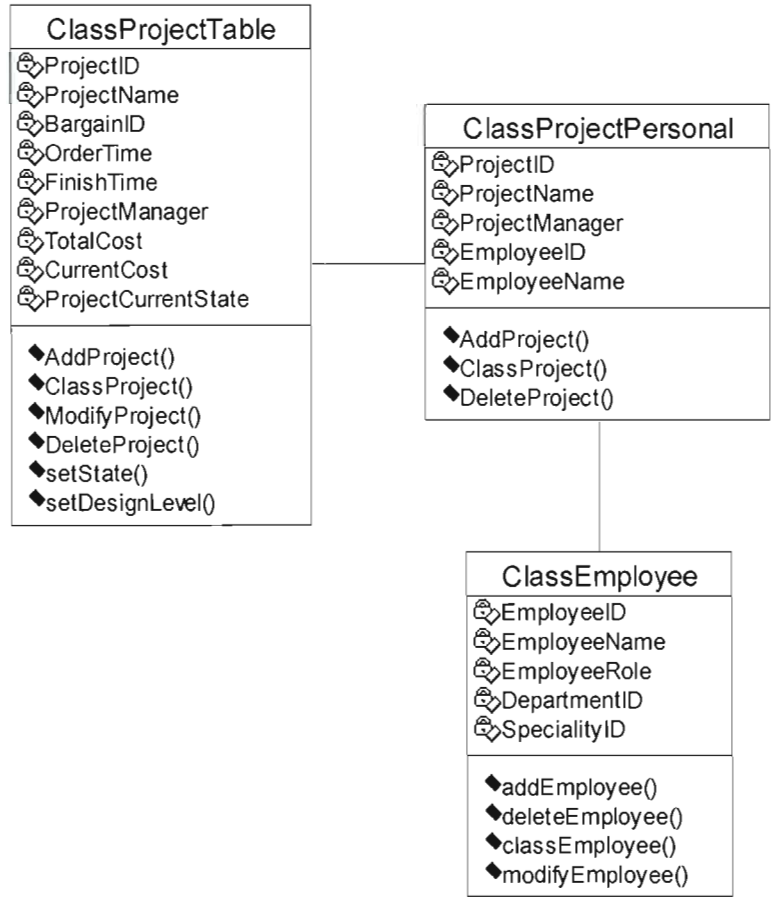


Fig 30 database class view

8.4 Summary

In this chapter, the application background of this model is provided first. Next, through a series of pictures, we analyzed and described the configuration method of this system. Finally, we demonstrated work status and flow of this system in practical application in examples.

The characteristics of this system and some comparison data will be provided in the next chapter.

CHAPTER 9

CONCLUSION

9.1 The characteristics of this system

This system has integrated Mobile Agent's migration, communication, life cycle and other functional modules, and established a distributed data service model. In summary, it has the following characteristics:

Because this design utilizes system architecture of Mobile Agent and Mobile Agent service equipments, we subdivide each functional module of MA distributed data query module according to their functionality strictly, and merge Mobile Agent transparency technology into each functional module to facilitate easy system expand and maintenance.

In this system, utilizing MA to carry parsed subtasks, the inquiry is executed in parallel at multiple sites and the result is returned in parallel. In the distributed process, there is great flexibility, and it becomes possible to operate cross hetero-platform cooperated. It utilizes the characteristics and advantages of Mobile Agent in distributed process method and improves the efficiency of cross platform operations.

This system implemented the combination of inquiring execution with inquiring business, and abstracts some low level communication protocol and operation interface in each phase of MA's life cycle. For the secondary developers, it implemented transparency MA implementation detail and

network operation. This enables them to use existing interface and take the advantages at the same time, thus shorten the cycle for secondary development.

9.2 Comparison with traditional method in theory

The model uses Mobile Agent technology for distributed inquiring, compared with the traditional method, it has the following advantages:

1. During the process of executing distributed computing, Mobile Agent only needs to keep the connection of two correlative distributed process sites when sending MA and returning result. It is unnecessary to keep an active connection in networks all the time. This method reduces the cost of maintaining network connections and decreases network's load comparing with the traditional one.

2. The traditional way to implement distributed processing is to add an abstract layer in order to hide the platform differences. In this system, because Mobile Agent service equipment and Mobile Agent are built on top of uniform rules and ACL language —KQML. It achieves consistence on semantics level. It is unnecessary to implement an extra abstract layer for additional hardware and software cost.

3. In the traditional distributed system, the interaction between sites has lag. And during the interaction, the communication channel must be maintained at some cost, and canceled after finishing the interaction. In this model, because of using Mobile Agent's behavior method, interacting task drives instance interacting activities. The interaction is executed by Mobile Agent service equipment through sending Mobile Agents to take the advantage of distributed computing the efficiency provided by

Mobile Agent technology. This will reduce interaction communication cost, and improve communication efficiency.

9.3 Theoretic Comparison with the existed model based on Mobile Agent

This model has improved the function modules of the existed distributed data service model based on Mobile Agent. Compared with existed system, this module has following characteristics:

1. Because message proxy and message store and transmit echanisms are used in this module, we reduced transmission failure rate caused by uncertain location of Mobile Agent, and improved reliability of sending and receiving message.

2. In this model, security mechanisms are designed for Mobile Agent and Mobile Agent service equipment. This increased difficulty for stealing and changing message, and improved system security.

3. In the database access layer, proxy mode is used between modules of service equipment and external network interface, and modules of service equipment and data request interface. It achieves low coupling between modules and improves its reusability and portability..

9.4 Experiment results and data comparison。

I have made comparison between this system and traditional method to execute distributed data service module.

The traditional distributed data service system is based on J2EE. The site and database configuration are the same with this module. Table 8 shows the corresponding experiment data.

Table 8 The traditional distributed system test data

Number of sites	Time of return result
2	16.0s
3	18.3s
4	22.1s
5	27.3s

Meanwhile, we compared this system with current system based on the Mobile Agent, the test data is shows in Table 9.

Table 9 Current Agent system test data

Number of sites	Time of return result
2	16.6s
3	17.9s
4	21.2s
5	26.1s

The corresponding test data of this system is shown in Table 10.

Table 10 This system test data

Number of sites	Time of return result
2	16.9s
3	18.0s
4	19.5s
5	24.3s

From the above comparison dada, we can draw the following conclusions:

1. In case of smaller number of sites, the model using the traditional method to execute distributed data service has advantages because additional cost is needed to maintain the MA's operation in the system based on the Mobile Agent to maintain the MA's operation.

2. When there is larger number of sites, the system based on the Mobile Agent has advantages. This advantage becomes significant as number of sites increases, which is determined by the inherent advantage of the Mobile Agent and the method of executing distributed computing.

3. In this model, because some appropriate improvement has been done on communication, lifecycle and other modules of Mobile Agent, the efficiency is improved. This advantage becomes significant as number of sites increases as well.

9.5 The expectations about this system

During the process of executing distributed computing, this system takes full advantage of the migration, independence and low communicating cost of Mobile Agent. The efficiency has been increased comparing with the traditional model and other similar model. This system can be further improved from the following aspects:

1. Because of the intelligence and sociality of Mobile Agent, we can ally the Mobile Agents that performing the similar inquiring task together, and share information. This will further strengthen the asynchronous cooperation of distributed execution processing and enhance the system's efficiency.

2. During the process of communication, communication flow can be further improved by using JKQML language. Using to the characteristic of distributed inquiring system based on the Mobile Agent,

a uniformed grammar and format of communication language can be defined. This will implement a common communication interaction language for distributed circumstance.

3. This system can be improved to become a general purposed distributed data service model by integrating functions of inquiring, increase, delete and update in one place.

4. Add more functions into SQL parse module to support more complicated SQL with the characteristic words such as Order, Group by, Sum, etc.

In this paper, we implemented and improved MA modules based on the current research so that they can be used in distributed circumstances.

Then, we merged the MA technology into the distributed data service model and build this model. We also analyzed each module and their work flows.

Finally, we presented some examples to demonstrate how the model will run in real world applications, and analyzed the characteristic of this model and made a comparison with other relevant software systems.

ANNEXES

```
packageHQGL.Aglet.Query;
import com.ibm.aglet.*;
import com.ibm.aglet.event.*;
import java.io.objectInput;
import java.io.objectOutput;
import java.io.IOException;
import java.net.URL;
import java.net.*;
import java.sql.*;
import java.util.*;
public class dbQueryAglet extends Aglet{
   private static final String SlaveClassName=
      "HQGLAglet.Query.patterns.dbQueryAgletSlave";
public Boolean handleMessage(Messagemsg){
try{
if(msg.sameKind("result")){
dbQueryAgletInfo arg=null;
_msw.appendResult(arg.toTextBlock());
setTheMessage("Finished!");}
else{
setTheMessage("Failed!");
}else{
super.handleMessage(msg);
return true;
}
}catch(Exception e){
System.out.println(e);
}
Return false;
}

protected void createslave(Vector destinations,object obj){
Arguments args=new Arguments();
args.setArg("msg",(String)obj);
try{
```

```
Slave.create(null , SlaveClassName,getAgletContext(),this,destinations,args);
}catch(IOExceptionae){
inError(ae.getMessage());
}
}

class QueryResult implements Serializable{
public String projectID;
public String projectName;
}
public class dbQueryAgletSlave extends Aglet{
                string querystring;
protected void initializeJob(){
Result=null;
}
protected void dbQueryJob() throwsAgletExecption{
try{
Argumentsb args=(Arguments)Argument;
querystring=newString((String)(args.getArg("msg")));
}cateh(Exception e){
e.printStackTrace();
}
Result=getDBInfo();
}
private object getDBInfo() throws AgletException{
Vector itsResult=new Vector();
AgletContextac=getAgletContext();
If(ac.getHostingURL()!=null){
Try{
Class.forName("sun.jdbc.odbc.JdbcodbcDriver , , );
}eatehOava.lang.ClassNotFoundExecptione){
System.err.Print("ClassNotFoundExecption:"):
System.err.Println(e.getMessage()):
}
try{
Connection conn=DriverManager.getConnection
("jdbc:odbcoraleJXDB_bzsx;uid=**;pwd=**");
```

```
Statement stmt=conn.createStatement();
Resultset rset=stmt.executeQuery(querystring);
while(rset.next()){
QueryResult result=new QueryResult();
result.projectID=rset.getstring(1);
result.projectName=rset.getstring(2);
itsResult.addElement(result);
}
conn.close();
}catch(SQLException ex){
System.err.Println("SQLException:"+ex.getMessage());
}
}
dbQueryAgletInfo info=new dbQueryAgletInfo(itsResult):
return info;
}
}

PaekageHQGLAglet.Insert;
import com.ibm.aglet.*;
import com.ibm.aglet.event.*;
import java.io.objectInPut;
import java.io.objectOutPut;
import java.io.IOExecption;
import java.net.URL;
import java.net.*;
import java.sql.*;
import java.util.*;
public classInsertMaster extends Aglet{
transientAgletProxyremoteProxy=null;
string tagart;
string insertsql;
public void onCreation(Object init){
tagart=getDestination():
public void run(){
try{
AgletContext context=getAgletContext();
remoteProxy=context.createAglet;
(null,"HQGLAglet.addprojectAglet",getProxy());
```

```
URL url=new URL(tagart);
remoteProxy=proxy.dispatch(url);
}catch(InvalidAgletExecption ex){
ex.printstackTrace();
}catch(Execption ex){
ex.printstackTrace();
}
if(remoteProxy!=null){
remoteProxy.SendMessage(new message("insert",insertsql))
}
}
public boolean handleMessage(Message msg){
if(msg.sameKind("result"){
System.out.println((String)msg.getarg());
return true;
}
return false;
}/
public void onDisposing(){
System.out.println("Disposing");
}
}end InsertMaster

package HQGLAglet.Insert;
import com.ibm.aglet.*;
import com.ibm.aglet.event.*;
import java.io.objectInput;
import java.io.objectOutput;
import java.io.IOException;
import java.net.URL;
import java.net.*;
import java.sql.*;
import java.util.*;
public class addprojectAglet extends Aglet{
string InsertString;
AgletProxy masterProxy=null;
public void onCreation(Object init){
masterProxy=(AgletProxy)init;
try{
```

```
DatabaseMetaData dma;
Class.forName("sun.jdbe.odbc.JdbcOdbcDriver");
}catch (java.lang.ClassNotFoundExceptione){
System.err.print("ClassNotFoundException:");
System.err.println(e.getMessage());
}
try{
Connection con=DriverManager.getConneetion
("jdbc:odbcoraleJXDB_bzsx;uid=**:pwd=**");
dma=con.getMetaData();
System.out.println("Connected to"+dma.getUPR());
System.out.println("Drive"+dma.getDriverName());
System.out.println("Version"+dma.getDriverVersion());
System.out.PriniIn("");
ResultSet rset=stmt.executeQuery(Insertstring);
}catch(SQLException ex){
System.err.println("SQLException:"+ex.getMessage());
}
}public boolean handleMessage(Message msg){
if(msg.sameKind("insert"){
Insertstring=(String)msg.getArg();
try{
PreparedStatement pstm;
Statement stmt=Con.createStatement();
Pstm=con.prepareStatement(Insert String);
Pstm.executeUpdate();
Pstm.close();
}catch(SQLException ex){
System.err.println("SQLException:"+ex.getMessage());
}
con.close();
masterProxy.sendMessage(new Message("result","finished");
return true;
}
con.close();
masterProxy.sendMessage(new Message("result", "finished"));
return false;
}
}
```

```
package HQGLAglet.Insert;
import com.ibm.aglet.*;
import com.ibm.aglet.event.*;
import java.io.ObjectInput;
import java.io.ObjectOutPut;
import java.io.IOException;
import java.net.URL;
import java.net.*;
import java.sql.*;
import java.util.*;
public classCooperationAglet extends Aglet{
string sqlstring=null;
message order;
string home=null;
SimpleItinerary itinerary=null;
public void onCreation(Object init){
sqlstring=getsql();
order=new Message("trip",sqlstring);
itinerary=new SimpleItinerary(this);
home=getAgletContext().getHostingURL().tostring();
}
public boolean handleMessage(Message msg){
if(msg.sameKind("trip")){
trip(msg);
}elseif(msg.sameKind("atHome")){
atHome(msg);
}elseif(msg.sameKind("doQuery")){
doQuery(msg);
}else{
retum false;
}
return true;
}
public synchronized void trip(Message msg){
string destination=(string)msg.getArg();
try{
itinerary.go(destination,"doQuery");
}catch(Exception){
```

```
ex.printStack Trace();
}
}
public void doQuery(Message msg){
string projectID;
string projectName;
sqlstring=(String)msg.getArg();
try{
Connection conn=DriverManager.getConnection
("jdbc:odbcoraleJXDB_bzsx;uid=**;pwd=**");
Statement stmt=conn.createStatement();
ResultSet rset=stmt.executeQuery(sqlString);
QueryResult result=new QueryResult();
result.projectID=rset.getstring(1):
result.projectName=rset.getstring(2);
itsResult.addElement(result);
}
conn.close();
}catch(SQLException ex){
System.err.println("SQLException:"+ex.getMessage());
}

itinerary.go(home , "atHome");

}
public void atHome(Message msg){
System.out.println((string)itsResult.projectID);
System.out.println((string)itsResult.projectName);
Dispose();
}
}
```

REFERENCE

[1]Yunyong Zhang,Jinde Liu. Mobile Agent technology. Beijing:Tsinghua University Press, 2003.

[2]Wei Zhang, Dan MA, Benli Wang. Mobile Agent system, communication and standard. 2002. 4: 136-139.

[3]Yongbo Wang , Changle Zhou. Mobile Agent research summarize. Application Research of Computers.2000, 6: 9-11.

[4]Zhenghu Luo , Jingan Yang, Xiangfeng Luo etc. MA based distributed computing model research. Mini-Micro Systems. 2000, 3: 300-304.

[5]Bole Shi, Baokang Ding, Aoying Zhou etc. Database system. Shanghai: Higher Education Press , 1999.

[6] DavidWong, NoemiPaciorek, DanaMoore. Java-based Mobile Agents. Communications of the ACM, 1999,3: 92-102.

[7]Yan Xiong, Fuyou Miao, Pei Hua etc. Mobile computing database access technology base on MA. Mini-Micro Systems. 2002, 10(23): 1165-1168.

[8] AgletsWorkbench by IBM Research GroupWebsite: http://www.trl.ibm.com/aglets/

[9]De Zhang , Yisheng Dong. The optimization of database association query on Internet. Chinese Journal of Computers. 2000. 2(23): 171-176.

[10]Yijie Wang, Yongjun Wang, Xicheng Lu. Distributed inquire algorithmic research. Journal of Software. 2001, 2(12): 219-224.

[11] Murphy A, Picco GP. Reliable communication for highly mobile Agents.Proceedings of the Agent Systems and Architecture/Mobile Agents (ASA/MA). 1999(99):141-150.

[12]Hong Wang, Guangzhou Zeng, Shouxun Lin. A location transparent communication implement of Mobile Agent system. Chinese Journal of Computers. 2001, 4(24): 442-446:

[13]Juan Yang, Jianguo Li. Message transmit mechanism of location transparent MA. Journal of Computer Applications. 2004, 3(24): 25-30.

[14]Xinpeng Tao, Xiuyun Feng, Xiang Li, Guoqiang Zhang. Communication mechanism in Mogent system.Journal of Software. 2000,11(8): 1060-1065 (in Chinese with English abstract).

[15]Belle WV, Verelst K, D'Hondt T. Location transparent routing in mobile Agentsystems merging name lookups with routing. Proceedings of the 7th IEEE Workshop onFuture Trends of Distributed Computing Systems. 1999. 207-212.

[16]Dajun Cao, Liangxian Xu, Xun Wang. Mobile Agent. Computer Engineering and Applications.2002,

01:164-166.
[17]Jianguo Ding, Huilin Liu, Hansheng Chen etc. Security authenticates mechanism of Mobile Agent. Computer Engineering. 2001 , 2(27): 73-74.
[18] HeYan-xiang, LiXu-huietc. The security in mobile agents. Computer Applications, 2001, 131(7): 5-8(inChinese).
[19] Wong D,Paciorek N, Moore. Java-based Mobile Agent Communication of ACM.1999, 3(42): 92-102.
[20] Schelderup K,Olnes J. Mobile Agent Security Issues and Directions. Intelligence in Service and Network.1999(1579):155-167.
[21] Martin A J. A message passing model for highly concurrent computation. Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications. Pasadena, CA, USA, VoL 2, 1998.
[22] Dennis Tsichritzis, Alexander Herrige, Ciaran Bryce. ASAP: Agent System Architectures and Platforms. Jan Vitek University of Geneva conference INFORMATIK/INFORMATIQUE , 5/ 1999.
[23] Adelinde M Uhrmacher,Petra Tyschler,Dirk Tyschler. Modeling and Simulation of Mobile Agents. Future Generation Computer Systems. 2000(17):107-118.
[24] http://www.trl.ibm.com/aglets/ The IBM's aglets home page.
[25]Ping Wang, Guoren Wang etc. Mobile Agent migration mechanism research and implement. Journal of Northeastern University(Natural Science).2001, 12(22): 600-603.
[26]Jinde Liu, Yunyong Zhang. An applied MA(aglet) summarize. Computer Applications. 2001, 8(21): 1-3.
[27]Nieye Jie, Dating Liu, Huiling Ma. Agent's system structure. Application Research of Computer. 2001.9: 52-55.
[28]Qianxia Xu, Distributed database access base on MA. Journal of GUILIN institute of electronic technology. 2002, 1(22): 76-80.
[29]Guorong Wu, Qinghao Shen. Constructoion of Mobile Agent-based Secure Electronic Auction System. Computer Engineering. 2003, 7(29): 113-115.
[30]Bo Liu, Taoshen Li. The Research of Agent-based Distributed Database Management. Application Research of Computer. 2002, 5: 36-37(-63).
[31]Yanxiang He, Xuhui Li, Zhuomin Du etc. An Agent-based Software Service Framework Model. Mini-Micro Systems. 2003 7(29): 1178-1182.
[32]Man Hao, Yang Cao. Analysis of the Mobile Agents' Security. Data Communications. 2001 No.4 P.19-21,26.
[33]Junfeng Gu, Liang Zhu, Fanyuan Ma etc. Mobile Agent system. Mini-Micro systems. 2001, 7(22): 860-863.