Accessing Database Information Using Mobile Agents

Zili Zhang

Department of Computer Science, Southwest China Normal University, Chongqing 400715, China
School of Information Technology, Deakin University, Geelong Victoria 3217, Australia

Abstract: Many real-world applications on the Internet require accessing database information and typical technology employed is client/server plus Web (C/S + Web). Although past few years saw many success applications by using this technique, there are still some drawbacks that need to be overcome. One of the drawbacks is that the transaction often fails if the network connection is unstable. Another disadvantage is high bandwidth requirement and latency. This paper argues that mobile agent technology provide an easy way to overcome the shortcomings in C/S + Web in database access on the Internet. A success case study using mobile agents to admit new students to China's institutions is then presented. The mobile agent was created by using IBM's Aglets Software Development Kit (SDK). Based on the experimental results, it is evident that mobile agent technology is well suited for such applications.

Key words: Software agents, mobile agents, client/server, database access and information retrieval

Introduction

The typical scenario in many real-world applications on the Internet (especially in e-commerce field) is as follows: Users send their requests to remote servers through Web browser. Once the server receives the request, it will access the related background databases to verify the identification of the user, retrieve the required information and so on. Finally, the remote server will send back the results to the users. No matter what kind of services provided by the servers, one crucial operation involved in this process is accessing the databases. The widely adopted technique is C/S + Web. With this technology, many success applications have been developed. However, there are some drawbacks when using this technique to access databases on the Internet.

In C/S paradigm, the transaction often fails if the network connection is unstable. Once the network connection is disconnected, the whole transaction will be failed and has to be restarted from the beginning. The second disadvantage is excessive requirement of bandwidth and latency. In C/S paradigm, a transaction between the client and the server may require many round trips over the network and each trip will create network traffic and consumes bandwidth and latency. Limitation of server capacity is another issue in C/S + Web paradigm. When a massive amount of requests have been sent to the server simultaneously, the server cannot handle the workload and is likely to crash, resulting in the Website cannot be retrieved until system administrators fix the problems. For example, airline companies encourage passengers to book tickets online and try to promote this by offering special price in a limited number of air tickets that can only be booked online. People then rush to visit the Website to book the air tickets within the same period of time because of seat limitation. The server is very likely crashed as there is a very high workload in a very short time period. Is there any technique that is easy to eliminate those disadvantages?

Mobile agent technique is a very promising candidate. An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives (Wooldridge, 1997). It possesses the following minimal characteristics: autonomy, social ability, reactivity and proactivity. Mobility is an orthogonal property of agents. A mobile agent is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another. The ability to travel allows a mobile agent to move to a system that contains an object with which the agent wants to interact and then to take advantage of being in the same host or network as the object (Lange and Oshima, 1998). Mobile agents can work very well under unreliable network connection. By migrating to the servers, mobile agents can request services or even process data locally on the server side. This reduces the network load.

Thus far, there are some successful applications of mobile agents in database accessing. Papastavrou, Samaras and Pitoura (Papastavrou et al., 2000) proposed a new framework called “DBMS-Aglet Framework” for Web-based distributed access to database systems based on Java-based mobile agents. The idea was to use DBMS-applet for creating mobile agent that travels directly to the remote SQL server. Once the mobile agent arrives at the SQL server, it initiates a local JDBC driver, connects to the database.
Zili Zhang: Accessing Database Information Using Mobile Agents

and performs any queries specified by the sending client. When the mobile agent finishes its task, it dispatches itself back to the DBMS-applet in the client machine. These mobile agents were not equipped with capability to further process the retrieved data on the server side.

Brewington, Gray and Moizumi et al. developed a distributed information-retrieval application called “technical-report searcher” (Brewington et al., 1999). The application uses mobile agent technology to retrieval information from different hosts across the network. The idea is to let the mobile agent decide whether to spawn child or only interact with the individual document collections across the network. If the mobile agent decides to spawn child, the parent mobile agent can merge and filter the results from individual child mobile agent once the child send back the results to the parent.

The emphasis of this paper is to find out in which situation mobile agents should be used for database access on the Internet by conducting a series of experiments. Based on this, a real application is provided to indicate what kind of application is suitable for mobile agents and demonstrate how to develop mobile agents using IBM’s Aglets Software Development Kit (SDK) (Lange and Oshima, 1998). The remainder of this paper is structured as follows. Section 2 describes the database access models using mobile agents for accessing databases on the Internet. Section 3 highlights the advantages of mobile agents in database access on the Internet through different experiments. A case study that mobile agents are used to admit new students to China’s institutions is provided in Section 4. Finally, Section 5 is the concluding remarks.

Database Access Models Using Mobile Agents: In order to reveal the advantages of mobile agents in database access on the Internet, two mobile agents were developed. One mobile agent can only retrieve the information from the database and bring back all the retrieved data to the user. The other can not only retrieve the information but also process the retrieved data locally at the server side and only bring back a small portion of retrieved data to the user. The first case is called mobile agent without further processing capability, the latter called mobile agent with further processing capability.

When using mobile agents to access database information on the Internet, the following five components are needed:

* Database Server: The database server is where the databases located. The MySQL database server has been chosen for experiments in the next section.

* Mobile Agent Server: The mobile agent server is the platform where mobile agents dispatch themselves to other hosts and where mobile agents land in the remote hosts. Without it, mobile agents cannot dispatch or land at any host. Tahiti mobile agent server will be used in the experiments, which is part of the Aglet SDK (Lange and Oshima, 1998).

* JDBC Driver: In order to provide database access for mobile agents, Java Database Connectivity (JDBC) is needed. It acts as a bridge that binds mobile agents and database together. As MySQL was selected as the database server, JDBC driver for the MySQL database server has to be used. Thus, MySQL Connector/J is selected.

* User Interface: The user interface is used to collect information from users, manage and send mobile agents to different hosts. In the experiments, a Java applet is built as such an interface.

* Mobile Agent: IBM’s Aglet Software Development Kit (Aglet SDK) was selected as the mobile agent development platform. Aglet is a Java-based mobile autonomous agent that is developed by using Aglets SDK. Aglet can dispatch itself, travel itself to another host on the network and resume execution at the destination host.

The models used in the two cases have some difference. For the first case (mobile agents without further processing capability), the model is shown in Fig. 1. In this model, when the client needs to retrieve information from the database at the remote server, the client needs to use the user interface (Java applet) to create a mobile agent at the local mobile agent server. After that, the user interface applet will send the mobile agent to the remote server. Once the mobile agent arrives at the remote server, it will initialize a local JDBC driver and use it to connect the database and then performs any queries. When the mobile agent finishes its task, it brings back the results and displays them through the user interface applet. There are a few advantages using this model. Firstly, the mobile agent does not rely on network connection since mobile agents can perform normally even the network connection is unstable. Secondly, the client does not need to download the JDBC driver in advance as the mobile agent can use the remote JDBC driver to connect to the database. The database access model using mobile agents with further processing capability is shown in Fig. 2. In this case, mobile agents have the capability to process the query results on the server side. Once the mobile agent obtains the query results, it will process them locally at the server based on user preference (e.g., filtering the results). After processing the query results, it usually brings back fewer amounts of data back to the user interface applet.

Advantages of Mobile Agents: Based on the models
proposed in the previous section, a series of experiments were conducted to verify that mobile agents do overcome shortcomings in C/S + Web situation.

**Mobile Agents under Unreliable Network Connection:**

The first experiment conducted is to test the reliability of mobile agents under unstable network connection. In this experiment, a mobile agent is dispatched to a remote host. During this process, the network connection was disconnected manually. After a few minutes, the network connection was restored. It is noticed that once the connection is alive, the mobile agent can reach its destination even though there is a little delay. While in typical C/S situation, a “time out” error was reported and the whole transaction has to be restarted from the beginning when the network connection was restored. It is evident that mobile agents are more reliable under unstable network connection.

**Mobile Agents With and Without Further Processing Capability:**

The performance of mobile agents with and without further processing capabilities was tested and compared with that of C/S situation by setting up the following experimental environment: A special Java applet that can access database information on the remote server was developed. This is the typical C/S application. An experimental database was also designed and $10,000$ records were populated. First, both the mobile agent and the applet have the SQL statement "select * " that can retrieve all records in the database. That is both of them bring back 10,000 records to the users. The test was repeated ten times and the results are shown in Table 1.

**Table 1: Execution time (in second) of Applet and Mobile Agent without Further Processing**

<table>
<thead>
<tr>
<th>Test No.</th>
<th>Mobile Agent</th>
<th>Applet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20.43</td>
<td>3.68</td>
</tr>
<tr>
<td>2</td>
<td>18.95</td>
<td>3.46</td>
</tr>
<tr>
<td>3</td>
<td>19.94</td>
<td>3.74</td>
</tr>
<tr>
<td>4</td>
<td>18.84</td>
<td>3.41</td>
</tr>
<tr>
<td>5</td>
<td>19.50</td>
<td>5.00</td>
</tr>
<tr>
<td>6</td>
<td>25.54</td>
<td>3.41</td>
</tr>
<tr>
<td>7</td>
<td>19.82</td>
<td>3.35</td>
</tr>
<tr>
<td>8</td>
<td>18.94</td>
<td>3.35</td>
</tr>
<tr>
<td>9</td>
<td>19.17</td>
<td>3.35</td>
</tr>
<tr>
<td>10</td>
<td>18.56</td>
<td>3.40</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>19.969</strong></td>
<td><strong>3.615</strong></td>
</tr>
</tbody>
</table>

In this case, as there is a migration time for the mobile agent, the applet is much faster than the mobile agent to retrieve and bring back the same amount of data. The mobile agent was then equipped with a further processing capability, which can filter out 8,000 records and only bring back 2,000 records to the user. The applet still brings back 10,000 records back. The testing results are shown in Table 2. In this case, the mobile agent performed better than the applet. It is because the migration overhead can be covered by the time gained in data transfer.

**Table 2: Execution time (in second) of Applet and Mobile Agent with Further Processing**

<table>
<thead>
<tr>
<th>Test No.</th>
<th>Mobile Agent</th>
<th>Applet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17.30</td>
<td>17.30</td>
</tr>
<tr>
<td>2</td>
<td>16.86</td>
<td>29.77</td>
</tr>
<tr>
<td>3</td>
<td>16.92</td>
<td>29.88</td>
</tr>
<tr>
<td>4</td>
<td>16.10</td>
<td>30.82</td>
</tr>
<tr>
<td>5</td>
<td>16.76</td>
<td>30.93</td>
</tr>
<tr>
<td>6</td>
<td>16.81</td>
<td>30.10</td>
</tr>
<tr>
<td>7</td>
<td>18.68</td>
<td>30.10</td>
</tr>
<tr>
<td>8</td>
<td>16.42</td>
<td>30.82</td>
</tr>
<tr>
<td>9</td>
<td>16.20</td>
<td>37.51</td>
</tr>
<tr>
<td>10</td>
<td>17.80</td>
<td>30.21</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>16.985</strong></td>
<td><strong>31.238</strong></td>
</tr>
</tbody>
</table>

The experiment results indicate that the mobile agents with further processing capability should be used if possible.

**Solution to the Overloading Issue:**

As mentioned, in the C/S + Web the server may be overloaded if there are too many users try to connect to the server at the same time. By using mobile agent technology, the server is no longer receiving large numbers of requests. Instead, mobile agents will carry the task and dispatch to the host of the Website to perform the task locally. Nevertheless, there is still a possibility that many mobile agents will arrive at the server simultaneously. Solutions can be easily found for this issue when using mobile agents. One of the solutions we tested is to setup a queue in the mobile agent server in the server side. When mobile agents arrive at the server, they will be put in a queue in the mobile agent server and the mobile agent server will use the first come first serve method to allow mobile agents access resources at the server. The size of the queue is set in advance to prevent too many mobile agents dispatching to the server at the same time. If too many mobile agents have been sent to the same host at the same time and the queue is full, the mobile agent server will reject any late coming mobile agents until the queue is not full again. By using the queue method in mobile agent technology, the server will never be overloaded because the mobile agent server is able to control the number of mobile agents accessing the server simultaneously.
Zili Zhang: Accessing Database Information Using Mobile Agents

Case Study—Admitting New Students to China’s Institutions: From the discussions of the previous two sections, it is clear that mobile agents can work reliably under unstable network environment. Moreover, if the mobile agents are equipped with further processing capability, their performance can be further improved. All these make mobile agents are quite suitable for the Internet admission for China’s institutions. In this section, the development of such mobile agents will be described. For the sake of further discussion, some terms and background knowledge about the application is briefly described first.

Background of Student Admission in China’s Institutions: The generic term ‘institutions’ is used to describe the various types of institutions. Terms such as university and college are used throughout China for educational institutions at all levels and in all sectors. The higher education in China usually consists of regular education sector, adult education sector and self-study examinations. ‘Institutions’ is used to refer to higher education in the regular sector. ‘Regular’ is the term used in this paper to describe those institutions which primarily admit students immediately after their completion of the full program of school education to full-time Bachelor degree or Diploma programs. Since 1978 all applicants for full-time programs at regular higher education institutions in China have been required to complete senior secondary education and to sit for the National College Entrance Examination (NCEE). In order to sit for the NCEE students must achieve a specified level of academic achievement in senior secondary school; satisfy certain political and health requirements. The examination is held over three consecutive days in early July (or early June). Students sit for either the science or humanities stream. Both have five subjects (the total score of each subject is 150). Students then are asked to fill in a “wish form” to rank their choices. Students can choose two to three institutions for each category from four categories such as key point institutions, non-key point institutions etc.

In all provinces, autonomous regions and municipalities, there are special admission offices (known as ZhaoSheng BanGongShi in China) that are responsible for collecting students’ information and organizing the admission. The information of each student gathered by the admission office includes the NCEE scores, medical examination results, ‘wish’ information, three year study records in senior secondary school, graduation examination scores at the end of senior secondary school and evaluation of moral, ideological and political progress (including self-evaluation, peer commentary and teacher assessment) etc. After the admission offices collect students’ information, they allocate students’ documents to different institutions according to students’ NCEE scores and ‘wish’ information etc. The representatives of institutions then check students’ documents and make the final decisions. Admitting new students to China’s institutions is a very important event in China. It is the focus of attention of hundreds of millions of Chinese people. The admission must be as just and fair as possible for all students and, at the same time, save money and time. The admission process has experienced three stages since 1978. At the first manual processing stage, there were many errors or mistakes in the collected students’ information and allocated students’ documents, needless to say the justice and fairness. With the introduction of computers, especially LAN (Local Area Networks) technologies at the second stage, the situation was improved a little bit. With the advent of Internet in China in 1994, people began to attempt to admit new students through the Internet. There are some problems in the currently adopted “C/S + Web” model due to the narrow bandwidth of the communication infrastructure in China. Introducing mobile agents in the admission process can reduce the network traffic load and maximize the just and fair degree of admission. Thus mobile agent technique is the most promising and practical approach for admitting new students to China’s institutions. In the following subsections, how to design such mobile agents is presented and corresponding experimental results are provided.

Admission Mobile Agent Model: The admission mobile agent was developed by using Java and Aglets SDK. Aglets provide the basic capabilities required for mobility. It is a simple framework where the programmer overrides predefined methods to add desired functionality. An Aglet is defined as a mobile Java object that visits Aglet-enabled hosts in a computer network. The complete Aglet object model includes some abstractions such as aglet, context, proxy, message, itinerary and identifier. For more information on Aglet, see http://www.tri.ibm.co.jp/aglets/ and (Lange and Oshima, 1998). The following discussions are based on Aglets SDK 1.1 Beta 1.

In the admission mobile agent model, the key abstractions of Aglet were used: aglet, proxy, context and identifier and one more module is added: Java rule base for admission (Fig. 3).

* Aglet An aglet is a mobile Java object that visits aglet-enabled hosts in a computer network. It is autonomous, because it runs in its own thread of execution after arriving at a host and reactive.

193
because it can respond to incoming messages.

* Proxy A proxy is a representative of an aglet. It serves as a shield that protects the aglet from direct access to its public methods. The proxy also provides location transparency for the aglet; that is, it can hide the aglet’s real location. This means that an aglet and its proxies can be separated so that a local proxy hides the remoteness of the aglet.

* Context A context is an aglet’s workplace. It closely corresponds to the place concept introduced in (Lange and Oshima, 1998). It is a stationary object that provides a means for maintaining and managing running aglets in a uniform execution environment where the host system is secured against malicious aglets.

* Identifier An identifier is bound to each aglet. This identifier is globally unique and immutable throughout the lifetime of the aglet.

* Java rule base There are many rules representing the admission criteria of a specific institution in the rule base. These criteria may be different for the students from different provinces. These rules are activated by the onArrival() method of the aglet. Some rule examples are as follows:

Rule 1: IF the student will study computer science, THEN the NCEE scores of mathematics and English subjects must be at least 100.

Rule 2: IF a student has strong point in sports, THEN the total NCEE score can be 10 points less (than the set standard in that province.)

Rule 3: IF a student has strong point in music or fine arts, THEN the total NCEE score can be 5 points less.

We will not discuss the implementation details of these rules. Interested readers refer to (Bigus and Bigus, 1998).

The admission mobile agents follow the same life-cycle model of the aglet shown in Fig. 4.

* Creation The creation of an aglet takes place in a context. The new aglet is assigned an identifier, inserted into the context and initialized. The aglet starts executing as soon as it has been successfully initialized.

* Cloning The cloning of an aglet produces an almost identical copy of the original aglet in the same context. The only differences are the assigned identifier and the fact that execution restarts in the new aglet. Note that execution threads are not cloned.

* Dispatching Dispatching an aglet from one context to another will remove it from its current context and insert it into the destination context, where it will restart execution (execution threads do not migrate). We say that the aglet has been “pushed” to its new context.

* Retraction The retraction of an aglet will pull (remove) it from its current context and insert it into the context from which the retraction was requested.

* Activation and deactivation The deactivation of an aglet is the ability to temporarily halt its execution and store its state in secondary storage. Activation of an aglet will restore it in the same context.

* Disposal The disposal of an aglet will halt its current execution and remove it from its current context. Actually, Retraction, Activation and deactivation were not used in this model.

A Scenario: In the admission situation, the admission office in each province assigns a unique number to each institution beforehand. When an aglet is created, it asks the user to input some information about the institution. The information includes the unique number assigned by the admission office, the official name of the institution, the category the institution belongs to, the stream (science or humanities) etc. This is achieved by overriding the onCreation() method. The aglet then asks the user to give the URL of a specific admission office and calls its dispatch method with this URL:

d i s p a t c h ( n e w URL("atp://specific_admission_office.edu.cn/context"));

ATP is the Agent Transfer Protocol, an HTTP-like protocol developed for mobile Internet agents. All these are accomplished in the run() method. After that, the aglet will disappear from the institution’s machine and will reappear in the same object state at the specified admission office’s server. First, a special technique called object serialization is used to preserve the state information of the aglet. It makes a sequential byte representation of the aglet. Next, it is passed to the underlying transfer layer, which brings the aglet (byte code and state information) safely over the network. Finally, the transferred bytes are deserialized to re-create the aglet’s state.

On arrival the site, the aglet sends its identity information (the unique number assigned by the admission office etc.) to the server to verify its identity. The server compares this information with that in its database. The server will send back an “accept” information to the aglet if the information is identical. Otherwise, the aglet is refused to run on the server. All these are done in the onArrival() method.

After the aglet is approved to execute on the server, it begins to retrieve the students’ documents allocated to the institution which the aglet is on behalf of. It makes “acceptance”, “rejection” or “suspension” decisions according to its admission rules. The aglet then comes back to the institution’s machine with the suspended student documents. Some persons at the institution
will process these suspended student documents and make "acceptance" or "rejection" decisions. Again, the aglet will be dispatched to the admission office’s server with these decisions.

Security Problems: The central problem of mobile agents is the security problem (Lange and Oshima, 1998; Pham and Karmouch, 1998; Brenner et al., 1998; Klusch, 1999 and Vigna, 1998). A user will give an agent important tasks or trust it with confidential data only when he is sure that the agent will treat his data confidentially and there are no possibilities that other persons or objects can affect the agent’s work, for example, by illegally accessing the agent’s confidential data. This is also the same situation in our application. The data that the aglet processes and transfers are very confidential. The most possible and also the most dangerous security attacks related to our admission mobile agents are as follows:

* **Illegal Access** Unauthorized reading, copying or modifying of student’s documents, especially the NCEE scores etc.
* **Masquerade** An aglet enters a server of a specific admission office by seemingly representing a trusted institution, i.e., adopting the identity of another institution.
* **Repudiation** An aglet admitted some new students or the server allocated some student documents later denies their taking place.

The private key method was used to encrypt the data to be transmitted. This can avoid the unauthorized reading and thus the associated loss of trust. A special check-sum created to avoid the modification of information can be used to detect a change to the
information content. We adopt private key method because of our relatively small problem scale—each institution keeps at most 31 keys and each admission office keeps less than five hundred keys. The key maintenance is not difficult. The keys are sent to different institutions with the unique number by using non-electronic communication paths. With a digital signature it is possible to verify the originator of an agent signed by the originator or to check the integrity of an agent signed by the sender. We plan to use digital signature technique in our mobile agent. Currently, we have not implemented it yet.

Implementation Discussion: In this subsection, some important implementation-related issues of the mobile agent are discussed for demonstration purpose, which include message handling, database connection and query execution.

![Mobile Agent Model for Admission](image)

**Message handling:** Mobile agent handles an incoming message object by invoking the `handleMessage` method. The implementation of this method consists of a switch statement that tests for the category of the incoming message. Corresponding action will be taken based on the category of the incoming message. It also returns a Boolean value to acknowledge that it has handled the message. The following is one example of message handling:

```java
public boolean handleMessage(Message msg) {
    if (msg.sameCategory("atHome")) {
        atHome(msg);
    } else if (msg.sameCategory("makeConnection")) {
        makeConnection(msg);
    } else {
        return false;
    }
    return true;
}
```

**Database Connection and Information Retrieval:** Connecting to a database and then retrieving information from the database is the main task of the mobile agent. To access the database, the mobile agent has to load the JDBC driver and uses it to connect the database. The following code shows how to load the driver:

```java
Class.forName("com.mysql.jdbc.Driver").newInstance();
After the JDBC driver is loaded, the location of the database server, the name of database, user name and password should be specified. A simple example of how to connect to a database is given below:
Class.forName(JDBCdriver).newInstance(); //load the driver
java.sql.Connection conn;
// specify the JDBC URL
String conn_string = "jdbc:mysql://localhost/test?user=blah&password=blah";
conn = DriverManager.getConnection(conn_string);
```

**Query Execution:** When the mobile agent connected to the database server, it can execute any query to retrieve information from the database. There are three different methods to execute a query, i.e., `executeQuery` method, `executeUpdate` method and `execute` method.

- `executeQuery` method is used to execute a SQL statement and the result is stored in a single `ResultSet`:
  
  ```java
  ResultSet rs = stmt.executeQuery("select * from student");
  ```

- `executeUpdate` method is used to execute SQL insert, update, or delete statement only. Moreover, this method will return a number of affected records.
  
  ```java
  RecNum = stmt.executeUpdate("delete from student where age > 30");
  ```

- `execute` method is used to execute a SQL statement that may return multiple results. This method can also execute any types of SQL statements.
  
  ```java
  Stmt.execute("create database student");
  ```

**Experimental Results:** The admission mobile agent is currently in its experimental stage. Some experiments were conducted using the data of past few years. Two admission offices—one in Sichuan province and one in Chongqing municipality and three institutions—Sichuan University (SCU), Chongqing University (CQU) and Southwest China Normal University (SWNU)—were involved. The experimental results are shown in Tables 3 to 6. From the tables one can see that the students' documents that need to be brought back to the institutions by the admission aglet account for less than 20% of the total documents. This means that the network load is reduced at least 80% of the originally load. Hence, the use of mobile agents in admission can save the travel expense as well as the communication costs (save up to 80% or more of the cost in the third stage). Using mobile agents can also maximize the just
and a fair degree to students in admission process. Therefore, mobile agent technique is the most promising and practical modern approach for admission of new students in China.

Comments:
- Planned—The planned admission student number of an institution in a province.
- Allocated—The allocated student document number to an institution. It is usually 120% of the planned number. In most cases, it is not exactly of that number. It may be less than that number if there are no enough students whose ‘wish’ are the specified institution and meet the criteria. It may also be greater than that number if there is more than one student with the same total NCEE score.
- Accepted—Accepted student number by the aglet.
- Rejected—Rejected student number by the aglet.
- Suspended—The student number that the aglet can not make decisions.
- Percentage—The suspended student number accounts for the total allocated student document number. This part of student documents must be brought back to the institution by the aglet.

Conclusions
Accessing database information is the key operation in
many real applications on the Internet. There are some drawbacks in the typical used technology C/S + Web. Using mobile agents to access database information on the Internet can overcome the disadvantages in C/S + Web. Using mobile agent in the admission of new students to China’s institutions can reduce the network load dramatically. At the meantime, the interference from people is reduced to a minimum in the admission process. In return, this maximized the just and fair degree of admission. This makes mobile agent is the most promising and suitable technique for admission new students to China’s institutions. Before the mobile agents go into practical operation, the related security services must be improved and enhanced.

The performance of mobile agents also relies on the further processing capability bestowed. In the admission situation, the accuracy of decisions made by the mobile agents depends heavily on the quality of the admission rules. This implies that the domain knowledge also plays an important role in the admission process. Thus, the admission rules need to be elaborated according to the situations in different provinces and different institutions.

References