YAVO
On-line Trading System using Mobile Agents

by

Yao Chen, B.Sc.
Shanghai University, 1995, China

Thesis
submitted in partial fulfillment of the requirements for
the Degree of Master of Science (Computer Science)

Acadia University
Spring, 2000

0-612-51991-0

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

Dedication

To my family and friends.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor Dr. T. Muldner for his guidance, inspiration and valuable time throughout this work. Thanks also are extended to Dr. Leslie Oliver for being my internal examiner and Dr. Carolyn Watters for being my external examiner.

I would also like to thank my family and my friends for their love, encouragement, and support.

ABSTRACT

This thesis describes YAVO, an on-line trading system using mobile agent technology. The YAVO system assists end-users selling and buying goods or services through mobile agents on the Internet. A system administrator creates trader accounts and distributed agent marketplaces through the Web. An authorized trader then sends an agent to an agent marketplace to buy or sell specific goods or services based on user-specified strategies. The YAVO system is modeled on Voyager, a full-featured agent-enhanced distributed computing environment. Java Servlets are used as a bridge between the Web client (a browser) and the back end servers (Java Web Server and Voyager Server). Persistent storage on the Java Web Server is provided through JDBC (Java Database Connectivity).

Chapter 1

1 Introduction

When renting an apartment or purchasing a used car, people research extensively via reading newspapers, watching television, or talking with friends. After gathering the information, a lot of time is spent on comparing the prices, analyzing the condition of products, negotiating with others, and making the final deal. How can people be spared from the waste of time and energy that arises from dealing with these time-consuming tasks?

The emergence of the Internet opens up a convenient door for purchasers. For example. the company Amazon.com provides an easy way to purchase books through the Internet. Although it is more convenient than traditional purchasing methods, electronic purchasing is still largely non-automated. A buyer still must search, gather, and analyze information on goods and sellers. The buyer must also compare products, make decisions, and enter payment information before making a purchase.

Mobile agent technologies can be used to automate time-consuming stages of the purchasing process. A mobile agent is a software program that may be transported to a remote machine and executed autonomously. The main benefits of mobile agents include reduced network load and latency. In other words, the bandwidth on the network can be used more efficiently through mobile agent technology.

Usually an electronic commerce (E-commerce) transaction involves multiple interactions between the buyer and seller. A mobile agent represents its creator and negotiates with other mobile agents on behalf of the creator. In that capacity, mobile agent technology is very suitable for E-commerce applications.

This thesis will introduce a "consumer-to-consumer" on-line trading system called YAVO, which is designed and implemented by the author of this thesis. An administrator creates agent marketplaces on different machines or at different ports (places) on the same machine. The administrator also creates trader accounts and a banking system. The trader then sends a seller or buyer agent to one of the marketplaces where the buyer agent finds, and negotiates with other seller agents. This is done to make the best deal on the trader's behalf. The trader can also send monitoring agents to gather pertinent information, without a purchase being made.

The characteristics of a good design for an E-commerce system are simplicity, reusability, security, and performance [HBS 99]. The YAVO system that I have designed, implemented, and tested is founded on those characteristics.

To achieve simplicity and reusability, the sizes of the business objects in YAVO are kept as small as possible. One object performs a single function or a small set of closely related functions. These objects are inherently small, simple, and easy to maintain and extend. The major objects of the YAVO system consist of an administrator, a trader, an agent marketplace, a mobile agent, and a bank. The ***administrator*** is responsible for creating and maintaining traders, agent marketplaces, and a banking system. The ***trader*** creates and sends mobile agents to marketplaces to sell or buy goods or services. ***Agent Marketplaces*** are located at different machines or ports, offer market information to agents and charge service fees to the agents. Buyer agents, seller agents, and monitoring agents are ***mobile agents*** that are sent to agent marketplaces to buy, sell, or monitor goods or services. Configured by traders, mobile agents reside in the marketplace for a certain number of days and change the product price at a certain speed. The ***banking system*** provides financial transaction support to the YAVO system. It issues electronic cash (E-cash) to the traders, and offers withdraw and deposit functions to traders and agent marketplaces. A trader gives a certain amount of E-cash to the agent to purchase a product and pay for service charges.

Security is an important issue for a mobile agent system because mobile agents travel through the network with confidential information such as electronic cash and trader information. Mobile agents need to be guarded against malicious machines, and hosts have to be protected from malicious agents. One of the ways to protect mobile agents is to send them to marketplaces on trusted hosts only. In YAVO, all the agent marketplaces are located only on trusted machines. To protect hosts a security manager is used to authenticate and authorize each mobile agent. The agent can be granted permission to read and/or write files, or can be granted no file access at all. Security is very critical in E-commerce applications. Users accessing a Web site are assigned roles appropriate to their needs. For instance, the traders and the administrator in YAVO have different roles and are therefore granted different access rights to the Web site.

To improve performance, a docking station is introduced in YAVO. A docking station is permanently installed in association with the network and functions as an agent-buffer that stores and forwards an agent. A docking station is used when unreliable networks fail to send an agent to the destination agent marketplace. Upon network failure, mobile agents will move to a docking station and wait in a line. Once the docking station connects to the recovered network, which then connects to the destination agent marketplace, the agent will be sent out to its destination.

To implement YAVO, a mobile agent platform had to be chosen. I chose ObjectSpace's Voyager. Voyager is an agent-enhanced full-featured distributed computing platform. It supports mobile objects and autonomous agents. It also includes services for persistence, group communication, and basic directory service, etc. This computing platform "raises productivity and shrinks the time to solution by combining the power of Voyager's industrial strength performance with the services needed to design and deploy... distributed enterprise applications more easily..." [GG 99].

YAVO has been developed to research the use of mobile agents for consumer-to-consumer E-commerce on the Internet. The mobile agent technology is the focus of my research.

1.1 Overview of the Thesis

Chapter Two will provide an overview of mobile agent technology. A general introduction to E-commerce is provided in Chapter Three. The reader will learn about the concepts, benefits, and the applications. Chapter Three also describes how the mobile agent technology is used in E-commerce. The functionality and the design of YAVO are detailed in Chapter Four. Chapter Five describes the implementation of YAVO. Chapter Six provides an application example that will include screen shots of the programs in action. Lastly Chapter Seven summarizes the research and makes recommendations for future work.

Chapter 2

2 Mobile Agent

2.1 Introduction

The growth of public information networks brings people closer and makes life easier. People can purchase a birthday gift through the Internet, buy or sell books on-line, make a bid on interested goods, etc. Stand-alone applications cannot fulfill these complicated tasks. Computers have to communicate with each other in a networked environment. Client/Server technology is commonly used in today's communication networks. But the increasing number of Internet users brings about a network bandwidth problem. A new approach called mobile agent programming is introduced to solve this problem.

2.1.1 Client/Server Programming

The Client/Server model splits an operation into two parts across a network. A client makes requests from a user machine to a server that services the requests, typically on a large, centralized system. Let us look at a simple example. Jane wants to bake a pumpkin pie for her mum. She has the pie ingredients but she does not know how to bake. She contacts a local bakery to bake the pie for her. The baker comes to pick up the ingredients and asks Jane to choose a recipe. After the pie is done, the baker brings the pie to Jane and charges for the service. In a computer communication world, Jane is a client, the bakery is a server that offers services to clients, and the baker is the network media between the client and the server that allows the client to request the services from the server. In the Client/Server approach, the client establishes a connection with one or more servers and sends them messages to complete a task. We can see that Client/Server programming needs good quality network connections because client's requests require full round trips to be completed and data may need to be copied across the network. This approach consumes network bandwidth for each message.

2.1.2 Mobile Agent Programming

Mobile agent programming allows a client to bring procedures and data to a server and execute them there, which involves high-speed, local communications. Thus, it overcomes the limitations in Client/Server programming. Let us discuss another example, Mary wants to prepare a sumptuous birthday dinner at home, but she does not have the time and the facilities to prepare the dishes. She gives the food, the menu, and all the recipes to her servant, Judy, and sends her to a restaurant where Judy cooks all the dishes and brings them back. We can call Judy a mobile agent who goes to the server (restaurant) and uses the facility there to fulfill all the tasks requested by Mary. Using mobile agent programming paradigm, a user's computer and its server can interact without using the network once the network has transported an agent between them. Thus this mobile agent approach consumes network bandwidth only when the agent moves.

2.2 Benefits

There are many benefits of mobile agents. First, they reduce the network load and provide high quality and high performance. Accomplishing a task in a network environment usually involves multiple interactions that generate a lot of network traffic.

Mobile agents package a conversation and interact with a destination host locally. Mobile agents also reduce the flow of large amounts of data that are stored at the remote hosts. The data is processed locally rather than transferred over the network. Mobile agents allow higher performance than the Client/Server model by moving the computation to the data rather than the data to the computation [LO99].

Second, they overcome network latency that is always a big problem for critical real-time systems in response to changes in their environments. Network latency is the time for a network packet to travel from the source to the destination. Mobile agents solve this problem because they prevent a lot of network communication between hosts and they are adaptive to the environment [LO99].

Third, they execute asynchronously and autonomously. Network computing often relies on expensive or fragile network connections. Sometimes it is not economically or technically practical to have a continuous connection between a machine and network. For example, a student with a notebook frequently connects and disconnects the notebook to the network. Mobile agents can solve this problem. After being dispatched, the mobile agents become independent and carry out tasks asynchronously and autonomously. The original host can reconnect at a later time to extract the mobile agents [LO99].

Fourth, they adapt dynamically. Mobile agents can sense and react to the changes in their execution environment. For instance, a mobile agent will move to a docking system upon network failure. The ability of mobile agents to react dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems [LO99].

Above all, a client can specify the behavior of the agent based on his or her own needs, which makes the mobile agent application more flexible and powerful.

2.3 Choosing the Appropriate Platform

There are many mobile agent platforms available. Voyager [Voyager 99], Aglets[Aglets 99] and D'Agents [D'Agents 99] are among the popular ones. A suitable platform is to be chosen to implement the YAVO system.

The Voyager system from ObjectSpace is a one hundred percent Java distributed computing system integrated with agent technology. Voyager includes a full-featured object request broker that handles the communication between clients and server. Voyager also provides services such as persistence, group communication, and directory services.

Aglets is a Java-based mobile agent system developed in IBM's Japan Lab. It presents the user with a visual environment for building mobile agents to search for, access, and manage information. Aglets system provides many services and facilities, such as a simple management environment, message passing facilities, authentication, and access control.

D'Agents from Dartmouth College is a multi-language mobile agent system. D'Agents currently supports Tool Command Language (TCL), Java and Scheme agents. D'Agents also offers some services, such as a docking system, a yellow page facility, and a debugger facility.

The remainder of this chapter will compare several aspects of these three systems: mobility, security, communication, directory, life span of agents, and languages.

2.3.1 Mobility

Mobility is the key feature of a mobile agent. Mobility brings out many benefits such as reducing network traffic, speeding communication and processing, and overcoming disconnection problems.

Voyager can move an agent to other programs, allowing the execution of itineraries. The agent sends itself a "moveTo()" message with two parameters. One is the destination address; the other is the name of the callback function, which will be executed on arrival. Voyager can also move an agent to other objects, thus allowing communication using high-speed local messaging. The agent can send itself a "moveTo()" message with a virtual reference to the destination object and a callback parameter. For instance, an agent in "dragon: 8000" can send itself a message to buy from a ticket agent in "owl: 9000" (8000 and 9000 represent a machine's port number). The agent sends a "moveTo()" message with two parameters: vticket - a virtual reference to the remote ticket object; and, buyTicket - the name of a callback function. The message is "moveTo(vticket, "buyTicket")." After moving to "owl: 9000", the mobile agent receives the callback message "buyTicket()" with a local native reference to the ticket agent and the mobile agent's "buyTicket ()" method is invoked.

In the Aglets system, two ways are used to move an agent. One is to dispatch an agent to a remote host. The other way is to retract the dispatched agent from the remote host. The migrated agent and its state would be serialized and de-serialized before and after moving.

In D'Agents, an agent can be moved by an "agent_jump" command, or an agent can send a copy of itself, called a child agent, to a remote host by an "agent_submit" command. The state of the migrated agent can also be saved and transferred to the remote machine. The remote machine will restore the state and the agent continues execution from the exact point of the jump. Once an agent has migrated to a machine, the agent can access resources and communicate with other agents on that machine. Once finished the local task, the agent can, if needed, migrate to the next machine.

2.3.2 Security

The mobility feature of a mobile agent brings out an important issue: the security mechanism. Usually security problems can be divided into three interrelated categories: protection of the machine, protection of the agent, and protection of a group of machines. The current focus on the security issue is to protect the machine. The agent's owner or sending machine digitally signs the agent. Then the receiving machine verifies the digital signature, accepting or rejecting the agent based on its signature, and assigns access restrictions to the agent. After this the agent is securely executed in an environment that enforces the restrictions.

Voyager has a security manager called Voyager Security Manager that restricts the operations of mobile agents. The security manager extends JDK Security Manager. Each different object has its own authority for each function.

In Aglets, certain security services are provided to protect both the machines and agents. One of the common security services is authentication that includes verifying user, verifying host, and verifying code or agent. To authenticate an agent, the PGP (Pretty Good Privacy) algorithm is used. A sender uses a randomly generated secret key (SecretSourceKey) to encrypt an entire agent, and then the sender encrypts the secret key by using the public key (PublicDestKey) of the destination machine. Then both of the keys and the agent will be sent to the destination machine. The destination machine can use its private key (PrivateDestKey) to open the public key (PublicDestKey). The destination machine will receive another private key from a source machine to open the secret key (SecretSourceKey). Other security services in the Aglets system include checking the integrity of an agent, authorization, and auditing security-related activities of an agent. Aglets security model supports security policies and describes the process whereby a secure system enforces these policies.

In D'Agents, an agent is also authenticated, authorized, and enforced to observe restrictions. Authentication is also based on PGP. To protect the system resources, security is maintained using a set of resource manager agents. To protect a group of machines, D'Agents uses a market-based approach in which agents pay for their resource usage with electronic cash. To protect an agent, several solutions have been introduced. For example, agents are sent to trusting machines only, otherwise they are permitted to communicate only with non-critical agents.

2.3.3 Communication

Agents are not isolated because they need to communicate with each other. The communication mechanism is a very important part of mobile agent systems.

Voyager has Message agents, which handle four types of messages: synchronous message, one-way message, future message, and one-way multicast message. To handle a ***synchronous message***, the caller blocks itself from its task until the message is completed and the return value is received. In asynchronous messaging, after sending a

one-way message, the caller does not self-block while the message completes. For a ***future message***, the caller does not self-block while the message completes. The caller receives a placeholder that can be used to retrieve the return value later by pulling, blocking, or waiting for a callback. For a ***one-way multicast message***, the caller sends a one-way message to all objects in a Space using a single operation. Multicast messages can be selectively broadcast to a subset of objects in a Space. Space is a type of computer-architecture designed to allow users to build distributed rooms in which objets reside. When you send messages into the Space, they are multicast in parallel to the objects in the Space.

In Aglets, agents communicate with one another through message passing, byte streams, and remote method invocation. There are two types of message passing. One is synchronous messaging; the other is asynchronous messaging. ***Synchronous messaging*** blocks the sender until a reply has arrived. ***Asynchronous messaging*** does not block the execution of the sender, and the sender retrieves the reply later. Aglets supports remote messaging. The remote messaging between agents is location-transparent because the Aglets Application Programming Interface (API) functions for remote message passing, and are the same as those for local messaging. Aglets system also supports multicasting messaging. Agents can subscribe and unsubscribe to specific multicast messages.

D'Agents has two-level communication mechanisms, which are low-level and high-level. In the low level mechanism, D'Agents provides byte streams and message passing. To use ***byte streams*** for communication, an agent establishes a direct connection with another agent using "agent_meet" command and exchanges message streams over the connection. In ***message passing***, two operations "agent_send" and "agent_receive" are used to send and receive messages between agents. A message is sent between two agents with blocking, non-blocking, and time restrictions. High-level protocols such as RPC (Remote Procedure Call) and KQML (Knowledge Query Manipulation Language) are implemented at the agent level.

2.3.4 Directory

A Directory service enables a connection to an existing object based on its name. This is a naming service, which allows users to build and link together hierarchies for the management of objects in a distributed system.

Voyager provides a directory service for looking up a remote object. With the directory service, an object can get a reference of a remote or mobile object through its alias after the remote object registers in the directory. Directories can span several virtual machines and can persist in a database.

In Aglets, every agent registers its current location in a database. From this information, the host can retract the migrated agent at any time.

D'Agents has a Yellow Page facility. It is a set of stationary yellow page agents, which maintain a set of references to specific service agents and other yellow page agents.

2.3.5 Lifespan of Agents

The life span of an agent defines the lifetime of an object before it is reclaimed by the system.

An object in Voyager can live as long as the server, live for a certain period of time, live until a specific point of time, or live until there are no more references to it. When an object reaches the end of its life span, it dies and is collected through a garbage-collection mechanism, which destroys an object and frees the machine's memory that was occupied by the object.

The default life span of an agent in the Aglets system is unlimited. The agent will be garbage-collected if there is no reference to the agent. The Aglets manager can dispose of each agent at any time.

Agents can live as long as the agent server of D'Agents.

2.3.6 Language

An agent programming language should be complete and sufficiently powerful, so that agents can make decisions, handle exceptional conditions, gather, organize, analyze, create, and modify information.

Both Aglets and Voyager use Java as an agent language because Java has many advantages. First, Java is platform independent and an agent can run on any platform like Windows, Unix, or Linux, etc. Second, Java is secure and does not allow illegal type casting or any pointer arithmetic. Java also has a security manager to check all potentially dangerous operations, such as file access and network connections. This is very useful for mobile agent systems that have to protect their agents and the hosts. Third, Java supports a dynamic class-loading feature, which allows the Java program to dynamically load classes. The loaded class is included in an application, locally or through the network. Thus, Java-based agents can load their code from a variety of sources, such as the local file system, the Web, and FTP (File Transfer Protocol) servers. Fourth, Java supports multithread programming. An agent executes independently of other agents within the same place. Fifth, Java's object serialization allows Java mobile agents to be serialized and de-serialized almost transparently. Agent mobility requires facilities that convert an agent and its state into a form suitable for network transmission. And on the receiving end the facilities allow the remote system to reconstruct the agent. Sixth, Java also provides Servlets that function similar to a Common Gateway Interface script and may launch and receive mobile agents. The YAVO system uses Java servlets as a bridge between the User Interface and the backend server.

In conclusion, the features of multi-platform support, secure, dynamic class loading, multithreading, object serialization and Java servlets make Java well suited for mobile agent technology.

D'Agents supports multiple languages including Tcl, Java, and Scheme. Both Tcl and Scheme are script languages. A script language is easier than Java to learn, and to develop applications; but the applications are limited because script language is not an object-oriented language.

2.3.7 Choice

The Aglets system is perceived to have the most easy-to-use user interface (UI). By interacting with the agent viewer window, the user can create a new agent, send a request to an agent to open its dialog panel, show the properties of the agent, destroy, clone, dispatch, and retract the dispatched agent. This UI feature is what other mobile agent platforms lack.

Voyager is a commercial system with complete documentation. It is not only a mobile agent system, but also a distributed computing system. It has more distributed system features than Aglets, such as CORBA integration. Voyager also provides features like space group communications and database-independent persistence. There is a lightweight object storage system in Voyager, where objects and agents can be flushed rapidly to disk. This storage system helps to manage large numbers of objects within a single application, and increases the reliability and recovery of applications after a system failure or shut down.

Although the Voyager system is more complicated than the Aglets system, it is more powerful. What can be implemented in Aglets can also be implemented in Voyager. Because of the advantages of Voyager over Aglets, Voyager has been chosen to be the mobile agent platform for the YAVO system.

2.4 Conclusion

In conclusion, a mobile agent is an independent program that travels across a network. It continuously and autonomously executes in a remote machine on behalf of a user. Mobile agent programming provides a new paradigm for traditional network computing. The Client/Server model needs constant communication between the client and the server. The mobile agent programming approach does not need a network connection once an agent arrives at the server. The mobile agent model has the advantages of reduction of network traffic, overcoming of network latency, and great flexibility. To implement the YAVO system, the Voyager platform is chosen because of the powerful features described above.

Chapter 3

3 Electronic Commerce and Mobile Agents

3.1 What is Electric Commerce?

3.1.1 Definition

Due to fast growth of the Internet and the World Wide Web, electronic commerce (E-commerce) is growing widely and rapidly. What is E-commerce? The first thing that comes to mind is on-line shopping, but Web shopping constitutes only a small part of E-commerce. This term also refers to on-line stock or bond transactions, on-line buying and downloading of software, Electronic Mail (E-mail), FAX transmissions, Electronic Data Interchange, etc. E-commerce is simply commerce conducted electronically. In the recent past, E-commerce has included technologies such as the telegraph, telephone, and fax. Presently, the focus is increasingly on networked computers [CL 97].

3.1.2 Categories of E-commerce

3.1.2.1 Business to Business

An example in this category would be a company that uses the Internet to order from its suppliers, receive bills, and make payments. The business has strong potential for growth.

3.1.2.2 Business to Government

Business-to-Government E-commerce covers all transactions between companies and governmental organizations. It is more restrictive than other categories due to government restrictions.

3.1.2.3 Business to Consumer

Business-to-Consumer is also called "Consumer Electronic Commerce" or "Retail Electronic Commerce." It deals with the on-line selling and buying of products and services. Companies publish their catalogs on-line, consumers order from the catalogs, make payments, and track the status of their orders on-line. The customer uses a web browser to access a Web-based virtual store through the Internet.

There are now shopping malls on the Internet, which offer all manner of consumer goods, from cakes and wine, to computers and cars. An example is Amazon.com's on-line bookstore [AMA 99]. Customers visiting Amazon.com can browse, search using keywords, and obtain detailed information on individual titles. Detailed information includes a cover page, the price, an excerpt, a table contents, customer reviews, and recommendations for similar types of books. Consumers can order and pay for books that are then delivered within twenty-four hours. Another example is Wine.com [WINE 99], Virtual Vineyards. This Web site offers wines and gourmet foods, and provides an outlet for a number of small Californian wine producers. There is detailed on-line information on various wines and foods, and also an on-line query service (using E-mail). Customers can order, and pay, using either credit cards or E-cash. Customer orders are transferred electronically from Virtual Vineyards' San Jose office to their Napa Valley warehouse, along with instructions for printing the shipping label and enclosures (such as tasting notes). Federal Express ships the goods to the customer. Customers can track the progress of the delivery on-line by accessing the Federal Express site. The number of small businesses with a Web presence has nearly doubled since 1998. According to results of a new survey [NEWS 99a], millions more are projected to come on-line next year.

3.1.2.4 Consumer to Consumer

Consumer-to-Consumer business covers business among individual consumers. An example of this category is an on-line bidding system such as the research described in this thesis. The YAVO system is an on-line trading system based on Consumer-to-Consumer interactions.

3.1.3 Technology

According to Hamer and Champey, "the real power of technology (such as electronic commerce, telecommunications services or multi-media) is not that it can make the old processes work better, but that it enables organizations to break old rules and create new ways of working - that is, to reengineer" [HC 99]. E-commerce is a generic title used for business models that use the range of technologies that are now available to improve the effectiveness of trading relationships. Typical technologies include electronic data interchange (EDI), bar codes, E-mail, the Internet, the World Wide Web, electronic forms, E-cash, etc. The following explains these technologies in detail.

3.1.3.1 Electronic Date Interchange

In addition to the traditional Internet protocols such as HTTP, E-commerce uses several of its own standards, most of which apply to business-to-business transactions. One of the most popular standards is Electronic Data Interchange (EDI). In the early 1970's, the US government created the Electronic Data Interchange (EDI). EDI is the electronic exchange of standard business forms: quotations, purchase orders, invoices, receipts, etc. Information stored on one computer is translated by software programs into standard EDI format for transmission to one or more trading partners. In turn, the trading partners' computers translate the information into a form that can be understood.

EDI is a central part of electronic commerce because it enables businesses to exchange information electronically, faster, more cheaply and accurately than paper-based systems. EDI is widely used in manufacturing, shipping, warehousing, construction, food processing, banking, insurance, retailing, government, health care, and much more. About fifty thousand private-sector companies in the United States, such as Federal Express, Eastman Kodak, American Airlines, Nike, and Prudential Insurance currently use EDI [ECO 99].

3.1.3.2 Open Buying on the Internet

Among the other E-commerce standards is Open Buying on the Internet (OBI). This standard ensures that buying organizations and selling organizations are able to collaborate. The standard contains the architecture, detailed technical specifications & guidelines, and compliance & implementation information. Any organization or individual can acquire a copy of the OBI standard and use it to build a product, service, or solution [OBI 99]. Leading technology companies such as Microsoft and Oracle support OBI because its solutions complement, rather than replace existing EDI infrastructures.

3.1.3.3 Bar Codes

Bar codes are used for automatic product identification by a computer. These codes consist of a rectangular pattern of lines that vary in width and space. Specific characters (e.g. numbers 0-9) are assigned unique patterns, thus creating a "font" which computers can recognize based on interpreting the light from a laser reflected off the lines.

The most obvious example of bar codes occurs on consumer products such as packaged foods. These codes allow the products to be scanned at the check out counter, where the product is identified and the price is automatically entered.

3.1.3.4 Electronic Mail

The typical use of E-mail involves messages that are composed by individuals and sent in digital form to the recipient via the Internet. This communication has become a very important part of everyday life. Relationships among people are becoming closer in nature through communication by E-mail.

3.1.3.5 The Internet

The Internet is a network of phone and data lines all over the world and is used to transfer data through the use of a specific protocol or language. The Internet initially began as a means for the US government and research institutes to share data and communicate. In 1994, the Internet was opened to the public, to allow commercial use to any and all entities looking to take advantage of this medium. Since that time, the Internet has grown tremendously. It has become an efficient tool for communications between people and businesses. This network is growing very quickly and as the Internet becomes more accessible, its usage will also become more widespread.

3.1.3.6 The World Wide Web

The World Wide Web is a collection of documents written and encoded with the Hypertext Markup Language (HTML) using HTTP. With the aid of a piece of software (called a 'browser'), a user can request these documents and display them on the user's local computer. This can take place even though the document is copied from a computer on a totally different network elsewhere in the world. HTML documents contain many kinds of information such as text, pictures, video, sound, and hyperlinks that direct users immediately to other web pages. It is this ability to jump from site to site that gave rise to the term the "World Wide Web." Browsing the Web (or "surfing the Net") can be a fascinating activity, especially for people new to the Internet. The World Wide Web is, by far, the most heavily used protocol on the Internet.

3.1.3.7 Electronic Forms

An electronic form is a digital analogue of the paper form. The electronic form is an image that looks like a form but appears on a computer screen, and is filled out using a mouse and keyboard. Electronic forms allow storage in databases, automatic information routing, and integration into other applications. An example of an electronic form is on-line registration at the Acadia University Web site. Electronic forms bring great efficiency and convenience to our lives.

3.1.3.8 E-Cash

Digital or electronic cash (E-cash) allows a person to pay for goods or services by transmitting numbers from one computer to another. The numbers, just like monetary bills, are issued by a bank, and represent specified sums of real money. Two of the main features of digital cash, as with real cash, are anonymity and reusability. These features are the key difference between E-cash and credit card transactions over the Internet. E-cash brings great convenience to sellers and buyers in finalizing a transaction.

3.2 Why E-commerce?

3.2.1 Benefits for Consumer

Using the Internet to purchase products and services gives a consumer these benefits: access to more information, easier market research and comparison, lower product and service cost, and lower purchasing cost.

- Access to more information:

On-line shopping is more consumer-driven than shopping conducted traditionally. On-line information is available worldwide and provides extensive marketing information on products of interest. In addition, consumers can do an on-line series of tests of digital products (e.g. music) and immediately download these products.

- Easier market research and comparison:

The ability of the Web to gather, analyze, and control large quantities of specialized data, enables comparison-shopping and speeds up the process of finding items. From the comfort of home or office, the consumer can have access to price comparisons on the Web any time of the day, any day of the year, without having to wait in line.

- Lower product and service cost:

As suppliers are able to compete in an electronically open marketplace, prices of products or services are naturally lower. Competition among suppliers also leads to better quality and variety of goods, through expanded markets and the ability to produce customized goods.

- Lower purchasing costs:

Purchasing goods or services for a corporation is a very complex process. It includes identifying the supplier, defining customer specifications, transmitting purchase orders, receiving notification, paying invoices etc. Some large companies have been using EDI over private networks to reduce labor, printing, and mailing costs in the procurement process.

3.2.2 Benefits for Vendors

The use of the Internet to sell, distribute and maintain products and services leads to significant cost savings and increased sales opportunities. The benefits are: lower cycle times, more efficient and effective customer services, lower marketing and sales costs, and new sales opportunities.

- Lower Cycle times:

Cycle time is the total time it takes to build a product. E-commerce allows the cycle time to be shortened, allowing more products to be produced for the same, or lower costs. Sharing information electronically allows the different members of the group to work on projects together, rather than having to wait for each member to finish his/her step before the next one can be taken.

- More efficient and effective customer service:

On-line product descriptions, technical support, and order status information free up customer service staff to handle more complicated questions and better manage customer relations. Customers request as much information as desired, and the vendor obtains relevant information from customers for the purpose of serving them more effectively in the future.

- Lower marketing and sales costs:

Companies publish information, services, or digital product categories on the web to shrink the distribution costs (cost-of-sales) to zero. For example, digital products can be delivered immediately through the Web. Furthermore buyers and sellers can access and contact each other directly, potentially eliminating some of the marketing cost. In a traditional sale, the more customers there are, the more sales persons are needed. By contrast, E-commerce can add new customers with little or no additional cost. Because its sales functions are housed in a computer server, rather than at an actual store with live sales people, the reach of the service is only limited by the capacity of the servers to respond to inquiries and orders.

- New sales opportunities:

The Internet is worldwide. As a result, businesses on the web can reach new markets globally. Many companies find they attract new customers through web business. These is due to the ability to reach potential customers easily and cheaply, as well as eliminating delays between the different steps of the business process. One success story is the Dell on-line store [GREENSPAN 99]. Its on-line sales more than doubled during 1998 rising to more than S14 million per day, accounting for 25 percent of the company's total revenues. During the quarter ending 30 April 1999, on-line sales rose further to an average of S18 million per day and now account for 30 percent of the company's S5.5 billion first quarter revenues. Dell expects this percentage to increase to 50 percent by 2000.

3.2.3 Problems

A serious deficiency arises from the use of the Web as a marketing channel. Many users do not trust the Web as a payment channel. Any person, who transfers the data of a credit card on the Web, cannot be sure about the salesman's identity. The salesman cannot be sure about the buyer's identity. The one who pays can not be sure that his or her credit card number will not be collected somewhere on the Web and be used for some malicious purpose. Also, the salesman cannot be sure that the credit card owner will not deny the acquisition.

In connection with the previously outlined problems, there are number of questions concerning marketing through the Web: validity of an electronic signature, legality of an electronic contract, violations of trademark and copyright, etc. The most glaring inadequacy in the Internet's commercial infrastructure is the lack of a secure form of monetary transfer. Solutions for these issues are on the way. There are different options available in electronic money transfer, from simple exchanges of credit card numbers to smart cards that enable anonymous, highly flexible, and fully automated digital accounting systems. Most of these techniques involve cryptography [Solinsky 99]. Cryptography is a technology to encrypt and decrypt messages for transmission. There are two kinds of cryptography: symmetric key and public key. The key is what you use to "unlock" a message. In symmetric key cryptography, the sender and receiver have the same key. In public key cryptography, there is a public key for sending and a private key for receiving. Cryptography brings order and security to the otherwise natural disorder of the Internet. For further information on Web security and commerce, please refer to the book "Web Security & Commerce" [GS 97].

3.3 Where is E-commerce Applied?

The most common form of business is the actual exchange of currency or credit for goods and services. As Internet security issues are resolved, businesses are selling more and more products on-line, directly to their customers. According to a recent study, electronic retail sites will see peak sales in November 1999, especially among newer Internet users. Fifty-one percent of all Internet users and 73 percent of new users said they are planning to do a majority of their on-line shopping in November [NEWS 99b]. "A toy maker in Peoria can sell more dolls in her hometown, her home state, or clear across the country. The Internet shatters geographical boundaries, giving all small businesses a virtual local and national sales force with just a few clicks, [NEWS 99b]." The number of small businesses that have Web sites is expected to grow in the next few years.

Businesses also promote and market their products through the Internet. The World Wide Web provides an electronic means for organizations to display materials such as product catalogs, price lists, and brochures all on-line. For instance, Hewlett Packard (http://www.hp.com) has the "Access HP" Web site that provides thousands of pages of information, including general company information, news, worldwide contact points, new product announcements, and details of HP's wide range of products and services.

Another business use of the Internet is to disseminate information or to publish customer support documents. The use of E-mail is an effective way to communicate. Many high technology companies are providing on-line technical support, allowing their customers to easily find answers to problems and to download needed files and software round the clock, saving the company money on support personnel and mailing costs.

3.4 Why is Mobile Agent Technology Useful in E-commerce?

Mobile agents are very useful in E-commerce. They help automate tasks including trading goods over the Internet. Currently, most of the commerce needs human interactions. People decide when to buy goods, how much they want to buy, and so on. Sometimes a commercial transaction may require real-time access to distributed resources. Mobile agent technology is a very appealing solution for this kind of problem.

The following section will discuss how mobile agents automate some steps in the Consumer Buying Behavior. Basically a consumer's buying behavior consists of six steps [MGM 99]: need identification, product brokering, merchant brokering, negotiation, purchase delivery, and product service and evaluation.

3.4.1 Need for Identification

First of all, in the stage of need identification, the buyer may not know what to buy or may not be planning to buy anything. Agents can gather information to help the buyer to identify his or her need. For example, agents can monitor goods for sale and notify the customer when certain events occur that may be of interest to the consumer. For example, in the YAVO system, a trader can send a monitoring agent to the distributed agent marketplaces and gather recent agent marketplace information for the trader.

3.4.2 Product Brokering

Secondly, in the stage of product brokering, the buyer may not know what kind of product to buy. Agents can help to retrieve information, evaluate product alternatives, and finally provide a list of products according to the buyer's requirement. For instance, a trader wants to rent a two-bedroom apartment. The buyer agents that the trader sends into the rental agents' marketplaces can help the trader to find all of the potential seller agents who are renting two-bedroom apartments.

3.4.3 Merchant Brokering

Thirdly, in the stage of merchant brokering, the buyer may not have decided from whom to buy. Agents can collect merchant-specific information and evaluate merchant alternatives, thus providing the information about the best merchant according to the buyer's requirements. Continuing the previous example, if the buyer agent finds a list of potential seller agents, the buyer agent will compare additional features along with the price of all the seller agents, and then choose the one that best meets the trader's need.

3.4.4 Negotiation

Fourthly, in the stage of negotiation, agents will help the buyer close a deal as soon as possible. Negotiation can be very simple if the prices and other aspects of the deal are fixed. On the other hand negotiation can also be very complex in some markets, such as the stock market, second-hand market, and fine antiques market. Negotiation varies accordingly in different markets. In the YAVO system, if a buyer agent or a seller agent can not find a deal after one day, the buyer agent or the seller agent will adjust its price based on a price changing strategy set by its trader. The price changing strategy is one of the mathematics functions such as linear, quadratic, or exponential, etc. The agent will then negotiate with other agents again and try to find a new deal.

3.4.5 Purchase and Delivery

Fifthly, in the stage of purchase and delivery, agents can represent the buyer in closing the deal and can make a payment to the bank system or directly to the seller. The product will be delivered if the transaction succeeds. In the YAVO system, E-cash is used to make all transactions. E-cash, a number representing sums of real money, is as convenient as real money. Mobile agents carry E-cash in hand to pay for goods or services, transmitting the E-cash without the intervention of a bank or a broker.

3.4.6 Product Service and Evaluation

Finally, the stage of product service and evaluation includes product service, customer service, and evaluation of the satisfaction of the whole process. Agents can help to evaluate and improve the consumer's buying decisions. The YAVO system does not use agents to help in this step. However, it could be an added feature in the future.

3.4.7 Summary

In conclusion, mobile agents embody the intentions of their creators, act, and negotiate on their behalf, reduce transactions costs in E-commerce, and revolutionize how we do commerce in the future.

3.5 Conclusion

In summary, E-commerce is defined as transacting business electronically. The use of E-commerce technologies is becoming a condition of trading imposed by large customers, especially in the retail, manufacturing and automotive sectors. E-commerce increases the speed and efficiency of business transactions and processes, and improves customer relationships and services. Moreover, increased competition results in a reduction of prices of goods and services. E-Commerce is growing explosively on the Internet. By the end of year 2000, millions of individuals and companies will be buying, selling, bidding, advertising, brokering, and collaborating on a daily basis. As the Internet merges with other branches of the information highway, E-commerce will become a fully integrated part of our lives.

Chapter 4

4 YAVO: Functionality, Analysis, and Design

4.1 Functionality

4.1.1 System Purpose

YAVO is an automatic on-line trading system using Java, Web, and mobile agent technology, thus making YAVO easy-to-use, and platform independent. The purpose of the YAVO system is to assist end-users selling and buying goods through mobile agents on the Internet. A system administrator creates trader accounts and distributed agent marketplaces. An agent is sent to an agent marketplace (for this example it will be assumed that there is one marketplace per good) representing an end-user in buying and selling specific goods based on user-specified strategies. In the agent marketplace, each buyer agent negotiates with several seller agents daily. The negotiation interval is flexible. It can be one hour or half a day. If a pair of agents reaches an agreement on a deal, they will complete the payment using E-cash. Through this means traders (buyers or sellers) do not have to spend time searching for information, negotiating each potential deal, and worrying about payment.

4.1.2 System Features

System features are the functions that an end-user can perform on the system. The end-users of YAVO system are the traders who buy or sell goods through YAVO and the system administrator who maintains the traders, bank and agent marketplaces.

The following are the features designed for traders:

- View all the agent marketplaces and their policies
- Configure and send a buyer or seller agent to an agent marketplace
- View the current status of the created agents
- View all the messages from the agents
- View the current balance in the bank account (not implemented)
- View the amount of E-cash in hand (not implemented)
- Withdraw E-cash from the bank (not implemented)
- Deposit E-cash to the bank (not implemented)

The system administrator can perform the following tasks:

- Create traders accounts
- Create agent marketplaces in different environments
- Create a bank object on a host

4.1.3 Business-Related Objects and Their Responsibilities

An object is a person, thing, or place. Each object is associated with data attributes and behavior. In the system features we find business-related objects, human interaction objects, and database objects. The major objects of a system are business-related objects that respond directly to most of the system features. In the following, we discuss three types of business related objects: actors, places, and things.

4.1.3.1 Actors

An actor is a person, an organization, or an agent that fulfills specific tasks. In YAVO, the actors are traders and mobile agents. A trader can be either a buyer or a seller. The mobile agents include buyer agents and seller agents.

4.1.3.1.1 Traders

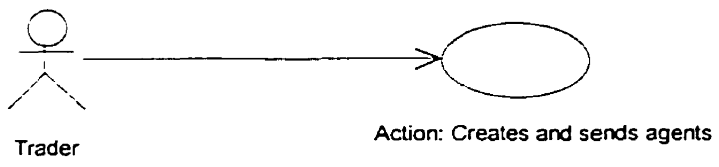


Figure 4-1 Use Case Diagram for Trader (UML [Rational 98])

A trader is a buyer or seller who creates and sends agents to sell or buy goods through the YAVO system where every trader has an authorized account maintained by an administrator. When a trader is created, it has some E-cash in the bank and some E-cash in hand. Every trader is also a holder of a bank account, thus enabling the trader to withdraw or deposit E-cash into a bank.

4.1.3.1.2 Mobile Agents

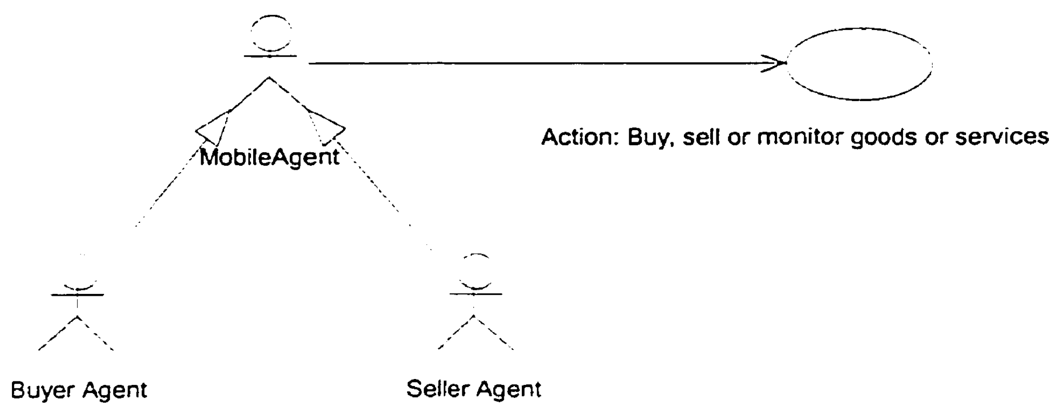


Figure 4-1 Use Case Diagram for Mobile Agent (UML)

A mobile agent is a piece of software that can move across the Internet and execute autonomously and continuously in a specific environment. Mobile agents include buyer agents and seller agents. An agent can also be classified as a singleton agent or a multicast agent. A singleton agent is the only agent that is sent by a trader to find a specific item, whereas multicast agents are sent to look for the same item in several agent marketplaces. A singleton agent does not have a transaction ID while a multicast agent carries the same transaction ID as the other multicast agents who are looking for the same item. If one of the multicast agents completes a deal and returns back to the trader, the trader can dispose of all the other multicast agents who are looking for the same item.

Mobile agents have the following attributes that a user can view at any time:

- Agent ID

  Agent ID is a unique id issued to an agent when it is created.

- Starting price

  The starting price is the desired price that the trader specifies.

- Final price

  The final price is the acceptable price that the trader specifies.

- Lifetime of an agent

  A trader specifies the lifetime of an agent, which indicates how long an agent is going to stay alive in the agent marketplaces. The time unit in YAVO is one day. For example, a buyer agent stays in a marketplace for 5 days.

- Price Changing Strategy

  According to a user-specified price changing function, an agent changes its price daily. For example, a buyer agent changes its price according to a linear function.

- Need Trader Approval Flag

  This flag is set by a trader to indicate if a trader's approval is required before closing a deal. With this option. a trader can have the final decision concerning the deal.

- Current Price

  The current price, which ranges from the starting to final price, is automatically changed on daily basis depending on the price changing strategy.

- Status

  The status of an agent can be "alive," "success," or "dead." "Alive" indicates an agent is working in an agent marketplace, "success" indicates an agent has found a deal and stops working, and "dead" means an agent could not find a deal during its lifetime.

- Goods information

An agent carries goods information such as goods type and the detail of the goods.

- Transaction ID

  A transaction ID differentiates between a singleton agent and a multicast agent. This value for a Singleton agent is empty. All the multicast agents, who are looking for the same item, carry the same transaction ID.

- Bank Account

  Each agent carries the bank account number of its trader. The marketplace obtains the bank account number when the agent arrives and serves as a broker between buyer agent and seller agent. The bank account information is only accessible by the agent marketplace.

- E-cash

  E-cash is used in transactions among agents and agent marketplaces.

After moving to a marketplace, a buyer agent queries the goods features and the current price of each seller agent daily. If the buyer agent is satisfied with the goods and the price, it adds this seller agent to its potential seller agent list. After contacting all seller agents in a marketplace, there can be several potential seller agents. The buyer agent compares the price and optional features of the potential seller agents. The buyer agent then chooses the best seller agent. Before finding a deal, a mobile agent updates its current price daily. Seller agents decrease the price, and buyer agents increase the price.

If a final approval is needed from a trader, the agent sends a message to the trader and waits for the response. The agent assumes that the trader has canceled the deal if there is no response within one day. If the trader replies and cancels the deal, the agent marketplace takes the deposit and service charges from the agent, and the agent goes back to its original place. The agent dies after returning the remaining E-cash to its trader. If both the buyer and seller approve the deal, the buyer agent pays the seller agent by E-cash, the marketplace charges both of them the service fee, and finally the agents go back home. The agents die after returning the rest of money to their traders. After the deal is finalized, the product or service can not be returned or exchanged.

4.1.3.1.3 Administrator

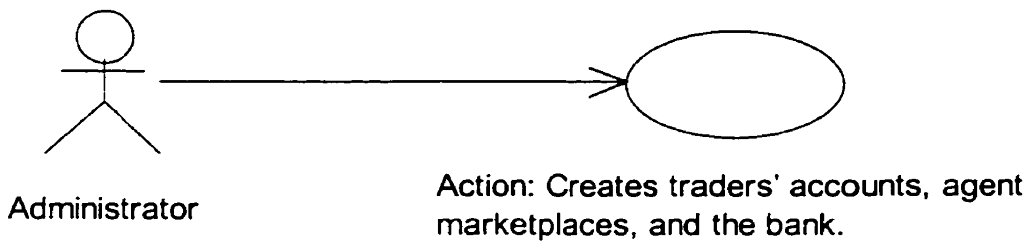

Administrator
Action: Creates traders' accounts, agent marketplaces, and the bank.

Figure 4-1 Use Case Diagram for Agent Marketplace (UML)

An administrator creates traders' accounts, agent marketplaces, and the bank. An account must be created for a trader before he or she can use the YAVO system. The administrator specifies the username and password for each trader and issues a certain amount of E-cash to each trader. The administrator creates the bank and agent marketplaces on the ports of different hosts or different ports of one machine. The administrator specifies each marketplace's attributes such as name, maximum number of agents, type of goods, location, service charge, and deposit.

4.1.3.2 Places

The place denotes where the objects live. In YAVO, the major places in which seller agents and buyer agents negotiate with each other are called agent marketplaces.

The other place is a bank. (I did not implement this object.) An administrator has the responsibility to create all these objects using appropriate factories. In the following we will describe the agent marketplace followed by a discussion of the bank.

4.1.3.2.1 Agent Marketplaces

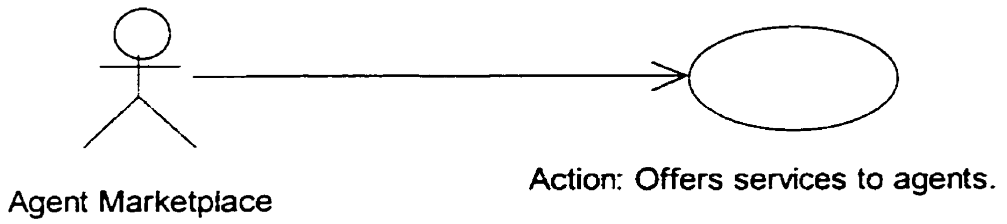


Figure 4-1 Use Case Diagram for Agent Marketplace (UML)

Agent marketplaces, where different mobile agents live and have social contact, are located on different hosts or different ports of a same machine. Their functions are to offer helpful services to agents and to manage the agent marketplaces by some policies. After logging into the YAVO system, the trader can view the attributes and policies of all the agent marketplaces. The attributes or policies are described in the following:

- The type of goods in marketplaces, i.e.: used cars, used books, or houses for rent.
- Maximum number of active agents in the marketplace
- Current number of buyer agents or seller agents in the marketplace
- Service charges for seller agents or buyer agents

The cost is based on the number of days that the agent stays in the agent marketplace.

- Deposit

If the trader refuses a found deal that is agreed upon by both a buyer agent and a seller agent, the deposit is given to the agent marketplace.

- Bank account
- Lists of seller agents
- List of buyer agents

When an agent arrives at a marketplace, the agent marketplace can retrieve the agent information including lifetime, starting price, current price, the amount of E-cash on the agent, and transaction ID. The marketplace checks if the total number of current agents exceeds the maximum number of its capacity and checks if the agent has brought enough E-cash into the marketplace. The E-cash required includes the service charge at the marketplace, the deposit to the agent marketplace, and the cost of the good. If the agent marketplace is full or if the agent does not bring enough money, the marketplace asks the agent to go back to its trader. The agent dies after returning the money to the trader. If both of the conditions are satisfied, the marketplace checks the type of the incoming agent and adds it to either the buyer agent or the seller agent list. The Agent marketplace offers seller agents information when buyer agents try to negotiate with seller agents.

4.1.3.2.2 Bank

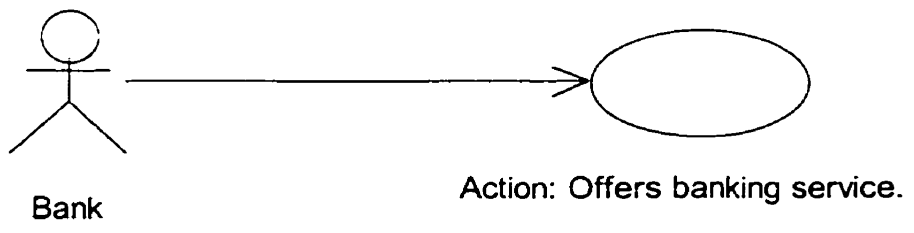


Figure 4-1 Use Case Diagram for Bank (UML)

A bank offers banking services for the YAVO system. It holds accounts for traders and agent marketplaces. All the E-cash used in YAVO is issued and authorized by the bank. A trader can check the current balance in the bank, deposit, or withdraw E-cash in the bank. Each agent marketplace also has a bank account. An agent marketplace can have the same banking services as the trader.

4.1.3.3 Things

4.1.3.3.1 E-cash

E-cash is used in all the transactions in YAVO. E-cash represents specified sums of real money and allows a person to pay for goods or services by transmitting a number issued by a bank. The key features of E-cash are anonymity and reusability. This feature is a key difference between E-cash and credit card transactions over the Internet.

4.1.3.3.2 Goods

In the YAVO system, there are many kinds of goods, such as a house for rent, used books, CDs, etc. A buyer specifies the goods to have both required and optional features. The details of the two kinds of features are different from goods to goods. For example, a tenant needs an apartment to have required features as number of bedrooms and location. The tenant may also prefer to have a dishwasher and a fireplace as optional features. When creating a buyer agent, a trader specifies the priorities of all the optional features and the current price. A buyer agent considers the priorities when it finds more than one potential deal. In the following, I will give an example to demonstrate a comparison of the seller agents using a point scheme.

A buyer is looking for a two-bedroom apartment. The required features that the buyer specified are two rooms, eight-month lease, close to Acadia University. The optional features are a dishwasher, and an indoor-laundry. The buyer sets the priorities of optional features and price by points. If an apartment has a dishwasher, it has 10 points, if it has an indoor-laundry, it has 9 points, if it has the cheapest price, it has 8 points. A buyer agent finds two apartments that meet with buyer's required features and the price. Apartment 1 has an indoor laundry and the cheapest price; thus the total points are 17. A dishwasher accords apartment 2 a total of points 10. The buyer agent chooses the apartment with the highest points; thus apartment 1 is selected to be the final deal. If the final points are equal among some potential seller agents, the latest one is chosen to be the final deal.

4.2 Analysis and Design

4.2.1 CRC Analysis (CRC cards)

In the following section, CRC cards of all the objects described above are provided. CRC cards specify the behaviors (methods) and attributes (variables) of objects in a class.

4.2.1.1 Traders

Class: Trader	
Responsibilities: Provides general functions for a trader.	
Collaborators: Bank, E-cash, Buyer Agent, Seller Agent	
Super classes:	
Subclasses:	
Methods	**Variables**
Obtain E-cash	Bank Account
Obtain bank account	E-cash
Obtain password	Buyer Agents
Change password	Seller Agents
Deposits money	Username
Withdraws money	Password
. . .	...

Table 4-1 CRC card for Trader

4.2.1.2 Mobile Agents

Class: Mobile Agent	
Responsibilities: Updates current price and time-stamp. provides agent information.	
Collaborators: E-cash. Trader. Agent Marketplace	
Super classes:	
Subclasses: Seller Agent. Buyer Agent	
Methods	**Variables**
Update life time	Agent ID
Update current price	Starting Price
Send message to trader	Final Price
Set status	Lifetime
Return home	Price Changing Strategy
Dispose itself	Need Trader Approval Flag
Obtain current price information	Goods Type
Obtain goods information	Current Price
Obtain starting price information	Transaction ID
Obtain final price information	Bank Account
Obtain lifetime information	E-cash
Obtain transaction ID information	...
...	

Table 4-1 CRC card for Mobile Agents

Class: Seller Agent	
Responsibilities: Provides additional functions to seller agent. i.e.: accepting E-cash from buyer	
Collaborators: Buyer Agent, E-cash, Trader, Agent Marketplace	
Super classes: Agent	
Subclasses:	
Methods	**Variables**
Confirm A Deal	
Take in E-cash	
...	

Table 4-1 CRC card for Seller Agents

Class: Buyer Agent
Responsibilities: Provides additional function to buyer agent, i.e.: negotiating with seller
Collaborators: Seller Agent, E-cash, Trader, Agent Marketplace
Super classes:

Subclasses:	
Methods	**Variables**
Negotiate with seller agent	Best Seller Agent
Add a potential agent to a list	Potential Seller Agents
Compare seller agents	...
Set the best seller agent	
...	

Table 4-2 CRC card for Buyer Agents

4.2.1.3 Administrator

Class: Administrator	
Responsibilities: Maintain trader accounts, bank and agent marketplaces	
Collaborators: Bank, Trader, Agent Marketplace	
Super classes:	
Subclasses:	
Methods	**Variables**
Creates trader account	Usernames
Deletes trader account	User Passwords
Create bank	Bank
Create agent marketplace	Agent Marketplaces
Delete agent marketplace	...
...	

Table 4-1 CRC card for Administrator

4.2.1.4 Agent Marketplace

Class: Agent Marketplace	
Responsibilities: Offers information and service to agents.	
Collaborators: Agent. E-cash	
Super classes:	
Subclasses:	
Methods	**Variables**
Evaluates agent	Name
Adds seller agent	Maximum Number of Agents
Adds buyer agent	Service Charge
Deletes seller agent	Deposit
Deletes buyer agent	Type Of Good
Search seller agents	Location
Marketplace Is full	Bank Account
Charges agent	Seller Agents
Send back agent	Buyer Agents

...	...

Table 4-1 CRC card for Agent Marketplace

4.2.1.5 Bank

Class: Bank	
Responsibilities: Offer banking service to traders and agent marketplaces.	
Collaborators: E-cash	
Super classes:	
Subclasses:	
Methods	**Variables**
Offers trader balance information	E-cash
Offers marketplace balance information	Trader Bank Accounts
Processes deposit	Agent Marketplace Bank Accounts
Processes withdrawal	...
Issues E-cash	
...	

Table 4-1 CRC card for Bank

4.2.1.6 Electronic Cash (E-cash)

Class: E-cash	
Responsibilities: It is used in all the transactions.	
Collaborators:	
Super classes:	
Subclasses:	
Methods	**Variables**
Increases amount	Amount
Decreases amount	...
...	

Table 4-1 CRC card for E-cash

4.2.1.7 Goods

Class: Goods	
Responsibilities: Provides goods information and comparison mechanism.	
Collaborators:	
Super classes:	
Subclasses:	
Methods	**Variables**
Calculates priority points	Required Features
Obtains required features	Optional Features
Obtains optional features	Priority Points
Compares goods	Type Of Good
...	...

Table 4-1 CRC card for Goods

4.2.2 Static Model Design (Class Diagram)

The following class diagram shows the relationships and behaviors of hierarchies of classes through an UML (Unified Modeling Language). The Unified Modeling Language (UML) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex process of software design, making a "blueprint" for construction [RATIONAL

99]. More information on UML is available at Rational Rose Web site: http://www.rational.com/uml/index.jtmpl.

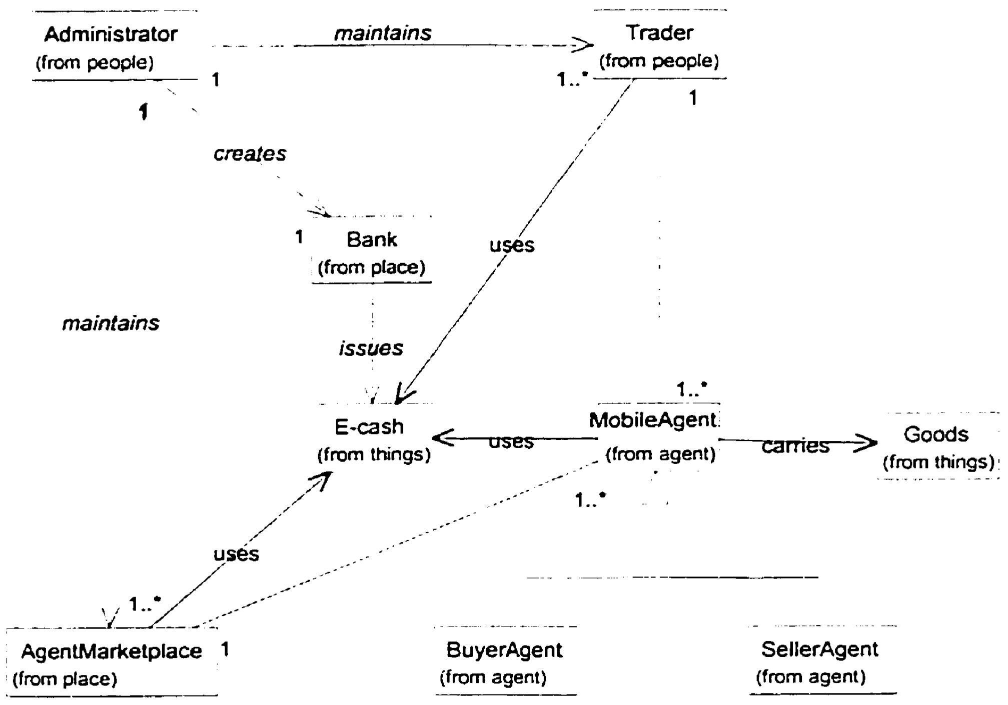


Figure 4-1 Class Diagram of the YAVO system (UML)

4.2.3 Dynamic Model – Scenarios

Dynamic model describes changes to the objects and their relationships over time. A scenario is a sequence of events that occurs during one particular execution of a system. An event is something that happens at a point in time, such as *a trader in YAVO withdraws E-cash* or *the bank updates account of the trader.* Each event transmits information from one object to another. For example, the event "*withdraw E-cash*" transmits a signal from the trader to the bank. The sequence of events and the objects exchanging events can be shown in an augmented scenario called an event trace diagram

[RBPEL 91]. This diagram shows each object as a vertical line and each event as a horizontal arrow from the sender object to the receiver object. Time increases from top to bottom, but the spacing is irrelevant. In the following section, three scenarios are shown through event trace diagrams: *a trader withdraws E-cash from the bank, a buyer agent negotiation with a seller agent*, and *comparison of two seller agents.*

4.2.3.1 A Trader Withdrawal of E-cash from the Bank

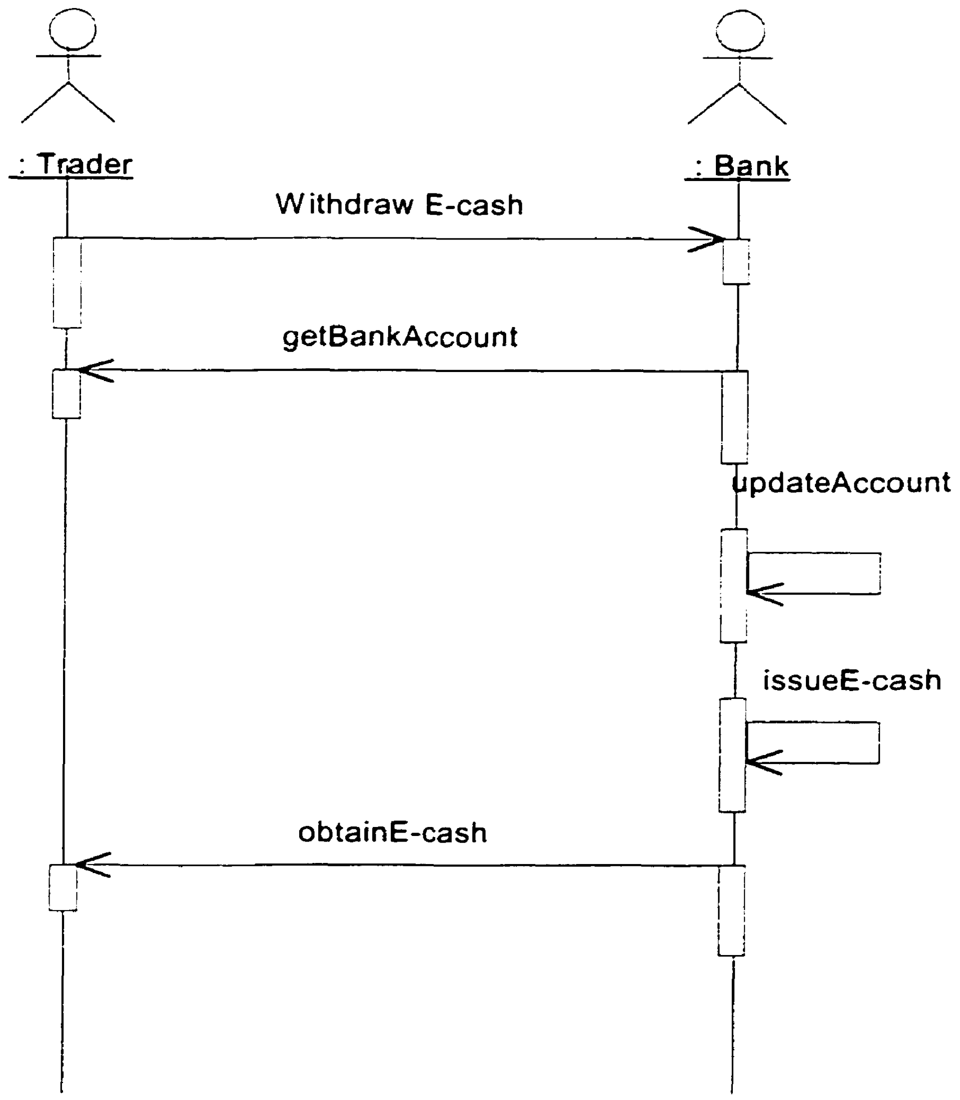


Figure 4-1 Event trace for a trader withdrawal of E-cash from the bank

4.2.3.2 A buyer agent negotiation with a seller agent

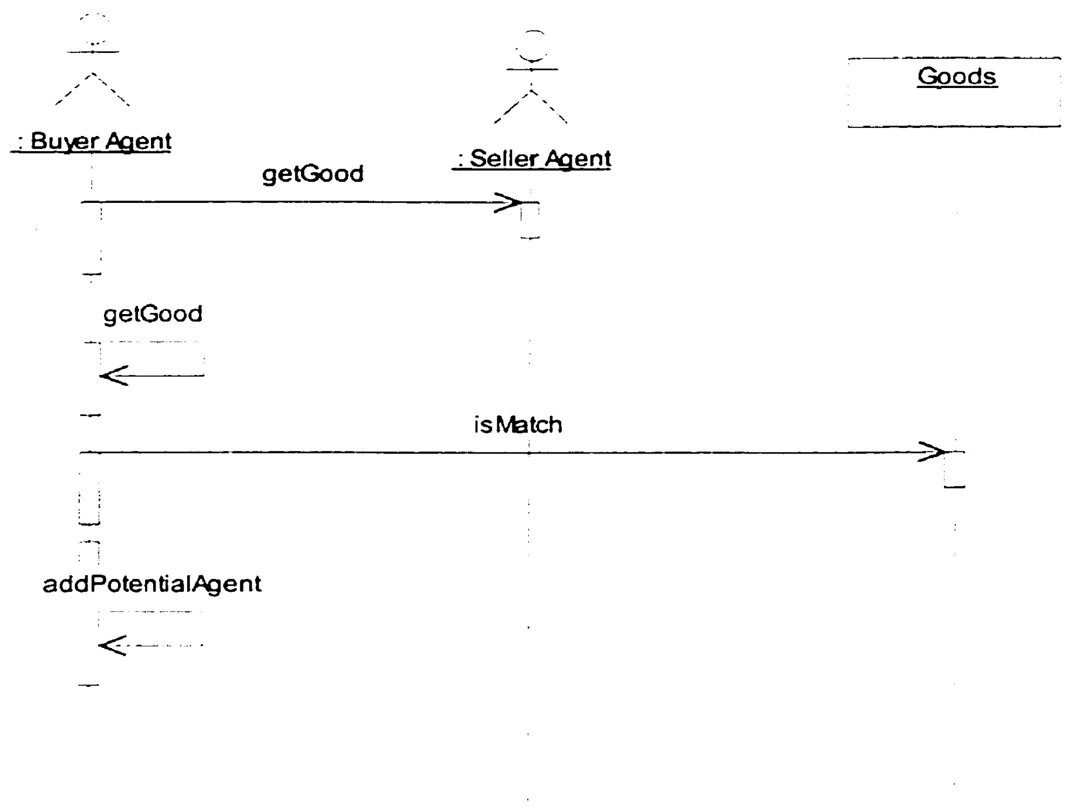


Figure 4-1 Event trace for a buyer agent negotiation with a seller agent

4.2.3.3 Comparison of two seller agents

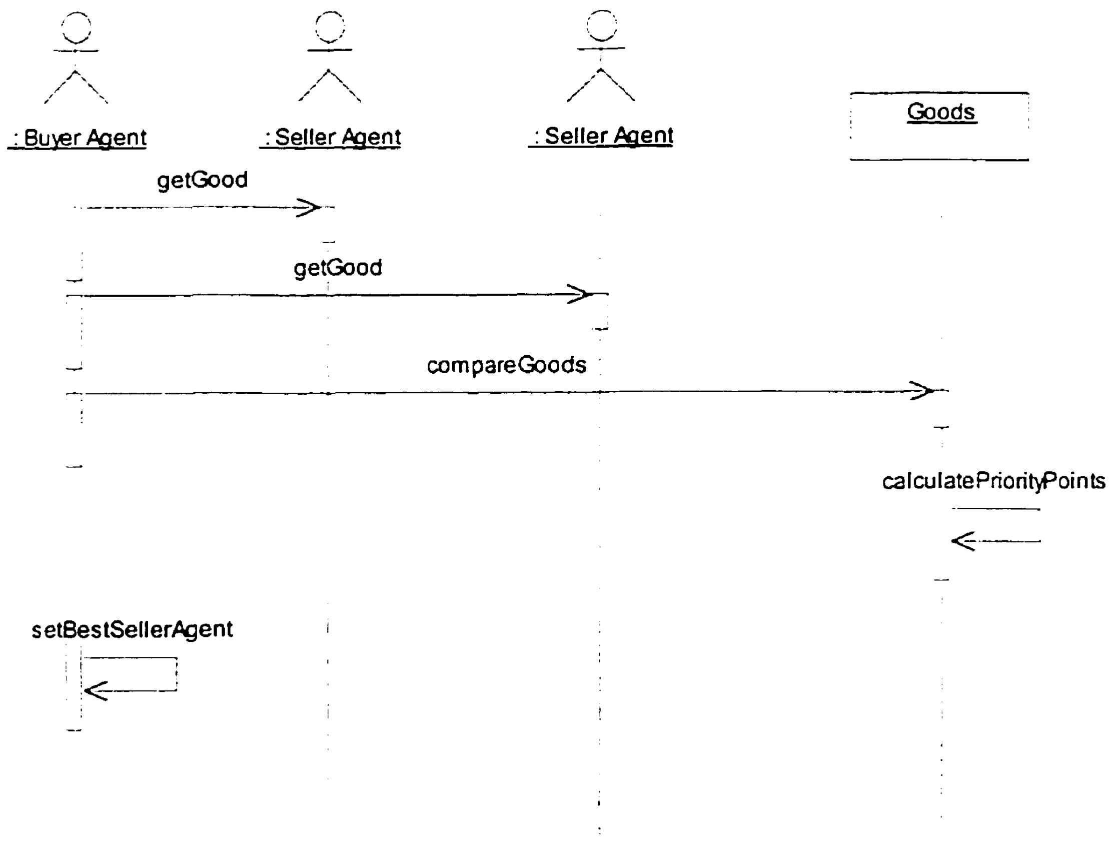


Figure 4-1 Event trace for comparison of two seller agents

4.3 Human-Interaction Objects

The human-interaction objects provide interfaces between business-related objects and the end-users. In YAVO we have the following interfaces:

- Trader login
- Display agent marketplaces information
- Trader creates and sends agents
- Trader views agents status

- Trader views and responds messages
- Trader logout
- Administrator sets up trader account
- Administrator creates bank
- Administrator creates agent marketplaces

4.4 Database objects

The database objects provide interfaces between business-related objects and databases.

In YAVO we need the following database objects:

- Trader accounts and password information
- Agent marketplaces information
- Bank information
- Mobile agents information
- Messages from all mobile agents

Chapter 5

5 YAVO Implementation

5.1 Web Application Topology

Web applications usually include Web clients (such as Web browsers), Web servers. existing application. data from external non-Web services, and standard Internet protocols. Figure 5-1 illustrates the major elements of the YAVO Web application. Web server and external services (i.e. database) are logical tiers capable of running on the same physical machine or different machines. The front-end and business logic parts of a Web application are run on separate machines.

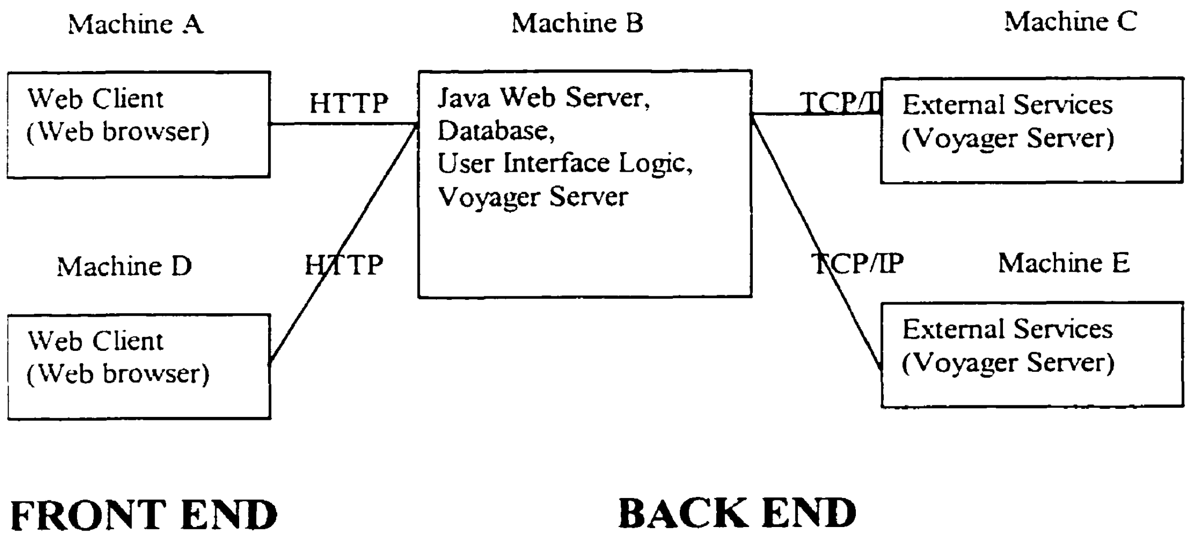

Machine A
Machine B
Machine C
Web Client (Web browser)
HTTP
Java Web Server, Database, User Interface Logic, Voyager Server
TCP/IP
External Services (Voyager Server)
Machine D
HTTP
TCP/IP
Machine E
Web Client (Web browser)
External Services (Voyager Server)
FRONT END
BACK END

Figure 5-1 Infrastructure of YAVO

Web clients communicate with Web Servers typically using HTTP to access business logic and data. The primary role of the client is to accept and validate user input, and present results received from the Web server to the user. The business logic of the YAVO system runs on the back end (servers), not on the front end (clients). This infrastructure provides the following advantages:

- A broader range of client devices, such as PCs and network computers, can be supported since the dependency on client capabilities is reduced.
- The Web server integrates access to resources (databases, etc.), which simplifies application design, improves scalability, and provides greater security of the resources.
- Business logic running on the server is easier to protect, upgrade, and maintain.
- Business logic running on the server allows users' application environments to be centrally managed and restored on different client machines.
- The client part of the Web application is small and downloads quickly.
- YAVO application is based upon HTTP and HTML, so that any browser can run it.

The YAVO Web application is a series of interactions between users and particular Web sites. The entire Web interaction process begins with a single page displayed in the browser. The user clicks on a button or link on the page causing a request to be sent to the Java Web Server. The request is processed on the Web application server and a new page is sent back to Web browser showing the results of the request and presenting buttons or links for the next request. Thus the YAVO application consists of a set of processing steps or interactions. Each interaction gets a request generated from a page and each interaction must produce a response in the form of a Web page that will serve as the input for subsequent interactions. Java Servlets technology is used to process the HTTP request, and then, based on the results of the business logic, generates a dynamic Web page.

5.1.1 Java Servlets

5.1.1.1 What are Java Servlets?

One of the most important technologies used in YAVO is the Java Servlets. What are Servlets? Servlets are Java objects that extend the functionality of information servers, such as HTTP or Web servers [Hunter 98]. Servlets are similar to Common Gateway Interface (CGI) scripts but are vastly different in many ways. A servlet can be thought of as a server-side applet. Servlets are loaded and executed by a Web server on the server as are CGI scripts. The following list describes the basic flow when using Servlets:

1. The client (most likely a web browser) makes a request via HTTP.
2. The web server receives the request and forwards it to the servlet. If the servlet has not yet been loaded, the web server will load it into the Java virtual machine and execute it.
3. The servlet will receive the HTTP request and perform some type of process.
4. The servlet will return a response back to the web server.

The web server will forward the response to the client.

5.1.1.2 Why use Servlets?

In their most basic form, servlets are a great replacement for CGI scripts. CGI scripts are typically written in Perl or C and are usually tied to a particular server platform. Since servlets are written in Java, they are platform independent. Java's promise of write once, run anywhere can now be realized on the server as well. Servlets have other distinct advantages over CGI scripts.

- Servlets are persistent. They are loaded only once by the web server and can maintain services (such as a database connection) between requests. CGI scripts, on the other hand, are transient. Each time a request is made to a CGI script, it must be loaded and executed by the web server. When the CGI script is complete, it is removed from memory and the results are returned to the client. All program initialization (such as connecting to a database) must be repeated each time that a CGI script is used.
- Servlets are fast. Since servlets only need to be loaded once, they offer much better performance over CGI.
- Servlets are platform independent. As mentioned before, servlets are written in Java, which inherently brings platform independence to the development effort.
- Servlets are extensible. Since servlets are written in Java, this brings all of the other benefits of Java to servlets. Java is a robust, object-oriented programming language, which easily can be extended to suit system needs.
- Servlets are secure. The only way to invoke a servlet from the outside world is through a web server. This brings a high level of security, especially if your web server is protected behind a firewall. A firewall is a device (usually a computer running a specially written or modified operating system) that isolates an organization's internal network from the Internet at large, allowing specific connections to pass and blocking others [GS97].

Servlets are powerful. They utilize the full power of Java, such as networking and URL access, multithreading, image manipulation, data compression, database connectivity, remote method invocation (RMI), and object serialization.

5.1.1.3 Application of Servlets

The application of servlets is explained below:

- Allowing collaboration between people. A servlet can handle multiple requests concurrently, and can synchronize requests. This allows servlets to support systems such as on-line conferencing.
- Forwarding requests. Servlets can forward requests to other servers and servlets. Thus, servlets can be used to balance load among several servers that mirror the same content, and to partition a single logical service over several servers, according to task type or organizational boundaries.

5.1.1.4 Java Servlets in YAVO

The following are the eleven servlets programs in the YAVO system.

- AgentServlet

  Program AgentServlet displays all agents' information of a trader on the Web.

- BankServlet

  Program BankServlet displays bank information on the Web. Only the system administrator can access this information page.

- CreateAgent1

  Servlet CreateAgent1 is the first step in creating a mobile agent. In response to a trader's request to create an agent, the servlet posts a Web page that guides the trader to input agent information such as starting price, final price, goods type, lifetime, and agent type. After the trader submits the agent information, the servlet passes all the information to servlet CreateAgent2.

- CreateAgent2

  Servlet CreateAgent2 is the second step in creating an agent. Based on the information from CreateAgent1, this servlet posts a dynamic Web page whose content is different for a buyer and a seller agent (a seller agent is not configured to specify the priority of features). The trader enters detail information of the interested goods, location of an agent marketplace, and the priority of additional features (this only applies to buyer agent). After the submission of the trader, the information will pass to servlet CreateAgent3.

- CreateAgent3

  This servlet program is the last step in creating an agent. Based on the mobile agent information from servlet CreateAgent2, CreateAgent3 generates a mobile agent, obtains a reference to the agent marketplace that is specified in mobile agent information, and sends the agent to work in the marketplace. In addition, CreateAgent3 inserts the created agent information into an agent database on the machine running the Java Web Server and generates a confirmation page that lists all the information entered by the trader.

- Login

  Servlet Login authenticates a trader who tries to login on the YAVO system. Only authorized traders can use the YAVO system.

- Logout

  Logout servlet logs out the trader from YAVO.

- Market

  Market servlet displays market information on the Web. Only the system administrator can access this information page.

- MarketServlet

  MarketServlet servlet creates an agent marketplace on a user-specified location, and updates agent marketplace database. Only the system administrator can create agent marketplaces.

- MessageServlet

  MessageServlet servlet displays all the messages of a trader on the Web.

- TraderServlet

  TraderServlet servlet displays all the traders' information on the Web. Only the system administrator can access this information page.

5.1.2 Java Web Server

Sun's Java Web Server was chosen to be the Web server for YAVO because the product has been free for evaluation, easy-to-use, secure, and platform-independent. In addition, the Java Web Server supports Java Servlets technology, thus enabling server-side Java applications.

5.1.3 Voyager Functionality Examples

5.1.3.1 Using Interfaces for Distributed Computing

The Java programming language supports a feature called an interface. An interface contains no code. Instead, it defines a set of method signatures that must be defined by any class that implements the interface. A variable whose type is an interface may refer to any object whose class implements the interface. Voyager leverages this feature to simplify distributed computing [Voyager 99]. In YAVO, an agent marketplace is a remote object and it is represented by a special proxy object that implements the same interfaces as its remote counterpart. A variable whose type is an interface may refer to a remote object via a proxy, because both the remote object and its proxy implement the same interfaces. Here is an example of an interface:

```
public interface IAgentMarketplace
 {
        int getMaxNumOfAgents();
        int getServiceCharge();
    int getDeposit();
        String getGoods();
        String getLocation();
        String getNameOfM();

        Hashtable getBuyerAgents();
        Hashtable getSellerAgents();
        String toString();
        String toWebString();
        String toTableString(int rowNumber);
        boolean checkNumber(String s);
        Vector search(IRentalAgent agent);
        void addSeller(int id, IRentalAgent agent);
        void addBuyer(int id, IRentalAgent agent);
        IRentalAgent removeSellerAgent(int id);
        IRentalAgent removeBuyerAgent(int id);
}
```

5.1.3.2 Creating Remote Objects

To create an object such as the agent marketplace at a specified location, "Factory.create()" method is used. This method returns a proxy to the newly created object and creates the proxy class dynamically if it does not already exist [Voyager 99].

Here is the example of using the method to create an agent marketplace at machine dyna132-28, port 8000:

```
        Object[] arg_market = {name,maxNumOfAgents,serviceCharge,deposit,goods,
market};
        IAgentMarketplace market = (IAgentMarketplace) Factory.create(
marketClass,arg_market,"//dyna132-28:8000" );
```

5.1.3.3 Using Name Services

Naming services are used to bind names to objects for later lookup [Voyager 99]. The following code continues from the previous example. It binds a created agent marketplace to the name "Rent8000".

```
        Namespace.bind("//dyna132-28:8000/Rent8000",market);
```

The following example uses "Namespace.lookup()" to obtain a proxy for the agent marketplace object. Mobile agents use this function in order to move to a remote agent marketplace.

```
        IAgentmarketplace market = (IAgentmarketplace) Namespace.lookup
("//dyna132-28:8000/Rent8000");
```

5.1.3.4 Sending Messages

In YAVO, a mobile agent communicates with the agent marketplace locally. Local messages are much faster than remote messages. In the following example, an agent sends a message to an agent marketplace object, after it arrives at the marketplace. (In the following code, "id" stands for the Agent ID, "this" is a reference to an agent.)

```
   if (actiontype.equals("buy")){
        market.addBuyer(id,this);
    }
  else{
        market.addSeller(id,this);
    }
```

5.1.3.5 Creating Mobile Agents by Dynamic Aggregation

An object's behavior may be extended at runtime. Traditional mechanisms of inheritance and polymorphism do not help to solve these problems, so Voyager introduces a new feature called dynamic aggregation that attaches new code and data to an object at runtime. To make an object become a mobile autonomous agent, "Agent.of()" is used to obtain an object's agent facet and then uses the methods defined in the interface "IAgent". Facet is a secondary object that attaches to a primary object at runtime. A primary object and its facets form an aggregate that is typically persisted, moved, and garbage collected, as a single unit [Voyager 99]. For example, in the YAVO system, a seller agent object can move itself to a remote agent marketplace object "market" and execute its function upon arrival (In the following code, "this" is the reference to an agent object, "atMarket" refers to a callback function that the agent will execute upon arrival):

```
Agent.of(this).moveTo(market, "atMarket");
```

5.1.3.6 Timers

Voyager has timer services such as the Stopwatch and Timer classes. A Stopwatch object clocks time intervals and prints time measurement statistics such as the cumulative lap time, average lap time, and last lap time. The lap stands for every start/stop cycle. A Timer object generates timer events and adds listeners to timer. The Timer class acts like an alarm clock. A Timer object sends a TimerEvent to one or more listeners. Upon receiving an event, a listener performs an action [Voyager 99]. In YAVO, the Timer class is used for a mobile agent to update its current price and lifetime daily. The following example constructs a timer, one listener (a mobile agent), sets the timer to generate periodic events, and adds the listener to the timer. The timer object will notify the agent once each day.

```
Timer timer = new Timer();
timer.addTimerListener( new TimerListenerThread((TimerListener)agent) );
//for the sake of testing, timer will wake up agent
// every 20 seconds
timer.alarmEvery( 20000 );
```

After the agent receives a timer event, the method "timerExpired" of the agent will execute automatically:

```
timerExpired(TimerEvent aTimerEvent)
{
...
// Updates lifetime.
// Updates current price.
// Negotiates with other agents.
...
}
```

5.1.4 Database System

5.1.4.1 Introduction

It is hard to find a professional web site today that does not have some database connectivity [Hunter 97]. A Data Base Management System (DBMS) computer program manages a permanent self-descriptive repository of data. There are many reasons why a DBMS is very important:

- Crash recovery. A database is protected from hardware crashes, disk media failures, and user errors.
- Sharing between users. Multiple users can access the database at the same time.
- Sharing between applications. Multiple application programs can read and write data to the same database.
- Security. Data can be protected against unauthorized read and write access.

5.1.4.2 JDBC

JDBC (Java Database Connectivity) is used in YAVO implementation. JDBC is a set of interfaces and classes designed to perform actions against any database. An individual database system is accessed via a specific JDBC driver. The company Sun packages a free JDBC-ODBC bridge driver with the JDK to allow access to standard ODBC data sources, such as a Microsoft Access database [Hunter 97].

5.1.4.3 YAVO Databases

In the YAVO Web application, there are five databases on the machine running the Java Web Server: Agents Database, Bank Database, Marketplace Database, Message Database, and Traders Database. The following section describes the detail design of databases.

- Agents Database

Agents Database stores all the mobile agents. It contains goods-specified tables. For example, the database table *rental-agents* is illustrated in the following table:

Action Type	Text
Goods Type	Text
Starting Price	Number
Final Price	Number
Life Span	Number
Price Change Strategy	Text
Approval	Text
Distance1	Text
Distance2	Text
Number of Bedrooms	Text
Unit Type	Text
Furnishing	Text
Utility	Text
Bus	Text
Cable	Text
Pets	Text
Marketplace	Text
First Priority	Text
Second Priority	Text
Third Priority	Text
Fourth Priority	Text
Username of Trader	Text
Agent Status	Text
Current Price	Text
Agent ID	Text

Table 5-1 Design of the Rental-agents table in Agents Database

- Bank Database

Bank Database contains the information of the bank that is created by the system administrator. It contains a database table *bank* that is illustrated in the following:

Location	Text
E-cash	Number

Table 5-2 Design of Table Bank in Bank Database

- Marketplace Database

Marketplace Database stores the information of all the agent marketplaces that are created by the system administrator. It contains a database table *marketplace* that is shown in the following:

Name	Text
Maximum Number of Agents	Text
Service Charge	Text
Deposit	Text
Goods Type	Text
Location	Text

Table 5-3 Design of Table Marketplace in Marketplace Database

- Message Database

Message Database contains messages from all the agents. It contains a database table *message* that is illustrated in the following:

Message	Text
Agent ID	Text
Trader's Username	Text
Other Agent ID	Text
Other Trader's Username	Text

Table 5-4 Design of Table Message in Message Database

- Traders Database

Traders Database stores the accounts information of the traders that are created by the system administrator. It contains a database table *traders* that is shown in the following:

Username	Text
Password	Text
E-cash	Number

Table 5-5 Design of Traders Message in Traders Database

Chapter 6

6 Application Examples

An example of a rental place is used to demonstrate the YAVO system. Several screenshots are used to illustrate the graphical user interfaces that the administrator and traders will typically encounter while using this system. Not all possible circumstances are covered.

In preparation, the Java Web Server and Voyager Servers are to be started by following the steps below:

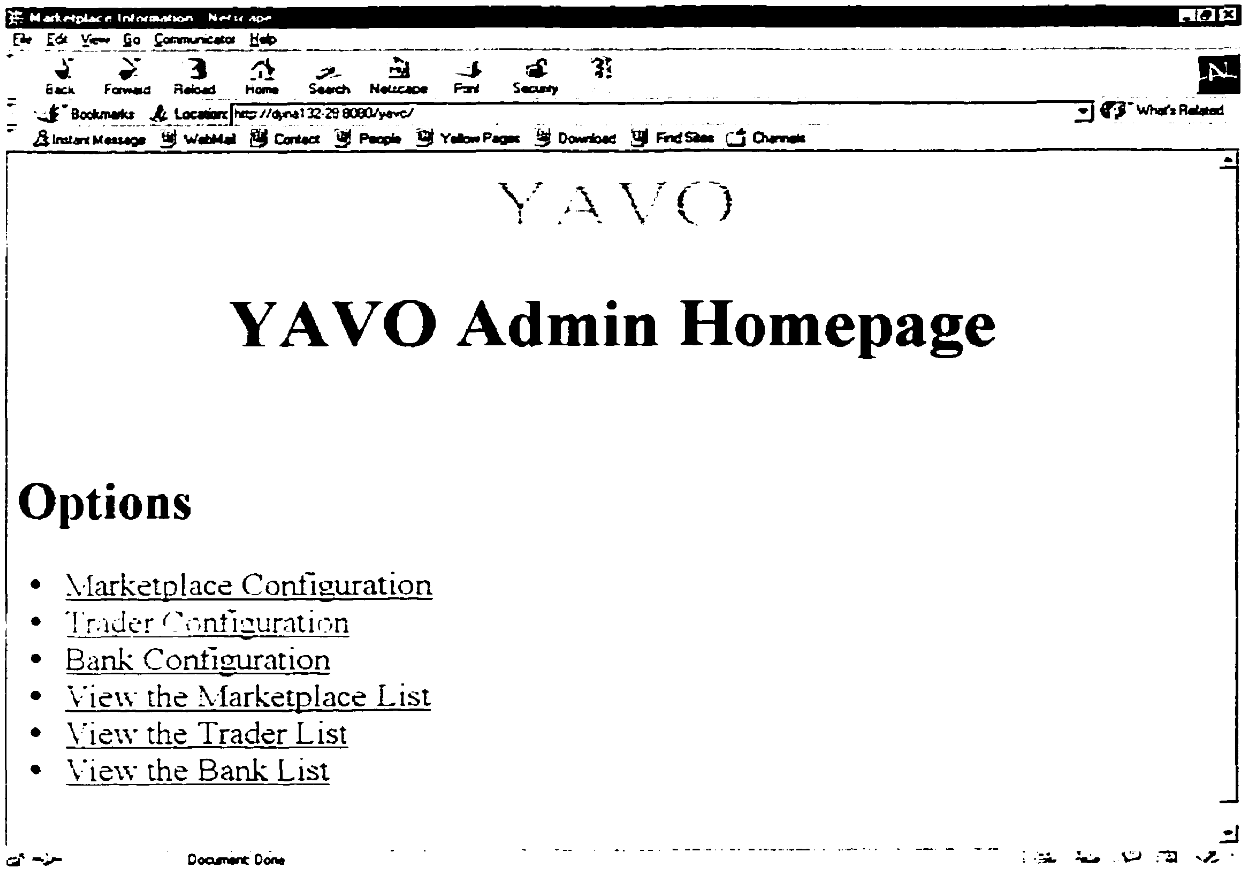
- Open three MS-DOS Prompt windows.
- In the first MS-DOS Prompt window, start Java Web Server by typing "httpd."
- In the second MS-DOS Prompt window, start a Voyager Server at port 8000 by the command "voyager 8000 -verbose".
- In the third MS-DOS Prompt window, start another Voyager Server at port 9000 by the command "voyager 9000 -verbose".

6.1 Administrator

As described in Chapter Four, the system administrator creates all the trader accounts and the agent marketplaces. The following examples illustrate the administrator's home page, the creation of several trader accounts, and the creation of several agent marketplaces.

6.1.1 Administrator Home

The administrator's home page provides several options for different features such as the marketplace configuration, the trader configuration, a view of all the marketplaces, and a view of all the traders.


Marketplace Information - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
Bookmarks Location: http://dyna132-29:8080/yavo/
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document: Done

YAVO

YAVO Admin Homepage

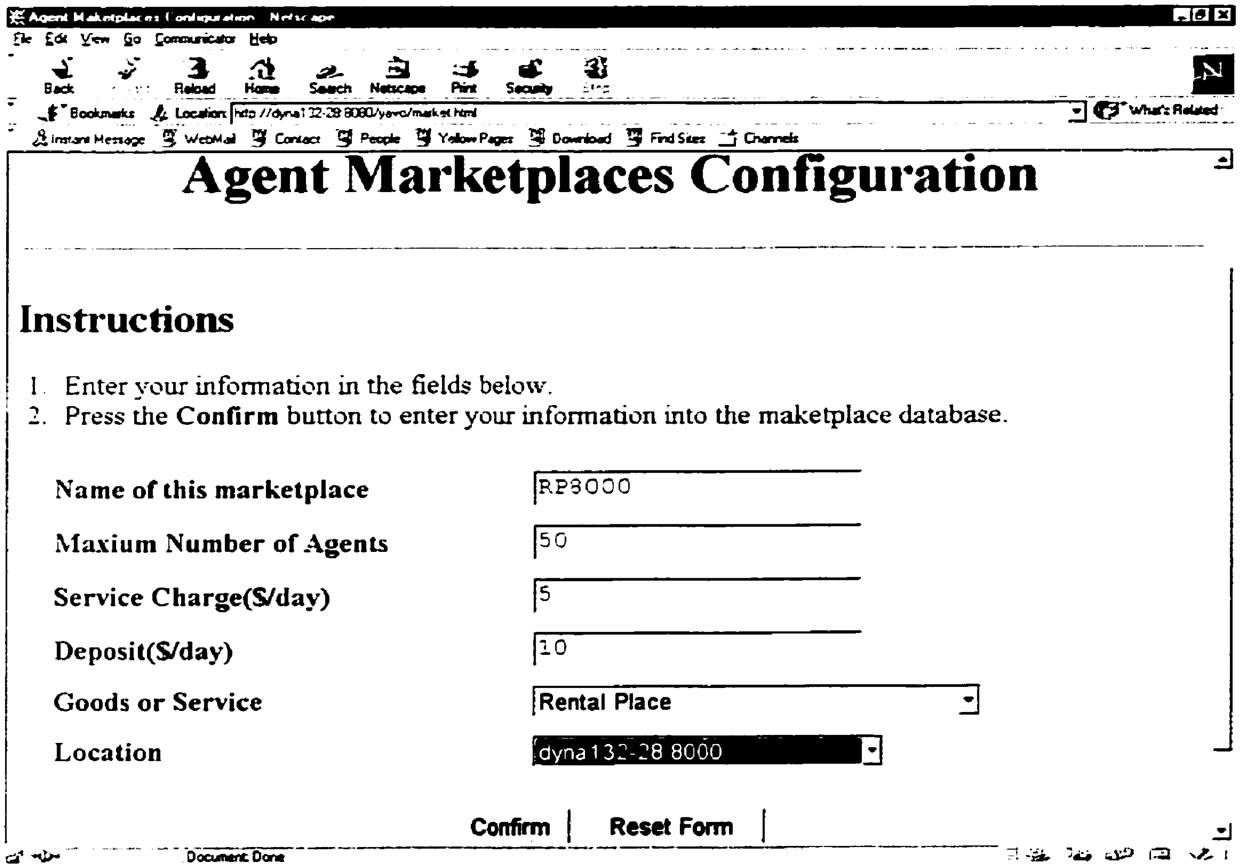
Options

- Marketplace Configuration
- Trader Configuration
- Bank Configuration
- View the Marketplace List
- View the Trader List
- View the Bank List

Figure 6-1 YAVO Administrator Homepage

6.1.2 Creation of Agent Marketplaces

In order to create an agent marketplace, the administrator must specify the name, maximum number of agents, service charge, deposit fee, goods/services, and the location of the marketplace.



Agent Marketplaces Configuration

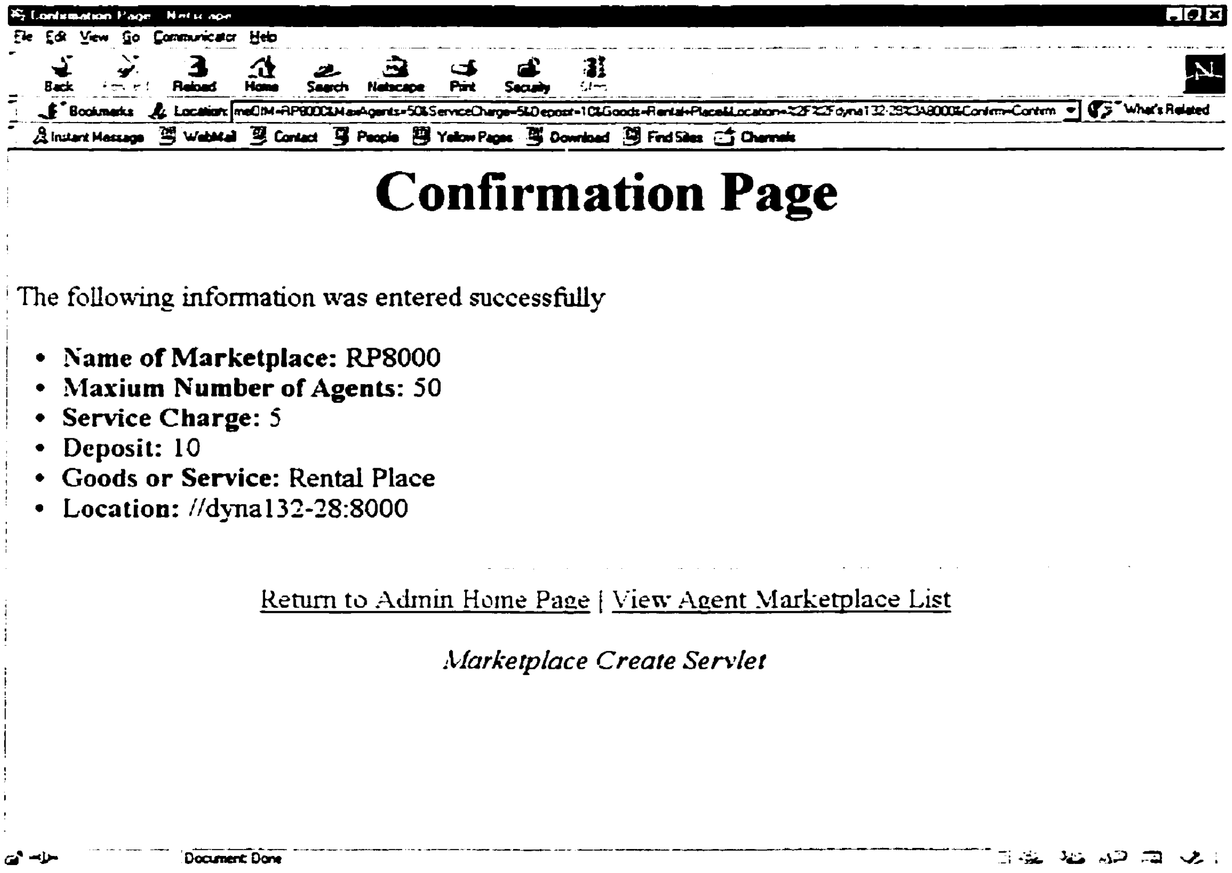
Instructions

1. Enter your information in the fields below.
2. Press the **Confirm** button to enter your information into the maketplace database.

Name of this marketplace	RP8000
Maxium Number of Agents	50
Service Charge($/day)	5
Deposit($/day)	10
Goods or Service	Rental Place
Location	dyna132-28 8000

Figure 6-1 Agent Marketplaces Configuration Page

After the administrator submits the information, a rental place marketplace is created at port 8000 on machine dyna132-28.


File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document Done

Confirmation Page

The following information was entered successfully

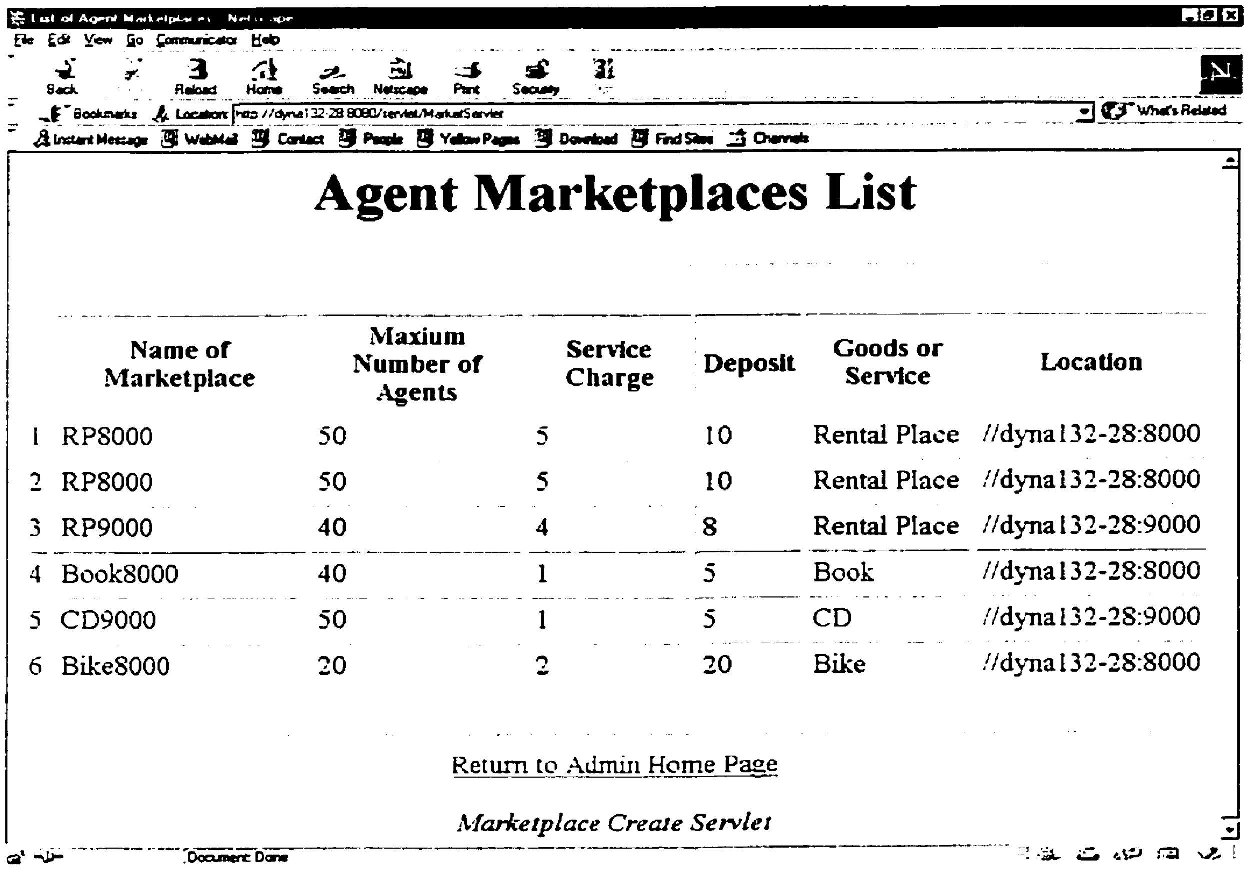
- **Name of Marketplace:** RP8000
- **Maxium Number of Agents:** 50
- **Service Charge:** 5
- **Deposit:** 10
- **Goods or Service:** Rental Place
- **Location:** //dyna132-28:8000

Return to Admin Home Page | View Agent Marketplace List

Marketplace Create Servlet

Figure 6-2 Creation Confirmation of an Agent Marketplace

Similarly, several agent marketplaces are created and shown in Figure 6-4. They are created on different machines or different ports on the same machine.


File Edit View Go Communicator Help
Back Reload Home Search Netscape Print Security
Bookmarks Location: http://dyna132-28:8080/servlet/MarketServlet
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document: Done

Agent Marketplaces List

	Name of Marketplace	Maxium Number of Agents	Service Charge	Deposit	Goods or Service	Location
1	RP8000	50	5	10	Rental Place	//dyna132-28:8000
2	RP8000	50	5	10	Rental Place	//dyna132-28:8000
3	RP9000	40	4	8	Rental Place	//dyna132-28:9000
4	Book8000	40	1	5	Book	//dyna132-28:8000
5	CD9000	50	1	5	CD	//dyna132-28:9000
6	Bike8000	20	2	20	Bike	//dyna132-28:8000

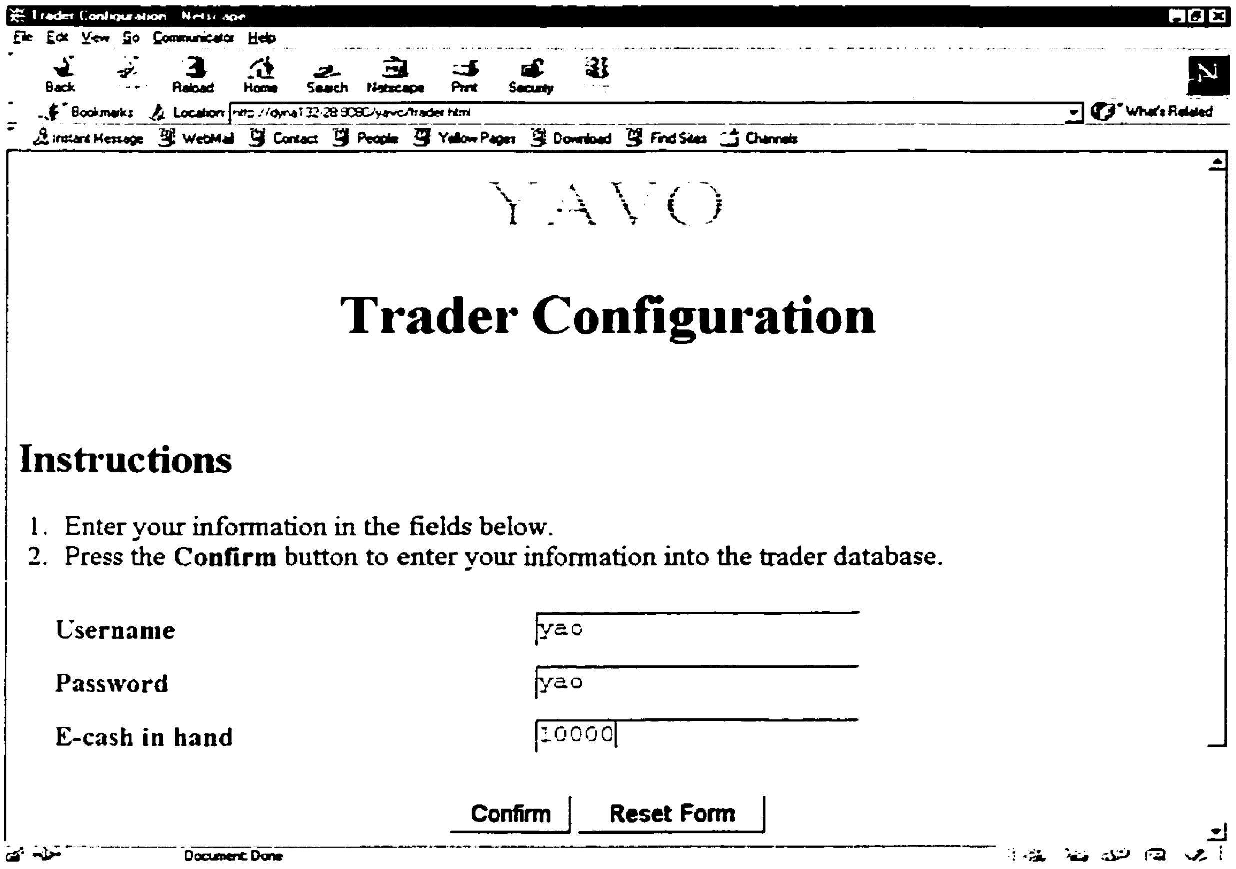
Return to Admin Home Page

Marketplace Create Servlet

Figure 6-3 Agent Marketplaces List Page

6.1.3 Creation of Traders

In order to create trader accounts in the YAVO system, the administrator must specify a username, password, and an amount of E-cash.



YAVO

Trader Configuration

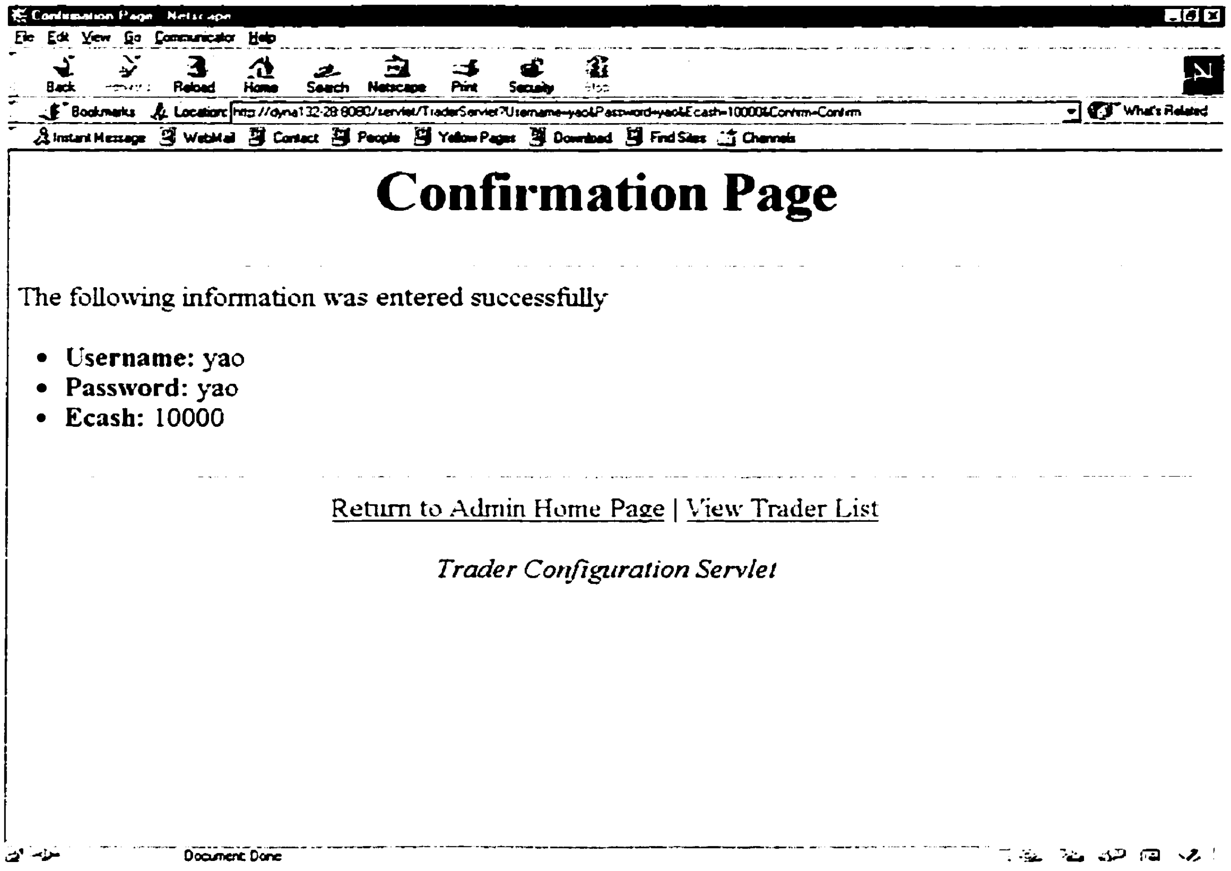
Instructions

1. Enter your information in the fields below.
2. Press the **Confirm** button to enter your information into the trader database.

Username	yao
Password	yao
E-cash in hand	10000

Figure 6-1 Trader Configuration Page

After the administrator submits the trader's account information, the confirmation page is shown in Figure 6-6.


File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://dyna132-28:8080/servlet/TraderServlet?Username=yao&Password=yao&Ecash=10000&Confirm=Confirm
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document: Done

Confirmation Page

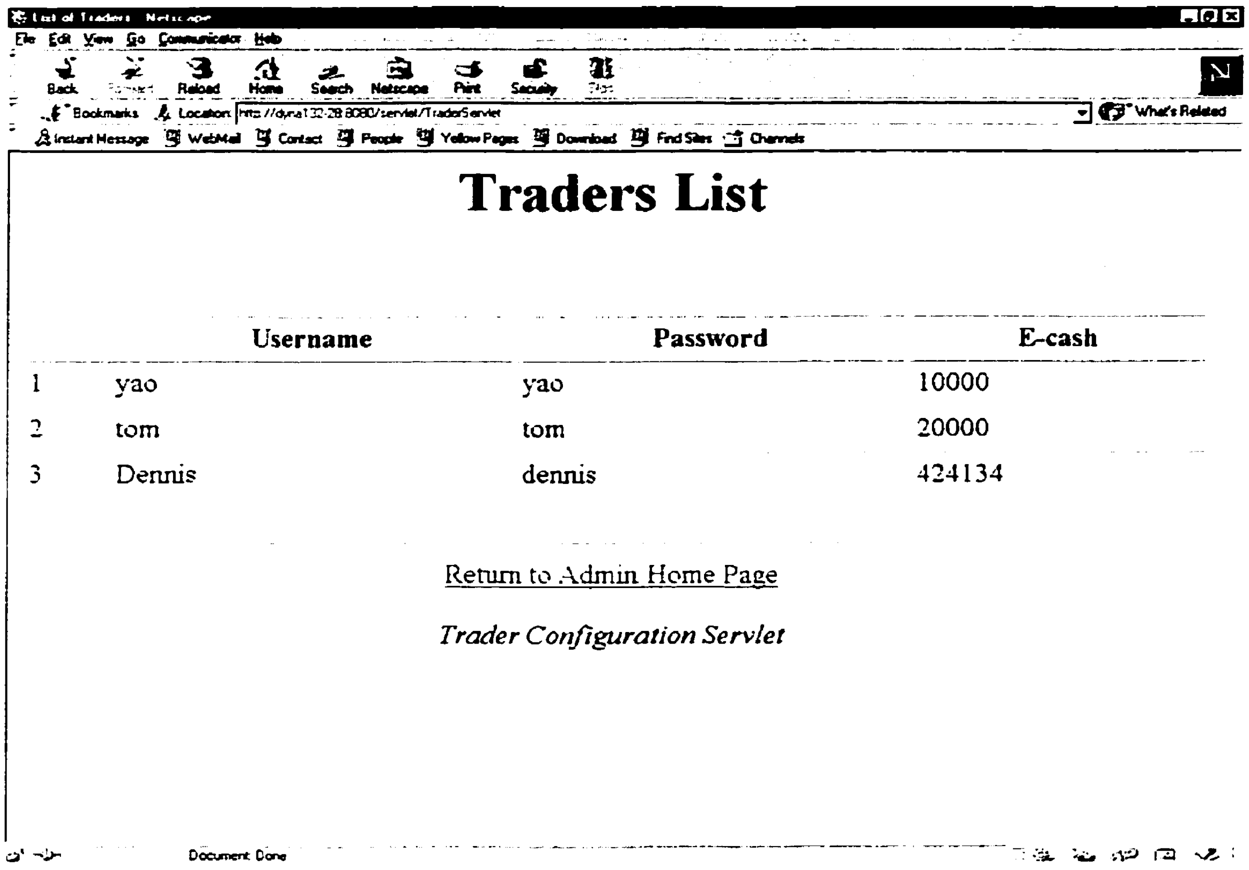
The following information was entered successfully

- **Username:** yao
- **Password:** yao
- **Ecash:** 10000

Return to Admin Home Page | View Trader List

Trader Configuration Servlet

Figure 6-2 Creation Confirmation of a Trader



List of Traders - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://dyna132-28:8080/servlet/TraderServlet
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document: Done

Traders List

	Username	Password	E-cash
1	yao	yao	10000
2	tom	tom	20000
3	Dennis	dennis	424134

Return to Admin Home Page

Trader Configuration Servlet

Figure 6-3 List of Traders Page

6.2 Trader

Once the user has logged in, the system displays the main user interface screen. For example, figure 6-2 shows the administrator's main interface screen, which appears different from other user interfaces. In this section we illustrate the creation of several mobile agents, their runtime status, and any trading messages for their traders.

6.2.1 Trader Login

Figure 6-8 shows the trader's login page.

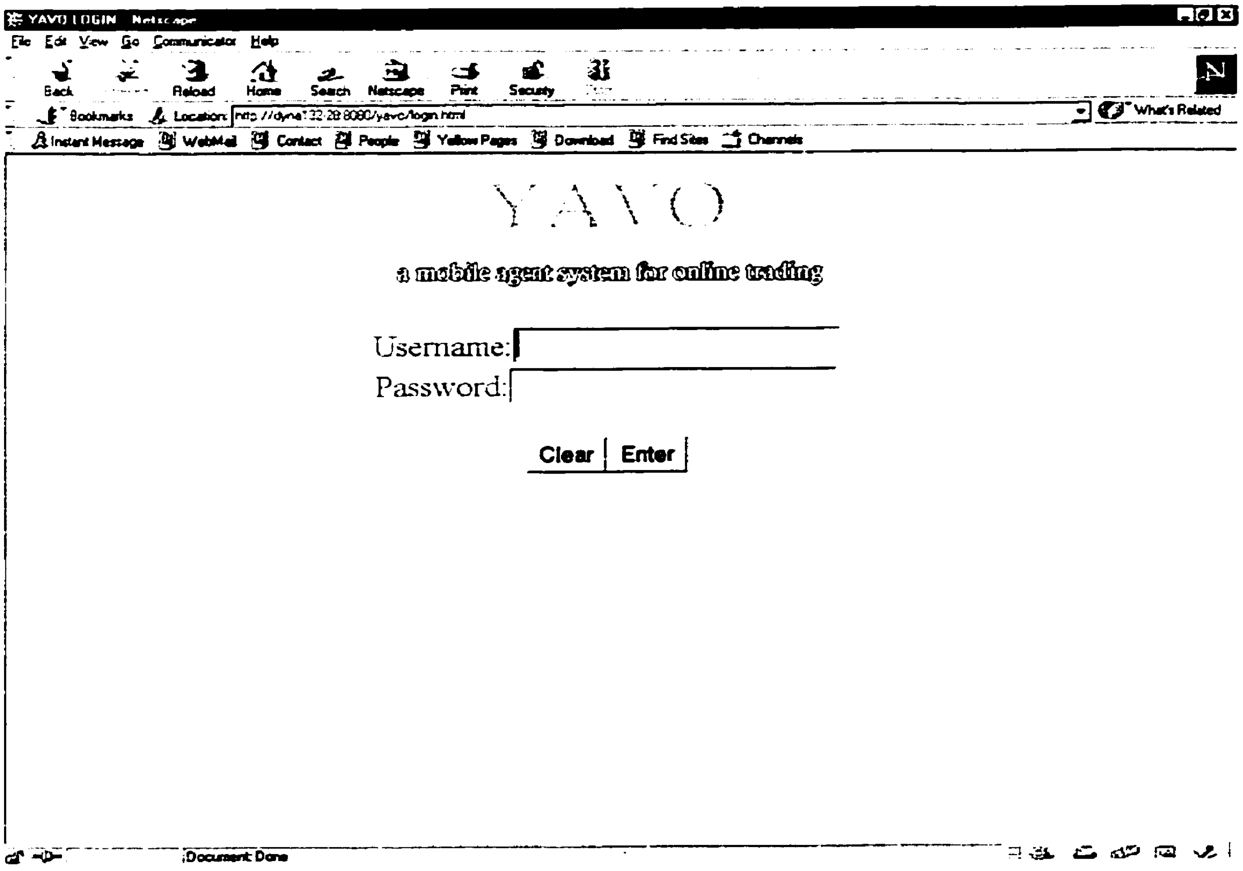
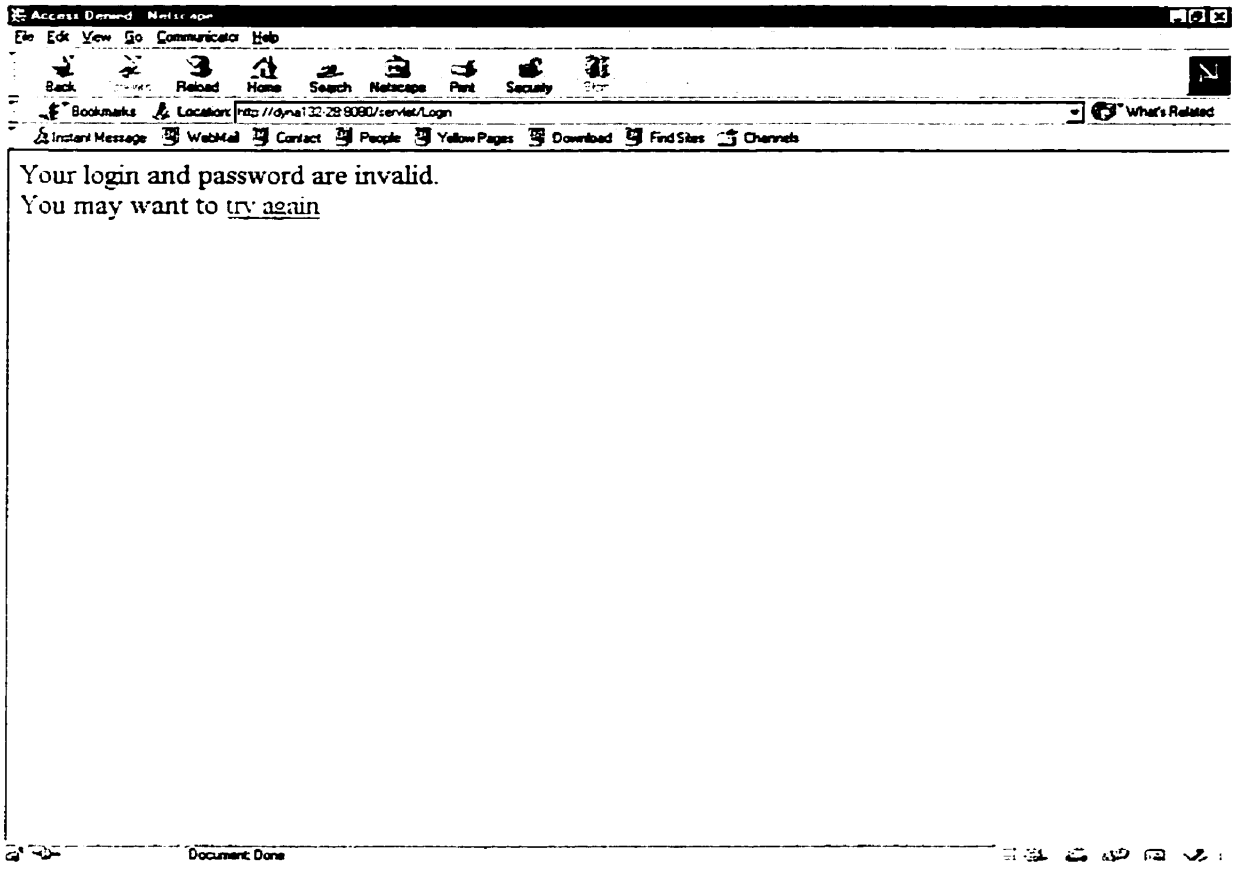


Figure 6-1 Trader Login Page

If the login information is invalid, the following page appears.

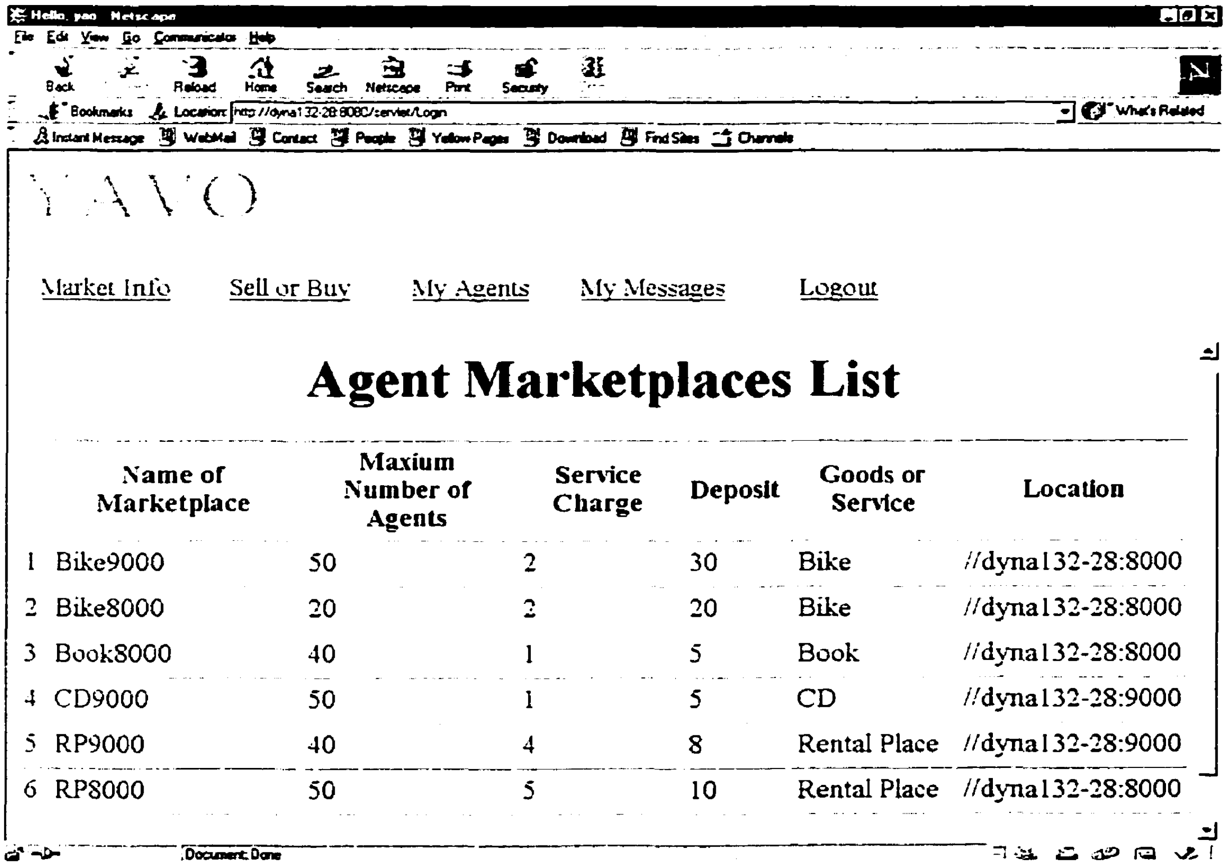

Access Denied - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://dyna132-29:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document: Done

Your login and password are invalid.
You may want to try again

Figure 6-2 Invalid Trader Account

6.2.2 Trader Home (Marketplaces Information)

In this example, the trader Yao has logged into the YAVO system, and then any agent marketplace information is provided. This information helps the trader to choose a marketplace.


Hello, yao - Netscape
File Edit View Go Communicator Help
Back Reload Home Search Netscape Print Security
Bookmarks Location: http://dyna132-28:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Document: Done

YAVO

Market Info Sell or Buy My Agents My Messages Logout

Agent Marketplaces List

	Name of Marketplace	Maxium Number of Agents	Service Charge	Deposit	Goods or Service	Location
1	Bike9000	50	2	30	Bike	//dyna132-28:8000
2	Bike8000	20	2	20	Bike	//dyna132-28:8000
3	Book8000	40	1	5	Book	//dyna132-28:8000
4	CD9000	50	1	5	CD	//dyna132-28:9000
5	RP9000	40	4	8	Rental Place	//dyna132-28:9000
6	RP8000	50	5	10	Rental Place	//dyna132-28:8000

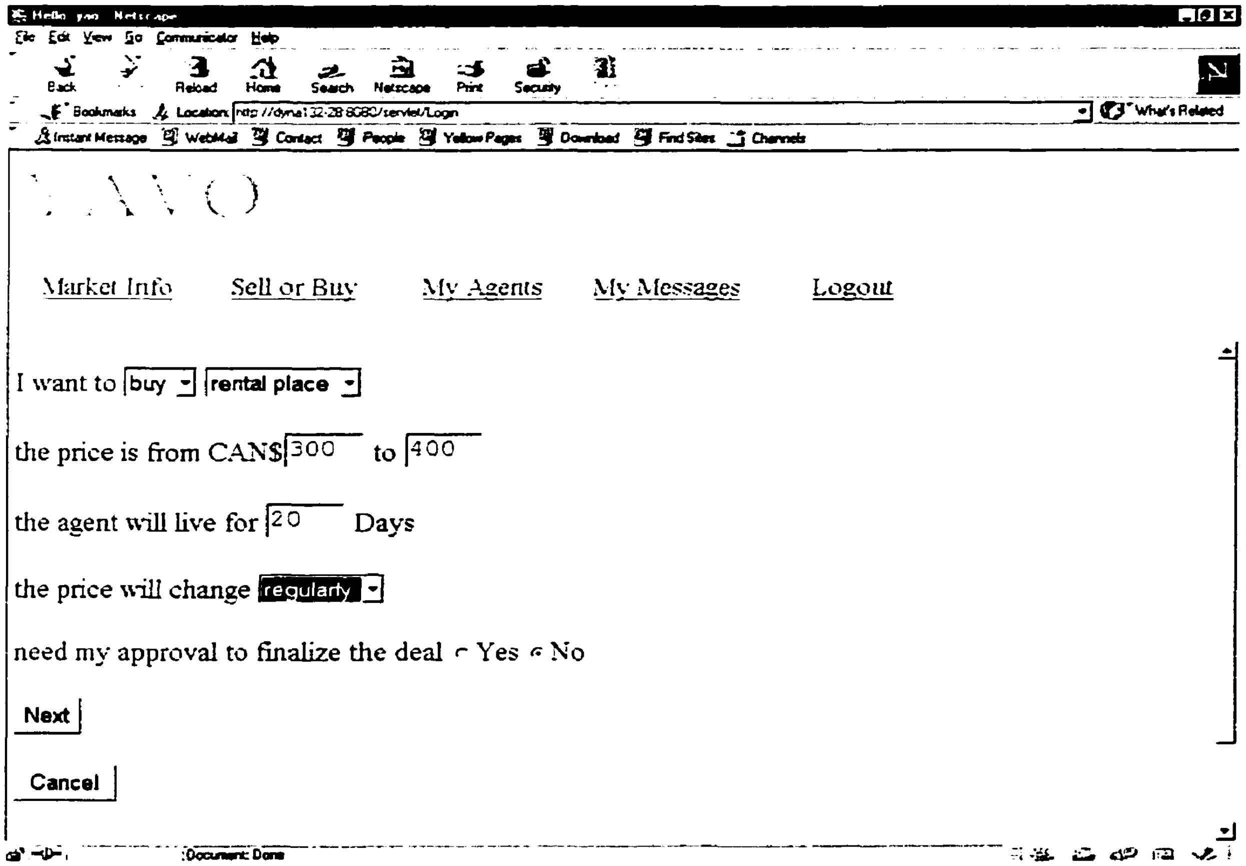
Figure 6-1 Trader Homepage (Agent Marketplaces' Information)

6.2.3 Creating Mobile Agents for Trading

In the following section, several mobile agents are created to trade residences. The trader Yao creates and sends two buyer agents, each to one of two marketplaces. Also, the trader Tom creates and sends two seller agents, each to one of two marketplaces.

6.2.3.1 Create Buyer Agents

Figure 6-11 shows the first step for Yao to create a buyer agent.


Hello yao - Netscape
File Edit View Go Communicator Help
Back Reload Home Search Netscape Print Security
Bookmarks Location
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
YAO
Market Info
Sell or Buy
My Agents
My Messages
Logout
Next
Cancel
Document: Done

I want to buy rental place

the price is from CAN$300 to 400

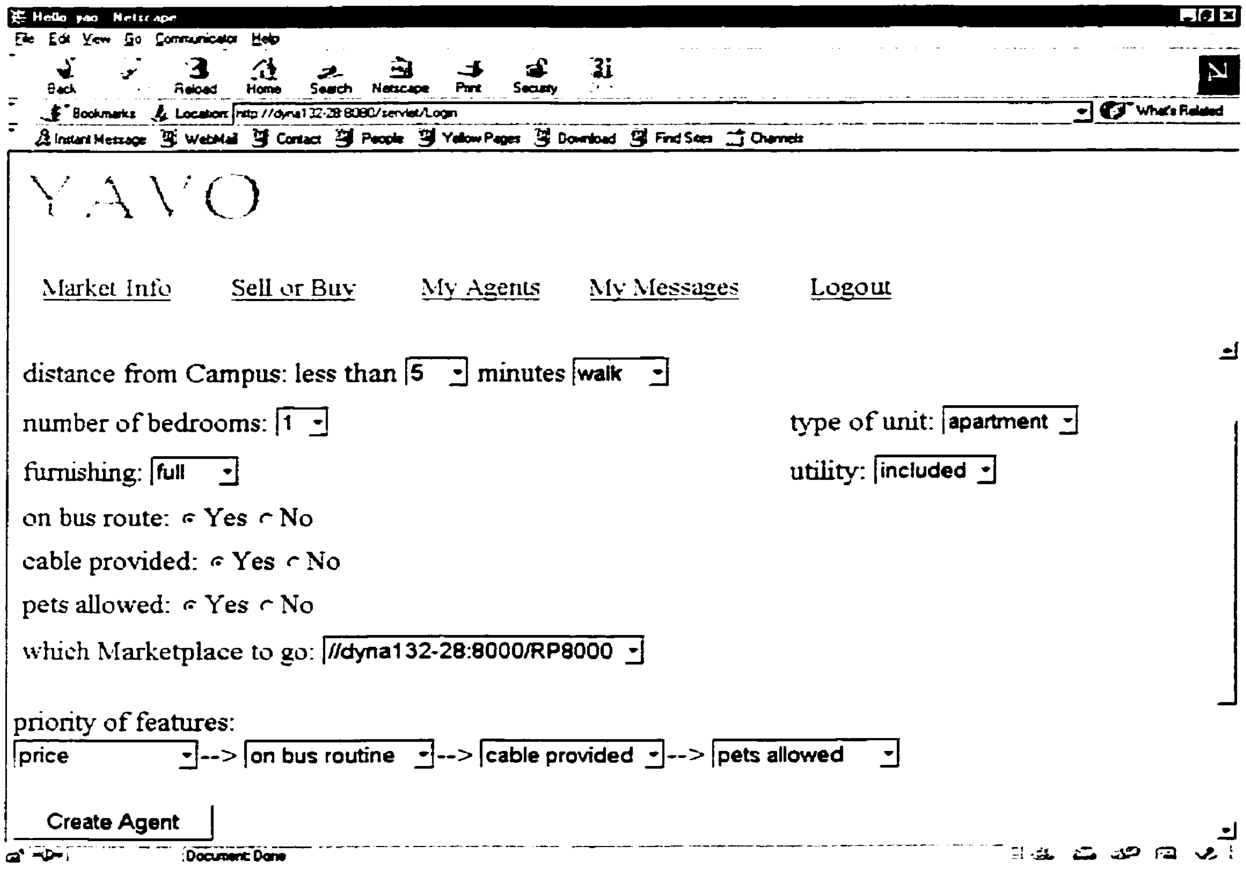
the agent will live for 20 Days

the price will change regularly

need my approval to finalize the deal Yes [✓ No]

Figure 6-1 First Step to Create a Buyer Agent

Figure 6-12 shows the second step for Yao to create a buyer agent.


Hello yao Netscape
File Edit View Go Communicator Help
Back Reload Home Search Netscape Print Security
Bookmarks Location: http://dyna132-28:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
YAVO
Market Info Sell or Buy My Agents My Messages Logout
Create Agent
Document: Done

distance from Campus: less than 5 minutes walk

number of bedrooms: 1

type of unit: apartment

furnishing: full

utility: included

on bus route: [✓ Yes] No

cable provided: [✓ Yes] No

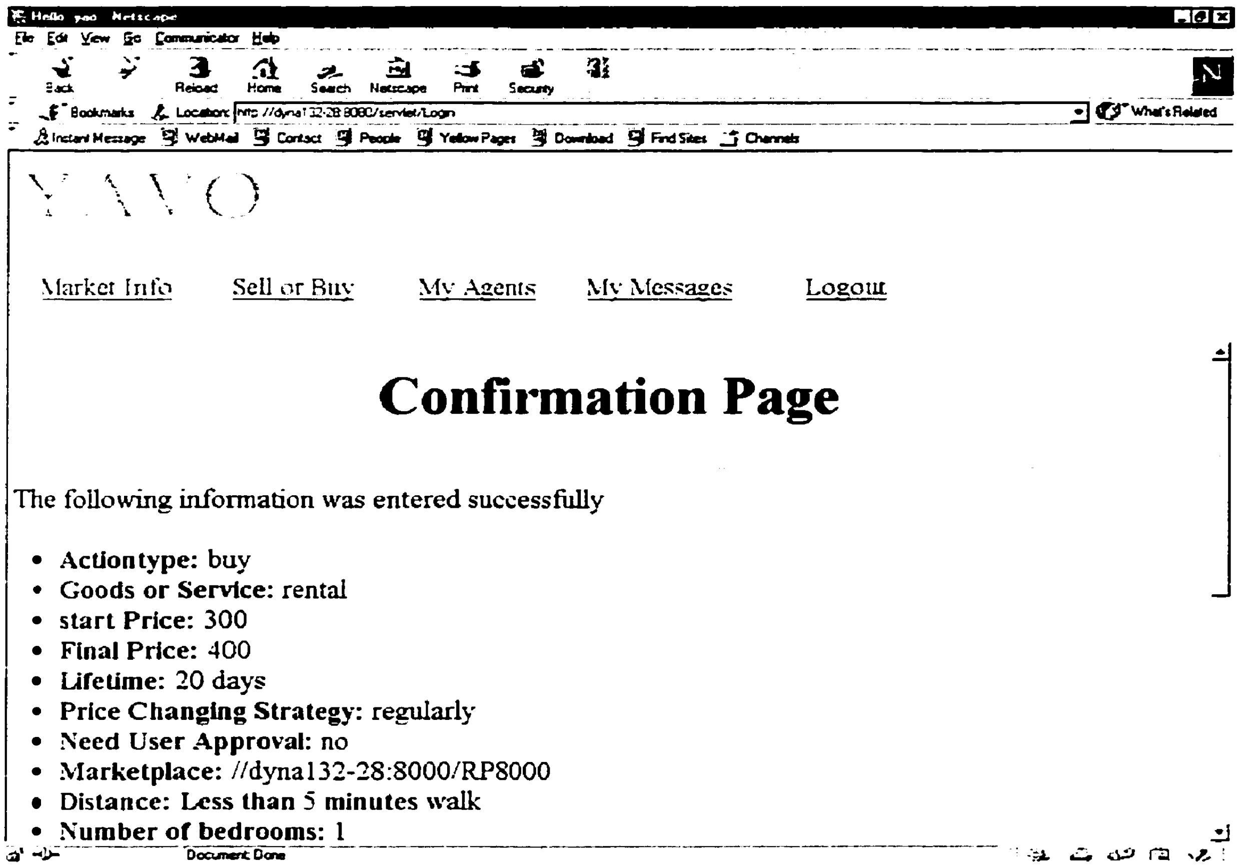
pets allowed: [✓ Yes] No

which Marketplace to go: //dyna132-28:8000/RP8000

priority of features:
price --> on bus routine --> cable provided --> pets allowed

Figure 6-2 Second Step to Create a Buyer Agent

After a buyer agent has been created, the following page appears to confirm the creation.


Hello yao Netscape
File Edit View Go Communicator Help
Back Reload Home Search Netscape Print Security
Bookmarks Location: http://dyna132-28:8080/servlet/Login What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
YAYO
Market Info Sell or Buy My Agents My Messages Logout
Document: Done

Confirmation Page

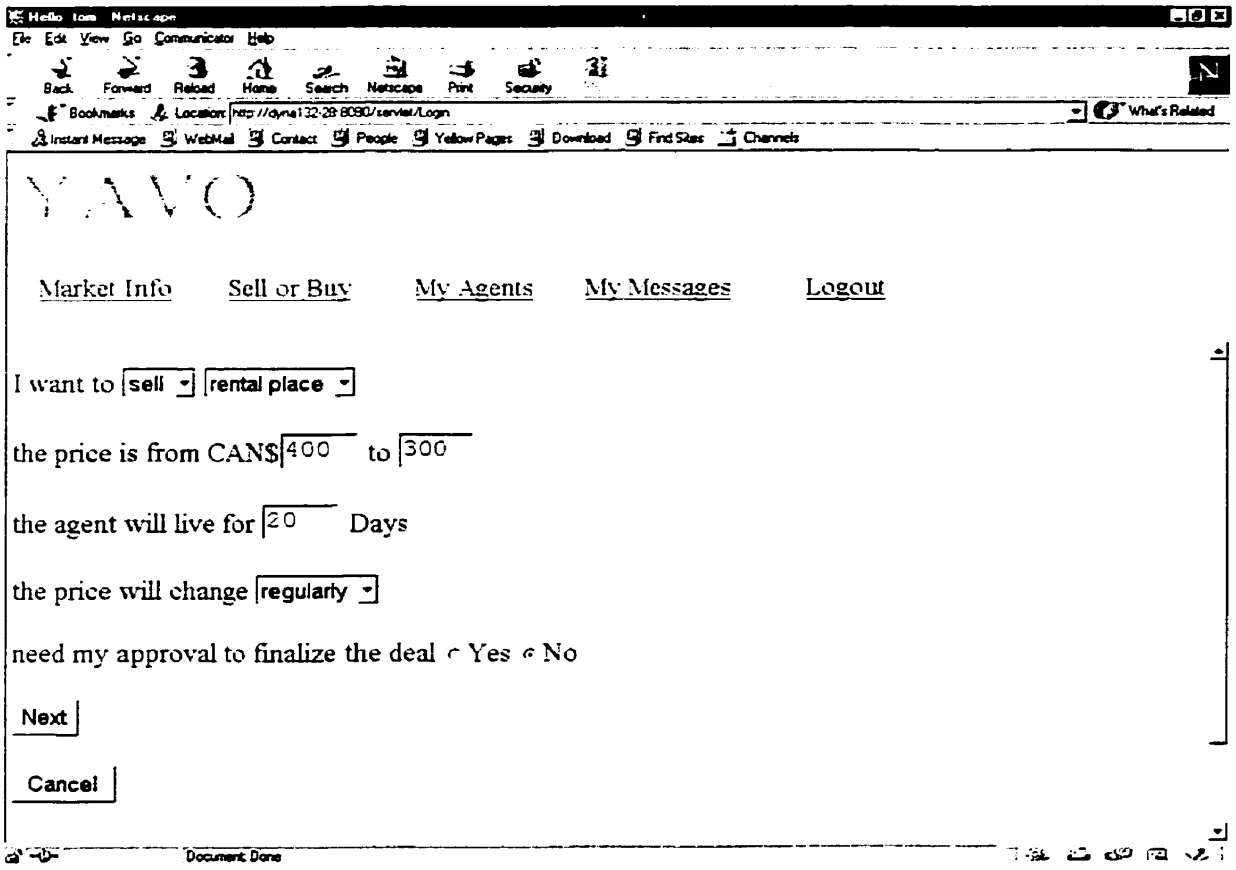
The following information was entered successfully

- **Actiontype:** buy
- **Goods or Service:** rental
- **start Price:** 300
- **Final Price:** 400
- **Lifetime:** 20 days
- **Price Changing Strategy:** regularly
- **Need User Approval:** no
- **Marketplace:** //dyna132-28:8000/RP8000
- **Distance:** Less than 5 minutes walk
- **Number of bedrooms:** 1

Figure 6-3 Creation Confirmation of an Agent

6.2.3.2 Create Seller Agents

Figure 6-14 shows the first step for the trader Tom to create a seller agent.


Hello tom - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
Bookmarks Location: http://dyna132-28:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
YAVO
Market Info Sell or Buy My Agents My Messages Logout
Next
Cancel
Document: Done

I want to sell rental place

the price is from CAN$ 400 to 300

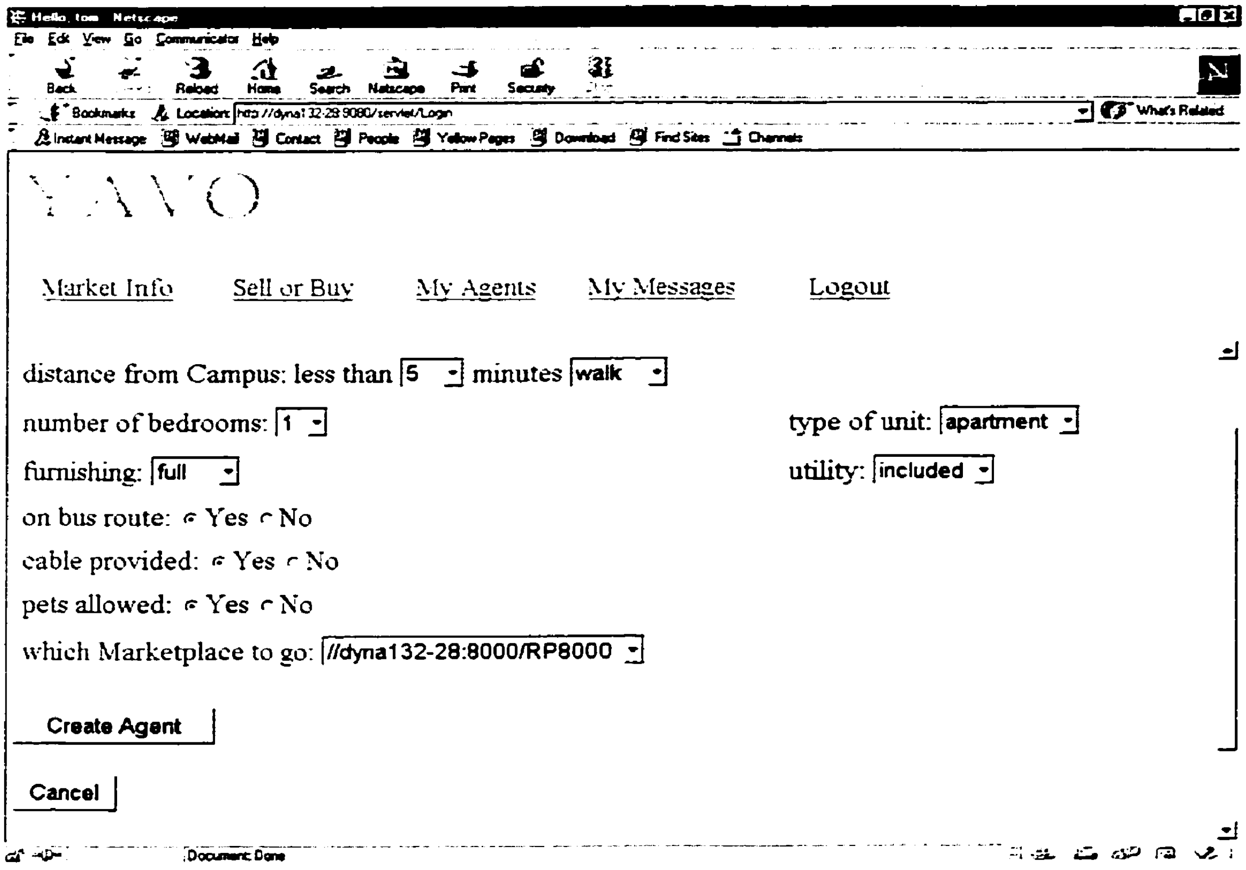
the agent will live for 20 Days

the price will change regularly

need my approval to finalize the deal Yes [✓ No]

Figure 6-1 First Step to Create a Seller Agent

Figure 6-15 shows the second step for the trader Tom to create a seller agent.


Hello, tom - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://dyna132-28:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
YAVO
Market Info
Sell or Buy
My Agents
My Messages
Logout
Document: Done

distance from Campus: less than 5 minutes walk

number of bedrooms: 1

type of unit: apartment

furnishing: full

utility: included

on bus route: [✓ Yes] No

cable provided: [✓ Yes] No

pets allowed: [✓ Yes] No

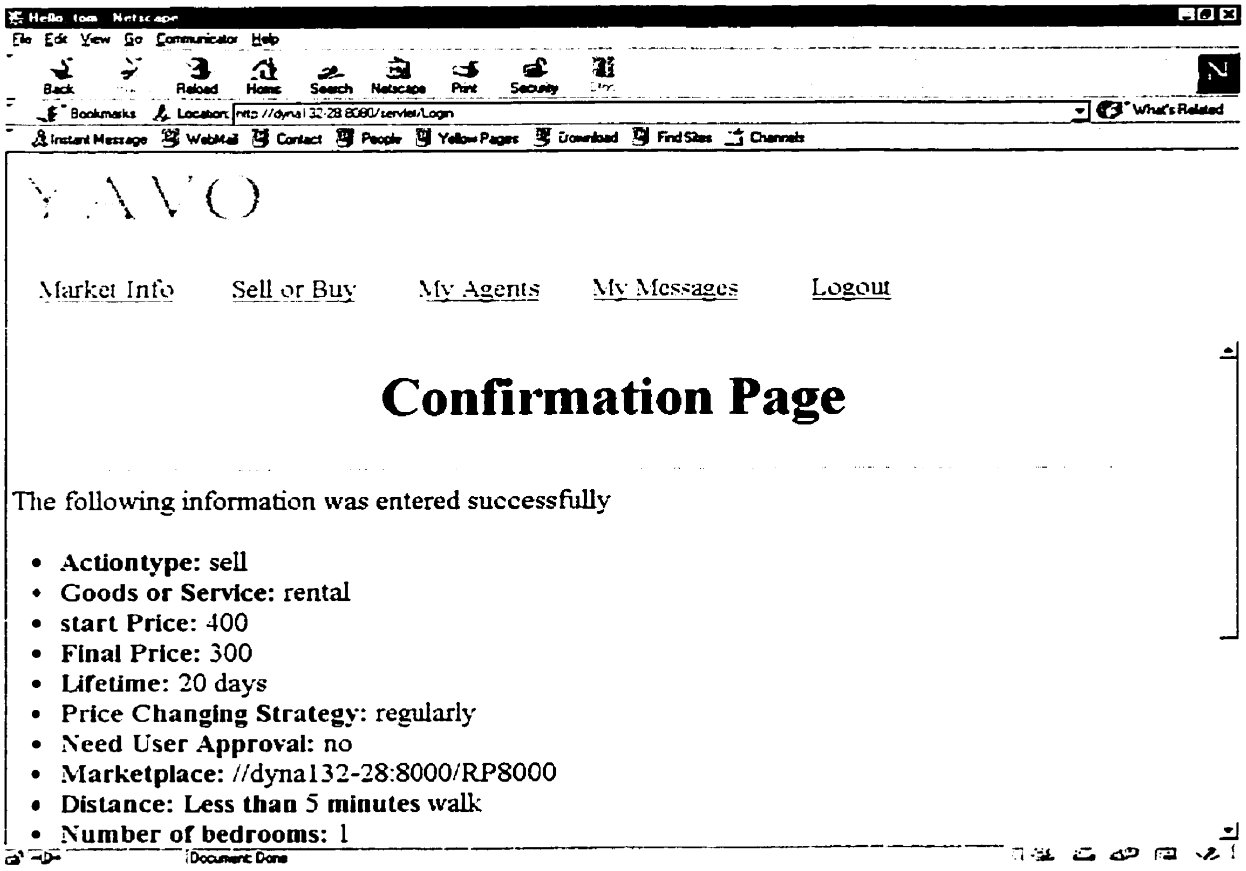
which Marketplace to go: //dyna132-28:8000/RP8000

Create Agent

Cancel

Figure 6-2 Second Step to Create a Seller Agent

After a seller agent has been created, the following page appears to confirm the creation.


Hello tom - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://dyna132-28:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
YAVO
Market Info
Sell or Buy
My Agents
My Messages
Logout
Document: Done

Confirmation Page

The following information was entered successfully

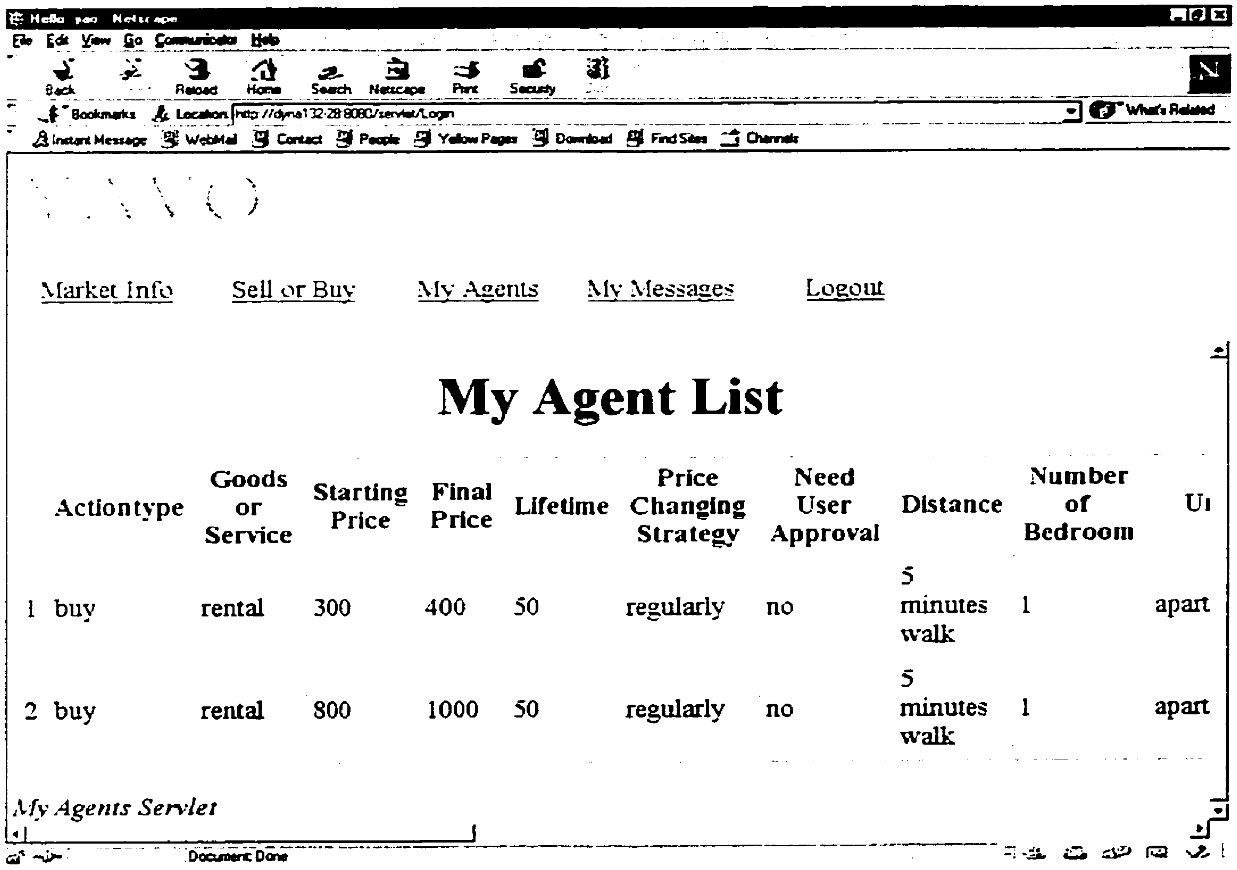
- **Actiontype:** sell
- **Goods or Service:** rental
- **start Price:** 400
- **Final Price:** 300
- **Lifetime:** 20 days
- **Price Changing Strategy:** regularly
- **Need User Approval:** no
- **Marketplace:** //dyna132-28:8000/RP8000
- **Distance:** **Less than 5 minutes** walk
- **Number of bedrooms:** 1

Figure 6-3 Creation Confirmation of an Agent

6.2.4 View All My Agents

After all the buyer agents and seller agents have been created, the traders Yao and Tom can view the runtime information of all their agents. In the following section, four screen shots are provided for each trader. There is a certain amount of elapsed time between each screen shot.

Figure 6-17 shows the status of all the agents that belong to Yao.


Hello yao Netscape
File Edit View Go Communicator Help
Back Reload Home Search Netscape Print Security
Bookmarks Location: http://dyna132-28:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
Market Info
Sell or Buy
My Agents
My Messages
Logout
Document Done

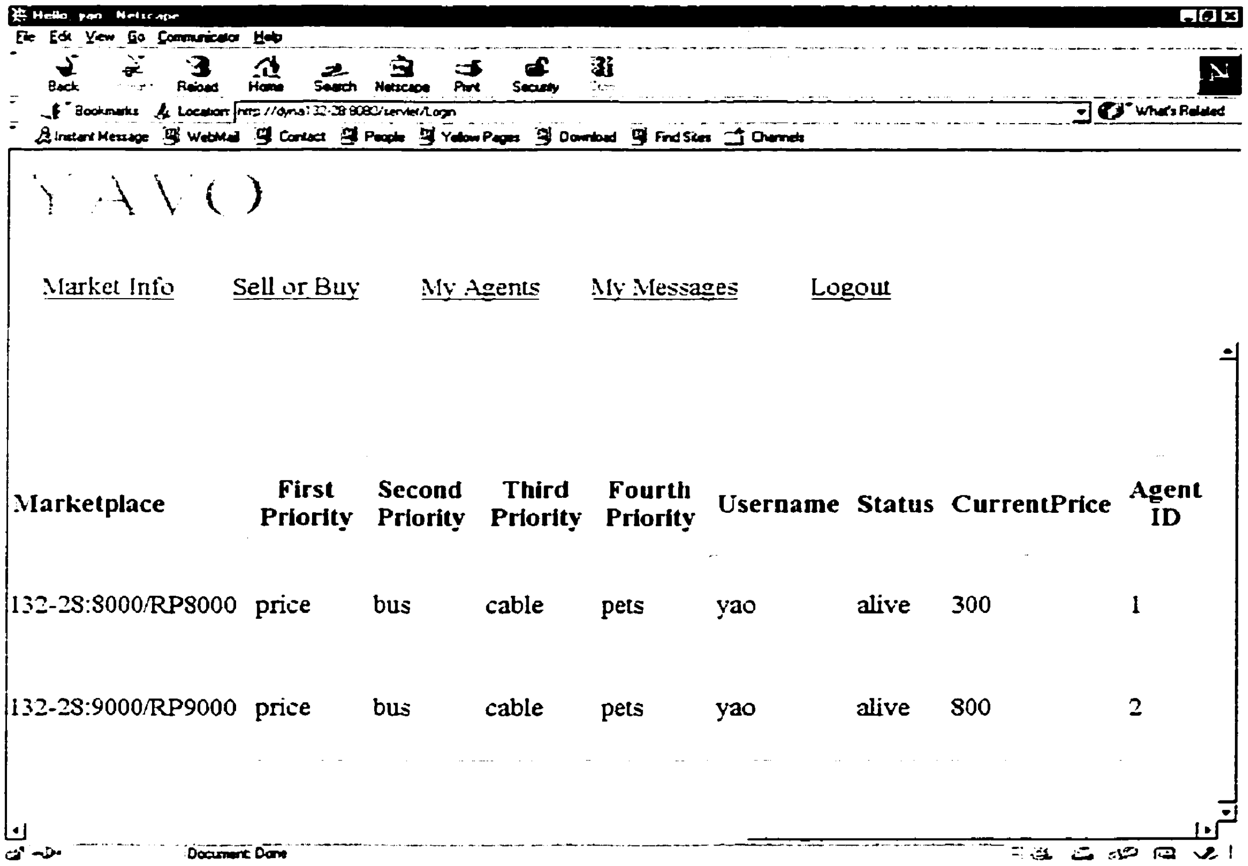
My Agent List

	Actiontype	Goods or Service	Starting Price	Final Price	Lifetime	Price Changing Strategy	Need User Approval	Distance	Number of Bedroom	Ui
1	buy	rental	300	400	50	regularly	no	5 minutes walk	1	apart
2	buy	rental	800	1000	50	regularly	no	5 minutes walk	1	apart

My Agents Servlet

Figure 6-1 Agent Status

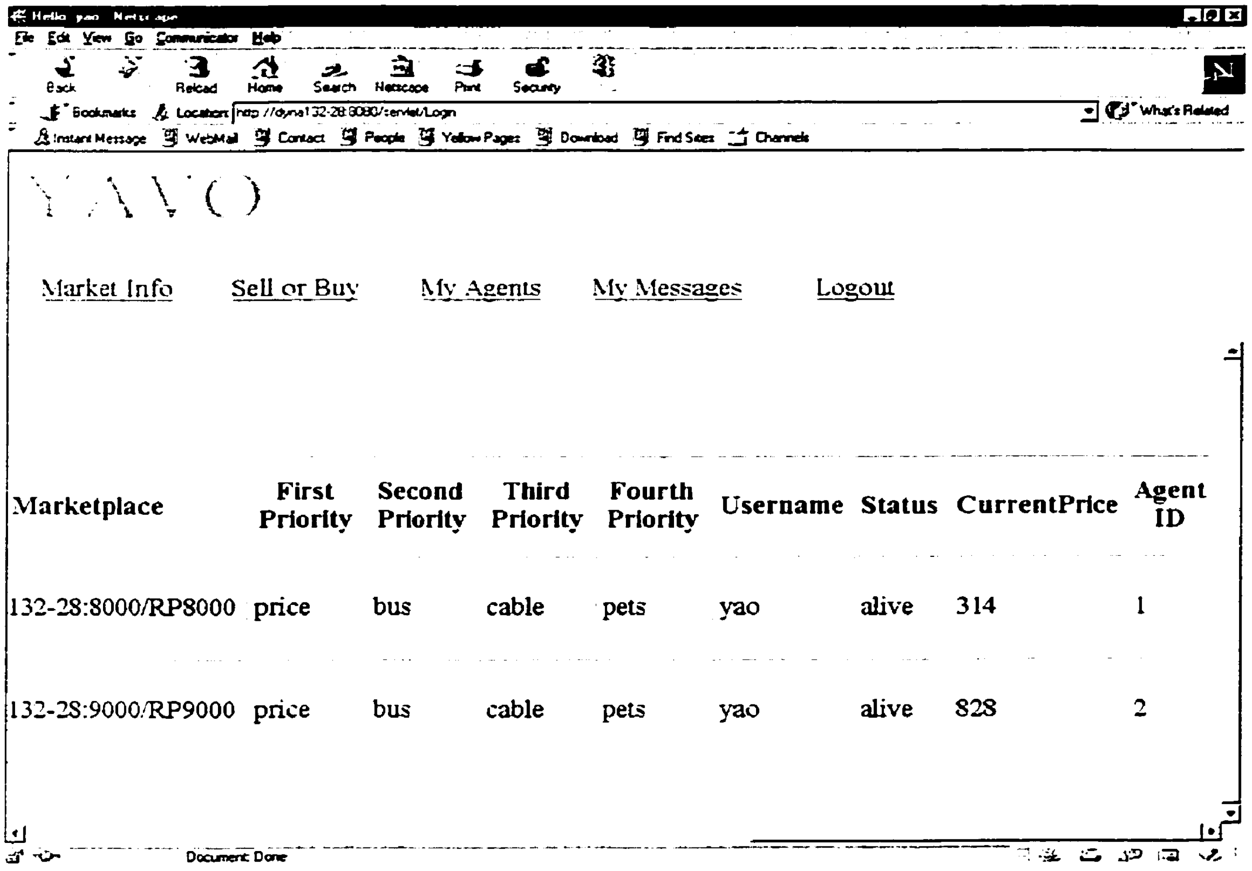
Figure 6-18 shows the status of all the agents that belong to Yao.


Hello yao - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://dyna132-28:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
YAVO
Market Info
Sell or Buy
My Agents
My Messages
Logout
Document: Done

Marketplace	First Priority	Second Priority	Third Priority	Fourth Priority	Username	Status	CurrentPrice	Agent ID
132-28:8000/RP8000	price	bus	cable	pets	yao	alive	300	1
132-28:9000/RP9000	price	bus	cable	pets	yao	alive	800	2

Figure 6-2 Agent Status

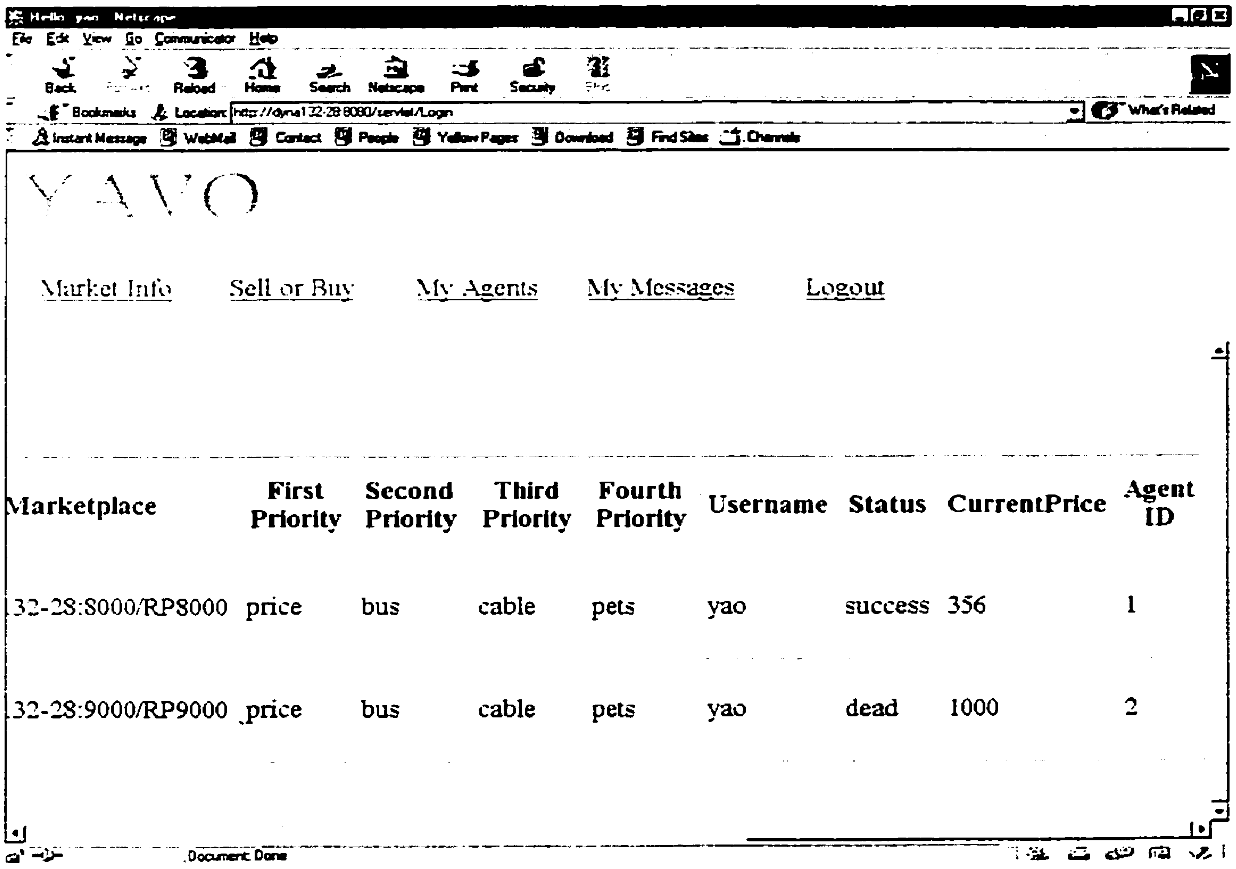
Figure 6-19 shows the status of all the agents that belong to Yao.


File Edit View Go Communicator Help
Back Reload Home Search Netscape Print Security
Bookmarks Location: http://dyna132-28:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
YAVO
Market Info Sell or Buy My Agents My Messages Logout
Document: Done

Marketplace	First Priority	Second Priority	Third Priority	Fourth Priority	Username	Status	CurrentPrice	Agent ID
132-28:8000/RP8000	price	bus	cable	pets	yao	alive	314	1
132-28:9000/RP9000	price	bus	cable	pets	yao	alive	828	2

Figure 6-3 Agent Status

Figure 6-20 shows the status of all the agents that belong to Yao.

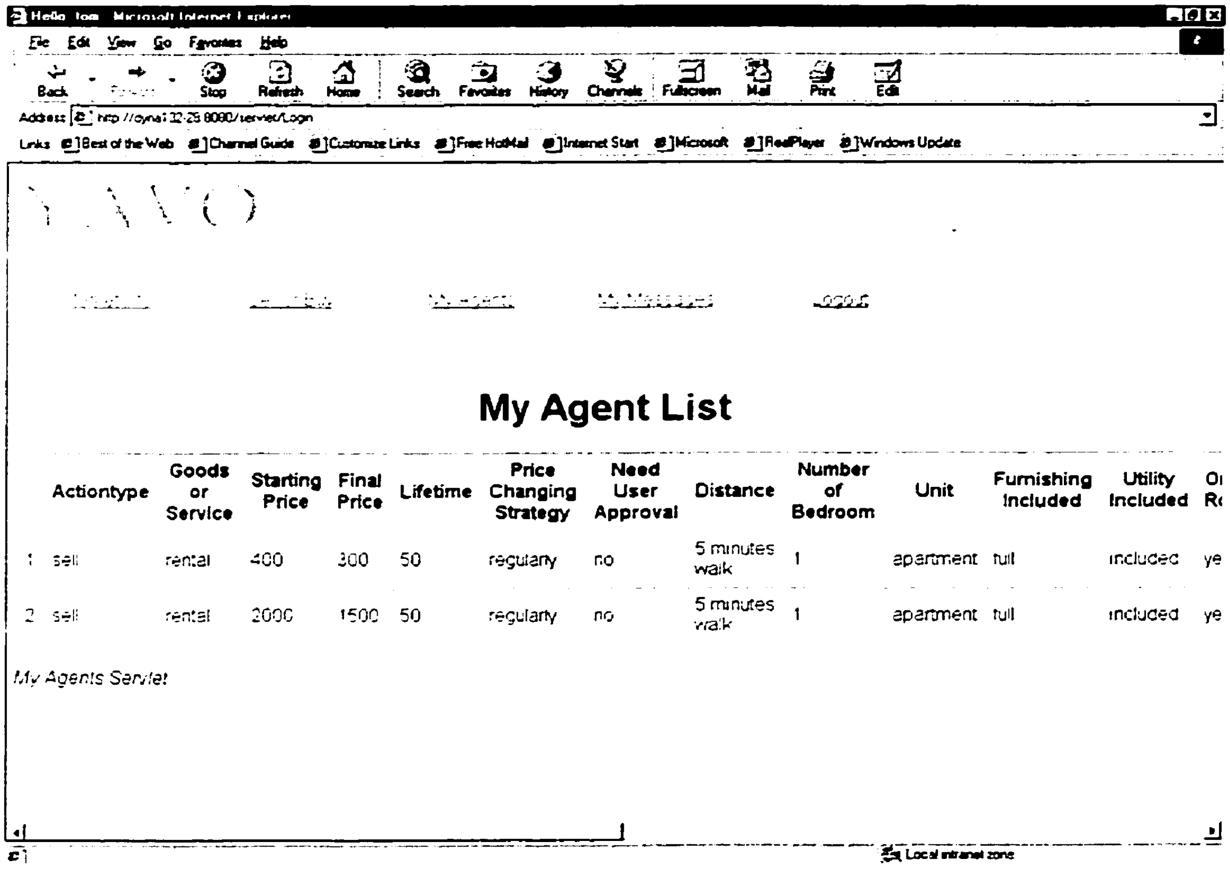


Hello yao - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://dyna132-28:8080/servlet/Login
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels
YAVO
Market Info Sell or Buy My Agents My Messages Logout
Document: Done

Marketplace	First Priority	Second Priority	Third Priority	Fourth Priority	Username	Status	CurrentPrice	Agent ID
32-28:8000/RP8000	price	bus	cable	pets	yao	success	356	1
32-28:9000/RP9000	price	bus	cable	pets	yao	dead	1000	2

Figure 6-4 Agent Status

Figure 6-21 shows the status of all the agents that belong to Tom.


File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address http://cyna132-25:8080/servlet/Login
Links Best of the Web Channel Guide Customize Links Free HotMail Internet Start Microsoft RealPlayer Windows Update
Local intranet zone

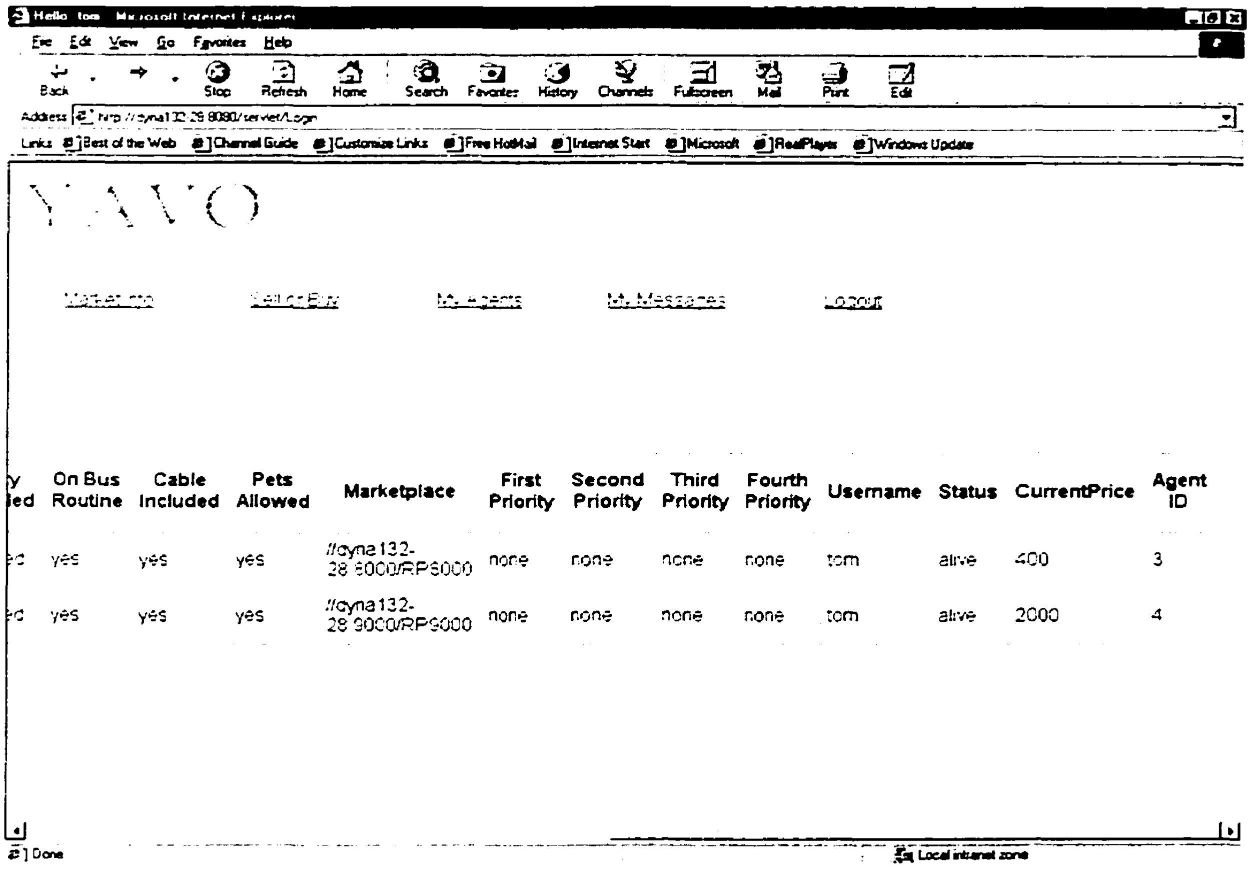
My Agent List

	Actiontype	Goods or Service	Starting Price	Final Price	Lifetime	Price Changing Strategy	Need User Approval	Distance	Number of Bedroom	Unit	Furnishing Included	Utility Included	O R
1	sell	rental	400	300	50	regularly	no	5 minutes walk	1	apartment	full	included	ye
2	sell	rental	2000	1500	50	regularly	no	5 minutes walk	1	apartment	full	included	ye

My Agents Servlet

Figure 6-5 Agent Status

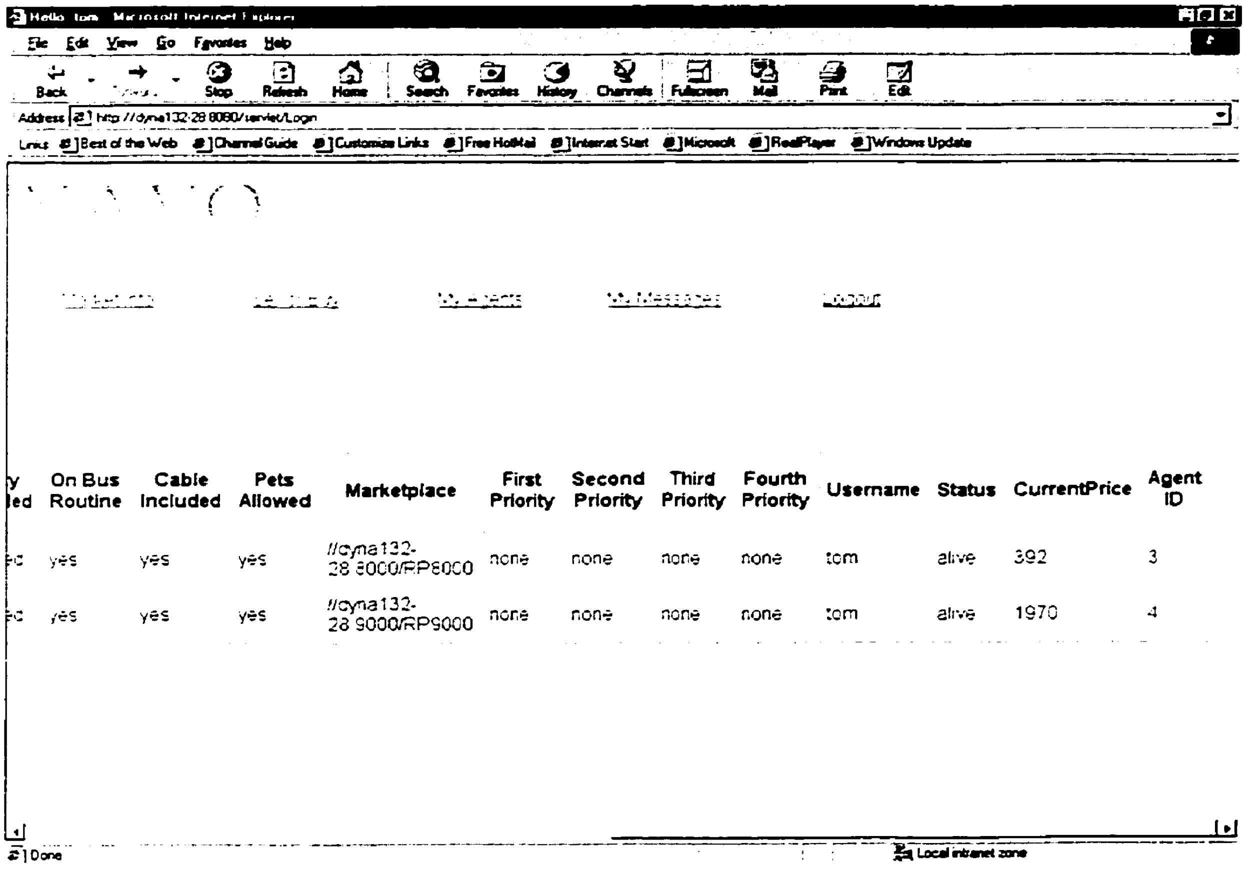
Figure 6-22 shows the status of all the agents that belong to Tom.



y ied	On Bus Routine	Cable Included	Pets Allowed	Marketplace	First Priority	Second Priority	Third Priority	Fourth Priority	Username	Status	CurrentPrice	Agent ID
ed	yes	yes	yes	//cyna132-28:9000/RP9000	none	none	none	none	tom	alive	400	3
ed	yes	yes	yes	//cyna132-28:9000/RP9000	none	none	none	none	tom	alive	2000	4

Figure 6-6 Agent Status

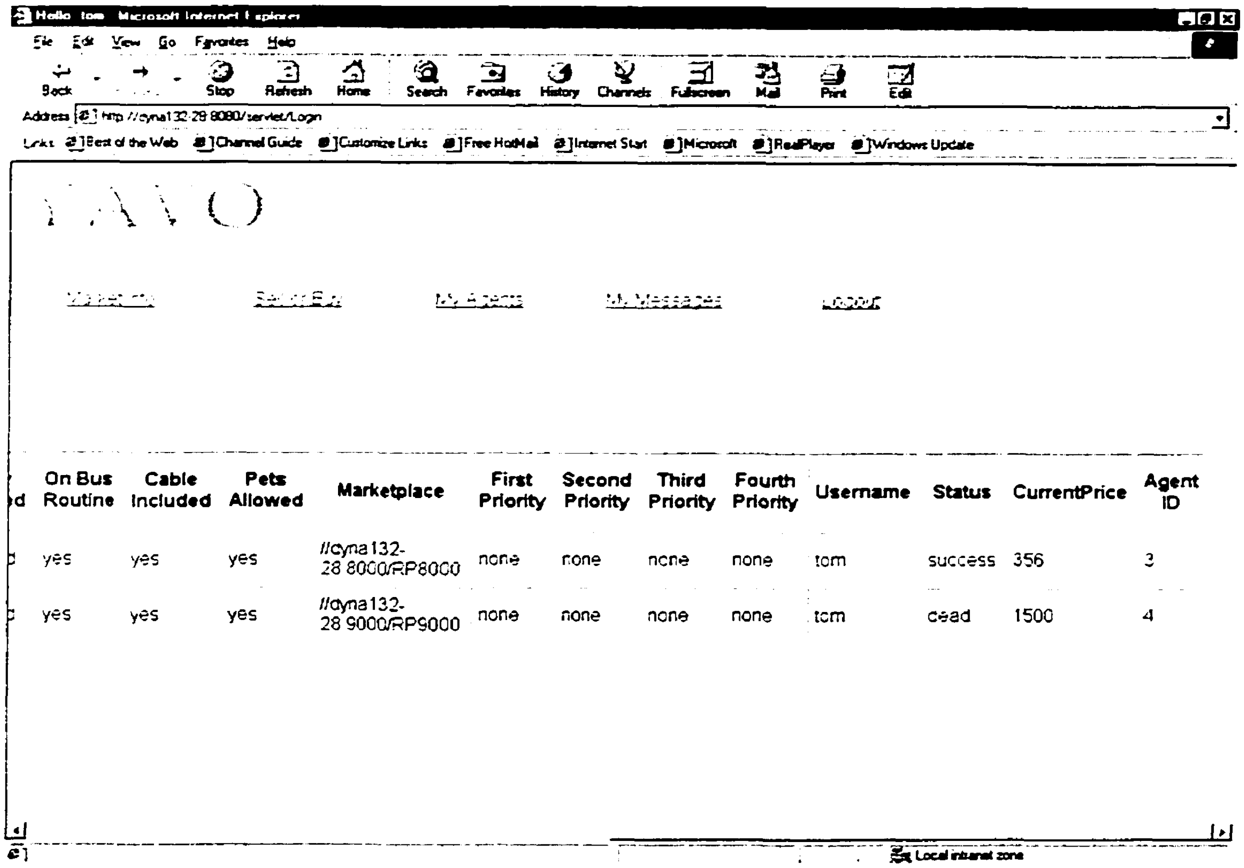
Figure 6-23 shows the status of all the agents that belong to Tom.



y led	On Bus Routine	Cable Included	Pets Allowed	Marketplace	First Priority	Second Priority	Third Priority	Fourth Priority	Username	Status	CurrentPrice	Agent ID
ed	yes	yes	yes	//cyna132-28 8000/RP8000	none	none	none	none	tom	alive	392	3
ed	yes	yes	yes	//cyna132-28 9000/RP9000	none	none	none	none	tom	alive	1970	4

Figure 6-7 Agent Status

Figure 6-24 shows the status of all the agents that belong to Tom.

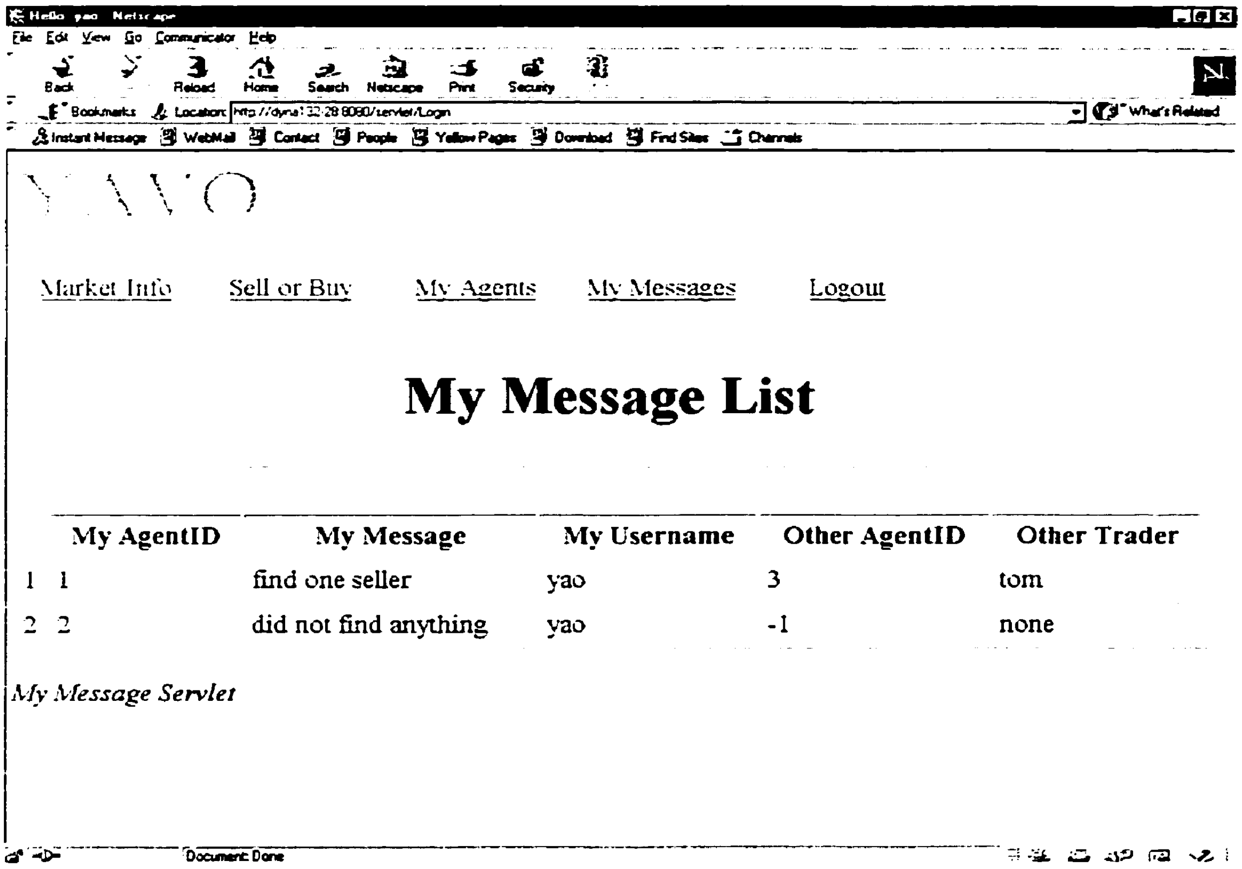

Hello tom Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address http://cyna132-28:8080/servlet/Login
Links Best of the Web Channel Guide Customize Links Free HotMail Internet Start Microsoft RealPlayer Windows Update
Local intranet zone

d	On Bus Routine	Cable Included	Pets Allowed	Marketplace	First Priority	Second Priority	Third Priority	Fourth Priority	Username	Status	CurrentPrice	Agent ID
[illegible]	yes	yes	yes	//cyna132-28 8000/RP8000	none	none	none	none	tom	success	356	3
[illegible]	yes	yes	yes	//cyna132-28 9000/RP9000	none	none	none	none	tom	dead	1500	4

Figure 6-8 Agent Status

6.2.5 View All My Message

Figure 6-25 shows the messages from all the agents that belong to Yao.



Market Info   Sell or Buy   My Agents   My Messages   Logout

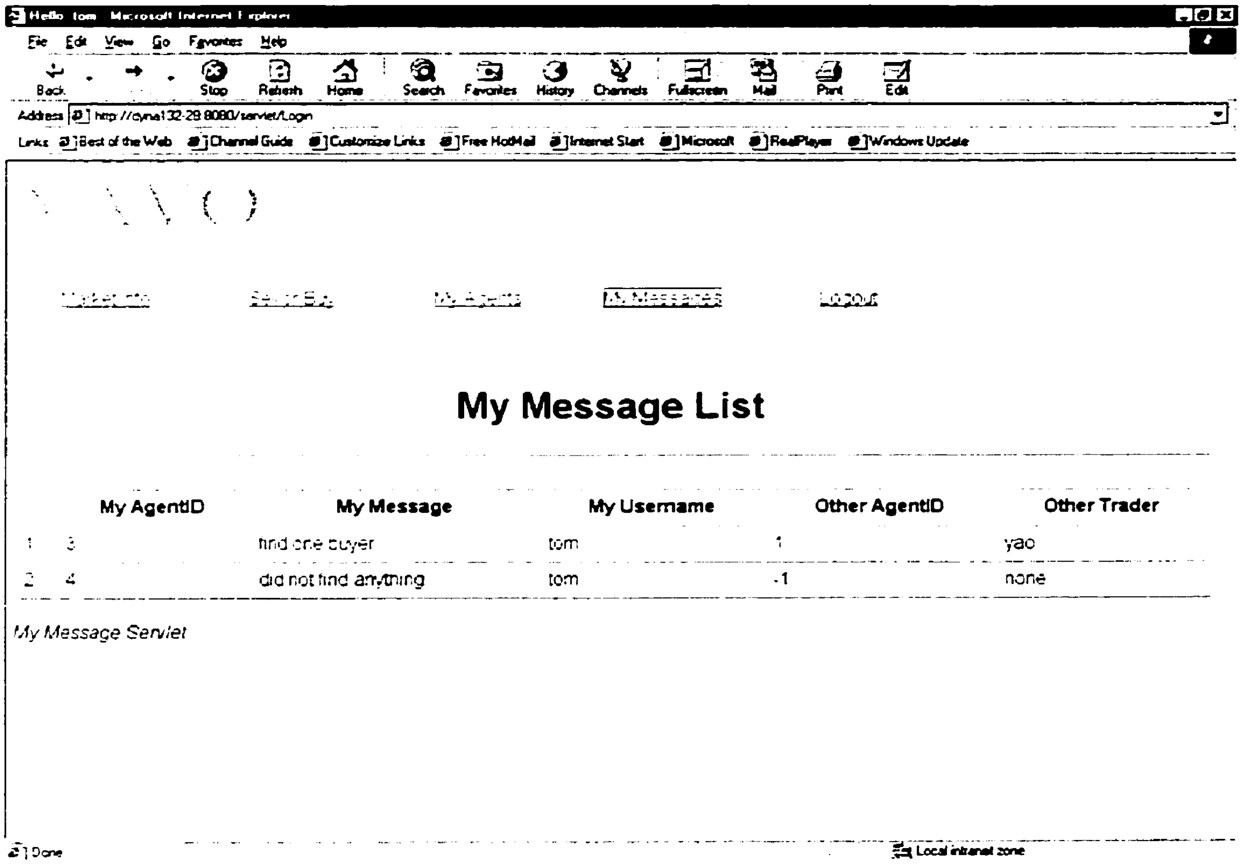
My Message List

	My AgentID	My Message	My Username	Other AgentID	Other Trader
1	1	find one seller	yao	3	tom
2	2	did not find anything	yao	-1	none

My Message Servlet

Figure 6-1 View Messages

Figure 6-26 shows the messages from all the agents that belong to Tom.



Address http://cyna132-28:8080/servlet/Login

My Message List

	My AgentID	My Message	My Username	Other AgentID	Other Trader
1	3	find one buyer	tom	1	yao
2	4	did not find anything	tom	-1	none

My Message Servlet

Figure 6-2 View Messages

Chapter 7

7 Conclusions

This thesis has covered the topic of mobile agent technology used in E-commerce. An on-line trading system, YAVO, demonstrates the application of this technology. YAVO is based on Voyager, an agent integrated distributed environment. Java technologies such as Servlets and JDBC are used to implement the YAVO system.

The advantage of the mobile agent model is illustrated by comparing it with the Client/Server model in distributed computing. The mobile agent model reduces network traffic, overcomes network latency, and brings greater flexibility to applications. This is the future trend of distributed computing applications.

The concept of E-commerce is introduced in relation to the definition, the categories, the technologies, the benefits, and the applications. The benefits of E-commerce has lead to its explosive growth. Mobile agents automate some steps in Consumer Buying Behavior. Mobile agents act on behalf of their creators, reduce the transaction costs in E-commerce, and change the ways we will conduct commerce in the future.

The YAVO system assists end-users selling and buying goods or services through mobile agents on the Internet. A system administrator creates trader accounts and distributed agent marketplaces through the Web. An authorized trader then sends an agent to an agent marketplace to buy or sell specific goods or services based on user-specified strategies. Based on these system functionalities, several business-related objects and their responsibilities were described. These objects are a mobile agent, an agent marketplace, a trader, and a bank.

As a Web application, YAVO includes Web clients (such as Web browsers), Java Web Server, Voyager Server, Database System, and User Interface (UI) logic. Java Servlets are the UI logic, which connects the Web clients to the back end servers. Java Servlets examples used in YAVO were listed.

The mobile agent platform of YAVO is Voyager, a full-featured agent-enhanced distributed computing environment. It provides rich functionalities for YAVO implementation. The examples are interfaces used for distributed computing, creating a remote object, using naming services, sending messages, creating mobile agents by dynamic aggregation, and using Timer classes. Persistent storage on the Java Web Server is provided through JDBC. Microsoft Access databases were used in YAVO because the JDBC-ODBC bridge driver is freely packaged with JDK from the Sun. The detailed design of all the databases was described in the YAVO implementation.

The non-implemented parts of YAVO are a banking system, a docking station, and negotiating mobile agents. There is only one service - rental place is implemented in this version. The non-implemented work will be added in the future. There are some drawbacks in the current version of YAVO. For example, the mobile agent is not intelligent enough to learn from its environment. The initial conditions of mobile agents cannot be modified at a later time. Also, the YAVO system has only been designed for a small scale application.

Bibliography

[Aglets 99]	Aglets. Available via http://www.trl.ibm.co.jp/aglets
[AMA 99]	Amazon.com (December 1999). Avaible at: http://www.amazon.com
[CZ 98]	William R. Cockayne and Michael Zyda (1998). Mobile Agents. Manning Publications Co.
[D'Agents 99]	D'Agents. Available via http://agent.cs.dartmouth.edu/software/agent2.0/
[ECO 99]	Electronic Commerce Office (November 1999). "Your Introduction to Electronic Commerce". Available at: http://net.gap.net/ch1.htm
[ECP 99]	Electronic Commerce Premier(December 1999). Available at: http://www.nemonline.org/mirl/ec/primer.htm
[GG 99]	Graham Glass (December 1999). "Reducing Development Effort using the ObjectSpace Voyager ORB" Available at: http://www.objectspace.com/products/vgrWhitePapers.htm
[GREENSPAN 99]	Alan Greenspan [6 May 6 1999]. "The Emerging Digital Economy". Available via http://www.ecommerce.gov/ede/chapter1.html
[GS 97]	Simson Garfinkel, Gene Spafford (1997). Web Security & Commerce. O'Reilly
[Hunter 98]	Jason Hunter (1998). Java Servlet Programming. O'Reilly; Cambridge.
[HBS 99]	Dan Harkey, Ken Burgett, Tim Stone (December 1999). "From E-Technology to E-Commerce". Available at: http://www.software.ibm.com/webservers/harkey/hmay99.html
[HC 99]	Michael Hamer and James Champey (December 1999). "Reengineering the Corporation: A Manifesto for Business Revolution." Available at: http://info.cardiff.ac.uk/uwcc/masts/ecic/eleccomm.html

[LO 99]	Danny B. Lange and Mitsuru Oshima (March 1999). "Seven Good Reasons for Mobile Agents." Communication of the ACM.
[MGM 99]	P. Maes, R. H. Guttman and A. G. Moukas (March 1999). "Agents that Buy and Sell: Transforming Commerce as we Know It." The Communications of the ACM.
[NEWS 99a]	the InternetNews.com Staff [22 November 1999]. "Survey: Small Biz Online Doubles Since 1998"; E-Commerce News Archives. Available at: http://www.internetnews.com/ec-news/article/0,1087,archive_4_242981,00.html
[NEWS 99b]	the InternetNews.com Staff [9 November 1999]. " Report: 19 Million Americans to Spend S7.8 Billion Online for Holidays"; E-Commerce News Archives. Available at: http://www.internetnews.com/ec-news/article/0,1087,archive_4_234761,00.html
[OBI 99]	OBI Library (November 1999). "Open Buying on the Internet." Available via http://www.openbuy.org/obi/library/white-paper.html
[Rational 98]	Rational Software (December 1999). "The Unified Modeling Language (UML)." Available via http://www.rational.com/
[RATIONAL 99]	UML Resource Center (November 1999). Available via http://www.rational.com/uml/index.jtmpl
[RBPEL 91]	James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen (1991). Object-Oriented Modeling and Design. Prentice-Hall, Inc. A Simon & Schuster Company Englewood Cliffs, New Jersey
[Solinsky 99]	Jason Solinsky (November 1999). "An Introduction to Electronic Commerce". Available at: http://www.worldquest.com/wqu/elecomm.htm
[Voyager 99]	Voyager ORB Developer Guide(December 1999). Available via http://www.objectspace.com/products/vgrProdDocs.asp
[WINE 99]	Wine.com (December 1999). Available via http://www.wine.com