Available online at www.sciencedirect.com
SCIENCE DIRECT®
Decision Support Systems 39 (2005) 371–383
Decision Support Systems
www.elsevier.com/locate/dsw

Mobile agents for a brokering service in the electronic marketplace

Timon C. Du[a], Eldon Y. Li[b,*], Eric Wei[c]

[a] Department of Decision Sciences and Managerial Economics, The Chinese University of Hong Kong, Hong Kong, China
[b] Department of Information Management, Yuan Ze University, 135 Yuan Tung Road, Chung Li 320,Taiwan
[c] Department of Industrial Engineering, Chung Yuan Christian University, Taiwan

Received 20 April 2002; received in revised form 8 January 2004; accepted 17 January 2004
Available online 11 March 2004

Abstract

The electronic marketplace is a new medium for exchanging information, goods, services, and payments. It houses infrastructure, facilitates transactions, and matches buyers with sellers. A mobile-agent-based marketplace allows corporate data to be maintained by local buyers and sellers and transferred to the marketplace only when orders are matched. This provides the participating companies with autonomy and independence. This study proposes a framework for using mobile agents to demonstrate autonomous behavior in the electronic marketplace. A prototype system is developed to demonstrate the implementation of this framework. Furthermore, a simple simulation design is used to explore the performance of mobile agents using two different product-purchasing policies. Statistics regarding execution time are collected and analyzed. The results suggest that mobile agents can easily aggregate orders and shorten the execution time for matching orders.

Keywords: Electronic commerce; Electronic marketplace; Brokering agents; Mobile agents; Simulation; System performance; Aggregate orders

1. Introduction

Despite the recent meltdown of dot-com companies, business-to-business (B2B) electronic commerce (EC) has been steadily growing. More and more buyers, sellers, and service providers are integrating their business operations with Web technologies, pushing the EC platform to become a new medium for exchanging information, products, services, and payments. Such exchanges can be carried out through electronic marketplaces, which are the core components of the digital economy. An electronic marketplace has three main functions: brokering buyers and sellers, facilitating transactions, and providing institutional infrastructure [2]. The first function determines product features, aggregates orders, determines prices, and matches buyers with sellers. The other two functions relate to the process and infrastructure of transactions, such as logistics, settlements, legal matters, and regulations.

Electronic marketplaces manage participants, information, and business processes to increase the liquidity of trade (e.g., www.freemarkets.com, www.metals.com and www.e-wood.com [10]). A typical B2B electronic marketplace may be supplier oriented or buyer oriented. The former is for expanding sales to potential buyers, while the latter is for finding com-

* Corresponding author. Tel.: +886-3-463-8909; fax: +886-3-463-8884.
E-mail address: eli@saturn.yzu.edu.tw (E.Y. Li).

0167-9236/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2004.01.003

petitive suppliers and products. However, both approaches encounter the difficulty of information sharing. For example, the product catalog and order information need to be stored in either the supplier-side database or the buyer-side database. Companies, especially large ones, rarely accept their data being stored in another company's database. Instead, a reputable third-party company can run a successful intermediary-oriented marketplace, since it can guarantee that the corporate data are securely maintained. However, the need is for the intermediary to maintain a voluminous amount of data to match buyers with sellers. Sometimes, the intermediary platform can be linked to the local corporate information systems for repetitive buyers and sellers. However, this approach requires much effort in system integration and provides low flexibility for system changes. Alternately, an agent-based marketplace may be established to allow corporate data to be maintained by local buyers and sellers and transferred to the marketplace only when orders are to be matched. This distributes the workload to the participating companies and reduces bandwidth usage due to client–server communications. Furthermore, it offers the advantages of autonomy and independence to these companies. Surprisingly, the existing literature shows no documented usage of mobile agents in the electronic marketplace. Only one study [12] prototyped a multi-agent system for dealing with real estate on the Internet, but that was not an electronic marketplace. In this study, we will use mobile agents to demonstrate autonomous behavior in the electronic marketplace. We will also demonstrate the cost savings achieved by aggregating buyers' orders. An automatic business transaction environment is established through message exchanges among mobile and stationary agents. This framework has architecturally neutral, autonomous, personal, and mobile characteristics, which are suitable for the distributed architecture of electronic commerce.

2. Mobile agents with intelligence

Agents are autonomous objects created for dynamic and distributed applications that are responsible for executing designated tasks. Referring to the definition offered by Wooldridge and Jennings [17], agents can be identified as either strong or weak. Strong agents have the capabilities of (1) mentalist notions, (2) rationality, (3) veracity, and (4) adaptability and learning. These capabilities come mainly from the technology of artificial intelligence. Weak agents, in contrast, can complete tasks autonomously, interact with external objects, and be reactive or proactive toward environmental change based on a pre-planned scheme.

Barbosa and Silva [3] use the technology of intelligent agents and knowledge to improve transactions in the electronic marketplace. The agents, including buyer agents, seller agents, and intermediary agents, match transactions in the electronic marketplace based on prewritten knowledge. However, this approach places a heavy workload on the intermediary and thus limits the number of participants that can be served by the intermediary. In contrast, in this study, the mobile agent queries the remote sellers for the aggregate of buyers' orders. The workload is distributed among the sellers and the intermediary.

The mobile agent has been implemented in many distributed environments to share system loading and increase flexibility, such as the agent-based system (ABS) and the multi-agent system (MAS). Several applications have been successfully developed using mobile agents. Examples are the supply chain SMART project (http://smart.npo.org), the virtual enterprise [11], information retrieval [6], the Internet-based auction house [14], and distributed network management [9]. Some studies have further integrated mobile agents with CORBA, such as MESIS resource management [4] and broadband intelligent networks [7]. In general, the mobile agent system can be applied to the areas of electronic commerce, personal assistance, secure brokering, distributed information retrieval, telecommunication networks services, workflow applications and groupware, monitoring and notification, information dissemination, and parallel processing [13].

3. An agent-based electronic marketplace

In an agent-based electronic marketplace, there are four possible relationships between buyers and suppliers: one-to-one, one-to-many, many-to-one, and many-to-many. The one-to-one relationship is the simplest; a mobile agent is placed between one buyer and one supplier to negotiate the contract. Similarly,

one-to-many and many-to-one relationships, e.g., supplier-oriented marketplaces and buyer-oriented marketplaces, can also involve placing mobile agents between buyers and suppliers. In a many-to-many relationship, a buyer can purchase products from many suppliers and a supplier can sell products to many buyers. This is where a marketplace is particularly useful; it can provide an intermediary to match both sides. In this relationship, the orders from different buyers may or may not be aggregated. However, due to the price sensitivities of different suppliers, the intermediary can provide functions to aggregate demands for the same type of product to negotiate a better price. This strategy can also decrease the system query and transaction costs. Fig. 1 presents a framework for using mobile agents in an electronic marketplace. This framework is an adaptation of the models in MAgNET [8], TESTBED [16], BROKERAGE [12], and Grasshopper 2.2 [1]. There are five major components in the framework: a main system, an agent manager, a two-stage matching process, a buyer system, and a supplier system. The introduction of the agent platform and a discussion of the first three components are presented below.

3.1. The agent platform

The agent-based marketplace is developed on top of an agent platform, supported by Tahiti, of IBM's Tokyo laboratory, to provide an application interface for modeling complicated agent behavior. It can create, clone, and dispatch mobile agents. The platform has two essential classes, *AgletProxy* and *AgletContext*. The *AgletProxy* is a proxy server that sends the client requests to the remote server and brings back the results after completing the assignment. The communication between agents can only pass through the designated functions. Therefore, the *AgletProxy* is like a protective umbrella that provides the transparency of objects. The *AgletContext* provides an implementation environment for the Tahiti platform. When a mobile agent is sent to a remote Tahiti server, it is

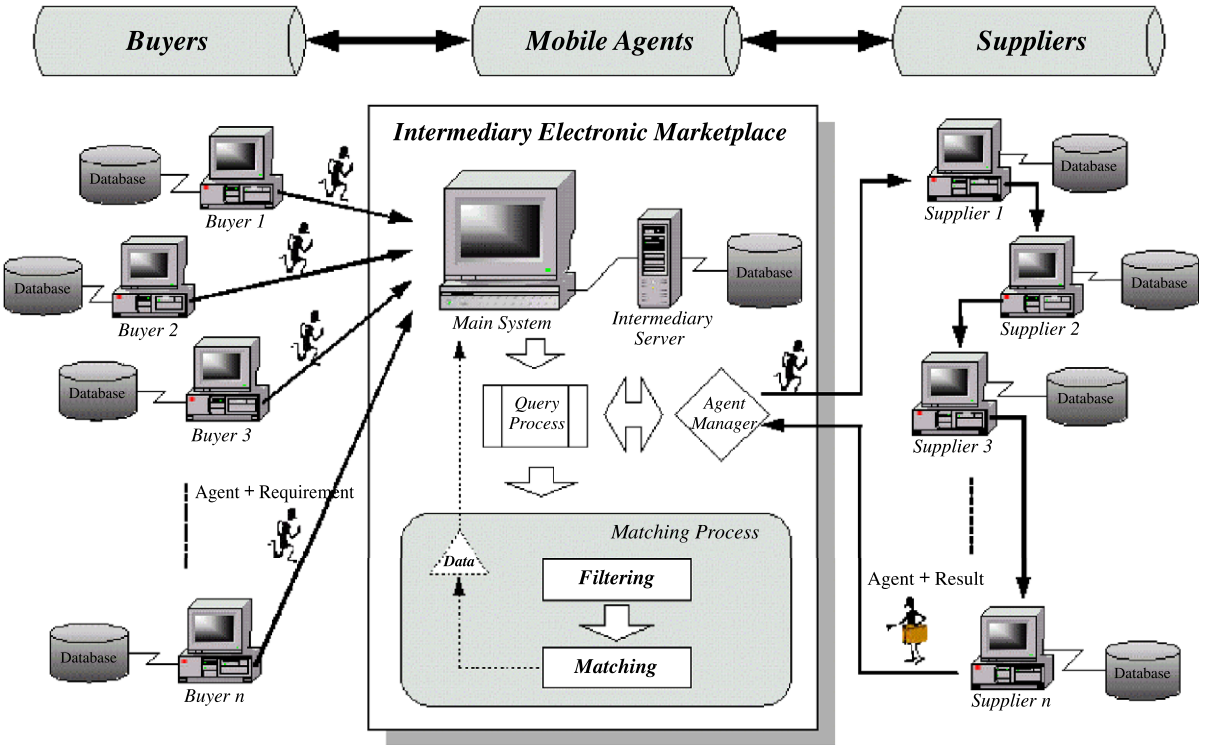

Fig. 1. A framework for an agent-based electronic marketplace.

decomposed into several message streams by its own *AgletContext* of the local Tahiti server and then sent to the *AgletProxy* of the remote Tahiti server. The message streams are then recomposed by the *AgletContext* of the remote Tahiti server. Fig. 2 shows this detailed process in the Object Management Group's UML (Unified Modeling Language) format. The function descriptions are listed in Table 1. In this process, agent operation *doOperation()* is passed to the remote proxy server, and the proxy initiates the implementation methods with the security mechanism *getSecurityManager()* in the remote server. The response is obtained from executing *invokeCallback()* of *ConcreteAgent*—an agent that is a class object in the Tahiti platform and is responsible for handling user-specified operations. The last activity, *doFinalMethod()*, which is activated by the request of *doOperation()*, is then carried out by the mobile agent. During the program execution of callback, necessary help can be initiated by *invokeHelperMethod()*. The mobile agent then moves to its next destination or returns to the sender following the itinerary.

3.2. Main System

The Main System contains the mobile-agent-based intermediary server and the database. It coordinates the operations of three sources of agents: the buyers, the suppliers, and the intermediary mobile agents. The demands, input by buyers through the GUI or an Applet, are placed in *buy_req* database records that are defined as, *buy_req* (P^B, I^B, T^B, Q^B, S^B, $QoS1^B$, $QoS2^B$, ...) where P^B is the upper limit of the unit-purchasing price of products, I^B is the quantity, T^B is the delivery date, Q^B is the quality level, S^B are the potential suppliers, and $QoS1^B$ is the supplementary criteria.

The record contains the upper limit of the buyer's product unit purchasing price, the order quantity, the delivery date, the quality level, a list of potential suppliers, and other supplementary criteria (there may be more than one, or none). Buyers send the *buy_req* information to the intermediary. The intermediary collects the individual demands for each product (*buy_req*) into a database record *dem_spec*. Note that the orders in *dem_spec* are not aggregated. The intermediary can combine the orders into an aggregate order in every certain time frame. The aggregate order contains the average unit product price, the total demand quantity, the list of all candidate suppliers, and other supplementary criteria. It is placed in a database record, *agg_req*. The intermediary further generates an itinerary for the mobile agent, based on potential suppliers listed in each *buy_req*

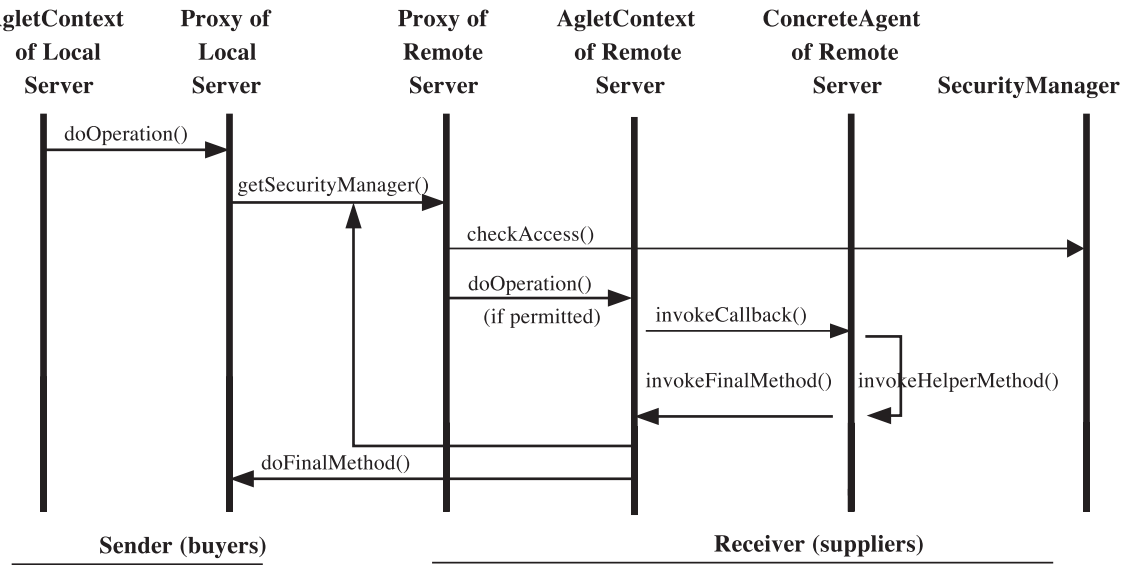

Fig. 2. The agent activities in a UML sequence diagram.

Table 1
Function descriptions of mobile agents in the Tahiti platform

Function Name	Description	Input	Output
doOperation()	request operations	receive the request from local server	pack the data with security settings to the local proxy
getSecurityManager()	get access privileges of aglets including file system, network access, and others	receive the data and security settings of local agents	send the data and security settings to remote proxy
checkAccess()	check the permission for agent access	get the security settings of the agent	grant or deny access
invokeCallback()	call the functions defined in Helper	get the agent data, such as product items and prices	locate corresponding private functions for the caller
invokeHelperMethod()	implement the private or protected functions and only provide to Callback function	pass input parameters to private functions	activate system-defined functions
invokeFinal Method()	call system-defined functions, such as moveTo, save, die, backHome, clone, getId, and sendMessage in Tahiti	pass input parameters to system-defined functions	output system-defined functions and prepare the feedback values
doFinalMethod()	implement system-defined functions and send the results back to the local server	get the feedback values	send the feedback values to the local server

received by the intermediary. A mobile agent accepts either *dem_spec* or *agg_req* and visits potential suppliers to acquire the estimate, *bid_spec*, of each supplier.

The estimate is defined as *bid_spec* (P^S, I^S, T^S, Q^S, $QoS1^S$, $QoS2^S$, …), where P^S is the unit product price, I^S is the order quantity (from *buy_req*) or the largest capacity (the current inventory) of the supplier, T^S is the earliest delivery time, Q^S is the quality level, and $QoS1^S$ is the supplementary criteria (there may be more than one, or none).

The *bid_spec* estimate contains the supplier's product-unit selling price (including shipping and handling cost), the order quantity or the largest deliverable quantity of the supplier (whichever is smaller), the earliest delivery date, the quality level, and other supplementary criteria.

Before mobile agents can reach the servers of the suppliers, they must pass the suppliers' security check. The correct account and password allows the mobile agent to access authorized supplier servers. While the mobile agent enquires about the price on the server, a stationary local agent, residing in the supplier server, is activated. There are two scenarios with respect to the suppliers: a sufficient inventory and an insufficient inventory. The local stationary agent returns the request by generating *bid_spec* if the inventory is sufficient for the demand. If the current inventory is insufficient, then the local stationary agent requests the mobile agent to wait. The local agent then calls the back-end production information system of the supplier to evaluate the capacity based on the delivery date in *dem_spec* or *agg_req*. Since the production information system needs a longer time to obtain the predicted production quantity, the mobile agent clones a temporary agent to continue the inquiry process with the supplier. The mobile agent then follows the itinerary and moves to the next supplier. The temporary agent returns to the intermediary after obtaining the estimate. Finally, the intermediary integrates the estimates received from both the mobile agent and the temporary agents and recommends a list of suppliers to the buyers by sending another mobile agent to the buyers.

3.3. Agent Manager

At any point in time, the entire marketplace system may have countless agents. These agents are managed by the Agent Manager component. The function of the Agent Manager is to control the detailed message flows of the various agents (both mobile and station-

ary) in the marketplace. It contains *Agent Name Service* (ANS), *Agent Recommend Service* (ARS), and *Brokering Control Engines*, as shown in Fig. 3.

An electronic commerce environment normally involves three parties: a buyer, a supplier, and a broker. In an agent-based computer system, a buyer sends buying agents to designated suppliers to proceed with predefined tasks, such as querying quantity and price. However, the drawback is that the buyer may miss offers of better prices or quality other than those selected. Conversely, if many buyers send their buying agents to the same suppliers querying the same product information, then the system performs many redundant tasks that waste network resources. Therefore, a brokering service can be used to eliminate the problems of missed opportunity and redundancy. To effectively carry out the brokering service, a brokering control engine is needed. This engine normally includes four functions: *selection*, *evaluation*, *filtering*, and *assignment*. The selection function compares buyers' orders with suppliers' estimates. If there are many buyers for many products, then it is necessary to have a coordination process for grouping the buyers of each product. Consequently, the members of a buyer group are dynamically formed. The selected suppliers should be evaluated based on the buyers' terms because different suppliers may have different specifications matching the buyers' requirements. Therefore, the members of a supplier group are also dynamically formed. The two dynamically formed groups of buyers and suppliers can then be linked efficiently by the evaluation function. The filtering function further screens out unqualified buyers or suppliers. Finally, the assignment function decides the matching pair of supplier and buyer.

In our framework, as depicted in Fig. 3, every new agent must be registered with the ANS before it becomes active in the marketplace. The intermediary uses ANS to identify and interact with *Local Agents* that are stationary and reside on the buyers' or suppliers' side. In the marketplace, *Message Generator* is responsible for generating communication information, *Message Queue* is the input queue, *Message Router* outputs the information to the Local

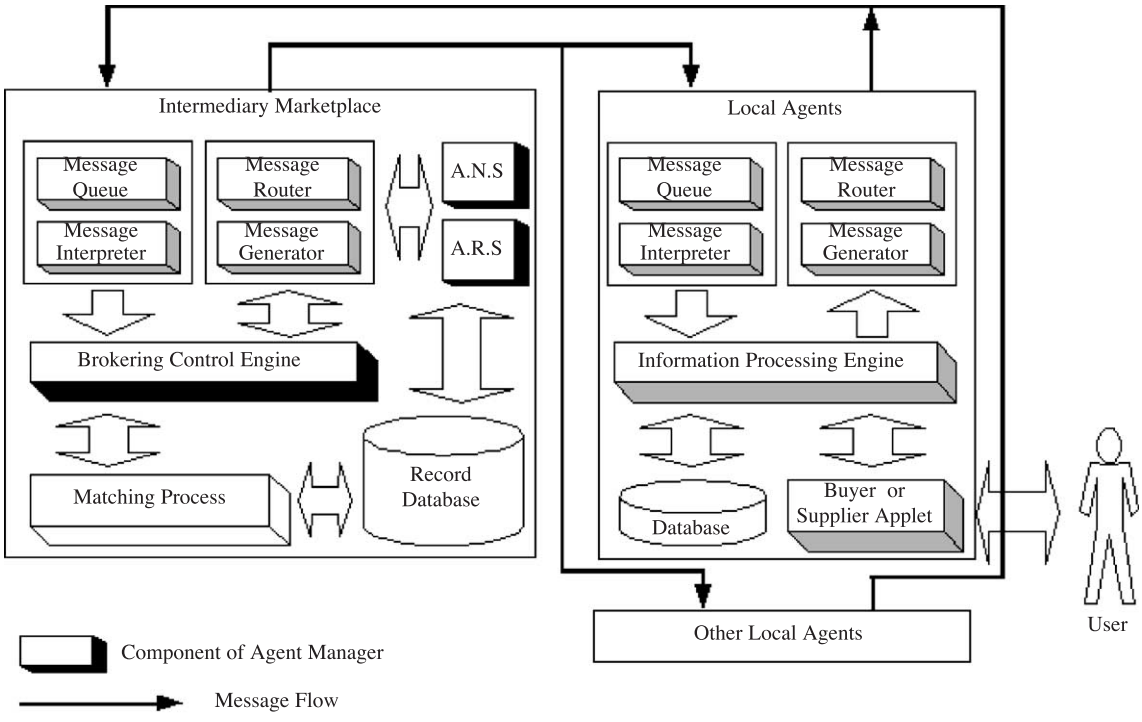

Fig. 3. Message flow between agents.

Agent, and *Message Interpreter* analyzes the contents of the received information. A *Brokering Control Engine*, as described in the system design of Ref. [12], manages the interfaces between messages and the *Matching Process*.

Since the brokering control engine can effectively eliminate redundant tasks and reduce opportunity costs, it can also ensure that better matching results can be achieved. At the same time, the privacy of the trading parties can be preserved. Moreover, using the brokering control engine, a system developer can also implement an order-aggregating policy to increase the purchasing volume of each product so that the unit price can be decreased.

Matching Process is a two-stage process that implements filtering and matching operations on the acquired estimates by referring to the transaction history in the *Record Database*. The matching recommendations are generated by an Agent Recommend Service (ARS) and stored into the Record Database for later use.

Local Agents have communication functions and an information-processing engine. The communication functions are the same as those of the message processors in the *Intermediary Marketplace*. The information-processing engine analyzes the data from buyers or suppliers. On the buyer's side, the users' demands, (*buy_req*), are sent by local agents to the intermediary through communication functions. On the supplier's side, the local agent receives the *dem_spec* or *agg_req* from the intermediary, checks the inventory, triggers the back-end production information system if necessary, generates a *bid_spec* estimate, and submits the estimate back to the mobile agent of the intermediary.

3.4. Two-stage matching process

To find the best combination of suppliers for the product ordered, a two-stage matching process, consisting of the filtering stage and the matching stage, is performed. The intermediary triggers this process after receiving a *bid_spec* from suppliers.

3.4.1. Filtering stage

This stage removes the estimates of unqualified suppliers based on the pre-determined order price. That is, the estimate should meet the constraint of $P^B \geq P^S_{min}$. Note that the estimates also need to satisfy any other constraints and preferences that are stated in $QoS1^B$.

3.4.2. Matching stage

The matching process looks for the supplier that can provide the best price in the estimate. That is, it looks for (S*, P*, I*) by maximizing $\sum_{i\in B} P_i^B \times I_i^B - \sum_{j\in S*} P_j^S \times I_j^S$, where $\sum_{j\in S*} I_j^S \geq \sum_{i\in B} I_i^B$ and $I_j^S \in [B_j^0, B_j]$.

The demand of Buyer B_i is $P_i^B \times I_i^B$ and the supply of Supplier S_j is $P_j^S \times I_j^S$. The domain of I_j^S is $[B_j^0, B_j]$, and B_j^0 is the minimum order quantity. This process can reduce purchasing costs by matching the demand with the best price from the suppliers.

In the many-to-many relationship, the orders from different buyers are aggregated because it is reasonable to assume that different suppliers have different price sensitivities; that is, the larger the quantity ordered, the better the discount that can be acquired. Therefore, the intermediary should provide functions to integrate demands for negotiating a better price. This strategy can also decrease the system query cost and transaction cost. The price–quantity curve is depicted by $P(x) = P_0(e^{-\frac{b_x}{Q_{max}}}) + a$, where P_0 is the initial price and $\frac{b_x}{Q_{max}}$ is the discount rate, with two characteristics.

1. Due to the price sensitivity being different for different suppliers, some suppliers may lower the price rapidly when orders increase, but some may not. In addition, there is always a lowest price that the supplier can accept, even if the ordering quantity further increases. Therefore, the price–quantity curve is exponential.
2. The quantity available from a supplier has an upper limit, $[B_j]$, which is the current inventory or the maximum capacity that a company can produce before the due date. This also means that different suppliers have different levels of saleable quantities.

During the system operation, the intermediary receives orders from buyers in a certain time frame. The information regarding orders is combined into an aggregate order that includes ordering quantities, purchasing criteria, suppliers' locations, and so on. The aggregate order is assigned to the mobile agent, and the suppliers' addresses are arranged as the itinerary of the agent. If k is the number of individual orders, then the

aggregate order is *agg_req* (P^A, I^A, S^A, $QoS1^A$, $QoS2^A$, ...), where P^A is the averaged unit product price

$$P^A = \frac{\sum_{i=1}^{k}{}_{i\in B} P_i^B \times I_i^B}{\sum_{i=1}^{k}{}_{i\in B} I_i^B},$$

I^A is the total demand,

$$I^A = \sum_{i=1}^{k}{}_{i\in B} I_i^B,$$

S^A are the candidate suppliers, $S^A = S_1^B \cup S_2^B \cup S_3^B \ldots \cup S_k^B$, and $QoS1^A$ is the supplementary criteria (there may be more than one, or none). Again, the estimates from suppliers need to meet the criteria of $P^A \geq P_{min}^S$ at the filtering stage.

Generating the estimate in a supplier is complicated because the aggregate quantity ordered is normally larger than that which a single supplier can provide. Therefore, the maximum suppliable quantity, $[B_j]$, is assumed as the current inventory of a supplier, and the price is decided by the suppliable quantity of the price–quantity curve. Another issue is that a single order may find the best supplier, price, and quantity (S*, P*, I*) for its own needs, but the aggregate order finds a set of best-chosen suppliers, which may not be the best choice for an individual. The product purchasing policies that are used in this study for finding the best combination of suppliers are discussed in the following section.

4. System implementation

To implement an agent-based electronic marketplace with a many-to-many buyer–supplier relationship, a prototype system is developed following four phases: (1) develop the mobile agents and database, (2) develop buyer servers, the intermediary server, and supplier servers, (3) perform the simulation experiment, and (4) analyze the simulation results. The descriptions of these phases follow.

4.1. Develop the mobile agents and database

This study uses Java (JDK1.2) as the programming language, Aglets 1.1 SDK Beta3, from IBM's Tokyo laboratory, for the mobile agent interface, and Oracle 8*i* for the back-end database that records historical transactions. The Oracle transaction database is linked to a thin client Java driver by JDBC. A thin client can provide faster execution time than the Oracle Call

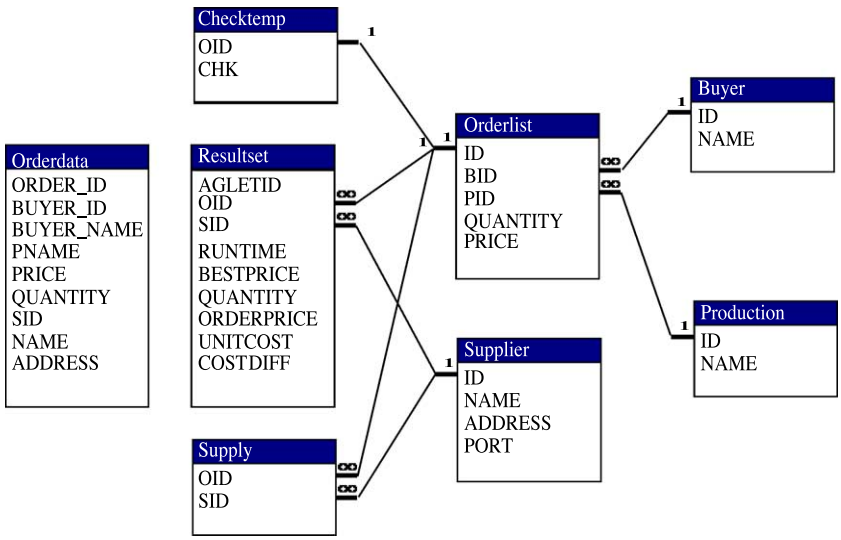

Fig. 4. The entities and relationships in the database of the intermediary agent.

Interface with Oracle Net8 service. MyDb, written with Oracle JDeveloper 3.2, is used to edit and query data. The database table is shown in Fig. 4.

4.2. Develop buyer servers, intermediary server, and supplier servers

In the electronic marketplace, the buyers issue orders and the intermediary server aggregates orders. Since the design does not allow negotiation between buyers and suppliers, the supplier's quotation is not sent to the buyers for evaluation. That is, in the system design, the same mobile agents will not be sent back to the buyers. Therefore, it is acceptable to locate both the buyer servers and the intermediary server in the same place. In fact, for this experiment, both the buyer servers and the intermediary server are located in the same building on the university campus. In contrast, the suppliers are located in seven different buildings, each housing a supplier server. The buildings are connected by an FDDI Backbone with a maximum transmission speed of 100 Mbps. The supplier databases, written in Microsoft Access, are linked by JDBC-ODBC programs and store information on the current inventories and price tables. In mobile-agent-based electronic commerce, the intermediary server plays a very important role. The intermediary server handles the heavy workload of managing the agent name service, aggregating demands, filtering unqualified buyers or suppliers, generating the itinerary of the mobile agent, matching orders, and keeping information about suppliers, such as product specifications. Therefore, a high-performance computer system should be used as the intermediary server.

4.3. Perform the simulation experiment

To perform the simulation, two algorithms for finding the best combination of suppliers were implemented, each with a different type of product purchasing policy.

4.3.1. Lowest price first (LPF)

Buyers purchase the products from the suppliers that are offering the lowest price. To implement the policy, the algorithm (1) sorts suppliers in ascending order based on the average price for the quantity [B_j]; (2) adds the first supplier into the selection list; (3) deducts the supplied quantity from the total order quantity; (4) includes the next supplier in the list if an ordering quantity remains; and (5) stops when the order is fully met.

4.3.2. Largest quantity first (LQF)

Buyers purchase the products from the suppliers that can provide the largest volume at a time. The algorithm (1) sorts suppliers in descending order by the quantity [B_j]; (2) includes the first supplier in the combination; (3) deducts the supplied quantity from the ordered quantity; (4) assigns the next supplier into the combination if an ordering quantity remains; and (5) stops when the order is fully met.

The simulation was implemented for 24 consecutive hours, which were divided into three sessions to represent the different types of network traffic. As the main focus of this study was to show the feasibility of using mobile agents in an electronic market the complexity of the simulation was minimized. For simplicity, each session was designed to process 100 batch orders randomly generated from the 20 orders listed in Table 2. The order parameters in Table 2 indicate the different price sensitivities that the buyers might have. The expected number of orders in each batch was set to 10. A total of 300 batch orders were generated during

Table 2
Possible order parameters used by the system for order generation

Quantity	Price
11,000	22.30
13,000	21.20
6000	26.00
8000	24.30
3000	28.40
16,000	21.00
11,000	22.30
10,000	22.90
5000	26.70
7000	25.10
15,000	21.30
12,000	21.70
8000	24.30
16,000	21.00
9000	23.60
4000	27.50
12,000	21.70
9000	23.60
14,000	21.30
7000	25.10

the simulation, and a total of 2844 order transactions were collected, giving approximately 10 order transactions per batch order. For simplicity, only one product was simulated in the system. The simulation was run three times with the same set of transactions. One used 2844 mobile agents for individual orders. The other two times were for aggregate orders with LPF and LQF policies. Each used 300 mobile agents. These mobile agents were sent to the supplier servers during the studied period. The system used Table 3 to simulate the different price-quantity sensitivities of the seven suppliers. The data regarding the execution time was collected and statistically analyzed.

4.4. Analyze the simulation results

The execution time of a batch order is obtained from the start time of the first order to the end time of the last order. The observation of the transactions in the three different sessions shows that the average execution time between 12:00 a.m. and 8:00 a.m. is the shortest and the variance is small. This is because the network traffic is lighter during this period. In contrast, the roaming time of the mobile agents is longest between 8:00 a.m. and 4:00 p.m. due to the heavy loading of the network. This implies that the performance of mobile agents will be affected by the network bandwidth. The simulation results also show that the aggregate order approach can improve system performance. The average execution time for the individual orders in a batch is 66.261 s. Compared with 31.978 s for the aggregate order, this represents a time saving of approximately 52%. It is noteworthy that without the electronic marketplace, the buyers' orders cannot be aggregated, and that without the mobile agents, the orders cannot find the best suppliers.

Table 3
Suppliers' price–quantity sensitivity

Quantity[a]	Supplier 1 ($)	Supplier 2 ($)	Supplier 3 ($)	Supplier 4 ($)	Supplier 5 ($)	Supplier 6 ($)	Supplier 7 ($)
1000	30.60	33.30	29.00	31.50	30.30	31.60	32.70
2000	29.80	31.30	28.10	30.20	29.10	30.00	31.00
3000	29.00	29.50	27.30	29.00	28.00	28.50	29.40
4000	28.30	27.90	26.60	27.80	27.10	27.30	28.00
5000	27.60	26.40	26.00	26.80	26.30	26.30	26.80
6000	27.00	25.10	25.40	25.90	25.50	25.30	25.70
7000	26.40	23.90	24.90	25.00	24.90	24.50	24.80
8000	25.80	22.90	24.50	24.20	24.40	23.80	23.90
9000	25.20	21.90	24.10	23.50	23.90	23.20	23.20
10,000	24.70	21.10	23.80	22.80	23.50	22.60	22.50
11,000	24.20	20.30	23.50	22.20	23.10	22.20	21.90
12,000	23.70	19.60	23.20	21.60	22.80	21.80	21.30
13,000	23.30	19.00	22.90	21.10	22.50	21.40	20.90
14,000	22.90		22.70	20.60	22.20	21.10	20.40
15,000	22.50		22.50	20.10	22.00	20.80	20.10
16,000	22.10		22.40	19.70	21.80		19.70
17,000	21.70		22.20	19.30	21.70		19.40
18,000	21.40		22.10		21.50		19.20
19,000	21.00				21.40		18.90
20,000	20.70				21.30		18.70
21,000	20.40				21.20		
22,000	20.20				21.10		
23,000					21.00		
24,000					21.00		
25,000					20.90		
26,000					20.90		
27,000					20.80		
28,000					20.80		

[a] The position of a supplier's lowest price indicates the deliverable quantity of that supplier.

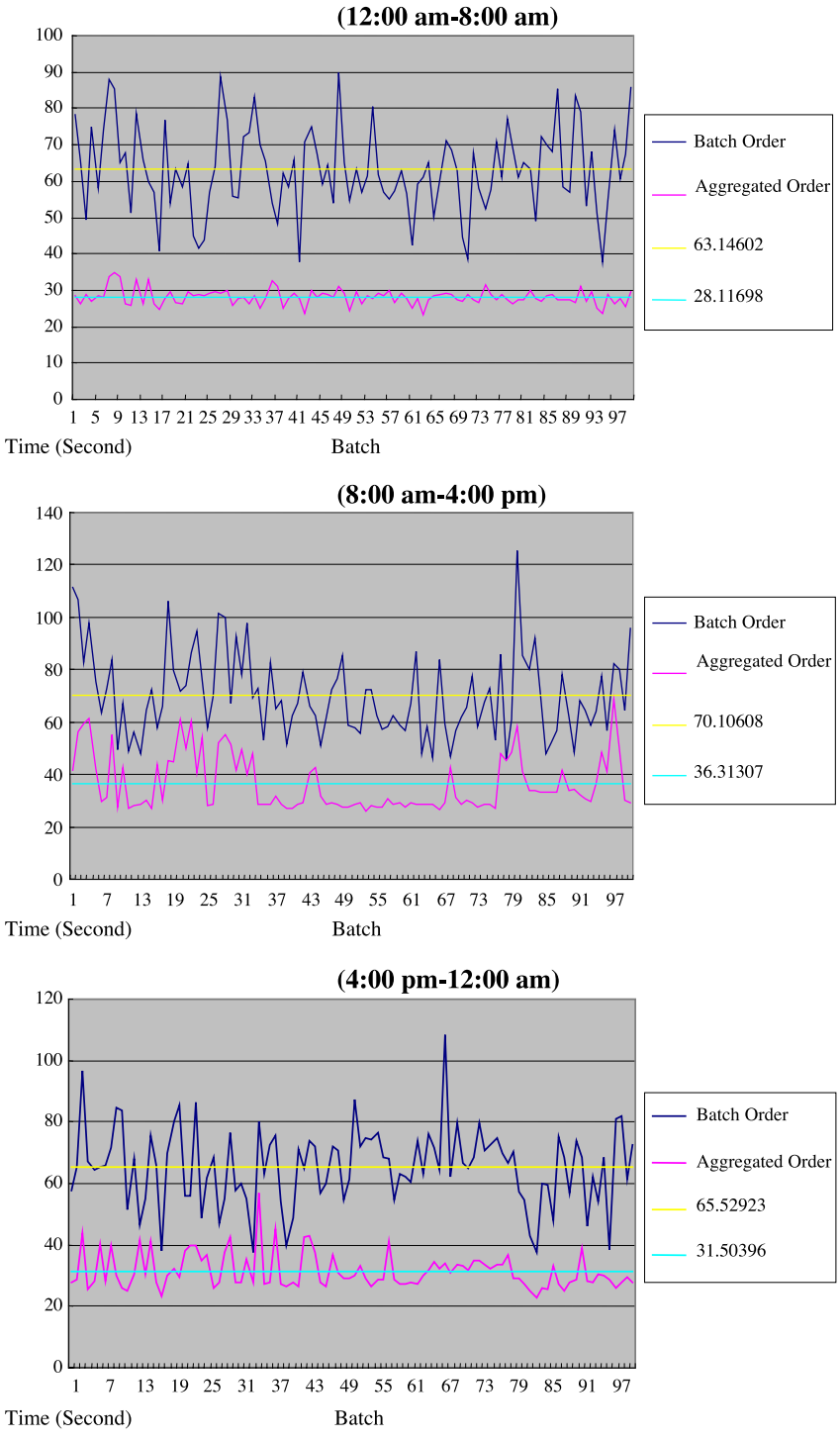

Fig. 5. The average execution time between batch orders and aggregated orders.

The comparison of the average execution time between batch orders and aggregated orders is shown in Fig. 5. The average execution time of the batch orders is obtained by averaging the time of reception of the first order and the time of completion of the last order. Fig. 5 shows that the execution time is smaller when the network traffic is lighter (from 12:00 a.m. to 8:00 a.m.) and is larger when the network traffic is heavier (from 8:00 a.m. to 4:00 p.m.). This implies that the performance of the mobile agent is related to the network traffic, and that the mobile agent can easily aggregate orders and increase the speed of matching orders, which results in a shorter execution time.

5. Conclusions and recommendations

The electronic marketplace finds product features, aggregates orders, and determines prices to match buyers and sellers. This study proposes a mobile-agent-based electronic marketplace that matches transactions while allowing data to be maintained by local buyers and sellers. The framework allows mobile agents to operate independently and asynchronously, and significantly reduces the bandwidth usage over the network. Hence, it provides the advantages of autonomy and independence to the participating companies and is suitable for the distributed architecture of electronic commerce. This is particularly true if the agents are implemented under a heterogeneous environment; the framework allows the minimum system integration effort to implement the agent-based system. Furthermore, this study demonstrates the ease of aggregating buyers' orders for improving price savings using intermediary mobile agents; a task that is very difficult, if not impossible, for individual local agents. The speed of order matching is significantly increased when the orders are aggregated using mobile agents. In practice, the object-oriented mobile-agent system can be implemented on heterogeneous platforms and interoperate with cross-platform applications.

The proposed agent-based marketplace can be implemented in either a B2C or B2B environment. In a B2C environment, this approach is more suitable for selling high-brand-recognized, less expensive, frequently purchased, or well-known packaged product items. In a B2B environment, a third-party-run intermediary agent server is needed before any buyers or suppliers can proceed with transactions. In either environment, the agent-based framework eases the heavy loading on the data storage and operations of the intermediary. However, although it has significantly improved on the conventional centralized architecture, it is also clear that when the system grows the bottleneck of the agent-based transactions will be the Agent Manger. For this reason, the system in this study was developed using the *JDBC Thin Driver* and *Java* to prevent the service of the intermediary from being suspended because of the decreased system performance. Therefore, the system architecture of the Agent Manager needs to be carefully designed.

Several further studies can be pursued. First, the Aglet from IBM provides only prototype-level functions. It is possible that the operations can be improved together with an improvement of the mobile agents for a comprehensive electronic marketplace infrastructure. Second, security was not fully implemented in the proposed electronic marketplace. Security issues, such as data security, access control, and data encryption are all of concern. Unlike data encryption that is taken care of by the system and transparent to the users, the design of both data security and access control are important to the success of an electronic marketplace. This is especially true when a higher degree of collaboration is involved, such as collaborative product development or vender-managed inventory. The issues of who can access the data and what privileges should be given need to be carefully considered. Third, this study mainly adopts the concepts of weak agents to develop the agent-based marketplace. As the environment becomes more complex, it is recommended that agent behavior should be expanded to include the capabilities of reasoning, adapting, learning, and negotiating. Studies of intelligent mobile agents have shown a possible way to a complicated and automatic environment. However, the level of success is far from satisfactory. Fourth, the marketplace can be further integrated with the back-end operations, such as supply chain management and product definition management, so that the procurement transactions can be coordinated with the internal information system as well as the production and manufacturing systems. This integration is feasible if the agent platform is built into the existing enterprise system. Fifth, certain challenges regarding mobile agent technology remain. The highly secure agent execution environment, the virus

controls mechanism, the life cycle support, and the transmission efficiency are awaiting further study [5,15]. Finally, this study used a simple simulation design to explore the performance of a prototype system of a mobile-agent-based electronic marketplace. The outcome of this simulation study is by no means conclusive. A more complete design of the simulation study is needed before any generalized conclusions can be drawn.

References

[1] Anonymous, Grasshopper Basics and Concepts: Release 2.2, IKV++, Berlin, Germany, (2001 March).

[2] Y. Bakos, The emerging role of electronic marketplaces on the internet, Communications of the ACM 41 (8) (1998 August) 35–42.

[3] G.P. Barbosa, F.Q.B. Silva, An electronic marketplace architecture based on technology of intelligent agents and knowledge, Lecture Notes in Computer Science 2033 (2001) 39–60.

[4] P. Bellavista, A. Corradi, C. Stefanelli, An integrated management environment for network resources and services, IEEE Journal on Selected Areas in Communications 18 (5) (2000) 676–685.

[5] M. Breugst, T. Magedanz, Mobile agents-enabling technology for active intelligent network application, IEEE Network 12 (3) (1998 May–June) 53–60.

[6] G. Cabri, L. Leonardi, F. Zambonelli, Mobile-agent coordination models for internet applications, IEEE Computer 33 (2) (2000 February) 82–89.

[7] F. Chatzipapadopoulos, M. Perdikeas, L. Venieris, Mobile agent and CORBA technologies in the broadband intelligent network, IEEE Communication Magazine 38 (6) (2000 June) 116–124.

[8] P. Dasgupta, N. Narasimhan, L.E. Moser, P.M. Melliar-Smith, MAgNET: mobile agents for networked electronic trading, IEEE Transactions on Knowledge and Data Engineering 11 (4) (1999) 509–525.

[9] T.C. Du, E.Y. Li, A.P. Chang, mobile agents in distributed network management, Communications of the ACM 46 (7) (2003) 127–132.

[10] S. Feldman, E-business: electronic marketplace, IEEE Internet Computing 4 (4) (2000) 93–95.

[11] A. Jain, M. Aparicio, M. Singh, Agents for process coherence in virtual enterprises, Communications of the ACM 42 (3) (1999 March) 62–69.

[12] J.J. Jung, G.S. Jo, Brokerage between buyer and seller agents using constraint satisfaction problem models, Decision Support Systems 28 (4) (2000 June) 293–304.

[13] D. Lange, M. Oshima, Programming and Deploying Java Mobile Agents with Aglets, Addison-Wesley, Reading, MA, 1998.

[14] T. Sandholm, Q. Huai, Nomad: mobile agent system for an internet-based auction house, IEEE Internet Computing 4 (2) (2000 March–April) 80–86.

[15] D. Schoder, T. Eymann, The real challenges of mobile agents, Communications of the ACM 43 (6) (2000 June) 111–112.

[16] K.M. Sim, R. Chan, A brokering protocol for agent-based E-commerce, IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews 30 (4) (2000) 474–484.

[17] M.J. Wooldridge, N.R. Jennings, Agent theories, architectures and languages: a survey, Lecture Notes in Computer Science 890 (1995 February) 1–39.

Timon C. Du received his BS degree in Mechanical Engineering from the National Chung-Hsing University, Taiwan, in 1989. He obtained his Master's and PhD degrees in Industrial Engineering from the Arizona State University. Currently, Dr. Du is an Associate Professor at the Chinese University of Hong Kong, Hong Kong, and director of MSc in E-Business Management Program. His research interests are in e-business, data mining, collaborative commerce, and semantics webs.

Eldon Y. Li is Professor and Dean of College of Informatics at Yuan Ze University in Taiwan. He was a professor and the Coordinator of MIS Program at the College of Business, California Polytechnic State University, San Luis Obispo, California, USA. He visited the Department of Decision Sciences and Managerial Economics at the Chinese University of Hong Kong during 1999–2000. He was the Professor and Founding Director of the Graduate Institute of Information Management at the National Chung Cheng University in Chia-Yi, Taiwan. He holds a PhD from Texas Tech University. His current research interests are in human factors in information technology (IT), strategic IT planning, software engineering, quality assurance, and information and systems management. He is the Editor-in-Chief of the *International Journal of Electronic Business*, the *International Journal of Internet and Enterprise Management*, and the *International Journal of Internet Marketing and Advertising*.

Eric Wei received both his BS and MS degrees in Industrial Engineering from Chung Yuan Christian University in 1998 and 2000, respectively. Mr. Wei is currently working in industry. His research interests include information technology, database management, and electronic commerce.