Final Report  
An Electronic Commerce System  
Term Project for CS5204 Advanced Operating Systems 

Fangxing Li and Wei Sun  
Department of Electrical and Computer Engineering  
Virginia Tech  
Blacksburg, VA 24060

Abstract:

An electronic commerce system is designed and implemented in this project. This system consists of a client, two stores and a bank. The client can view and order products from either of two stores. The product information will be send to client for a view operation. A valid order operation will update product database and bank database.

This distributed system is implemented with Java Remote Method Invocation (RMI) mechanism. Java monitor model of synchronized method is employed to implement the reader-writer problem with writer priority. For efficiency in updating product databases, a two-phase commit protocol is used here. Also, Java Swing classes are used to implement the graphic user interface (GUI) at the client site.

The primary objective of this project is to explore the advanced concepts in distributed system such as remote method invocation, synchronization with object model and two-phase commit protocol.

1. System Description

This work implements a simple electronic commerce system. The system consists of four different machines for a client, two stores, and a bank. The system allows the client to view and purchase items from either store. As part of a purchase order, the client provides an "account number" which the store will verify by contacting the bank. Before authorizing the expenditure, the bank will seek confirmation from the client that the amount and the store are acceptable. The interaction between the client and the store should be asynchronous. That is, the client should not block while waiting for the store to reply to its requests to view or purchase items. This allows the client to conduct business with several stores simultaneously. Network channel is assumed to be secure here.
The architecture of this system is described as in Figure 1. Details will be given in the following section.

2. System Design

2.1. Client

Since the client needs to interact with users, a graphic user interface (GUI) is implemented at the client site to make it user-friendly. The client also needs to allow users to query and order products from either of two stores. Operations of queries and orders are asynchronous at the client site. When the client sends an order to one of the store site, the client needs to provide an account number, which the store will verify by contacting the bank. Before authorizing the expenditure, the bank will seek confirmation from the client to make sure that the amount and the store are correct. So three major tasks are needed at the client site.

- Graphic user interfaces
- Remote interaction between the bank and the client
- Asynchronous query and order between the client and the store
To make the online shopping easily, the user will be guided through some simple windows. When the client enters the online shopping system, there is a welcome window to allow the client to select one of the two stores. After selection, the product-listing window will be shown so that the user can view and order products. Remote interaction confirmation between the bank and the client, and the asynchronous query and order between the client and the store are all visualized using some simple dialogs. All those graphic user interfaces are implemented with JAVA Swing.

Before authorizing the expenditure, the bank will seek confirmation from the client about the amount and the store site. The client and the bank are at different locations, so this interaction is a remote interaction. This remote interaction is implemented at the client site with Java Remote Method Invocation (RMI) mechanism. In this remote interaction, the client site acts as a server to the bank. The client implements a remote method userConfirm(). So the bank can get a remote reference to this remote method in the client and get the confirmation about the amount and the department.

Since the client should not block while waiting for order result, this electronic commerce system supports asynchronous interaction between the client and the store. In other words, the client can concurrently view and wait for order result. This function is implemented through threads. After the client submits an order, the client site starts a new thread immediately by calling start() to connect the store about the order result.

2.2 Store

Store sites will implement two remote methods of query and order. The store should allow concurrent query operation, and prohibit any concurrent operations of query/order or order/order. Hence, data integrity will be guaranteed. Also, order operation should take higher priority. A Java monitor model is employed here to implement mutual exclusions.

The query is quite straightforward and simple. Whenever a query operation is invoked, it will first obtain the reading-lock. Once getting the lock, it will transmit the current database at the store site to the client. If the lock is not available, it will wait. This is internally implemented by Java synchronized method. The lock will be released at the end of the query.
Order operations are much more complicated than query operations. A two-phase commit protocol is used here to achieve better performance. The detail is given as follows.

- An order will be checked to make sure that enough products are available. If so, writing-lock will be obtained to update the database. Otherwise, return a message to the client. After leaving this phase, writing-lock will be released.

- Then, the store will remotely invoke the bank’s verify() method to make sure that the order is a valid one. If verify() returns false, this shows the order should not be committed. Then, the store will undo the updating of the product database. Otherwise, the order is valid and no undo is executed. A writing-lock must be obtained at the beginning of undo operation and released at the end.

This is believed to improve system performance. Since verify() will call userConfirm() at the client site, verify() could take long delay because userConfirm() involves user interactions. So, we should not put verify() into our mutual-exclusive database updating. Otherwise, the database could be blocked for long time. This explains why verify() must be separated from commit and undo.

Commit must be done during phase one. If it is done after verify(), this will make the database not consistent. For example, assume there are 3 books available and a user orders 2 books. If no commit before the verify(), there will be still 3 books at the store. If right before verify(), another user makes another order of 2 books. Then, 4 books will be ordered while there are only 3 books. The data cannot be consistent. This is why we need such operation sequence like commit $\rightarrow$ verify $\rightarrow$ undo (if necessary), in which commit and undo are protected by writing-lock.

2.3 Bank

As stated in above section, the bank site will implement a remote method verify() which will be used by the stores. This method will verify that the user’s account has enough money to make the purchase. If not, the store will be notified to cancel the transaction. If there is enough
money in that account, the bank will invoke a remote method, userConfirm(), at the client site to make sure that the user really wants this purchase. If userConfirm() returns false, verify() will notice the store to cancel (undo) the transaction.

Concurrent access to the bank from the same account is not allowed. Assume one user has $100 in his/her account and he/she makes two purchases of $60 each. Concurrent read will make the bank think the user has money for both purchases. Also, a Java monitor model with synchronized method is employed here.

2.4 Diagrams

Two UML diagrams are shown in this section to depict the system implementation. In Figure 2, the dependence of three packages is shown. Each package provides an interface denoted by a circle. Solid lines stand for provider relations and dashed arrow-lines stand for depend-on relations.

![Figure 2 Diagram of Packages and Interfaces](image)

The sequence of the system interaction is depicted with a UML sequence diagram as shown in Figure 3, which is self-explainary.
Figure 3 UML Sequence Diagram in a Logic View for a Query Activity Followed by an Order Activity
3. Summary

The objective of this project is fulfilled through the design and implementation of this e-commerce system. Four major achievements are reached in this project.

- Java RMI mechanism for distributed programming
- Java monitor model with synchronized method to keep data integrity
- Graphic user interface(GUI) design with Java Swing classes
- Techniques for teamwork and trouble-shooting

As a course project, this system is far from perfect. Followed are the suggestions for future improvements for this system.

- The network communication channel is assumed to be secure here. However, in the real world, this may not be true. So, some protection mechanisms need to be provided for security reason, particularly about the user account information.
- At store and bank sites, the system should provide some facilities for system administrators to manage the data.