Andrew Robinson

Marcus Karl

Operating System Concepts

CS 4348.002 – S16

25 Apr 2016

**Project #3: Network Communication Using Sockets**

**Purpose**

The purpose of this project was to implement a multithreaded message server using sockets. Additionally, when implementing the programs in java the use of the ServerSocket and Socket classes were required along with an understanding of mutual exclusion. Also, the running platform was UNIX command line on UTD’s cs1 or cs2 servers.

**Project Implementation**

The project was written in Java and designed to be compiled and run through Java command line. The project was broken up into two programs, a server side program, and a client side program. Additionally, communication between the two programs is done through the means of object passing, not strings or bytes. MsgCommObj class was created and serialized for use as the object being passed. The MsgCommObj class was given five fields of data and methods to retrieve and set the data fields. The data fields are a to user name, from user name, message, option, and date & time field. The server side program binds to a given port passed as an argument during program startup. After words, the server listens for incoming connections and automatically passes incoming connections to a new thread for handling interactions with the client. The server side program uses a thread pool for handling client threads, which dynamically allocates threads as needed depending on the number of clients connecting. The server side provides six basic functions for the client. The functions are: return a list of all know users, return a list of all connected users, send a message to a specific user, send a message all known users, send a message to all connected users, and send all stored messages to user. All server threads have the same instance of the message server passed to them upon initialization, and the instance is used for interaction of client mailboxes and message passing on the server. Semaphores are used for protection of adding new users, and storing or retrieving messages from a user’s Linked List which stores all messages and functions as the mailbox.

Client side program interaction sends and retrieves messages from server. When retrieving messages from the server, the client adds messages to a local Linked List for viewing the messages one at a time. The client program has the capability to view a specific message and reply to the message before viewing any other stored messages. After viewing a message, it is deleted and unable to be recovered.

**Personal Experience**

Implementing the project in Java was straight forward. Two parts of the project were challenging. First was a misunderstanding of ServerSocket and the belief that a client would need a new port to connect to in order to free up the listening port for another client to connect to. Discussion with the professor cleared up confusion regarding this issue. Second, was the actual passing of an object back and forth from client to server. An issues was encountered when actually sending the message, MsgCommObj was not serializable. However, after research it was discovered that implementing Serializable in the class and adding a version ID to the class allowed the object to be passed without any problems as the other side of the connection now knew that it had the correct class.

**Team Members Task**

ServerMain.java - written by Marcus Karl/Andrew Robinson

ClientCommThread.java - written by Marcus Karl

Messages.java - written by Marcus Karl/Andrew Robinson

MsgCommObj.java - written by Marcus Karl/Andrew Robinson

ClientMain.java - written by Andrew Robinson

Design.doc - written by Andrew Robinson

Summary.doc - written by Marcus Karl

Team\_members.txt - written by Andrew Robinson

Readme.txt - written by Marcus Karl