



OCTOBER 2017

“PROJECT -1 REPORT”

Submitted for the course

Of

DISTRIBUTED SYSTEMS

Under the guidance of

Dr. JIA RAO

Submitted by

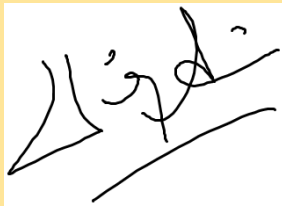
WASIQ ALI ABBASI	–	1001583762
ADITHYA VADLAMANI	–	1001417682

DECLARATION

We hereby declare that, in our project-1 submitted by us, we have neither given nor received any unauthorized assistance on this work.

Sign

Date

A handwritten signature in black ink, appearing to be 'Wasiq Ali Abbasi', written on a white background.

10/01/2017

(Wasiq Ali Abbasi)

A handwritten signature in blue ink, appearing to be 'Adithya Vadlamani', written on a white background.

10/01/2017

(Adithya Vadlamani)

IMPLEMENTATION DETAILS

The problem that was given to us had multiple parts. We divided the task into small sub-tasks and then implemented each sub-task separately. Following is the step-by-step process of how we solved the problems:

1. We made a simple server that created a socket on localhost port: 8080 and listened to any incoming connections. When we started, we just made this server single-threaded
2. We made a simple client which connected to the localhost server on port 8080
3. We initially just implemented the command protocols between client and server. The client asks the user to input the command which the client then communicates to the server and the server send acknowledgement of receiving that command back to the client
4. After that we implemented the file based operations. Following are the steps that are taken against each command:
 - a. **Upload:** The client reads the file locally and send it to the server through the socket connection. The server then implements the changes at its end accordingly
 - b. **Download:** After receiving download command, the server reads its file locally and sends the date to the client through the socket connection. The client then implements the changes at its end accordingly
 - c. **Delete:** Upon receiving the delete command, the server removes the file from its end
 - d. **Rename:** When the user enters the rename command at client-end, the user is then prompted to enter the new name. The new name is then sent to the server through the socket, which the server then implements at its end
5. After this, the server was converted to multi-threaded server by implementing a worker thread. After the client connects to the server, the server creates a handler thread that handles the command sent by the client, and the server itself again goes to listening mode on `socket.accept()`;
6. After this, the dropbox synchronization feature was implemented at client-end. When the client starts and the dropbox feature is enabled, a thread was created which is put to sleep. This thread wakes up every 15 seconds to check

if the file has been deleted, altered or created at client-end. If so, the appropriate command is sent to the server.

KEY LEARNINGS

In this project, we learnt about client server model and its complexities. We have learnt the synchronization techniques needed between a client and server. Also, in case of a multi-threaded environment, we learnt about synchronization among different threads.

ISSUES ENCOUNTERED

- 1) Synchronization between client server in a single threading environment.
- 2) The synchronization became more complicated when the environment became multi-threaded and as the number of clients grew.

CHALLENGES IN CONVERTING SINGLE-THREADED TO MULTI-THREADED

- 1) As a single shared file was being accessed by multiple threads in a multithreaded environment, keeping a synchronized version of the file was a challenge.
- 2) The complexity increases as the number of threads grow in a multithreaded environment.

WHY DISTRIBUTED LOCKING IS NECESSARY FOR MULTI-THREADED SERVER

Locking is necessary when a single shared resource is being accessed by multiple threads. In case of our project, this shared resource was a single file that was being read both at the client as well as server side by multiple threads. If multiple concurrent accesses are given to the shared resource, it can cause issues which may leave our data inconsistent.