SYSC 3303 – TFTP PROJECT

Winter 2018

Group 6

Noor Ncho
Benjamin St.Pierre
Jozef Tierney
Omar Dawoud

Table of Contents

Work Breakdown

Iteration #1

Jozef Tierney- Added functionality to Client.java and ThreadRunner.java to take in user input and created the README.txt file.

Omar Dawoud- Created the PacketPrint.java and packetFile.java files and UML diagram to describe the system.

Ben St. Pierre- Created the MasterServer.java and ThreadRunner.java files using ideas from assignment 1. Created Thread functionality to Client.java, ErrorSim.java and Server.java. Also added Javadoc and comments to the classes.

Noor Ncho- Created the UCM Diagram for the program.

Iteration #2

Omar Dawoud – Worked on the File Reading and Writing actions, and created the UML diagram

Benjamin St. Pierre – Created the README.txt and Work breakdown files. Made updates to classes from the previous iterations and submitted all files for this iteration.

Noor Ncho – Created the timing diagrams and the ErrorHandler.java, that is responsible for checking for any errors before transmitting files.

Iteration #3

Jozef Tierney - Worked on getting user Input in ThreadRunner.java and Client.java as well as File Reading and Writing action. Created the iteration README.txt file and submitted all files for this iteration.

Omar Dawoud – Worked on the File Reading and Writing actions, and created the UML diagram

Benjamin St. Pierre – Created the terminal windows for each of the for the classes that need it and added synchronization to the project. Also updated PacketPrint.java and added comments and Javadoc

Noor Ncho – Created the new timing diagram for error code 1, 2 and 3. Also updated ErrorSim.java.

Iteration #4

Jozef Tierney-Added Javadoc and comments to several files of the project and worked on the file reading and writing functionality. Also responsible for submitting all the work for this iteration.

Omar Dawoud- Updated the UML for this iteration and the README.txt file for this iteration.

Ben St.Pierre- Setup the functionality to allow packets to send the changes that occur in the client/ errorSim/ server/ masterServer, made changes to PacketPrint.java, and created the work Breakdown.

Noor Ncho - Created new timing diagrams for error code 5 and 6, and updated ErrorSim.java to include these errors.

Iteration #5

Noor Ncho - Created the final submission Report

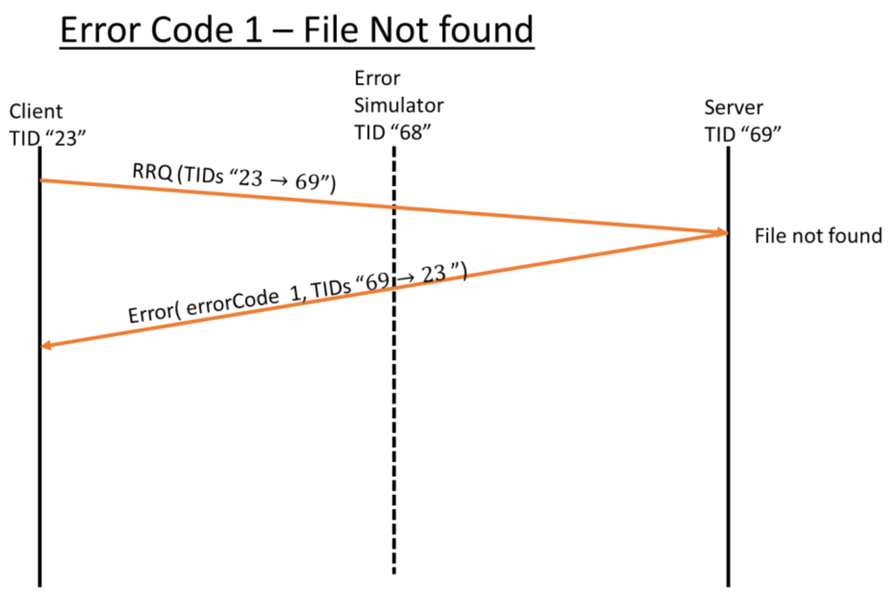
Ben St.Pierre - created read and write process methods for client and server, worked on multi computer functionality, updated javadoc and comments, and code cleanup

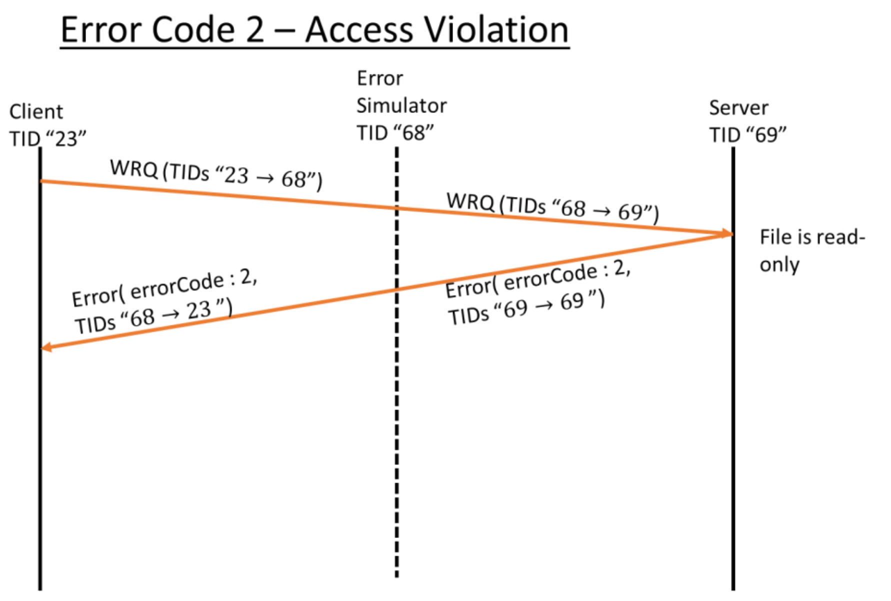
Jozef Tierney-Fixed errors pointed out in project presentation, worked on multi computer functionality, updated readme

Omar Dawoud - Updated the UML diagram

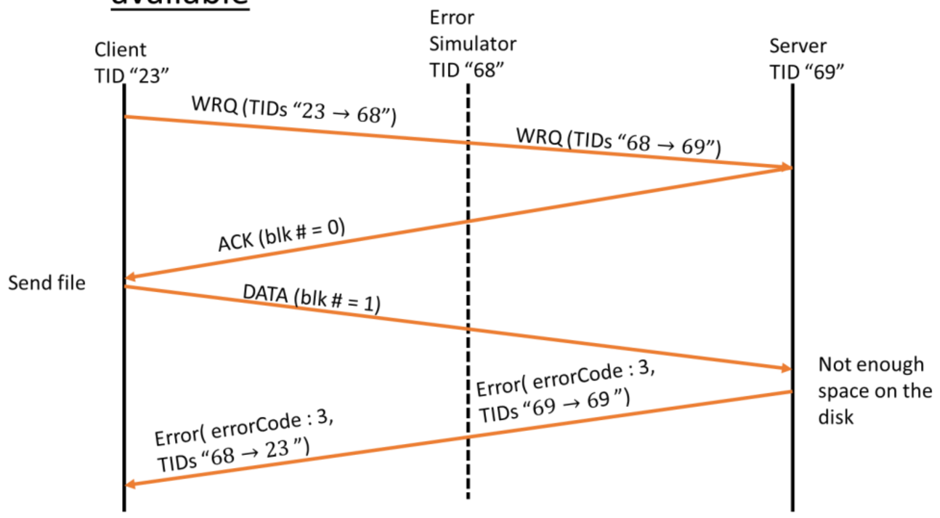
Diagram

Timing Diagrams

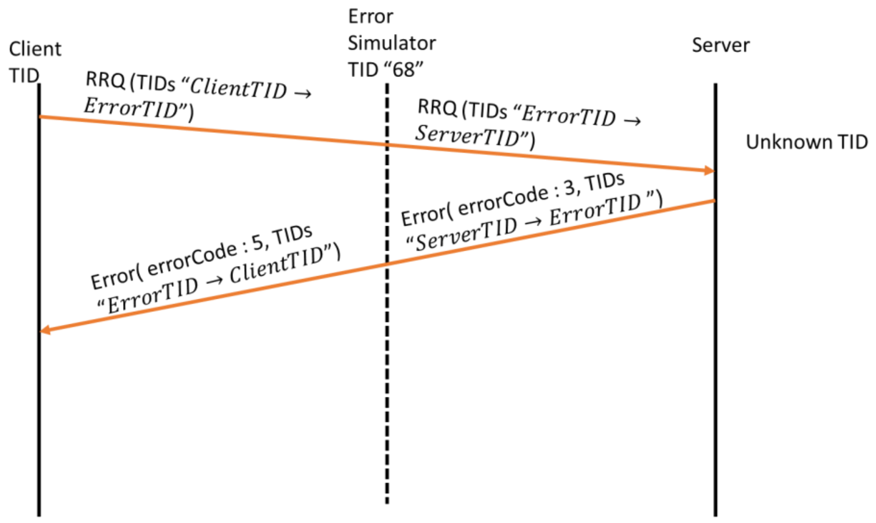




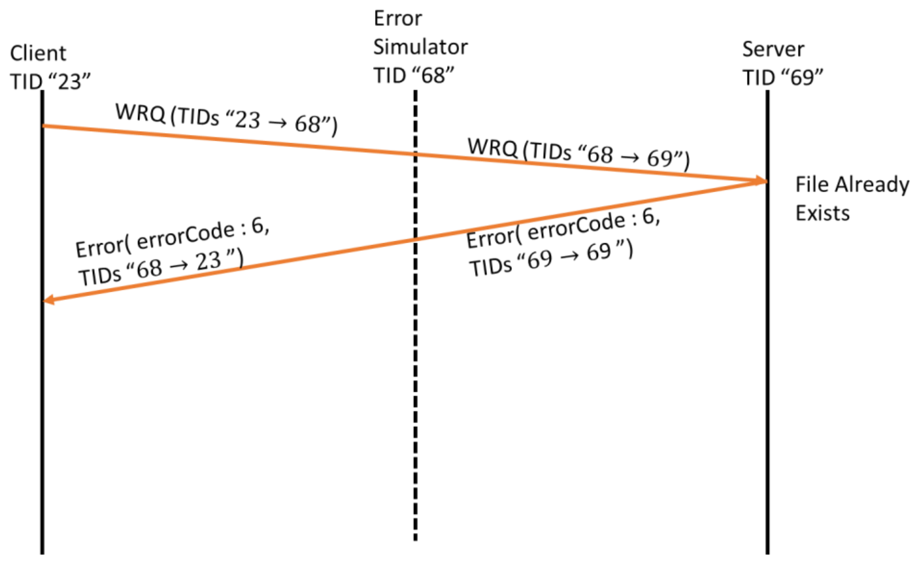
Error Code 3 – Disk Full or Not enough space available



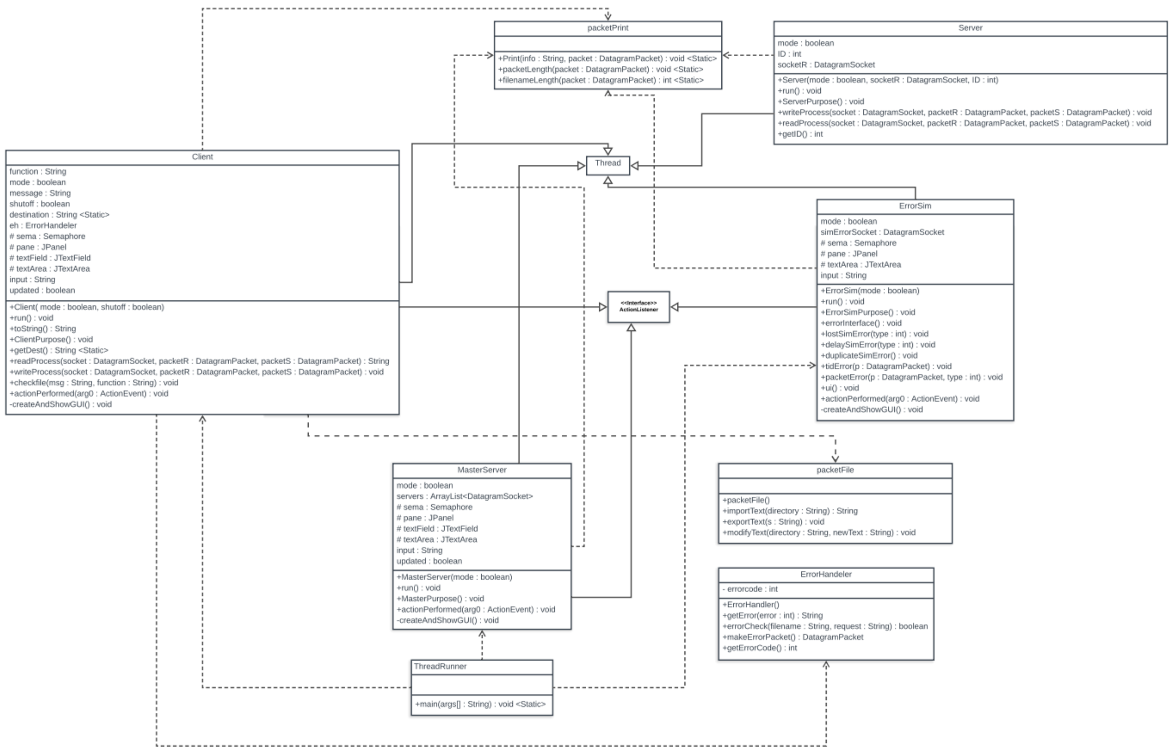
Error Code 5 – Unknown Transfer ID



Error Code 6– File Already Exists

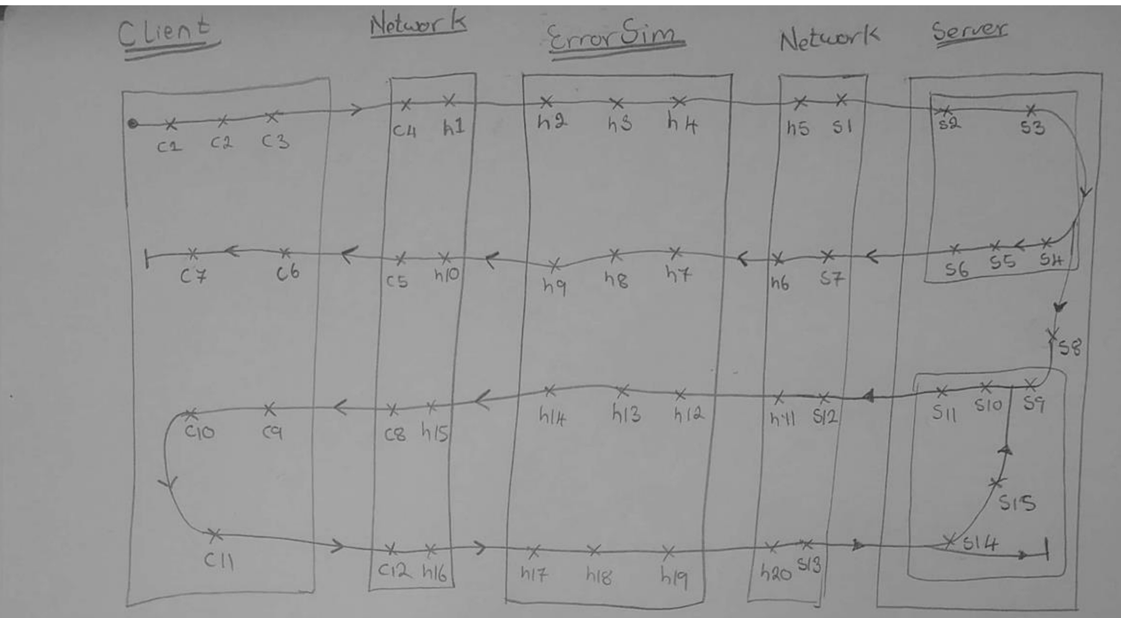


UML Diagram



packetPrint
+Print(info : String, packet : DatagramPacket) : void <Static>
+packetLength(packet : DatagramPacket) : void <Static>
+filenameLength(packet : DatagramPacket) : int <Static>
Server
mode : boolean
ID : int
socketR : DatagramSocket
+Server(mode : boolean, socketR : DatagramSocket, ID : int)
+run() : void
+ServerPurpose() : void
+writeProcess(socket : DatagramSocket, packetR : DatagramPacket, packetS : DatagramPacket) : void
+readProcess(socket : DatagramSocket, packetR : DatagramPacket, packetS : DatagramPacket) : void
+getID() : int
Client
function : String
mode : boolean
message : String
shutoff : boolean
destination : String <Static>
eh : ErrorHandeler
# sema : Semaphore
# pane : JPanel
# textField : JTextField
# textArea : JTextArea
input : String
updated : boolean
+Client(mode : boolean, shutoff : boolean)
+run() : void
+toString() : String
+ClientPurpose() : void
+getDest() : String <Static>
+readProcess(socket : DatagramSocket, packetR : DatagramPacket, packetS : DatagramPacket) : String
+writeProcess(socket : DatagramSocket, packetR : DatagramPacket, packetS : DatagramPacket) : void
+checkfile(msg : String, function : String) : void
+actionPerformed(arg0 : ActionEvent) : void
-createAndShowGUI() : void
Thread
<<Interface>>
ActionListener
ErrorSim
mode : boolean
simErrorSocket : DatagramSocket
# sema : Semaphore
# pane : JPanel
# textArea : JTextArea
input : String
+ErrorSim(mode : boolean)
+run() : void
+ErrorSimPurpose() : void
+errorInterface() : void
+lostSimError(type : int) : void
+delaySimError(type : int) : void
+duplicateSimError() : void
+tidError(p : DatagramPacket) : void
+packetError(p : DatagramPacket, type : int) : void
+ui() : void
+actionPerformed(arg0 : ActionEvent) : void
-createAndShowGUI() : void
MasterServer
mode : boolean
servers : ArrayList<DatagramSocket>
# sema : Semaphore
# pane : JPanel
# textField : JTextField
# textArea : JTextArea
input : String
updated : boolean
+MasterServer(mode : boolean)
+run() : void
+MasterPurpose() : void
+actionPerformed(arg0 : ActionEvent) : void
-createAndShowGUI() : void
packetFile
+packetFile()
+importText(directory : String) : String
+exportText(s : String) : void
+modifyText(directory : String, newText : String) : void
ErrorHandeler
- errorcode : int
+ErrorHandler()
+getError(error : int) : String
+errorCheck(filename : String, request : String) : boolean
+makeErrorPacket() : DatagramPacket
+getErrorCode() : int
ThreadRunner
+main(args[] : String) : void <Static>

UCM Diagram



C1 - forms request
C2 - set type of request
C3 - create datagram
C4 - send datagram
C5 - receive datagram
C6 - extract message
C7 - print message
C8 - recieves datagram
C9 - extract data
C10 - form ACK
C11 - forms datagram
C12 - sends datagram

h1 - receives datagram
h2 - extract request
h3 - forms message
h4 - forms datagram
h5 - sends datagram
h6 - receive datagram
h7 - extract message
h8 - form message
h9 - forms data gram
h10 - sends data gram.
h11 - receives datagram
h12 - extract data
h13 - forms data
h14 - forms datagram
h15 - sends datagram
h16 - receive datagram
h17 - extract ACK
h18 - form ACK
h19 - form datagram
h20 - send datagram

S1 - receives datagram
S2 - extra message
S3 - verifies type of request
S4 - create new datagramSocket
S5 - form response
S6 - form datagram packet
S7 - sends datagram
S8 - creates new thread
S9 - creates new Socket
S10 - forms ACK
S11 - forms data gram
S12 - sends data gram
S13 - receives datagram
S15 - extract ACK

Set-up & Testing Instructions(README.txt)

1. Launch project in eclipse
2. Execute the Main function in threadRunner.java
3. Run the project in and follow it prompts in the GUI windows
4. Select a mode ‘Quiet’ or ‘Verbose’ or enter ‘Quit’
    a. Verbose: Outputs the details of the packet during execution of the program
    b. Quiet: Details remain hidden in the program.
    c. Quit: Shuts down the program
5. Enter how many clients should be initialise. The minimum number being 1.
6. Several windows will open at this point. 1 server, 1 errorSim and a specified number of clients from step 5.
    a. Select what type of function you would like to errorSim to run in (Normal/ ErrorSim/ Quit). As an intermediate that does not change any of the packets or a simulator for different possible errors.
    b. Client(s) will continue to prompt inputs form the user for specifying the request type, (read/ write/quit)
    c. Finally, the server functions the same function as errorSim.
7. Next in the Client(s) you are prompted to declare whether you desire a read or write packet.
8. When prompted, input the directory including the file name (a .txt file) into the text window. The file must be in the project directory. A sample file called test.txt has been created with the message hello world in it.
    a. Eg – [project folder path]\test.txt
9. After a transfer is done the process begins again.

Note: Quit can be entered at any time to shut down servers and close the program. Alternatively closing one of the terminal windows will also shut down the thread.