TFTP-File-Transfer-System

SYSC 3303 – Team/Group 5

# Team Member(s)

* Harshan Anton (100962136)
* Aly Khan Barolia (101008206)
* Arsalan Sadiq (101020133)
* Jeff Tudor (100891259)

Table of Contents

[Team Member(s) 1](#_Toc510529407)

[Introduction 3](#_Toc510529408)

[Included Files in submissions 3](#_Toc510529409)

[ Client.java 3](#_Toc510529410)

[ ClientConnectionThread.java 3](#_Toc510529411)

[ IntermediateHost.java 3](#_Toc510529412)

[ ThreadedServer.java 3](#_Toc510529413)

[Breakdown of responsibilities of each team member for each iteration 4](#_Toc510529414)

[Execution/Termination Instructions 5](#_Toc510529415)

[**Diagrams** 6](#_Toc510529416)

[UCMs for read/write including the error simulator (from iteration #1) 7](#_Toc510529417)

[Timing diagrams showing all the error scenarios 9](#_Toc510529418)

[UML class diagram 14](#_Toc510529419)

# Introduction

The goal of this iteration is to create multiple clients, an Intermediate Host, and a server to support steady-state file transfer. We designed and implemented a file transfer system based on the TFTP specification (RFC 1350). The system will consist of TFTP client(s) running on one computer, an intermediate Host, and a multithreaded TFTP server on separate computers.

Updates in version numbers:

**Starting Version:** Established Connections for File Transfer without Error Detection and Correction

**Version 1.0:** Implemented of File Transfer without Error Detection and Correction

**Version 2.0:** Added I/O Error Handling (ERROR Packets 1, 2, 3, 6)

**Version 3.0:** Added Network Error Handling (Timeout/Retransmission)

**Version 4.0:** Added TFTP Packet Format Errors (ERROR Packets 4, 5)

**Version 5.0:** Implemented of File Transfer between Different Computers

# Included Files in submissions

* Client.java
* ClientConnectionThread.java
* IntermediateHost.java
* ThreadedServer.java
* Complete UML diagram
* UCM diagrams for both RRQ and WRQ
* Timing diagrams

* Client.java  
  The client communicates to the server through the intermediate host and can send either of the following packets: RRQ, WRQ, ACK and DATA. This is the class that the user interacts with to specify which kind of request will be sent to the server. The client will either write out a file in its own system out to the server or read in data from server and write it into a file of its own.
* ClientConnectionThread.java   
  This class does the work for the server. This class spawns a new thread for any new client connections made to the server. After communication is established between server and client, this thread takes over and deals with all future transactions.
* IntermediateHost.java  
  The intermediate host also known as the Error Simulator, is a class that exists between the Client and Server class. All packets are gone through this class to check for duplication, missing packets, error packets, etc. The intermediate host simply funnels ACK and DATA packets from the client to the server, and vice versa.
* ThreadedServer.java  
  This class represents the server. This class has a dedicated port which is port 69. The server listens to this port and whenever a client contacts it (through the intermediate host), the server spawns a thread of class ClientConnectionThread, dedicated to dealing with transactions from that client. There will only be one server, with a thread for each connecting client.

# Breakdown of responsibilities of each team member for each iteration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Team Members | Iteration 1 | Iteration 2 | Iteration 3 | Iteration 4 | Iteration 5 |
| Harshan Anton | Tried to make the Iteration run successfully by creating a packet and socket and linking them with the IP | Made the server multi-threaded and handling of packets received at thread and fixed errors on intermediate host. | Worked on error packets like Duplicate, delayed and lost in the intermediate host | Worked on error packets based on illegal TFTP operation in the intermediate host and created the diagrams | Fixed Transfer between different computer, made sure all the iteration 1 – 4 works smoothly |
| Aly Khan Barolia | Gave the code from Assignment 1 and implemented TFTP and debugged on imperative problems | Fixed bugs on imperative problems and created Diagrams and documentation | Made sure the packets contains the correct messages and created the diagrams | Worked on error packets based on illegal TFTP operations based on the client side | Fixed Bugs on the client side and server side and made last minute changes documentation |
| Arsalan Sadiq | Tried to make the Iteration run successfully by handling the packets in client class UML, UCM Diagrams | Worked on error handling on client side helped on the intermediate host and | Worked on error packets like delayed, lost in the intermediate host | Worked on error packets based on unknown TID in the intermediate host and implemented from client side | Fixed Bugs, made sure all the iteration 1 – 4 works smoothly Refactored code |
| Jeff Tudor | Implemented the shutdown of client and server | Worked on error packets and error handling on client side and intermediate host | Make sure the Client and server handles the error properly | Made sure client, intermediate host and server run smoothly with the test cases run as accordingly as it should receive | Test cases run smoothly, made sure all the iteration 1 – 4 works smoothly created Diagrams |

# Execution/Termination Instructions

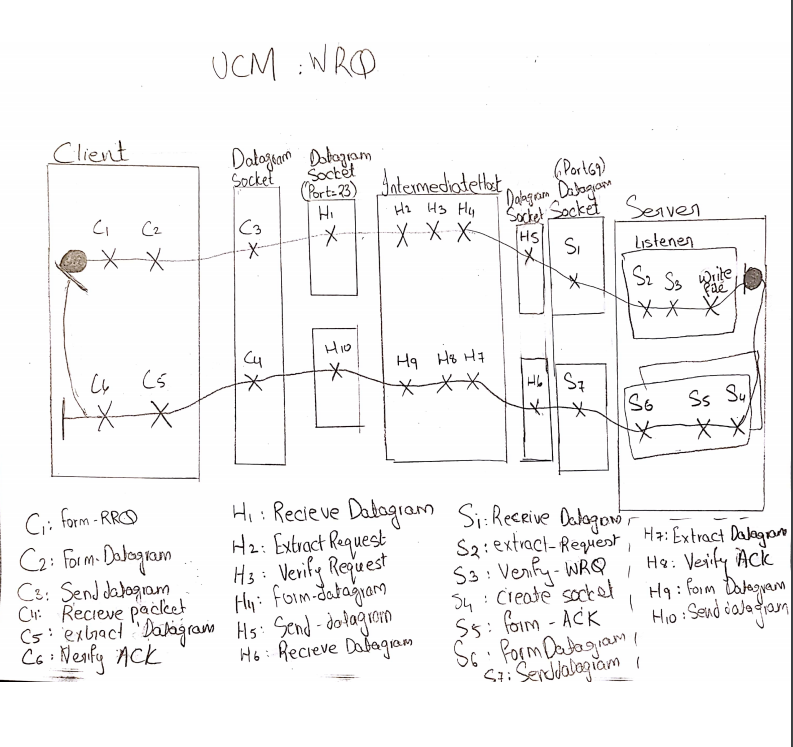
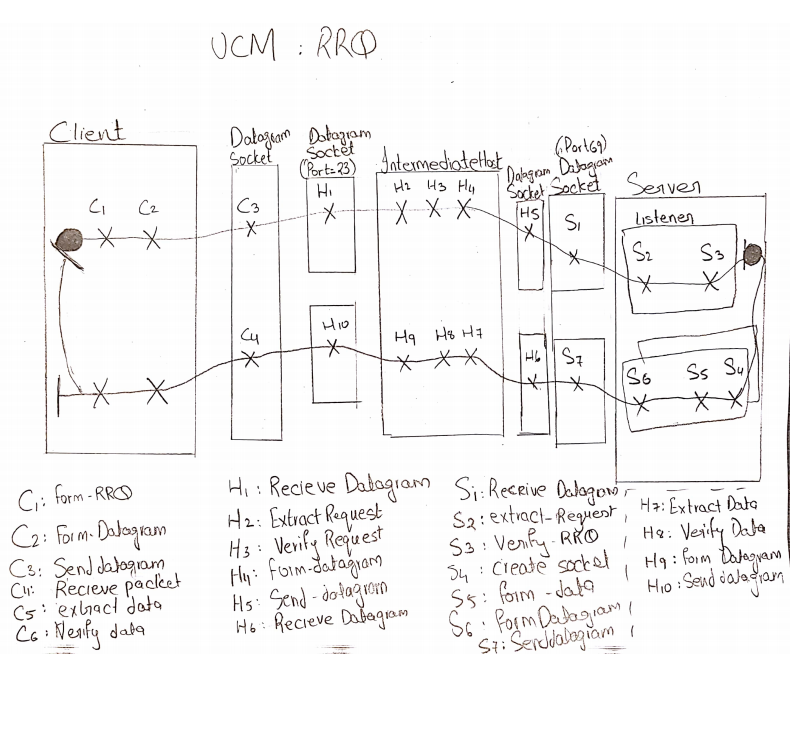
Begin by placing a file to use for transfers in the root of the program directory. In our testing we used the file "sample.txt". Place any file in the same folder as sample.txt you wish to use for transfers.

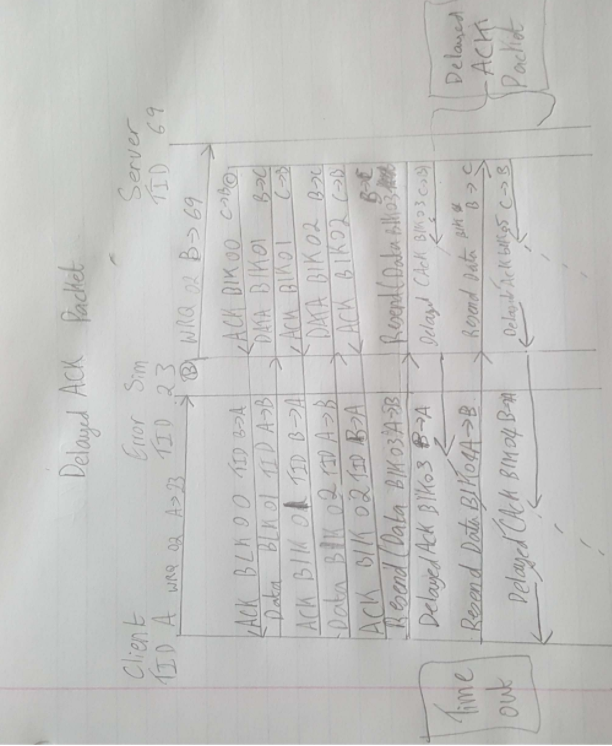
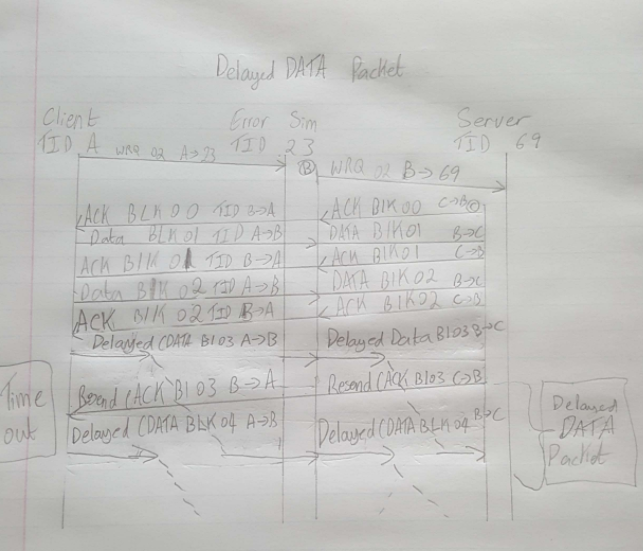
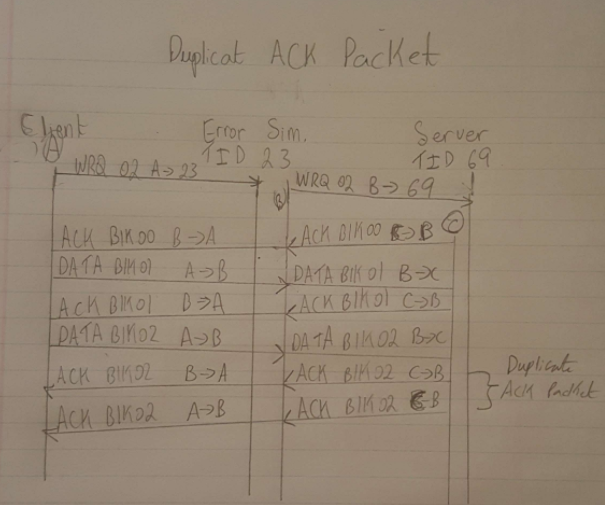
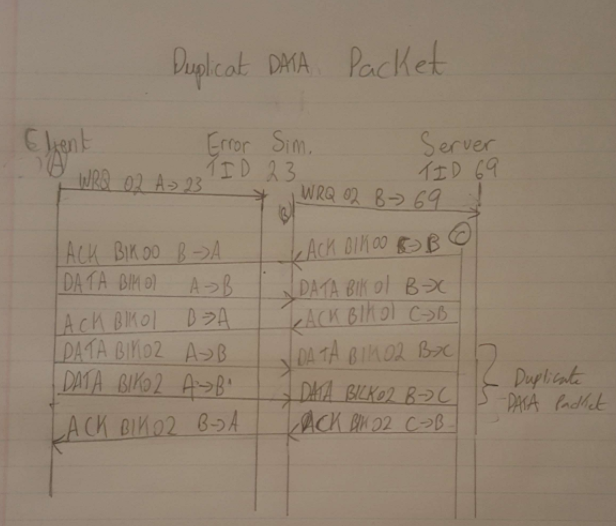
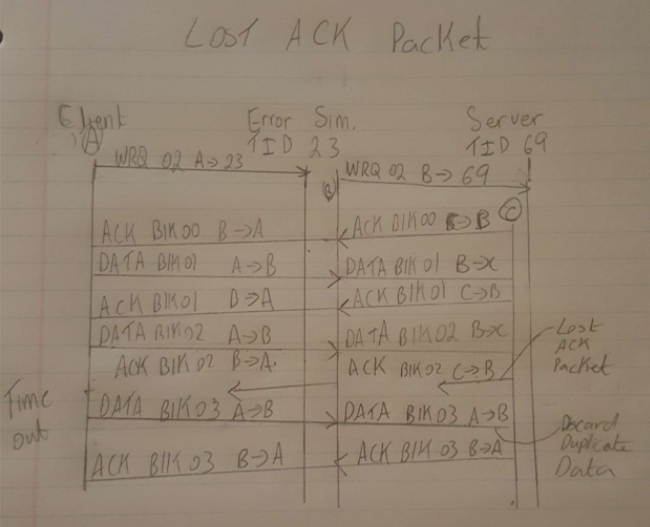
1. run the ThreadedServer.java class as java applications
2. run the IntermediateHost.java class as java applications
3. run the client.java class as java applications
4. Follow the prompts in the console window for the IntermediateHost class
5. Once it says “waiting for packet from client” move over to the console screen for the Client class
6. Follow and answer those prompts and the file Transfer will begin

First within the console for the intermediate host the error simulator options must be chosen, here the user can choose which type of simulation to run (or to not run a simulation) and the specifications

Second within the console for client the user chooses which type of request to send, and the name of the necessary files.

**Diagrams**

UCMs for read/write including the error simulator (from iteration #1)  


Timing diagrams showing all the error scenarios  
  


UML class diagram  
  
