SYSC 3303 – TFTP Project - 1000000

ITERATION 5

Colin Kealty: 100855810
Mohammed Ahmed-Muhsin: 100855437
Kais Hassanali: 100861319
Samson Truong: 100848346

Last Updated: 6/16/2014 12:47:54 AM

COMPILATION AND UTILIZATION

1) File>Import
2) select General > Existing Projects into Workspace
    a. "Select root directory" - browse and navigate to the folder of iteration
    b. If not already checked, check the box in the "Projects:" box next to the project which you imported
3) Ensure that there is a Server folder in the working directory. In this directory, you can put any files that you want to transfer FROM the server to the client
4) Compile the 9 files to ensure that all intended features work properly. The three steps outlined below must be followed in sequence to ensure full utility:
    1. Run ServerUI.java
        1.1. Start server
        1.2. Shutdown can be initialize at any time by entering "1"
            1.2.1. No new request will be processed, but current processes will finish
    2. Run ErrorSim.java
        2.1. Select the ErrorSim operation mode
        2.2. Specify IP if running on distributed environment
    3. Run ClientUI.java
        3.1. Select the client operation mode
        3.2. Input an IP address the server/ErrorSim or use local IP
        3.3. Select if you are reading or writing
            3.3.1. If you are Reading, make sure the file exists in the Server directory
        3.4. Type in the file name [nameoffile.txt]
        3.5. Type in the proper directory
    4. After a successful transfer, you need to select the operation mode for the ErrorSim again by switching consoles to the ErrorSim and then switching back to the client console to select the file to transfer

Notes:

- *All output for a Read will be in the directory you specified in the name: [nameOfFile.txt}*
- *All output for a Write will be in the Server directory in the name: [nameOfFile.txt]*

KNOWN ISSUES

(none)

TESTED AND WORKING FOR

Windows 7 at Carleton University Lab AA 508 running JAVA 1.7.0_25

Windows 8.1 i7 960 @ 3.20GHz 8 GB RAM running JAVA 1.7.0_25

Windows 8.1 i5 3570k @ 3.80GHz 16 GB RAM running JAVA 1.7.0_25

IDE AND JAVA DEVELOPED ON

Eclipse IDE Release 4.3.2 running JAVA 1.7.0_25

TESTING INSTRUCTIONS

To test errors, use the provided file "512+.txt" which is greater than 512 bytes. This is to ensure that there are enough blocks to corrupt one (if there are not enough, then the file transfer will just complete normally). Ensure that you select a block to corrupt that isn't after the transaction or there will be no result. Use ErrorSim's UI in order to test any scenario.

TESTING RESULTS

View attached document "TestReport.pdf" to see all the testing done. Upon using any of the provided files (512.txt which is a file that is exactly 512 bytes OR Over 512.txt which is a file that is over the 512 bytes), you can use fc [firstfilename] [secondfilename] in command line to check the output file and the source file.

The ErrorSim can sit either on the client side or in between client and server on a separate machine. The important part is that there must be an ErrorSim in between EACH client and the server so multiple transfers can be done at the same time.

For example, if we have Client A and Client B on Computer 1 and 2, Server on Computer 3, then ErrorSim A and ErrorSim B must be on Computer 4 and 5 respectively and sit between Client A and B respectively.

DESIGN CHOICES

- The Client and Server will timeout 5 times before truly timing out
- Our client allows for files to be overwritten with no error message being thrown. This was up to us for the design of our system and we have implemented an over writing rule if the file exists already on the client
- Server will not allow files to be overwritten, an error will occur.
- In a duplicated request, server will react as if two separate request has been submitted. One request will be completed and the other will timeout.
- An invalid TID situation, the server/client will only send an error message without disclosing the expected TID value

CHANGELOG

CLIENT:

Includes ClientUI.java and Client.java

7.0

- Added functionality in ClientUI.java to accept IP address as user input
- Changed Client.java to use IP address
- Cleaned up the entirety of the client, now has better, cleaner prints and cleaner code
- File streams will close on IOExceptions

6.0:

- Added normal mode and errorsim mode support
- Added error packet 4 and 5 handling time

5.0:

- Improved packet recognition to shut down at the appropriate time
- New behavior which can handle error packets with error packets, 1 2 3 and 6

4.0:

- Implemented a client UI to choose the mode and the which file to transfer
- Implemented the handling of errors

3.0:

- Added file I/O support to actually receive and send a file

2.0:

- Improved documentation and in-line comments
- Improved system messages on what is being done by the program
- Implemented a single read, write, and invalid request to be sent
- Improved information printing regarding the bytes and the string
- Create separate methods for the read, write, and invalid request

1.0:

- All DatagramSockets and Packets implemented
- Basic message sent to intermediate to ensure all connections working
- Basic information about the packet displayed at each instance

ERRORSIM

Includes ErrorSim.java and ConnectionManagerESim.java:

7.0

- Ability to select the level of output being displayed
- Improved output for cleaner and more understandable output
- Multiple bug fixes in Lost Operation
- All operations respond correctly to errors being sent

6.0:

- Added functionality to simulate errors 4 and 5
- Tweaked UI to allow for more options and more specific error simulation

5.0:

- User is able to select exactly which packet to lose, delay or duplicate and how much by
- Fixed issues in the duplication of a DATA or ACK packet for both a READ and a WRITE request
- Improved user input interface to indicate which error they would like to simulate

4.0:

- Mode selection has been added
    - **Lost**: A predetermined percentage of packets will be lost (ie. simply not transferred) to the client or server
    - **Delayed**: A predetermined percentage of packets will be delayed by a random time (XXXXs - XXXXXs) to the client of server
    - **Duplicated**: A predetermined percentage of packets will be duplicated (with a random delay between them) to the server or client.
- User can properly shut down the ERRORSIM
- Bug fix where the ERRORSIM will only send to the server's Well-Known port (69). ERRORSIM will now send the secondary packets to the correct port.

3.0:

- Added multi-threading capabilities to program
- Will remain to be sending and receiving files as needed with no errors simulated

2.0:

- Improved documentation and in-line comments
- Improved system messages on what is being done by the system
- Improved information printing regarding the bytes and the string
- Reworked the sendSocket to remain open throughout the session

1.0:

- Implemented all DatagramSockets required (receive, sendReceive, and send)
- Implemented all DatagramPackets required (client and server)
- Exception handling with try/catch blocks
- Basic information about the packet displayed at each instance

SERVER

Includes ConnectionManager.java, Server.java, FileAlreadyExist.java and ServerUI.java:

7.0

- Server now accepts file transfers from Clients on other computers
- Cleaned up error outputs to be more specific and helpful
- Large file transfer now properly works
- Various bug fixes
- More detailed commenting
- File streams will close on IOExceptions

6.0:

- Added error packet 4 and 5 handling time

5.0:

- New behavior which can handle error packets with error packets, 1 2 3 and 6

4.0:

- Implemented error handling

3.0:

- Separated into a listening module which spawns out the threads to deal with request
- Multi-threaded requests to handle the reading and writing of the file

2.0:

- Improved documentation and in-line commenting
- Improved the printing of information about the packet
- Added a parser which will recognize a valid request (read or write) and an invalid one
- Keep the server alive forever until killed

1.0:

- Implemented a procedure that can receive a general request and send a response
- Print all necessary information about the packet
- Create a sending DatagramSocket and close it after a successful send

DEBUGGING

By default, the ErrorSim runs in verbose mode. There is an option to choose which mode to run ErrorSim in (Silent/Verbose/Debug).

Silent: Only errors will be displayed
Verbose: Only error messages and mode changes will be displayed
Debug: This will include packet information as well as error messages and mode changes

SUPPORT

For technical support or to report a bug, please contact:

- KaisHassanali@cmail.carleton.ca
- SamsonTruong@cmail.carleton.ca
- MohammedAhmedMuhsin@cmail.carleton.ca
- ColinKealty@cmail.carleton.ca