P2 Design-Test Document

Partner A: Humberto Torralba

Partner B: Manuel Torralba

1. Server Design

Overview

Describe the overall server design. You must include the number of threads you create including the main thread, what each thread is for and doing, any loops you have for each thread (and do this for both event-loop and thread-based server).

Justification

Justify why your design is efficient and show evidence it can handle load efficiently. Specify the number of clients, the file size for each client, and the response rate/delay each client.

Data structures

List any notable data structures you used and justify the use of the data structure. Also specify how you handle synchronization for the data structure if there were any need for synchronization.

How timeouts are handled

Describe how the timeouts are handled.

How shutdown is handled

Describe how you handled the shutdown gracefully. (That is when you hit 'q' or ctrl+d in the command prompt.)

Any libraries used

List any libraries that you used for the implementation and justify the usage.

Corner cases identified

Describe corner cases that you identified and how you handled them.

2. Server Testing

Testing your server

Describe how you tested your server. Include the description of each test case and the setting (such as local only, one local one remote, how many clients, use of mock client, etc.).

3. Client Design

Overview

Describe the overall client design. Specify any differences between event-loop based client design and thread-based client design. You must include the number of threads you create including the main thread, what each thread is for and doing, any loops you have for each thread (and do this for both event-loop and thread-based client).

Response

In the thread-based client, we have up to 3 threads running at a given moment. One (main) thread will create the socket and handle input from the user. It will create thread #2 to handle reading from the server. Thread #2 will create a third thread to handle the timer, as specified in Piazza note @165. Thread #1 will have a loop to continuously handle receiving input from the command line. Thread #2 will have a loop to continuously handle ALIVE (and other erroneous) messages from the server. Thread #3 will not have any loops, but rather just set a global variable showing that a timeout has occurred. It will also be in charge of sending the goodbye message to the server and closing the socket.

Justification

Justify why your design above is efficient. Also show evidence how your client can handle sending packets from a large file (each line is close to UDP packet max and also contains many lines) and receiving packets from the server at the same time. Note you do not want to cause the false TIMEOUT from the server because you are too busy just sending out the packets to the server when the server actually has sent you a packet before the TIMEOUT.

Response

Our design can handle sending multiple packets from a large file because a separate thread is in charge of reading messages from the server. One thread will not slow the other down. The way the threads are setup, there is no blocking occurring between them, that is, never at one point in time is a thread actively waiting on another to finish/update data.

Data structures

List any notable data structures you used and justify the use of the data structure. Also specify how you handle synchronization for the data structure if there were any need for synchronization.

Response

We have no data structures in use. We only use 1 static variable that is shared between all threads that is set to show whether or not a timeout has occurred. No explicit synchronization techniques were used as threads naturally share everything between them except for their stacks.

How timeouts are handled

Describe how the timeouts are handled.

How shutdown is handled

Describe how you handled the shutdown gracefully. (That is when you hit 'q' or ctrl+d in the command prompt.)

Any libraries used

List any libraries that you used and justify the usage.

Corner cases identified

Describe corner cases that you identified and how you handled them.

4. Client Testing

Testing your client

Describe how you tested your client. Include the description of each test case and the setting (such as local only, one local one remote, how many clients, use of mock server, etc.).

5. Reflection

Reflection on the process

What was most challenging in implementing the server? What was most challenging testing the server?

What was most challenging in implementing the client? What was most challenging testing the client?

What was most fun working on the project? What was most not-so-fun?

If you are to do this all over again how would you do it differently?

Reflection on pair programming

Log of the amount of time spent driving and the amount of time spent working individually for each part (e.g., X drives 1 hour; Y drives 45 minutes; X works alone for 1 hour, etc.)

What went well/or not-so-well doing pair programming? What was your take away in this process?

Submission

Remember to export to pdf and push it to your github team repo under the project root (the same level as README.txt)