CS3102 - P2 Report

190022658

April 18, 2022

1 Overview

The objective of the practical was to write a C program to design, implement, and test a simple, reliable transport protocol on top of UDP.

2 How to use

There is a Makefile included. So a make/make all command should output server and client executables. "client" should be run with a hostname of the server as its argument. "server" runs before the client and without any argument. The makefile also includes a clean command which removes the object files and the executables.

3 Design

3.1 ReliableConnection

This is the api which is both for the server and the client.

There is a struct Probe defined in the header file which contains a sequence number, 3 flags and padding, timestamp and a string (for message). Using a void pointer might have been a better idea instead of a string. I could also have put all the flags inside one 8bit integer but since it would still take the same space there was no need to. The struct's size is 24 bytes.

For ease, I created a hton_probe and ntoh_probe functions which change the struct from host order to network order and vice-versa. I use the byteorder64.h from studres here. I also created a print_probe function and the empty_probe to print and empty a probe respectively. The empty_probe function also sets the timestamp value to reduce repetition.

There is a sendData function which accepts local and remote sockets, an int (0 or 1) and a message. This is also used to establish and terminate connections. The int is supposed to work as a flag. This function sends a struct and waits for a reply. This uses the idle-rq protocol. The timeout is set for 200ms and can be changed via the header file. The protocol is incomplete as checksums have not been used which causes some features to be left out.

The serverListen function accepts the local socket and waits for a connection using a loop. Inside there are if statements for each states and respond respectively.

3.2 ReliableServer

Sets up a socket and uses the api function serverListen to wait for input and respond. Uses getuid instead of hardcoding the value.

3.3 ReliableClient

Sets up the sockets and uses them to establish connection, send data and terminate the connection.

4 Conclusion

This practical could have been better. I could have analysed the connection, corrected the FSM and done some of the Part 2 extensions but this is the submissions due to circumstances. I did learn from the last practical and not repeat some of the same mistakes and learned about reliable connections.