CPSC 441 - Assignment 2 - Instruction Manual

Ahad Hamirani - 30063218

How to compile:

For indirection server	For Microservers	Port Numbers for Microservers
g++ indirection.cpp	g++ Translator.cpp	Translator: 1786
./a.out <Microservers IP> <Port Number>	./a.out <Port Number>	Convertor: 4592
		Voting: 2968

First compile the microservers using g++. I have provided an example of how compilation would look like for the Translator server but this step will be identical for all of the microservers. Next when running the code make sure to include the port number as shown above. In this instance port numbers for all the microservers have been hardcoded in the indirection servers. Above I have provided the port numbers for each microserver so use the correct one when running the servers. Next, compile the indirection server using g++. Next when running the code make sure to include the IP in which the micro servers are running and a port number. This port number should match the port number you plan to use when connecting as a client through telnet.

How to use my solution:

Firstly make sure that you are on different IP's for the microservers (make sure that all of the microservers are running on the same IP address), indirection server and client. Then compile the microservers and run them using the port numbers provided above. If done correctly all of the three microservers will display "waiting for connections…" in the terminal. This means that the microservers are successfully set up and ready. After this, compile and run the indirection server. If done correctly a new line will appear in the terminal and a connection accepted message will appear once the client has connected.. Finally, using telnet connect as a client to the indirection server using the port number used when running the indirection server. If done correctly telnet will prompt you to send a message to the indirection server. Here you can send any message and the indirection server will respond with a numbered list of available interactions. Now you can choose which interaction you want to execute by typing the number corresponding to the interaction followed with the enter key. If you choose the "Translate" interaction the server will prompt you to enter an english word. If the entered word is in the list of translated words (can be found in the Translator.cpp file) the translation will be returned, if not an error will be returned. If you choose the "Currency Exchange" interaction the server will prompt you to enter 3 inputs <Source currency> <Destination currency> <value>. If the inputs are presented correctly the proper exchange value will be returned if not then an appropriate error describing the mistake will return. If you choose the "Vote" interaction and have already voted from your current client IP address it will return stating that you cannot vote twice. If you have not yet voted it will provide you with an encryption key and prompt to enter a candidate's id. If the id is correct, votes of that candidate will be updated if not then an error will be returned. If you choose "View voting report" and have either not voted yet (from current ip) or the voting deadline has not passed it wont let you view the results, but if both are satisfied it will return a table with the voting results. If you choose "View all candidates" a list of all candidates and their ids will be returned. If you choose "Close connection" the client and the indirection server will be closed.

Features implemented:

Indirection server features: Can take in the input from client, process it, and communicate with appropriate micro-server via UDP to perform the client's request prior to returning the final result data back to the client via TCP. Can provide client with timeout error if the UDP microserver connection has timed out (more than 3 seconds to get recvfrom after sendto). Can compute and store clients IP and forward it to the voting micro-server. Can close the client connection.

Translator server features: Creates a UDP server. Can take in input, translate one word from English to French and send it back. Can also appropriately handle invalid inputs. Supports 5 different English words.

Currency Converter server features: Creates a UDP server. Can take in input, process it and send back converted value's from CAD to US Dollar, Euro, British Pound, Bitcoin and Etherium for valid inputs and appropriate errors for invalid inputs.

Voting server features: Creates a UDP server. Can take input, process it and perform 3 different tasks accordingly. Firstly, it can return a list of candidate names and their assigned ID's. Next, it can perform secure voting by creating and sending a public encryption key, receiving the encrypted ID, decoding it, checking if the id exists in the candidate's ids, if it does then updating the voting results if not then returning an appropriate error. Finally, it can also return voting results. **Note**: The server also checks the client's ip and only lets the client vote if it hasn't already voted from that ip address. Moreover, the server only returns the voting results to clients that have already voted.

Bonus: I have also implemented the bonus feature, so clients can only vote before a certain time and only see the results after a certain time. The time zone here is universal time. This time can be changed in the voting.cpp file. **Note**: When testing the code please change the hours and mins in the voting.cpp file to the current universal time to see the correct outputs.

Testing:

I tested my solutions using my laptop(running macOS), 136.159.5.27 linux server and 136.159.5.25 linux server. I tested my micro-servers by initially creating a tester UDP client and testing all the logical test cases. I also tested the indirection server with the TCP client to make sure all implemented features in micro-servers are properly carried through the indirection server. Finally, I also test the UDP timeout by running only the indirection server.