Department of Computer Science
Computer Networks
Due: Thursday 24th October 2024

Marking: 10% and up to 5 Bonus Marks

Your name:
TA Name:

This is the third of three programming projects for Computer Networks, and is worth a total of 10% of your final mark. Additionally up to 5 bonus marks may be awarded for this project.

You may work on this individually or as part of a team of 2.

You may reuse your code from Project 1, or use the very simple, sample client-server project provided as a base. Note that the sample project has little to do with the protocol you need to implement here and contains some errors.

Important: Please read the instructions carefully.

In particular:

- The programming language is C or C++
- Although you can use boost for string handling, you may not use Boost.Asio (The BSD Socket API lies under all networking frameworks including Boost.Asio. We want to expose you to networking fundamentals, so that you understand where some of the issues with using networking frameworks come from.)
- **You must join a Group in Canvas, even if you are the only person in the group. The group ID of your canvas group, will also be the ID of your server.**
- You must provide clear instructions in a README file on how to compile and run your program, and what OS it was compiled on. If your program does not compile, please provide a detailed explanation of why you think that is, and why you decided not to use a source code management system with frequent commits for your project.
- Code should be well commented and structured.
- Although you must develop your own code for this assignment, it is at the same time a class project, and relies on co-operation to succeed. It is perfectly ok to co-ordinate servers, discuss issues with inter-server messaging, agree on additions to the protocol, etc.

Programming Assignment

Newsflash 2030.

As a result of the 2022 Hunga Tunga eruption increasing stratospheric water content by 10%, and the resulting rapid increase in global average temperatures during the rest of the decade, Iceland is enjoying a much improved climate and increasing economic strength. However, as the fighting on the eastern front moves into its 8th year, and Russian troops encircle Budapest, the struggle on the Internet intensifies. A few days ago major cyber hostilities broke out between the four superpowers. Three hours ago all social media went off the Internet, the mobile telephone network failed, and email and all other forms of communication have become extremely unreliable. In Iceland the medical system has resorted to communication through bicycle couriers, and all financial transactions are currently suspended.

As a founding member of the League of Little Nations, Iceland has been asked by the Dutchy of Grand Fenwick to lead the rapid development of alternative and robust peer-to-peer forms of communication for the citizens of the Northern Democratic Alliance. With the priority being to get any form of simple messaging up and running, they have provided a poorly written basic specification for the protocol it should use.

To this end your task is to write a simple store and forward botnet message server, with accompanying Command and Control (C&C) client, following the specification below. The overall goal is to link your server and client into a class wide botnet, which provides alternative routes for messages, and is not subject to the single points of failure that the old social media empires were vulnerable to. In order to do this, you will need to give some thought to routing commands through the network, connecting to and leaving the botnet, storing messages for disconnected servers, message expiry, and handling commands for other groups. Remember also that there is other information besides that purely in the messages - for example, knowing which link to your server the message arrived on, and which server is at the other end of that link.

Beware, the superpowers seem to have gotten wind of the plan of the League of Little Nations. They are trying to infiltrate our network, steal secrets and disrupt communication. Be on the lookout.

There are also mysterious Number servers, sending messages seemingly pointlessly and randomly into the network. Can you find them? Can you read their messages?

Rules of Engagement:

0. Don't crash the (bot) network.
1. Try not to crash the campus network either. At least, not outside working hours.
2. Messages should not be longer than 5000 characters. You may truncate any messages you receive longer than that.
3. "Be strict in what you send, be tolerant in what you receive!"

4. The executable for your server should be named as tsamgroup<Group ID> eg. tsampgroup1, the first parameter should be the port it is accepting connections on.
5. Your server should identify itself within the Botnet using your group ID, eg. A5_1 (Note, Instructor servers will also be running, prefixed by I_, and some other names exist as well)
6. While part of the botnet, your server must maintain connections with at least 3 other servers and no more than 8 servers.
7. TCP ports 4000-4200 are available on the TSAM server (130.208.246.249) for external connections.
8. Your clients must connect to your server only, and issue any commands to other servers on the botnet via your server.
9. You must use two token characters to indicate the start and end of each message between servers: ASCII character 0x01 (SOH) for start of message, and 0x04 (EOT) for the end of message. Don't forget to use bytestuffing when necessary.
10. You are encouraged to reach out to other groups to be on the botnet with them at the same time, and co-operate in getting things working.

You may host the server on your machine, or on the TSAM server, and run the clients from your local laptop. If you are not able to do this for any reason, please bring this to our attention during *the first week of the assignment.*

Note: The nohup and disown commands can be used from a bash/zsh shell to run a command independently from the terminal, such as a server. If you do this on skel, please clean up any old processes.

Server Specification

Your server should listen on one TCP port for other servers (or your client) to connect to it.

You may run the server on the TSAM server or on some other computer, just make sure it is actually reachable for others to connect to your server.

The server port you are listening on should be included in the command line, so that servers running on the TSAM server can be found by other groups. For example:

./tsamgroup1 4044

where 4044 is the TCP port for server connections.

Server Communication

Commands between servers should be sent using the following format:

$< SOH >$<command>,< comma separated parameters >$< EOT >$

Server Commands

The server should support at least the following commands in communicating with other servers. Other commands are allowed if they facilitate network communication. The server should keep a timestamped log of all commands sent and received. This log may contain other information about the server state to facilitate debugging.

HELO,<FROM_GROUP_ID>	Reply with the SERVERS response (below) This should be the first message sent by any server, after it connects to another server.
Respond with: SERVERS	Provide a list of *directly connected - i.e. 1-hop*, servers to this server. Note, this is a response to the HELO command. The first IP address in the list must be the IP of the server sending the command. Servers should be specified with their name (e.g., A5_ID), the HOST IP, and PORT they will accept connections on, comma separated within the message, each server separated by ;

eg. SERVERS,A5_1,130.208.243.61,8888;A5_2,10.2.132.12,10042;

KEEPALIVE,<No. of Messages>	Periodic message to 1-hop connected servers, indicating the no. of messages the server sending the KEEPALIVE message has waiting for the server at the other end. Do not send more than once/minute.
GETMSGS,<GROUP_ID>	Get messages for the specified group/named server. This may be for your own group, or another group.
SENDMSG,<TO_GROUP_ID>,<FROM_GROUP_ID>,<Message content>	Send message to another group. The message content may be arbitrary data, but the whole command should not exceed 5000 bytes.
STATUSREQ	Reply with STATUSRESP as below
STATUSRESP,<server, msgs held>,...	Reply with comma separated list of servers and no. of messages you have for them

eg. STATUSRESP,A5_4,20,A5_71,2

Client Commands

Communication between your client and server should use the protocol below. You may implement additional commands if you wish. The client should print out all commands sent and responses received with a timestamp (date and time, please no nanoseconds).

GETMSG,GROUP_ID	Get a single message from the server for the GROUP_ID
SENDMSG,GROUP_ID,<message contents>	Send a message to the server for the GROUP_ID
LISTSERVERS	List servers your server is connected to

Assignment

1 ..... *10 points*

Submit your code together with a README (text file) and a Makefile as a single zip file. Any additional files and information (wireshark traces, parts of log files, etc.) must be clearly named and listed in the README file stating which of the following points or bonus points you expect to get.

(a) (4 points) Implement client and server as described above. All local commands to the server must be implemented by a separate client over the network.

(b) (1 point) Provide a wireshark trace of communication between *your* client and server for all commands implemented in the client to server protocol

(c) (1 point) Have been successfully connected to by an Instructor's server. (Provide timestamped log)

(d) (1 point) Successfully receive messages from at least 2 other groups (Provide timestamped log)

(e) (1 point) Successfully send messages to at least 2 other groups (Provide timestamped log)

(f) (1 point) Code is submitted as a single zip file, with README and Makefile. (Do not include hg/git/etc. repositories or other hidden files!) The README describes how to compile and run the code, lists the commands that are implemented in the client and server and describes the behaviour of the server in response to these commands and potentially other external events.

(g) (1 point) Code is well structured and documented. The server produces a readable log file showing all received and sent commands and other useful information about the internal state and behaviour of the server.

Note, that there is difference between *message* and *command*. Messages are send between groups using the SENDMSG command and only count as received when the receipient fetched them from their server using their client.

Bonus Points

Note: The maximum grade for the assignment is 10 points. No more than 5 bonus points will be awarded additionally. The bonus points carry over to other assignments and the final grade (but do not count towards the passing threshold for the final exam).

List in your README file the bonus points you want to claim together with relevant information that shows you are eligable.

2 ..... *10 points*

(a) (1 point) Submit your solution for the client/server protocol implementation within the first week. For this, you need to implement the client and at least a stub for the server that communicates with the client. The server does not need to implement all the commands for the peer-to-peer communication, yet.

Submit the code for both client and server and the wireshark trace for 1(b).

Note: You are allowed to improve the code later for the final submission and may get some feedback earlier.

(b) (3 points) Obtain bonus points for connectivity:

- 0.5 points for messages from servers from students in Akureyri (The Akureyi group whose messages are reported will receive a matching 0.5 points, if they are reported at least 2 times by non-Akureyri groups.)
- 1 point per 5 different groups you can show messages received from

Provide relevant parts of your timestamped log to get these points.

(c) (2 points) Identify the enemy. Provide IP and port of the process, timestamp, some relevant portion of your server log and a reason why you think this was an intruder in the network. (1 point per correctly identified instance, several instances need to be at least 2 hours apart)

(d) (2 points) **Either**: Server is *not* running on the TSAM server or any campus machine or the VPN.

- To do this from home you will probably need to perfom port forwarding on your local NAT. You can test on the TSAM server, but all project submissions must use a non-RU IP address.
- If you are connecting through eduroam from the student dormitory you won't be able to use the dormitory network for these points. Please contact the instructor if this is an issue.

(e) (2 points) **Or**: Write a brief - no more than 1 page - description listing the several security issues of the botnet.