Department of Computer Science
Computer Networks
Due: Thursday 23rd October 2025

Marking: 10% of the final grade, 100 points and up to 50 bonus points

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 3.

You may reuse your code from Assignment 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++
- You may use additional libraries for string handling and logging, but not for networking. If in doubt ask! All dependencies other than standard C/C++ libraries must be included in your submission.
- **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.**
- 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.

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. 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)
5. TCP ports 4000-4200 are available on the TSAM server (130.208.246.98) for external connections.
6. Your clients must connect to your server only, and issue any commands to other servers on the botnet via your server.
7. 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.

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. This allows to keep the server running, even if the ssh connection gets closed. If you do this on the TSAM server, please clean up any old processes.

Server Specification

- Your server must listen on one TCP port for other servers (and 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 executable of your server should be called "tsamgroupX" with X being your group number. The server port you are listening on must be the first command line argument. For example:

  ./tsamgroup1 4044

  where 4044 is the TCP port for server connections.
- While part of the botnet, your server must try to maintain connections with at least 3 other servers and no more than 8 servers. Connections to other groups' servers should be prioritized over instructor servers.
- The server must keep a timestamped log of all commands sent and received. This log may contain other information about the server state to facilitate debugging.
- The server must operate autonomously! The server must not accept nor wait for any input on the terminal. All communication with the server must happen through TCP sockets. The client is only needed for sending and receiving message to/from other groups and potentially inspecting and configuring the operations of your server at run-time.
- Do not hard-code any IP addresses or port numbers. Use additional command line arguments or commands from your client for this.

Server Communication

All commands between servers must be sent using the following format:

`<SOH><length><STX><command><ETX>`

with

- the ASCII characters 0x01 (`<SOH>`), 0x02 (`<STX>`), and 0x03 (`<ETX>`) denoting the start of the header, start of text, and end of text, respectively, and
- `length` being a 16-bit unsigned integer (in network-byte order) denoting the length of the transmission in bytes (including the five bytes for framing the message: `<SOH><length><STX>` and `<ETX>`).

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.

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. 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 server names 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 must print out (and/or log to a file) all commands sent

and responses received with a human-readable timestamp (date and time, please no nanoseconds).

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

Assignment

1. ..... *100 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) (40 points) Implement client and server as described above. All local commands to the server must be implemented by a separate client over the network. The server must neither accept nor wait for user input.

(b) (10 points) Provide a wireshark trace of communication between *your* client and server for all commands implemented in the client-to-server protocol

(c) (10 points) Have been successfully connected to by someone else's server. That means, the other server needs to initiate the connection (call `connect`) and your server accept it (call `accept`). (Provide timestamped log of accepting an incoming connection, showing source IP and port)

(d) (10 points) Successfully receive messages (not commands!) from at least 2 other groups. (Provide timestamped log)

(e) (10 points) Successfully send messages (not commands!) to at least 2 other groups. (Provide timestamped log)

(f) (10 points) 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, which OS you developed on, 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. You should especially describe any additional commands or behaviour of your server that is not list here in the assignment description.

(g) (10 points) 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 sent between groups using the SENDMSG command and only count as received when the receipient fetched them from their server using their client.

Bonus Points

The maximum grade for the assignment is 100 points. No more than 50 bonus points can 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 eligible.

2 ..... *80 points*

(a) (10 points) **Early Bot:** 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) (10 points) **Ready Bot:** Obtain 10 bonus points for the server being connected to the botnet (and responding) on at least 5 days for at least 2 hours each. Provide relevant parts of your timestamped log to get these points, e.g., showing commands received from other servers on different days and different times.

(c) (20 points) **Chatty Bot:** Obtain 10 points per 10 different groups you can show messages (**not** commands!) received from, up to 20 points. Provide relevant parts of your timestamped log or screenshots to get these points, e.g., showing server log delivering messages to your client or showing received messages in your client.

(d) (10 points) **Indirect Bot:** Obtain 10 bonus points for the server delivering at least 10 messages to the correct recipient for which your group is neither sender nor receiver. Provide relevant parts of your timestamped log to get these points, e.g., showing the relevant SENDMSG commands received and sent by your server.

(e) (10 points) **Edge Bot:** Obtain 10 points for your server being connected to at least 3 peers, **but none of the instructor servers**, at 5 different times (at least 1h apart). Provide relevant parts of your timestamped log to get these points.

(f) (20 points) **External Bot:** 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 router. You can test on the TSAM server, but all project submissions must use a non-RU IP address and your bot should show up in our logs with 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.

To get the points, provide relevant parts of your timestamped log. We will check whether your bot shows up in the instructor server's logs with a non-RU IP address at the given times.