Department of Computer Science
Computer Networks
Due (1st draft): Monday 10th October 2022
Due (final): Sunday 23th October 2022

Marking: 10% plus bonus marks

Your name:
TA Name:
Time Taken:

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

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

You may reuse your code from Project 1, or use the very simple, sample client-server project provided as a base. Note that some of the code in the sample project is purely there for illustrative purposes and may be irrelevant for this project.

The submission on the final deadline will be used for grading, but to encourage you to start early, you can get bonus points (and possibly useful feedback on your implementation) for partial functionality implemented and submitted by the 1st draft deadline.

There will be an additional (voluntary) lab session with TAs on Saturday October 8th, room M201, from 11h-16h

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. Include one line, either "OS: Linux" or "OS: MacOS", to indicate 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, etc. Just don't share your code.

Programming Assignment

Things have not been going well this decade, and in the last few days major cyber hostilities have broken out between the superpowers. Three hours ago all social media went off the Internet, and email and other forms of communication have become extremely unreliable. As a founding member of the League of Little Nations, Iceland has been asked by the Dutchy of Grand Fenwick to lead the development of alternative and robust peer-to-peer forms of communication for the citizens of the remaining democractic countries.

To this end you will 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 messages through the network, connecting to and leaving the botnet, storing messages for disconnected servers, message expiry, and handling messages 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.

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.
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 tsam<Group ID> eg. tsampgroup1, the first parameter should be the port it is accepting connections on from other servers.
5. Your server should identify itself within the Botnet using your group ID, eg. P3_Group_1 (Note, Instructor servers will also be running, prefixed by I_, and number stations will send messages as N_)
6. While part of the botnet, your server must maintain connections with at least 3 other servers and no more than 16 servers.
7. TCP ports 4000-4100 are available on skel.ru.is for external connections. However, running your server on your own machine is preferred. You may need to setup port-forwarding on your NAT router to make this possible.

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 skel.ru.is, 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 *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 be careful to clean up any stray processes during/after testing and debugging.

Server Specification

You do not have to run your server on skel.ru.is, as long as it can connect to at least one server that is running on skel.ru.is. If you are having problems connecting to skel, contact an instructor in the first week.

Your server should listen on one TCP port for other servers to connect to it. It may listen to a separate port for your clients to connect.

The server port you are listening on should be included in the command line, so that servers running on skel.ru.is can be found. For example:

./tsampgroup1 4044

where 4044 is the TCP port for server connections. You should allow a maximum of 16 connections to other servers.

Server Communication

Messages 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 with other servers. Other messages are allowed if they facilitate network communication.

JOIN,<GROUP_ID>	This should be the first message sent by any server initiating a connection, after it connects to another server. The remote server shall reply with the SERVERS response
SERVERS,<serverlist>	Provides a list of servers (the first entry is the server itself; additional entries: all other *directly connected – i.e. 1-hop* servers. Note that this is a response to the JOIN command. Each server entry should be specified as GROUP_ID, the HOST IP, and PORT they will accept connections on, comma separated within the message, entries separated by ;

eg. SERVERS,P3_GROUP_1,130.208.243.61,8888;P3_GROUP_2,10.2.132.12,10042;

KEEPALIVE,<No. of Messages>	Periodic message to 1-hop connected servers, indicating the number of messages the server sending the KEEPALIVE message has waiting for the receiver. Do not send more than once per minute.
FETCH_MSGS,<GROUP_ID>	Request messages for the specified group. This may be for your own group, or another group.
SEND_MSG,<TO_GROUP_ID>,<FROM_GROUP_ID>,<Message content>	Send a message to another group
STATUSREQ,<FROM_GROUP_ID>	Request an overview of the messages held by the server. Reply with STATUSRESP as below
STATUSRESP,FROM_GROUP,TO_GROUP,<group, msgs held>,...	Send a comma separated list of groups and no. of messages you have for them

eg. STATUSRESP,P3_GROUP_2,I_1,P3_GROUP4,20,P3_GROUP71,2

Client Commands

Communication between the client and server should use the protocol below. You may implement additional commands if you wish. The client should timestamp (day, minute and second, please no nanoseconds) all messages that it sends to some destination (group ID)

FETCH,GROUP_ID	Get a single message from the server for the GROUP_ID
SEND,GROUP_ID,<message contents>	Send a message to the server for the GROUP_ID
QUERYSERVERS	List servers your server is connected to

Assignment

1. Implement client and server as described above. All local commands to the server must be implemented by a separate client over the network. The client should sanitize the user input to make sure only valid messages are send to the server. (40 points)
2. Provide a wireshark trace of communication between *your* client and server for all commands implemented (10 points)
3. Have been successfully connected to by an Instructor's server. (10 point)
4. Successfully receive messages from at least 2 other groups (Provide timestamped log) (20 point)
5. Successfully send messages to at least 2 other groups (Provide timestamped log) (10 point)
6. Code is submitted as a single tar file, with README and Makefile. (Do not include hg/git/etc. repositories.) (10 point)

Bonus Points

Note: The maximum grade (including bonus points) for the assignment is 150 points.

- Early deadline submission of client and server code that (maximum 10 points):
  - Implements the client commands (the client must be able to send all commands specified above to the server; the server may send a "not implemented" response for commands not yet implemented. (5 points)
  - Connects to at least one other server, and lists that server when your client sends the QUERYSERVERS command. (5 points)
- Obtain bonus points for connectivity (maximum 20 points):
  - 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.)
  - 0.5 point per 10 different groups you can show messages received from
- Decode a hashed message from a number station (10 bonus points)

- Server is *not* running on skel.ru.is or any campus machine. (10 bonus points)
  - To do this from home you will probably need to perform port forwarding on your local NAT.
  - To get these points, provide a wireshark trace of messages sent between your server and other when it was running on a machine other than skel.
  - If you are connecting through eduroam from the student dormitory you won't be able to collect these points. Please contact the instructor if this is an issue.

To obtain these bonus points, you have to state in your README file which of these points you want to claim and where we can find the evidence for it.