Department of Computer Science
Computer Networks
Due: Wednesday 24th October 2018

Marking: 10% and up to 4 Bonus Marks

Your name:
TA Name:
Time Taken:
Estimated Time: 20 hours

This is the third of three programming projects for Computer Networks, and is worth a total of 10% of your final mark. Additionally 4 bonus marks may be awarded for this project.

You may work on this individually or as part of a group of no more than 3.

You may reuse your code from Project 2, or use the very simple, sample client-server project that will be provided.

Optional: Please include a rough estimate of how long it took you do the assignment so that we can calibrate the work being assigned for the course. (The estimated time is provided purely as a guideline.)

Important: Please read the instructions carefully.

In particular:

- The programming language is C or C++
- **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. (1 mark)
- Code should be well commented. (2 marks)
- It is fine to use online sources for inspiration but you must cite them. You may use the sample chat server and client provided as a basis for this project, or reuse your own code from the previous assignment.

Introduction

In this programming assignment you are asked to write a simple botnet server, with accompanying Command and Control (C&C) client, following the specification below.

Programming Assignment

The goal of this assignment is to link your server and client into a class wide botnet, and use them to fulfill the assignment todo list. In order to do that, you will need to give some thought to routing commands through the network, creating a local snapshot of the botnet's network topology, connecting to and leaving the botnet, and routing commands for other groups. In particular, remember that there is other information besides that purely in the messages - for example, where the message is being relayed or not, can be inferred from knowing which link to your serer the message arrived on.

Rules of Engagement:

0. Don't crash the (bot) network.
1. Try not to crash the campus network either.
2. The executable for your server should be named as tsam<Group ID> eg. tsamvgroup1
3. Similarly, your server should identify itself within the Botnet using your group ID, eg. V_Group_1 (Note, Instructor servers will also be running, prefixed by I_)
4. While part of the botnet, your server must maintain connections with at least 2 other servers and no more than 5 servers.
5. TCP ports 4000-4100 are available on skel.ru.is 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 must use two token characters to indicate the stand and end of each message between servers: ASCII character, 01 (SOH) for start, and 04(EOT) for end of message, with bitstuffing as necessary.
8. 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 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 two ports for other servers to connect to it, and continue to listen to a separate port for your clients to connect.

The first port on the command line is for TCP connections to the server from other servers. The second port is a UDP port, purely for information requests. Both of the server ports should be specified on the command line (to make it easy for the servers running on skel.ru.is to be found.) For example:

```
./tsamvgroup1 8888 8889
```

where 8888 is the TCP port for server connections, and port 8889 is a UDP port. Connections to 8889 should receive a single message in reply, listing the servers known to this server (the contents of the LISTSERVERS command below.) This is to allow servers to find servers that aren't full, from servers which are.

The server should support at least the following commands with other servers, *in addition to the commands in Project 2*:

Each registered group in Canvas will receive their own list of 5 unique hashes. Your server should provide access to your hashes, but not to any other group's.

Assignment

1. Implement server as described above, and any necessary C&C Client support
2. Provide at least 3 routing snapshots of the network, at least 2 hours apart. The snapshot should contain at least 3 servers that are more than 1-hop from your server. You may, and probably should, provide more than 3 snapshots, but do not provide more than 1 per minute connected in the botnet.
3. Have been successfully connected to by an Instructor's server.

Bonus Points

Note: The maximum grade (including bonus points) for the assignment is 14 points.

- Reconstruct the hidden message in the MD5 hashes distributed in the network (1 points)

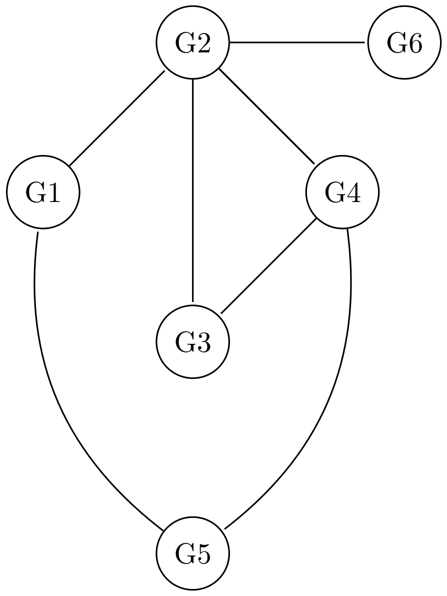
LISTSERVERS	provide list of servers known to this server, servers should be specified as GROUP_ID, HOST IP, and PORT comma separated within the message, each server separated by ;
eg.	V_GROUP_1,130.208.243.61,8888;V_GROUP_2,10.2.132.12,888;
KEEPALIVE	Periodic message to 1-hop connected servers, indicating still alive and processing messages. Do not send more than once/minute.
LISTROUTES	Provide as complete as possible list of all servers in the network and the shortest routes to them at the *time of command*. Nb. this should be timestamped, and use the table format shown in the Appendix below.
CMD,<TOSERVERID>,<FROMSERVERID>,<command>	Send a command to another server. This server can be *any* server in the botnet. If your server receives a CMD that is not for it, it should make best efforts to relay the message to the specified SERVER, otherwise it should execute the command provided that it conforms to this API.
RSP,<TOSERVERID>,<FROMSERVERID>,<command response>	Reply to a CMD from another server. The expected command response is specified for each command that requires a response. Nb.The recipient must understand your response, so do not deviate from the specification here - "be strict in what you send, but tolerant in what you receive"
FETCH,<no.>	Fetch a hashed string from this server with the supplied index R(1..5)(See below)

- Obtain bonus points for connectivity (maximum 3 points): Servers being claimed must be listed in the routing snapshots being submitted for the claim on at least 3 separate occasions, and the other group must also submit a routing snapshot including your server.(Yes, you can co-ordinate on this, both groups receive the points.)
  - 0.5 points for every server from Akureyri
  - 0.5 points for every server with an HMV group member
  - 0.5 points for every server not hosted on skel.ru.is
- Provide a chat transcript, and matching wireshark log of a chat with a client at another server. Both groups must provide a transcript and wireshark log, and supporting routing snapshots taken during the chat. (2 points)

Notes:

Monitor incoming traffic on your server, and in particular make sure all traffic is sanitised, and only fits the API provided. Observe good network programming practice and always check inputs for potential buffer overflows.

Appendix: Routing Table



Generate the routing table as the set of shortest paths from your node to all the others currently connected in the network on one line as shown below, with a single header for the file. If there is more than one shortest path, you only need to provide one. For 1-hop connections, the route is via the local server (i.e. G5), for multi-hop connections just list all the transit nodes, with a "-" between them. Please use the : character as a separator for each node

For example, for the network topology above, assuming your server is G5, the routing table should look like, using the tab character (:) to separate each node:

	G1:	G2 :	G3:	G4:	G5:	G6 :	
G5 :	G5:	G4,G1:	G4:	G5:	- :	G1-G2:	TIMESTAMP