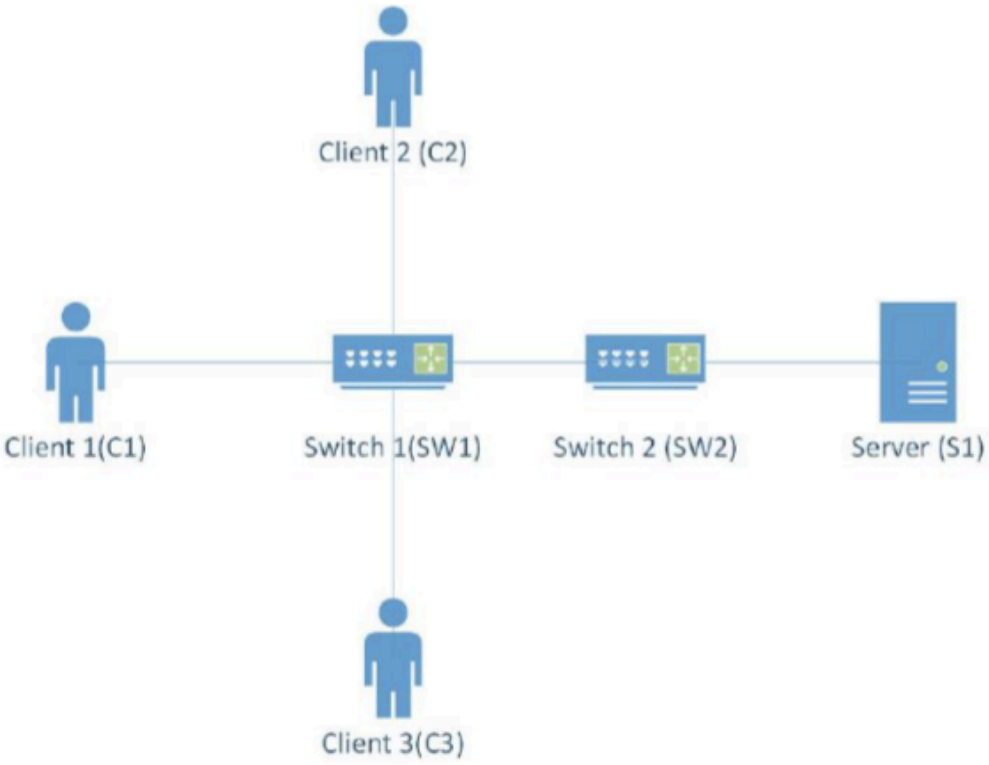
CS164 Final Project – Fall 19 - Mini Facebook

Part 1: Setup Topology

In this project, you will use Mininet to emulate a network topology consisting of one server and three client nodes and implement a simple Facebook application on top of this topology using a client - server model. Clients run client codes to make use of the capabilities they are provided with and the server is responsible for authenticating users, receiving posts and messages from users and sending them to the proper intended recipient. The topology you will be setting up looks like this:



This is your underlying network. To set it up, log into your Mininet VM, copy the file *finalTopol.py* to mininet and run this command:

```
mininet@mininet-vm:~$ sudo mn --custom finalTopol.py --topo finaltopol -x
```

Make sure you take a careful look at the file finalTopol.py and try to understand how the topology is defined. There are several errors/missing lines (4) in the file so you have to correct them before you run the above the command.

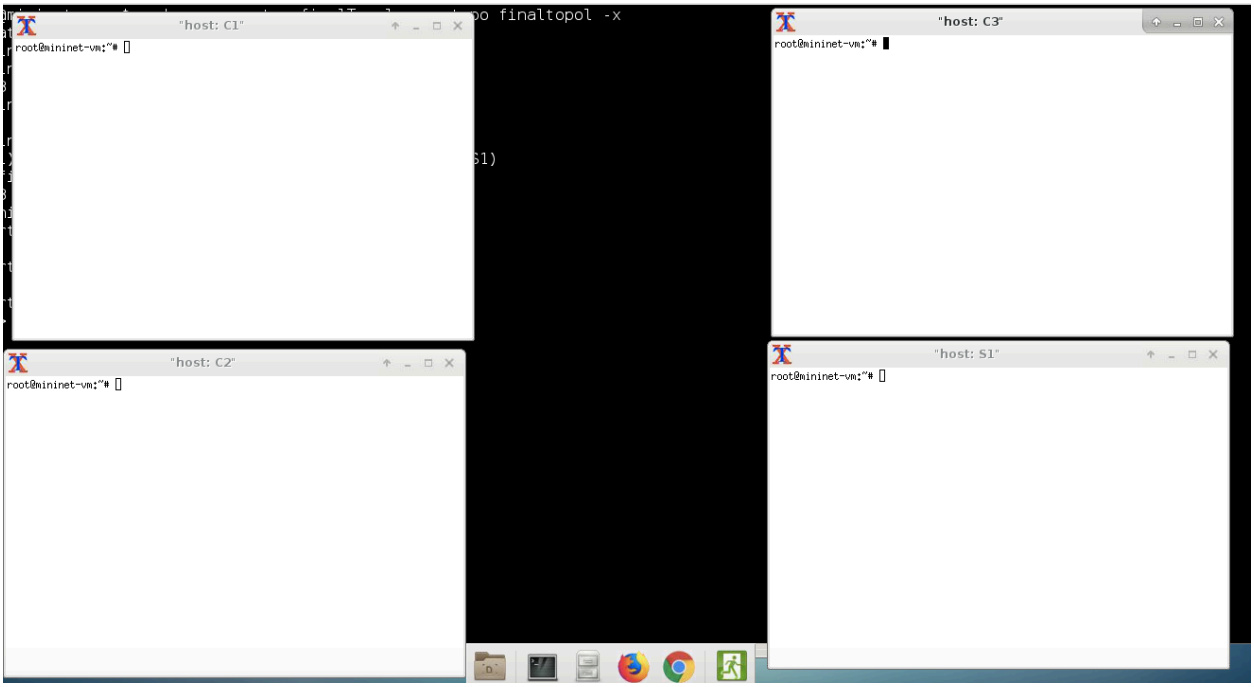
To learn more about Mininet (and to help identify the syntactical errors/missing lines), you can check the link below for a tutorial:

http://mininet.org/walkthrough/

After you have fixed the file and run the command, if everything is successful, you will see this:

```
mininet@mininet-vm:~$ sudo mn --custom finalTopol.py --topo finaltopol -x
*** Creating network
*** Adding controller
*** Adding hosts:
C1 C2 C3 S1
*** Adding switches:
SW1 SW2
*** Adding links:
(C1, SW1) (C2, SW1) (C3, SW1) (SW1, SW2) (SW2, S1)
*** Configuring hosts
C1 C2 C3 S1
*** Running terms on localhost:10.0
*** Starting controller
c0
*** Starting 2 switches
SW1 SW2
*** Starting CLI:
mininet>
```

Also, you will see a number of external terminals open for each of the nodes:



On each of these terminals, you can run any commands, including running the client or server codes from your previous labs. Note that the IP address for these nodes is not “127.0.0.1” anymore. To find out what each of their address is, you can run `ifconfig` on each terminal.

To make sure the network is set up correctly and all nodes are connected, you can use `pingall` in the main terminal. If everything is working fine, you will see this after running it :

```
mininet> pingall
*** Ping: testing ping reachability
C1 -> C2 C3 S1
C2 -> C1 C3 S1
C3 -> C1 C2 S1
S1 -> C1 C2 C3
*** Results: 0% dropped (12/12 received)
mininet>
```

Part 2: Program Client and Server Code – User Sessions

Next, you will write your server and client codes. You can also use simple telnet for clients if you can make it work. Note that NOTHING is stored on the client side. The server takes care of everything. For every functionality required by and available for client, you should modify the server program to support that.

It is recommended that you use localhost when writing your codes for ease of debugging, and after completing each phase, take it into the Mininet emulated network, run it and prepare it for the demo . Your code should be able to do the following:

1. Whenever a user connects to the server, they should be asked for their username and password.
2. Username should be entered as clear text but passwords should not (should be either obscured or hidden).
3. User should log in successfully if username and password are entered correctly. A set of username/password pairs are hardcoded on the server side.
4. User should be provided with a menu. The menu includes all possible options (commands) user can use and how they can use them. These options include Logout, Change Password, etc. As you add functionality, you can add new options to this menu.
5. Change Password: User should be able to change their password. To do this, old and new passwords should be entered (neither as clear text).
6. Logout: User should be able to logout and close their connections with the server.
7. A user should see their messages in real - time (live) if they are online when someone sends them messages.

Part 3: Program Client and Server Code – Private Messaging

8. Send Message: A user should be able to send a private message to any other user (whether or not the recipient of the message is online) .
9. View Count of Unread Messages: A user should see the number of unread messages when logging in.
10. Read Unread Messages: A user should be able to read all unread messages.

11. Send Broadcast message: A user should be able to send a message to the server which only forwards to all clients who are currently connected.

Part 4: Program Client and Server Code – Extra Credit (OPTIONAL)

12. Send Friend Request: A user should be able to send any other user a friend request.
13. See Friend Requests: A user should see the number of unanswered received friend requests when logging in.
14. Respond to Friend Request: A user should be able to see all friend requests and decide on them one by one (with Accept or Reject).
15. When a friend request is accepted, the two users become friends. Therefore, they will be able to see each other's status updates. Assume all status updates' visibility is friends - only; meaning only friends can see each others' status posts (they are not public).
16. Post Status: A user should be able to post status updates .
17. See Timeline (Wall): A user should be able to see their timeline at any time, which is the history of all of their past status update posts, in chronological order.

Demo: This project takes up the last lab sessions. You should demo phases 1, 2 and 3 during lab session 8, 9 and 10 respectively or during the TA office hours the week of the lab. During the demo you will run the code to show the functionality is working with no errors and explain how you programmed the features. Note that we will **not** restart the server during the demo so you do not necessarily need to connect to a database to store data.

You are expected to work on each phase during that week, demo and upload your code by Saturday 11:59pm that week. The code file(s) should have the format filenameX.py where X indicates parts 1, 2 or 3 for that week. You will be graded based on the weekly deadlines for this project and the final code for the entire project will be due by Saturday 11:59pm in the week of lab 10. The extra credit part has 10 bonus points which can make up for points you may lose on THIS project .