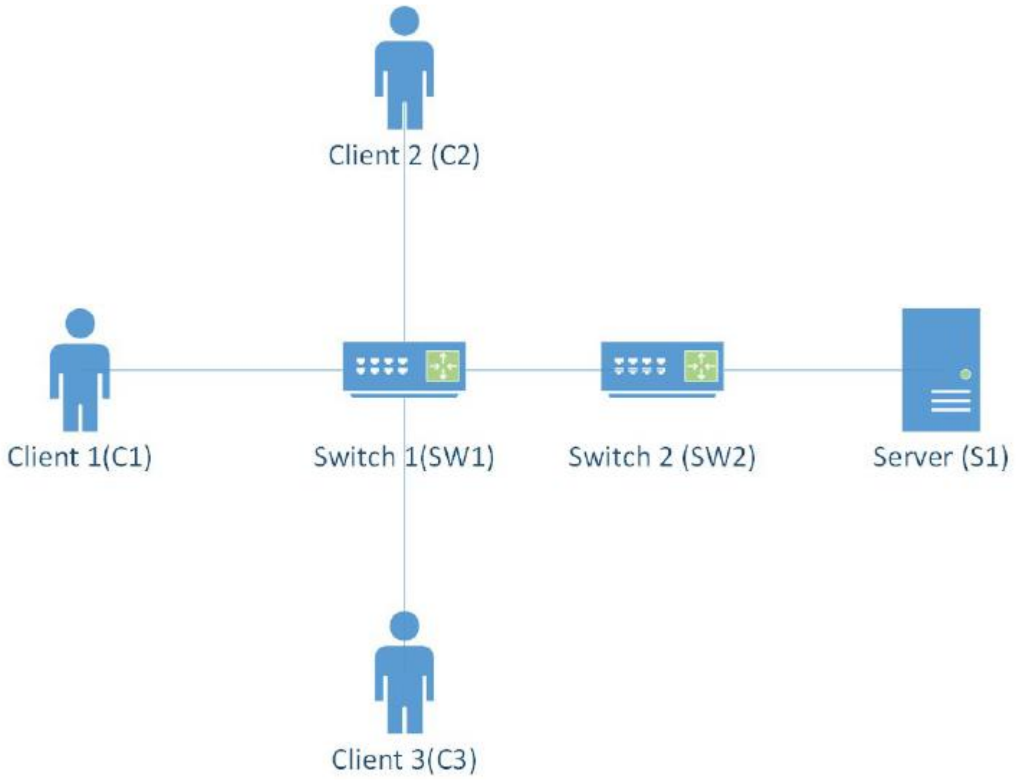
CS164 Final Project – Fall 16

Mini Facebook

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 server is responsible for authenticating users, receive posts and messages form users and send and display them for the proper intended users. The topology you will be setting up looks like this:

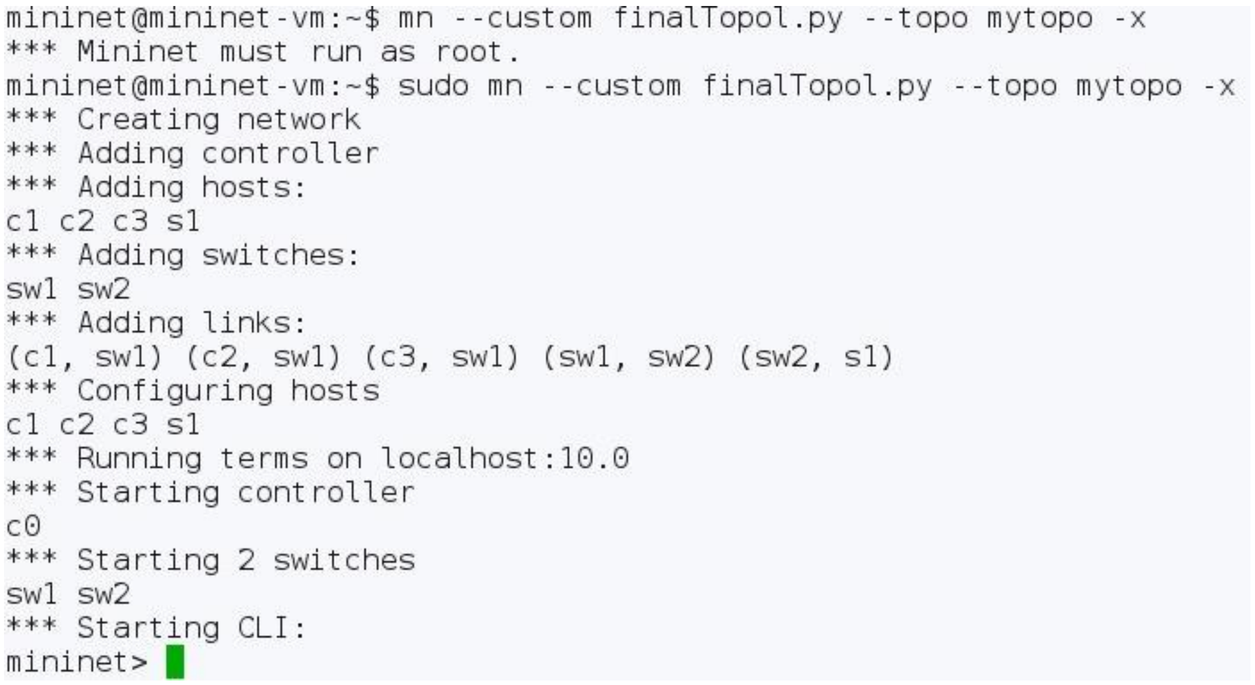


This is your underlying network. To set it up, log in to your Mininet VM and run this command:

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

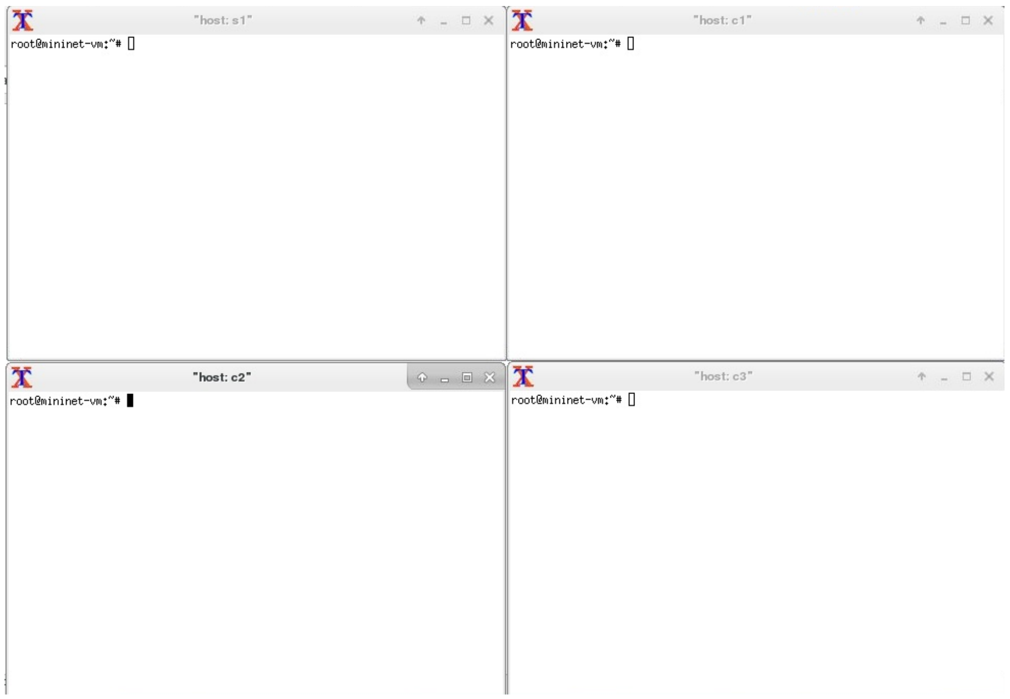
The file "finalTopol.py" is provided. Make sure you take a careful look at it and try to understand how the topology is defined. There is actually a syntactical error in the file so you have to correct it before you run the above the command.

After you run the command and everything is successful, you will see this:



```
mininet@mininet-vm:~$ mn --custom finalTopol.py --topo mytopo -x
*** Mininet must run as root.
mininet@mininet-vm:~$ sudo mn --custom finalTopol.py --topo mytopo -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:



```
root@mininet-vm:~#
```

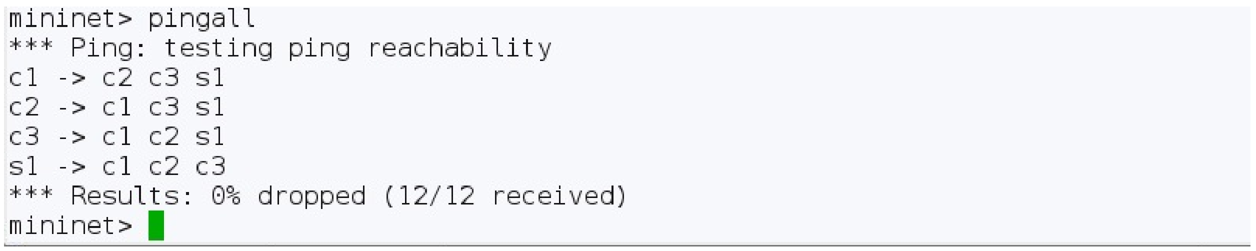
```
root@mininet-vm:~#
```

```
root@mininet-vm:~#
```

```
root@mininet-vm:~#
```

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> 
```

To learn more about Mininet, you can check the link below:

http://mininet.org/walkthrough/

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.

Phase 1 – User sessions

- Whenever a user connects to server, he should be asked for his username and password.
- Username should be entered as clear text but passwords should not (should be either obscured or hidden).
- User should log in successfully if username and password are entered correctly. A set of username/password pairs are hardcoded on the server side.
- User should be provided with a menu. The menu includes all possible options (commands) user can use and how he can use them. These options include Logout, Change Password, etc. In every phase you should add new options to this menu.
- Change Password: User should be able to change his password. To do this, old and new passwords should be entered (neither as clear text).
- Logout: User should be able to logout and close his connection with the server.

Phase 2 – Private Messaging

- 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).
- A user should see the number of unread messages when logging in.
- See Unread Messages: A user should be able to read all unread messages.

- A user should see his messages in real-time (live) if he is online when someone sends him a messages.

Phase 3 – Friendships

- Send Friend Request: A user should be able to send any other user a friend request.
- A user should see the number of unanswered received friend requests when logging in.
- 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).
- A user should see his received friend requests in real-time (live) if he is online when someone sends him a friend request.
- 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).
- Post Status: A user should be able to post status updates.
- See Timeline (Wall): A user should be able to see his timeline at any time, which is the history of all of his past status update posts, in chronological order.
- See News Feed (Homepage): A user should be able to see his news feed, which is the last ten status updates by his friends, in chronological order.

Extra Credit – Likes and Comments

- Comment: A user should be able to write a comment for his or any of his friends' status updates.
- Like: A user should be able to like his or any of his friends' status updates.
- A user should see the number of new comments and likes on his posts when logging in.
- A user should see his new received comments and likes in real-time (live) if he is online when someone likes or writes a comment for one of his posts.
- Likes and comments should appear on timeline and news feed right next to their associated status updates.

Demo: This project takes up your last three lab sessions. You should demo phases 1, 2 and 3 on lab session 8, 9 and 10 respectively. You are expected to work on each phase during that week, upload your code before coming to lab and demo it during the lab session. You will have to meet the weekly deadlines for this project, and you will be graded based on that. The extra credit part has 25 bonus points which can make up for points you may lose on THIS project.

Good Luck!