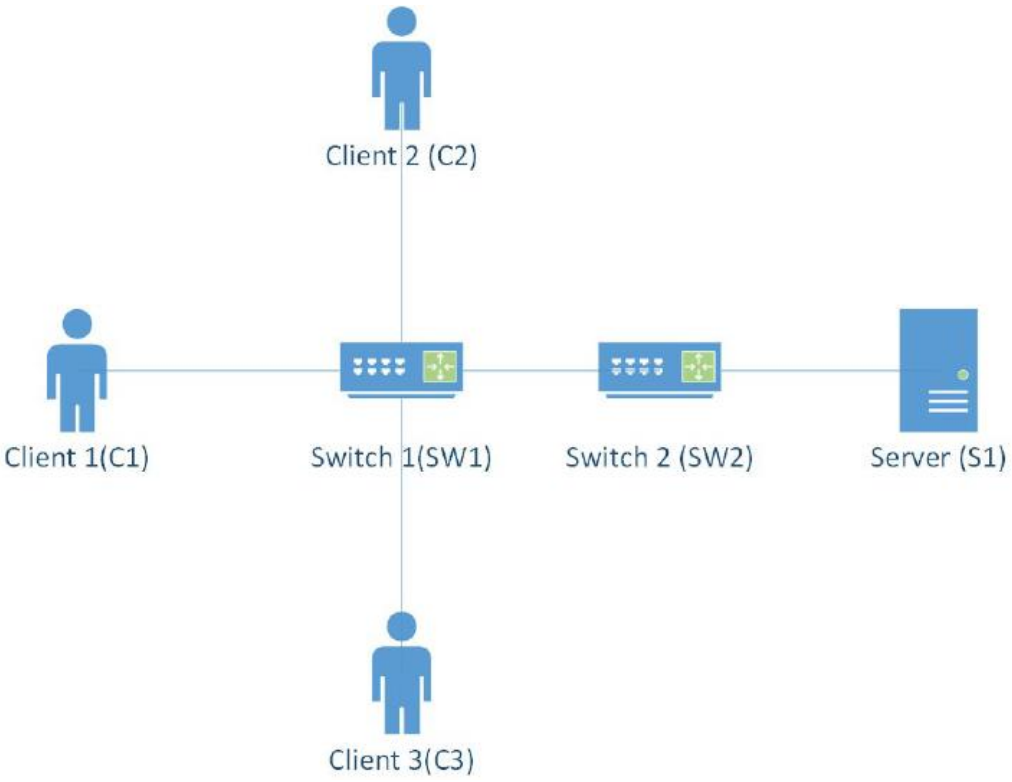
CS164 Final Project – Fall 17 Deadline: December 8th, 2017 (Last Lab session)

Multi-Player Hangman

In this project, you will use Mininet to emulate a network topology consisting of one server and three client nodes and implement a multi-player Hangman game 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 player actions form users and apply 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:

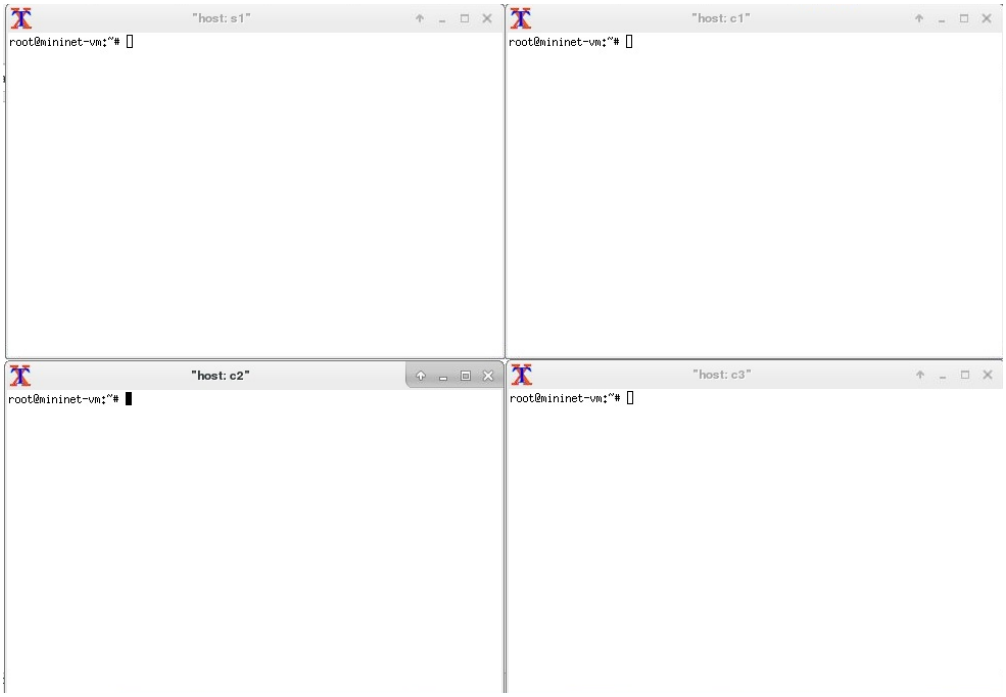
```
mininet@mininet-vm:~$ sudo mn --custom finalTopol.py --topo mytopo -x
```

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 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:~$ 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 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>
```

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 them, take it into the Mininet emulated network, run it and prepare it for the demo.

Client-side interface

The initial menu the client sees on his screen (after connecting to the server) has the following options:

1. Login
2. Make New User
3. Hall of Fame

   (Server sends list of the top scorers in the game)
4. Exit

   (client Exits the game)

To sign up, client picks option 2 (make new user) and this next menu shows up:

What is Your User Name?

(User types a username)

What is Your Password?

(User types a password)

If the Username is not already taken, this account is made and client returns to initial menu.

After Selecting option 1 (login) server sends a message to client and ask user to type his username first and send it and type his password after it, if they are correct, user is logged in to the system. If they are correct, the following menu is shown:

1. Start New Game
2. Get list of the Games
3. Hall of Fame
4. Exit

After selecting option 1 (Start New Game), user gets to choose the difficulty level of the game via the following menus:

Choose the difficulty:

1. Easy
2. Medium
3. Hard

The user types the difficulty level, then the game starts. The difficulty level determines the number of rows of allowable wrong letter guesses for the word.

By picking the 'Get list of the games' option, user is presented with the list of already started games, and he can pick one of those games to join.

Throughout the game, players can join and participate in the game. With every move a player makes, the result will appear on all participating players' screen. An example game information, for some instance of time during the game for the word "hangman", looks like the following:

h _ _ g _ _ _ (correctly guessed letters)

e i r b (incorrectly guessed letters so far. In easy mode, they are three times the number of word's letters (i.e. three rows), in medium mode they are twice, and in hard mode they are equal)

Ali 1

Mohammad 1 *

Marcos 0

The above also shows the list of players playing the game right now, together with their current scores and also indicating whose turn it is next (marked by *).

Guessing each letter correctly adds 1 to the score of that player and the score for guessing the whole word correctly equals the length of the word (which is added to the winner's previous score from guessing individual letters; e.g. if Mohammad guesses the whole word correctly now,

his score would be 1+7=8). Obviously, words only consist of lower case alphabets letters (no numbers or special characters).

The game is round-based and each person can guess one letter at his turn. If he guesses it correctly, he can guess another letter too. But on an incorrect guess the turn is given to the next player.

A player can guess the whole word any time during the game (even if it is not his turn). If his guess is incorrect, he has lost the game and is removed from the game. (No update will be sent to him and his client needs to show the previous menu for selecting another game or making a new game) You can simply use number of letter in the typed guess string to distinguish between the two cases. If only one character is typed, it is single letter guess but if it is more than one, it should be counted as whole word guess.

Server-side interface

Server has a menu for game admin and has the following options:

1. Current list of the users

   (Print list of the current users and return to the previous menu after printing the list)

2. Current list of the words

   (Print current words that can be selected by server randomly as the word)

3. Add new word to the list of words

   (New word is typed and it is added to the word list)

Demo: You will need to show properly working client and server codes, on the correct topology set-up and correct flow of the game.

Best of Luck!