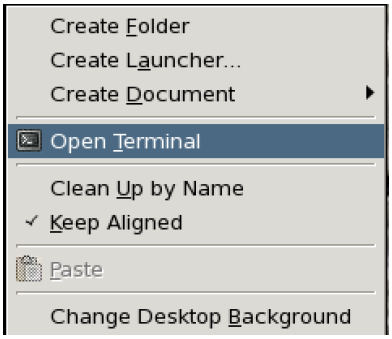
Lab 1

This lab has four parts. The first two parts will be about the network utilities `traceroute` and `whois`. The third part will be about mininet which is a software environment that will allow us to simulate a wide range of real-world networks in future labs. The last part will be about the differences between persistent and non-persistent HTTP connections.

Note: There will be a number of commands that will have to be typed into a terminal. They will all have the following format:

```
$ <command> <args>
```

where the `$` represents the command line prompt. Each command should be typed exactly as it appears and must be completed by hitting the enter key once typed. If you are unfamiliar with the lab's Linux environment, once you've logged in, you can open a terminal by right-clicking on the desktop and selecting the appropriate option as seen below:



Part 1: traceroute

The traceroute utility, as the name implies, is a tool for tracing the route a packet might take as it travels between two hosts on a network. It prints out the IP addresses for each of the "junctions" that the packet must go through on its route. The following describes how to use `traceroute` from a lab computer.

1) `traceroute` can only be executed successfully from a computer on the CS network named `hammer`. We must first open a terminal and execute the following:

```
$ ssh hammer
```

2) The first time you `ssh` to a computer, you will see a message about the authenticity of the host. Dismiss this message by typing "yes". Then, enter in your CS account password. Once you receive a prompt, all the commands you enter into the terminal will be executed from `hammer` instead of from your current lab computer.

3) To get a sense for the command, type:

```
$ traceroute google.com
```

The output should be about 9 lines long with each line containing an IP address. Each of these IP address represents a computer/router/switch between `hammer` and the particular Google server servicing the `traceroute` request.

4) The output of the `traceroute` command (and many others) can be saved to a file by using output redirection. For instance, in order to save the output of the above `traceroute` command to a file called "traceroute.txt", use the following command:

```
$ traceroute google.com > traceroute.txt
```

5) The file you just created can now be opened in any text editor and the results of the traceroute command will be found there. The following will display the file on `hammer`:

```
$ vi traceroute.txt
```

Demo: Recall that the URL's for websites around the world can end with different suffixes depending on the country in which they are hosted. For instance, some-website.co.uk implies that the website in question is hosted in the United Kingdom. Keeping this in mind, use the `traceroute` command to trace two routes. The first should be between `hammer` and a server in Asia. The second should be between `hammer` and a server in Europe (excluding the UK). Output the results of these commands to files called "traceroute1.txt" (for the Asia trace) and "traceroute2.txt" (for the Europe trace). Be prepared to show these files to your TA.

Part 2: whois

`whois` is a utility that allows a user to discover more information about the registered owner of an IP address or domain name. To see what kind of information the `whois` utility provides, type the following:

```
$ whois google.com
```

Scroll through the output. Notice that some of the details included are addresses and phone numbers for those who own the domain name (in this case, Google).

Demo: Review the output from your `traceroute` commands in Part 1. Pick two IP address from each of the traces (whichever seem most interesting to you) and use the `whois` command to find out more about those IP addresses. Output the results of those four `whois` commands to four different files named "whois1.txt", "whois2.txt", etc. Be prepared to show those files to your TA and explain what you were able to discover about the IP addresses you researched with `whois`.
(You can close your connection to hammer server now, because we do not need it in the rest of the experiment)

Part 3: Setting up mininet

The software used for this part of the lab is free and can be installed on a computer of your choice. The links are here for mininet and here for VirtualBox. What follows are instructions for

setting up your mininet environment on a lab computer which is all that is necessary to complete the labs for this class and all that the TA's will be able to help you with. If you decide to install software on your own computer, it will be up to you to make sure it works properly for use in the labs. Fortunately, if you choose to go this route, the websites linked to above have good resources for helping you get started.

To set up mininet on a lab computer, we will need to first create a virtual machine in which to run it. Fortunately, all the files and software necessary are already installed on the lab computers so the following steps shouldn't take more than about 20-30 minutes if followed carefully.

1) Open a terminal and enter the following commands:

```
$ cd /extra
$ ls
```

The purpose of these commands is to list the contents of the `/extra` directory. You should see a directory named after your cs account name (hereafter referred to as `<your directory>`). This was created automatically for you when you first logged in and is stored locally on the lab computer you are currently logged in to. This is important to note since this directory will be where all of your mininet-related files will be stored: on this specific lab computer only. You will only be able to access your mininet environment by accessing this lab machine. Take note of where you are sitting and the name of the lab machine. You can get the name with:

```
$ uname -n
```

The name will end with `cs.ucr.edu`. Later, we will show you how to use your lab machine's name to access this computer remotely (from another lab machine or from home).

2) Copy the zipped mininet disk image into your directory:

```
$ cp mininet-2.2.0-150106-ubuntu-14.04-server-amd64.zip <your directory>
```

3) Unzip the file:

```
$ cd <your directory>
$ unzip mininet-2.2.0-150106-ubuntu-14.04-server-amd64.zip
```

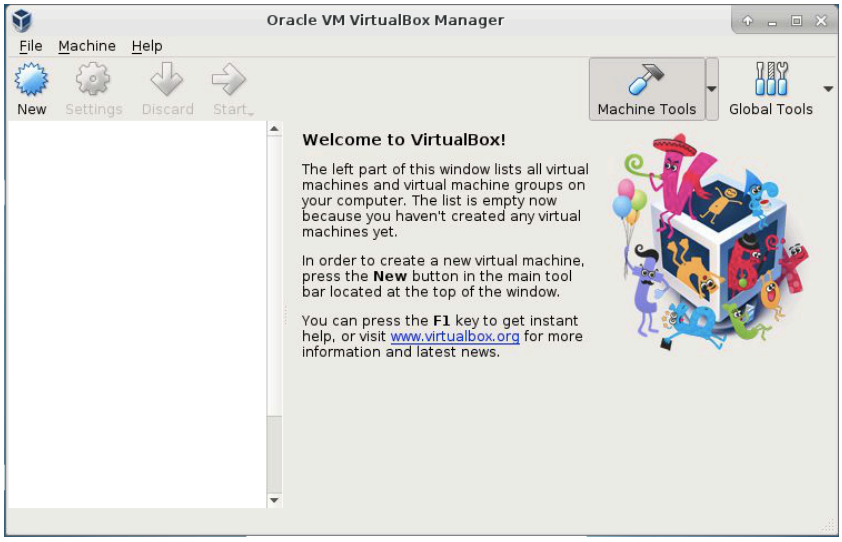
This last command may take a minute or so to complete.

4) We will now use the unzipped file to create a virtual machine in VirtualBox. Start up VirtualBox with:

```
$ VirtualBox &
```

If you're not familiar with the syntax, the `&` allows VirtualBox to run "in the background" which means that you can still use your current terminal to enter commands while VirtualBox is running.

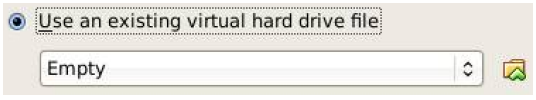
5) Ignore any messages about updating VirtualBox by clicking "Cancel". Your window should look like this:



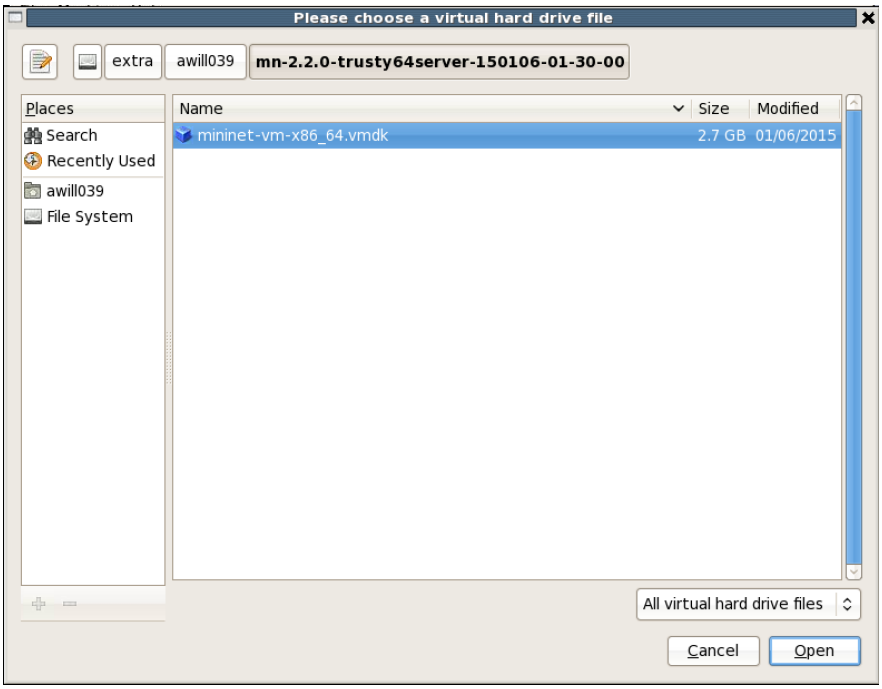
6) Click on the "New" button in the upper left. In the window that appears, enter the settings as below and click "Next".

7) Set the memory size to be 1024 MB and click "Next".

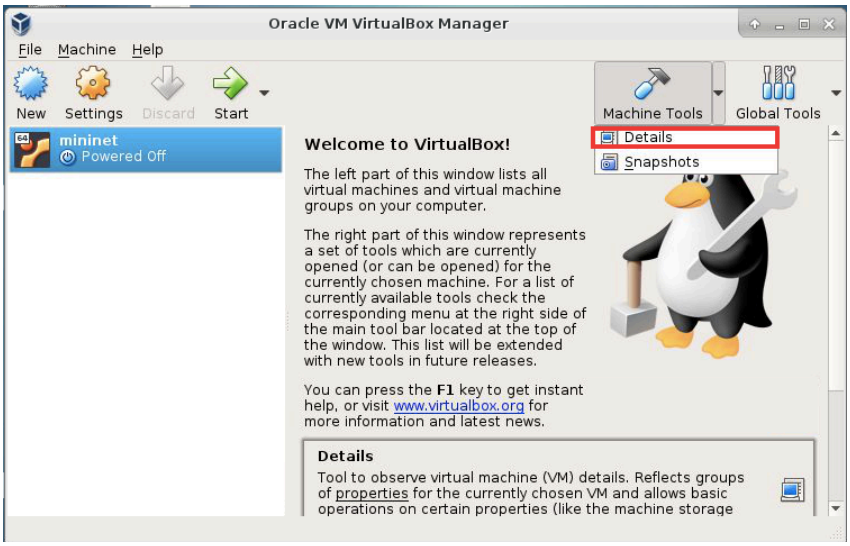
8) Select "Use an existing virtual hard drive file" and click the folder-with-green-arrow icon shown in the bottom right corner below:



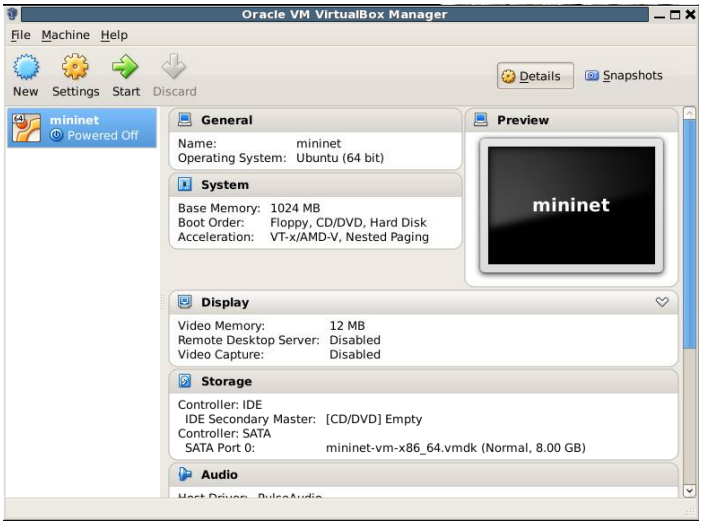
9) Navigate to the "mininet-vm-x86_64.vmdk" file seen below and click "Open". Note that the file path for this file can be seen in the top part of the image (you should see `<your directory>` instead of `awill039` in the file path).



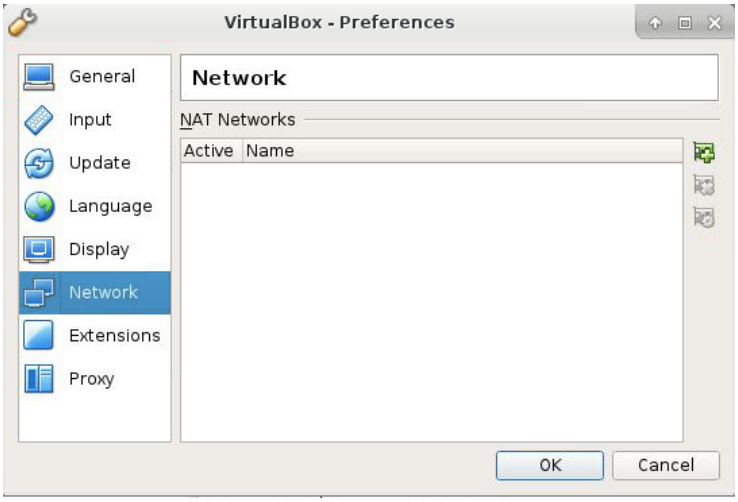
10) Click the "Create" button. You should now see something similar to this. If you see this window, then you have successfully created the virtual machine. Since we used a ready-made disk image to create this VM, the VM will be pre-loaded with the software necessary to run mininet. We will now setup the VM's network settings.



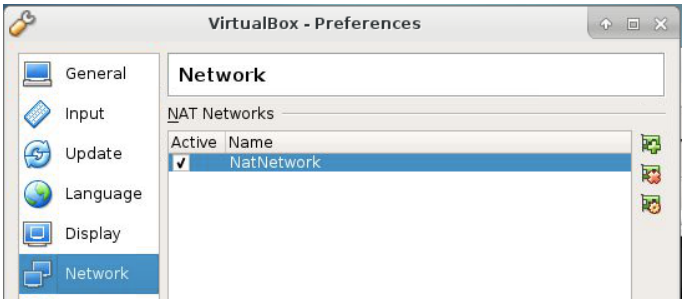
Click Machine Tools-> Details to change to the next screen shot



11) In the "File" menu, select the "Preferences..." option. In the window that comes up, select "Network" on the left. Your window should look like this:



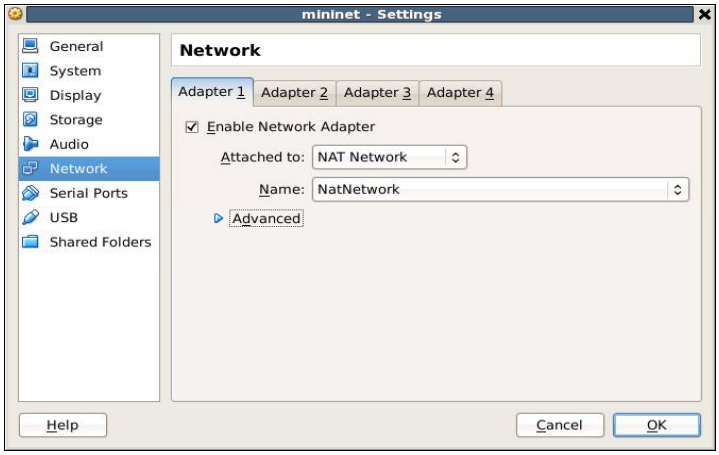
12) Click the green icon on the right. Our window now lists something called "NatNetwork".



Click "OK".

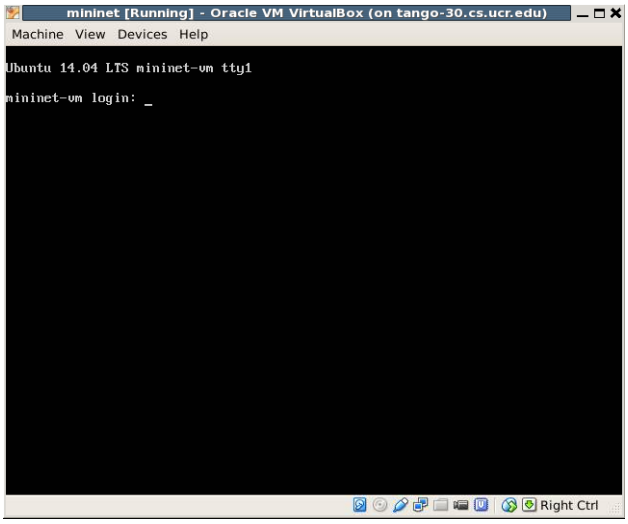
13) Back in the main window, use the scroll bar on the right hand side to scroll down. We're looking for the "Network" section. It should like the following:

When you arrow over the word "Network" the text will turn blue as above. You can now click the word itself to open up the virtual machine's network settings. Make your network settings look like the following:



Click "OK".
(Please notice the fact that the "Attached to" value is set to "NAT Network" and not "NAT".)

14) At this point, we can start up the VM. From the VirtualBox Manager window, click the green-arrowed "Start" button. You may see a warning about the lack of audio devices or one about the host I/O cache, these warnings can be dismissed. After a minute or so, the VM should be fully booted. You should see a login prompt:



The user account you will use to log in has already been created. The username is `mininet` and the password is `mininet`. Go ahead and log in.

15) For the sake of practicing good security measures, we will now change the password for the mininet user. Type the following command:

```
$ passwd
```

You will be prompted to enter in the current password and then a new password (twice). Make it something that you will be able to remember in a few weeks time if necessary (or keep a record of it).

16) The very last step is to shutdown your VM. You can do this by simply clicking the "x" in the top right corner of your VM's window and selecting "Power off the machine".

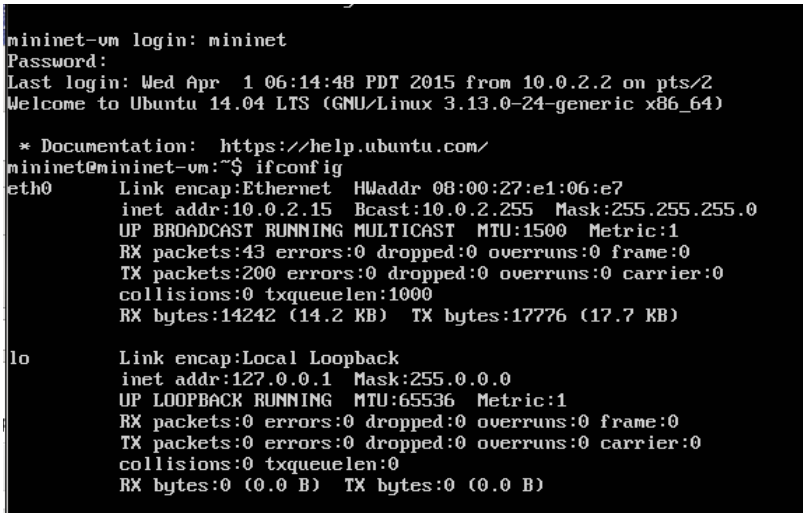
This completes the setup of the mininet environment for lab 1. Future labs will discuss how to access your VM remotely.

Demo: Be able to show your TA that you can boot your VM and log into the mininet account.

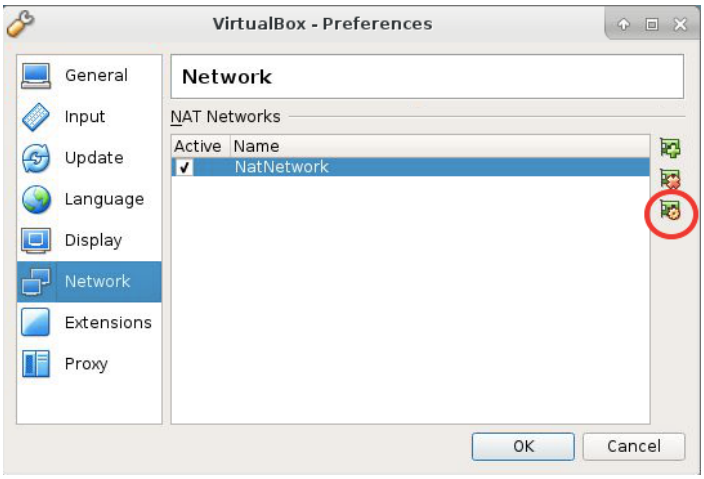
Part 4: Persistent Vs. Non-Persistent HTTP

HTTP uses TCP connections to exchange data. This connection can be persistent or non-persistent. Persistent connections use one TCP connection for transferring multiple objects. But, in the non-persistent version, just one object can be transferred in one connection. In this part we are going to observe the effects of using persistent connections in the HTTP protocol.

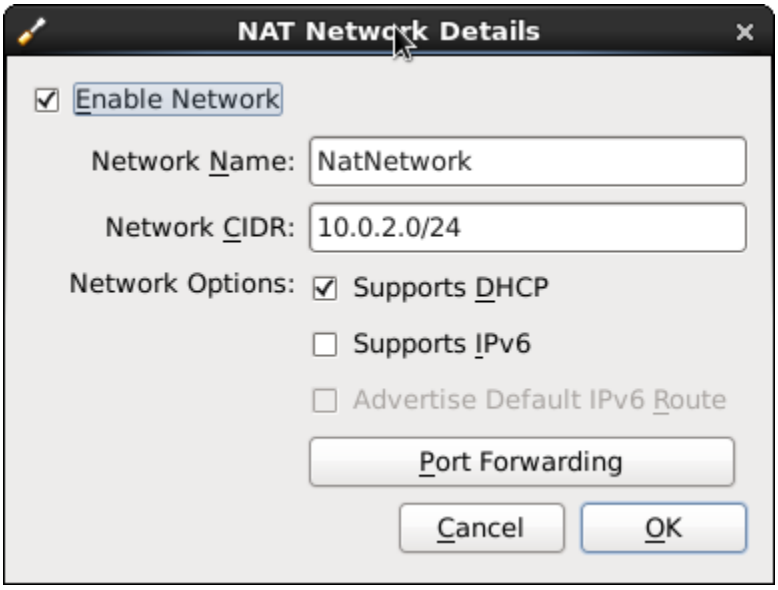
1) If you have turned your VM off, turn it on again and log in, because we are going to run this experiment on the VM system.
2) In the next steps we will need the IP address of your VM. To find the IP address of a system, we can use the `ifconfig` command. By using this command you will see the list of system interfaces (as below). Write down the "inet addr" of "eth0". You will need it later. It will probably be the same one you see in the picture (10.0.2.15).



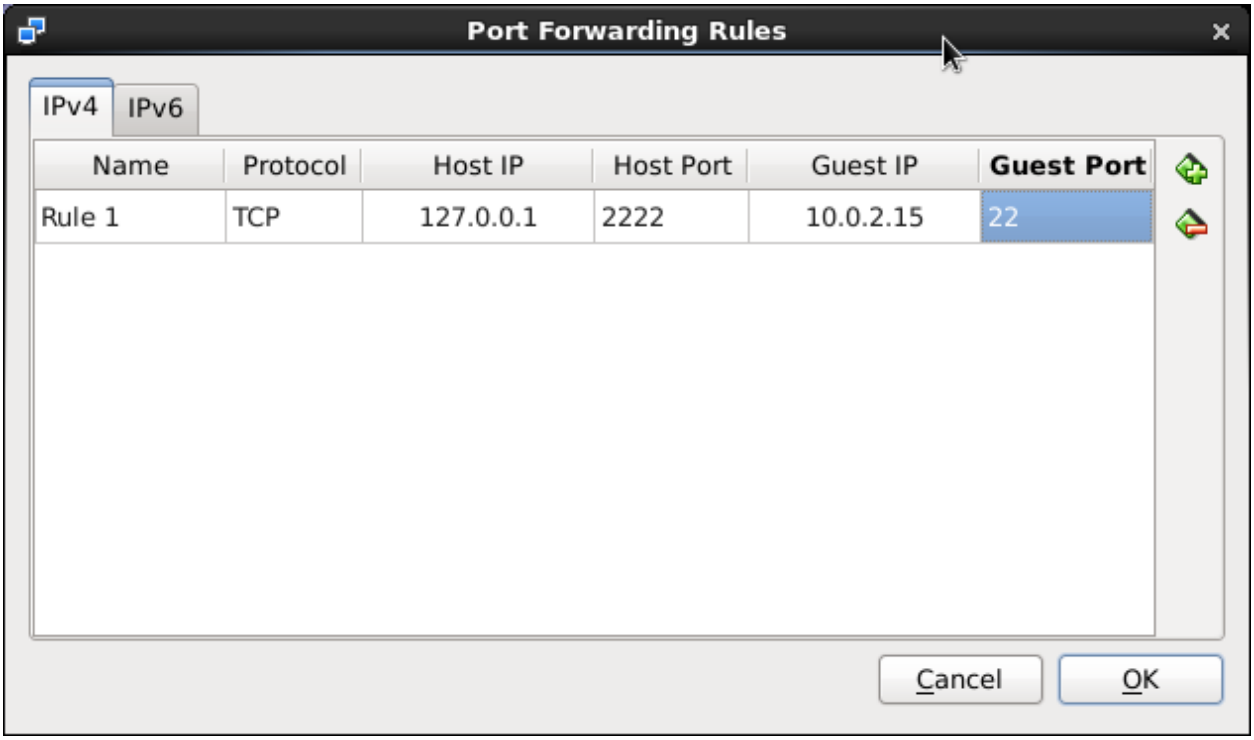
3) In the next step we want to enable port forwarding in the virtual network of VirtualBox. In VirtualBox Manager window, go to File -> Preferences. Click on “Network” in the sidebar. Select “NatNetwork” and click on edit NAT network icon.



Click on Port Forwarding button to add a new port forwarding rule to this NAT network.



Click on insert new rule (the icon with the green plus) and fill the table with the values seen below. Set the name to be whatever you prefer, leave the Protocol as TCP, set the host IP to 127.0.0.1 and Host Port to a number larger than 2000 (like 2222 which is not a bad choice). Set the Guest IP to the IP of the VM that you recorded previously. The guest port must be 22, because the default port of a SSH server is 22.

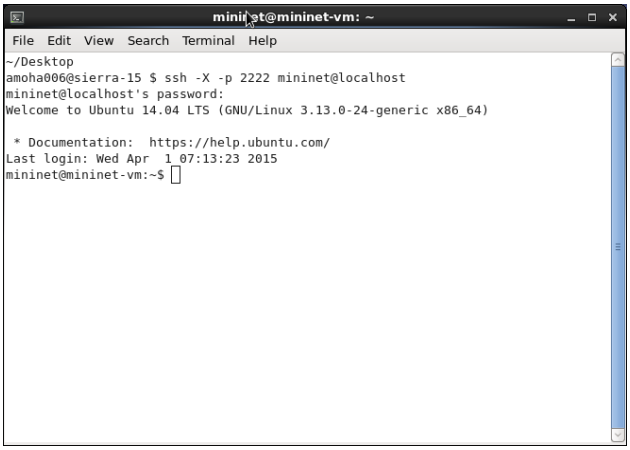


Then press “OK” on all the open windows.

4) Open a new terminal on your lab machine. Type the following command:

```
ssh -X -p 2222 mininet@localhost
```

After entering your password, you will be connected to your VM as the mininet user through a SSH connection. Your terminal should look similar to this:



Note that the “-X” option in the `ssh` command provides X forwarding for you. This means you can run an application with a graphical user interface over this SSH connection. The application we want to run is called Wireshark. The steps to use Wireshark are below.

5) Wireshark is an application which monitors packets coming into and going out from network interfaces. Before we can use it, we have to set Wireshark up to run under the mininet user.

a) Run the following command:

```
$ sudo dpkg-reconfigure wireshark-common
```

and select the “yes” option.

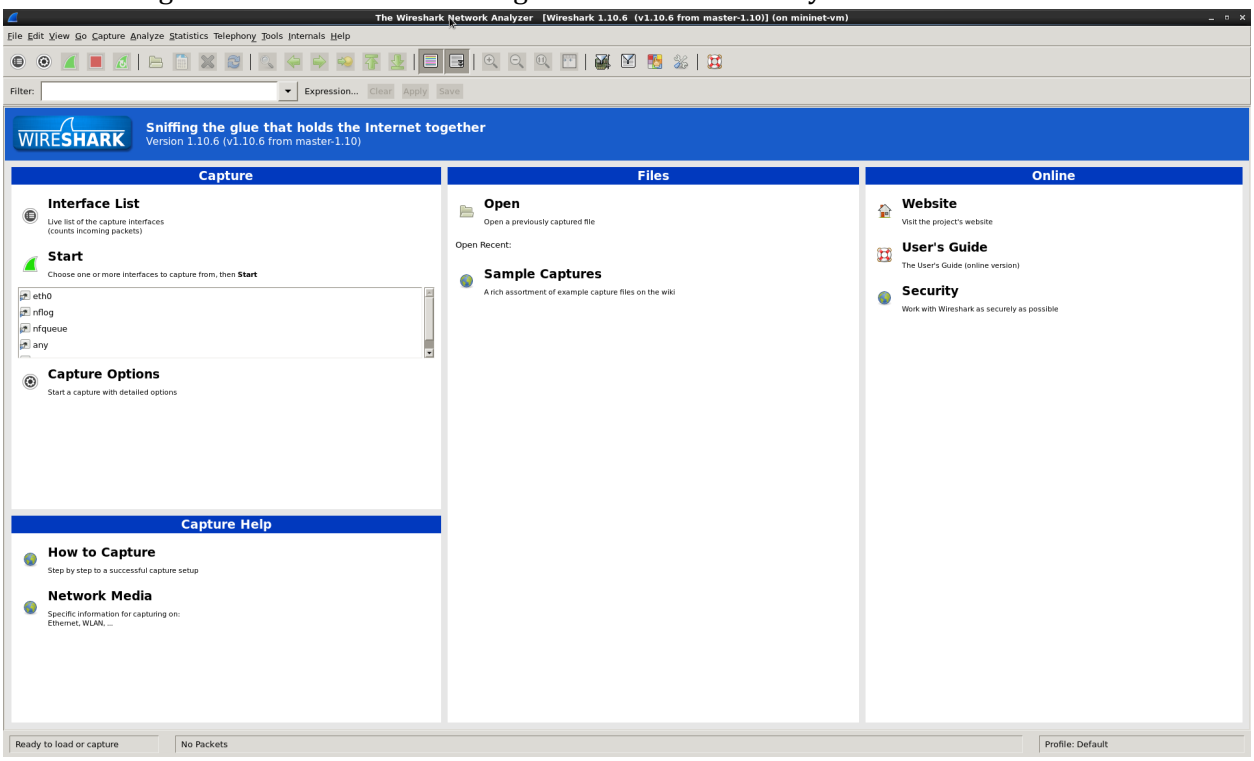
b) Then run:

```
$ sudo usermod -a -G wireshark mininet
```

6) We need to make these changes stick so type “exit” at the prompt and log in again using the ssh command above. Now we can run Wireshark:

```
$ wireshark &
```

After running this command the following window will show on your monitor.



NOTE: If the wireshark screen freezes and you cannot click or input any parameter close the window and follow the steps in Appendix 1 to access wireshark. Once wireshark is open and you can return here for the rest of the lab

7) Select the “eth0” interface in the “Interface List” on the left and click “Start”. You will see that this program will start capturing packets and it will show them to you. We will use this method to monitor the packets which are exchanged in HTTP connections to a website.

8) Go back to the terminal that is connected by SSH to your VM and type the following command:

```
$ wget -r -l 1 -e robots=off http://www.cnn.com -U mozilla
```

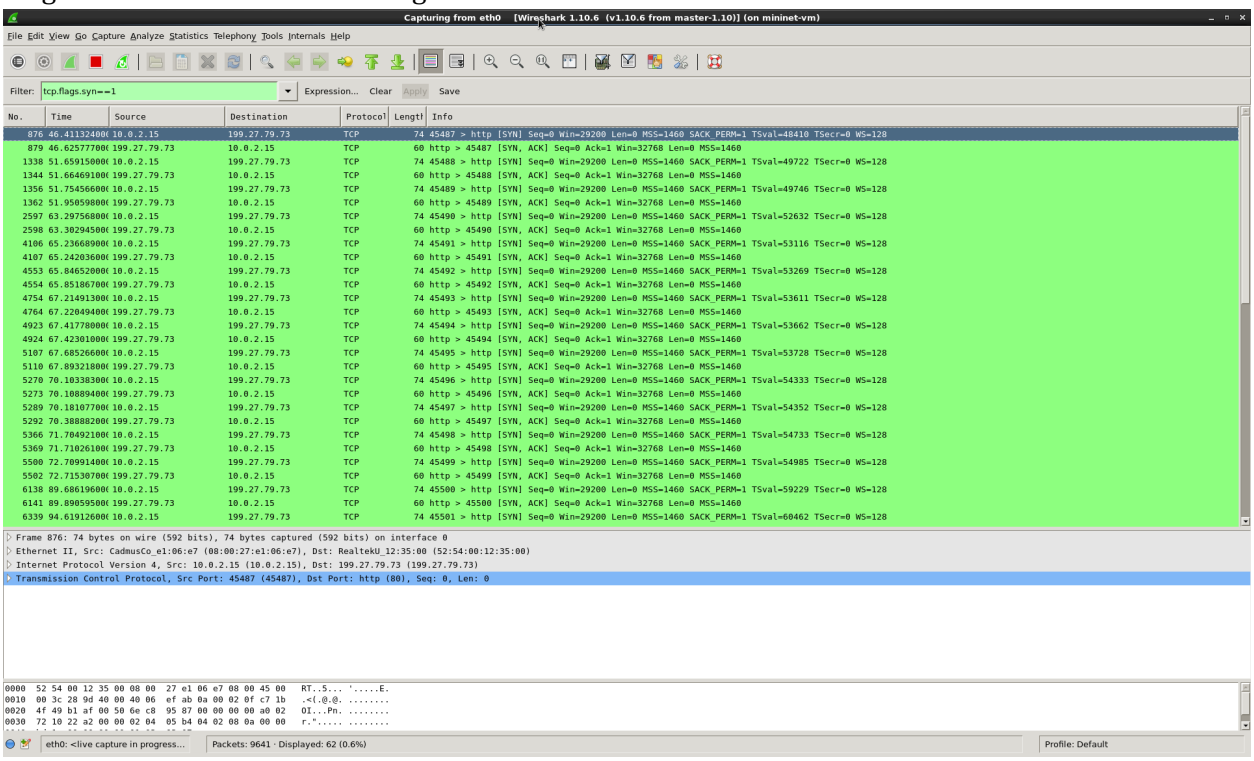
This command sends an HTTP request to http://www.cnn.com. The “-r” flag means that it will send a recursive request and the “-l” flag shows the depth of recursion, which is one here. The “-e robots=off” means that the command should ignore the site’s request to avoid crawling it. Finally, “-

U mozilla” sets the agent string to “mozilla”. The agent string is a description used by the web server to help it to detect the browser/system of the user requesting the information.

9) Go back to the Wireshark and in the filter field type:

```
tcp.flags.syn==1
```

Hit enter. This filter shows only the packets that have their syn flag set. Each packet with a set syn flag shows the beginning of a new TCP connection. You can find the number of these packets in the bar in the bottom of Wireshark next to where it says “Displayed”. Each packet has a timestamp. The difference in timestamps between the first packet and the last one allows you to calculate the total length of time that data was being transferred.



10) Write down all the numbers described above and then stop the wireshark and start it again. There is no need to store the packets, so select do not save packet option when you stop it.

Now run the following command:

```
wget -r -l 1 -e robots=off --no-http-keep-alive http://www.cnn.com -U mozilla
```

Go to the Wireshark and repeat the filtering, counting, and calculating time interval steps. Write down your results and be ready to explain the reason for the difference to your TA.

Demo: Show the last wireshark window and explain why there is a difference between the two cases.

Appendix 1: Accessing Wireshark in Mininet

On your lab machines (or personal computer):
1. Open X2Go Client (Applications -> Internet -> X2GoClient)

2. Create New Session
   Host: bolt.cs.ucr.edu
   Login: yourUCRNetID (eg eliri001)
   SSH port: 22
   Session Type: XFCE

3. Connect to this new session using your CS password
If you see bash errors
   SSH to 'bolt' from Terminal
   open file ~/.bashrc
   comment out the lines that cause the error

4. From bolt in X2GoClient screen, open Terminal and do 'ssh -X yourLabMachine'
   (e.g. if you are using computer number 15 at room wch129, YourLabMachine would be 'wch129-15')
Now that you have ssh-ed to your machine, you can ssh to mininet and bring up wireshark

```
ssh -X -p 2222 mininet@localhost
```