Twisha Sharma
CSE 150
Parsa

PreLab 3 - Introduction to SDN and OpenFlow

Assignment

[18 pts] Introductory Questions:

1. [10 pts] Use your own words to answer the following questions. For each question, provide your reference from either the list of References above or from within this document:

a) What is Software Defined Networking?

It is an approach to networking that socrates the control and the data plane within network devices. This allows for more flexible and programmable network management.

b) What is OpenFlow?

OpenFlow is a communication protocol that enables communication between the control plane and the data plane. OpenFlow is a key component of SDN as it allows the central controller to manage and control the behavior of network devices.

c) What is Mininet?

Mininet is an emulator that is used for creating virtual networks. The user can create a virtual network on a single machine.

d) What is the purpose of a Controller in Mininet?

In mininet, teh controller serves as the “brain” that manages and controls the behavior of the switches in the network.

e) What is the POX Controller? How is it different from the Mininet default Controller?

POX is a SDN controller within python. It serves as a platform for making custom controllers and also supports various SDN protocols. POX is open source meaning users can access and modify its source code.

2. [4 pts] Invoking a Controller in Mininet:

In the Mininet environment different controllers can be activated:

- “sudo mn” invokes Mininet along with its internal controller
- “sudo mn --controller=remote” invokes a remote controller.

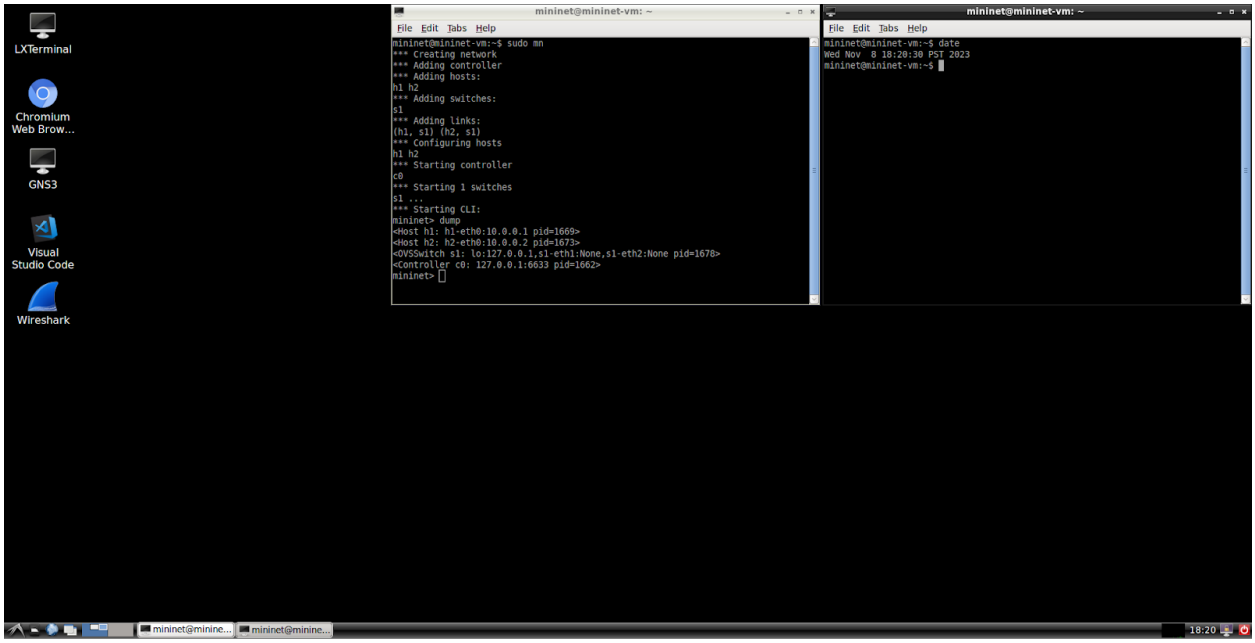
a. What command would you use to invoke the remote controller with IP Address 127.0.0.1 and port 8900?

sudo mn --controller=remote,ip=127.0.0.1,port=8900

b. [pts] After running “sudo mn” in a terminal, run the command “dump” after Mininet has finished setting up. What is the IP address and Port number of the controller being used by Mininet? Include a timestamped screenshot of

the output of the command and circle in red separately the IP address and Port number being used by the controller.

IP Address of the controller is 127.0.0.1:6633



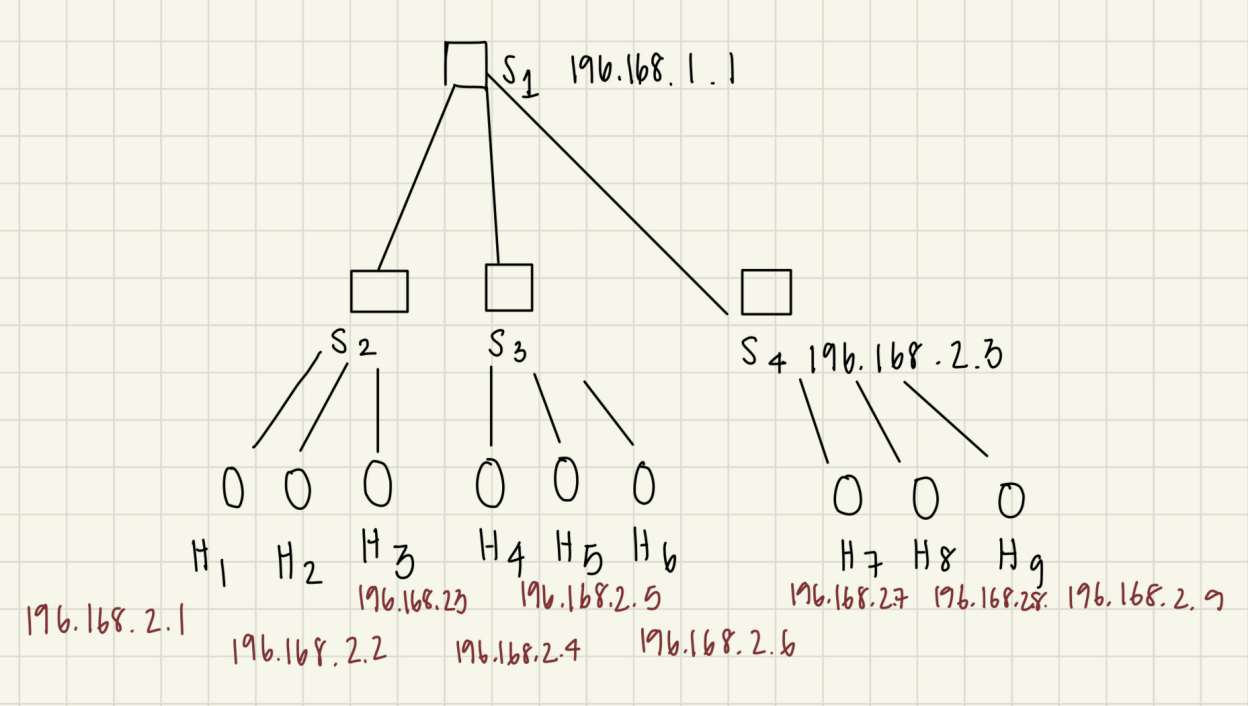
3. [4 pts] Default Topologies

a. What command invokes the controller to create a tree topology where the "root" switch is connected to 3 other switches and each of these switches isn connected to 3 unique hosts?

sudo mn --topo tree, depth=2,fanout=3

b. Using a drawing tool, draw an image of this topology, labeling the switches

and hosts with the IP addresses assigned to their interfaces.



Building a Firewall with the POX Controller:

A firewall is a wall you place in buildings to stop a fire from spreading. In the case of networking, it is the act of providing security by not letting specified traffic pass through the firewall. This feature is good for minimizing attack vectors and limiting the network "surface" exposed to attackers. Now we are going to learn a bit more about firewalls and how you can create your own firewall using OpenFlow-enabled switches. In this process you will learn how to customize your remote controller.

Provided for you: You are provided with:

- the Mininet configuration, prelab3.py, to set up a network which uses a remote controller
- a skeleton POX controller, prelab3controller.py.

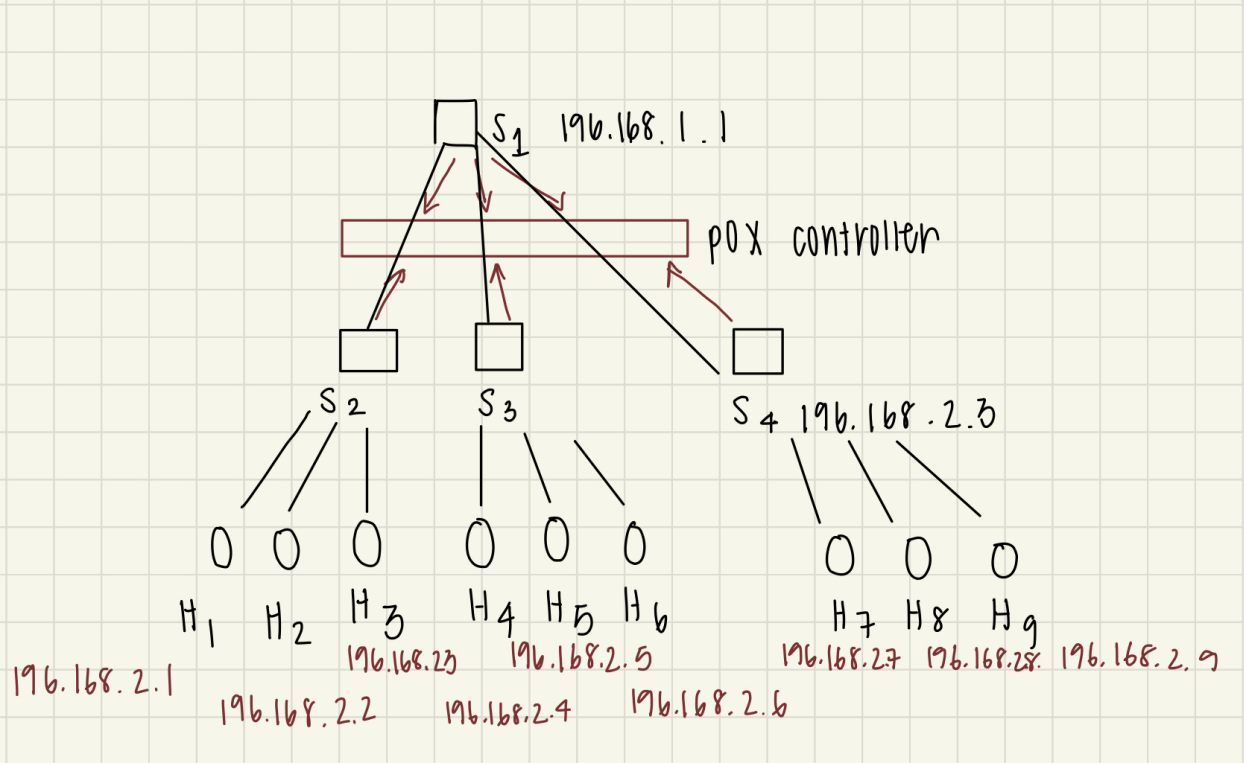
[25 pts] Understanding and Building Your Firewall:

4. [4 pts] POX Controller

a. Why are we using the POX controller (instead of the default controller) to implement the Firewall?

We are using the POX controller because it supports advanced SDN features and protocols - including OpenFlow. This means that there is more control over the network elements and communication between the controller and the switches.

b. Add the POX controller to your drawing in 3b using arrows to show the communication between the devices and the controller.



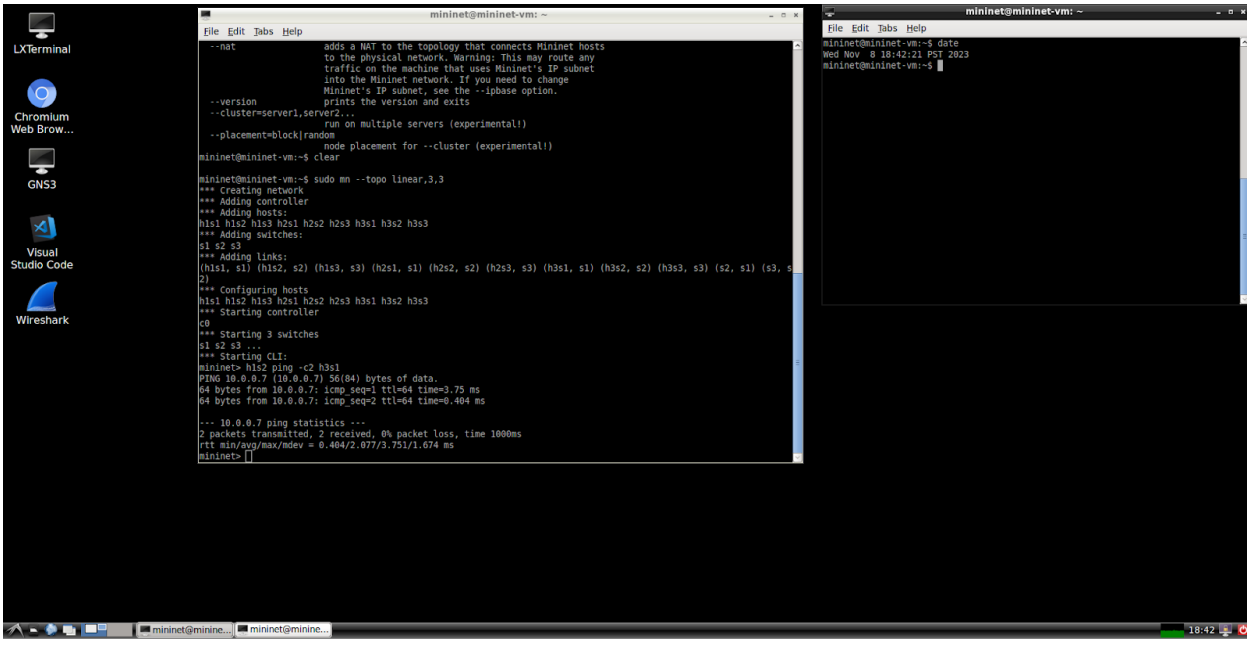
5. [4 pts] OpenFlow Messages

Work through the example here to see the OpenFlow messages generated by running Ping on a simple network. Start Mininet by running a simple default linear topology with 3 Switches that are connected to each other and are connected to 3 hosts each. Then, run Ping (with the count adjusted to 2) between h1s2 and h3s1. Capture the transfer in Wireshark and use a filter to display the OpenFlow messages.

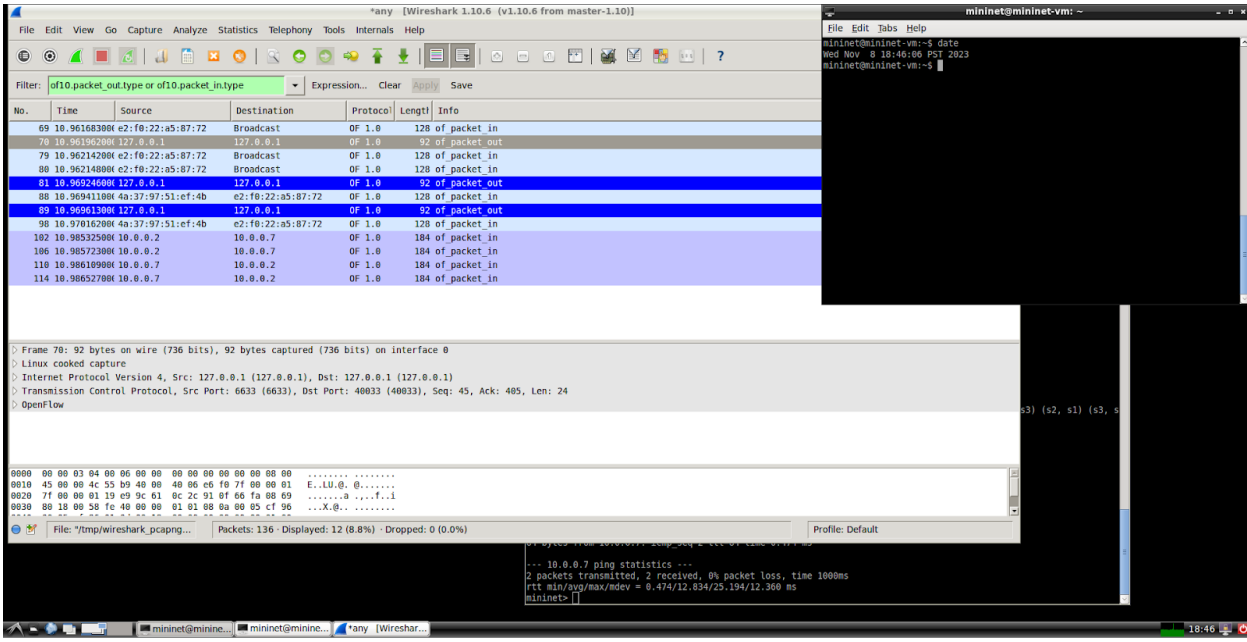
a) Show a screenshot of the terminal running your single command to invoke Mininet with the topology and the output of Ping.

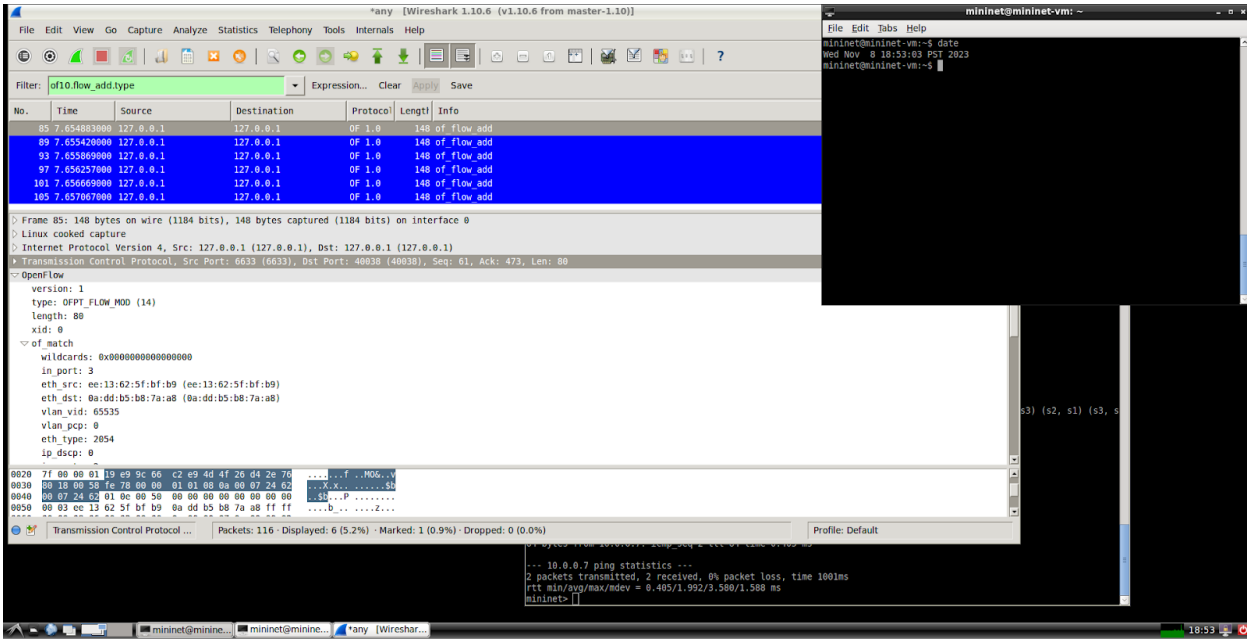
sudo mn --topo=linear,3,3

h1s2 ping -c2 h3s1



b) In another screenshot circle in red (if any) of the following packets that are displayed in Wireshark: Packet-In, Packet-Out, Flow-Mod and Flow-Expired





You can see in the frames above that there are some Flow-Mod packets above. There are no Flow-Expired packets likely because we have either rules with no time limit or havent hit the time limit yet.

c) Explain the purpose of each OpenFlow message that you found.

The flow_mod is a message that the controller uses to tell an OpenFlow enabled switch how to handle certain types of network traffic. The “mod” field of the message specified whether to add a new set of rules, chang existing rules or delete rules in the switches memory

6. [6 pts] Requirements, Guidance and Getting Started with the Firewall:

How to Get Started:

● Read documentation / resources as needed

● Study the file pox/pox/forwarding/l2_learning.py

Your Python Code: Time to get started! The firewall you build will be used by the small enterprise network shown below in Topology 1. To build the firewall you will modify the skeleton code in prelab3controller.py. The basic rules that you will need to implement in OpenFlow are given below in Table 1. Remember, this topology has already been created for you in prelab3.py and you do not need to do any modifications for Questions 6-8, but Question 10 requires you to modify prelab3.py. Before you begin coding, there are several important aspects of implementing the firewall to understand. We must understand the firewall rules, how the rules are "installed" in the switches, how the switches will forward packets that they receive and what are the underlying key function calls required to implement the firewall. Q6, 7 and 8 get you into position to begin coding.

● Rule installation: The firewall will use the POX controller. When you create a rule in the POX controller, you need to have POX "install" the rule in the switch so that the switch "remembers" what to do for 30 seconds. You will be downgraded if your switch simply asks the controller what to do for every packet it receives. Hint: To do this, look up ofp_flow_mod -> idle_timeout and hard_timeout in the POX WIKI.

● Packet Forwarding: In this assignment when the action received from the controller is to forward a packet, packet flooding is used by the switches.

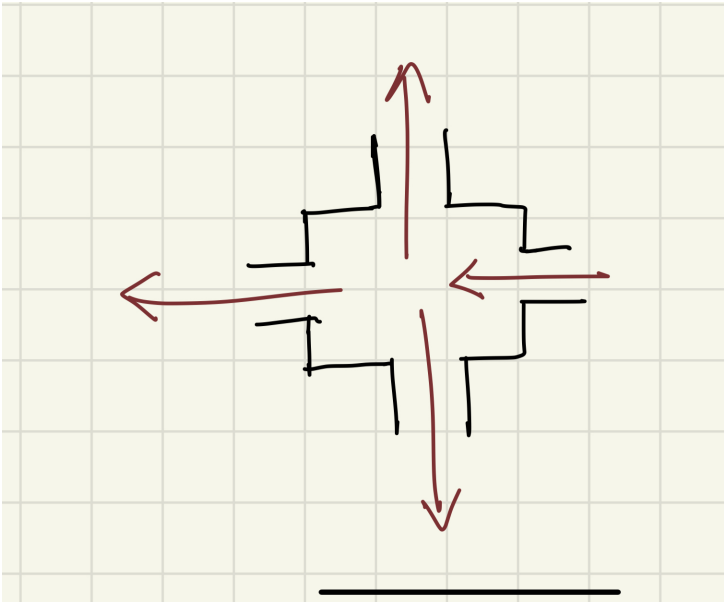
a) How will you "install" the rule in the switch for 30 seconds?

I would use the idle_timeout and the hard_timeout fields in the OpenFlow flow entry because these fields determine how long a rule should be in effect in the switch before it expires. Set some variable flow.entry = ofp_flow_mod(). Then I would set the flow.entrys's idle time and hard time out like so : flow.entry.idle.timeout = 30; flow.entry.hard.timeout = 30

b) What is packet flooding?

Packet flooding is when a switch forwards an incoming packet out of all its ports but the one that packet was received from.

c) Draw an image of a switch with 4 ports. Show a packet entering on one port and then show packet flooding.



7. [5 pts] Basic Firewall Rules (Table 1):

a. Briefly explain the meaning for Rules 1, 3, 4 and 9 from Table 1 below. A sentence or two for each rule is sufficient.

Rule 1 - This rule allows any source host regardless of IP to communicate with any destination host regardless of IP, using ARP.

Rule 3 - This rule allows any source host categorized as a workstation to communicate with any other host in the workstation category using the TCP protocol.

Rule 4 - This rule allows any devices within the “IT department workstation” category can establish a TCP connection with laptops labeled Laptops (1 and 2)

Rule 9 - This rule allows any workstation or laptop regardless of its IP to communicate with a designated DNS server using the UDP protocol.

b. Give an example of a packet transfer(using TCP protocol) originating from workstation3 in the current topology (Topology 1) to another host which would be dropped according to rule 10. Write the destination name along with its IP address.

Workstation3 (200.20.2.7/24) attempts to communicate with an external server, such as WebServer (200.20.2.2/24) because the destination is outside the local network - which is not allowed as it is not explicitly allowed per the table, it will be dropped

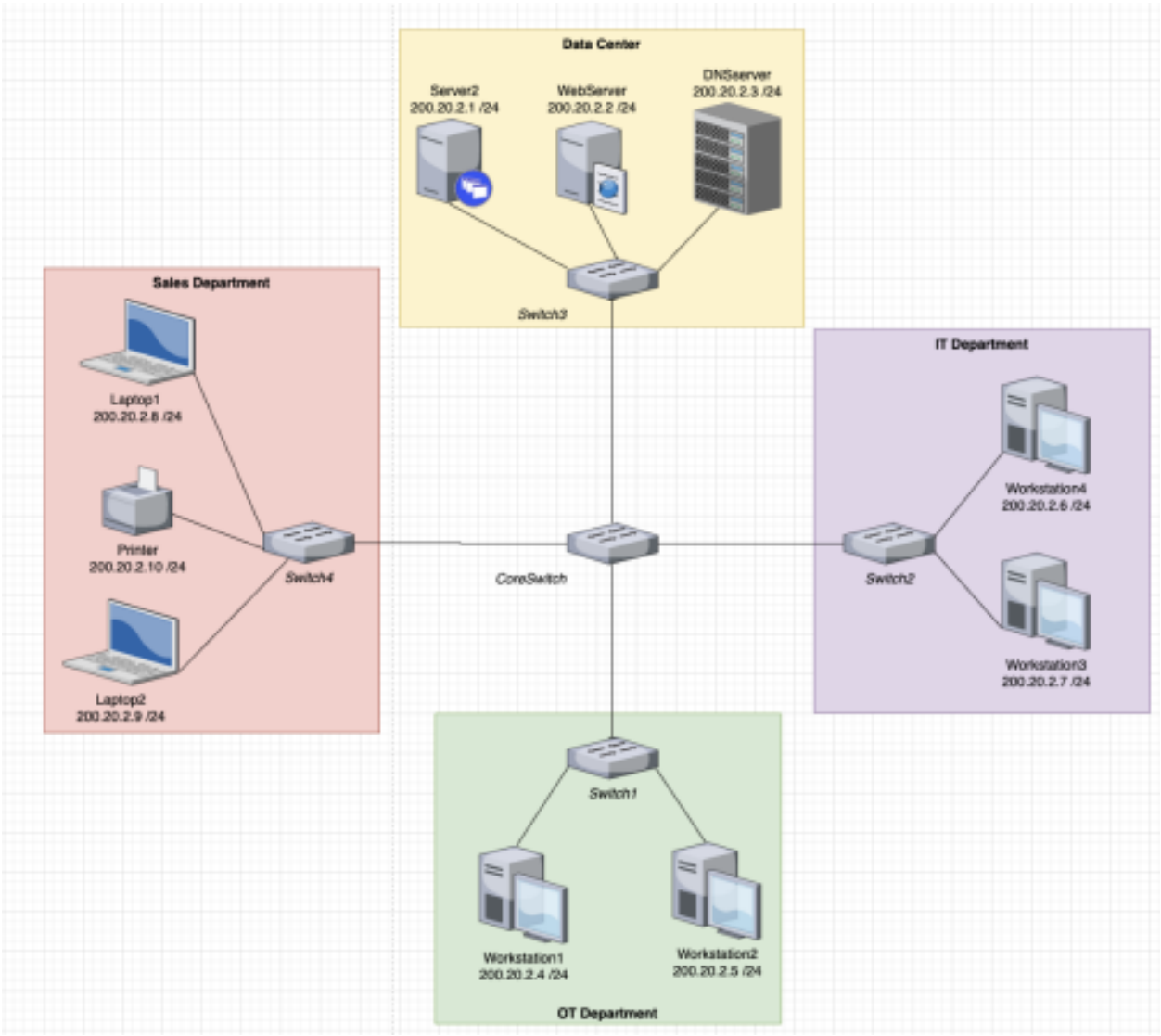
c. If you were to run ping between Workstation3 and Printer, do you expect it to work? Explain your answer based on the rules in Table 1.

It would likely work because the second rule specifies any ip address to any ip address to per the table the ping would work as it uses ICMP protocols.

d. If you were to run iperf between Laptop1 and WebServer, do you expect it to work? Explain your answer based on the rules in Table 1.

No it would not work because none of the rules on the table specify communication between the Laptop1 and Webserver using TCP protocol, and according to rule 10, any traffic not explicitly specified should be dropped so the iperf traffic between the 2 hosts would be dropped.

8. [6 pts] Write pseudocode for your Firewall using Table 1 below. (Use the host names instead of the IP addresses in your pseudocode. e.g., use Laptop1 rather than 200.20.2.9.) Check pox/pox/forwarding/l2_learning.py to identify key functions that might be useful in your firewall.



In your code use the naming convention below:

Device	**Required Naming Convention**
Workstations	ws1 ws2 ws3 ws4
Laptops	Laptop1 Laptop2
Servers Other	Serv2 ServDNS ServWeb Printer
Switches	s1 s2 s3 s4
CoreSwitch	s0

Note: Fill in the IP addresses for each of the hosts in Table 1. The first two rules MUST

be implemented FIRST.

Rule #	Src Host	Src Ip	Dst Host	Dst IP	Protocol	Action
1	Any	Any	Any	Any	ARP	accept
2	Any	Any IPv4	Any	IPv4	ICMP	accept
3	Any Workstation	200.20.2.4/24 200.20.2.5/24 200.20.2.6/24 200.20.2.7/24	Any Workstation	200.20.2.4/24 200.20.2.5/24 200.20.2.6/24 200.20.2.7/24	TCP	accept
4	IP Department Workstations	200.20.2.6/24 200.20.2.7/24	Laptops (1 and 2)	200.20.2.8/24 200.20.2.9/24	TCP	accept
5	Laptops (1 and 2)	200.20.2.8/24 200.20.2.9/24	IT Department Workstations	200.20.2.6/24 200.20.2.7/24	TCP	accept
6	Laptops (1 and 2)	200.20.2.8/24 200.20.2.9/24	Server2	200.20.2.1/24	TCP	accept
7	Server2	200.20.2.1/24	Laptops (1 and 2)	200.20.2.8/24 200.20.2.9/24	TCP	accept
8	Any Workstation or Laptop	200.20.2.4/24 200.20.2.5/24 200.20.2.6/24 200.20.2.7/24 200.20.2.8	DNS server	200.20.2.3/24	UDP	accept

		/24 200.20.2.9 /24				
9	DNSserver	200.20.2.3 /24	Any Workstation or Laptop	200.20.2.4 /24 200.20.2.5 /24 200.20.2.6 /24 200.20.2.7 /24 200.20.2.8 /24 200.20.2.9 /24	UDP	accept
10		Any		Any	-	drop

Table 1- Basic Firewall

PseudoCode:

```
#if the packet has an ipv4
  #assign source and destination ip

# If the packet is ARP protocol
      #accept
# If the packet is ICMP protocol
      #accept
# If the packet is TCP protocol
    # Check if rule 3 will work
    #src ip must be found in the workstation list- and dst ip must be found in the workstation list
    # Else check if rule 4 will work
    #src host ip must be found in the ITworkstations list and the dst ip must be found in the laptops list
    # Else check if rule 5 will work
    # src host ip must be found in the laptops list and the dst ip must be found in the ITworkstations list
    # Else check if rule 6 will work
    # src host ip must be found in the laptops list and the dst ip must be found in list server2
    # Else check if rule 7 will work
```

src host ip must be found in the server2 list and the dst ip must be found in the DNSserver list

If the packet is UDP protocol

Else check of rule 8 will work

src host ip must be found in the workstationsORlaptops list and the dst ip must be found in the DNSserver list

Else check if rule 9 will work

src host ip must be found in the DNSserver list and the dst host must be found in the workstationsORlaptops list

[43 pts] Running the POX Controller (your code): Important: Make sure you have a backup of your code!! (save a backup frequently)
When your code is complete, put your firewall code (in prelab3controller.py) into the ~/pox/pox/misc directory.
To run this assignment, both files must be running at the same time.
In 2 different terminal windows:
● first start the controller file

● then run the mininet file. (It should make sense to you why the order matters.)
Here are the commands:
Launch the controller with the command:
sudo ~/pox/pox.py misc.prelab3controller
To run the mininet file, place it in ~ and run the command sudo python ~/prelab3.py

9. [20 pts] Code complete - Testing the Basic Firewall

a. Pingall:

● Run the pingall command. Which of the firewall rules contributed to the result? Explain you answer(s).

● Include a timestamped screenshot of the results.

The command I ran was pingall, it worked and all my hosts pinged, however I ran this on the Lab computers and forgot to include this screen shot :(so sorry.

b. iperf:

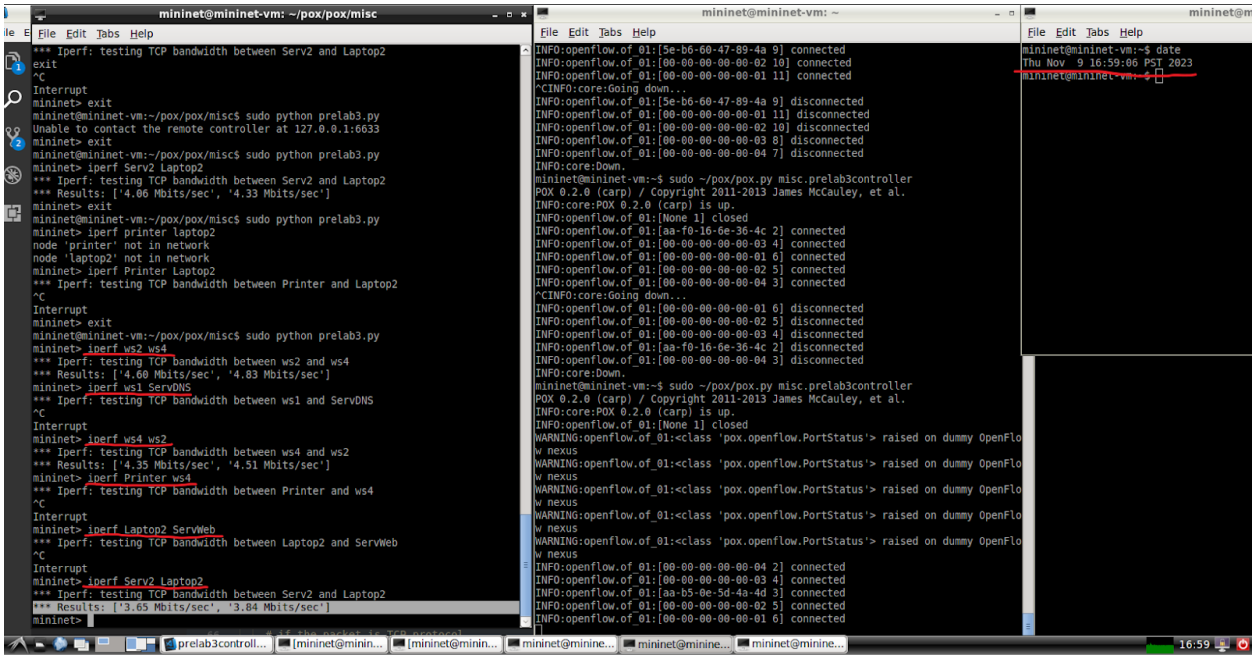
Let's test the TCP connection of your basic firewall. For this, use the iperf command (see man page) to simulate TCP traffic and fill out Table 2 (the first two are given as examples).

Are your results what you expect? Why or why not?

Include a timestamped screenshot and explain your answers by marking up the screenshot in a very visible color.

My results were exactly as I expected based on the firewall rules specified in Table 1. Link 1, passed because of rule 3 - TCP from any workstation to

any workstation. Link 2, failed because of rule 8 - UDP from any workstation or Laptop to DNS Server. Link 3, passed because of rule 3 - TCP from any workstation to any workstation. Link 4, failed because of rule 10 - drop any traffic not specified on the table. Link 5, failed because of rule 6 - TCP from Laptops to Server2. Link 6, passed because of rule 6 - TCP from Laptops to Server2.



#	Link	Iperf Command	Pass/Fail
1	Workstation 2 - Workstation 4	iperf ws2 ws4	pass
2	Workstation1 - DNSserver	iperf ws1 ServDNS	fail
3	Workstation4 - Workstation2	iperf ws4 ws2	pass
4	Printer - Workstation4	iperf Printer ws4	fail
5	Laptop2 - WebServer	iperf Laptop2 ServWeb	fail
6	Server2 - Laptop2	iperf Serv2 Laptop2	pass

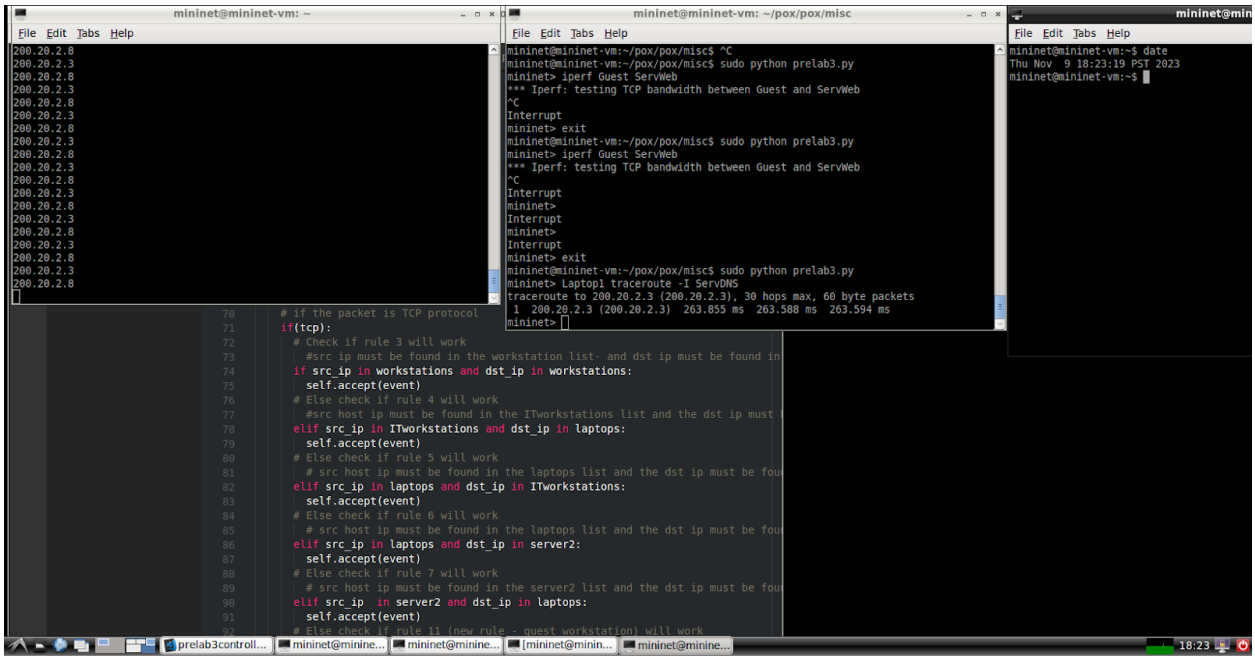
Table 2 - Iperf testing table

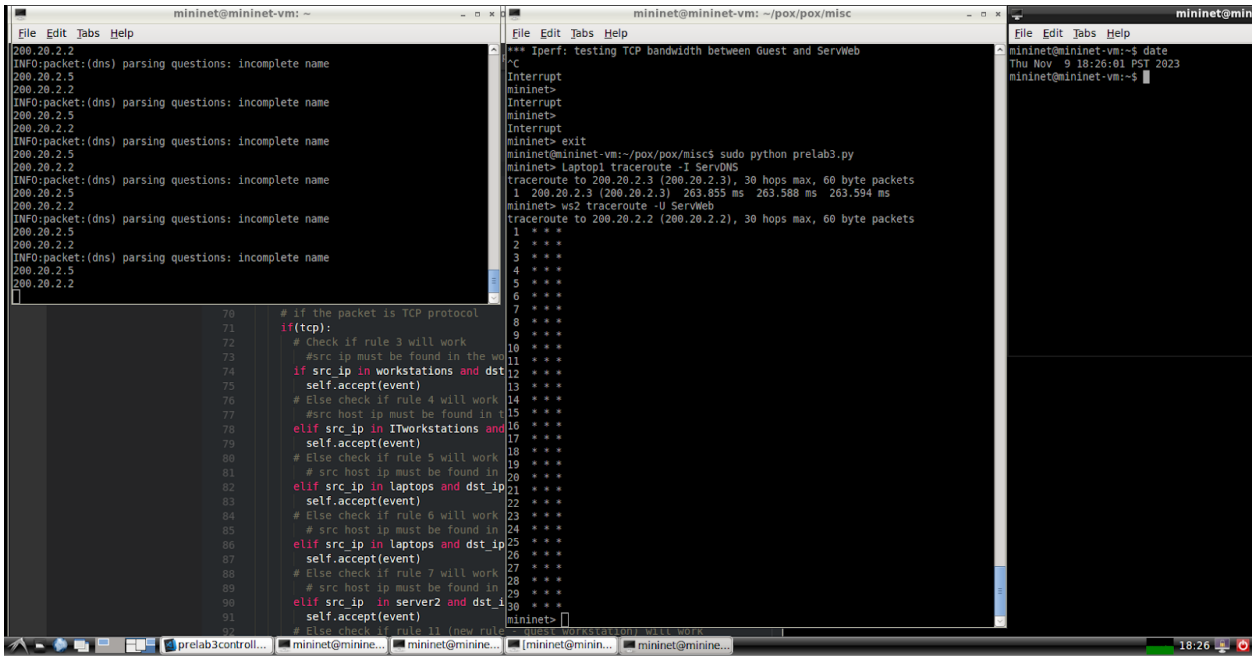
c. Traceroute:
Stop the controller and exit the mininet prompt. Install Traceroute in mininet using ‘sudo apt-get install traceroute’. Then, restart the controller and run the prelab3.py.

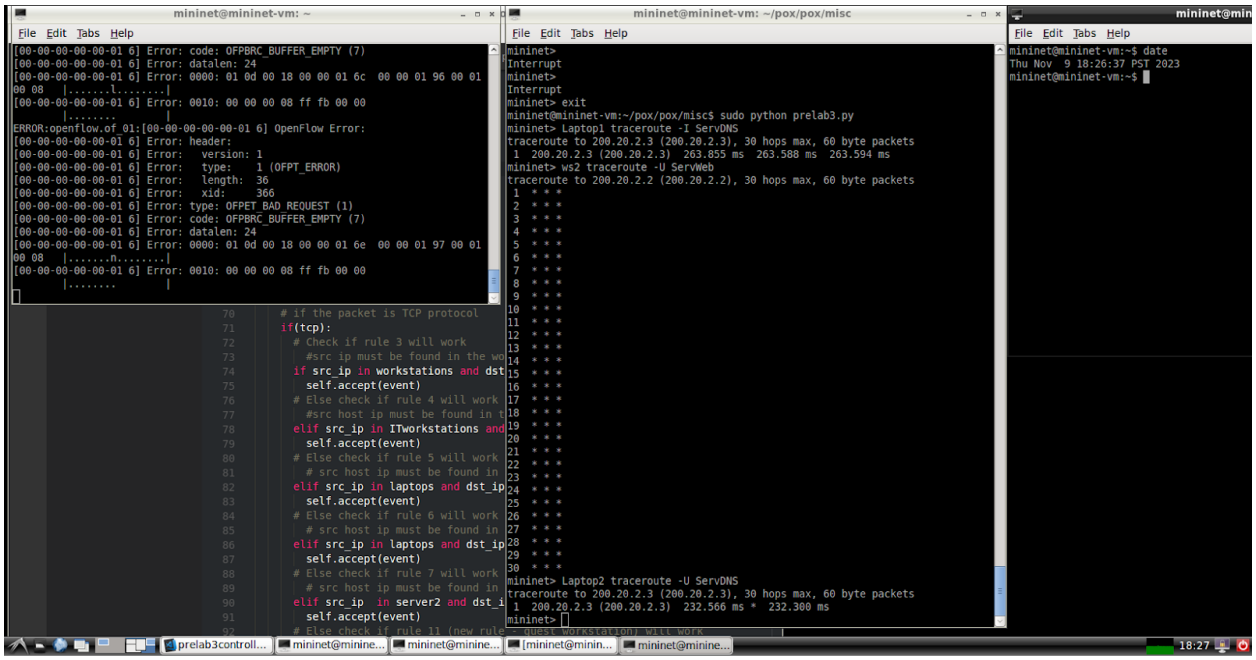
Let’s test ICMP and UDP connections of your firewall using Traceroute commands. Ensure that you force Traceroute probes with the protocol specified in column 1 from the below table. Based on the rules in Table 1, is the output what you expected? Explain. Include a timestamped screenshot showing the results. (potentially 4 separate screenshots, presented in order)

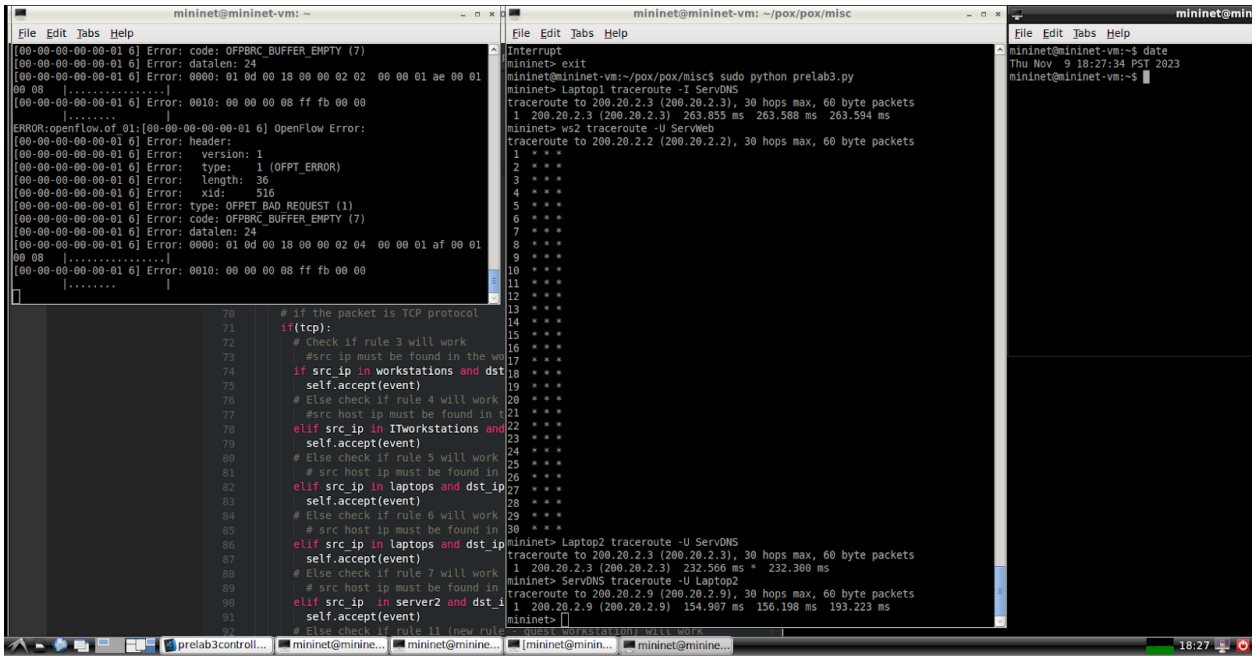
> I expected all the results I got! Based on Table 1, the firewall rules, the results are consistent with what traffic is allowed or blocked. Link 1, passes due to rule 2 - which allows TCP from any to any. Link 2 fails because of rule 6 - TCP from Laptops to server2. Link 3 passes because of rule 8 - UDP from any workstation to Laptop or DNSserver. Link 4 passes because of rule 9 - UDP from DNS server to Any Workstation or Laptop.

Protocol	Link	Traceroute Command	Pass/Fail
ICMP	Laptop1 - DNSserver	Laptop1 traceroute -I ServDNS	Pass
UDP	Workstation - WebServer	ws2 traceroute -U ServWeb	Fail
UDP	Laptop2 - DNSserver	Laptop2 traceroute -U ServDNS	Pass
UDP	DNSserver - Laptop2	ServDNS traceroute -U Laptop2	Pass


mininet@mininet-vm: ~
File Edit Tabs Help
200.20.2.8
200.20.2.3
200.20.2.8
200.20.2.3
200.20.2.8
200.20.2.3
200.20.2.8
200.20.2.3
200.20.2.8
200.20.2.3
200.20.2.8
200.20.2.3
200.20.2.8
200.20.2.3
200.20.2.8
200.20.2.3
200.20.2.8
mininet@mininet-vm: ~/pox/pox/misc
File Edit Tabs Help
mininet@mininet-vm:~/pox/pox/misc$ ^C
mininet@mininet-vm:~/pox/pox/misc$ sudo python prelab3.py
mininet> iperf Guest ServWeb
*** Iperf: testing TCP bandwidth between Guest and ServWeb
^C
Interrupt
mininet> exit
mininet@mininet-vm:~/pox/pox/misc$ sudo python prelab3.py
mininet> iperf Guest ServWeb
*** Iperf: testing TCP bandwidth between Guest and ServWeb
^C
Interrupt
mininet>
Interrupt
mininet>
Interrupt
mininet> exit
mininet@mininet-vm:~/pox/pox/misc$ sudo python prelab3.py
mininet> Laptop1 traceroute -I ServDNS
traceroute to 200.20.2.3 (200.20.2.3), 30 hops max, 60 byte packets
1 200.20.2.3 (200.20.2.3) 263.855 ms 263.588 ms 263.594 ms
mininet>
mininet@min
File Edit Tabs Help
mininet@mininet-vm:~$ date
Thu Nov 9 18:23:19 PST 2023
mininet@mininet-vm:~$
70 # if the packet is TCP protocol
71 if(tcp):
72 # Check if rule 3 will work
73 #src ip must be found in the workstation list- and dst ip must be found in
74 if src_ip in workstations and dst_ip in workstations:
75 self.accept(event)
76 # Else check if rule 4 will work
77 #src host ip must be found in the ITworkstations list and the dst ip must
78 elif src_ip in ITworkstations and dst_ip in laptops:
79 self.accept(event)
80 # Else check if rule 5 will work
81 # src host ip must be found in the laptops list and the dst ip must be fou
82 elif src_ip in laptops and dst_ip in ITworkstations:
83 self.accept(event)
84 # Else check if rule 6 will work
85 # src host ip must be found in the laptops list and the dst ip must be fou
86 elif src_ip in laptops and dst_ip in server2:
87 self.accept(event)
88 # Else check if rule 7 will work
89 # src host ip must be found in the server2 list and the dst ip must be fou
90 elif src_ip in server2 and dst_ip in laptops:
91 self.accept(event)
92 # Else check if rule 11 (new rule - guest workstation) will work
prelab3controll...
mininet@minine...
mininet@minine...
[mininet@minin...
mininet@minine...
18:23


mininet@mininet-vm: ~
File Edit Tabs Help
200.20.2.2
INFO:packet:(dns) parsing questions: incomplete name
200.20.2.5
200.20.2.2
INFO:packet:(dns) parsing questions: incomplete name
200.20.2.5
200.20.2.2
INFO:packet:(dns) parsing questions: incomplete name
200.20.2.5
200.20.2.2
INFO:packet:(dns) parsing questions: incomplete name
200.20.2.5
200.20.2.2
INFO:packet:(dns) parsing questions: incomplete name
200.20.2.5
200.20.2.2
INFO:packet:(dns) parsing questions: incomplete name
200.20.2.5
200.20.2.2
mininet@mininet-vm: ~/pox/pox/misc
File Edit Tabs Help
*** Iperf: testing TCP bandwidth between Guest and ServWeb
^C
Interrupt
mininet>
Interrupt
mininet>
Interrupt
mininet> exit
mininet@mininet-vm:~/pox/pox/misc$ sudo python prelab3.py
mininet> Laptop1 traceroute -I ServDNS
traceroute to 200.20.2.3 (200.20.2.3), 30 hops max, 60 byte packets
1 200.20.2.3 (200.20.2.3) 263.855 ms 263.588 ms 263.594 ms
mininet> ws2 traceroute -U ServWeb
traceroute to 200.20.2.2 (200.20.2.2), 30 hops max, 60 byte packets
1 * * *
2 * * *
3 * * *
4 * * *
5 * * *
6 * * *
7 * * *
8 * * *
9 * * *
10 * * *
11 * * *
12 * * *
13 * * *
14 * * *
15 * * *
16 * * *
17 * * *
18 * * *
19 * * *
20 * * *
21 * * *
22 * * *
23 * * *
24 * * *
25 * * *
26 * * *
27 * * *
28 * * *
29 * * *
30 * * *
mininet>
mininet@min
File Edit Tabs Help
mininet@mininet-vm:~$ date
Thu Nov 9 18:26:01 PST 2023
mininet@mininet-vm:~$
70 # if the packet is TCP protocol
71 if(tcp):
72 # Check if rule 3 will work
73 #src ip must be found in the wo
74 if src_ip in workstations and dst
75 self.accept(event)
76 # Else check if rule 4 will work
77 #src host ip must be found in t
78 elif src_ip in ITworkstations and
79 self.accept(event)
80 # Else check if rule 5 will work
81 # src host ip must be found in
82 elif src_ip in laptops and dst_ip
83 self.accept(event)
84 # Else check if rule 6 will work
85 # src host ip must be found in
86 elif src_ip in laptops and dst_ip
87 self.accept(event)
88 # Else check if rule 7 will work
89 # src host ip must be found in
90 elif src_ip in server2 and dst_i
91 self.accept(event)
92 # Else check if rule 11 (new rule - guest workstation) will work
prelab3controll...
mininet@minine...
mininet@minine...
[mininet@minin...
mininet@minine...
18:26





10. [23 pts] Adding your own Firewall Rules: DNS Server, Guest Access and Network Printer

Note : For questions requiring you to state the firewall rule that has been implemented, use the
following format:
SRC IP DEST IP PROTOCOL ACCEPT/DROP

For example, if packets sent from host A with ip address a.a.a.a, to host B with ip address b.b.b.b using TCP protocol have to be accepted , then the corresponding format for the firewall rule should be as follows: a.a.a.a b.b.b.b TCP ACCEPT

A. Ping Flood

Ping flood is a common Denial of Service (DoS) attack in which an attacker takes down a DNS server by overwhelming it with Ping requests. To protect your DNS server from the Ping flood attack, add a Firewall rule for the DNS Server.

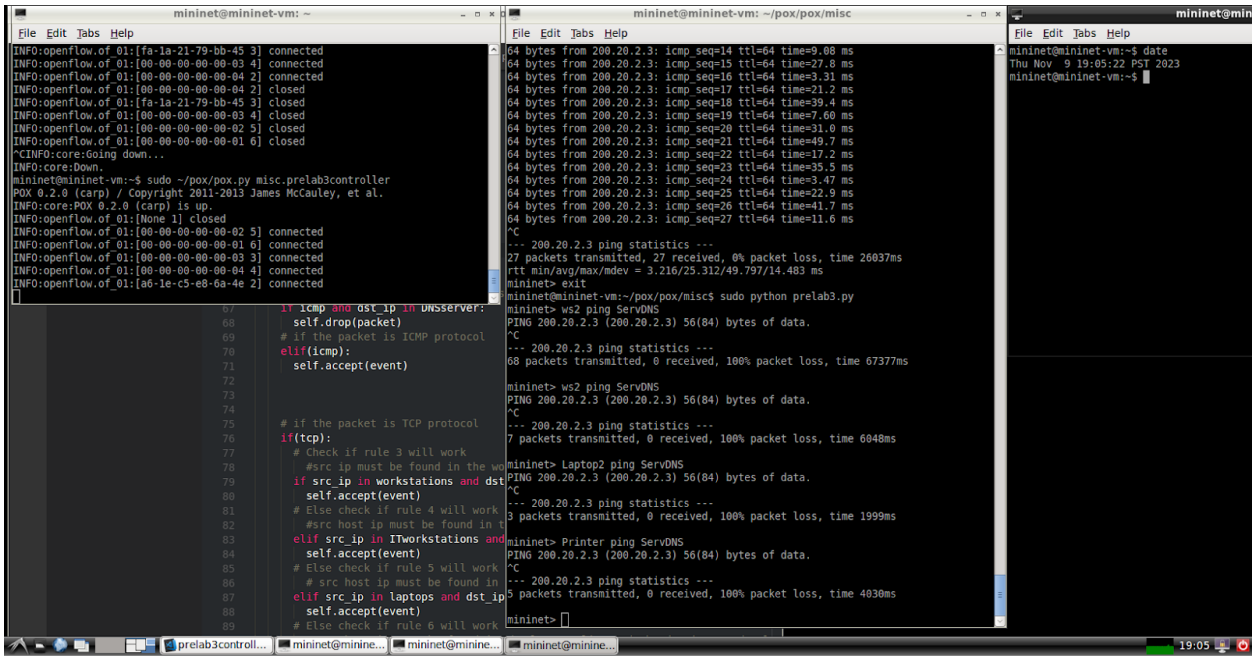
- State the rules you have implemented and explain your reasoning
- Write down the mininet command to test the firewall logic implemented in (i).
- Run your test(s) and verify with screenshots and explanation that your rules are working.

ws2 ping ServDNS

Laptop2 ping ServDNS

Printer ping ServDNS

The “rule” I implemented was to drop any traffic whose destination Ip address was that of the DNS server. These requests were filtered with my code to also only drop if sent with an ICMP, as ping uses ICMP. The below screenshots show that any device pinging to the DNSserver has no way of getting traffic through as no connection gets made.



B. Guest Access

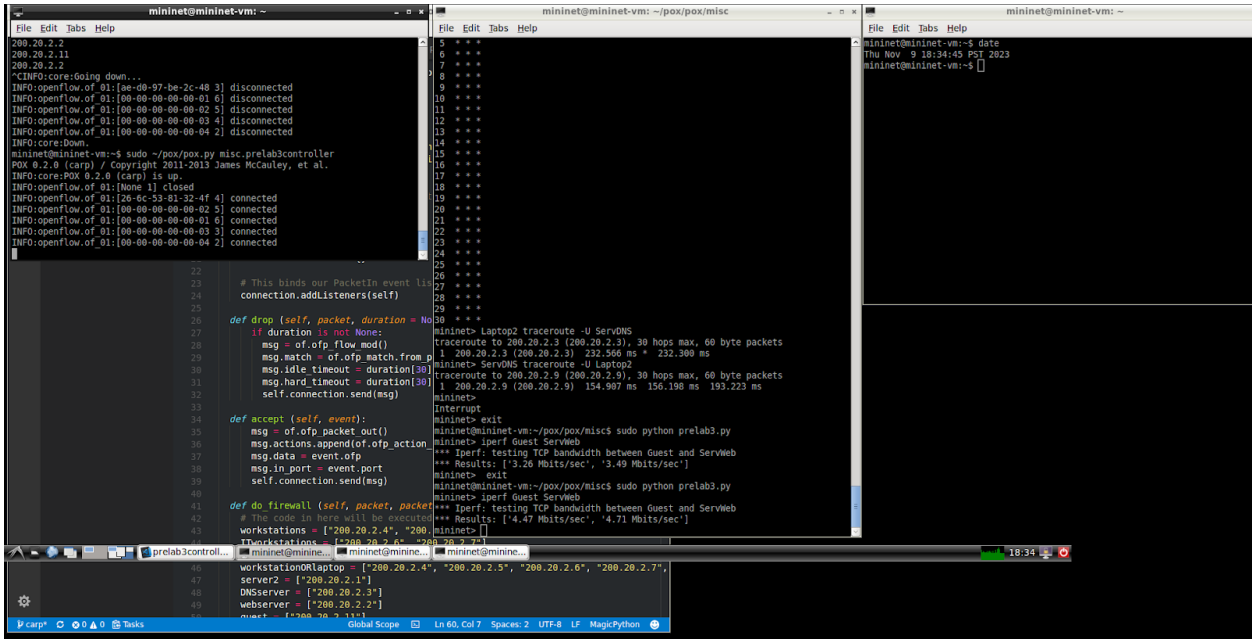
The company has a guest network through which guests in the network can access the WebServer. Change your topology to add a new Guest Workstation to the IT

department and assign it the IP address 200.20.2.11/24. Add Firewall rules for guests to access the Web Server to make HTTP requests.

- State the rules you have implemented and explain your reasoning
- Write down the mininet command to test the firewall logic implemented in (i)
- Run your test(s) and verify with screenshots and explanation that your rules are working.

iperf Guest ServWeb

Here I had to add a host within the topology code, called “guest workstation” and link it to switch 2 as that is the switch that the IT Department workstations are connected to. After that within my controller code, I had to add an additional If statement (rule 11 within my code comments) under the TCP statements that accepted traffic if the source IP address was a guest workstation and the destination IP address was a WebServer IP. I also had to add a secondary if statement (rule 12 within my code comments) so that the web server could let traffic flow to the guest workstation. In the screenshot below you can see that the connection is established and there was traffic between guest and the webserver.



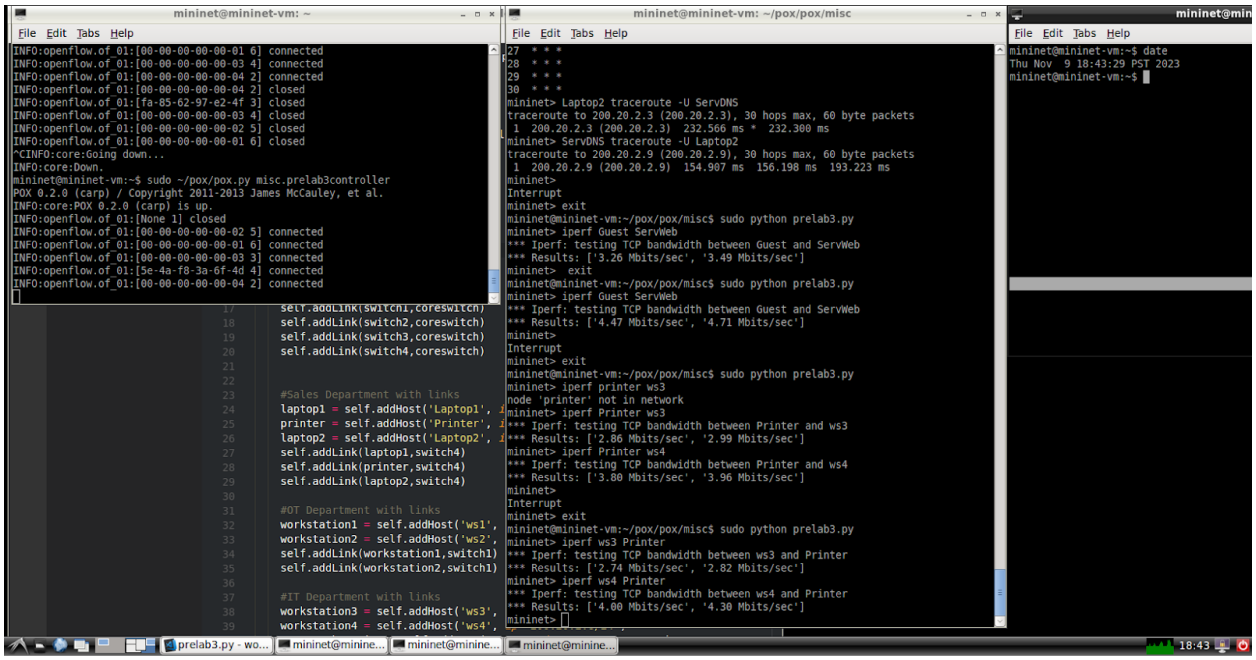
C. The Network Printer

In the Sales department the network printer uses a proprietary printing protocol for printing purposes that runs on top of TCP. Add Firewall rules for opening up the Printer to any workstation in the IT department.

- State the rules you have implemented and explain your reasoning
- How can it be tested?
- Run your test(s) and verify with screenshots and explanation that your rules are working.

Test by running: iperf ws3 Printer or iperf ws4 Printer

I implemented another TCP rule, under the TCP if statements that allowed the printer to connect to the IT workstations and the IT workstations to connect to the printer (rules 13 and 14 within my code comments). The screenshots below show that there is traffic between ws3 and the printer and ws4 and the printer.



D. Add your new rules to Table 1 and show the “new” Table 1. (Note that the order of the rules matter)

Rule #	Src Host	Src Ip	Dst Host	Dst IP	Protocol	Action
1	Any	Any	Any	Any	ARP	accept
2	Any	Any IPv4	Any	IPv4	ICMP	accept

3	Any Workstation	200.20.2.4/24 200.20.2.5/24 200.20.2.6/24 200.20.2.7/24	Any Workstation	200.20.2.4/24 200.20.2.5/24 200.20.2.6/24 200.20.2.7/24	TCP	accept
4	IP Department Workstations	200.20.2.6/24 200.20.2.7/24	Laptops (1 and 2)	200.20.2.8/24 200.20.2.9/24	TCP	accept
5	Laptops (1 and 2)	200.20.2.8/24 200.20.2.9/24	IT Department Workstations	200.20.2.6/24 200.20.2.7/24	TCP	accept
6	Laptops (1 and 2)	200.20.2.8/24 200.20.2.9/24	Server2	200.20.2.1/24	TCP	accept
7	Server2	200.20.2.1/24	Laptops (1 and 2)	200.20.2.8/24 200.20.2.9/24	TCP	accept
8	Any Workstation or Laptop	200.20.2.4/24 200.20.2.5/24 200.20.2.6/24 200.20.2.7/24 200.20.2.8/24 200.20.2.9/24	DNS server	200.20.2.3/24	UDP	accept
9	DNSserver	200.20.2.3/24	Any Workstation or	200.20.2.4/24 200.20.2.5	UDP	accept

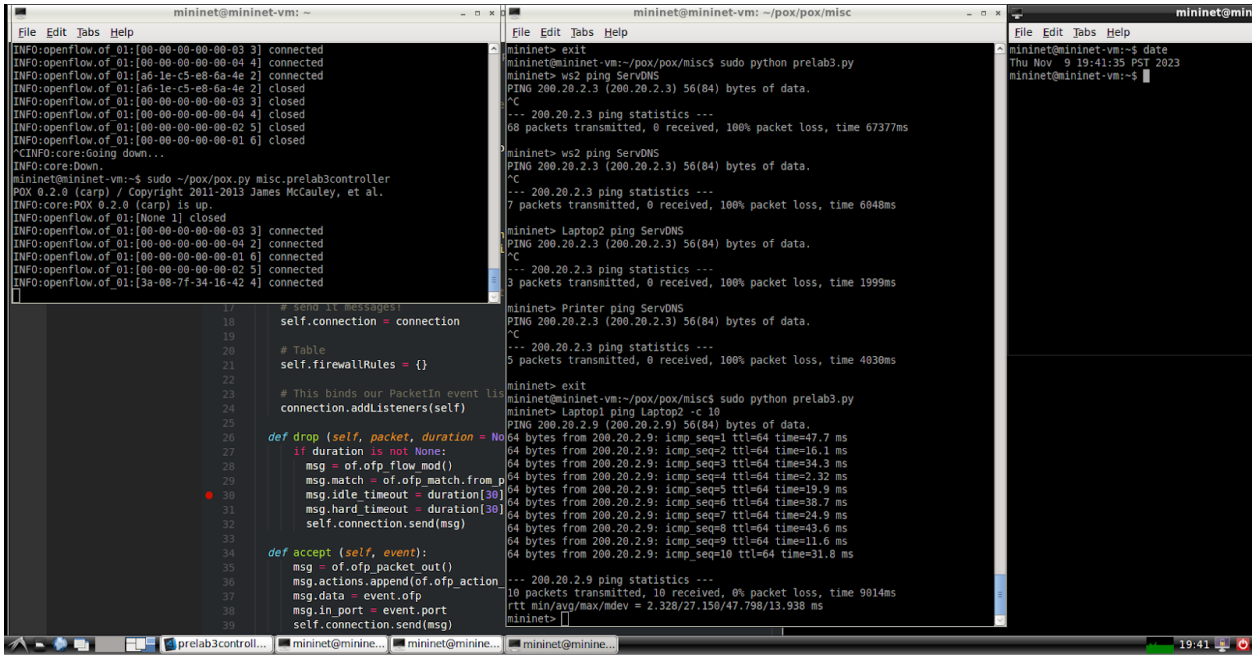
			Laptop	/24 200.20.2.6 /24 200.20.2.7 /24 200.20.2.8 /24 200.20.2.9 /24		
10	guest	200.20.2.11	WebServer	200.20.2.2	TCP	accept
11	WebServer	200.20.2.2	guest	200.20.2.11	TCP	accept
12	IT department Workstations	200.20.2.6 /24 200.20.2.7 /24	Printer	200.20.2.10	TCP	accept
13	Printer	200.20.2.10	IT Department Workstations	200.20.2.6 /24 200.20.2.7 /24	TCP	accept
14		Any		Any	-	drop

[14 pts] Testing the Final Completed Firewall: Let's check out your full Firewall!

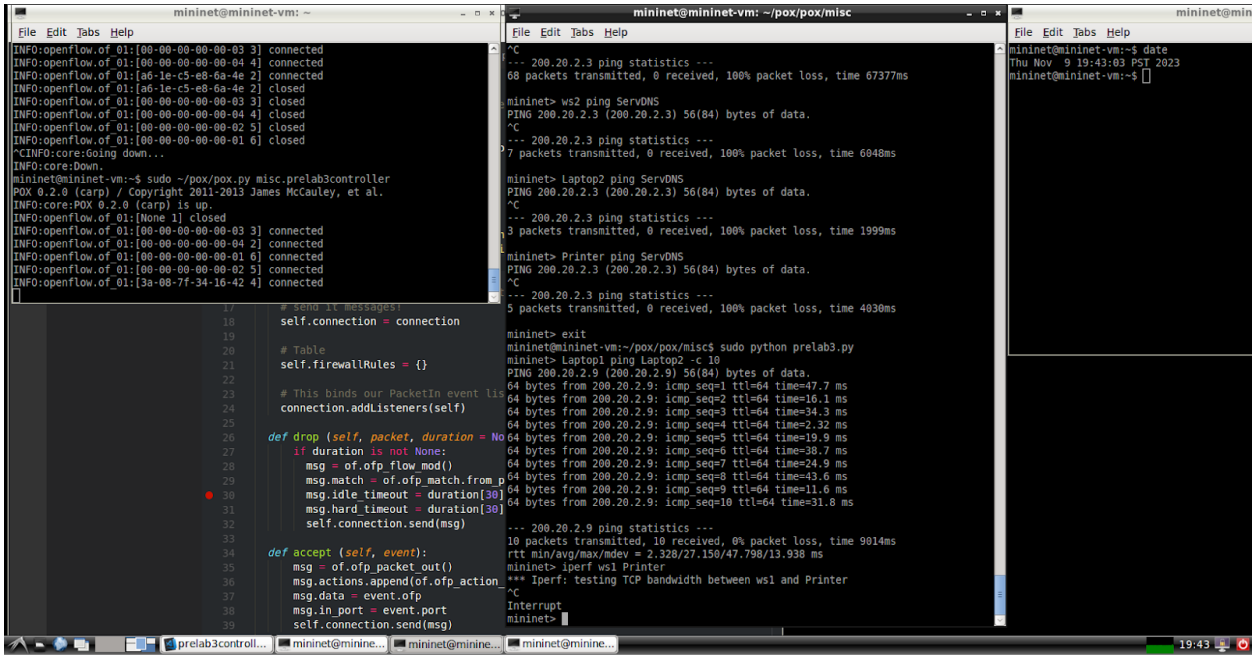
11. [8 pts] Random Commands

Perform the following commands in mininet and take a timestamped screenshot of the results. Highlight the section of the output which indicates the result of the command, was the command successful or not? Explain why it succeeded or failed (Hint: Think about the rules you have implemented).

a . Laptop1 ping Laptop2 -c 10

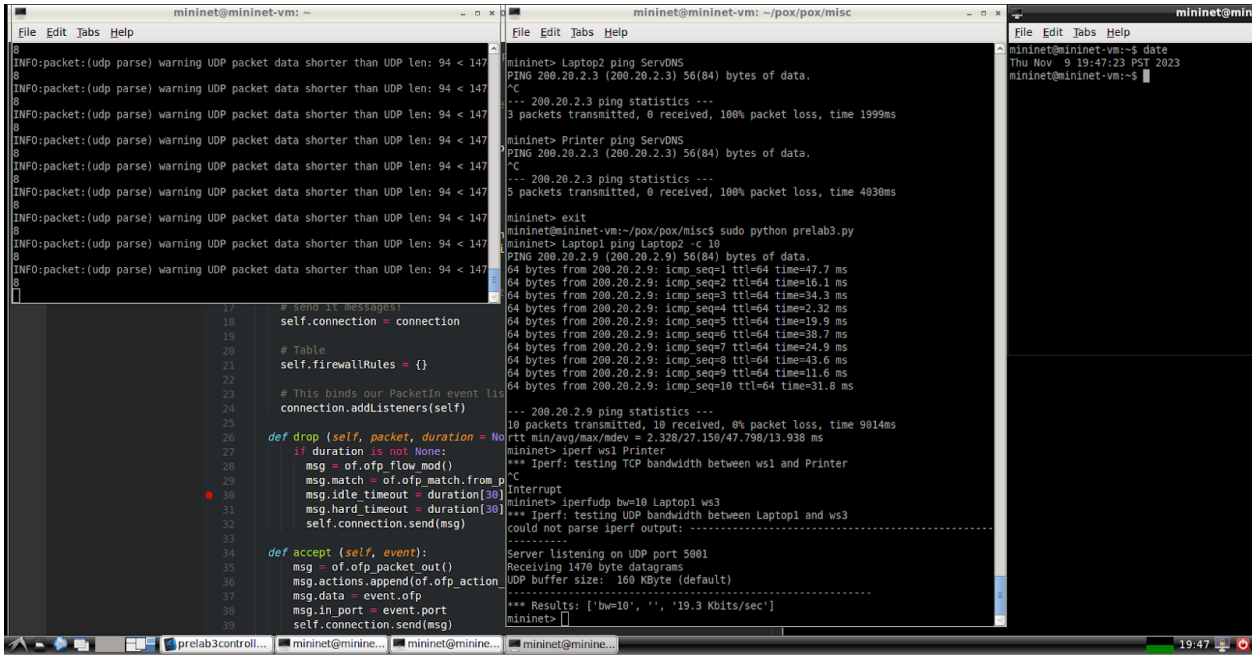


b. iperf ws1 Printer

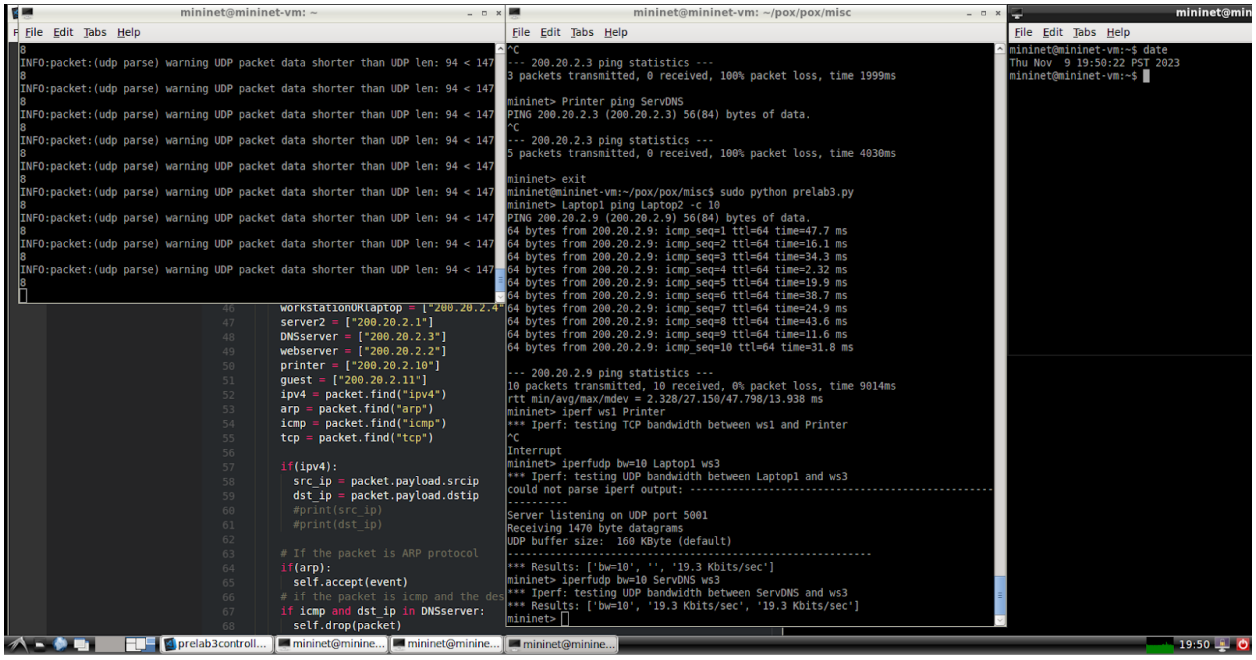


Here the connection is being dropped because of the rules of the firewall. Workstation 1 is within the OT Department which means that the firewall is working perfectly and that the traffic should not be allowed as per Rule 14 on the updated table

c. iperfudp bw=10 Laptop1 ws3



d. iperfudp bw=10 ServDNS ws3



12. [6 pts] If you wanted to run Traceroute between Workstation1 and Server2, would your firewall currently support that operation? Why or why not? State any assumptions you are making, such as the protocol(s) used by Traceroute. Explain your answer.

Yes it would run. According to rule 2 on the updated table , running the command “ws1 traceroute -I Serv2” would work because both Workstation1 and Server2 have IPv4 Ip address and the -I will involve the ICMP protocol. It's important to note that traceroute doesn't default to the ICMP protocol, which is why it is

necessary for the command to be run with -I after traceroute to specify ICMP protocols.

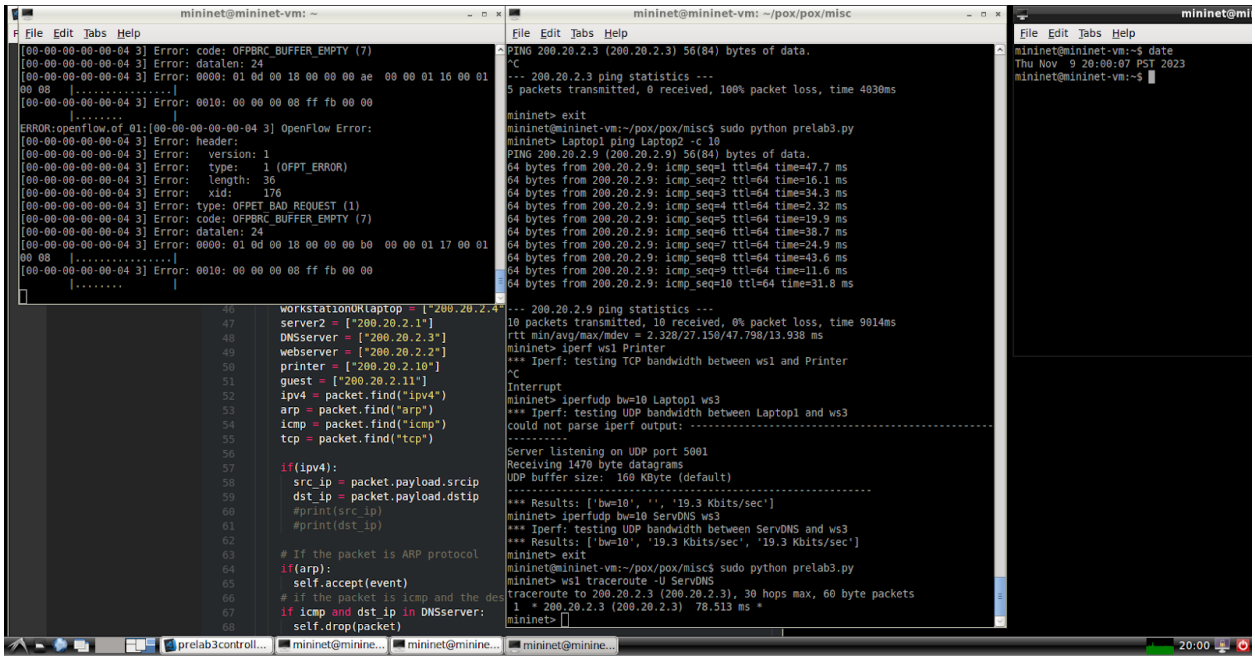
[5 pts] EXTRA CREDIT

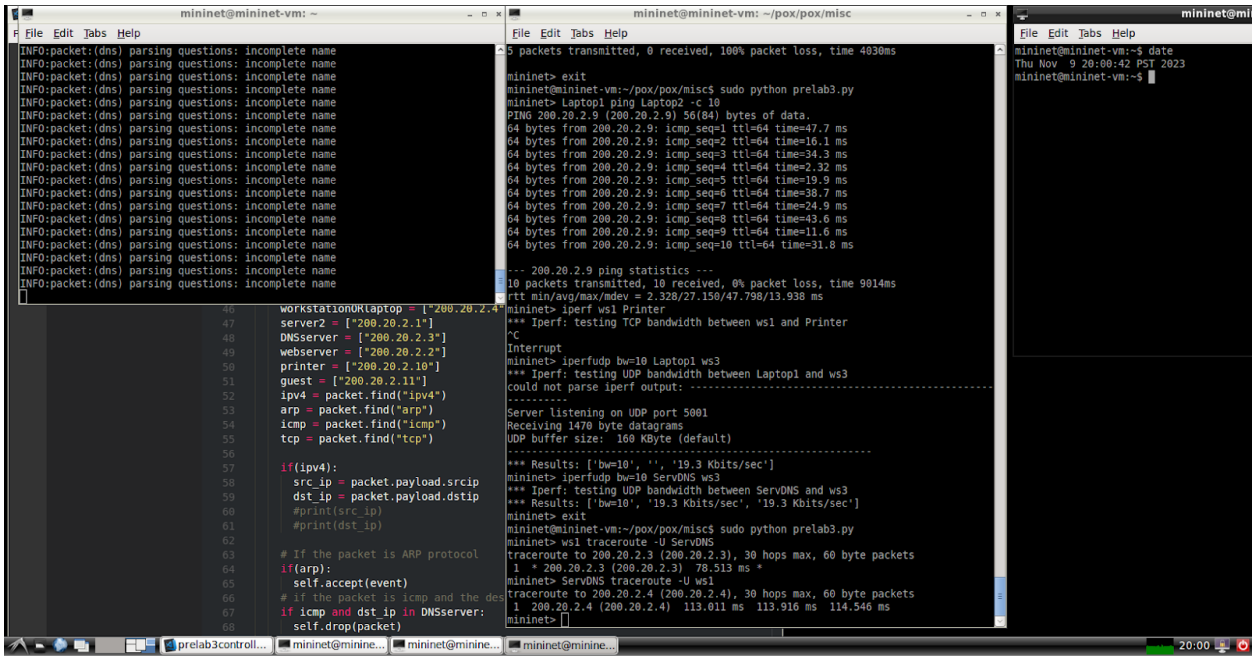
● Run traceroute, forcing UDP probes, between Workstation1 and DNSserver.

● Now, run traceroute, forcing UDP probes, between DNSserver and Workstation1.

Rules regarding UDP (originally rules 8 and 9, but may have changed since adding rules) say these
both should work. In fact, they did work previously when you tested them in 9 c).

Which traceroute command doesn't work now? Explain why it no longer works and explain using a timestamped screenshot with both commands.


mininet@mininet-vm: ~
File Edit Tabs Help
[00-00-00-00-00-04 3] Error: code: OFPBRC_BUFFER_EMPTY (7)
[00-00-00-00-00-04 3] Error: datalen: 24
[00-00-00-00-00-04 3] Error: 0000: 01 0d 00 18 00 00 00 ae 00 00 01 16 00 01 00 00 |................|
[00-00-00-00-00-04 3] Error: 0010: 00 00 00 08 ff fb 00 00 |........|
ERROR:openflow.of_01:[00-00-00-00-00-04 3] OpenFlow Error:
[00-00-00-00-00-04 3] Error: header:
[00-00-00-00-00-04 3] Error: version: 1
[00-00-00-00-00-04 3] Error: type: 1 (OFPT_ERROR)
[00-00-00-00-00-04 3] Error: length: 36
[00-00-00-00-00-04 3] Error: xid: 176
[00-00-00-00-00-04 3] Error: type: OFPET_BAD_REQUEST (1)
[00-00-00-00-00-04 3] Error: code: OFPBRC_BUFFER_EMPTY (7)
[00-00-00-00-00-04 3] Error: datalen: 24
[00-00-00-00-00-04 3] Error: 0000: 01 0d 00 18 00 00 00 b0 00 00 01 17 00 01 00 00 |................|
[00-00-00-00-00-04 3] Error: 0010: 00 00 00 08 ff fb 00 00 |........|
workstationOKlaptop = ["200.20.2.4"]
server2 = ["200.20.2.1"]
DNSserver = ["200.20.2.3"]
webserver = ["200.20.2.2"]
printer = ["200.20.2.10"]
guest = ["200.20.2.11"]
ipv4 = packet.find("ipv4")
arp = packet.find("arp")
icmp = packet.find("icmp")
tcp = packet.find("tcp")
if(ipv4):
src_ip = packet.payload.srcip
dst_ip = packet.payload.dstip
#print(src_ip)
#print(dst_ip)
# If the packet is ARP protocol
if(arp):
self.accept(event)
# if the packet is icmp and the des
if icmp and dst_ip in DNSserver:
self.drop(packet)
mininet@mininet-vm: ~/pox/pox/misc
File Edit Tabs Help
PING 200.20.2.3 (200.20.2.3) 56(84) bytes of data.
^C
--- 200.20.2.3 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4030ms
mininet> exit
mininet@mininet-vm:~/pox/pox/misc$ sudo python prelab3.py
mininet> Laptop1 ping Laptop2 -c 10
PING 200.20.2.9 (200.20.2.9) 56(84) bytes of data.
64 bytes from 200.20.2.9: icmp_seq=1 ttl=64 time=47.7 ms
64 bytes from 200.20.2.9: icmp_seq=2 ttl=64 time=16.1 ms
64 bytes from 200.20.2.9: icmp_seq=3 ttl=64 time=34.3 ms
64 bytes from 200.20.2.9: icmp_seq=4 ttl=64 time=2.32 ms
64 bytes from 200.20.2.9: icmp_seq=5 ttl=64 time=19.9 ms
64 bytes from 200.20.2.9: icmp_seq=6 ttl=64 time=38.7 ms
64 bytes from 200.20.2.9: icmp_seq=7 ttl=64 time=24.9 ms
64 bytes from 200.20.2.9: icmp_seq=8 ttl=64 time=43.6 ms
64 bytes from 200.20.2.9: icmp_seq=9 ttl=64 time=11.6 ms
64 bytes from 200.20.2.9: icmp_seq=10 ttl=64 time=31.8 ms
--- 200.20.2.9 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 2.328/27.150/47.798/13.938 ms
mininet> iperf ws1 Printer
*** Iperf: testing TCP bandwidth between ws1 and Printer
^C
Interrupt
mininet> iperfudp bw=10 Laptop1 ws3
*** Iperf: testing UDP bandwidth between Laptop1 and ws3
could not parse iperf output: ----------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 160 KByte (default)
*** Results: ['bw=10', '', '19.3 Kbits/sec']
mininet> iperfudp bw=10 ServDNS ws3
*** Iperf: testing UDP bandwidth between ServDNS and ws3
*** Results: ['bw=10', '19.3 Kbits/sec', '19.3 Kbits/sec']
mininet> exit
mininet@mininet-vm:~/pox/pox/misc$ sudo python prelab3.py
mininet> ws1 traceroute -U ServDNS
traceroute to 200.20.2.3 (200.20.2.3), 30 hops max, 60 byte packets
1 * 200.20.2.3 (200.20.2.3) 78.513 ms *
mininet>
mininet@mi
File Edit Tabs Help
mininet@mininet-vm:~$ date
Thu Nov 9 20:00:07 PST 2023
mininet@mininet-vm:~$
prelab3controll...
mininet@minine...
mininet@minine...
mininet@minine...
20:00


mininet@mininet-vm: ~
File Edit Tabs Help
INFO:packet:(dns) parsing questions: incomplete name
workstationOKlaptop = ["200.20.2.4"]
server2 = ["200.20.2.1"]
DNSserver = ["200.20.2.3"]
webserver = ["200.20.2.2"]
printer = ["200.20.2.10"]
guest = ["200.20.2.11"]
ipv4 = packet.find("ipv4")
arp = packet.find("arp")
icmp = packet.find("icmp")
tcp = packet.find("tcp")
if(ipv4):
src_ip = packet.payload.srcip
dst_ip = packet.payload.dstip
#print(src_ip)
#print(dst_ip)
# If the packet is ARP protocol
if(arp):
self.accept(event)
# if the packet is icmp and the des
if icmp and dst_ip in DNSserver:
self.drop(packet)
mininet@mininet-vm: ~/pox/pox/misc
File Edit Tabs Help
5 packets transmitted, 0 received, 100% packet loss, time 4030ms
mininet> exit
mininet@mininet-vm:~/pox/pox/misc$ sudo python prelab3.py
mininet> Laptop1 ping Laptop2 -c 10
PING 200.20.2.9 (200.20.2.9) 56(84) bytes of data.
64 bytes from 200.20.2.9: icmp_seq=1 ttl=64 time=47.7 ms
64 bytes from 200.20.2.9: icmp_seq=2 ttl=64 time=16.1 ms
64 bytes from 200.20.2.9: icmp_seq=3 ttl=64 time=34.3 ms
64 bytes from 200.20.2.9: icmp_seq=4 ttl=64 time=2.32 ms
64 bytes from 200.20.2.9: icmp_seq=5 ttl=64 time=19.9 ms
64 bytes from 200.20.2.9: icmp_seq=6 ttl=64 time=38.7 ms
64 bytes from 200.20.2.9: icmp_seq=7 ttl=64 time=24.9 ms
64 bytes from 200.20.2.9: icmp_seq=8 ttl=64 time=43.6 ms
64 bytes from 200.20.2.9: icmp_seq=9 ttl=64 time=11.6 ms
64 bytes from 200.20.2.9: icmp_seq=10 ttl=64 time=31.8 ms
--- 200.20.2.9 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 2.328/27.150/47.798/13.938 ms
mininet> iperf ws1 Printer
*** Iperf: testing TCP bandwidth between ws1 and Printer
^C
Interrupt
mininet> iperfudp bw=10 Laptop1 ws3
*** Iperf: testing UDP bandwidth between Laptop1 and ws3
could not parse iperf output: ----------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 160 KByte (default)
*** Results: ['bw=10', '', '19.3 Kbits/sec']
mininet> iperfudp bw=10 ServDNS ws3
*** Iperf: testing UDP bandwidth between ServDNS and ws3
*** Results: ['bw=10', '19.3 Kbits/sec', '19.3 Kbits/sec']
mininet> exit
mininet@mininet-vm:~/pox/pox/misc$ sudo python prelab3.py
mininet> ws1 traceroute -U ServDNS
traceroute to 200.20.2.3 (200.20.2.3), 30 hops max, 60 byte packets
1 * 200.20.2.3 (200.20.2.3) 78.513 ms *
mininet> ServDNS traceroute -U ws1
traceroute to 200.20.2.4 (200.20.2.4), 30 hops max, 60 byte packets
1 200.20.2.4 (200.20.2.4) 113.011 ms 113.916 ms 114.546 ms
mininet>
mininet@mi
File Edit Tabs Help
mininet@mininet-vm:~$ date
Thu Nov 9 20:00:42 PST 2023
mininet@mininet-vm:~$
prelab3controll...
mininet@minine...
mininet@minine...
mininet@minine...
20:00