Pre-lab Questions:

1. [5] What command will show you which groups you are a member of?

   `groups`

2. [5] What does the environmental variable "$?" hold? (Hint: the command 'echo $?' will should you this on your screen)

   `$?` holds the return value of the last executed command.

3. [5] What key combination will suspend a currently running process and place it as a background process?

   `Ctrl + Z` to suspend a running process followed by `bg` to place in background process.

4. [5] With what command (and arguments) can you find out your kernel version and the "nodename"?

   `uname -vn`

   The arguments "v" refers to kernel version and "n" refers to nodename.

5. [5] What is the difference between the paths ".", "..", and "~"? What does the path "/" refer to when not proceeded by anything?

   `.` refers to the current working directory.

   `..` refers to the parent directory of the current working directory.

   ~ refers to home directory of the current user.

   / refers to the root directory.

6. [5] What is a pid? Which command would you use to find the "pid" for a running process?

   A PID or process identifier is a unique, non-negative integer, assigned by the operating system to identify a running process. Command `ps` displays information, including the PID, about the currently running processes.

7. [20] Write a single command that will return every user's default shell. [You may chain commands using piping and redirects] (Hint: See 'Chapter 19: filters' of linux-training.be as well as the man page for the /etc/passwd file: https://linux.die.net/man/5/passwd)

   `cut -d: -f7 /etc/passwd`

   Citation(s):

Daniel J. Barrett. (2023). KAPITEL 5: Das Arsenal erweitern. In *Produktiv auf der Linux-Kommandozeile: Sicher und souverän mit linux arbeiten*. O'Reilly.

8. [10] What is the difference between “sudo” and “su root”?

   `sudo` “substitute user and do” allows a user to execute a command as another user specified in `/etc/sudoers` configuration with root privileges.

   `su root` “switch user root” switches to the root account.

9. [10] How would you tell your computer to run a program or script on a schedule or set interval on Linux? E.g. Run this program once every 30 minutes.

   Use the `cron` daemon to schedule tasks or cron jobs at specified intervals. To run a program once every 30 minutes:

   First, edit the current user’s cron table.

   ```
   crontab -e
   ```

   Next, add this new line:

   ```
   */30 * * * * /path/to/script-or-command
   ```

   `/30`: This means every 30 minutes.

   `*`: Any value for the hour field (0-23).

   `*`: Any value for the day of the month field (1-31).

   `*`: Any value for the month field (1-12).

   `*`: Any value for the day of the week field (0-7, where both 0 and 7 represent Sunday).

   `/path/to/script-or-command`: The full path to the script or command you want to execute.

   Lastly, save the file and exit the editor.

10. [30] Write a shell script that only prints the even numbered lines of each file in the current directory. The output should be filename: line for each even numbered line. You do not need to print line numbers.

    ```
    #!/bin/bash

    for file in *; do
      if [[ -f "$file" ]]; then
    ```

```
    awk 'NR % 2 == 0 { print FILENAME ": " $0 }' "$file"
  fi
done
```

```
mininet@mininet-vm:~/Desktop/test$ ./JoeyMa-script.sh
file1: line 2
file1: line 4
file2: line 2
file2: line 4
file3: line 2
file3: line 4
JoeyMa-script.sh:
JoeyMa-script.sh:   if [[ -f "$file" ]]; then
JoeyMa-script.sh:   fi
JoeyMa-script.sh:
```

Citation(s):

Robbins, A. (2015). Chapter 1. Getting Started with awk. In *Effective Awk Programming: Universal Text Processing and Pattern Matching* (p. 12), O'Reilly.

The Lab:

1. In Mininet change the default configuration to have 4 hosts connected to a switch.

```
mininet@mininet-vm:~$ sudo mn --topo single,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

2. [30 pts] Save a screenshot of dump and pingall output. Explain what is being shown in the screenshot

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=6685>
<Host h2: h2-eth0:10.0.0.2 pid=6689>
<Host h3: h3-eth0:10.0.0.3 pid=6691>
<Host h4: h4-eth0:10.0.0.4 pid=6693>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None,s1-eth4:None
pid=6698>
<Controller c0: 127.0.0.1:6633 pid=6678>
mininet>
```

The `dump` command in Mininet prints the current network configuration, including hosts, switches, and the controller. Following problem 1, it shows the four hosts (h1, h2, h3, and h4)

with their respective IP addresses (10.0.0.1 to 10.0.0.4) and process IDs (pid). It also shows the Open vSwitch (OVSSwitch s1) with its local loopback address (lo:127.0.0.1) and its interfaces (s1-eth1 to s1-eth4) with 'None' as their IP addresses since they're switch interfaces. Lastly, it shows the default Mininet controller (Controller c0) with its IP address (127.0.0.1) and port (6633) along with its process ID (pid).

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

The `pingall` command in Mininet tests the connectivity between all pairs of hosts in the network. It sends ICMP ECHO_REQUEST packets (pings) from each host to every other host and checks if they receive a response (ICMP ECHO_REPLY). Following problem 1, the output shows that all hosts (h1, h2, h3, and h4) can successfully ping each other. There is no packet loss (0% dropped), and all 12 pings (4 hosts * 3 pings/host) were successfully received.

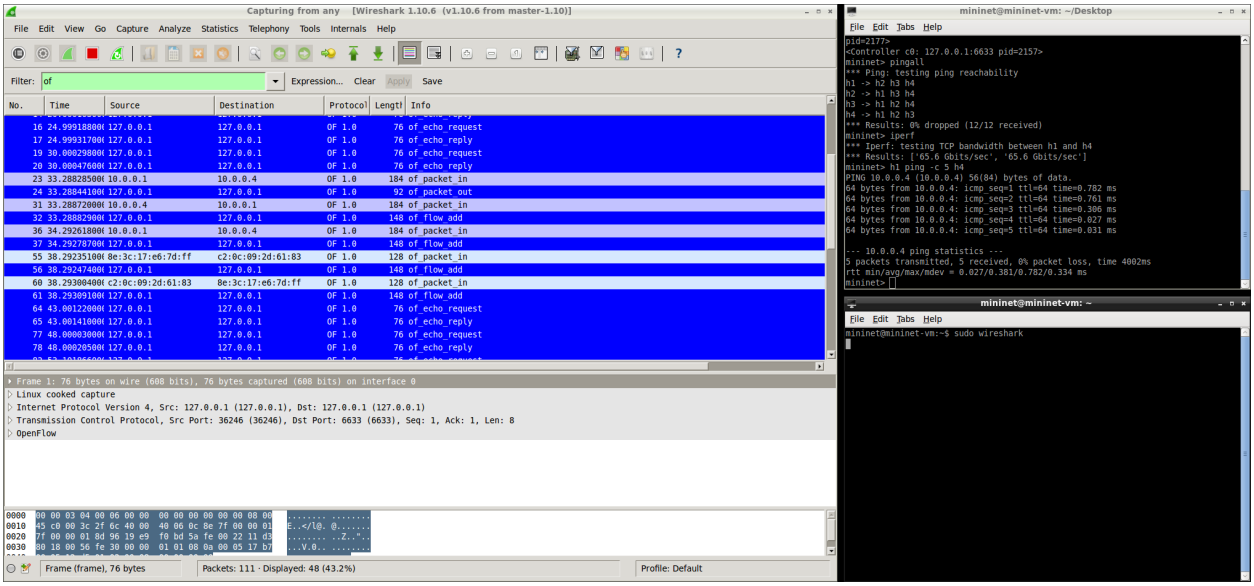
3. [10 pts] Run the iperf command as well, and screenshot the output, how fast is the connect?

   After running `iperf` The connection speed between h1 and h4 is measured at 65.6 Gbits/sec and 65.6 Gbits/sec.

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['65.6 Gbits/sec', '65.6 Gbits/sec']
mininet>
```

4. Run wireshark, and using the display filter, typing “of” in filter line and remember to click Apply after you start. Note: When you run wireshark you should do so as “sudo wireshark” in a new terminal. When you choose an interface to capture on, you should select “any”.
   a. [20 pts] Run ping from a host to any other host using hX ping -c 5 hY. How many of_packet_in messages show up? Take a screenshot of your results.

      After running `h1 ping -c 5 h4`, 5 of_packet_in messages showed up.



b. [20 pts] What is the source and destination IP addresses for these entries? Find another packet that matches the "of" filter with the OpenFlow typefield set to OF_PACKET_OUT. What is the source and destination IP address for this entry? Take screenshots showing your results.

For the first part of this question, the source of destination of these entries of of_packet_in are listed below:

Frame 23: Source IP 10.0.0.1 (h1) and Destination IP 10.0.0.4 (h4)

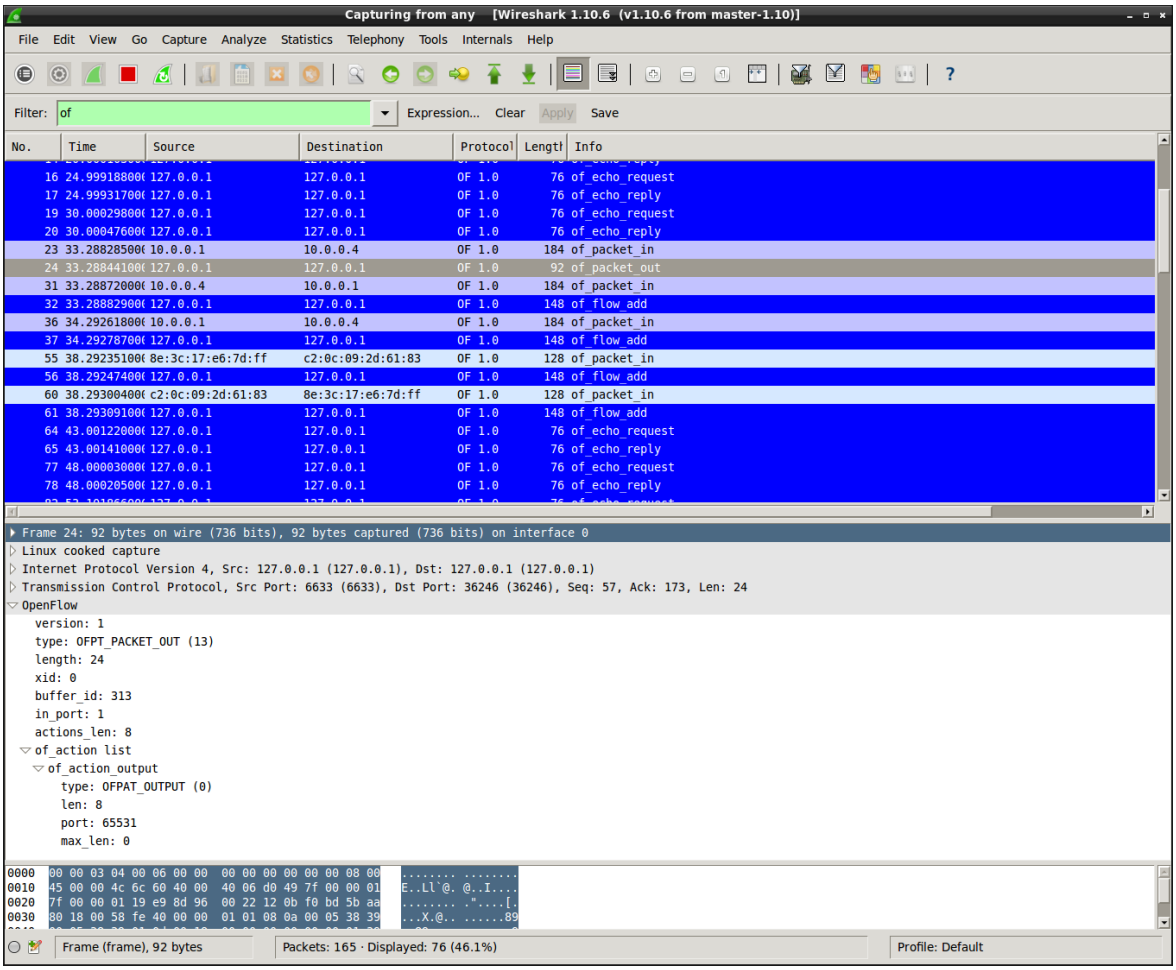
Frame 31: Source IP 10.0.0.4 (h4) and Destination IP 10.0.0.1 (h1)

Frame 36: Source IP 10.0.01 (h1) and Destination IP 10.0.0.4 (h4)

Frame 55: Source MAC 8e:3c:17:e6:7d:ff (IP 127.0.0.1) and Destination MAC c2:0c:09:2d:61:83 (IP 127.0.0.1)

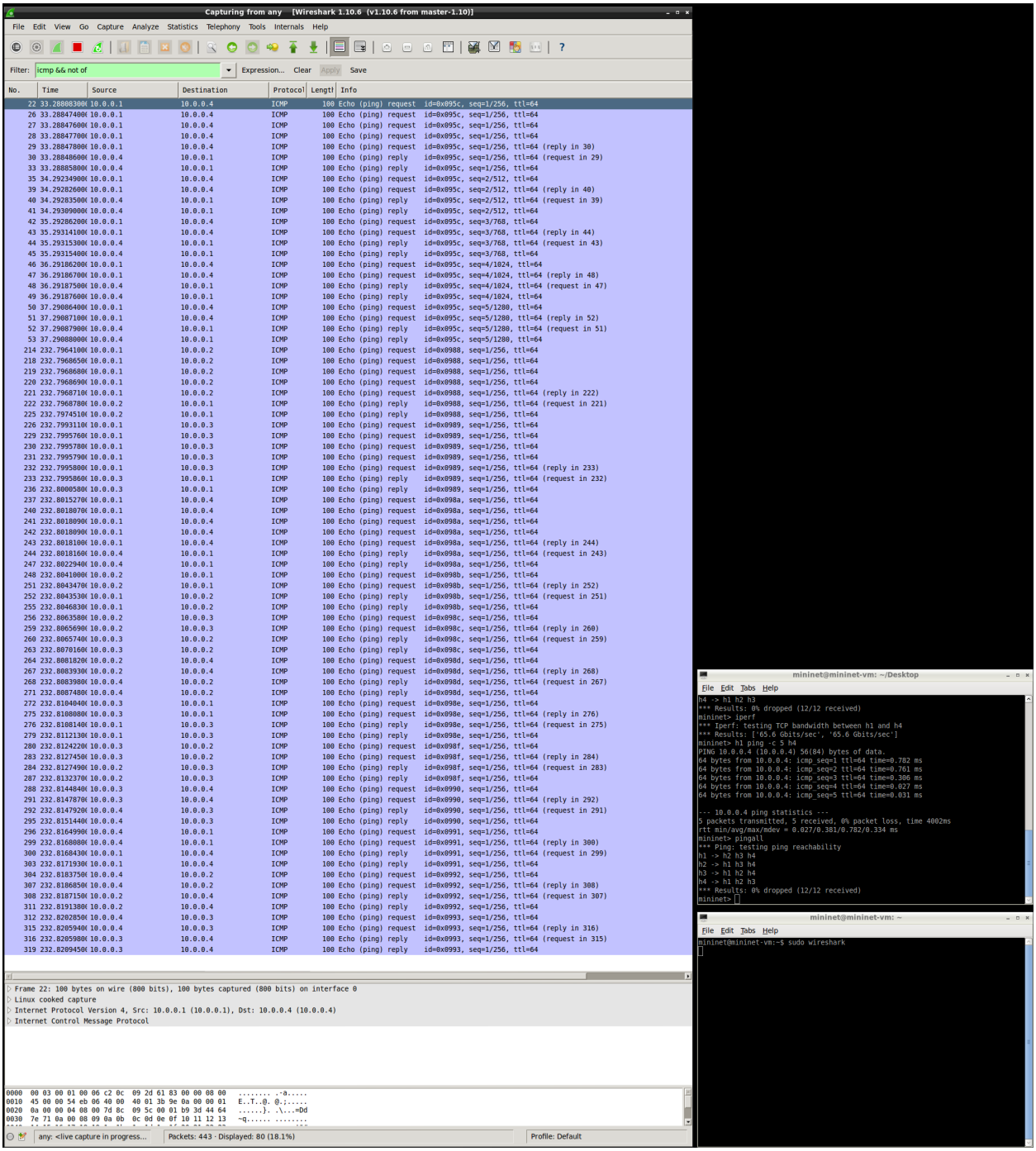
Frame 60: Source MAC c2:0c:09:2d:61:83 (IP 127.0.0.1) and Destination MAC 8e:3c:17:e6:7d:ff (IP 127.0.0.1)

For the second part of this question, the source of the OF_PACKET_OUT entry is 127.0.0.1 and its destination is 127.0.0.1.



c. [20 pts] Replace the display filter for “of” to “icmp && not of”. Run pingall again, how many entries are generated in wireshark? What types of icmp entries show up? Take a screenshot of your results.

57 more displayed entries were generated from existing 23 displayed entries before, totaling 80 displayed entries. There are four devices with IP addresses 10.0.0.1, 10.0.0.2, 10.0.0.3, and 10.0.0.4 communicating using the ICMP (Internet Control Message Protocol) with echo request (ping) and echo reply messages. The devices are receiving echo replies from the devices they sent echo requests to, which means that communication between these devices is working properly.


Capturing from any [Wireshark 1.10.6 (v1.10.6 from master-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: icmp && not of
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info
22 33.28808300(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=1/256, ttl=64
26 33.28847400(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=1/256, ttl=64
27 33.28847600(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=1/256, ttl=64
28 33.28847700(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=1/256, ttl=64
29 33.28847800(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=1/256, ttl=64 (reply in 30)
30 33.28848600(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=1/256, ttl=64 (request in 29)
33 33.28885800(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=1/256, ttl=64
35 34.29234900(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=2/512, ttl=64
39 34.29282600(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=2/512, ttl=64 (reply in 40)
40 34.29283500(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=2/512, ttl=64 (request in 39)
41 34.29309000(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=2/512, ttl=64
42 35.29286200(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=3/768, ttl=64
43 35.29314100(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=3/768, ttl=64 (reply in 44)
44 35.29315300(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=3/768, ttl=64 (request in 43)
45 35.29315400(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=3/768, ttl=64
46 36.29186200(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=4/1024, ttl=64
47 36.29186700(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=4/1024, ttl=64 (reply in 48)
48 36.29187500(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=4/1024, ttl=64 (request in 47)
49 36.29187600(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=4/1024, ttl=64
50 37.29086400(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=5/1280, ttl=64
51 37.29087100(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x095c, seq=5/1280, ttl=64 (reply in 52)
52 37.29087900(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=5/1280, ttl=64 (request in 51)
53 37.29088000(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x095c, seq=5/1280, ttl=64
214 232.7964100(10.0.0.1 10.0.0.2 ICMP 100 Echo (ping) request id=0x0988, seq=1/256, ttl=64
218 232.7968650(10.0.0.1 10.0.0.2 ICMP 100 Echo (ping) request id=0x0988, seq=1/256, ttl=64
219 232.7968680(10.0.0.1 10.0.0.2 ICMP 100 Echo (ping) request id=0x0988, seq=1/256, ttl=64
220 232.7968690(10.0.0.1 10.0.0.2 ICMP 100 Echo (ping) request id=0x0988, seq=1/256, ttl=64
221 232.7968710(10.0.0.1 10.0.0.2 ICMP 100 Echo (ping) request id=0x0988, seq=1/256, ttl=64 (reply in 222)
222 232.7968780(10.0.0.2 10.0.0.1 ICMP 100 Echo (ping) reply id=0x0988, seq=1/256, ttl=64 (request in 221)
225 232.7974510(10.0.0.2 10.0.0.1 ICMP 100 Echo (ping) reply id=0x0988, seq=1/256, ttl=64
226 232.7993110(10.0.0.1 10.0.0.3 ICMP 100 Echo (ping) request id=0x0989, seq=1/256, ttl=64
229 232.7995760(10.0.0.1 10.0.0.3 ICMP 100 Echo (ping) request id=0x0989, seq=1/256, ttl=64
230 232.7995780(10.0.0.1 10.0.0.3 ICMP 100 Echo (ping) request id=0x0989, seq=1/256, ttl=64
231 232.7995790(10.0.0.1 10.0.0.3 ICMP 100 Echo (ping) request id=0x0989, seq=1/256, ttl=64
232 232.7995800(10.0.0.1 10.0.0.3 ICMP 100 Echo (ping) request id=0x0989, seq=1/256, ttl=64 (reply in 233)
233 232.7995860(10.0.0.3 10.0.0.1 ICMP 100 Echo (ping) reply id=0x0989, seq=1/256, ttl=64 (request in 232)
236 232.8000580(10.0.0.3 10.0.0.1 ICMP 100 Echo (ping) reply id=0x0989, seq=1/256, ttl=64
237 232.8015270(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x098a, seq=1/256, ttl=64
240 232.8018070(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x098a, seq=1/256, ttl=64
241 232.8018090(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x098a, seq=1/256, ttl=64
242 232.8018090(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x098a, seq=1/256, ttl=64
243 232.8018100(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) request id=0x098a, seq=1/256, ttl=64 (reply in 244)
244 232.8018160(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x098a, seq=1/256, ttl=64 (request in 243)
247 232.8022940(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) reply id=0x098a, seq=1/256, ttl=64
248 232.8041000(10.0.0.2 10.0.0.1 ICMP 100 Echo (ping) request id=0x098b, seq=1/256, ttl=64
251 232.8043470(10.0.0.2 10.0.0.1 ICMP 100 Echo (ping) request id=0x098b, seq=1/256, ttl=64 (reply in 252)
252 232.8043530(10.0.0.1 10.0.0.2 ICMP 100 Echo (ping) reply id=0x098b, seq=1/256, ttl=64 (request in 251)
255 232.8046830(10.0.0.1 10.0.0.2 ICMP 100 Echo (ping) reply id=0x098b, seq=1/256, ttl=64
256 232.8063580(10.0.0.2 10.0.0.3 ICMP 100 Echo (ping) request id=0x098c, seq=1/256, ttl=64
259 232.8065690(10.0.0.2 10.0.0.3 ICMP 100 Echo (ping) request id=0x098c, seq=1/256, ttl=64 (reply in 260)
260 232.8065740(10.0.0.3 10.0.0.2 ICMP 100 Echo (ping) reply id=0x098c, seq=1/256, ttl=64 (request in 259)
263 232.8070160(10.0.0.3 10.0.0.2 ICMP 100 Echo (ping) reply id=0x098c, seq=1/256, ttl=64
264 232.8081820(10.0.0.2 10.0.0.4 ICMP 100 Echo (ping) request id=0x098d, seq=1/256, ttl=64
267 232.8083930(10.0.0.2 10.0.0.4 ICMP 100 Echo (ping) request id=0x098d, seq=1/256, ttl=64 (reply in 268)
268 232.8083980(10.0.0.4 10.0.0.2 ICMP 100 Echo (ping) reply id=0x098d, seq=1/256, ttl=64 (request in 267)
271 232.8087480(10.0.0.4 10.0.0.2 ICMP 100 Echo (ping) reply id=0x098d, seq=1/256, ttl=64
272 232.8104040(10.0.0.3 10.0.0.1 ICMP 100 Echo (ping) request id=0x098e, seq=1/256, ttl=64
275 232.8108080(10.0.0.3 10.0.0.1 ICMP 100 Echo (ping) request id=0x098e, seq=1/256, ttl=64 (reply in 276)
276 232.8108140(10.0.0.1 10.0.0.3 ICMP 100 Echo (ping) reply id=0x098e, seq=1/256, ttl=64 (request in 275)
279 232.8112130(10.0.0.1 10.0.0.3 ICMP 100 Echo (ping) reply id=0x098e, seq=1/256, ttl=64
280 232.8124220(10.0.0.3 10.0.0.2 ICMP 100 Echo (ping) request id=0x098f, seq=1/256, ttl=64
283 232.8127450(10.0.0.3 10.0.0.2 ICMP 100 Echo (ping) request id=0x098f, seq=1/256, ttl=64 (reply in 284)
284 232.8127490(10.0.0.2 10.0.0.3 ICMP 100 Echo (ping) reply id=0x098f, seq=1/256, ttl=64 (request in 283)
287 232.8132370(10.0.0.2 10.0.0.3 ICMP 100 Echo (ping) reply id=0x098f, seq=1/256, ttl=64
288 232.8144840(10.0.0.3 10.0.0.4 ICMP 100 Echo (ping) request id=0x0990, seq=1/256, ttl=64
291 232.8147870(10.0.0.3 10.0.0.4 ICMP 100 Echo (ping) request id=0x0990, seq=1/256, ttl=64 (reply in 292)
292 232.8147920(10.0.0.4 10.0.0.3 ICMP 100 Echo (ping) reply id=0x0990, seq=1/256, ttl=64 (request in 291)
295 232.8151440(10.0.0.4 10.0.0.3 ICMP 100 Echo (ping) reply id=0x0990, seq=1/256, ttl=64
296 232.8164990(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) request id=0x0991, seq=1/256, ttl=64
299 232.8168080(10.0.0.4 10.0.0.1 ICMP 100 Echo (ping) request id=0x0991, seq=1/256, ttl=64 (reply in 300)
300 232.8168430(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) reply id=0x0991, seq=1/256, ttl=64 (request in 299)
303 232.8171930(10.0.0.1 10.0.0.4 ICMP 100 Echo (ping) reply id=0x0991, seq=1/256, ttl=64
304 232.8183750(10.0.0.4 10.0.0.2 ICMP 100 Echo (ping) request id=0x0992, seq=1/256, ttl=64
307 232.8186850(10.0.0.4 10.0.0.2 ICMP 100 Echo (ping) request id=0x0992, seq=1/256, ttl=64 (reply in 308)
308 232.8187150(10.0.0.2 10.0.0.4 ICMP 100 Echo (ping) reply id=0x0992, seq=1/256, ttl=64 (request in 307)
311 232.8191380(10.0.0.2 10.0.0.4 ICMP 100 Echo (ping) reply id=0x0992, seq=1/256, ttl=64
312 232.8202850(10.0.0.4 10.0.0.3 ICMP 100 Echo (ping) request id=0x0993, seq=1/256, ttl=64
315 232.8205940(10.0.0.4 10.0.0.3 ICMP 100 Echo (ping) request id=0x0993, seq=1/256, ttl=64 (reply in 316)
316 232.8205980(10.0.0.3 10.0.0.4 ICMP 100 Echo (ping) reply id=0x0993, seq=1/256, ttl=64 (request in 315)
319 232.8209450(10.0.0.3 10.0.0.4 ICMP 100 Echo (ping) reply id=0x0993, seq=1/256, ttl=64
Frame 22: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.0.4 (10.0.0.4)
Internet Control Message Protocol
0000 00 03 00 01 00 06 c2 0c 09 2d 61 83 00 00 08 00 ..... .-a.....
0010 45 00 00 54 eb 06 40 00 40 01 3b 9e 0a 00 00 01 E..T..@. @.;.....
0020 0a 00 00 04 08 00 7d 8c 09 5c 00 01 b9 3d 44 64 ......}. .\...=Dd
0030 7e 71 0a 00 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 ~q..... .....
any: <live capture in progress...
Packets: 443 · Displayed: 80 (18.1%)
Profile: Default
mininet@mininet-vm: ~/Desktop
File Edit Tabs Help
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['65.6 Gbits/sec', '65.6 Gbits/sec']
mininet> h1 ping -c 5 h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.782 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.761 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.306 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.027 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.031 ms
--- 10.0.0.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4002ms
rtt min/avg/max/mdev = 0.027/0.381/0.782/0.334 ms
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
mininet@mininet-vm: ~
File Edit Tabs Help
mininet@mininet-vm:~$ sudo wireshark