Otto-Friedrich-University Bamberg

Professorship for Computer Science,
Communication Services, Telecommunication,
Systems and Computer Networks

Foundation of Internet Communication

Assignement1

Submitted by:
Group J

Reem Khalil
Shivasharan Reddy
Azar Ghadami
Reema Miranda
Manjunath B Marigoudar

Supervisor: Prof. Dr. Udo Krieger

Bamberg, May 10, 2020
Summer Term 2020

Chapter 1

Wireshark Handling

Explain what we are going to do in this section here

1.1 Provide a short documentation that explained its actions

Wireshark is a network protocol analayzer. Using wireshark, we can monitor what's happeing on the network.

Who use Wireshark?

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- QA engineers use it to verify network applications
- Developers use it to debug protocol implementations
- People use it to learn network protocol internalshttps://www.overleaf.com/project/5eac

Featues

- Available for UNIX and Windows.
- it is an open source software project, and is released under the GNU General Public License (GPL)
- Capture live packet data from a network interface including Ethernet, Wireless LAN,...

- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Save packet data captured.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.

Wireshark is not

- it isn't an intrusion detection system. It will not warn you when someone does strange things on your network . However, if it happens, Wireshark can help you figure out what is really going on.
- Wireshark will not manipulate things on the network.. Wireshark doesn't send packets on the network.

Supported platforms

Wireshark runs on most UNIX and UNIX-like platforms including Linux and most BSD variants. It also runs on various Windows platforms

1.2 How long was the period between the sending of the HTTP GET message until the receipt of the HTTP OK reply? Provide screenshots

183 ms.

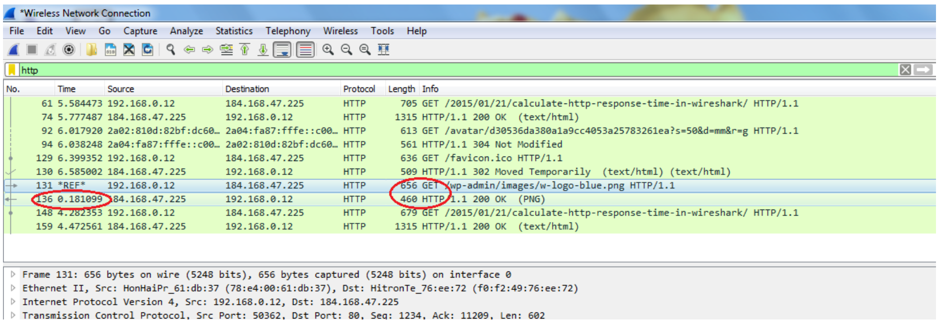


Figure 1.1: Http Response Time

1.3 What are the IP addresses of the servers www.fau.de and www.denic.de?

www.fau.de: 131.188.16.209 , www.denic.de: 81.91.170.12

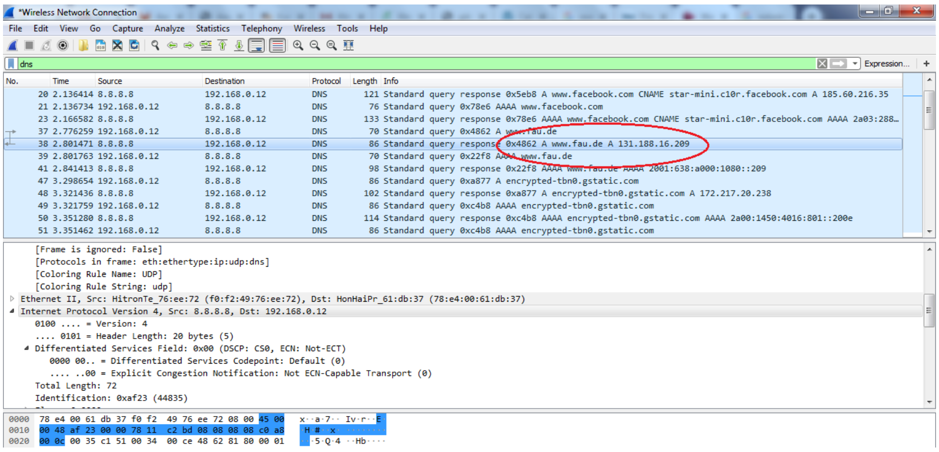


Figure 1.2: fau.de IP Address

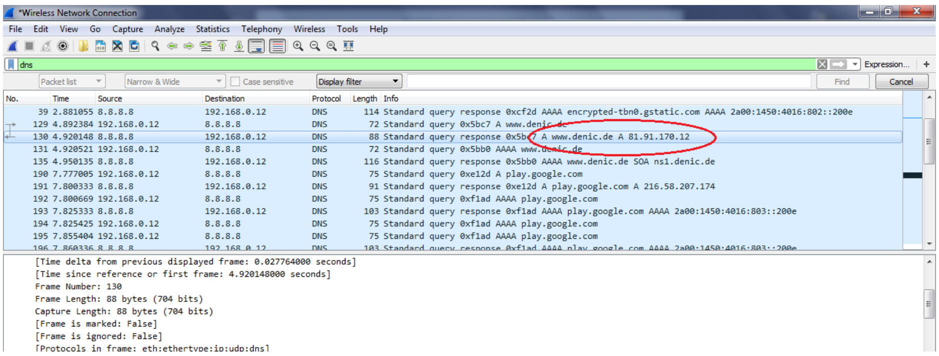


Figure 1.3: denic.de IP Address

Chapter 2

Wireshark Packet Filtering

2.1 Ping tool

- Ping tool is utility used for troubleshooting, testing an diagnosing issues related to network connectivity

- When we use Wireshark for capturing live packets in the background we parallelly use terminal and ping www.google.com

- We let Wireshark to capture the packets for few seconds

```
PING www.google.com (172.217.163.132): 56 data bytes
64 bytes from 172.217.163.132: icmp_seq=0 ttl=52 time=73.983 ms
64 bytes from 172.217.163.132: icmp_seq=1 ttl=52 time=85.190 ms
64 bytes from 172.217.163.132: icmp_seq=2 ttl=52 time=87.461 ms
64 bytes from 172.217.163.132: icmp_seq=3 ttl=52 time=58.847 ms
64 bytes from 172.217.163.132: icmp_seq=4 ttl=52 time=49.609 ms
^C
--- www.google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 49.609/71.018/87.461/14.741 ms
```

Figure 2.1: Pinging www.google.com

- Using wireshark packet filter ip.addr== host/destination address.
  eg:- ip.addr == 172.217.163.132

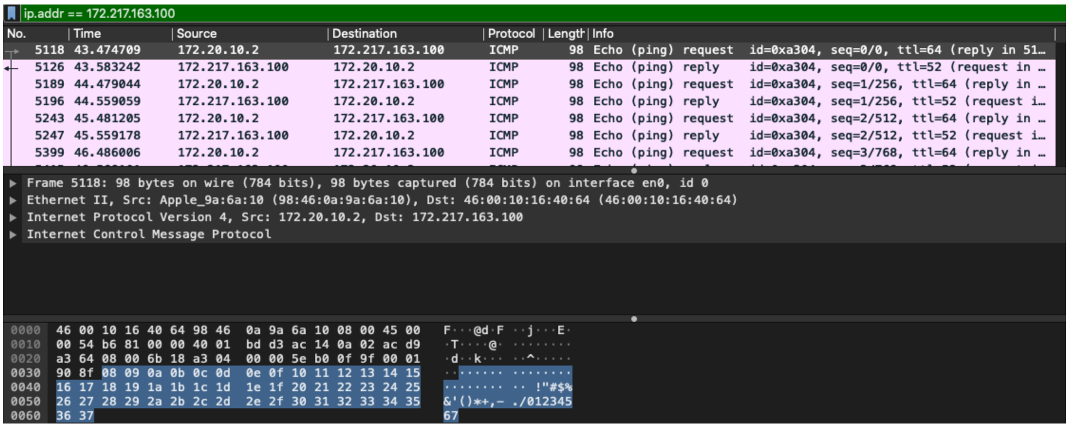


Figure 2.2: Display filter

2.2 Packets sent by the host IP Address

- The Packets sent by the host when we visit the URL http://www.caida.org/tools/visualization/mapnet can be displayed with the help of the display filter ip.addr == destination address.
  eg:- ip.addr == 192.168.43.251.

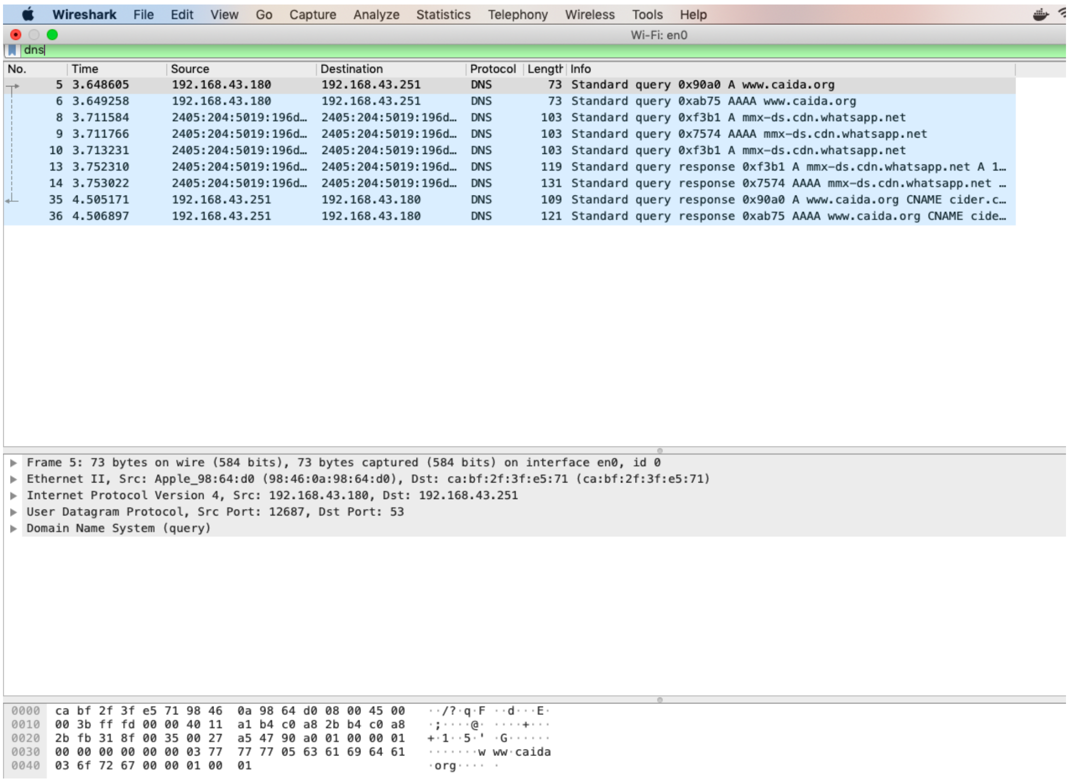


Figure 2.3: Destination ip Address with DNS filter

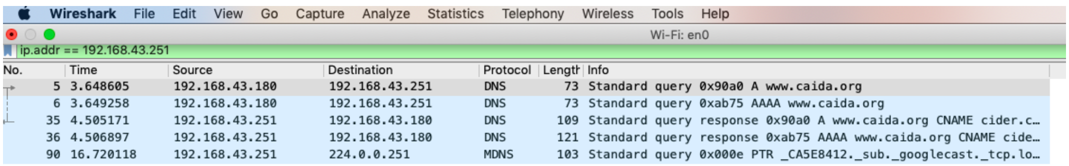


Figure 2.4: IP address display filter

2.3 Packets sent by the host MAC Address

- The Mac address of my system is 98:46:0a:98:64:d0.
- The packets sent by the host MAC address, if we visit the URL http://www.caida.org/tools/visualization/mapnet can be displayed with the filter eth.addr== mac address
  eg:- eth.addr== 98:46:0a:98:64:d0

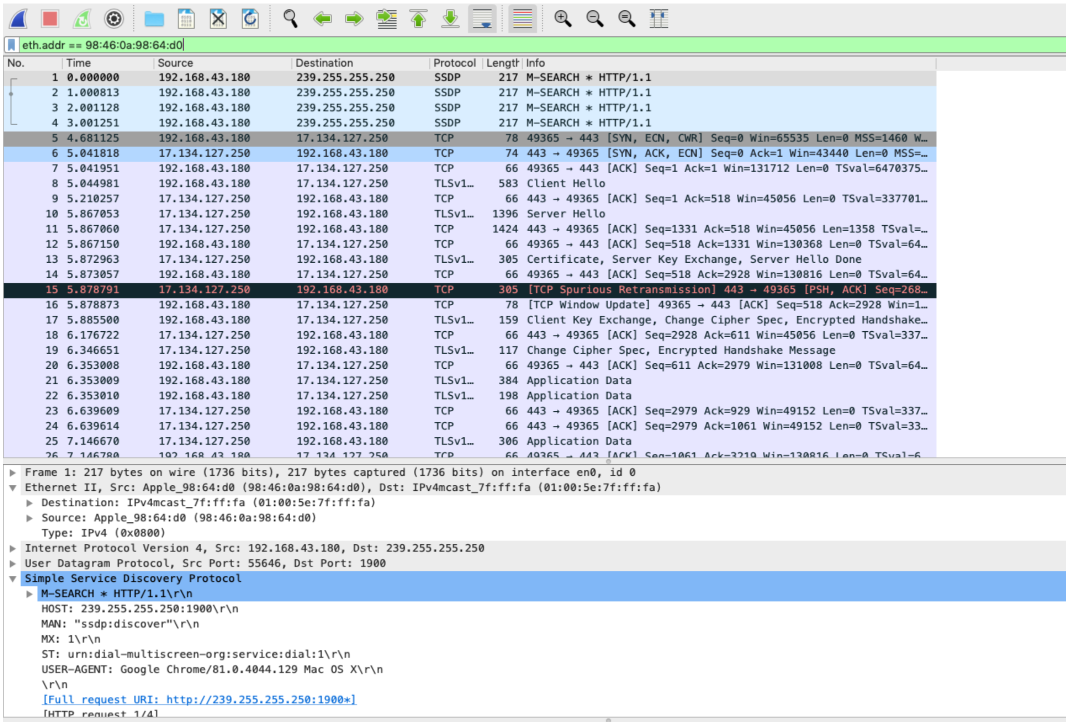


Figure 2.5: Mac address display filter

2.4 Difference between MAC and IP address

- MAC stands for Media Access Control where as IP stands for Internet Protocol address.
- The main difference between MAC and IP address is that, MAC Address indicates the physical address of computer. It uniquely identifies the devices on a network. Where as IP Address is the logical address of the computer is used to uniquely identifies the connection of network with that device in a network.

- Mac address is a six byte hexa decimal address while ip address is either four(IPV4) byte or six (IPV6)byte address.

- Mac address operates in the data link layer and ip address operates in the network layer

2.5 MAC address usage

- Yes, our system need MAC address for many reasons as below,

- MAC address is needed for to make a connection to local Ethernet (or wifi) network function.

- For a device to communicate with a machine on the LAN (Local Area Network) MAC address is mandatory.

- Its a low level unique ID for network device.

- The routers and switches use MAC address tables to find out what devices lie on what ports. This is used to intelligently move packets to the right port.

- It's easier to use MAC address than IP address as a network card can have more then one IP address assigned to it at once, hence it's more efficient to store the MAC instead.

2.6 Packets recieved by the host

- The packets recieved by the host can be displayed with the display filter: ip.dst=host ip address
  eg:- ip.dst=92.168.43.180

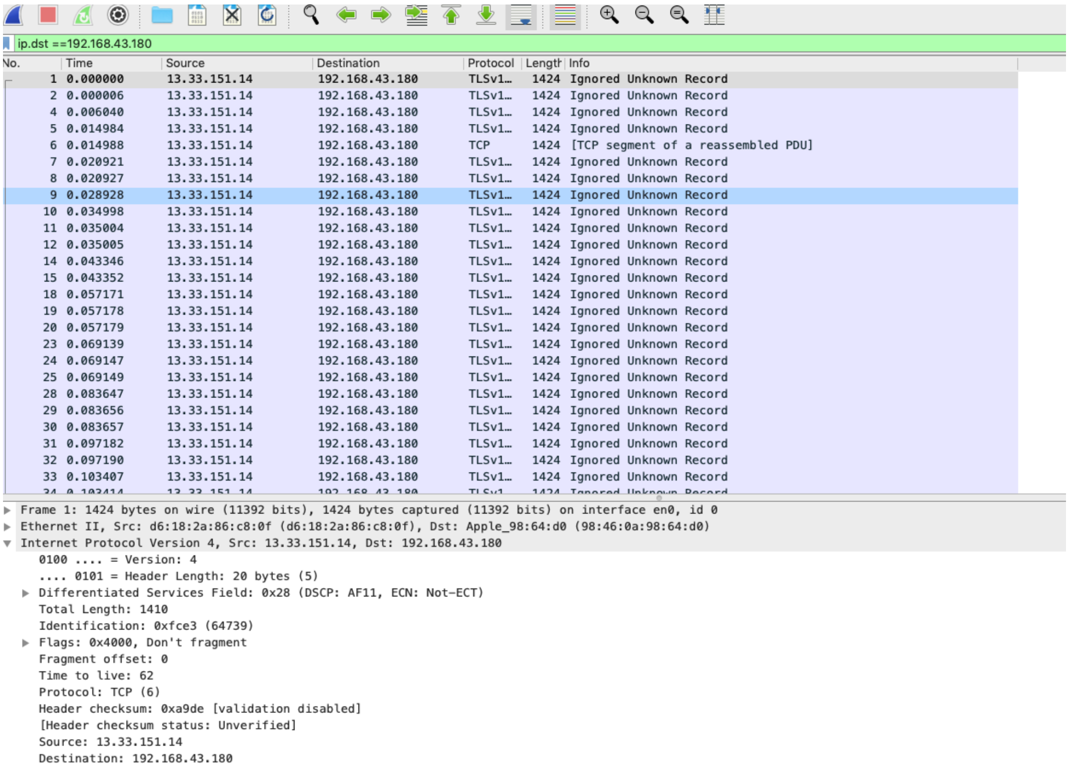


Figure 2.6: Packets recieved by the host

2.7 Tcp filter to capture packets

- Tcp expression that captures packets containing IP datagrams with a source or destination IP address equal to your IP address is as follows:

  1)host 192.168.43.180.

2.8 Tcp filter to capture packets between 2 hosts

- Capture filter expression that captures packets containing IP datagrams between two hosts with IP addresses 10.0.0.3 and 10.0.0.12 both on interface eth0 is as follows
  1)host 10.0.1.11 or host 10.0.1.12

2.9 TCP Packet Capture

- Tcpdump expression that captures TCP packets using port number 22.

  1) tcp dst port 22

  2) tcp src port 22

2.10 Display Packets with defined frame size

- Syntax for a display filter that shows only IP datagrams with a destination IP address equal to 192.168.178.1 and frame size greater than 350 bytes
  ip.dst == 192.168.178.1 and frame.len >350

Chapter 3

Basic Linux network administration

In this section we are going to set up a small emulated LAN of the topology in figure 3.1 using basic Linux network functionality and Kathará.

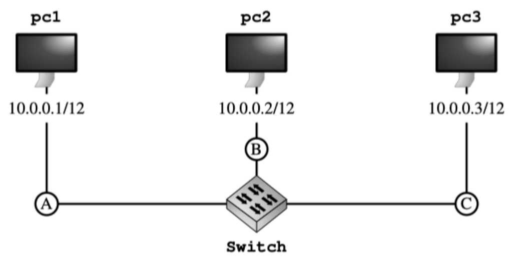


Figure 1: LAN setup

Linux PC	IP Address	Interface	CD
pc1	10.0.0.1/12	eth0	A
pc2	10.0.0.2/12	eth0	B
pc3	10.0.0.3/12	eth0	C
switch		eth0	A
		eth1	B
		eth2	C

Figure 3.1: LAN switching configuration

3.1 Creating the four devices

3.1.1 Creating Three PCs each assigned a network interface and unique collision domain

```
Ass1Ktr — -bash — 80×24
(base) Reems-MBP:Ass1Ktr reemeslam$ kathara vstart -n pc1 --eth 0:A
===================== Starting Machine ======================
Deploying links... |################################| 1/1
Deploying machines... |################################| 1/1
(base) Reems-MBP:Ass1Ktr reemeslam$ kathara vstart -n pc2 --eth 0:B
===================== Starting Machine ======================
Deploying links... |################################| 1/1
Deploying machines... |################################| 1/1
(base) Reems-MBP:Ass1Ktr reemeslam$ kathara vstart -n pc3 --eth 0:C
===================== Starting Machine ======================
Deploying links... |################################| 1/1
Deploying machines... |################################| 1/1
```

Figure 3.2: Created three PCs each assigned a network interface with unique collision domain

3.1.2 Creating a switch that has three network interfaces attached, where each interface is connected to a different collision domain

```
(base) Reems-MBP:Ass1Ktr reemeslam$ kathara vstart -n switch --eth 0:A 1:B 2:C
===================== Starting Machine ======================
Deploying links... |#####################           | 2/3Deploying links... |###
#############################| 3/3
Deploying machines... |################################| 1/1
```

Figure 3.3: Created a Switch with three network interfaces, where each interface is connected to a different collision domain

3.2 Configuring the four devices

3.2.1 Configuring an IP address for PC1, PC2 and PC2

```
reemeslam — root@pc1: / — kathara ◂ kathara connect -l pc1 — 80×24
[root@pc1:/# ip addr add 10.0.0.1/12 dev eth0
```

Figure 3.4: configured IP address for PC1

```
reemeslam — root@pc2: / — kathara ◂ kathara connect -l pc2 — 80×24
[root@pc2:/# ip addr add 10.0.0.2/12 dev eth0
```

Figure 3.5: configured IP address for PC2

```
reemeslam — root@pc3: / — kathara ◂ kathara connect -l pc3 — 80×24
[root@pc3:/# ip addr add 10.0.0.3/12 dev eth0
```

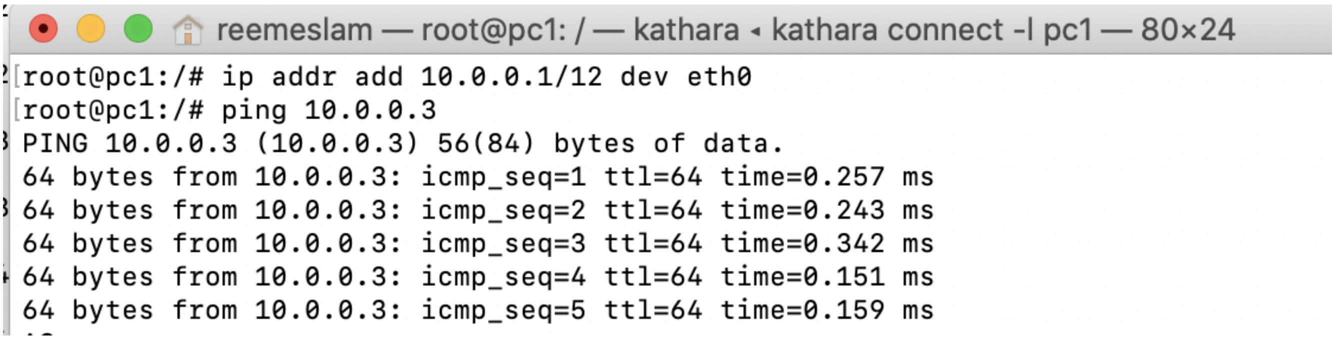
Figure 3.6: configured IP address for PC3

3.2.2 Configuring a bridge for the switch, set-up the environment and turning on the bridge

```
[root@switch:/# brctl addbr switch
[root@switch:/# brctl addif switch eth0 eth1 eth2
[root@switch:/# ifconfig eth1 up
[root@switch:/# ifconfig eth2 up
[root@switch:/# ifconfig switch up
```

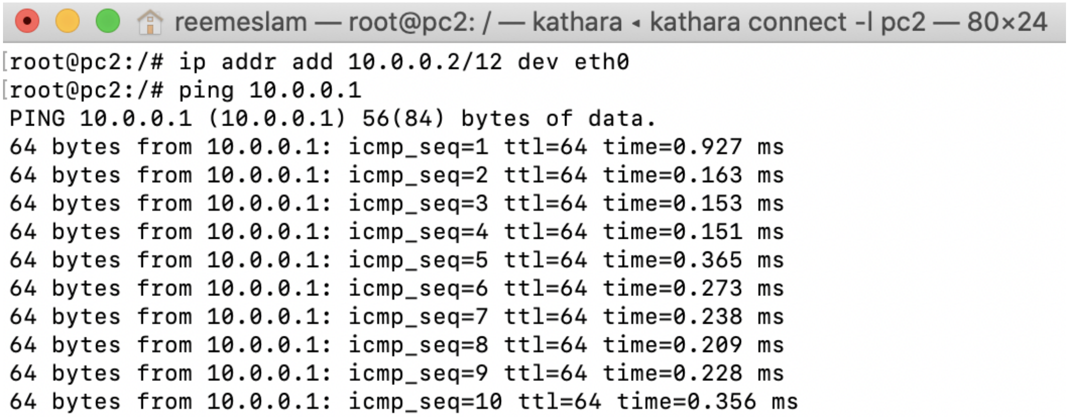
Figure 3.7: Created a bridge with name switch, Added the three interfaces to the switch, turned on eth1 & eth2 and Turned the bridge on

3.2.3 Testing the network by sending pings between PCs



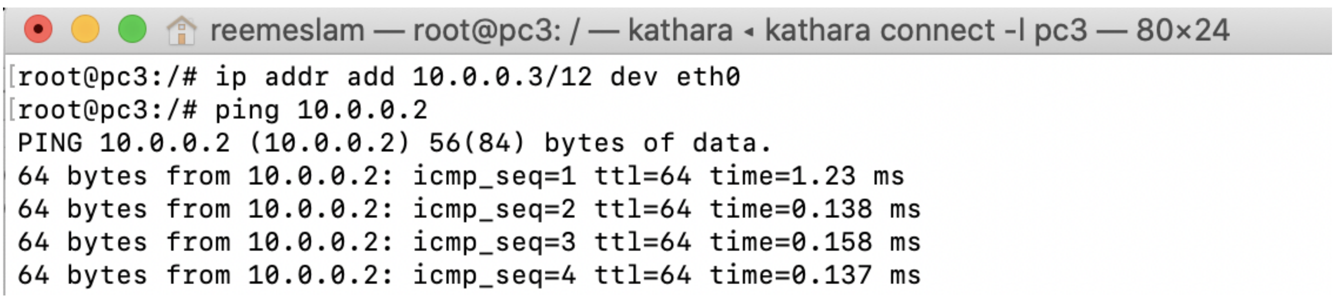
```
reemeslam — root@pc1: / — kathara ◂ kathara connect -l pc1 — 80×24
root@pc1:/# ip addr add 10.0.0.1/12 dev eth0
root@pc1:/# ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.257 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.243 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.342 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.151 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.159 ms
```

Figure 3.8: Sending ping from PC-1 to PC-3 to check connectivity



```
reemeslam — root@pc2: / — kathara ◂ kathara connect -l pc2 — 80×24
root@pc2:/# ip addr add 10.0.0.2/12 dev eth0
root@pc2:/# ping 10.0.0.1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.927 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.163 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.153 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.151 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.365 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.273 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.238 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.209 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=64 time=0.228 ms
64 bytes from 10.0.0.1: icmp_seq=10 ttl=64 time=0.356 ms
```

Figure 3.9: Sending ping from PC-2 to PC-1 to check connectivity



```
reemeslam — root@pc3: / — kathara ◂ kathara connect -l pc3 — 80×24
root@pc3:/# ip addr add 10.0.0.3/12 dev eth0
root@pc3:/# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.23 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.138 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.158 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.137 ms
```

Figure 3.10: Sending ping from PC-3 to PC-2 to check connectivity

3.3 Checking connectivity by Capturing the ICMP messages exchanged by devices on sending ping from Pc-1 to PC-3.

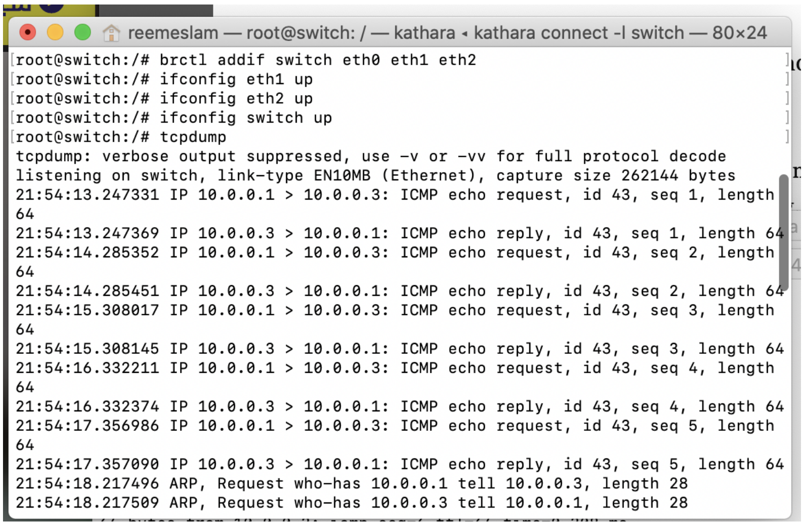


Figure 3.11: Capturing the ICMP messages between PC-1 and PC-3

3.4 .Conf and .startup files

For bigger complex networks using just terminals will not be the best approach, also each time you want to re-run the network you will need to configure everything again in the terminal. Better approach is to configure the environment in files and then run this files in the terminal.

3.4.1 .Conf file

The lab.conf will help us to setup the base infrastructure when the lab is initialized. The syntax is direct and easy as we can see in 3.12

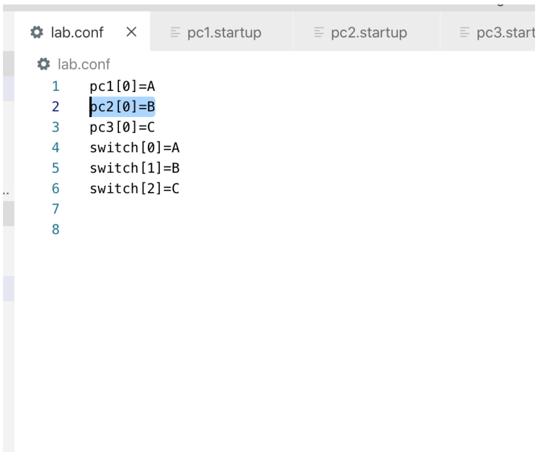


Figure 3.12: The .Conf file for the network

3.4.2 .Startup

In .startup file we add all of the commands we want to run in each device like, configuring IP address, configuring the bridge in switch.. etc. As you can see in 3.13 the commands are the same as used in the terminal.

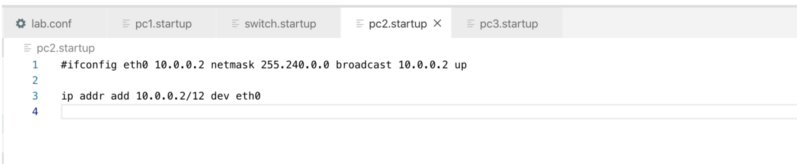


Figure 3.13: .Startup file for PC-2 device

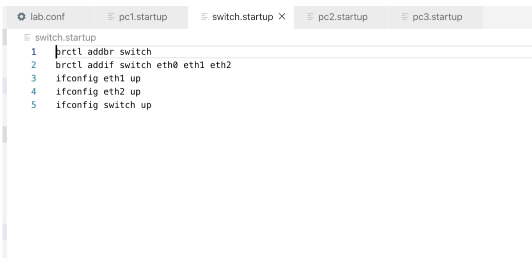


Figure 3.14: .Startup file for Switch device

After saving the files open the terminal in the same directory that has the files and run ”Kathara lstart”, The devices will open. Start testing the network by sending pings and capturing ICMP messages.

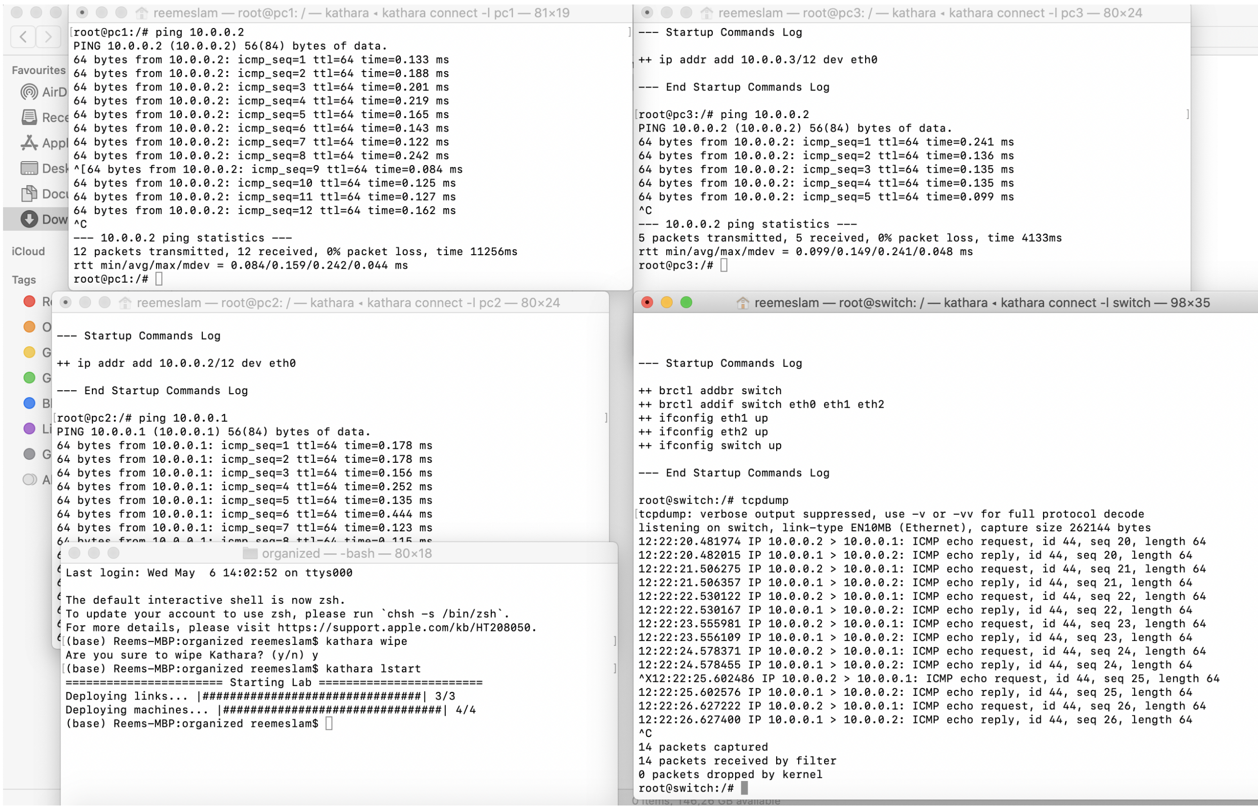


Figure 3.15: Overview for the whole network