

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 Eslam Mohamed Mekky 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 analyzer. Using wireshark, we can monitor what's happening 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 internals
<https://www.overleaf.com/project/5eacf93de71a280001bf5462>

Features

- 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.

No.	Time	Source	Destination	Protocol	Length	Info
61	5.584473	192.168.0.12	184.168.47.225	HTTP	705	GET /2015/01/21/calculate-http-response-time-in-wireshark/ HTTP/1.1
74	5.777487	184.168.47.225	192.168.0.12	HTTP	1315	HTTP/1.1 200 OK (text/html)
92	6.017920	2a02:810d:82bf:dc60::2a04:fa87:ffff::c00::	2a04:fa87:ffff::c00::	HTTP	613	GET /avatar/d30536da380a1a9cc4053a25783261ea?s=50&d=mm&r=g HTTP/1.1
94	6.038248	2a04:fa87:ffff::c00::	2a02:810d:82bf:dc60::	HTTP	561	HTTP/1.1 304 Not Modified
129	6.399352	192.168.0.12	184.168.47.225	HTTP	636	GET /favicon.ico HTTP/1.1
130	6.585002	184.168.47.225	192.168.0.12	HTTP	509	HTTP/1.1 302 Moved Temporarily (text/html) (text/html)
131	6.181099	192.168.0.12	184.168.47.225	HTTP	656	GET /wp-admin/images/w-logo-blue.png HTTP/1.1
140	6.282353	192.168.0.12	184.168.47.225	HTTP	460	HTTP/1.1 200 OK (PNG)
147	6.282353	192.168.0.12	184.168.47.225	HTTP	879	GET /2015/01/21/calculate-http-response-time-in-wireshark/ HTTP/1.1
159	4.472561	184.168.47.225	192.168.0.12	HTTP	1315	HTTP/1.1 200 OK (text/html)

Frame 131: 656 bytes on wire (5248 bits), 656 bytes captured (5248 bits) on interface 0
 Ethernet II, Src: HonHaiPr_61:db:37 (78:e4:00:61:db:37), Dst: HitronTe_76:ee:72 (f0:f2:49:76:ee:72)
 Internet Protocol Version 4, Src: 192.168.0.12, Dst: 184.168.47.225
 Transmission Control Protocol, Src Port: 58362, Dst Port: 80, Seq: 1234, Ack: 11209, Len: 602

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

No.	Time	Source	Destination	Protocol	Length	Info
20	2.136414	8.8.8.8	192.168.0.12	DNS	121	Standard query response 0x5eb8 A www.facebook.com CNAME star-mini.c10r.facebook.com A 185.60.216.35
21	2.136734	192.168.0.12	8.8.8.8	DNS	76	Standard query 0x78e6 AAAA www.facebook.com
23	2.166582	8.8.8.8	192.168.0.12	DNS	133	Standard query response 0x78e6 AAAA www.facebook.com CNAME star-mini.c10r.facebook.com AAAA 2a03:288...
37	2.776259	192.168.0.12	8.8.8.8	DNS	70	Standard query 0x4662 A www.fau.de
38	2.801471	8.8.8.8	192.168.0.12	DNS	86	Standard query response 0x4662 A www.fau.de A 131.188.16.209
39	2.801763	192.168.0.12	8.8.8.8	DNS	70	Standard query 0x22f8 AAAA www.fau.de
41	2.841413	8.8.8.8	192.168.0.12	DNS	98	Standard query response 0x22f8 AAAA www.fau.de AAAA 2001:638:0000:1000::209
47	3.298654	192.168.0.12	8.8.8.8	DNS	86	Standard query 0xa077 A encrypted-tbn0.gstatic.com
48	3.321436	8.8.8.8	192.168.0.12	DNS	102	Standard query response 0xa077 A encrypted-tbn0.gstatic.com A 172.217.20.238
49	3.321759	192.168.0.12	8.8.8.8	DNS	86	Standard query 0xc4b8 AAAA encrypted-tbn0.gstatic.com
50	3.351280	8.8.8.8	192.168.0.12	DNS	114	Standard query response 0xc4b8 AAAA encrypted-tbn0.gstatic.com AAAA 2a00:1450:4016:801::200e
51	3.351462	192.168.0.12	8.8.8.8	DNS	86	Standard query 0xc4b8 AAAA encrypted-tbn0.gstatic.com

[Frame is Ignored: False]
 [Protocols in frame: ethertype:ip:udp:dns]
 [Coloring Rule Name: UDP]
 [Coloring Rule Strings: udp]
 Ethernet II, Src: HitronTe_76:ee:72 (f0:f2:49:76:ee:72), Dst: HonHaiPr_61:db:37 (78:e4:00:61:db:37)
 Internet Protocol Version 4, Src: 8.8.8.8, Dst: 192.168.0.12
 0100 = Version: 4
 0101 = Header Length: 20 bytes (5)
 Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 0000 00.. = Differentiated Services Codepoint: Default (0)
 0000 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
 Total Length: 72
 Identification: 0xaf23 (44835)
 0000 78 e4 00 61 db 37 f0 f2 49 76 ee 72 00 00 05 00 x-a-7- Iv-n-
 0010 00 40 af 23 00 00 78 11 c2 bd 08 08 08 08 c0 a8 H# x
 0020 00 0c 00 35 c1 51 00 34 00 ce 48 62 81 80 00 01 :5 Q 4 -Hb-...

Figure 1.2: fau.de IP Address

Figure 1.3 shows a Wireshark packet capture of DNS traffic. The packet list on the left shows several DNS queries and responses. The packet details pane on the right shows the structure of a DNS response. A red circle highlights the IP address 81.91.170.12, which is the IP address of denic.de.

No.	Time	Source	Destination	Protocol	Length	Info
39	2.881855	8.8.8.8	192.168.0.12	DNS	114	Standard query response 0xf2d AAAA encrypted-tbn0.gstatic.com AAAA 2a00:1450:4016:802::200e
129	4.892384	192.168.0.12	8.8.8.8	DNS	72	Standard query 0x5bc7 A www.denic.de
130	4.920148	8.8.8.8	192.168.0.12	DNS	88	Standard query response 0x5b A www.denic.de A 81.91.170.12
131	4.920521	192.168.0.12	8.8.8.8	DNS	72	Standard query 0x5b00 AAAA www.denic.de
135	4.950135	8.8.8.8	192.168.0.12	DNS	116	Standard query response 0x5bb0 AAAA www.denic.de SOA nsl.denic.de
190	7.777005	192.168.0.12	8.8.8.8	DNS	75	Standard query 0xe12d A play.google.com
191	7.800333	8.8.8.8	192.168.0.12	DNS	91	Standard query response 0xe12d A play.google.com A 216.58.207.174
192	7.800669	192.168.0.12	8.8.8.8	DNS	75	Standard query 0xf1ad AAAA play.google.com
193	7.825333	8.8.8.8	192.168.0.12	DNS	103	Standard query response 0xf1ad AAAA play.google.com AAAA 2a00:1450:4016:803::200e
194	7.825425	192.168.0.12	8.8.8.8	DNS	75	Standard query 0xf1ad AAAA play.google.com
195	7.855404	192.168.0.12	8.8.8.8	DNS	75	Standard query 0xf1ad AAAA play.google.com
196	7.864366	8.8.8.8	192.168.0.12	DNS	103	Standard query response 0xf1ad AAAA play.google.com AAAA 2a00:1450:4016:803::200e

Time delta from previous displayed frame: 0.027764000 seconds
 Time since reference or first frame: 4.920148000 seconds
 Frame Number: 130
 Frame Length: 88 bytes (704 bits)
 Capture Length: 88 bytes (704 bits)
 [Frame is marked: False]
 [Frame is ignored: False]
 Protocols in frame: eth:ethertv0e:1a:udodns1

Figure 1.3: denic.de IP Address

Chapter 2

Wireshark Packet Filtering

2.1 Ping tool

- Ping tool is utility used for troubleshooting, testing and 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

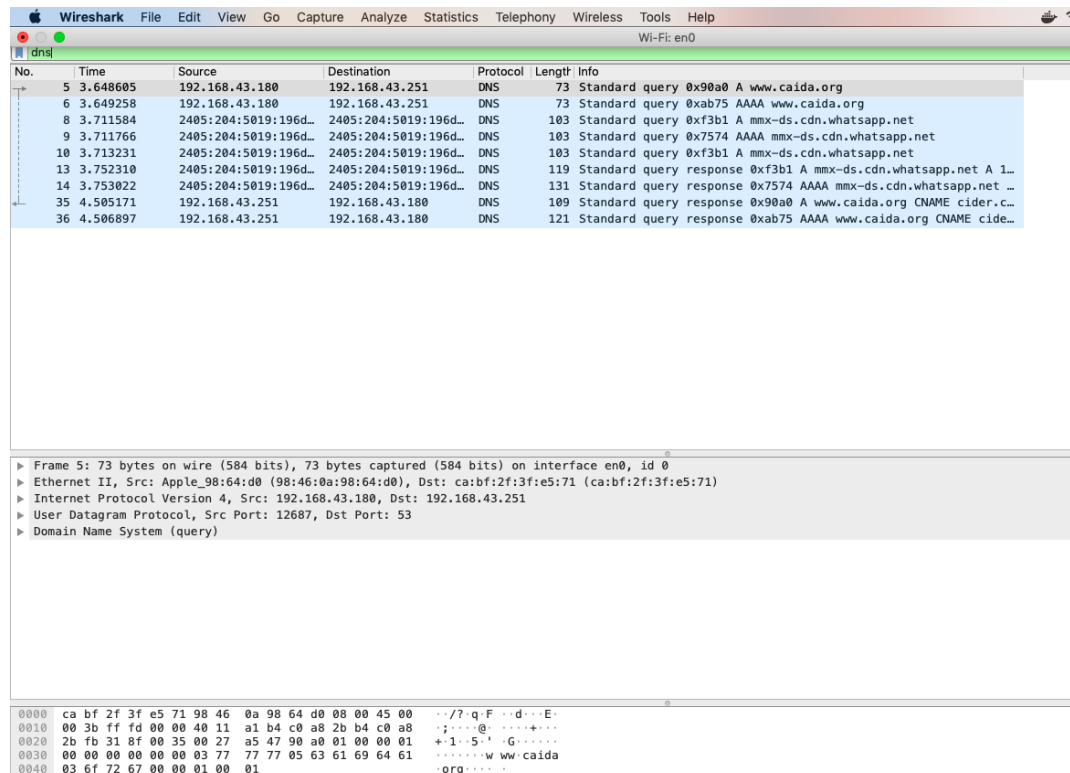
- Using wireshark packet filter `ip.addr == host/destination address`.
eg:- `ip.addr == 172.217.163.132`

ip.addr == 172.217.163.100							
No.	Time	Source	Destination	Protocol	Length	Info	
5118	43.474709	172.20.10.2	172.217.163.100	ICMP	98	Echo (ping) request	id=0xa304, seq=0/0, ttl=64 (reply in 51...
5126	43.583242	172.217.163.100	172.20.10.2	ICMP	98	Echo (ping) reply	id=0xa304, seq=0/0, ttl=52 (request in ...
5189	44.479044	172.20.10.2	172.217.163.100	ICMP	98	Echo (ping) request	id=0xa304, seq=1/256, ttl=64 (reply in ...
5196	44.559059	172.217.163.100	172.20.10.2	ICMP	98	Echo (ping) reply	id=0xa304, seq=1/256, ttl=52 (request in ...
5243	45.481205	172.20.10.2	172.217.163.100	ICMP	98	Echo (ping) request	id=0xa304, seq=2/512, ttl=64 (reply in ...
5247	45.559178	172.217.163.100	172.20.10.2	ICMP	98	Echo (ping) reply	id=0xa304, seq=2/512, ttl=52 (request in ...
5399	46.486006	172.20.10.2	172.217.163.100	ICMP	98	Echo (ping) request	id=0xa304, seq=3/768, ttl=64 (reply in ...
▶ Frame 5118: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface en0, id 0 ▶ Ethernet II, Src: Apple_9a:6a:10 (98:46:0a:9a:6a:10), Dst: 46:00:10:16:40:64 (46:00:10:16:40:64) ▶ Internet Protocol Version 4, Src: 172.20.10.2, Dst: 172.217.163.100 ▶ Internet Control Message Protocol							
0000	46 00 10 16 40 64 98 46	0a 9a 6a 10 08 00 45 00	F...@.F...j...E.				
0010	00 54 b6 81 00 00 40 01	bd d3 ac 14 0a 02 ac d9	.T...@.				
0020	a3 64 00 00 6b 18 a3 04	00 00 5c b0 0f 9f 00 01	.d...k...:A....				
0030	90 8f 00 00 0a 0b 0c 0d	0e 0f 10 11 12 13 14 15				
0040	15 17 18 19 1a 1b 1c 1d	1e 1f 20 21 22 23 24 25 !"#%&				
0050	26 27 28 29 2a 2b 2c 2d	2e 2f 30 31 32 33 34 35	&'()*+,-./012345				
0060	36 37		67				

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.dst == ip address of the url`.
eg:- `ip.addr == 192.168.43.82`.



No.	Time	Source	Destination	Protocol	Length	Info
5	3.648605	192.168.43.180	192.168.43.251	DNS	73	Standard query 0x90a0 A www.caida.org
6	3.649258	192.168.43.180	192.168.43.251	DNS	73	Standard query 0xab75 AAAA www.caida.org
8	3.711584	2405:204:5019:196d...	2405:204:5019:196d...	DNS	103	Standard query 0xf3b1 A mmx-ds.cdn.whatsapp.net
9	3.711766	2405:204:5019:196d...	2405:204:5019:196d...	DNS	103	Standard query 0x7574 AAAA mmx-ds.cdn.whatsapp.net
10	3.713231	2405:204:5019:196d...	2405:204:5019:196d...	DNS	103	Standard query 0xf3b1 A mmx-ds.cdn.whatsapp.net
13	3.752310	2405:204:5019:196d...	2405:204:5019:196d...	DNS	119	Standard query response 0xf3b1 A mmx-ds.cdn.whatsapp.net A 1...
14	3.753022	2405:204:5019:196d...	2405:204:5019:196d...	DNS	131	Standard query response 0x7574 AAAA mmx-ds.cdn.whatsapp.net ...
35	4.505171	192.168.43.251	192.168.43.180	DNS	109	Standard query response 0x90a0 A www.caida.org CNAME cider.c...
36	4.506897	192.168.43.251	192.168.43.180	DNS	121	Standard query response 0xab75 AAAA www.caida.org CNAME cide...

Frame 5: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface en0, id 0
Ethernet II, Src: Apple_98:64:d0 (98:64:0a:98:64:d0), Dst: ca:bf:2f:3f:e5:71 (ca:bf:2f:3f:e5:71)
Internet Protocol Version 4, Src: 192.168.43.180, Dst: 192.168.43.251
User Datagram Protocol, Src Port: 12687, Dst Port: 53
Domain Name System (query)

0000 ca bf 2f 3f e5 71 98 46 0a 98 64 d0 00 00 45 00 ../?qF...d...E.
0010 00 3b ff fd 00 00 40 11 a1 b4 c0 a8 2b b4 c0 a8 ;...@...+...
0020 2b fb 31 8f 00 35 00 27 a5 47 90 a0 01 00 00 01 +1..5..G.....
0030 00 00 00 00 00 03 77 77 77 05 63 61 69 64 61w ww caida
0040 03 6f 72 67 00 00 01 00 01org.....

Figure 2.3: Destination ip Address with DNS filter

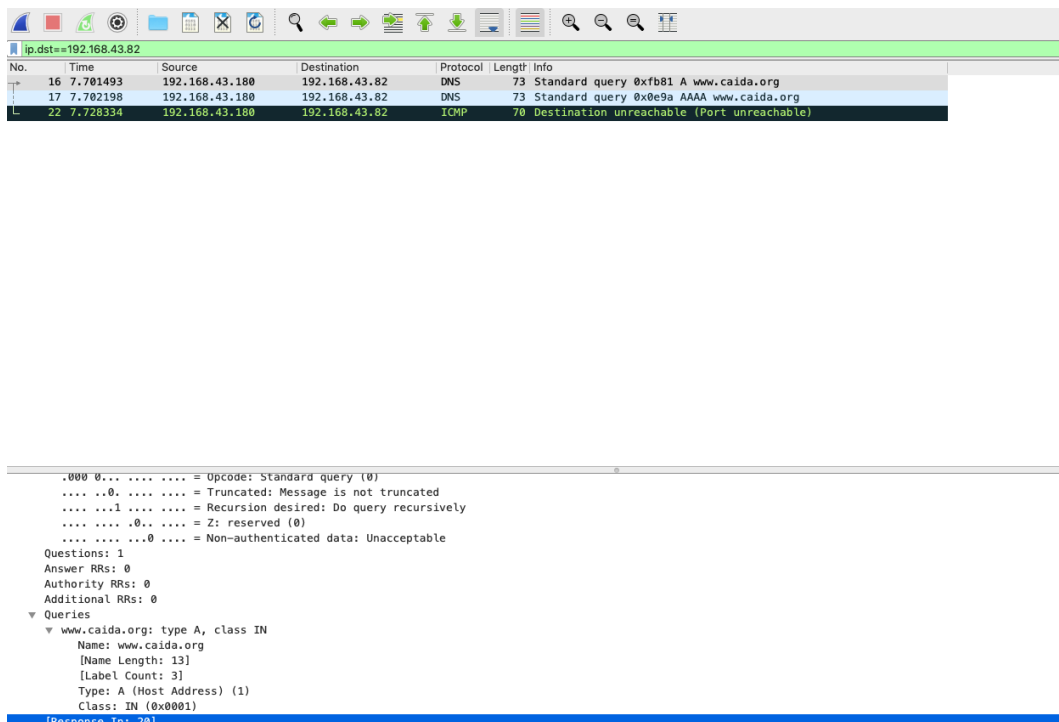


Figure 2.4: IP address display filter

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`

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.43.180	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
2	1.000813	192.168.43.180	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
3	2.001128	192.168.43.180	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
4	3.001251	192.168.43.180	239.255.255.250	SSDP	217	M-SEARCH * HTTP/1.1
5	4.681125	192.168.43.180	17.134.127.250	TCP	78	49365 → 443 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MSS=1460 W...
6	5.041818	17.134.127.250	192.168.43.180	TCP	74	443 → 49365 [SYN, ACK, ECN] Seq=0 Ack=1 Win=43440 Len=0 MSS=...
7	5.041951	192.168.43.180	17.134.127.250	TCP	66	49365 → 443 [ACK] Seq=1 Ack=1 Win=131712 Len=0 TSval=6470375...
8	5.044981	192.168.43.180	17.134.127.250	TLSv1...	583	Client Hello
9	5.210257	17.134.127.250	192.168.43.180	TCP	66	443 → 49365 [ACK] Seq=1 Ack=518 Win=45056 Len=0 TSval=337701...
10	5.867053	17.134.127.250	192.168.43.180	TLSv1...	1396	Server Hello
11	5.867060	17.134.127.250	192.168.43.180	TCP	1424	443 → 49365 [ACK] Seq=1331 Ack=518 Win=45056 Len=1350 TSval=...
12	5.867150	192.168.43.180	17.134.127.250	TCP	66	49365 → 443 [ACK] Seq=518 Ack=1331 Win=130368 Len=0 TSval=64...
13	5.872963	17.134.127.250	192.168.43.180	TLSv1...	305	Certificate, Server Key Exchange, Server Hello Done
14	5.873057	192.168.43.180	17.134.127.250	TCP	66	49365 → 443 [ACK] Seq=518 Ack=2928 Win=130816 Len=0 TSval=64...
15	5.878791	17.134.127.250	192.168.43.180	TCP	305	[TCP Spurious Retransmission] 443 → 49365 [PSH, ACK] Seq=268...
16	5.878873	192.168.43.180	17.134.127.250	TCP	78	[TCP Window Update] 49365 → 443 [ACK] Seq=518 Ack=2928 Win=1...
17	5.885500	192.168.43.180	17.134.127.250	TLSv1...	159	Client Key Exchange, Change Cipher Spec, Encrypted Handshake...
18	6.176722	17.134.127.250	192.168.43.180	TCP	66	443 → 49365 [ACK] Seq=2928 Ack=611 Win=45056 Len=0 TSval=337...
19	6.346651	17.134.127.250	192.168.43.180	TLSv1...	117	Change Cipher Spec, Encrypted Handshake Message
20	6.353008	192.168.43.180	17.134.127.250	TCP	66	49365 → 443 [ACK] Seq=611 Ack=2979 Win=131008 Len=0 TSval=64...
21	6.353009	192.168.43.180	17.134.127.250	TLSv1...	384	Application Data
22	6.353010	192.168.43.180	17.134.127.250	TLSv1...	198	Application Data
23	6.639609	17.134.127.250	192.168.43.180	TCP	66	443 → 49365 [ACK] Seq=2979 Ack=929 Win=49152 Len=0 TSval=337...
24	6.639614	17.134.127.250	192.168.43.180	TCP	66	443 → 49365 [ACK] Seq=2979 Ack=1061 Win=49152 Len=0 TSval=33...
25	7.146670	17.134.127.250	192.168.43.180	TLSv1...	306	Application Data
26	7.146780	192.168.43.180	17.134.127.250	TCP	66	49365 → 443 [ACK] Seq=1061 Ack=3210 Win=130816 Len=0 TSval=6...

▶ Frame 1: 217 bytes on wire (1736 bits), 217 bytes captured (1736 bits) on interface en0, id 0
 ▼ Ethernet II, Src: Apple_98:64:d0 (98:46:0a:98:64:d0), Dst: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
 ▶ Destination: IPv4mcast_7f:ff:fa (01:00:5e:7f:ff:fa)
 ▶ Source: Apple_98:64:d0 (98:46:0a:98:64:d0)
 Type: IPv4 (0x0008)
 ▶ Internet Protocol Version 4, Src: 192.168.43.180, Dst: 239.255.255.250
 ▶ User Datagram Protocol, Src Port: 55646, Dst Port: 1900
 ▼ Simple Service Discovery Protocol
 ▶ M-SEARCH * HTTP/1.1\r\n
 HOST: 239.255.255.250:1900\r\n
 MAN: "ssdp:discover"\r\n
 MX: 1\r\n
 ST: urn:dial-multiscreen-org:service:dial:1\r\n
 USER-AGENT: Google Chrome/81.0.4044.129 Mac OS X\r\n
 \r\n
 [Full request URI: http://239.255.255.250:1900/]
 [HTTP request 1/4]

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
2	0.000006	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
4	0.006040	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
5	0.014984	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
6	0.014988	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
7	0.020921	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
8	0.020927	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
9	0.028928	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
10	0.034998	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
11	0.035004	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
12	0.035005	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
14	0.043346	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
15	0.043352	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
18	0.057171	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
19	0.057178	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
20	0.057179	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
23	0.069139	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
24	0.069147	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
25	0.069149	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
28	0.083647	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
29	0.083656	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
30	0.083657	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
31	0.097182	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
32	0.097190	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record
33	0.103407	13.33.151.14	192.168.43.180	TLSv1...	1424	Ignored Unknown Record

▶ Frame 1: 1424 bytes on wire (11392 bits), 1424 bytes captured (11392 bits) on interface en0, id 0
 ▶ Ethernet II, Src: d6:18:2a:86:c8:0f (d6:18:2a:86:c8:0f), Dst: Apple_98:64:d0 (98:46:0a:98:64:d0)
 ▼ Internet Protocol Version 4, Src: 13.33.151.14, Dst: 192.168.43.180
 0100 ... = Version: 4
 ... 0101 = Header Length: 20 bytes (5)
 ▶ Differentiated Services Field: 0x28 (DSCP: AF11, ECN: Not-ECT)
 Total Length: 1410
 Identification: 0xfce3 (64739)
 ▶ Flags: 0x4000, Don't fragment
 Fragment offset: 0
 Time to live: 62
 Protocol: TCP (6)
 Header checksum: 0xa9de [validation disabled]
 [Header checksum status: Unverified]
 Source: 13.33.151.14
 Destination: 192.168.43.180

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 10.0.0.3 and host 10.0.0.12

2.9 TCP Packet Capture

- Tcpcap 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.

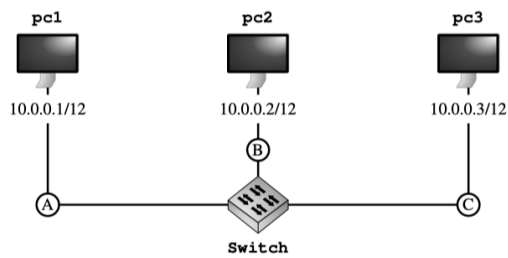


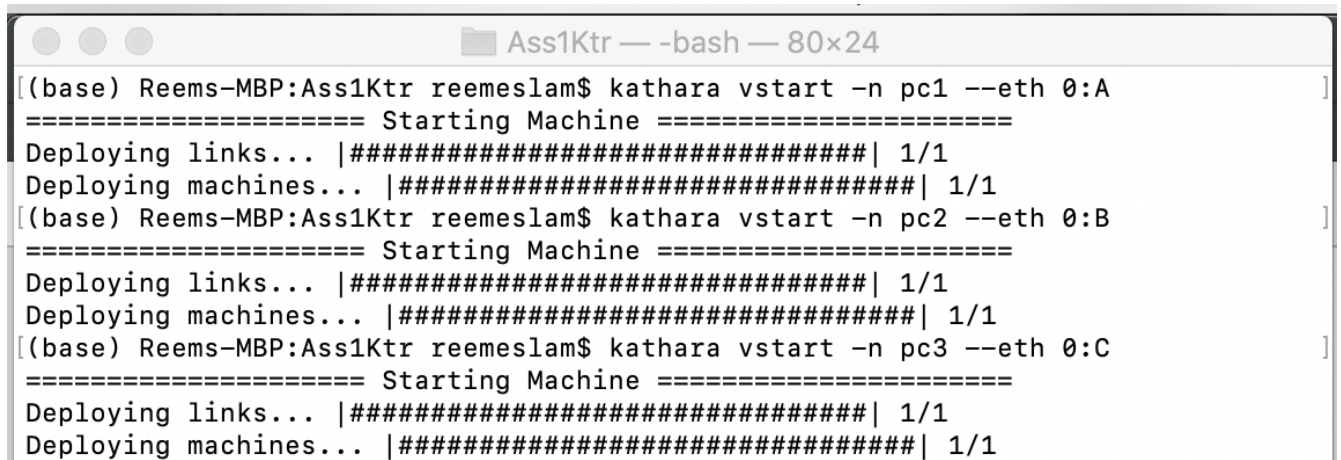
Figure 1: LAN setup

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

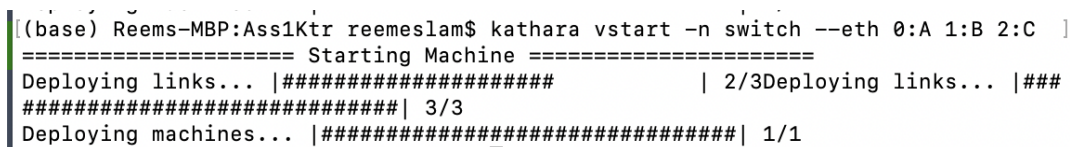
3.1.1 Creating Three PCs each assigned a network interface and unique collision domain



```
Ass1Ktr — -bash — 80x24
[(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 PC3

A terminal window titled 'reemeslam — root@pc1: / — kathara • kathara connect -l pc1 — 80x24'. The command prompt is '[root@pc1:/# ip addr add 10.0.0.1/12 dev eth0]'. The IP address '10.0.0.1' is highlighted in blue.

```
reemeslam — root@pc1: / — kathara • kathara connect -l pc1 — 80x24
[root@pc1:/# ip addr add 10.0.0.1/12 dev eth0
```

Figure 3.4: configured IP address for PC1

A terminal window titled 'reemeslam — root@pc2: / — kathara • kathara connect -l pc2 — 80x24'. The command prompt is '[root@pc2:/# ip addr add 10.0.0.2/12 dev eth0]'.

```
reemeslam — root@pc2: / — kathara • kathara connect -l pc2 — 80x24
[root@pc2:/# ip addr add 10.0.0.2/12 dev eth0
```

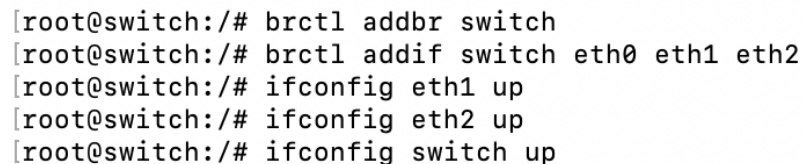
Figure 3.5: configured IP address for PC2

A terminal window titled 'reemeslam — root@pc3: / — kathara • kathara connect -l pc3 — 80x24'. The command prompt is '[root@pc3:/# ip addr add 10.0.0.3/12 dev eth0]'.

```
reemeslam — root@pc3: / — kathara • kathara connect -l pc3 — 80x24
[root@pc3:/# ip addr add 10.0.0.3/12 dev eth0
```

Figure 3.6: configured IP address for PC3

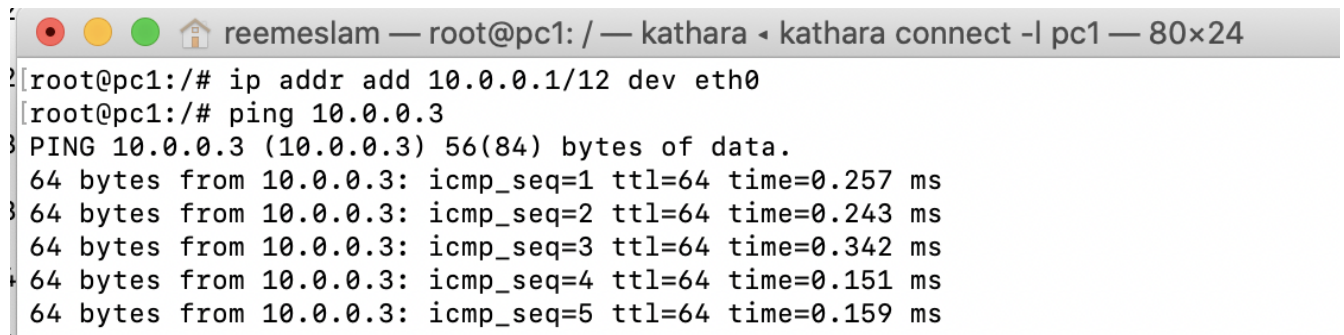
3.2.2 Configuring a bridge for the switch, set-up the environment and turning on the bridge

A terminal window showing the configuration of a bridge named 'switch' on a switch device. The commands are: 'brctl addbr switch', 'brctl addif switch eth0 eth1 eth2', 'ifconfig eth1 up', 'ifconfig eth2 up', and 'ifconfig switch up'.

```
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
```

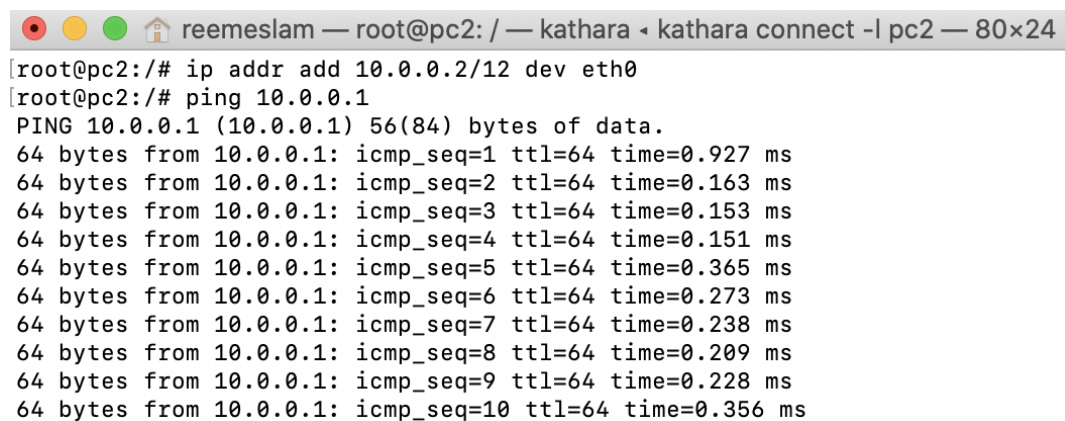
Figure 3.7: Created a bridge with name switch, Added the three interfaces to the switch, turned on eth1 & eth2 and Turned the bridge on

3.2.3 Testing the network by sending pings between PCs



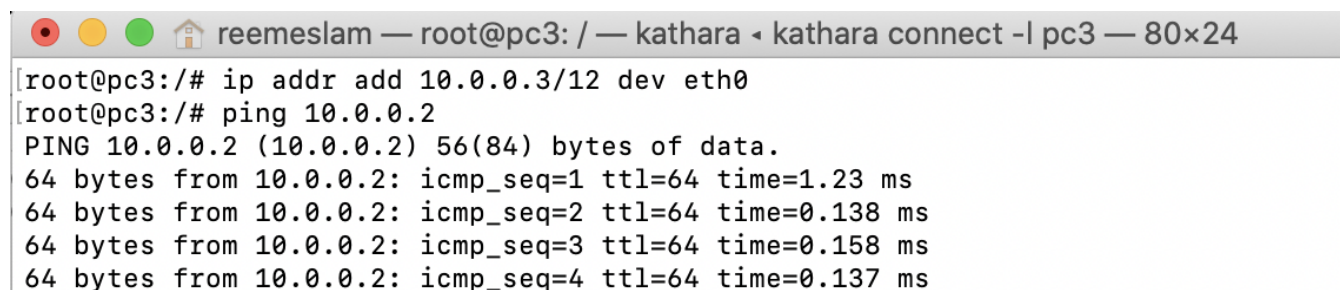
```
reemeslam — root@pc1: / — kathara ◀ kathara connect -l pc1 — 80×24
[root@pc1:/# ip addr add 10.0.0.1/12 dev eth0
[root@pc1:/# ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
 64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.257 ms
 64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.243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

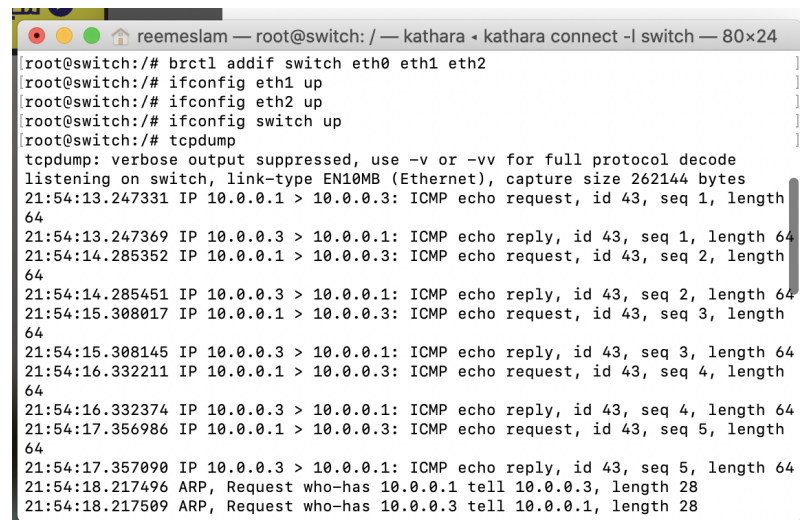
Figure 3.9: Sending ping from PC-2 to PC-1 to check connectivity



```
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.



```
reemeslam — root@switch: / — kathara • kathara connect -l switch — 80x24
root@switch:~# brctl addif switch eth0 eth1 eth2
root@switch:~# ifconfig eth1 up
root@switch:~# ifconfig eth2 up
root@switch:~# ifconfig switch up
root@switch:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on switch, link-type EN10MB (Ethernet), capture size 262144 bytes
21:54:13.247331 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 43, seq 1, length 64
21:54:13.247369 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 43, seq 1, length 64
21:54:14.285352 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 43, seq 2, length 64
21:54:14.285451 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 43, seq 2, length 64
21:54:15.308017 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 43, seq 3, length 64
21:54:15.308145 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 43, seq 3, length 64
21:54:16.332211 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 43, seq 4, length 64
21:54:16.332374 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 43, seq 4, length 64
21:54:17.356986 IP 10.0.0.1 > 10.0.0.3: ICMP echo request, id 43, seq 5, length 64
21:54:17.357090 IP 10.0.0.3 > 10.0.0.1: ICMP echo reply, id 43, seq 5, length 64
21:54:18.217496 ARP, Request who-has 10.0.0.1 tell 10.0.0.3, length 28
21:54:18.217509 ARP, Request who-has 10.0.0.3 tell 10.0.0.1, length 28
```

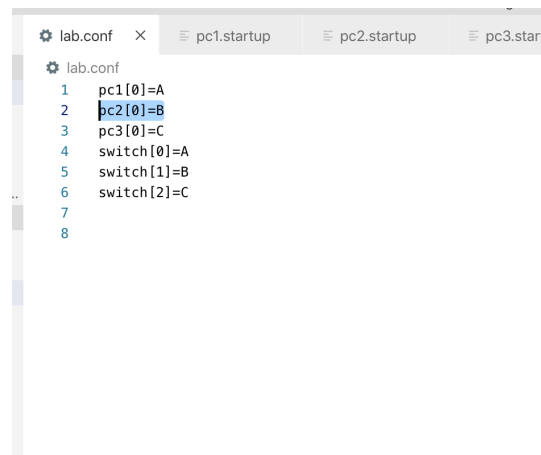
Figure 3.11: Capturing the ICMP messages between PC-1 and PC-3

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

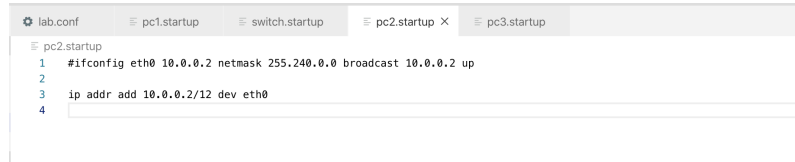


```
lab.conf
1 pc1[0]=A
2 pc2[0]=B
3 pc3[0]=C
4 switch[0]=A
5 switch[1]=B
6 switch[2]=C
7
8
```

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 we used in the terminal.



```
pc2.startup
1 #ifconfig eth0 10.0.0.2 netmask 255.240.0.0 broadcast 10.0.0.2 up
2
3 ip addr add 10.0.0.2/12 dev eth0
4
```

Figure 3.13: .Startup file for PC-2 device



```
switch.startup
1 brctl addbr switch
2 brctl addif switch eth0 eth1 eth2
3 ifconfig eth1 up
4 ifconfig eth2 up
5 ifconfig switch up
```

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.

```

root@pc1:/# 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=0.133 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.188 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.201 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.219 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.165 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.143 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.122 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.242 ms
^C
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.084 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.125 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.127 ms
64 bytes from 10.0.0.2: icmp_seq=12 ttl=64 time=0.162 ms
^C
--- 10.0.0.2 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11256ms
rtt min/avg/max/mdev = 0.084/0.159/0.242/0.044 ms
root@pc1:/#

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=0.178 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.178 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.156 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=64 time=0.252 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=64 time=0.135 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=64 time=0.444 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=64 time=0.123 ms
^C
64 bytes from 10.0.0.1: icmp_seq=8 ttl=64 time=0.114 ms
^C
root@pc2:/#

root@switch:/# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on switch, link-type EN10MB (Ethernet), capture size 262144 bytes
12:22:20.481974 IP 10.0.0.2 > 10.0.0.1: ICMP echo request, id 44, seq 20, length 64
12:22:20.482015 IP 10.0.0.1 > 10.0.0.2: ICMP echo reply, id 44, seq 20, length 64
12:22:21.506275 IP 10.0.0.2 > 10.0.0.1: ICMP echo request, id 44, seq 21, length 64
12:22:21.506357 IP 10.0.0.1 > 10.0.0.2: ICMP echo reply, id 44, seq 21, length 64
12:22:22.530122 IP 10.0.0.2 > 10.0.0.1: ICMP echo request, id 44, seq 22, length 64
12:22:22.530167 IP 10.0.0.1 > 10.0.0.2: ICMP echo reply, id 44, seq 22, length 64
12:22:23.555981 IP 10.0.0.2 > 10.0.0.1: ICMP echo request, id 44, seq 23, length 64
12:22:23.556109 IP 10.0.0.1 > 10.0.0.2: ICMP echo reply, id 44, seq 23, length 64
12:22:24.578371 IP 10.0.0.2 > 10.0.0.1: ICMP echo request, id 44, seq 24, length 64
12:22:24.578455 IP 10.0.0.1 > 10.0.0.2: ICMP echo reply, id 44, seq 24, length 64
^X12:22:25.602486 IP 10.0.0.2 > 10.0.0.1: ICMP echo request, id 44, seq 25, length 64
12:22:25.602576 IP 10.0.0.1 > 10.0.0.2: ICMP echo reply, id 44, seq 25, length 64
12:22:26.627222 IP 10.0.0.2 > 10.0.0.1: ICMP echo request, id 44, seq 26, length 64
12:22:26.627400 IP 10.0.0.1 > 10.0.0.2: ICMP echo reply, id 44, seq 26, length 64
^C
14 packets captured
14 packets received by filter
0 packets dropped by kernel
root@switch:/#
  
```

Figure 3.15: Overview for the whole network