Github Link - https://github.com/amansharma96/ser321-summer2023-C-famandee

Part 1: Linux, Setup

1. Linux Ubuntu

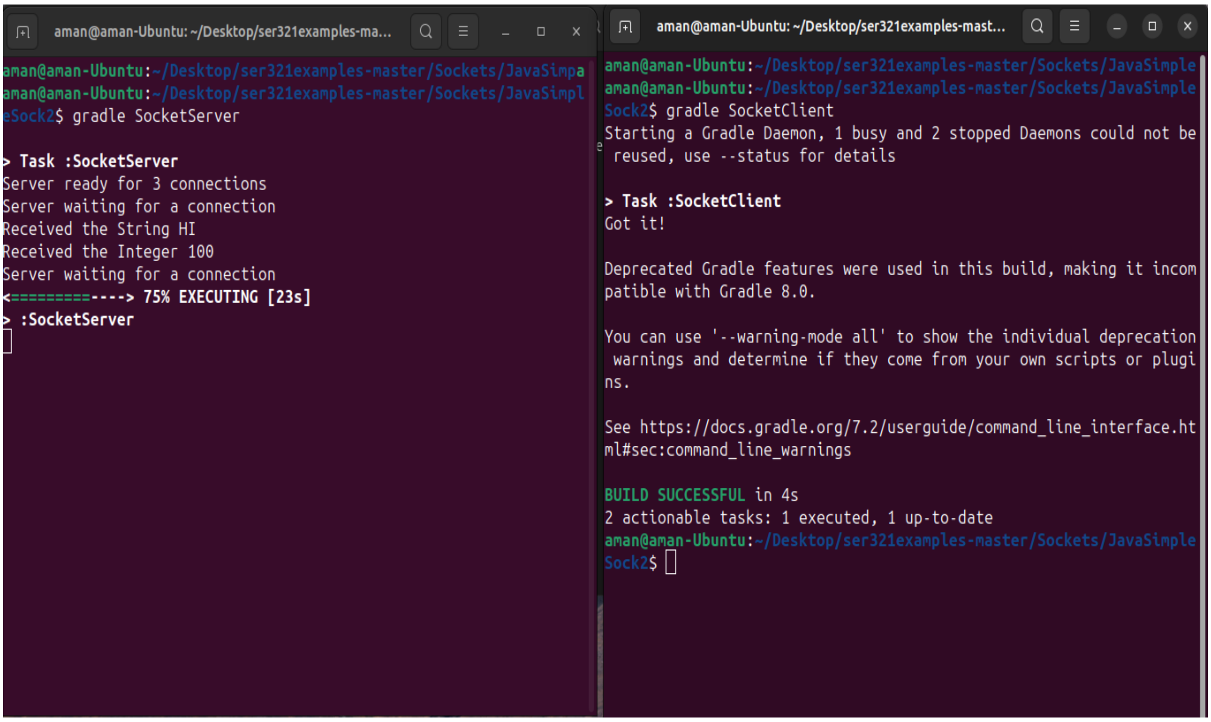
2. Command line tasks

    1. mkdir cli_assignment
    2. cd cli_assignment
    3. touch stuff.txt
    4. cat > stuff.txt
       My name is Aman Sharma and I am originally from India. I live in Seattle right now.
       I study in Arizona State University.
    5. wc -l stuff.txt
       wc -w stuff.txt
    6. cat >> stuff.txt
       This quarter I am taking 3 classes.
    7. mkdir draft
    8. mv stuff.txt draft
    9. cd draft
       touch .secret.txt
    10. cp -r draft final
    11. mv draft draft.remove
    12. mv draft.remove final
    13. ls -l -R
    14. gzip -cd NASA_access_log_Aug95.gz
    15. zmore NASA_access_log_Aug95.gz
    16. mv NASA_access_log_Aug95 logs.txt
    17. mv logs.txt cli_assignment
    18. head -100 logs.txt
    19. head -100 logs.txt > logs_top_100.txt
    20. tail -100 logs.txt
    21. tail -100 logs.txt > logs_bottom_100.txt
    22. cat logs_top_100.txt logs_bottom_100.txt > logs_snapshot.txt
    23. cat > logs_snapshot.txt
        Famandee - This is a great assignment 05/18/2023.
    24. less logs.txt
    25. awk -F '%' '{print $1}' marks.csv
    26. cut -d '%' -f 4 marks.csv
    27. awk -F '%' '{print $3}' marks.csv
    28. mv awk -F '%' '{print $3}' marks.csv > done.txt
    29. mv done.txt final
    30. mv done.txt average.txt

Part:3

1. Github Link - https://github.com/amansharma96/ser321-summer2023-C-famandee
2. Running examples -

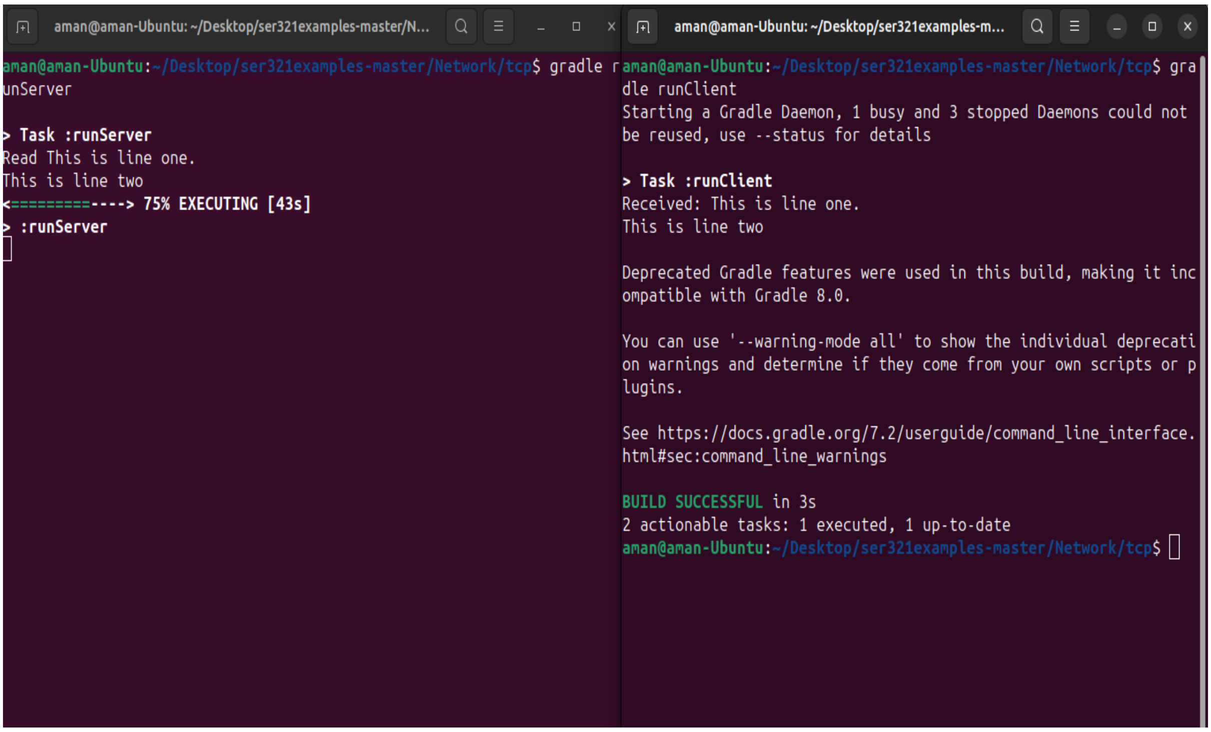
a. First we run Socket Server Client gradle file which make the connection between server and client.


aman@aman-Ubuntu: ~/Desktop/ser321examples-ma...
aman@aman-Ubuntu: ~/Desktop/ser321examples-mast...

```
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Sockets/JavaSimpa
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Sockets/JavaSimpl
eSock2$ gradle SocketServer

> Task :SocketServer
Server ready for 3 connections
Server waiting for a connection
Received the String HI
Received the Integer 100
Server waiting for a connection
<=========----> 75% EXECUTING [23s]
> :SocketServer
```

```
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Sockets/JavaSimple
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Sockets/JavaSimple
Sock2$ gradle SocketClient
Starting a Gradle Daemon, 1 busy and 2 stopped Daemons could not be
 reused, use --status for details

> Task :SocketClient
Got it!

Deprecated Gradle features were used in this build, making it incom
patible with Gradle 8.0.

You can use '--warning-mode all' to show the individual deprecation
 warnings and determine if they come from your own scripts or plugi
ns.

See https://docs.gradle.org/7.2/userguide/command_line_interface.ht
ml#sec:command_line_warnings

BUILD SUCCESSFUL in 4s
2 actionable tasks: 1 executed, 1 up-to-date
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Sockets/JavaSimple
Sock2$
```

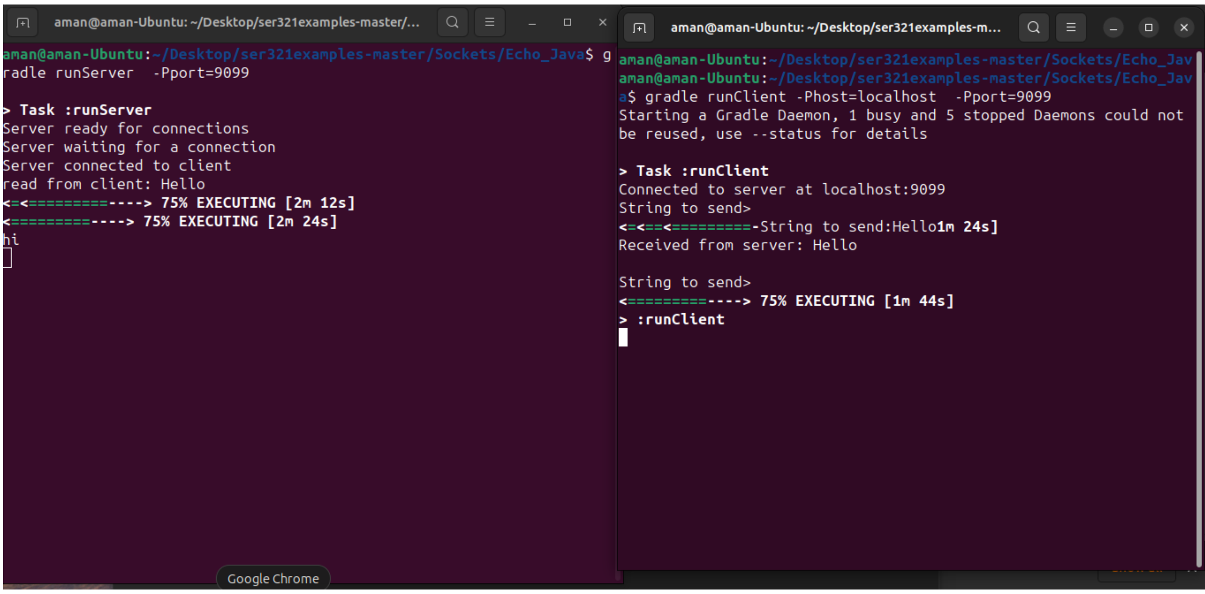
b. Second we run PEER to PEER gradle file which make connection to send messages.


aman@aman-Ubuntu: ~/Desktop/ser321examples-master/N...
aman@aman-Ubuntu: ~/Desktop/ser321examples-m...

```
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Network/tcp$ gradle r
unServer

> Task :runServer
Read This is line one.
This is line two
<=========----> 75% EXECUTING [43s]
> :runServer
```

```
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Network/tcp$ gra
dle runClient
Starting a Gradle Daemon, 1 busy and 3 stopped Daemons could not
be reused, use --status for details

> Task :runClient
Received: This is line one.
This is line two

Deprecated Gradle features were used in this build, making it inc
ompatible with Gradle 8.0.

You can use '--warning-mode all' to show the individual deprecati
on warnings and determine if they come from your own scripts or p
lugins.

See https://docs.gradle.org/7.2/userguide/command_line_interface.
html#sec:command_line_warnings

BUILD SUCCESSFUL in 3s
2 actionable tasks: 1 executed, 1 up-to-date
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Network/tcp$
```

c. Second we run PEER to PEER gradle file which make connection to send messages.


aman@aman-Ubuntu: ~/Desktop/ser321examples-master/...
aman@aman-Ubuntu: ~/Desktop/ser321examples-m...
Google Chrome

```
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Sockets/Echo_Java$ g
radle runServer  -Pport=9099

> Task :runServer
Server ready for connections
Server waiting for a connection
Server connected to client
read from client: Hello
<=<=========----> 75% EXECUTING [2m 12s]
<=========----> 75% EXECUTING [2m 24s]
hi
```

```
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Sockets/Echo_Jav
aman@aman-Ubuntu:~/Desktop/ser321examples-master/Sockets/Echo_Jav
a$ gradle runClient -Phost=localhost  -Pport=9099
Starting a Gradle Daemon, 1 busy and 5 stopped Daemons could not
be reused, use --status for details

> Task :runClient
Connected to server at localhost:9099
String to send>
<=<==<=========-String to send:Hello1m 24s]
Received from server: Hello

String to send>
<=========----> 75% EXECUTING [1m 44s]
> :runClient
```

4. Second System I used is LINUX Ubuntu.

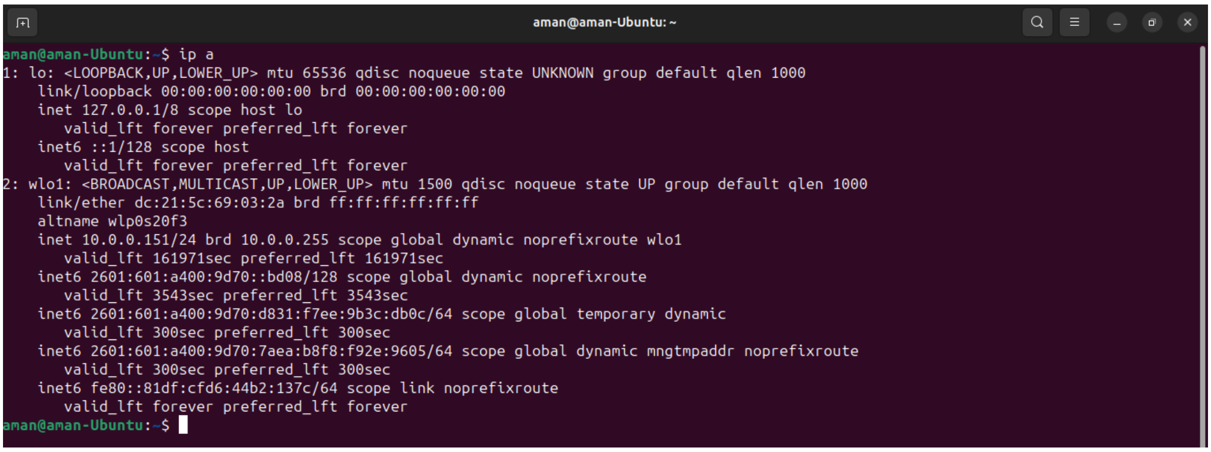
Task 3.4 : https://youtu.be/b97KUMNqr1s

4. Network traffic

4.1 Explore the Data Link Layer with ARP

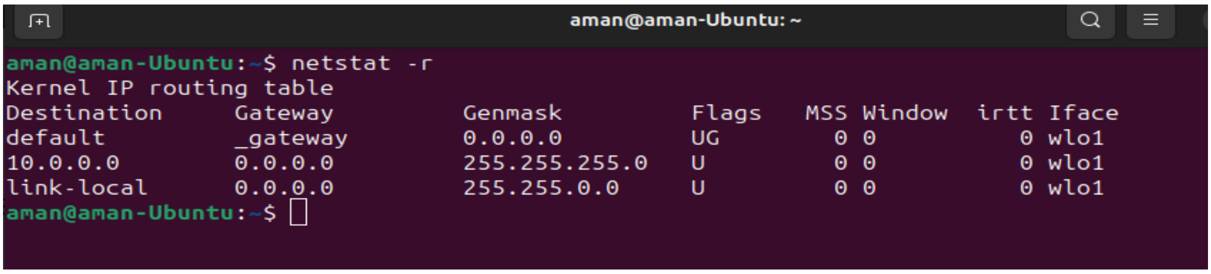
Step 1: Capture a Trace

1.



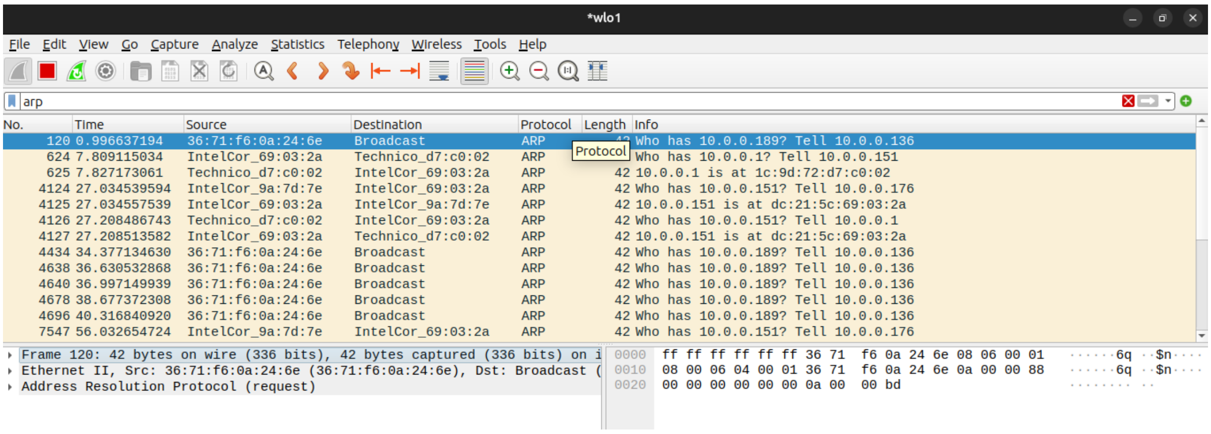
```
aman@aman-Ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: wlo1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether dc:21:5c:69:03:2a brd ff:ff:ff:ff:ff:ff
    altname wlp0s20f3
    inet 10.0.0.151/24 brd 10.0.0.255 scope global dynamic noprefixroute wlo1
       valid_lft 161971sec preferred_lft 161971sec
    inet6 2601:601:a400:9d70::bd08/128 scope global dynamic noprefixroute
       valid_lft 3543sec preferred_lft 3543sec
    inet6 2601:601:a400:9d70:d831:f7ee:9b3c:db0c/64 scope global temporary dynamic
       valid_lft 300sec preferred_lft 300sec
    inet6 2601:601:a400:9d70:7aea:b8f8:f92e:9605/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 300sec preferred_lft 300sec
    inet6 fe80::81df:cfd6:44b2:137c/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
aman@aman-Ubuntu:~$
```

2.



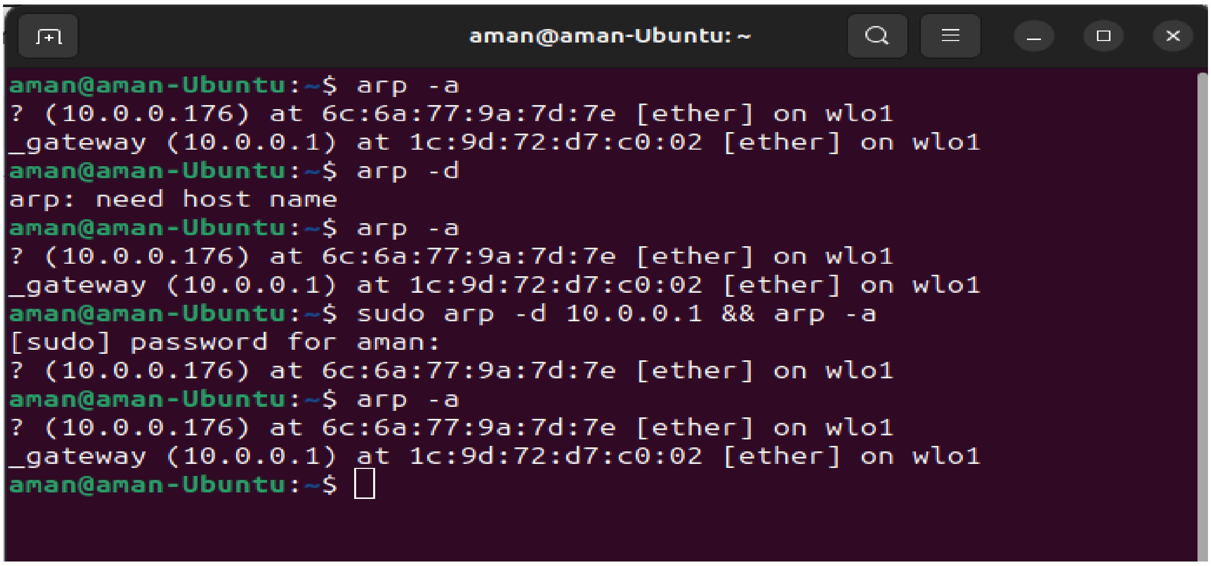
```
aman@aman-Ubuntu:~$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         _gateway        0.0.0.0         UG        0 0          0 wlo1
10.0.0.0        0.0.0.0         255.255.255.0   U         0 0          0 wlo1
link-local      0.0.0.0         255.255.0.0     U         0 0          0 wlo1
aman@aman-Ubuntu:~$
```

3.


*wlo1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Protocol

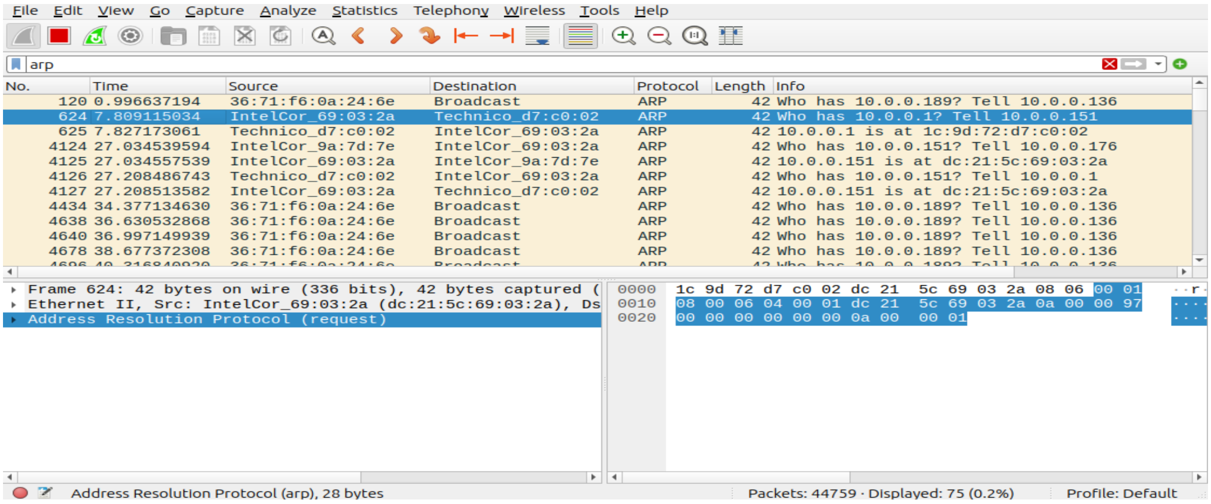
```
arp
No.   Time            Source              Destination         Protocol Length Info
  120 0.996637194     36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
  624 7.809115034     IntelCor_69:03:2a   Technico_d7:c0:02   ARP         Who has 10.0.0.1? Tell 10.0.0.151
  625 7.827173061     Technico_d7:c0:02   IntelCor_69:03:2a   ARP      42 10.0.0.1 is at 1c:9d:72:d7:c0:02
 4124 27.034539594    IntelCor_9a:7d:7e   IntelCor_69:03:2a   ARP      42 Who has 10.0.0.151? Tell 10.0.0.176
 4125 27.034557539    IntelCor_69:03:2a   IntelCor_9a:7d:7e   ARP      42 10.0.0.151 is at dc:21:5c:69:03:2a
 4126 27.208486743    Technico_d7:c0:02   IntelCor_69:03:2a   ARP      42 Who has 10.0.0.151? Tell 10.0.0.1
 4127 27.208513582    IntelCor_69:03:2a   Technico_d7:c0:02   ARP      42 10.0.0.151 is at dc:21:5c:69:03:2a
 4434 34.377134630    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
 4638 36.630532868    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
 4640 36.997149939    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
 4678 38.677372308    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
 4696 40.316840920    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
 7547 56.032654724    IntelCor_9a:7d:7e   IntelCor_69:03:2a   ARP      42 Who has 10.0.0.151? Tell 10.0.0.176

› Frame 120: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on i
› Ethernet II, Src: 36:71:f6:0a:24:6e (36:71:f6:0a:24:6e), Dst: Broadcast (
› Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 36 71  f6 0a 24 6e 08 06 00 01   ······6q ··$n····
0010  08 00 06 04 00 01 36 71  f6 0a 24 6e 0a 00 00 88   ······6q ··$n····
0020  00 00 00 00 00 00 0a 00  00 bd                     ········ ··
```

4.



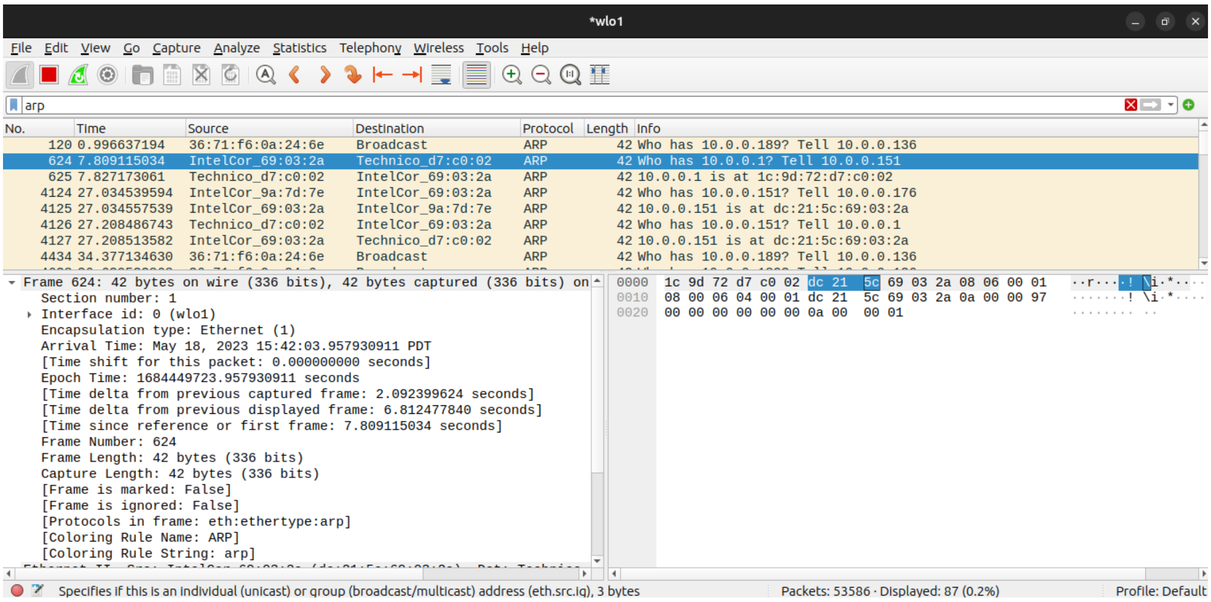
```
aman@aman-Ubuntu:~$ arp -a
? (10.0.0.176) at 6c:6a:77:9a:7d:7e [ether] on wlo1
_gateway (10.0.0.1) at 1c:9d:72:d7:c0:02 [ether] on wlo1
aman@aman-Ubuntu:~$ arp -d
arp: need host name
aman@aman-Ubuntu:~$ arp -a
? (10.0.0.176) at 6c:6a:77:9a:7d:7e [ether] on wlo1
_gateway (10.0.0.1) at 1c:9d:72:d7:c0:02 [ether] on wlo1
aman@aman-Ubuntu:~$ sudo arp -d 10.0.0.1 && arp -a
[sudo] password for aman:
? (10.0.0.176) at 6c:6a:77:9a:7d:7e [ether] on wlo1
aman@aman-Ubuntu:~$ arp -a
? (10.0.0.176) at 6c:6a:77:9a:7d:7e [ether] on wlo1
_gateway (10.0.0.1) at 1c:9d:72:d7:c0:02 [ether] on wlo1
aman@aman-Ubuntu:~$ 
```

5.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
arp
No.   Time            Source              Destination         Protocol Length Info
  120 0.996637194     36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
  624 7.809115034     IntelCor_69:03:2a   Technico_d7:c0:02   ARP      42 Who has 10.0.0.1? Tell 10.0.0.151
  625 7.827173061     Technico_d7:c0:02   IntelCor_69:03:2a   ARP      42 10.0.0.1 is at 1c:9d:72:d7:c0:02
 4124 27.034539594    IntelCor_9a:7d:7e   IntelCor_69:03:2a   ARP      42 Who has 10.0.0.151? Tell 10.0.0.176
 4125 27.034557539    IntelCor_69:03:2a   IntelCor_9a:7d:7e   ARP      42 10.0.0.151 is at dc:21:5c:69:03:2a
 4126 27.208486743    Technico_d7:c0:02   IntelCor_69:03:2a   ARP      42 Who has 10.0.0.151? Tell 10.0.0.1
 4127 27.208513582    IntelCor_69:03:2a   Technico_d7:c0:02   ARP      42 10.0.0.151 is at dc:21:5c:69:03:2a
 4434 34.377134630    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
 4638 36.630532868    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
 4640 36.997149939    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
 4678 38.677372308    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136
 4696 40.316840920    36:71:f6:0a:24:6e   Broadcast           ARP      42 Who has 10.0.0.189? Tell 10.0.0.136

› Frame 624: 42 bytes on wire (336 bits), 42 bytes captured (
› Ethernet II, Src: IntelCor_69:03:2a (dc:21:5c:69:03:2a), Ds
› Address Resolution Protocol (request)

0000  1c 9d 72 d7 c0 02 dc 21  5c 69 03 2a 08 06 00 01   ··r·
0010  08 00 06 04 00 01 dc 21  5c 69 03 2a 0a 00 00 97   ····
0020  00 00 00 00 00 00 0a 00  00 01                     ····

Address Resolution Protocol (arp), 28 bytes    Packets: 44759 · Displayed: 75 (0.2%)    Profile: Default
```

Step 2: Inspect the Trace



arp

No.	Time	Source	Destination	Protocol	Length	Info
120	0.996637194	36:71:f6:0a:24:6e	Broadcast	ARP	42	Who has 10.0.0.189? Tell 10.0.0.136
624	7.809115034	IntelCor_69:03:2a	Technico_d7:c0:02	ARP	42	Who has 10.0.0.1? Tell 10.0.0.151
625	7.827173061	Technico_d7:c0:02	IntelCor_69:03:2a	ARP	42	10.0.0.1 is at 1c:9d:72:d7:c0:02
4124	27.034539594	IntelCor_9a:7d:7e	IntelCor_69:03:2a	ARP	42	Who has 10.0.0.151? Tell 10.0.0.176
4125	27.034557539	IntelCor_69:03:2a	IntelCor_9a:7d:7e	ARP	42	10.0.0.151 is at dc:21:5c:69:03:2a
4126	27.208486743	Technico_d7:c0:02	IntelCor_69:03:2a	ARP	42	Who has 10.0.0.151? Tell 10.0.0.1
4127	27.208513582	IntelCor_69:03:2a	Technico_d7:c0:02	ARP	42	10.0.0.151 is at dc:21:5c:69:03:2a
4434	34.377134630	36:71:f6:0a:24:6e	Broadcast	ARP	42	Who has 10.0.0.189? Tell 10.0.0.136

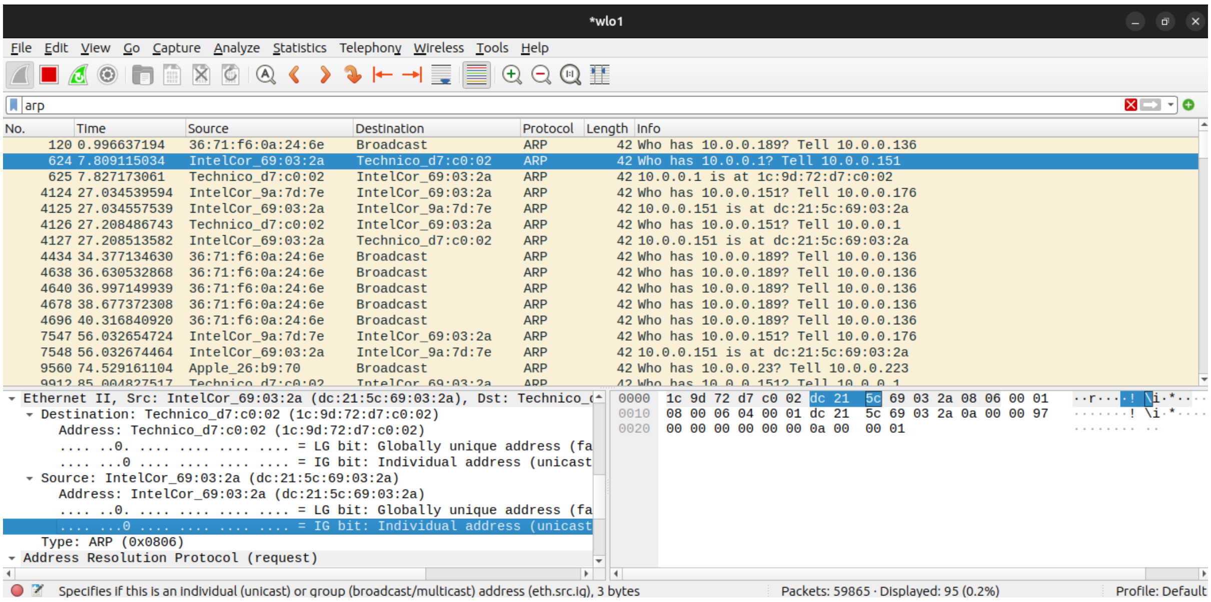
```
Frame 624: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on
  Section number: 1
  Interface id: 0 (wlo1)
  Encapsulation type: Ethernet (1)
  Arrival Time: May 18, 2023 15:42:03.957930911 PDT
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1684449723.957930911 seconds
  [Time delta from previous captured frame: 2.092399624 seconds]
  [Time delta from previous displayed frame: 6.812477840 seconds]
  [Time since reference or first frame: 7.809115034 seconds]
  Frame Number: 624
  Frame Length: 42 bytes (336 bits)
  Capture Length: 42 bytes (336 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:arp]
  [Coloring Rule Name: ARP]
  [Coloring Rule String: arp]
```

```
0000  1c 9d 72 d7 c0 02 dc 21  5c 69 03 2a 08 06 00 01   ··r···!\i·*···
0010  08 00 06 04 00 01 dc 21  5c 69 03 2a 0a 00 00 97   ·······! \i·*····
0020  00 00 00 00 00 00 0a 00  00 01                     ········ ··
```

Specifies if this is an individual (unicast) or group (broadcast/multicast) address (eth.src.ig), 3 bytes

Packets: 53586 · Displayed: 87 (0.2%)

Profile: Default



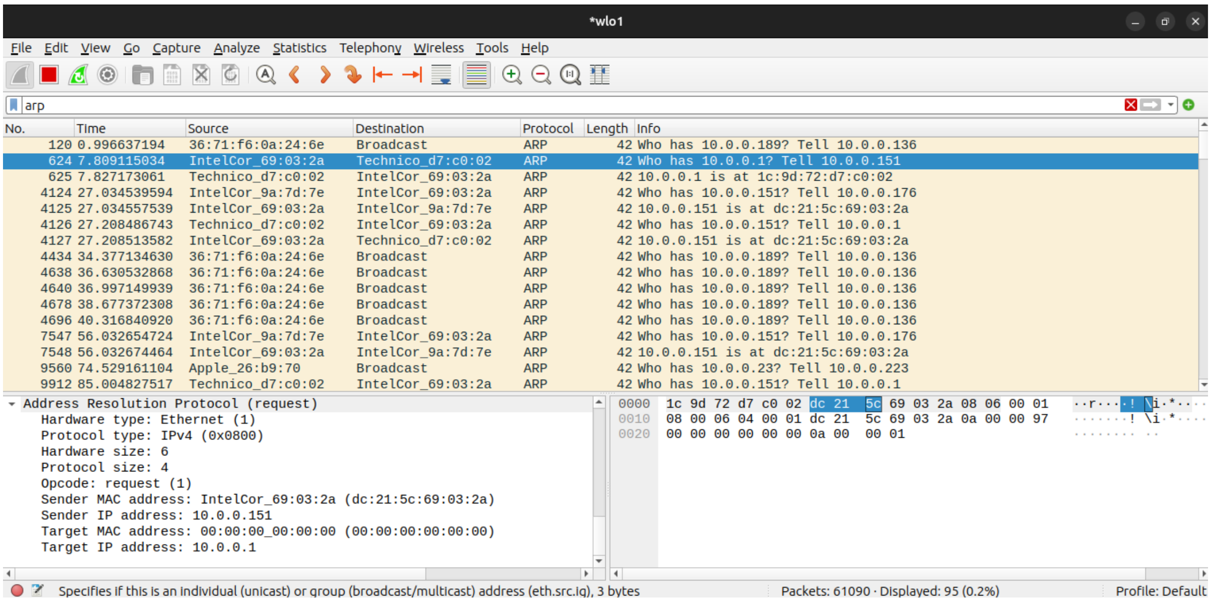
*wlo1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
arp

```
No.   Time          Source              Destination         Protocol Length Info
  120 0.996637194   36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
  624 7.809115034   IntelCor_69:03:2a   Technico_d7:c0:02   ARP        42 Who has 10.0.0.1? Tell 10.0.0.151
  625 7.827173061   Technico_d7:c0:02   IntelCor_69:03:2a   ARP        42 10.0.0.1 is at 1c:9d:72:d7:c0:02
 4124 27.034539594  IntelCor_9a:7d:7e   IntelCor_69:03:2a   ARP        42 Who has 10.0.0.151? Tell 10.0.0.176
 4125 27.034557539  IntelCor_69:03:2a   IntelCor_9a:7d:7e   ARP        42 10.0.0.151 is at dc:21:5c:69:03:2a
 4126 27.208486743  Technico_d7:c0:02   IntelCor_69:03:2a   ARP        42 Who has 10.0.0.151? Tell 10.0.0.1
 4127 27.208513582  IntelCor_69:03:2a   Technico_d7:c0:02   ARP        42 10.0.0.151 is at dc:21:5c:69:03:2a
 4434 34.377134630  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 4638 36.630532868  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 4640 36.997149939  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 4678 38.677372308  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 4696 40.316840920  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 7547 56.032654724  IntelCor_9a:7d:7e   IntelCor_69:03:2a   ARP        42 Who has 10.0.0.151? Tell 10.0.0.176
 7548 56.032674464  IntelCor_69:03:2a   IntelCor_9a:7d:7e   ARP        42 10.0.0.151 is at dc:21:5c:69:03:2a
 9560 74.529161104  Apple_26:b9:70      Broadcast           ARP        42 Who has 10.0.0.23? Tell 10.0.0.223
 9912 85.004827517  Technico_d7:c0:02   IntelCor_69:03:2a   ARP        42 Who has 10.0.0.151? Tell 10.0.0.1
```

```
▾ Ethernet II, Src: IntelCor_69:03:2a (dc:21:5c:69:03:2a), Dst: Technico_(
  ▾ Destination: Technico_d7:c0:02 (1c:9d:72:d7:c0:02)
      Address: Technico_d7:c0:02 (1c:9d:72:d7:c0:02)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (fa
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast
  ▾ Source: IntelCor_69:03:2a (dc:21:5c:69:03:2a)
      Address: IntelCor_69:03:2a (dc:21:5c:69:03:2a)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (fa
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast
    Type: ARP (0x0806)
▾ Address Resolution Protocol (request)
```

```
0000  1c 9d 72 d7 c0 02 dc 21  5c 69 03 2a 08 06 00 01   ··r····! \i·*····
0010  08 00 06 04 00 01 dc 21  5c 69 03 2a 0a 00 00 97   ·······! \i·*····
0020  00 00 00 00 00 00 0a 00  00 01                     ········ ··
```

Specifies if this is an individual (unicast) or group (broadcast/multicast) address (eth.src.ig), 3 bytes   Packets: 59865 · Displayed: 95 (0.2%)   Profile: Default



*wlo1
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
arp

```
No.   Time          Source              Destination         Protocol Length Info
  120 0.996637194   36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
  624 7.809115034   IntelCor_69:03:2a   Technico_d7:c0:02   ARP        42 Who has 10.0.0.1? Tell 10.0.0.151
  625 7.827173061   Technico_d7:c0:02   IntelCor_69:03:2a   ARP        42 10.0.0.1 is at 1c:9d:72:d7:c0:02
 4124 27.034539594  IntelCor_9a:7d:7e   IntelCor_69:03:2a   ARP        42 Who has 10.0.0.151? Tell 10.0.0.176
 4125 27.034557539  IntelCor_69:03:2a   IntelCor_9a:7d:7e   ARP        42 10.0.0.151 is at dc:21:5c:69:03:2a
 4126 27.208486743  Technico_d7:c0:02   IntelCor_69:03:2a   ARP        42 Who has 10.0.0.151? Tell 10.0.0.1
 4127 27.208513582  IntelCor_69:03:2a   Technico_d7:c0:02   ARP        42 10.0.0.151 is at dc:21:5c:69:03:2a
 4434 34.377134630  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 4638 36.630532868  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 4640 36.997149939  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 4678 38.677372308  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 4696 40.316840920  36:71:f6:0a:24:6e   Broadcast           ARP        42 Who has 10.0.0.189? Tell 10.0.0.136
 7547 56.032654724  IntelCor_9a:7d:7e   IntelCor_69:03:2a   ARP        42 Who has 10.0.0.151? Tell 10.0.0.176
 7548 56.032674464  IntelCor_69:03:2a   IntelCor_9a:7d:7e   ARP        42 10.0.0.151 is at dc:21:5c:69:03:2a
 9560 74.529161104  Apple_26:b9:70      Broadcast           ARP        42 Who has 10.0.0.23? Tell 10.0.0.223
 9912 85.004827517  Technico_d7:c0:02   IntelCor_69:03:2a   ARP        42 Who has 10.0.0.151? Tell 10.0.0.1
```

```
▾ Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: IntelCor_69:03:2a (dc:21:5c:69:03:2a)
    Sender IP address: 10.0.0.151
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.0.0.1
```

```
0000  1c 9d 72 d7 c0 02 dc 21  5c 69 03 2a 08 06 00 01   ··r····! \i·*····
0010  08 00 06 04 00 01 dc 21  5c 69 03 2a 0a 00 00 97   ·······! \i·*····
0020  00 00 00 00 00 00 0a 00  00 01                     ········ ··
```

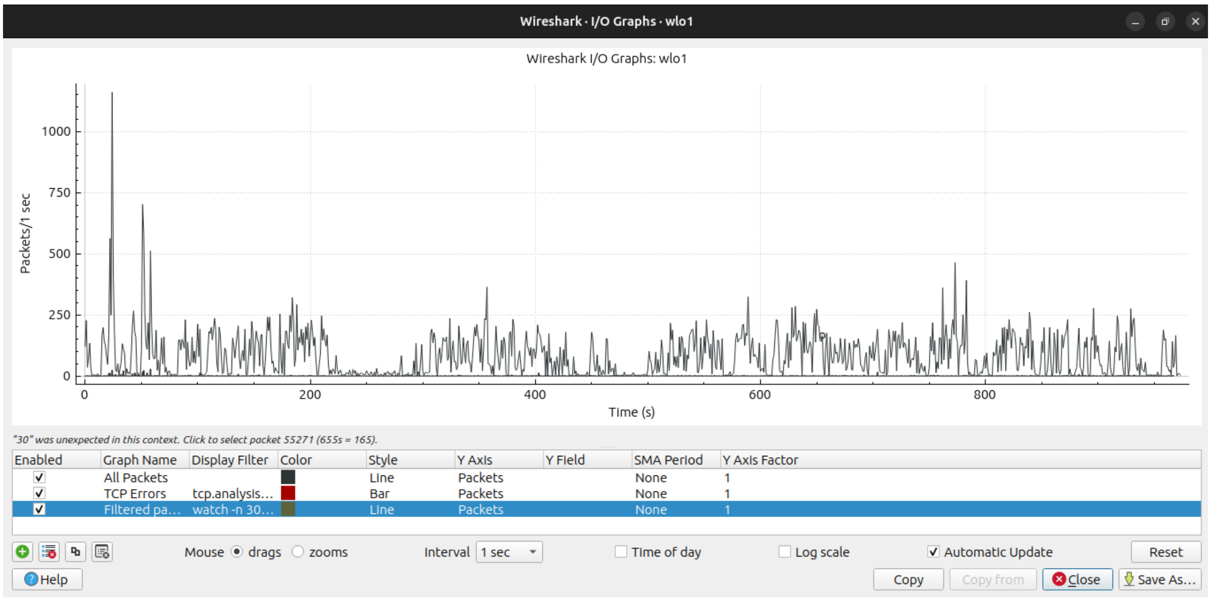
Specifies if this is an individual (unicast) or group (broadcast/multicast) address (eth.src.ig), 3 bytes   Packets: 61090 · Displayed: 95 (0.2%)   Profile: Default

Step 3: Details of ARP over Ethernet

1. Opcode used to indicate a request is 1, while 2 is used to indicate a reply.
2. ARP header size for a request and reply is 28 bytes.
3. The value carried on a request for the unknown target MAC address is 00:00:00:00:00:00.
4. Type:ARP (0x0806) is the ethernet Type value indicates that ARP is the higher layer protocol.

4.2. Understanding TCP networks sockets

Command : watch -n 30 “netstat -at | grep ‘ESTABLISHED\|LISTEN’ | tee -a Tcp.txt”

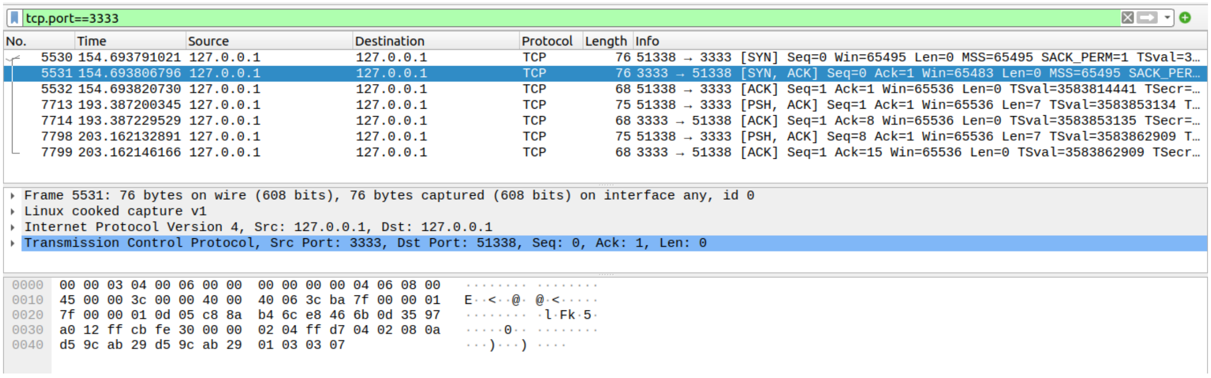


"30" was unexpected in this context. Click to select packet 55271 (655s = 165).

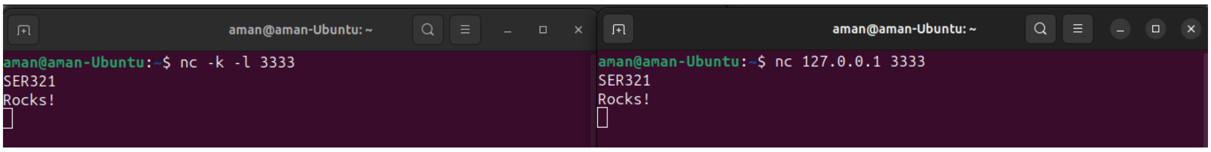
Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period	Y Axis Factor
☑	All Packets		■	Line	Packets		None	1
☑	TCP Errors	tcp.analysis…	🟥	Bar	Packets		None	1
☑	Filtered pa…	watch -n 30…	■	Line	Packets		None	1

4.3. Sniffing TCP/UDP traffic

Step 1: TCP



```
tcp.port==3333

No.  Time           Source      Destination  Protocol Length Info
5530 154.693791021 127.0.0.1   127.0.0.1    TCP      76 51338 → 3333 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=3…
5531 154.693806796 127.0.0.1   127.0.0.1    TCP      76 3333 → 51338 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PER…
5532 154.693820730 127.0.0.1   127.0.0.1    TCP      68 51338 → 3333 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=3583814441 TSecr=…
7713 193.387200345 127.0.0.1   127.0.0.1    TCP      75 51338 → 3333 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=7 TSval=3583853134 T…
7714 193.387229529 127.0.0.1   127.0.0.1    TCP      68 3333 → 51338 [ACK] Seq=1 Ack=8 Win=65536 Len=0 TSval=3583853135 TSecr=…
7798 203.162132891 127.0.0.1   127.0.0.1    TCP      75 51338 → 3333 [PSH, ACK] Seq=8 Ack=1 Win=65536 Len=7 TSval=3583862909 T…
7799 203.162146166 127.0.0.1   127.0.0.1    TCP      68 3333 → 51338 [ACK] Seq=1 Ack=15 Win=65536 Len=0 TSval=3583862909 TSecr…

› Frame 5531: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface any, id 0
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
› Transmission Control Protocol, Src Port: 3333, Dst Port: 51338, Seq: 0, Ack: 1, Len: 0

0000  00 00 03 04 00 06 00 00  00 00 00 00 04 06 08 00   ········ ········
0010  45 00 00 3c 00 00 40 00  40 06 3c ba 7f 00 00 01   E··<··@· @·<·····
0020  7f 00 00 01 0d 05 c8 8a  b4 6c e8 46 6b 0d 35 97   ········ ·l·Fk·5·
0030  a0 12 ff cb fe 30 00 00  02 04 ff d7 04 02 08 0a   ·····0·· ········
0040  d5 9c ab 29 d5 9c ab 29  01 03 03 07               ···)···) ····
```



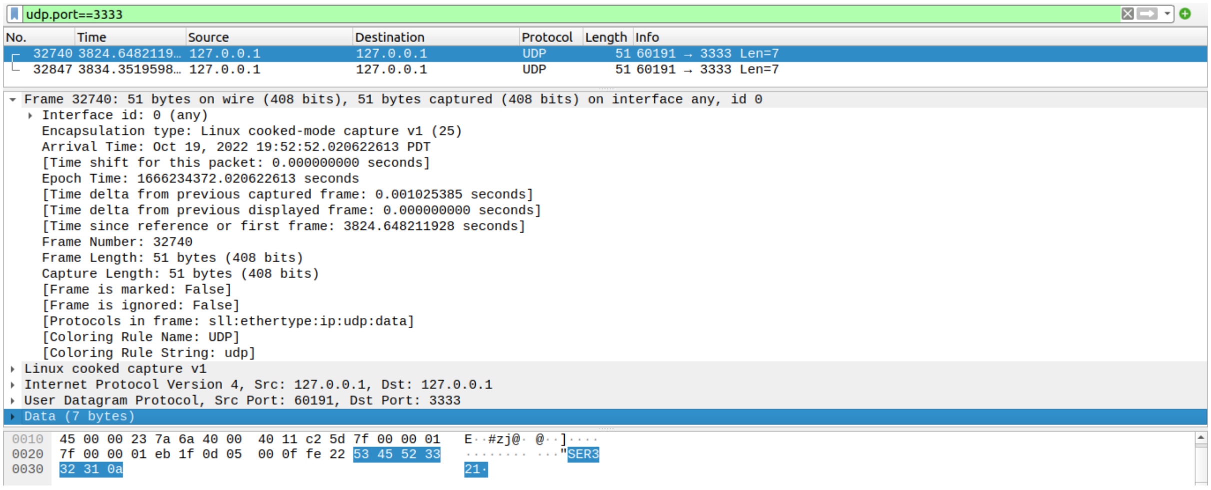
```
aman@aman-Ubuntu:~$ nc -k -l 3333
SER321
Rocks!
```

```
aman@aman-Ubuntu:~$ nc 127.0.0.1 3333
SER321
Rocks!
```

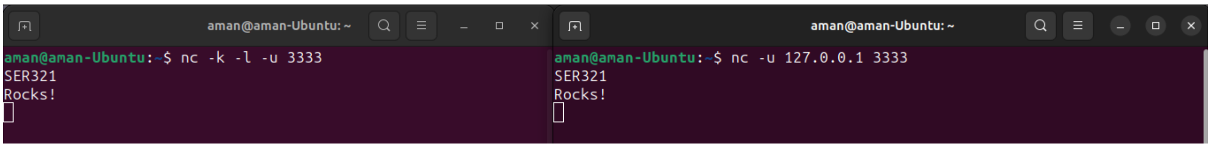
5

a. The nc -k -l 3333 command helps us to open tcp port 3333 and the second command did a loopback to port number 3333 to send data.
b. Frames: 8 – I counted all frames that were sent.
c. 2 packets were sent from the client side and server side.
d. In total 10 packets were needed to capture the whole process.
e. 152 bytes were sent (76 each).
f. Total of 608 bytes were sent over the wire.
g. The actual data was only 20 bytes but the packets containing data were 152 bytes which mean 456 bytes were overhead.

Step 2: UDP



```
udp.port==3333
No.    Time           Source      Destination  Protocol Length Info
32740 3824.6482119… 127.0.0.1   127.0.0.1    UDP          51 60191 → 3333 Len=7
32847 3834.3519598… 127.0.0.1   127.0.0.1    UDP          51 60191 → 3333 Len=7

Frame 32740: 51 bytes on wire (408 bits), 51 bytes captured (408 bits) on interface any, id 0
  Interface id: 0 (any)
  Encapsulation type: Linux cooked-mode capture v1 (25)
  Arrival Time: Oct 19, 2022 19:52:52.020622613 PDT
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1666234372.020622613 seconds
  [Time delta from previous captured frame: 0.001025385 seconds]
  [Time delta from previous displayed frame: 0.000000000 seconds]
  [Time since reference or first frame: 3824.648211928 seconds]
  Frame Number: 32740
  Frame Length: 51 bytes (408 bits)
  Capture Length: 51 bytes (408 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: sll:ethertype:ip:udp:data]
  [Coloring Rule Name: UDP]
  [Coloring Rule String: udp]
Linux cooked capture v1
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
User Datagram Protocol, Src Port: 60191, Dst Port: 3333
Data (7 bytes)

0010  45 00 00 23 7a 6a 40 00  40 11 c2 5d 7f 00 00 01   E··#zj@· @··]····
0020  7f 00 00 01 eb 1f 0d 05  00 0f fe 22 53 45 52 33   ········ ···"SER3
0030  32 31 0a                                           21·
```



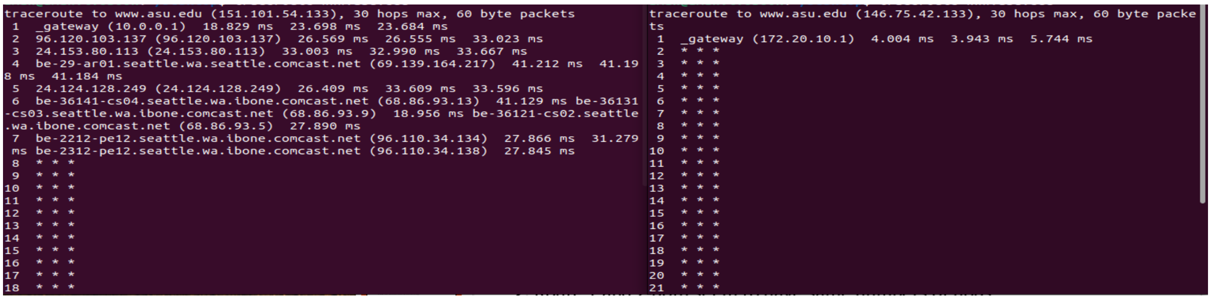
```
aman@aman-Ubuntu:~$ nc -k -l -u 3333
SER321
Rocks!
```

```
aman@aman-Ubuntu:~$ nc -u 127.0.0.1 3333
SER321
Rocks!
```

4.

a. The nc -k -l -u 3333 command helps us to open UDP port 3333 and the second command did a loopback to port number 3333 to send data.
b. Two frames were needed to capture those two lines.
c. Two packets were needed to capture those two lines.
d. Two packets were needed to capture the whole process.
e. Total of 102 bytes were sent over the wire.
f. 6. 14 bytes is the data (only the data) that was send.
g. 7. Compared to the whole process of 102 bytes only 14 bytes were containing information which is around 15% of the total byte.
h. UDP has less overhead than TCP as it does not have error correction or flow control. TCP also exchanges data like sequence number(seq) and acknowledgment(ack) while UDP does not exchange those data.

4.4 Internet Protocol (IP) Routing



```
traceroute to www.asu.edu (151.101.54.133), 30 hops max, 60 byte packets
 1  _gateway (10.0.0.1)  18.829 ms  23.698 ms  23.684 ms
 2  96.120.103.137 (96.120.103.137)  26.569 ms  26.555 ms  33.023 ms
 3  24.153.80.113 (24.153.80.113)  33.003 ms  32.990 ms  33.667 ms
 4  be-29-ar01.seattle.wa.seattle.comcast.net (69.139.164.217)  41.212 ms  41.19
8 ms  41.184 ms
 5  24.124.128.249 (24.124.128.249)  26.409 ms  33.609 ms  33.596 ms
 6  be-36141-cs04.seattle.wa.ibone.comcast.net (68.86.93.13)  41.129 ms be-36131
-cs03.seattle.wa.ibone.comcast.net (68.86.93.9)  18.956 ms be-36121-cs02.seattle
.wa.ibone.comcast.net (68.86.93.5)  27.890 ms
 7  be-2212-pe12.seattle.wa.ibone.comcast.net (96.110.34.134)  27.866 ms  31.279
 ms be-2312-pe12.seattle.wa.ibone.comcast.net (96.110.34.138)  27.845 ms
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
```

```
traceroute to www.asu.edu (146.75.42.133), 30 hops max, 60 byte packe
ts
 1  _gateway (172.20.10.1)  4.004 ms  3.943 ms  5.744 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
```

4. a. Route 2 seems to be fast
   b. Route 1 and 2 both seem to have the same number of hops.

4.5 Running client servers in different ways

4.5.1: Youtube Link : https://youtu.be/b97KUMNqr1s

4.5.2: I had to change the host address to the AWS's public IP address in order to connect it to the server which is running on AWS. And in Wireshark I had to filter to tcp port 8888 to see what data is being sent by client to server.

4.5.3: As running client on AWS requires a public IP of your computer and I was having issues finding a public IP address of my personal computer. And connecting it with a private IP address, it says connection refused while trying to connect from AWS.

4.5.4: We can reach our AWS server easily because it has public IP address which allows traffic to enter the system.
You will need a public IP address of your local computer and also you will need to allow traffic in order to reach it from outside your local network.
I think firewall is blocking traffic coming from the AWS server side and so it cannot exchange data from AWS.