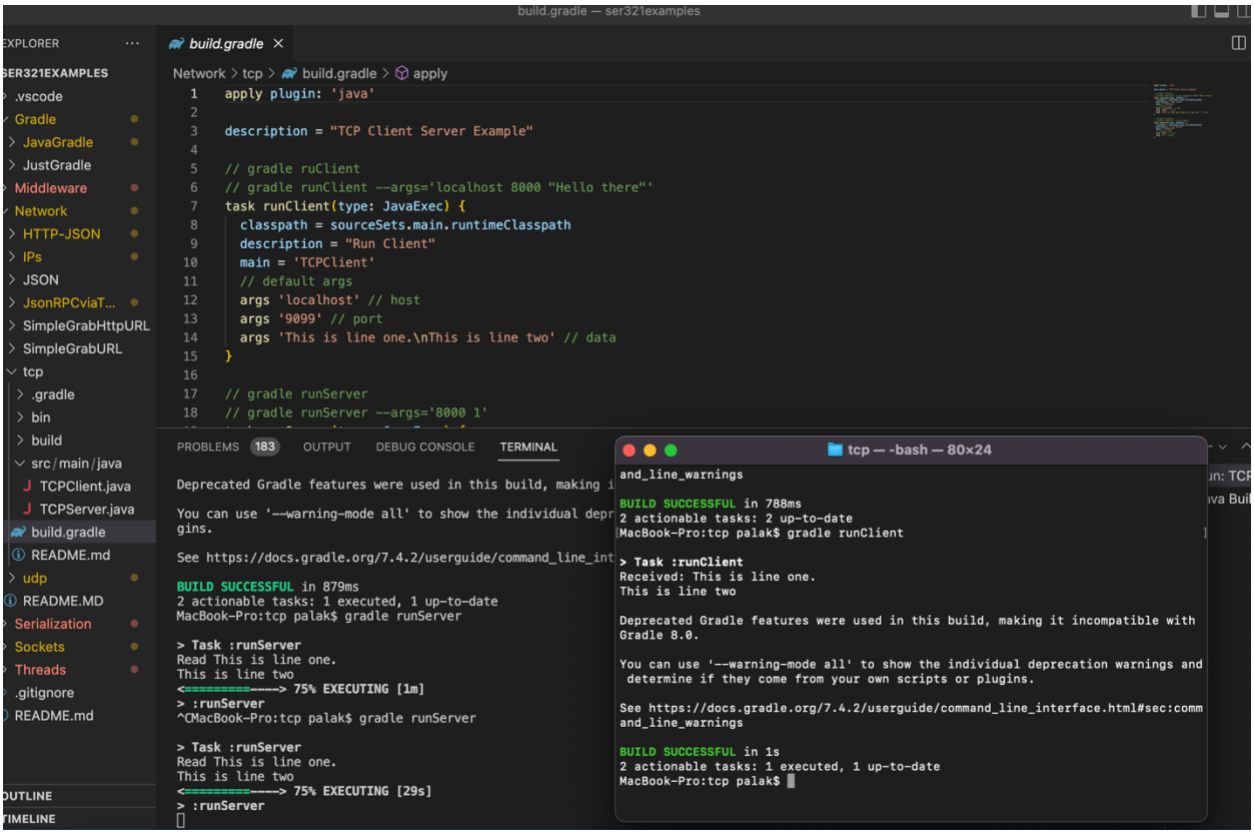
Part1. Command Line Tasks-Linux

1. mkdir cli_assignemnt
2. cd sli_assignment
3. touch stuff.txt
4. cat > stuff.txt
   add text, ctrl+d
5. wc stuff.txt
6. cat >> stuff.txt
   add text, ctrl+d
7. mkdir draft
8. mk 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 -AlR
14. zcat NASA_access_log_Aug95.gz
15. gunzip 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. echo ‘pojha1: This is a great assignment 5/20/2023’ >> logs_snapshot.txt
24. less logs.txt
25. cat marks.csv | tail -n+2 | cut -d “%” -f 1
26. cat marks.csv | cut -d “%” -f 4| soft -n
27. cat marks.csv | tail -n+2| awk -F “%” ‘ {sum+=$2; n++;} END {print sum/n}’
28. cat marks.csv | tail -n+2| awk -F “%” ‘ {sum+=$2; n++;} END {print sum/n}’ >> cli_assignemnt/done.txt
29. mv cli_assignment/done.txt cli_assignment/final/
30. mv cli_assignment/final/done.txt cli_assignment/final/average.txt

3.2 Running Examples

TCP



Serialize

```
MacBook-Pro:GroupSerialize palak$ gradle run

> Task :run
users serialized to users.ser
Server ready and waiting to export a group
Server done exporting a group
Group Administration received. Includes:
Tim
Joe
Sue

Deprecated Gradle features were used in this build, making it incompatible 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 plugins.

See https://docs.gradle.org/7.4.2/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 executed
```

Network/HTTP-JSON

```
MacBook-Pro:Network palak$ cd HTTP-JSON
MacBook-Pro:HTTP-JSON palak$ gradle tasks --all

> Task :tasks

------------------------------------------------------------
Tasks runnable from root project 'HTTP-JSON'
------------------------------------------------------------

Application tasks
-----------------
run - Runs this project as a JVM application

Build tasks
-----------
assemble - Assembles the outputs of this project.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles main classes.
clean - Deletes the build directory.
jar - Assembles a jar archive containing the main classes.
testClasses - Assembles test classes.

Build Setup tasks
-----------------
init - Initializes a new Gradle build.
wrapper - Generates Gradle wrapper files.

Distribution tasks
------------------
assembleDist - Assembles the main distributions
distTar - Bundles the project as a distribution.
distZip - Bundles the project as a distribution.
installDist - Installs the project as a distribution as-is.

Documentation tasks
-------------------
```

```
BUILD SUCCESSFUL in 1s
1 actionable task: 1 executed
MacBook-Pro:HTTP-JSON palak$ gradle javaToolChains

> Task :javaToolchains

 + Options
     | Auto-detection:     Enabled
     | Auto-download:      Enabled

 + AdoptOpenJDK 11.0.9.1+1
     | Location:           /Library/Java/JavaVirtualMachines/adoptopenjdk-11.jdk/Contents/Home
     | Language Version:   11
     | Vendor:             AdoptOpenJDK
     | Architecture:       x86_64
     | Is JDK:             true
     | Detected by:        macOS java_home

 + Amazon Corretto JDK 18.0.2+9-FR
     | Location:           /Users/palak/Library/Java/JavaVirtualMachines/corretto-18.0.2/Contents/Home
     | Language Version:   18
     | Vendor:             Amazon Corretto
     | Architecture:       x86_64
     | Is JDK:             true
     | Detected by:        Current JVM

 + Oracle JDK 15.0.1+9-18
     | Location:           /Library/Java/JavaVirtualMachines/jdk-15.0.1.jdk/Contents/Home
     | Language Version:   15
     | Vendor:             Oracle
     | Architecture:       x86_64
     | Is JDK:             true
     | Detected by:        macOS java_home
     | Is JDK:             true
     | Detected by:        macOS java_home


Deprecated Gradle features were used in this build, making it incompatible 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 plugins.

See https://docs.gradle.org/7.4.2/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 1s
1 actionable task: 1 executed
MacBook-Pro:HTTP-JSON palak$ gradle buildEnvironment

> Task :buildEnvironment

------------------------------------------------------------
Root project 'HTTP-JSON'
------------------------------------------------------------

classpath
No dependencies

A web-based, searchable dependency report is available by adding the --scan option.

Deprecated Gradle features were used in this build, making it incompatible 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 plugins.

See https://docs.gradle.org/7.4.2/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 647ms
1 actionable task: 1 executed
```

3.4 Set Up Second System
https://youtu.be/jHXOHTALRa8

AWS

```
^C[ec2-user@ip-172-31-17-218 JavaSimpleSock2]$ gradle SockServer

> 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 [36s]
> :SocketServer
```

```
MacBook-Pro:JavaSimpleSock2 palak$ gradle SockClient -Phost=18.222.174.7 -Pmessage=hi  -Pnumber
=100

> Task :SocketClient
Got it!

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

You can use '--warning-mode all' to show the individual deprecation warnings and determine if t
hey come from your own scripts or plugins.

See https://docs.gradle.org/7.4.2/userguide/command_line_interface.html#sec:command_line_warnin
gs

BUILD SUCCESSFUL in 1s
2 actionable tasks: 1 executed, 1 up-to-date
MacBook-Pro:JavaSimpleSock2 palak$
```

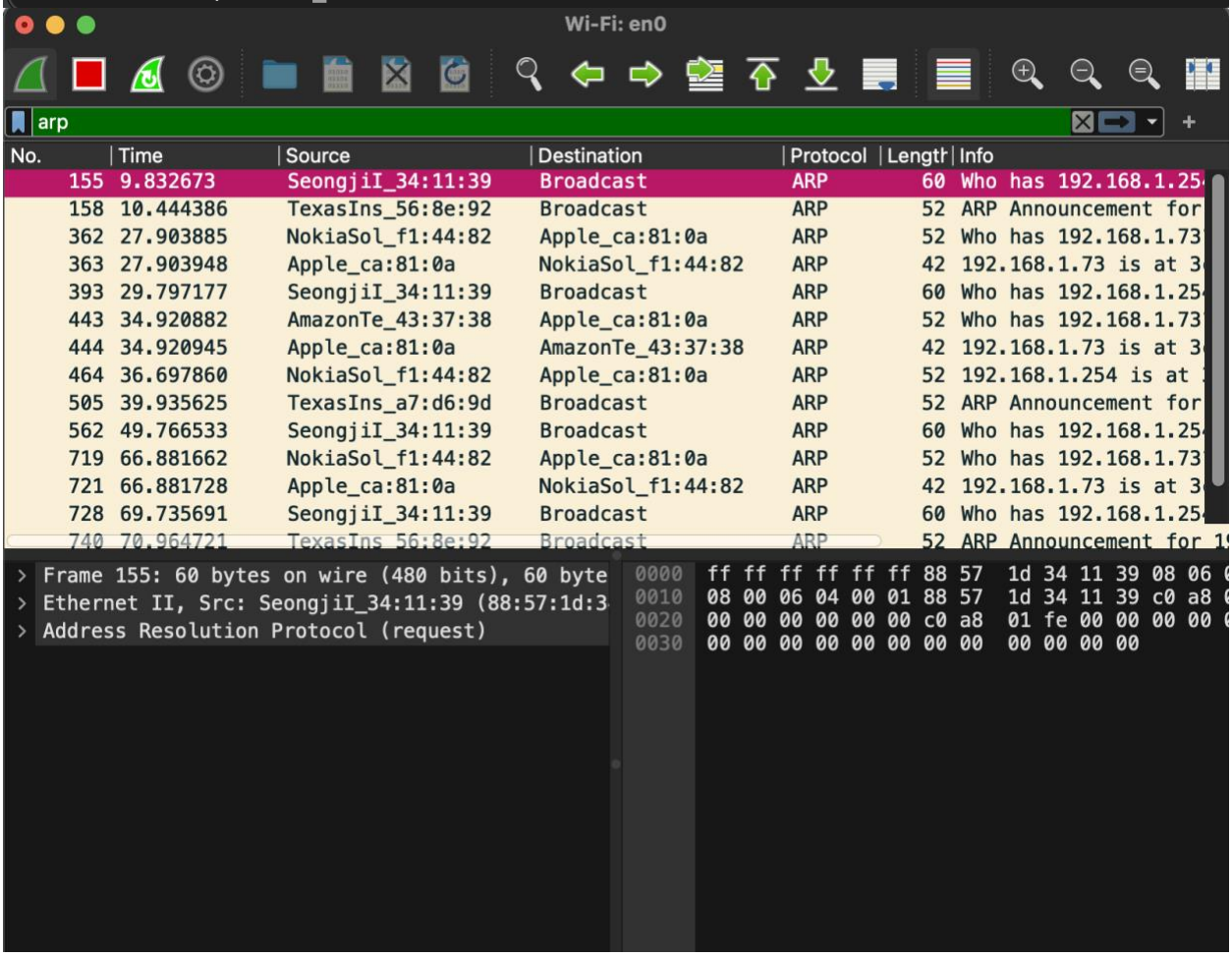
Part II

4.1

```
MacBook-Pro:~ palak$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
        inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
```

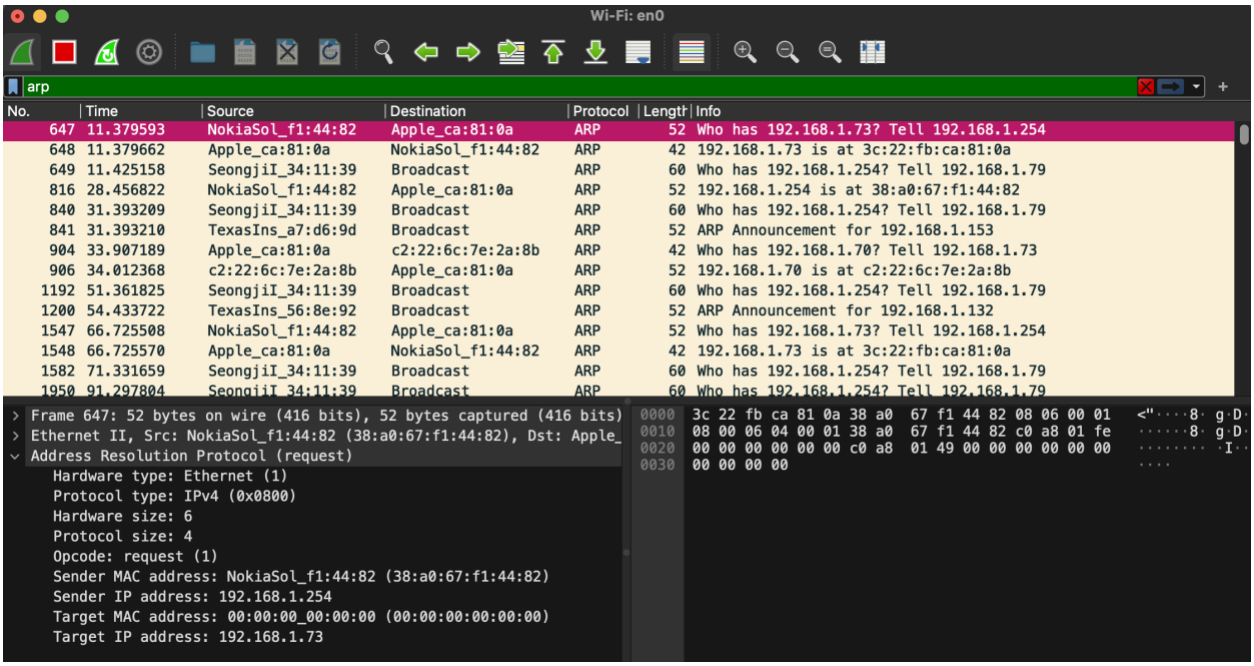
```
MacBook-Pro:~ palak$ netstat -r
Routing tables

Internet:
Destination        Gateway            Flags           Netif Expire
default            dsldevice.attlocal UGScg             en0
127                localhost          UCS               lo0
localhost          localhost          UH                lo0
169.254            link#6             UCS               en0      !
192.168.1          link#6             UCS               en0      !
dir-813.attlocal.n 6c:72:20:c:e7:6b   UHLWIi            en0   1055
unknownc2226c7e2a8 c2:22:6c:7e:2a:8b  UHLWIi            en0   1018
unknowna64ae703a21 a6:4a:e7:3:a2:1c   UHLWIi            en0    680
192.168.1.73/32    link#6             UCS               en0      !
```

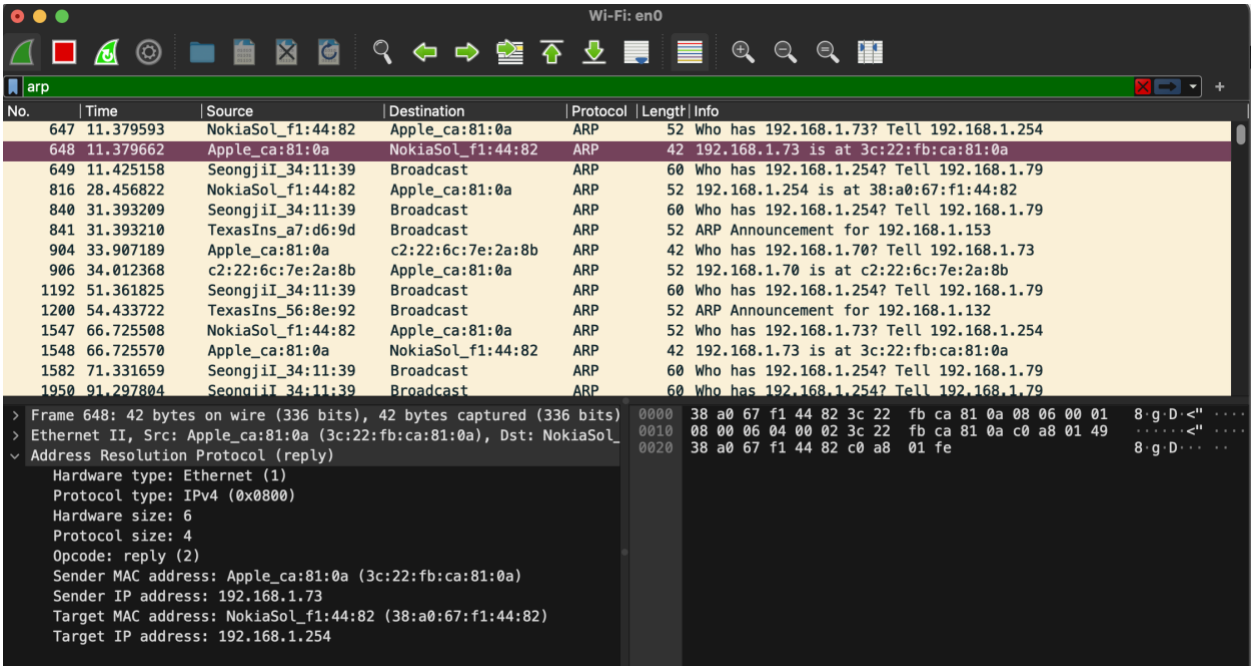
```
MacBook-Pro:~ palak$ route -n get default
   route to: default
destination: default
       mask: default
    gateway: 192.168.1.254
  interface: en0
      flags: <UP,GATEWAY,DONE,STATIC,PRCLONING,GLOBAL>
 recvpipe  sendpipe  ssthresh  rtt,msec    rttvar  hopcount      mtu     expire
       0         0         0         0         0         0      1500         0
MacBook-Pro:~ palak$
```



```
MacBook-Pro:~ palak$ arp -a
dir-813.attlocal.net (192.168.1.64) at 6c:72:20:c:e7:6b on en0 ifscope [ethernet]
unknownc2226c7e2a8b.attlocal.net (192.168.1.70) at c2:22:6c:7e:2a:8b on en0 ifscope [e
thernet]
unknowna64ae703a21c.attlocal.net (192.168.1.71) at a6:4a:e7:3:a2:1c on en0 ifscope [et
hernet]
macbook-pro.attlocal.net (192.168.1.73) at 3c:22:fb:ca:81:a on en0 ifscope permanent [
ethernet]
amazon-428ac459a.attlocal.net (192.168.1.87) at 94:3a:91:43:37:38 on en0 ifscope [ethe
rnet]
```

```
MacBook-Pro:~ palak$ sudo arp -d 192.168.1.254 && arp -a
Password:
192.168.1.254 (192.168.1.254) deleted
dir-813.attlocal.net (192.168.1.64) at 6c:72:20:c:e7:6b on en0 ifscope [ethernet]
unknownc2226c7e2a8b.attlocal.net (192.168.1.70) at c2:22:6c:7e:2a:8b on en0 ifscope [e
thernet]
```





1. What opcode is used to indicate a request? What about a reply?
   The opcode for request is 1 and 2 for reply.
2. How large is the ARP header for a request? What about for a reply?
   ARP header for request and reply Is 28.
3. What value is carried on a request for the unknown target MAC address?
   The value carried on a request for unknown target MAC address is all zeros (00:00:00:00:00)
4. What Ethernet Type value indicates that ARP is the higher layer protocol?
   The Ethernet Type value for ARP is 0x806.

4.2

Command: watch -n 30 “date >> out.txt & netstat -a | grep -w -E ‘ESTABLISHED|LISTEN’ >> out.txt”

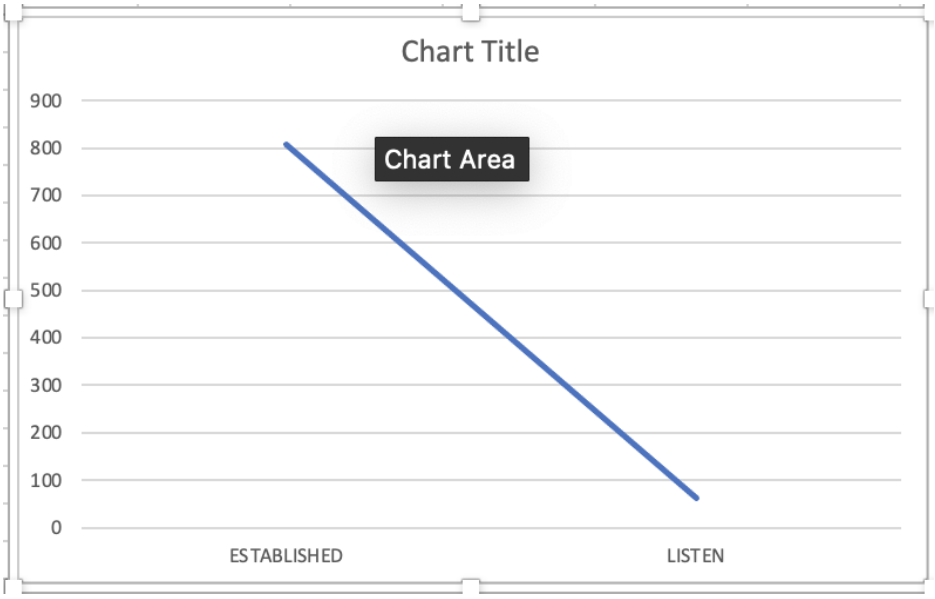
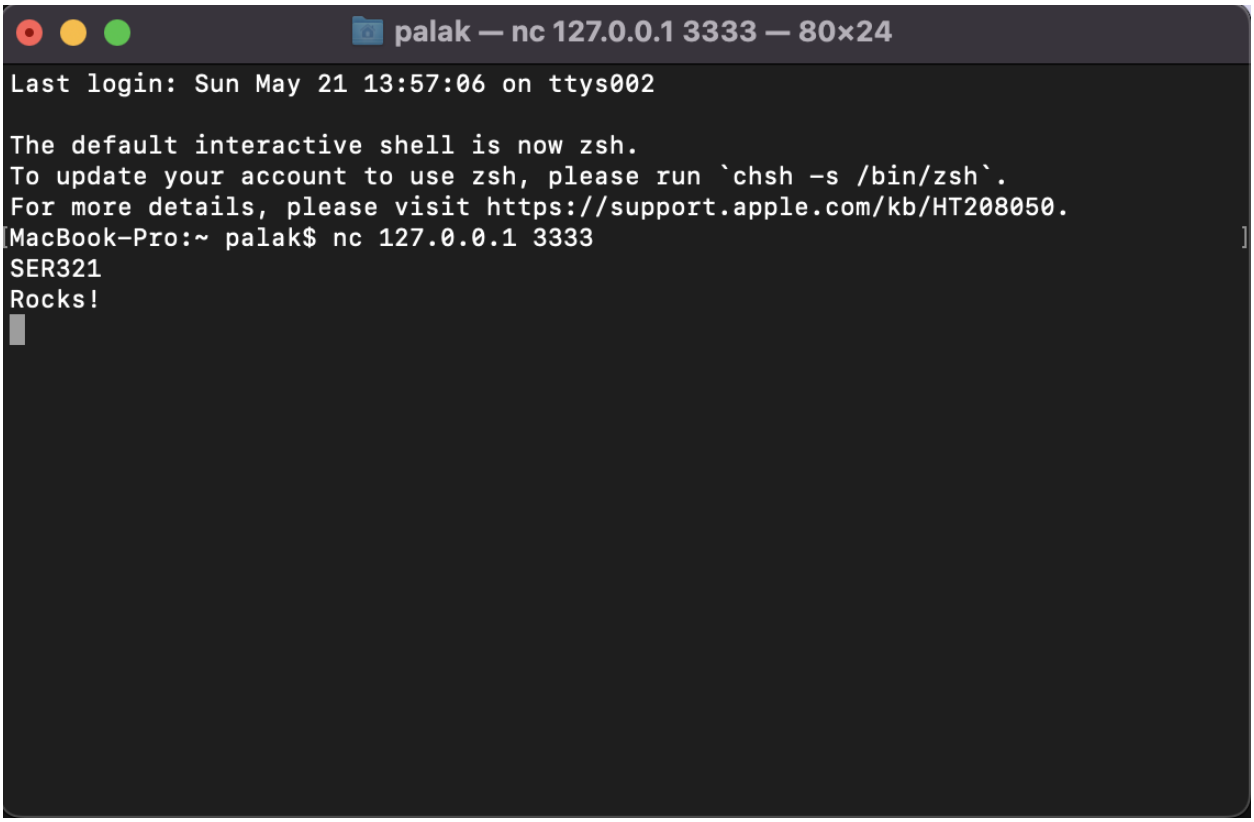


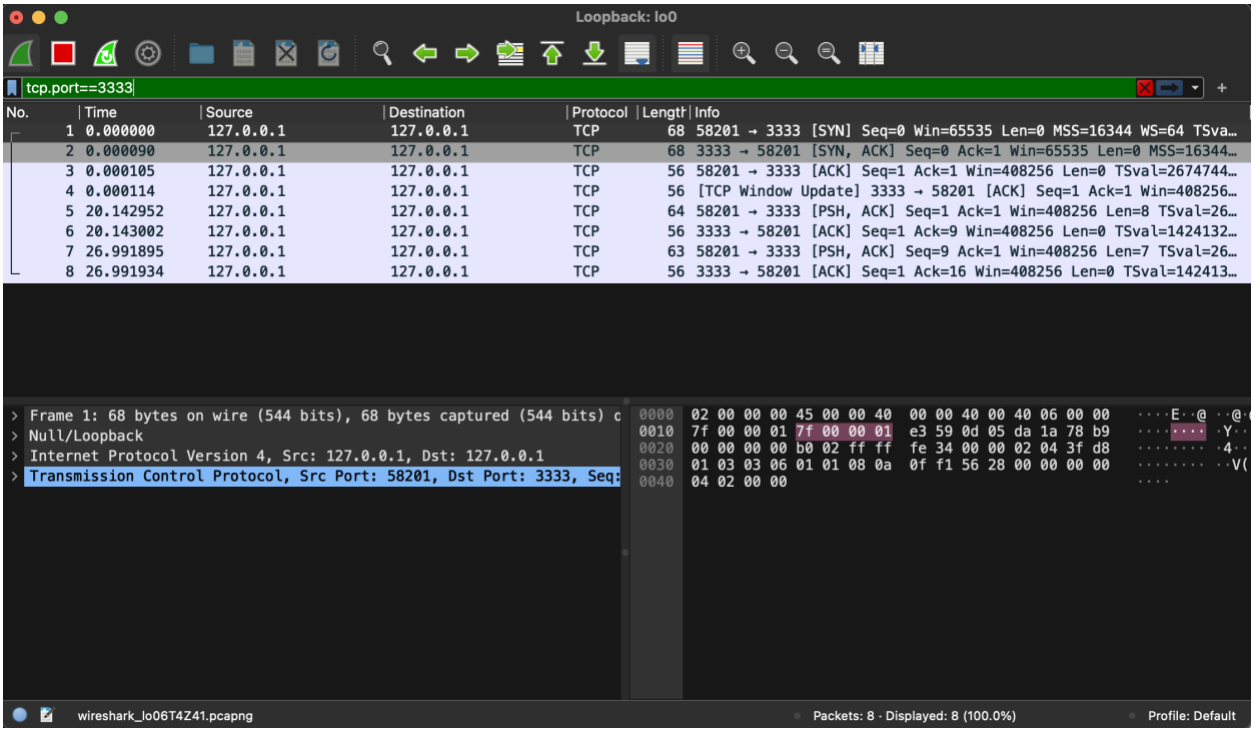
Chart Title
900
800
700
600
500
400
300
200
100
0
Chart Area
ESTABLISHED
LISTEN

4.3

TCP


palak — nc 127.0.0.1 3333 — 80×24
Last login: Sun May 21 13:57:06 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacBook-Pro:~ palak$ nc 127.0.0.1 3333
SER321
Rocks!

```
MacBook-Pro:~ palak$ nc -k -l 3333
SER321
Rocks!
```



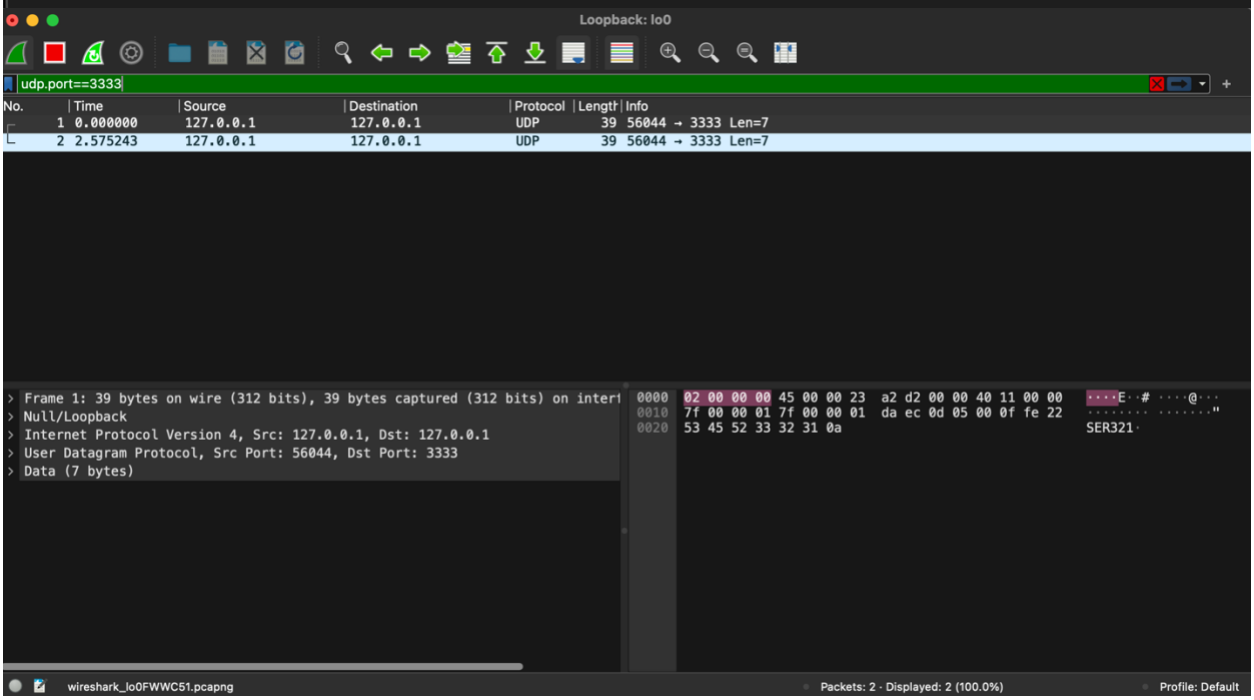
1. Explain both the commands you used in detail. What did they actually do?
   The command nc -k -l 3333 is to connect/listen to a specific port number. It is used to read and write to network connections. The -k is to repeat once it has come into contact with something. The listening port number in this case is 3333. In another terminal, we use the command mc 127.0.0.1 3333. This tells netcat to initiate communication on port 3333 of the device being used.
2. How many frames were send back and forth to capture these 2 lines (Frames: 4 – I counted all frames that were sent)?
   4 frames
3. How many packets were send back and forth to capture only those 2 lines?
   4 packets
4. How many packets were needed to capture the whole "process" (starting the communication, ending the communication)?
   12 packets
5. How many bytes is the data (only the data) that was send?
   14 bytes
6. How many total bytes went over the wire (back and forth) for the whole process?
   710 bytes
7. How much overhead was there. Basically how many bytes was the whole process compared to the actually data that we did send.

The total number of bytes for the whole process which includes the 12 packets is 710 bytes. The packets that carried the data amounted for 238 bytes.710-238 = 472 bytes with no data transfer. SER321 Rocks! Is only 14 bytes. The overhead is 66.5%.

UDP

```
MacBook-Pro:~ palak$ nc -k -l -u 3333
SER321
Rocks!
^C
MacBook-Pro:~ palak$
```

```
MacBook-Pro:~ palak$ nc -u 127.0.0.1 3333
SER321
Rocks!
^C
MacBook-Pro:~ palak$
```



1. Explain both the commands you used in detail. What did they actually do?
   The nc -k -l -u 3333 tells netcat to listen for a connection and continue after it comes into contact with something on port 3333. The -u command tells it to use UDP instead of TCP. In another terminal, the command nc -u 127.0.0.1 3333. This tells netcat to initiate communication on port 3333 and use UDP on the device used.

2. How many frames were needed to capture those 2 lines?
   2 frames

3. How many packets were needed to capture those 2 lines?
   2 packets

4. How many packets were needed to capture the whole "process" (starting the communication, ending the communication)?
   2 packets

5. How many total bytes went over the wire?
   78 bytes

6. How many bytes is the data (only the data) that was send?
   14 bytes

7. Basically how many bytes was the whole process compared to the actually data that we did send.?
   The whole process was 78 bytes and the actual data was 14 bytes.

8. What is the difference in relative overhead between UDP and TCP and why? Specifically, what kind of information was exchanged in TCP that was not exchanged in UDP? Show the relative parts of the packet traces.
   UDP has less overhead than TCP. The difference in the overhead can be seen in the packets. We see more packets in TCP because of direct connection, while UDP doesn't require direct connection. This is because TCP connects to the receiving computer/network directly while UDP sends data and relies on the devices in between to deliver information. The header for the TCP connections is larger than it is for UDP because of the connection-oriented protocol.

4.4
ASU Network

```
traceroute: Warning: www.asu.edu has multiple addresses; using 151.101.194.133
traceroute to pantheon-systems.map.fastly.net (151.101.194.133), 64 hops max, 52
 byte packets
 1  dsldevice (192.168.1.254)  3.766 ms  2.554 ms  2.869 ms
 2  108-226-52-1.lightspeed.fyvlar.sbcglobal.net (108.226.52.1)  3.757 ms  8.288
 ms  8.403 ms
 3  75.14.128.90 (75.14.128.90)  7.857 ms  6.164 ms  6.943 ms
 4  * * *
 5  * * *
 6  * * *
 7  32.130.16.29 (32.130.16.29)  22.183 ms  19.533 ms  19.998 ms
 8  * * *
 9  * * *
```

Non-ASU Network

```
MacBook-Pro:~ palak$ traceroute www.asu.edu
traceroute to pantheon-systems.map.fastly.net (146.75.126.133), 64 hops max, 52
byte packets
 1  172.20.10.1 (172.20.10.1)  7.594 ms  4.443 ms  4.147 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  10.166.254.69 (10.166.254.69)  126.938 ms  108.460 ms  108.305 ms
 7  10.164.160.109 (10.164.160.109)  125.602 ms  93.168 ms  176.700 ms
 8  10.164.162.165 (10.164.162.165)  131.219 ms  152.876 ms  135.013 ms
 9  * * *
10  * *
```

1. Which is the fastest?
   Route 1 was the fastest.

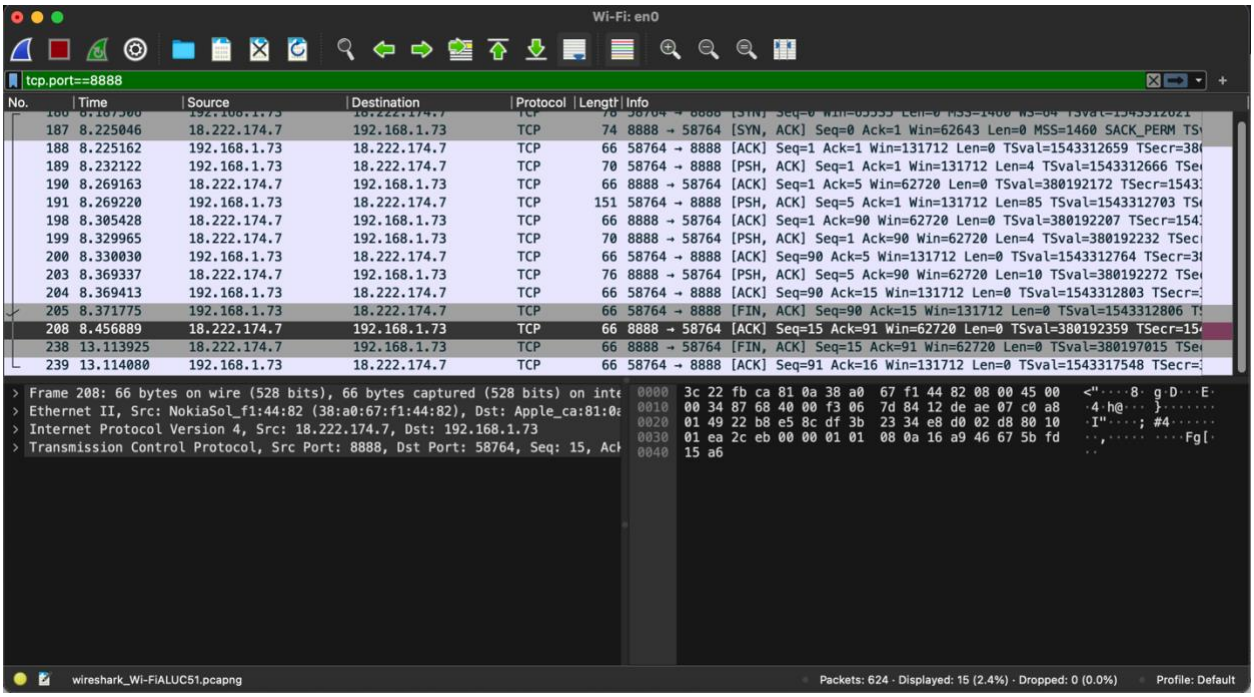
2. Which has the fewest hops?
   Route 1 had the fewest hops.

4.5
Running locally
https://youtu.be/7SzbWWOBK2Y

Running server on AWS

```
^C[ec2-user@ip-172-31-17-218 JavaSimpleSock2]$ gradle SockServer

> Task :SocketServer
Server ready for 3 connections
Server waiting for a connection
Received the String hello
Received the Integer 25
Server waiting for a connection
<==========----> 75% EXECUTING [8s]
> :SocketServer
^C[ec2-user@ip-172-31-17-218 JavaSimpleSock2]$
```

```
[MacBook-Pro:JavaSimpleSock2 palak$ gradle SockClient -Phost=18.222.174.7 -Pmessage=hel
lo  -Pnumber=25

> Task :SocketClient
Got it!

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

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

See https://docs.gradle.org/7.4.2/userguide/command_line_interface.html#sec:command_li
ne_warnings

BUILD SUCCESSFUL in 1s
2 actionable tasks: 1 executed, 1 up-to-date
MacBook-Pro:JavaSimpleSock2 palak$
```



Client on AWS

This doesn't work without issues. The user needs to know what host address to connect to. The EC2 instance I have only allows traffic into port 8888. My device is not set up to receive anything from that port from an outside source.

Client on AWS2

Reaching my server on AWS can be done, but anything leaving the AWS server or coming into my device through that port is much harder. A home router uses network address translation to hide the subnet. Since multiple devices can communicate on a global IP address, when that IP address from outside the network is addressed, it is accessing the entire router. You would have to change configurations on that router.