Part 1
2. Command line task

1. mkdir cli_assignment
2. cd cli_assignment
3. touch stuff.txt
4. cat >>stuff.txt
   type any text
5. wc stuff.txt
6. cat >>stuff.txt
7. mkdir draft
8. mv stuff.txt draft/
9. cd draft/
   touch .[secret.txt]
10. copy draft/ final/
11. mv draft draft.remove
12. mv draft.remove/ final/
13. ~cli_assignment $ ls -l
14. zcat NASA_access_log_Aug95.gz
15. gunzip NASA_access_log_Aug95.gz
16. mv NASA_access_log_Aug95 log.txt
17. mv ~/logs.txt ~/cli_assignment/
18. head -n 100 logs.txt
19. head -n 100 logs.txt >>logs_top_100.txt
20. tail -n 100 logs.txt
21. tail -n 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
24. cat logs.txt | less
25. cut -d% -f 2 marks.csv
26. cut -d'%' -f6 marks.csv | sort
27. cut -d'%' -f5 marks.csv | awk '{ sum += $1; n++ } END { if (n > 0) print sum / n; }'
28.cut -d'%' -f5 marks.csv | awk '{ sum += $1; n++ } END { if (n > 0) print sum / n; }' > done.txt
29. mv done.txt final/
30. mv done.txt average.txt

3.1 Setup GitHub repo
https://github.com/ssprasad-e36king/ser321-summer2022-C-asurite

3.2 Running Examples
Example 1
Threads > Accounts

```
Account — -zsh — 80×24
Last login: Mon May 23 02:54:14 on ttys000
shynalprasad@Shynals-MacBook-Pro ~ % cd /Users/shynalprasad/Desktop/ser321exampl
es-master/Threads/Account
shynalprasad@Shynals-MacBook-Pro Account % gradle run

> Task :run
Transaction started #1
Transaction started #5
Transaction started #4
Transaction started #2
Transaction started #3
Balance is 150

BUILD SUCCESSFUL in 1s
2 actionable tasks: 1 executed, 1 up-to-date
shynalprasad@Shynals-MacBook-Pro Account %
```

Is a small application which gives balance for transactions. It involves account, id and numDeposite. It goes through a loop with the number of transaction and it displays the calculated balance.

Example 2
Network>http json

```
Last login: Mon May 23 03:13:57 on ttys000
shynalprasad@Shynals-MacBook-Pro ~ % cd /Users/shynalprasad/Desktop/ser321examples-master/Network/HTTP-JSON
shynalprasad@Shynals-MacBook-Pro HTTP-JSON % gradle tasks --all

> Task :tasks

------------------------------------------------------------
Tasks runnable from root project
------------------------------------------------------------

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
-------------------
javadoc - Generates Javadoc API documentation for the main source code.

Help tasks
----------
buildEnvironment - Displays all buildscript dependencies declared in root project 'HTTP-JSON'.
components - Displays the components produced by root project 'HTTP-JSON'. [incubating]
dependencies - Displays all dependencies declared in root project 'HTTP-JSON'.
dependencyInsight - Displays the insight into a specific dependency in root project 'HTTP-JSON'.
dependentComponents - Displays the dependent components of components in root project 'HTTP-JSON'. [incubating]
help - Displays a help message.
model - Displays the configuration model of root project 'HTTP-JSON'. [incubating]
outgoingVariants - Displays the outgoing variants of root project 'HTTP-JSON'.
projects - Displays the sub-projects of root project 'HTTP-JSON'.
properties - Displays the properties of root project 'HTTP-JSON'.
tasks - Displays the tasks runnable from root project 'HTTP-JSON'.

Verification tasks
```

```
Verification tasks
------------------
check - Runs all checks.
test - Runs the unit tests.

Other tasks
-----------
API
compileJava - Compiles main Java source.
compileTestJava - Compiles test Java source.
prepareKotlinBuildScriptModel
processResources - Processes main resources.
processTestResources - Processes test resources.
startScripts - Creates OS specific scripts to run the project as a JVM application.

Rules
-----
Pattern: clean<TaskName>: Cleans the output files of a task.
Pattern: build<ConfigurationName>: Assembles the artifacts of a configuration.
Pattern: upload<ConfigurationName>: Assembles and uploads the artifacts belonging to a configuration.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.6.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 911ms
1 actionable task: 1 executed
shynalprasad@Shynals-MacBook-Pro HTTP-JSON %
```

This application works with json file and includes a simple api while evaluating the data evaluation and uploads in to the GitHub.

Example 3
Serialization>UserXml

```
UserXml — -zsh — 80×24
Last login: Mon May 23 03:18:34 on ttys000
shynalprasad@Shynals-MacBook-Pro ~ % cd /Users/shynalprasad/Desktop/ser321exampl
es-master/Serialization/UserXml
shynalprasad@Shynals-MacBook-Pro UserXml % gradle run

> Task :run
Ready to export a user
Done exporting a user as xml to user.xml
Importing a user as xml from user.xml
Read user: I AM

BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 executed
shynalprasad@Shynals-MacBook-Pro UserXml %
```

A login for using username and password and first and last name. It is xml serialization user object. For it to be a xml serialization, it should have parameter less constructor.

3.3
https://github.com/ssprasad-e36king/ser321-summer2022-C-asurite
see Assignment 1 folder

3.4
AWS EC2 setup as second device
https://youtu.be/COC7O0DHLxU

```
JavaSimpleSock2 — -zsh — 80×24
Last login: Tue May 24 18:47:23 on ttys000
shynalprasad@Shynals-MBP JavaSimpleSock2 % gradle SocketClient -Phost=54.177.179
.70 -Pmessage=secret -Pnumber=5

> Task :SocketClient
Got it!

BUILD SUCCESSFUL in 1s
2 actionable tasks: 1 executed, 1 up-to-date
shynalprasad@Shynals-MBP JavaSimpleSock2 % gradle SocketClient -Phost=54.177.179
.70 -Pmessage=secret -Pnumber=5

> Task :SocketClient
Got it!

BUILD SUCCESSFUL in 1s
2 actionable tasks: 1 executed, 1 up-to-date
shynalprasad@Shynals-MBP JavaSimpleSock2 %
```

```
shynalprasad — ec2-user@ip-172-31-10-128:~ — ssh -i Histology.pem ec...
s-west-1.compute.amazonaws.com
Warning: Identity file Histology.pem not accessible: No such file or directory.
Last login: Wed May 25 01:42:47 2022 from 241.sub-174-194-131.myvzw.com

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 1 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-10-128 ~]$ ls
SockServer.class  SockServer.java
[ec2-user@ip-172-31-10-128 ~]$ javac SockServer.java
[ec2-user@ip-172-31-10-128 ~]$ java SockServer
Server ready for 3 connections
Server waiting for a connection
Received the String secret
Received the Integer 5
Server waiting for a connection
Received the String secret
Received the Integer 5
Server waiting for a connection
```

Part II: Neetworking
4.1 Explore the Data Link Layer with ARP
Step 1: Capture Stack Trace

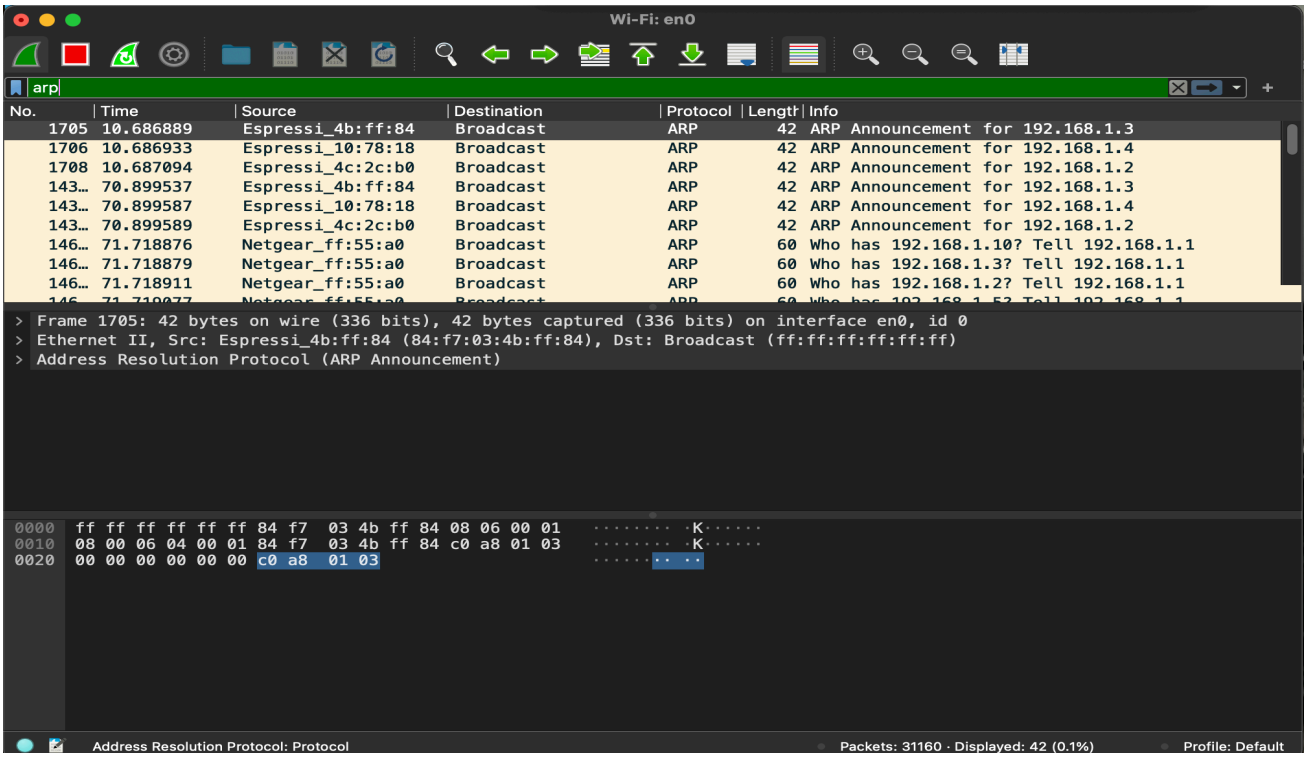
1: Ethernet address

```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        options=400<CHANNEL_IO>
        ether 78:4f:43:5e:85:ce
        inet 192.168.1.5 netmask 0xffffff00 broadcast 192.168.1.255
        media: autoselect
        status: active
```

2:IP address
Ipaddress

```
shynalprasad@Shynals-MacBook-Pro ~ % route -n get default
   route to: default
destination: default
       mask: default
    gateway: 192.168.1.1
  interface: en0
      flags: <UP,GATEWAY,DONE,STATIC,PRCLONING>
 recvpipe  sendpipe  ssthresh  rtt,msec    rttvar  hopcount      mtu     expire
       0         0         0         0         0         0      1500         0
shynalprasad@Shynals-MacBook-Pro ~ %
```

3: Wireshark
ARP filter



4: arp delete

Arp -a

```
shynalprasad@Shynals-MacBook-Pro ~ % arp -a
? (192.168.1.1) at 3c:37:86:ff:55:a0 on en0 ifscope [ethernet]
? (192.168.1.2) at 84:f7:3:4c:2c:b0 on en0 ifscope [ethernet]
? (192.168.1.3) at 84:f7:3:4b:ff:84 on en0 ifscope [ethernet]
? (192.168.1.4) at 34:b4:72:10:78:18 on en0 ifscope [ethernet]
? (192.168.1.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en0 ifscope permanent [ethernet]
shynalprasad@Shynals-MacBook-Pro ~ %
```

Arp -d

```
shynalprasad@Shynals-MacBook-Pro ~ % sudo arp -d 192.168.1.1
192.168.1.1 (192.168.1.1) deleted
shynalprasad@Shynals-MacBook-Pro ~ %
```

Arp -a

```
shynalprasad@Shynals-MacBook-Pro ~ % arp -a
? (192.168.1.1) at 3c:37:86:ff:55:a0 on en0 ifscope [ethernet]
? (192.168.1.2) at 84:f7:3:4c:2c:b0 on en0 ifscope [ethernet]
? (192.168.1.3) at 84:f7:3:4b:ff:84 on en0 ifscope [ethernet]
? (192.168.1.4) at 34:b4:72:10:78:18 on en0 ifscope [ethernet]
? (192.168.1.255) at ff:ff:ff:ff:ff:ff on en0 ifscope [ethernet]
? (224.0.0.251) at 1:0:5e:0:0:fb on en0 ifscope permanent [ethernet]
? (239.255.255.250) at 1:0:5e:7f:ff:fa on en0 ifscope permanent [ethernet]
shynalprasad@Shynals-MacBook-Pro ~ %
```

6: Arp traffic trace

arp

No.	Time	Source	Destination	Protocol	Length	Info
4542	28.105500	Espressi_4b:ff:84	Broadcast	ARP	42	ARP Announcement for 192.168.1.3
4591	28.515144	Espressi_10:78:18	Broadcast	ARP	42	ARP Announcement for 192.168.1.4
4592	28.515148	Espressi_4c:2c:b0	Broadcast	ARP	42	ARP Announcement for 192.168.1.2

Step 2: Inspect the Trace

```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: Netgear_ff:55:a0 (3c:37:86:ff:55:a0)
    Sender IP address: 192.168.1.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.1.11
```

```
Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Apple_5e:85:ce (78:4f:43:5e:85:ce)
    Sender IP address: 192.168.1.5
    Target MAC address: Netgear_ff:55:a0 (3c:37:86:ff:55:a0)
    Target IP address: 192.168.1.1
```

Step 3: Details of ARP over Ethernet

1. What opcode is used to indicate a request? What about a reply?
   Opcode for request is 1
   Opcode for reply is 2

2. How large is the ARP header for a request? What about for a reply? You will need to research this (hint: some sources define what belongs to the header differently, name which source you base your answer on)
   The header for a request and a reply is 28 bytes for both the request and reply.
   https://www.netometer.com/qa/arp.html

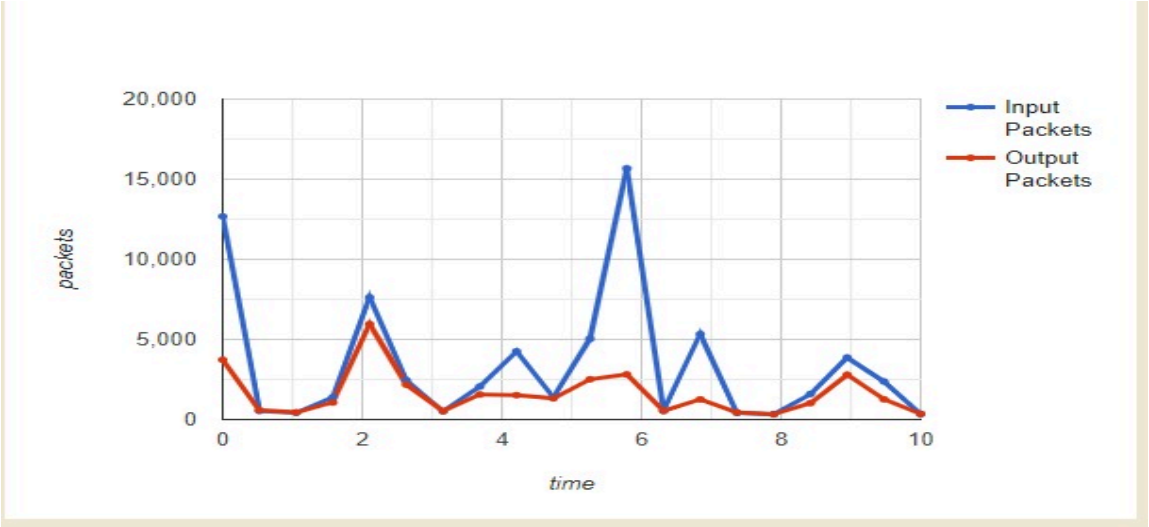
3. What value is carried on a request for the unknown target MAC address?
   00:00:00:00:00:00:

4. What Ethernet Type value indicates that ARP is the higher layer protocol?
   The Ethernet Type value for ARP is 0x806

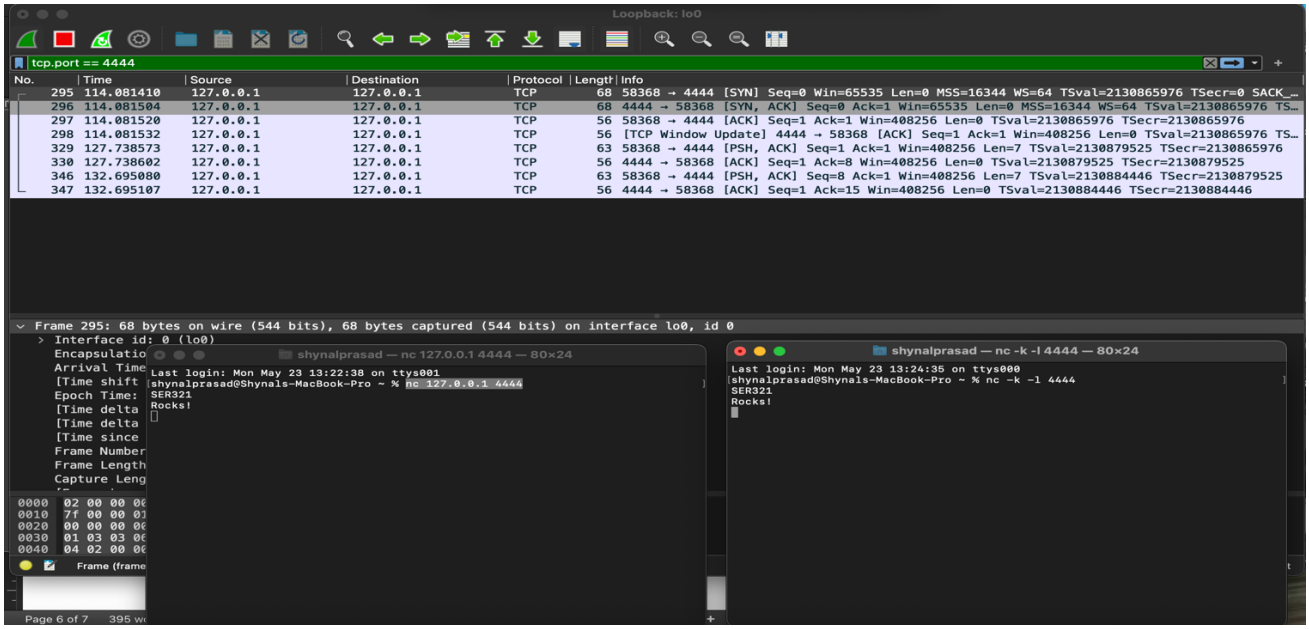
4.2 Understanding TCP network sockets

shynalprasad — -zsh — 80×24

```
shynalprasad@Shynals-MacBook-Pro ~ % netstat -n 30
            input        (Total)           output
   packets  errs      bytes    packets  errs      bytes colls
     12678     0   14453613       3731     0    1218332     0
       537     0     109645        576     0     161292     0
       423     0      63697        449     0     102641     0
      1394     0    1135156       1066     0     399302     0
      7621     0    5784100       5953     0    2041597     0
      2460     0    1581075       2182     0     581174     0
       511     0      84801        533     0     104017     0
      2059     0    1613870       1563     0     361798     0
      4260     0    5196905       1521     0     372158     0
      1406     0     820927       1324     0     459022     0
      5043     0    5386953       2511     0     732267     0
     15675     0   21654052       2815     0     664430     0
       541     0     153541        540     0     115066     0
      5332     0    6873060       2160     0     363912     0
       417     0      74075        441     0      97160     0
       335     0      37728        325     0      45347     0
      1594     0    1193050       1025     0     217564     0
      3869     0    3227224       2797     0     547550     0
      2374     0    2138770       1263     0     377434     0
       340     0      33384        351     0      48374     0
```



4.3
Step 1 TCP



a) Explain both the commands you used in detail. What did they actually do?
nc -k -l 3333 command was used to allow incoming traffic to receive data
nc 127.0.0.1 3333 used to send data using the 3333 port

b) How many frames were send back and forth to capture these 2 lines (Frames: 4 – I counted all frames that were sent)?
295 frames

c) How many packets were send back and forth to capture only those 2 lines?
4

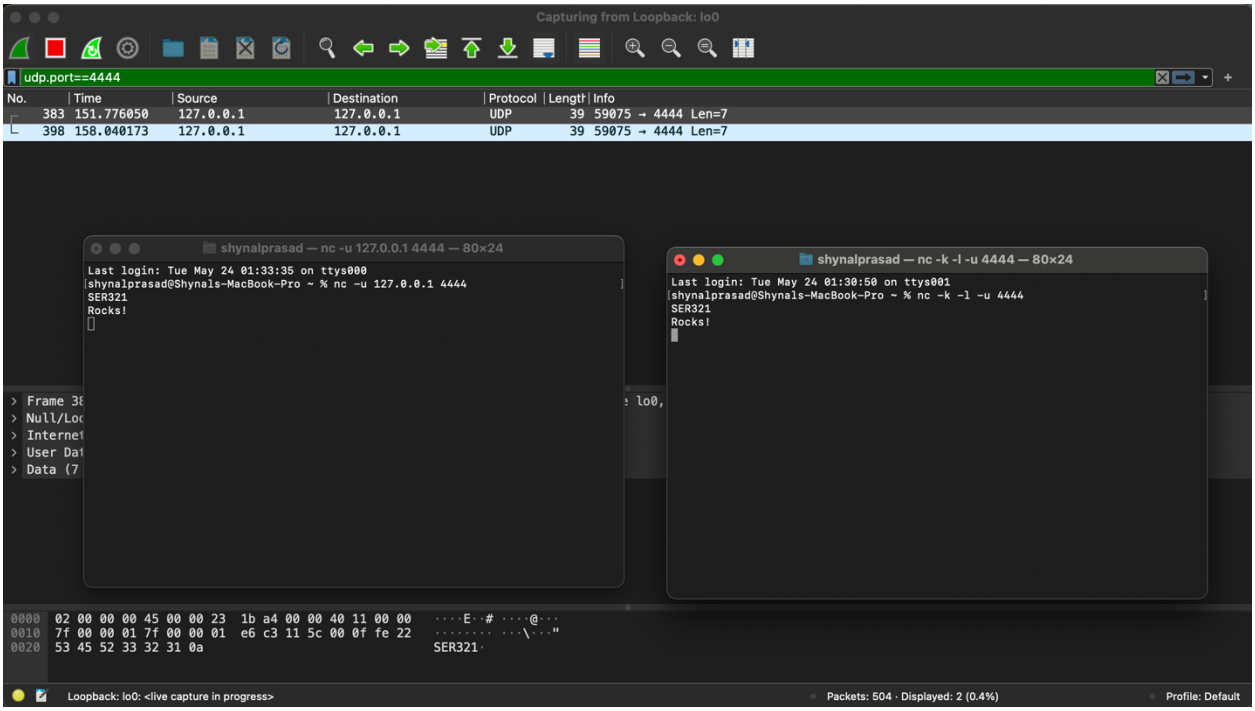
d) How many packets were needed to capture the whole "process" (starting the communication, ending the communication)?
8

e) How many bytes is the data (only the data) that was send?
357 bytes

f) How many total bytes went over the wire (back and forth) for the whole process?
486 bytes

g) How much overhead was there. Basically how many bytes was the whole process compared to the actually data that we did send.
129 bytes

Step 2 UDP



a) Explain both the commands you used in detail. What did they actually do?
nc -u 127.0.0.1 4444 were used to send data using the port 4444
nc -k -l -u 444 were used to receive data using the port 4444

b) How many frames were needed to capture those 2 lines?
2 frames were needed

c) How many packets were needed to capture those 2 lines?
398 packets were used

d) How many packets were needed to capture the whole "process" (starting the communication, ending the communication)?
398 packets were used

e) How many total bytes went over the wire?
39+39=78 bytes

f) How many bytes is the data (only the data) that was send?
78 bytes

4.4

Route 1

```
shynalprasad — traceroute www.asu.edu — 80×24
Last login: Tue May 24 13:36:15 on ttys001
shynalprasad@Shynals-MacBook-Pro ~ % traceroute www.asu.edu
traceroute to pantheon-systems.map.fastly.net (151.101.42.133), 64 hops max, 52
byte packets
 1  192.168.1.1 (192.168.1.1)  1.376 ms  0.909 ms  0.788 ms
 2  96.120.89.13 (96.120.89.13)  8.445 ms  8.092 ms  11.934 ms
 3  68.87.198.89 (68.87.198.89)  8.875 ms  8.664 ms  9.686 ms
 4  be-324-rar01.pleasanton.ca.sfba.comcast.net (162.151.79.133)  12.467 ms  12.
447 ms  10.080 ms
 5  be-398-ar01.hayward.ca.sfba.comcast.net (162.151.87.225)  11.107 ms  10.899
ms *
 6  be-36331-cs03.9greatoaks.ca.ibone.comcast.net (68.86.93.137)  13.938 ms
    be-36321-cs02.9greatoaks.ca.ibone.comcast.net (68.86.93.133)  12.228 ms
    be-36311-cs01.9greatoaks.ca.ibone.comcast.net (68.86.93.129)  12.517 ms
 7  be-2211-pe11.9greatoaks.ca.ibone.comcast.net (96.110.32.246)  13.142 ms
    be-2111-pe11.9greatoaks.ca.ibone.comcast.net (96.110.32.242)  13.971 ms  17.
047 ms
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
```

Route 2

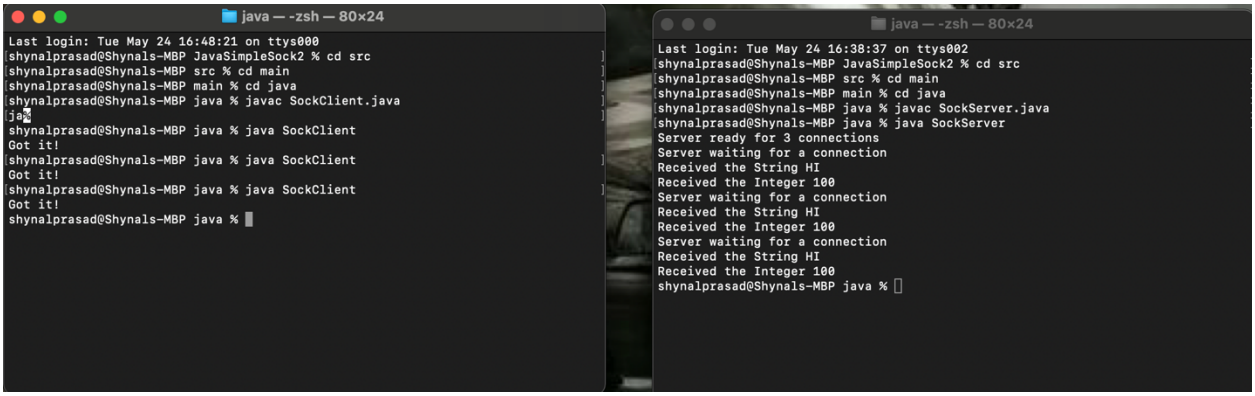
```
shynalprasad — traceroute www.asu.edu — 80×24
Last login: Tue May 24 13:55:33 on ttys000
shynalprasad@Shynals-MBP ~ % traceroute www.asu.edu
traceroute to pantheon-systems.map.fastly.net (151.101.42.133), 64 hops max, 52
byte packets
 1  192.168.43.1 (192.168.43.1)  6.759 ms  3.855 ms  2.952 ms
 2  225.sub-66-174-219.myvzw.com (66.174.219.225)  337.396 ms  328.490 ms  202.5
80 ms
 3  * * *
 4  * * *
 5  147.sub-69-83-165.myvzw.com (69.83.165.147)  68.650 ms  73.574 ms  36.582 ms
 6  * * *
 7  236.sub-69-83-160.myvzw.com (69.83.160.236)  43.227 ms  39.732 ms  35.654 ms
 8  183.sub-69-83-160.myvzw.com (69.83.160.183)  34.323 ms  33.454 ms  49.580 ms
 9  0.csi1.snvacanx-mse01-bb-su1.alter.net (140.222.6.94)  32.623 ms  50.340 ms
 48.319 ms
10  * * *
11  * * *
12  0.ae27.gw7.sjc7.alter.net (140.222.5.71)  52.563 ms  42.018 ms  30.066 ms
13  * * *
14  *
```

Which one is faster?
Route 1 is faster as you can see the time it takes in the route one is faster compared to route 2

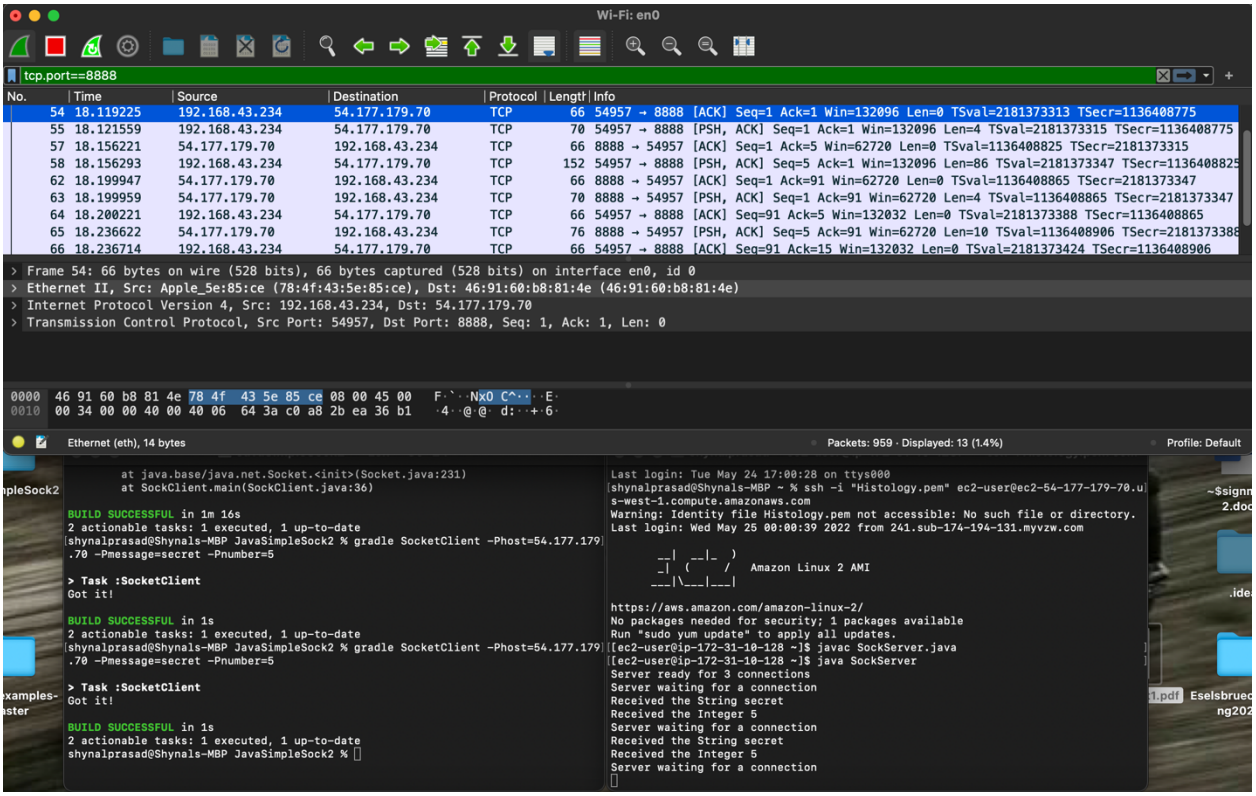
Which one has fewer hops?
Route 2 had fewer hops

4.5.1 Running things locally

https://youtu.be/yiYK4589Asc



4.5.2 Running server on AWS



Had to filter to the port 8888 to see the connection with Client locally and Server on AWS. To make this work I had to add port 8888 to have on bound traffic open on that port.

4.5.3

It is possible but you might have to do some changes. Might be with port setting or using the ip address of your router or your network.

4.5.4

The issue with running the server on local and client on the ec2 would be you would have to use the Ip address of the server for the client to connect to. And since the server is running locally, there could be a conflict with the ipaddress with your internet provider and your computer ipaddress. So that's why it would be a little harder but not impossible.