Brent Stockton
Ser 321

Assignment 1

2.2

JavaGradle runApp/runAppAgain

```
BUILD SUCCESSFUL in 609ms
1 actionable task: 1 executed
brentstockton@Natashas-iPhone JavaGradle % gradle runApp

> Task :runApp
1 * 2 = 2

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/7.2/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 executed
brentstockton@Natashas-iPhone JavaGradle % gradle runApp --args="5 7"

> Task :runApp
5 * 7 = 35

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/7.2/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 648ms
2 actionable tasks: 1 executed, 1 up-to-date
brentstockton@Natashas-iPhone JavaGradle % gradle runAppAgain -Pnum1=5 -Pnum2=10

> Task :runAppAgain
5 * 10 = 50

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/7.2/userguide/command_line_interface.html#sec:command_line_warnings
```

JavaGradle runFraction

```
brentstockton@brents-MacBook-Pro JavaGradle % vim build.gradle
brentstockton@brents-MacBook-Pro JavaGradle % gradle runFraction
Starting a Gradle Daemon, 1 incompatible Daemon could not be reused, use --status for details

> Task :runFraction
The fraction is: 1/3

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 pl
ugins.

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

BUILD SUCCESSFUL in 5s
2 actionable tasks: 2 executed
brentstockton@brents-MacBook-Pro JavaGradle %
```

JavaSimpleSock2

```
See https://docs.gradle.org/7.2/userguide/command_line_interface.html#sec:command_line
_warnings

BUILD SUCCESSFUL in 567ms
1 actionable task: 1 executed
rv%                                                    brentstockton@brents-MacBook-
Pro JavaSimpleSock2 % ServerSock
zsh: command not found: ServerSock
brentstockton@brents-MacBook-Pro 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 [2m 13s]
> :SocketServer
```

```
brentstockton@brents-MacBook-Pro JavaSimpleSock2 % gradle SockClient

> 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 they come from your own scripts or plugins.

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

BUILD SUCCESSFUL in 866ms
2 actionable tasks: 1 executed, 1 up-to-date
brentstockton@brents-MacBook-Pro JavaSimpleSock2 %
```

```
brentstockton@brents-MacBook-Pro JavaThreadSock % gradle
ThreadedSockServer -Pport=8088

> Task :ThreadedSockServer
Threaded server waiting for connects on port 8088
Threaded server connected to client-0
Threaded server waiting for connects on port 8088
From client 0 get string 1
<=========----> 75% EXECUTING [2m 20s]
> :ThreadedSockServer
```

```
[brentstockton@brents-MacBook-Pro JavaThreadSock % gradle ThreadedSockClient -Ppo
rt=8088 -Phost=localhost

> Task :ThreadedSockClient
Line number to get [0-4, empty to exit]>
<<=========----> 75% EXECUTING [8s]     If the timer has expired, the thread con
tinues
get>
<=========----> 75% EXECUTING [51s]
> :ThreadedSockClient
```

ThreadedSockServer

2.3

Gradle File in Assignment 1 folder

2.4

AWS Second System

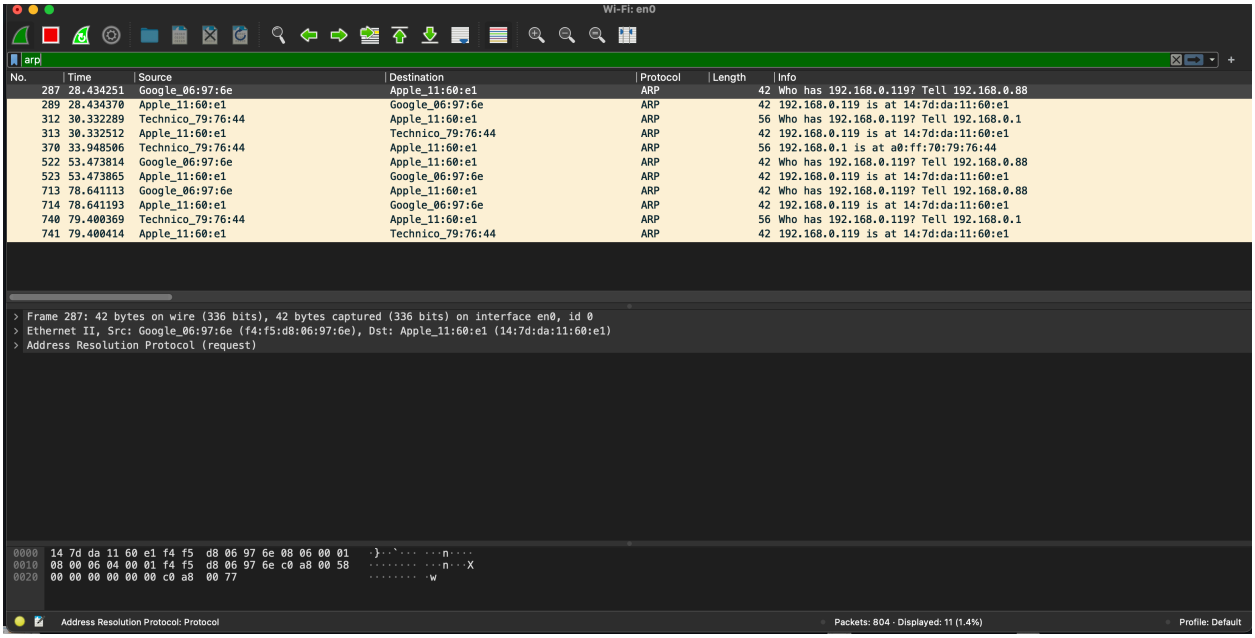
Youtube Video Link
https://youtu.be/df0DV4Rnh4A

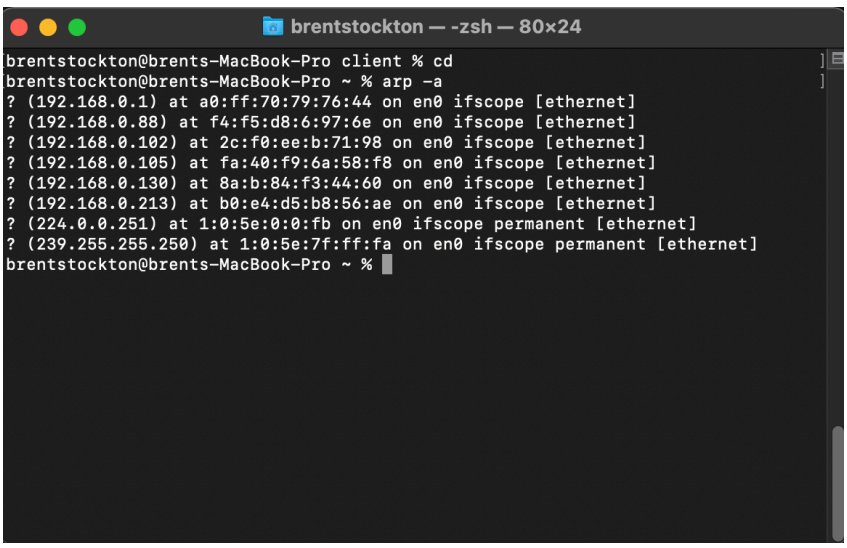
3.1

```
brentstockton@brents-MacBook-Pro ~ % netstat -r
Routing tables

Internet:
Destination        Gateway            Flags        Netif Expire
default            192.168.0.1        UGScg          en0
127                localhost          UCS            lo0
localhost          localhost          UH             lo0
169.254            link#10            UCS            en0      !
192.168.0          link#10            UCS            en0      !
192.168.0.1/32     link#10            UCS            en0      !
192.168.0.1        a0:ff:70:79:76:44  UHLWIir        en0   1184
192.168.0.12       18:b4:30:81:fc:f5  UHLWI          en0    867
192.168.0.88       f4:f5:d8:6:97:6e   UHLWIi         en0   1161
```

```
brentstockton@brents-MacBook-Pro Assignment1 % route -n get default
   route to: default
destination: default
       mask: default
    gateway: 192.168.0.1
  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
brentstockton@brents-MacBook-Pro Assignment1 %
```






```
brentstockton@brents-MacBook-Pro ~ % sudo arp -d 192.168.0.1 && arp -a
Password:
192.168.0.1 (192.168.0.1) deleted
```

Update Wire trace

arp

No.	Time	Source	Destination	Protocol	Length	Info
5	0.182988	Google_06:97:6e	Apple_11:60:e1	ARP	42	Who has 192.168.0.119? Tell 192.168.0.88
6	0.183069	Apple_11:60:e1	Google_06:97:6e	ARP	42	192.168.0.119 is at 14:7d:da:11:60:e1
347	13.1072…	Technico_79:76:44	Apple_11:60:e1	ARP	56	Who has 192.168.0.119? Tell 192.168.0.1
348	13.1072…	Apple_11:60:e1	Technico_79:76:44	ARP	42	192.168.0.119 is at 14:7d:da:11:60:e1
574	29.9666…	Google_b8:56:ae	Apple_11:60:e1	ARP	42	Who has 192.168.0.119? Tell 192.168.0.213
575	29.9667…	Apple_11:60:e1	Google_b8:56:ae	ARP	42	192.168.0.119 is at 14:7d:da:11:60:e1
1045	50.4845…	Technico_79:76:44	Apple_11:60:e1	ARP	56	Who has 192.168.0.119? Tell 192.168.0.1
1046	50.4845…	Apple_11:60:e1	Technico_79:76:44	ARP	42	192.168.0.119 is at 14:7d:da:11:60:e1
1085	55.0151…	Google_b8:56:ae	Apple_11:60:e1	ARP	42	Who has 192.168.0.119? Tell 192.168.0.213
1086	55.0152…	Apple_11:60:e1	Google_b8:56:ae	ARP	42	192.168.0.119 is at 14:7d:da:11:60:e1
1166	62.1070…	Technico_79:76:44	Apple_11:60:e1	ARP	56	192.168.0.1 is at a0:ff:70:79:76:44
1393	75.3777…	Google_06:97:6e	Apple_11:60:e1	ARP	42	Who has 192.168.0.119? Tell 192.168.0.88
1394	75.3778…	Apple_11:60:e1	Google_06:97:6e	ARP	42	192.168.0.119 is at 14:7d:da:11:60:e1
1405	80.0788…	Google_b8:56:ae	Apple_11:60:e1	ARP	42	Who has 192.168.0.119? Tell 192.168.0.213
1406	80.0789…	Apple_11:60:e1	Google_b8:56:ae	ARP	42	192.168.0.119 is at 14:7d:da:11:60:e1
1519	95.0286…	Technico_79:76:44	Apple_11:60:e1	ARP	56	Who has 192.168.0.119? Tell 192.168.0.1
1520	95.0286…	Apple_11:60:e1	Technico_79:76:44	ARP	42	192.168.0.119 is at 14:7d:da:11:60:e1

```
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_11:60:e1 (14:7d:da:11:60:e1)
  Sender IP address: 192.168.0.119
  Target MAC address: Google_06:97:6e (f4:f5:d8:06:97:6e)
  Target IP address: 192.168.0.88
```

```
Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: Google_06:97:6e (f4:f5:d8:06:97:6e)
  Sender IP address: 192.168.0.88
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.0.119
```

1. **What opcode is used to indicate a request? What about a reply?**

Request: Opcode: request (1)
Reply: Opcode: reply (1)

2. How large is the ARP header for a request? What about for a reply?

It is 28 bytes for both a request and a reply.

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

Value carried on a request for unknown target MAC address is usually all zeros/00:00:00:00:00

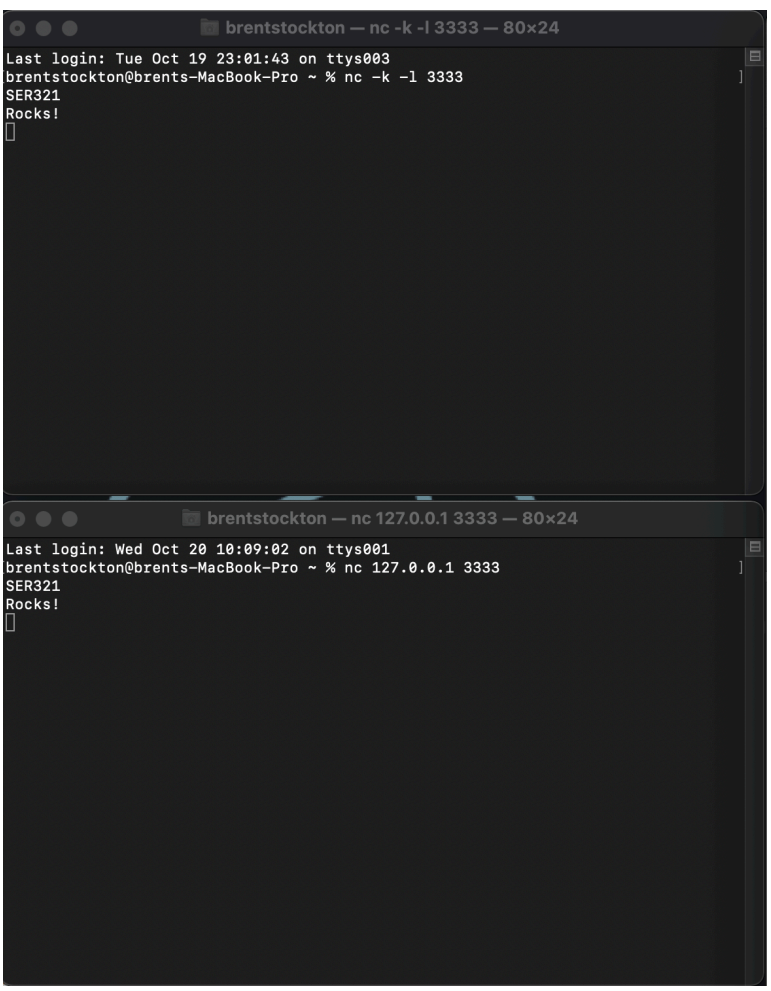
4. What Ethernet Type value indicates that ARP is the higher layer protocol?
Ethernet Type Value 0x806

3.2

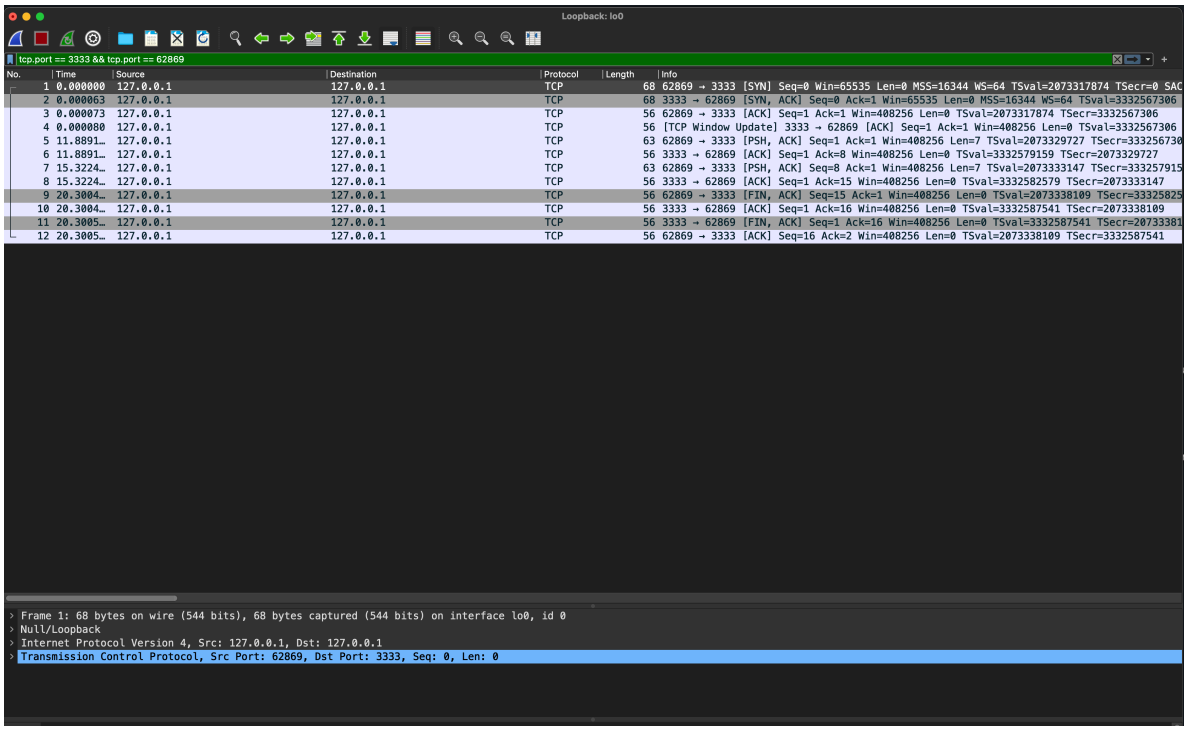
Could not figure out

3.3

TCP: Netstat Commands



TCP: Wireshark Capter



a) Explain both the command you used in detail? What did you actually do?
NC - Netcat is a command-line utility that reads and writes data across network connections. We used nc -k -l command flag to continue listening after disconnection and used nc (host) (port) to execute a port scan. We then passed two lines of string.

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

4

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

4

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

12

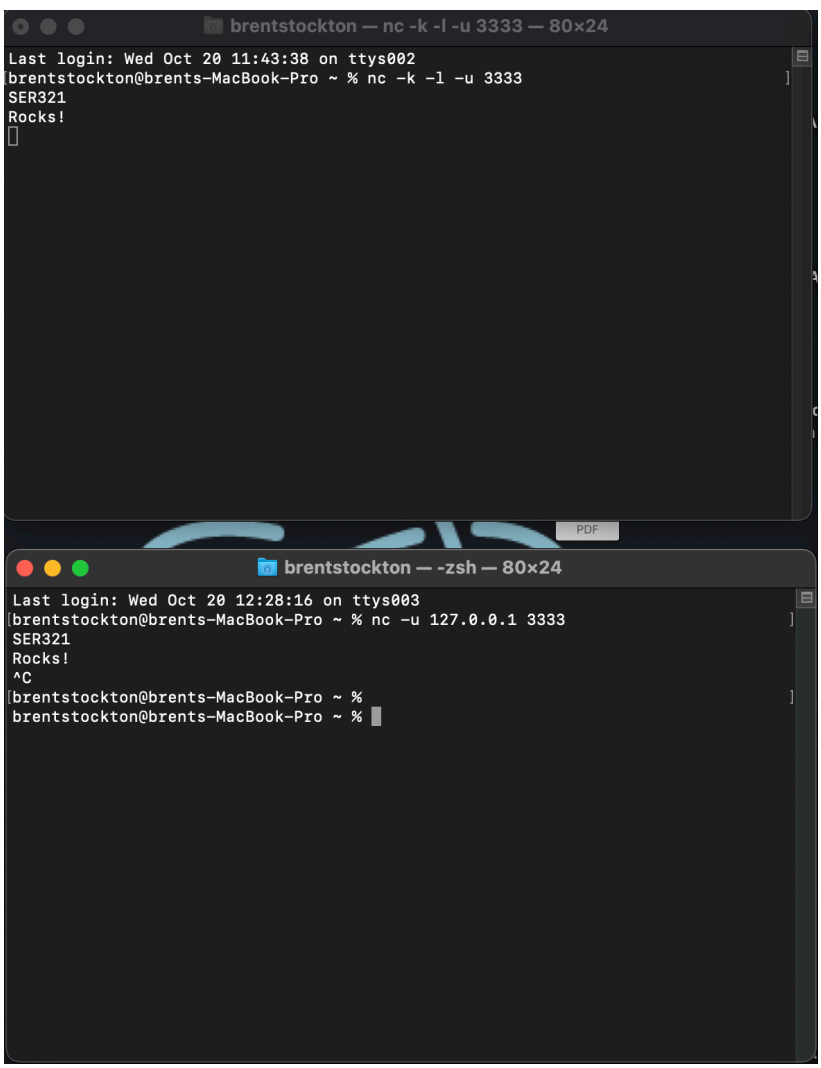
e) How many total bytes went over the wire? How much overhead was there (percent of bytes not in the above 2 lines)?

Total Bytes over wire: 710

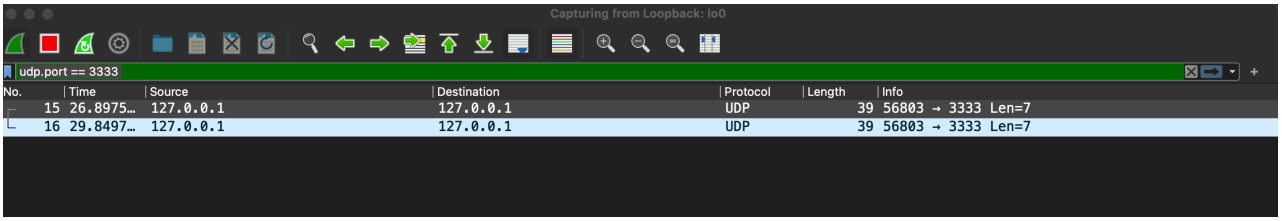
f) **How much overhead was there (basically the percentage of traffic that was not needed to send SER321 Rocks!)?**

Total Bytes: 710
Bytes to send SER321 Rocks!: 238
Bytes not involved: 710 - 238 = 472
Overhead: 66.5%

UDP: Netstat Commands



UDP: Wireshark Capture



a) Explain both the command you used in detail? What did you actually do?

NC - Netcat is a command-line utility that reads and writes data across network connections. We use nc -k -l to continue listening after disconnection and use the flag -u to specify UDP instead of TCP. We then used nc -u host port to execute a port scan. Then we passed the string.

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

2

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

2

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

2

e) How many total bytes went over the wire? How much overhead was there (percent of

bytes not in the above 2 lines)? Total Bytes over wire:

78

Overhead: 0%

f) 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 a lot less overhead than TCP. The reason is it is not connection oriented and does not provide sequencing, flow control and retransmission mechanisms where as TCP did exchange that.

TCP

```
63486 → 3333 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSva
3333 → 63486 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344
63486 → 3333 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=3356413
[TCP Window Update] 3333 → 63486 [ACK] Seq=1 Ack=1 Win=408256
63486 → 3333 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=7 TSval=335
```

UDP

```
56803 → 3333 Len=7
56803 → 3333 Len=7
```

<u>3.4</u>

<u>Route 1 (ASU Network)</u>

```
traceroute to pantheon-systems.map.fastly.net (199.232.154.133), 64 hops max, 52 byte packets
 1  192.168.0.1 (192.168.0.1)  3.880 ms  3.864 ms  2.810 ms
 2  ip68-104-128-1.ph.ph.cox.net (68.104.128.1)  14.240 ms  10.266 ms  7.042 ms
 3  wsip-184-178-205-236.ph.ph.cox.net (184.178.205.236)  7.679 ms  12.130 ms  11.568 ms
 4  100.120.101.38 (100.120.101.38)  9.235 ms  11.374 ms  12.162 ms
 5  100.120.100.0 (100.120.100.0)  24.869 ms  33.228 ms  8.806 ms
 6  68.1.0.187 (68.1.0.187)  8.568 ms  11.493 ms  11.967 ms
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * *
```

<u>Route 2 (Non-ASU Network)</u>

```
brentstockton@brents-MacBook-Pro ~ % traceroute www.asu.edu
traceroute: Warning: www.asu.edu has multiple addresses; using 151.101.2.133
traceroute to pantheon-systems.map.fastly.net (151.101.2.133), 64 hops max, 52 b
yte packets
 1  172.16.254.1 (172.16.254.1)  12.116 ms  6.072 ms  4.587 ms
 2  192.168.0.1 (192.168.0.1)  5.661 ms  4.871 ms  4.311 ms
 3  10.80.160.1 (10.80.160.1)  11.431 ms  12.854 ms  12.759 ms
 4  100.127.73.80 (100.127.73.80)  12.597 ms  12.903 ms  14.539 ms
 5  68.1.0.187 (68.1.0.187)  16.538 ms  14.111 ms  16.508 ms
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  *
```

3.4 Question 4 Answers

4. Now compare the 2 routes and answer the following questions a) Which is the fastest?

My network from Route 1 was the fastest.

b) Which has the fewest hops?

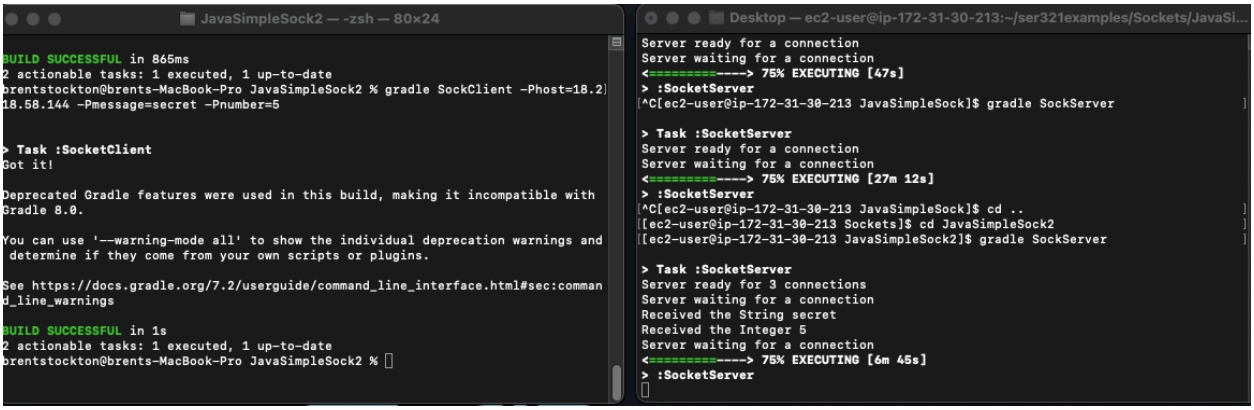
Route 2 had the fewest hops.

3.5

3.5.1 Video Link

https://youtu.be/qMUUSlPuksA

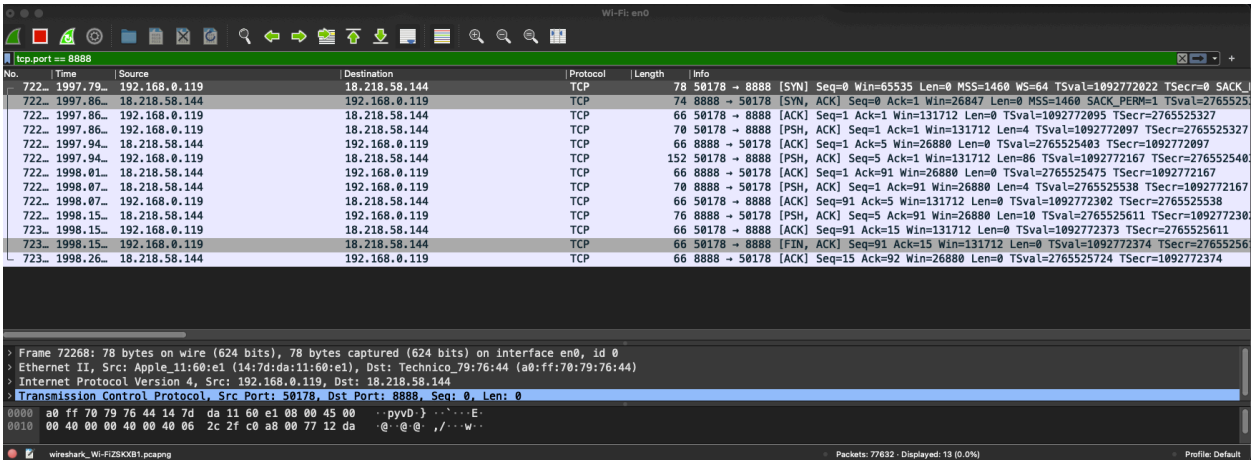
3.5.2 Commands



3.5.2 Answers

I ran JavaSimpleSock2 locally as well as on Local Client/Aws Server. Some things I noticed that were different when I ran the server on AWS was that the process completed in less frames but used more bytes to complete the message. Running Locally used more sends and Acknowledgments.

3.5.2 Wireshark Capture



3.5.3 Answer

It does not work without issues. For me the client continues to run and the local server waits for the connection without receiving the message and the number. For the AWS server I am using a Public IPv4 address that is accessed directly over the internet and is assigned to my network router through my ISP. My local client device has a private IP. This address lets devices on the same network communicate but make it more difficult for an external host to establish a connection. This is why it's easy to reach my server on AWS but not go in the other direction. To reach your server on a local network from an outside network you could use a VPN or possibly 3rd party software.