SER-321: Lower Layers
Terry Grant Simpson
10.19.2021

Some Setup and Examples

2.1: GitHub Invite: https://github.com/GTSimpson/ser321-fall2021-B-gtsimpso/invitations

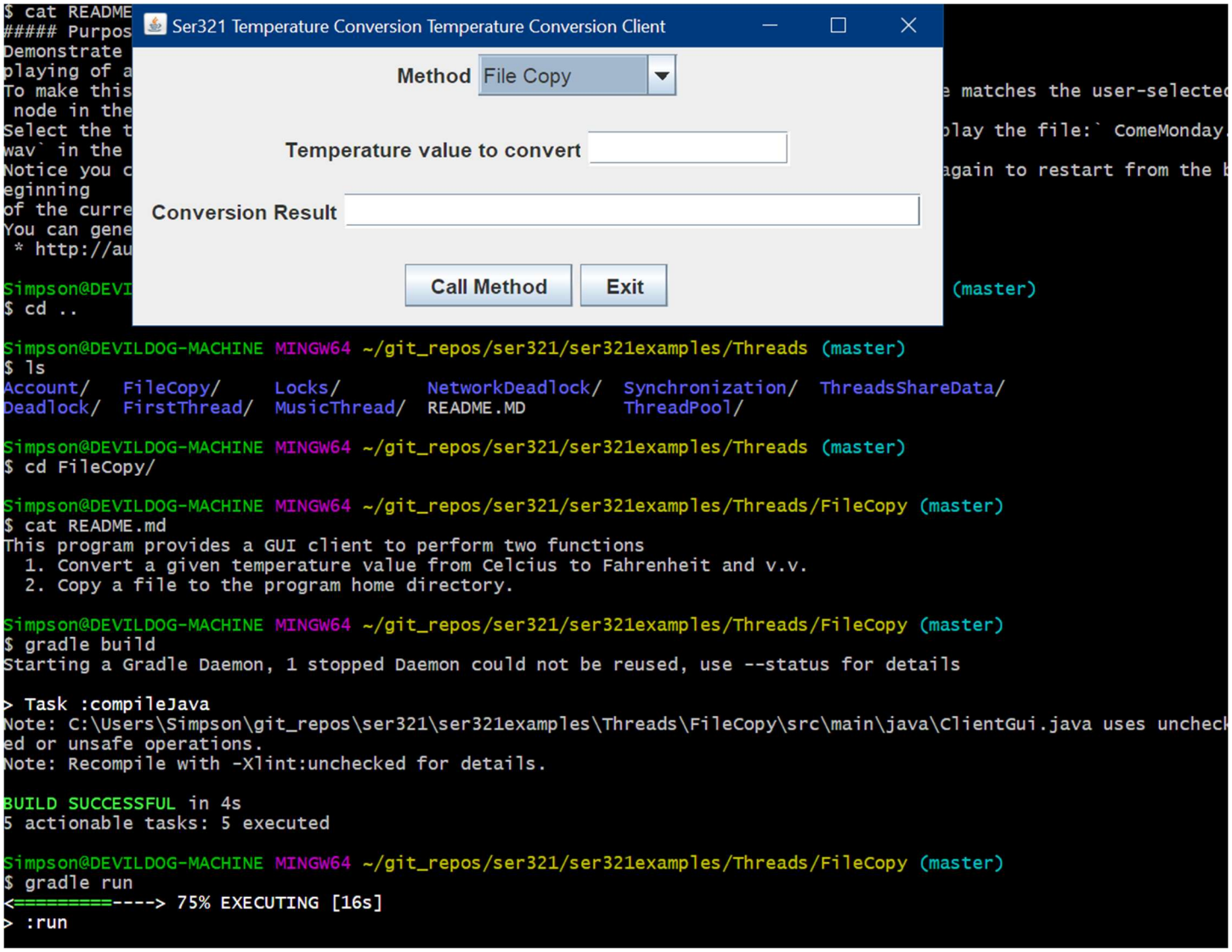
2.2: Example 1

```
Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads (master)
$ ls
Account/   FileCopy/     Locks/        NetworkDeadlock/  Synchronization/  ThreadsShareData/
Deadlock/  FirstThread/  MusicThread/  README.MD         ThreadPool/

Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads (master)
$ cd Account/

Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads/Account (master)
$ ls
README.md  build.gradle  src/

Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads/Account (master)
$ cat README.md
This is the example from your notes.

Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads/Account (master)
$ gradle build

BUILD SUCCESSFUL in 1s
5 actionable tasks: 5 executed

Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads/Account (master)
$ gradle run

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

BUILD SUCCESSFUL in 777ms
2 actionable tasks: 1 executed, 1 up-to-date
```

2.2: Example 2

```
Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads (master)
$ cd Deadlock

Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads/Deadlock (master)
$ ls
README.md  build.gradle  src/

Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads/Deadlock (master)
$ cat README.md
This program demonstrate how a deadlock can be created with synchronized methods:

 - https://docs.oracle.com/javase/tutorial/essential/concurrency/syncmeth.html
- https://docs.oracle.com/javase/tutorial/essential/concurrency/locksync.html

The key to why it locks can be found in this bullet point from the Tutorial:

- "When a thread invokes a synchronized method, it automatically acquires the intrinsic lock for that method's ob
ject and releases it when the method returns. The lock release occurs even if the return was caused by an uncaugh
t exception."

Since both the `bow()` and `bowback()` method are syncronized methods, they cannot
both be called on the same object at the same time, whichever is called first must
complete prior to the other executing.

The key to solving this is using a sycnronized statement rather than a synchronized
method. With this approach a seperate lock object can be shared and keep a deadlock
from occuring by not allowing the second bower to start before the first has finished.

A more sophisticated locking scheme can be accomplished with explicit Lock objects
and is described here:

- https://docs.oracle.com/javase/tutorial/essential/concurrency/newlocks.html



Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads/Deadlock (master)
$ gradle build

BUILD SUCCESSFUL in 882ms
5 actionable tasks: 5 executed

Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Threads/Deadlock (master)
$ gradle run

> Task :run
Alphonse: Gaston  has bowed to me!
Gaston: waiting to bow back
Gaston: Alphonse  has bowed to me!
Alphonse: waiting to bow back
<=========----> 75% EXECUTING [15s]
> :run
```

2.2: Example 3



2.3: Understanding Gradle

https://github.com/GTSimpson/ser321-fall2021-B-gtsimpso/invitations

2.4: Setup your second system

https://drive.google.com/file/d/1pvTGAQ_ecrNmnmWnKdfd7x5YifBUOdTQ/view?usp=sharing

Network

3.1 Explore the Data Link Layer with ARP

```
Simpson@DEVILDOG-MACHINE MINGW64 ~/Desktop
$ ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : Home
   Link-local IPv6 Address . . . . . : fe80::2825:79a0:a3f2:e51d%3
   IPv4 Address. . . . . . . . . . . : 192.168.0.5
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1
```

```
Simpson@DEVILDOG-MACHINE MINGW64 ~/Desktop
$ netstat -r
===========================================================================
Interface List
  3...10 7b 44 94 65 cb ......Realtek PCIe GBE Family Controller
  8...0a 00 27 00 00 08 ......VirtualBox Host-Only Ethernet Adapter
 15...9c fc e8 b7 d4 00 ......Microsoft Wi-Fi Direct Virtual Adapter
 11...9e fc e8 b7 d4 ff ......Microsoft Wi-Fi Direct Virtual Adapter #2
  4...9c fc e8 b7 d4 ff ......Intel(R) Wi-Fi 6 AX200 160MHz
  7...00 50 56 c0 00 01 ......VMware Virtual Ethernet Adapter for VMnet1
 14...00 50 56 c0 00 08 ......VMware Virtual Ethernet Adapter for VMnet8
 28...9c fc e8 b7 d5 03 ......Bluetooth Device (Personal Area Network) #2
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.0.1      192.168.0.5     25
```

Capturing from Ethernet

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

arp

No.	Time	Source	Destination	Protocol	Length	Info
986	29.245513	LGElectr[redacted]	Broadcast	ARP	60	Who has 192.168.0.51? T
987	29.245513	LGElectr[redacted]	Broadcast	ARP	60	Who has 192.168.0.13? T
988	29.245513	LGElect[redacted]	Broadcast	ARP	60	Who has 192.168.0.11? T
994	29.509233	ZyxelCo[redacted]	Broadcast	ARP	60	Who has 192.168.0.89? T
1005	29.917041	ASUSTekC[redacted]	Broadcast	ARP	42	Who has 192.168.0.10? T
1013	30.511288	ZyxelCom[redacted]	Broadcast	ARP	60	Who has 192.168.0.89? T

```
Simpson@DEVILDOG-MACHINE MINGW64 ~
$ arp -a

Interface: 192.168.0.5 --- 0x3
  Internet Address      Physical Address      Type
  192.168.0.1                                 dynamic
  192.168.0.35                                dynamic
  192.168.0.37                                dynamic
  192.168.0.255                               static
  224.0.0.22                                  static
  224.0.0.251                                 static
  224.0.0.252                                 static
  239.255.255.250                             static
  255.255.255.255                             static

Interface: 192.168.160.1 --- 0x7
  Internet Address      Physical Address      Type
  192.168.160.255                             static
  224.0.0.22                                  static
  224.0.0.251                                 static
  224.0.0.252                                 static
  239.255.255.250                             static

Interface: 192.168.56.1 --- 0x8
  Internet Address      Physical Address      Type
  192.168.56.255                              static
  224.0.0.22                                  static
  224.0.0.251                                 static
  224.0.0.252                                 static
  239.255.255.250                             static

Interface: 192.168.87.1 --- 0xe
  Internet Address      Physical Address      Type
  192.168.87.255                              static
  224.0.0.22                                  static
  224.0.0.251                                 static
  224.0.0.252                                 static
  239.255.255.250                             static
```

```
Simpson@DEVILDOG-MACHINE MINGW64 ~
$ arp -d 192.168.0.1 && arp -a

Interface: 192.168.0.5 --- 0x3
  Internet Address      Physical Address      Type
  192.168.0.35                                dynamic
  192.168.0.255                               static
  224.0.0.22                                  static
  239.255.255.250                             static

Interface: 192.168.160.1 --- 0x7
  Internet Address      Physical Address      Type
  192.168.160.255                             static
  224.0.0.22                                  static
  239.255.255.250                             static

Interface: 192.168.56.1 --- 0x8
  Internet Address      Physical Address      Type
  192.168.56.255                              static
  224.0.0.22                                  static
  239.255.255.250                             static

Interface: 192.168.87.1 --- 0xe
  Internet Address      Physical Address      Type
  192.168.87.255                              static
  224.0.0.22                                  static
  239.255.255.250                             static
```

85	2.090203	ZyxelCom[redacted]	Broadcast	ARP	60 Who has 192.168.0.5? Tell 192.168.0.1

```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: ASUSTekC_[redacted]
    Sender IP address: 192.168.0.5
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 192.168.0.1
```

```
Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: ASUSTekC[redacted])
    Sender IP address: 192.168.0.5
    Target MAC address: ZyxelCom_3[redacted]
    Target IP address: 192.168.0.1
```

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

Request opcode: request(1)

Reply opcode: reply(2)

*2. How large is the ARP header for a request? What about for a repl*y?

ARP header for a request and reply are 28 bytes.

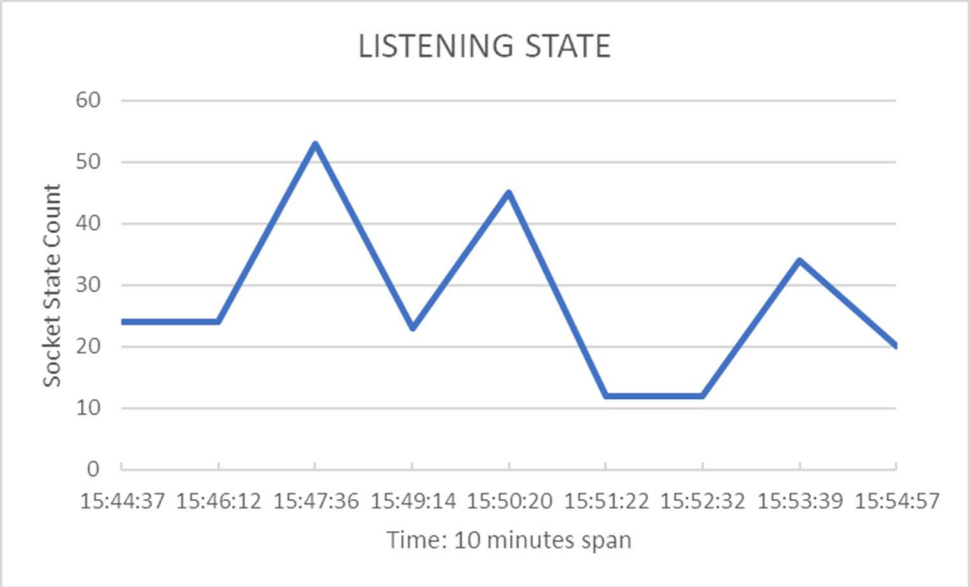
3. What value is carried on a request for the unknown target MAC address?

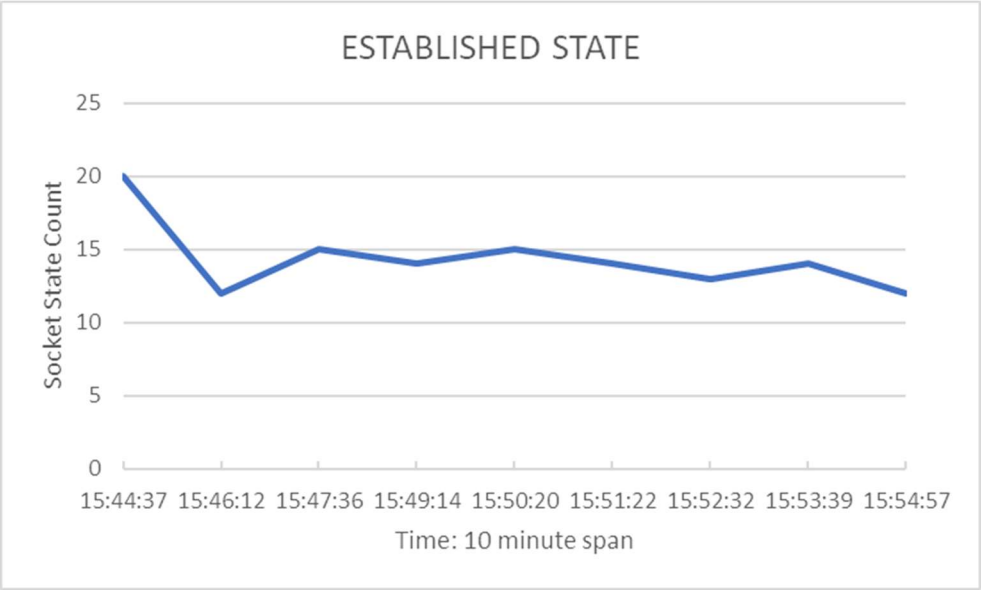
FFFF.FFFF.FFFF

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

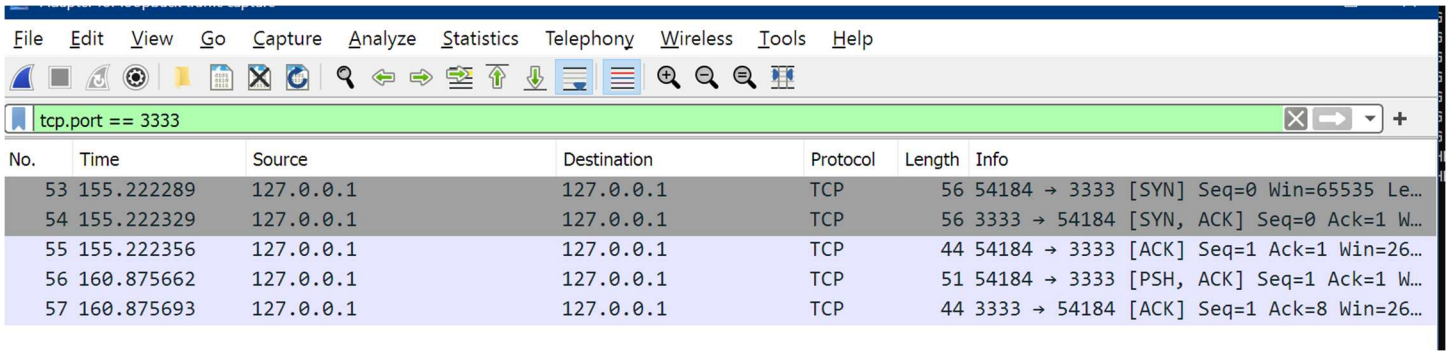
Type : ARP (0x0806)

3.2 Understanding TCP network sockets





3.3 Sniffing TCP/UDP traffic



No.	Time	Source	Destination	Protocol	Length	Info
53	155.222289	127.0.0.1	127.0.0.1	TCP	56	54184 → 3333 [SYN] Seq=0 Win=65535 Le…
54	155.222329	127.0.0.1	127.0.0.1	TCP	56	3333 → 54184 [SYN, ACK] Seq=0 Ack=1 W…
55	155.222356	127.0.0.1	127.0.0.1	TCP	44	54184 → 3333 [ACK] Seq=1 Ack=1 Win=26…
56	160.875662	127.0.0.1	127.0.0.1	TCP	51	54184 → 3333 [PSH, ACK] Seq=1 Ack=1 W…
57	160.875693	127.0.0.1	127.0.0.1	TCP	44	3333 → 54184 [ACK] Seq=1 Ack=8 Win=26…

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

The purpose of command *nc -k -l 3333* was to create a continuous listener on port *3333.* Option -k provide netcat to continuously listen for work over port 3333. Option -l allocated port 3333 as a listener and by default netcat ran the connection as a TCP.

How many frames were needed to capture those 2 lines?

Total of 4 frames were need to capture the 2 lines.

How many packets were needed to capture those 2 lines?

Total of 4 frames were need to capture the 2 lines.

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

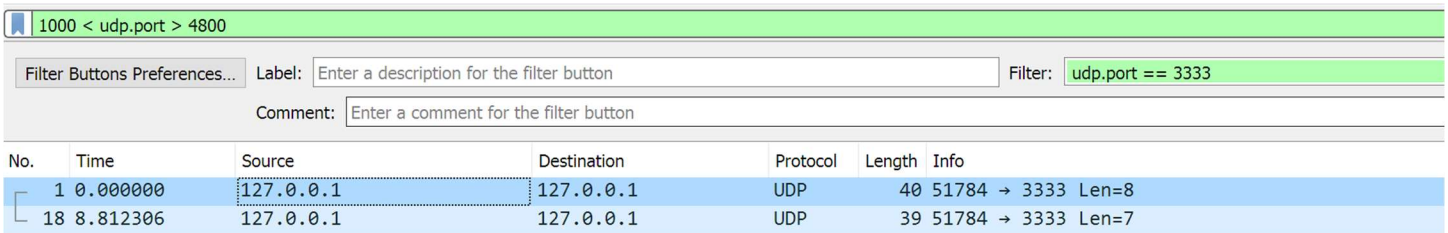
Total of 7 packets were need to complete the process.

How many total bytes went over the wire?

Total bytes over the wire equal 346 bytes.

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

About 55% of the total overhead.



Using the capture file (open it with Wireshark), answer the following questions

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

The purpose of command *nc -k -l -u 3333* was to create a continuous listener on port *3333 via UDP.* Option -k provide netcat to continuously listen for work over port 3333. Option -l allocated port 3333 as a listener and option -u ran a UDP connection.

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

Total of 2 frames were need to capture the 2 lines.

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

Total of 2 packets were need to capture the 2 lines.

d) How many packets were needed to capture the whole "process" (starting the

communication, ending the communication)?

Total of 2 packets were need to capture the entire process.

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 the wire equal 79 bytes. All (100%) of the overhead was a result of the two lines.

f) What is the difference in relative overhead between UDP and TCP and why?

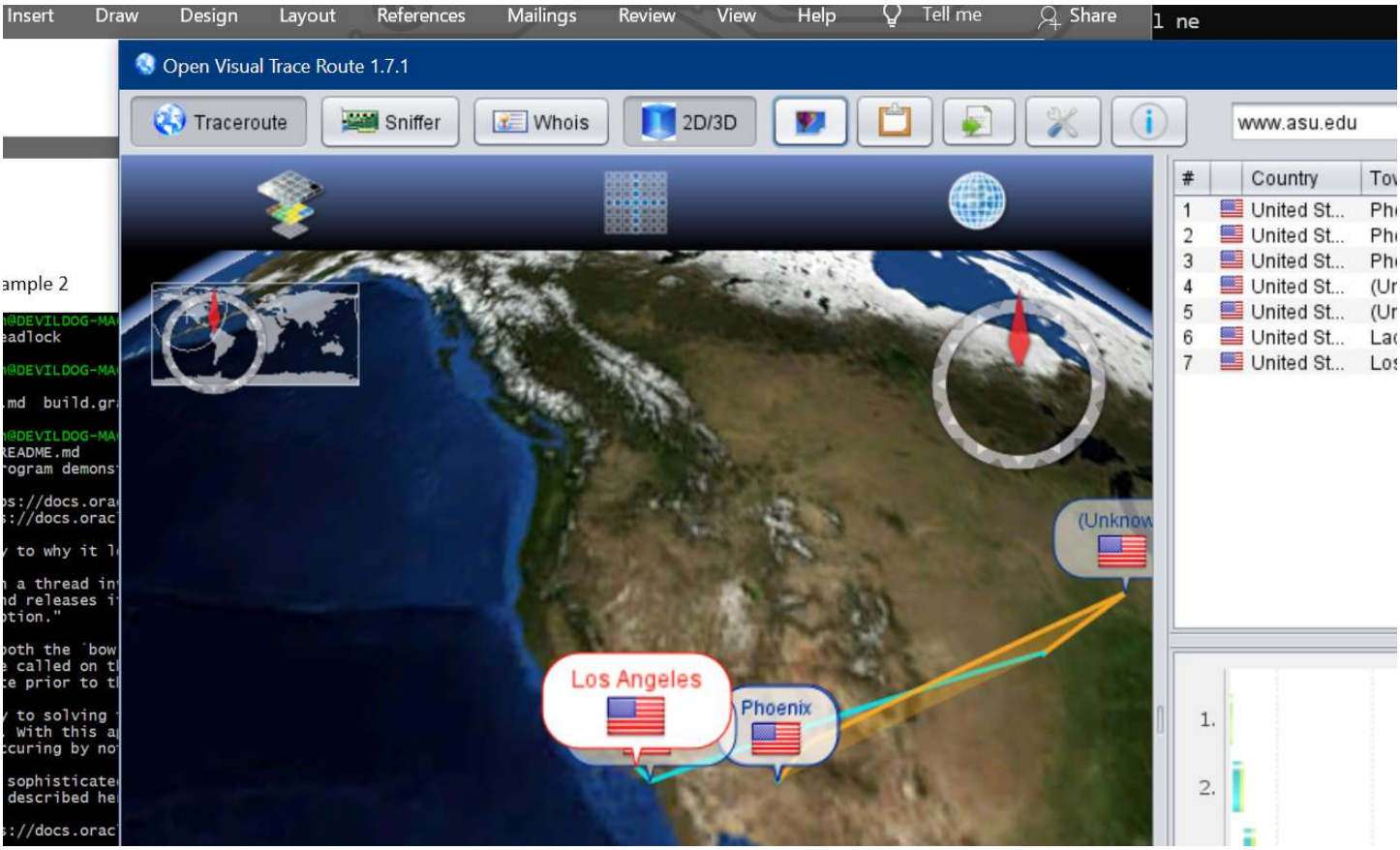
Specifically, what kind of information was exchanged in TCP that was not

exchanged in UDP? Show the relative parts of the packet traces.

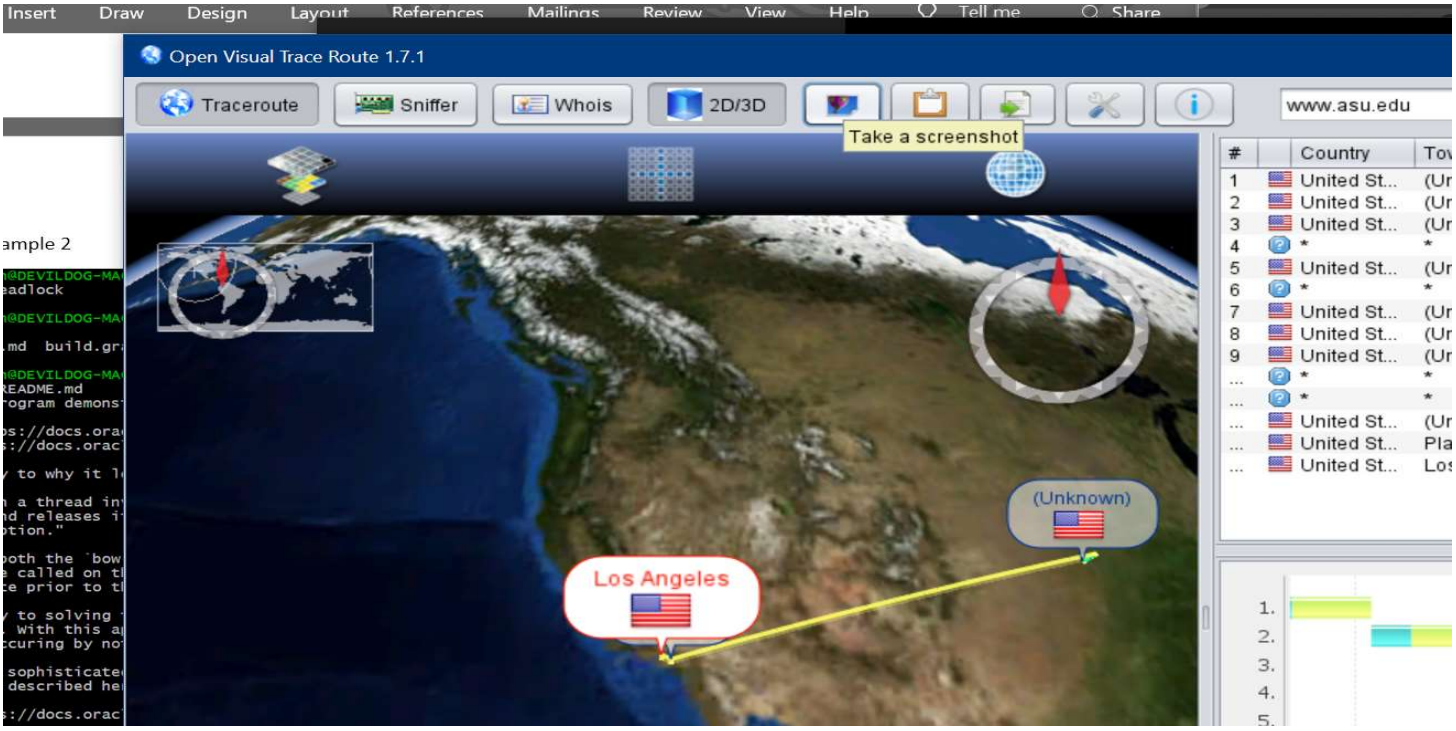
TCP requires a connection, where as UDP is a connectionless protocol. That said, the UDP header doesn't have the nearly the number of fields that the TCP header has. TCP exchange information that is not include in a UDP exchange includes: Sequence number, Acknowledgment number, TCP data offset, Reserved data, Control flags, Window size, Urgent pointer, and TCP optional data. Relative parts of TCP and UDP communication include Source and Destination port number, and checksum field.

3.4 Internet Protocol (IP) Routing

Route 1



Route 2



Now compare the 2 routes and answer the following questions

a) Which is the fastest?

The fastest was route 1.

b) Which has the fewest hops?

Route 2 had the fewest hops.

<u>3.5.1 Running client servers locally</u>

https://drive.google.com/file/d/1mbrM8FPF-IToHkmTKBD1TfQc51t-xPNA/view?usp=sharing

3.5.2 Running client server aws

```
Simpson@DEVILDOG-MACHINE MINGW64 ~/Documents/ASU/SER-321/setup
$ ssh -i ser321_key_pair.pem ec2-user@3.141.103.22
Last login: Thu Oct 21 00:40:20 2021 from 184.101.10.64

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

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-46-14 ~]$ ls
ser321examples
[ec2-user@ip-172-31-46-14 ~]$ cd ser321examples/
[ec2-user@ip-172-31-46-14 ser321examples]$ cd Sockets/SimpleSocket
-bash: cd: Sockets/SimpleSocket: No such file or directory
[ec2-user@ip-172-31-46-14 ser321examples]$ ls
Gradle  Middleware  Network  README.md  Serialization  Sockets  Threads
[ec2-user@ip-172-31-46-14 ser321examples]$ cd Sockets/
[ec2-user@ip-172-31-46-14 Sockets]$ ls
AdvancedCustomProtocol  GroupSerializeSocket    JavaThreadSock   SimpleCustomProtocol  Socket
Echo_C                  JavaSimpleSock          MulticastSocket  SimpleInterop         SocketState
Echo_Java               JavaSimpleSock2         PeerToPeer       SimplePeerToPeer      WebServer
Echo_Python             JavaSocketJSONProtocol  README.MD        SimpleWebServer
[ec2-user@ip-172-31-46-14 Sockets]$ cd JavaSimpleSock
[ec2-user@ip-172-31-46-14 JavaSimpleSock]$ ls
build.gradle  README.md  src
[ec2-user@ip-172-31-46-14 JavaSimpleSock]$ gradle SocketServer
Starting a Gradle Daemon (subsequent builds will be faster)

> Task :SocketServer
Server ready for a connection
Server waiting for a connection
<=========----> 75% EXECUTING [18s]
> :SocketServer
```

```
[ec2-user@ip-172-31-46-14 JavaSimpleSock]$ gradle SockServer

> Task :SocketServer
Server ready for a connection
Server waiting for a connection
Received the String Grant
Received the Integer 1
Received the String Grant 22
Received the Integer 2
Received the String Grant 33
Received the Integer 3
Received the String grant sss
<=========----> 75% EXECUTING [1m 33s]
> :SocketServer
```

```
MINGW64:/c/Users/Simpson/git_repos/ser321/ser321examples/Sockets/JavaSimpleSock

Simpson@DEVILDOG-MACHINE MINGW64 ~/git_repos/ser321/ser321examples/Sockets/JavaSimpleSock (master)
$ gradle SockClient

> Task :SocketClient
Please enter a String to send to the Server (enter "exit" to quit"):
<<=========----> 75% EXECUTING [4s]                                  Please enter a Number to send to the Server
(enter 0 to quit"): EXECUTING [5s]
<=========----> 75% EXECUTING [6s]                                                                          1
 and Grant ... Got it!ECUTING [7s]
Please enter a String to send to the Server (enter "exit" to quit"):
<====<=<=========----> 75% EXECUTING [10s]                           Please enter a Number to send to the Server
(enter 0 to quit"): EXECUTING [11s]
<<=========----> 75% EXECUTING [13s]                                                                        2
 and Grant 22 ... Got it!TING [14s]
Please enter a String to send to the Server (enter "exit" to quit"):
<=<===<=========----> 75% EXECUTING [16s]                            Please enter a Number to send to the Server
(enter 0 to quit"): EXECUTING [17s]
<<=========----> 75% EXECUTING [24s]                                                                        3
 and Grant 33 ... Got it!TING [25s]
Please enter a String to send to the Server (enter "exit" to quit"):
<====<==<=========----> 75% EXECUTING [1m 0s]                        Please enter a Number to send to the Server
(enter 0 to quit"): EXECUTING [1m 1s]
<=========----> 75% EXECUTING [1m 26s]
```

Needed to change the SockClient to IP address of AWS.