College of Engineering,Trivandrum

Experiment 13

Network Programming Lab

Author:
Alan Anto

Registration Number :
TVE16CS09

April 20, 2019

Contents

1 UDP in wireshark

1.1 Aim

Observing data transferred in client server communication using UDP using Wireshark and identifing the UDP datagram .

1.2 Theory

1.2.1 Wireshark

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.

Wireshark is cross-platform, using the Qt widget toolkit in current releases to implement its user interface, and using pcap to capture packets; it runs on Linux, macOS, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows. There is also a terminal-based (non-GUI) version called TShark. Wireshark, and the other programs distributed with it such as TShark, are free software, released under the terms of the GNU General Public License.Wireshark is very similar to tcpdump, but has a graphical front-end, plus some integrated sorting and filtering options.

1.2.2 Getting Wireshark

You can download Wireshark for Windows or macOS from its official website. If you're using Linux or another UNIX-like system, you'll probably find Wireshark in its package repositories. For example, if you're using Ubuntu, you'll find Wireshark in the Ubuntu Software Center.

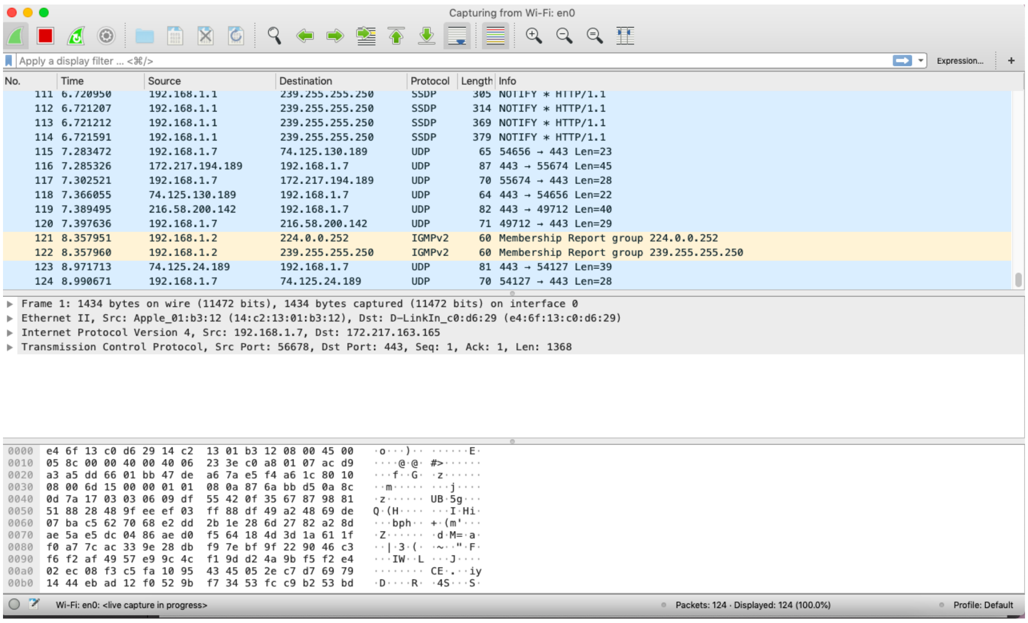
1.2.3 UDP Protocol

UDP stands for User Datagram Protocol. The UDP protocol works similarly to TCP, but it throws all the error-checking stuff out.When using UDP, packets are just sent to the recipient. The sender will not wait to make sure the recipient received the packet, it will just continue sending the next packets.There is no guarantee that all the packets are delivered and there is no way to ask for a packet again if it misses out, but losing all this overhead means the computers can communicate more quickly.

In UDP, the client does not form a connection with the server like in TCP and instead just sends a datagram. Similarly, the server need not accept a connection and just waits for datagrams to arrive. Datagrams upon arrival contain the address of sender which the server uses to send data to the correct client.

1.2.4 Capturing packets

After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface. For example, if you want to capture traffic on your wireless network, click your wireless interface.



```
No.   Time       Source           Destination      Protocol Length Info
  111 6.720950   192.168.1.1      239.255.255.250  SSDP      305 NOTIFY * HTTP/1.1
  112 6.721207   192.168.1.1      239.255.255.250  SSDP      314 NOTIFY * HTTP/1.1
  113 6.721212   192.168.1.1      239.255.255.250  SSDP      369 NOTIFY * HTTP/1.1
  114 6.721591   192.168.1.1      239.255.255.250  SSDP      379 NOTIFY * HTTP/1.1
  115 7.283472   192.168.1.7      74.125.130.189   UDP        65 54656 → 443 Len=23
  116 7.285326   172.217.194.189  192.168.1.7      UDP        87 443 → 55674 Len=45
  117 7.302521   192.168.1.7      172.217.194.189  UDP        70 55674 → 443 Len=28
  118 7.366055   74.125.130.189   192.168.1.7      UDP        64 443 → 54656 Len=22
  119 7.389495   216.58.200.142   192.168.1.7      UDP        82 443 → 49712 Len=40
  120 7.397636   192.168.1.7      216.58.200.142   UDP        71 49712 → 443 Len=29
  121 8.357951   192.168.1.2      224.0.0.252      IGMPv2     60 Membership Report group 224.0.0.252
  122 8.357960   192.168.1.2      239.255.255.250  IGMPv2     60 Membership Report group 239.255.255.250
  123 8.971713   74.125.24.189    192.168.1.7      UDP        81 443 → 54127 Len=39
  124 8.990671   192.168.1.7      74.125.24.189    UDP        70 54127 → 443 Len=28
```

```
▶ Frame 1: 1434 bytes on wire (11472 bits), 1434 bytes captured (11472 bits) on interface 0
▶ Ethernet II, Src: Apple_01:b3:12 (14:c2:13:01:b3:12), Dst: D-LinkIn_c0:d6:29 (e4:6f:13:c0:d6:29)
▶ Internet Protocol Version 4, Src: 192.168.1.7, Dst: 172.217.163.165
▶ Transmission Control Protocol, Src Port: 56678, Dst Port: 443, Seq: 1, Ack: 1, Len: 1368
```

```
0000  e4 6f 13 c0 d6 29 14 c2  13 01 b3 12 08 00 45 00   ·o···)·· ······E·
0010  05 8c 00 00 40 00 40 06  23 3e c0 a8 01 07 ac d9   ····@·@· #>······
0020  a3 a5 dd 66 01 bb 47 de  a6 7a e5 f4 a6 1c 80 10   ···f··G· ·z······
0030  08 00 6d 15 00 00 01 01  08 0a 87 6a bb d5 0a 8c   ··m····· ···j····
0040  0d 7a 17 03 03 06 09 df  55 42 0f 35 67 87 98 81   ·z······ UB·5g···
0050  51 88 28 48 9f ee ef 03  ff 88 df 49 a2 48 69 de   Q·(H···· ···I·Hi·
0060  07 ba c5 62 70 68 e2 dd  2b 1e 28 6d 27 82 a2 8d   ···bph·· +·(m'···
0070  ae 5a e5 dc 04 86 ae d0  f5 64 18 4d 3d 1a 61 1f   ·Z······ ·d·M=·a·
0080  f0 a7 7c ac 33 9e 28 db  f9 7e bf 9f 22 90 46 c3   ··|·3·(· ·~··"·F·
0090  f6 f2 af 49 57 e9 9c 4c  f1 9d d2 4a 9b f5 f2 e4   ···IW··L ···J····
00a0  02 ec 08 f3 c5 fa 10 95  43 45 05 2e c7 d7 69 79   ········ CE·.··iy
00b0  14 44 eb ad 12 f0 52 9b  f7 34 53 fc c9 b2 53 bd   ·D····R· ·4S···S·
```

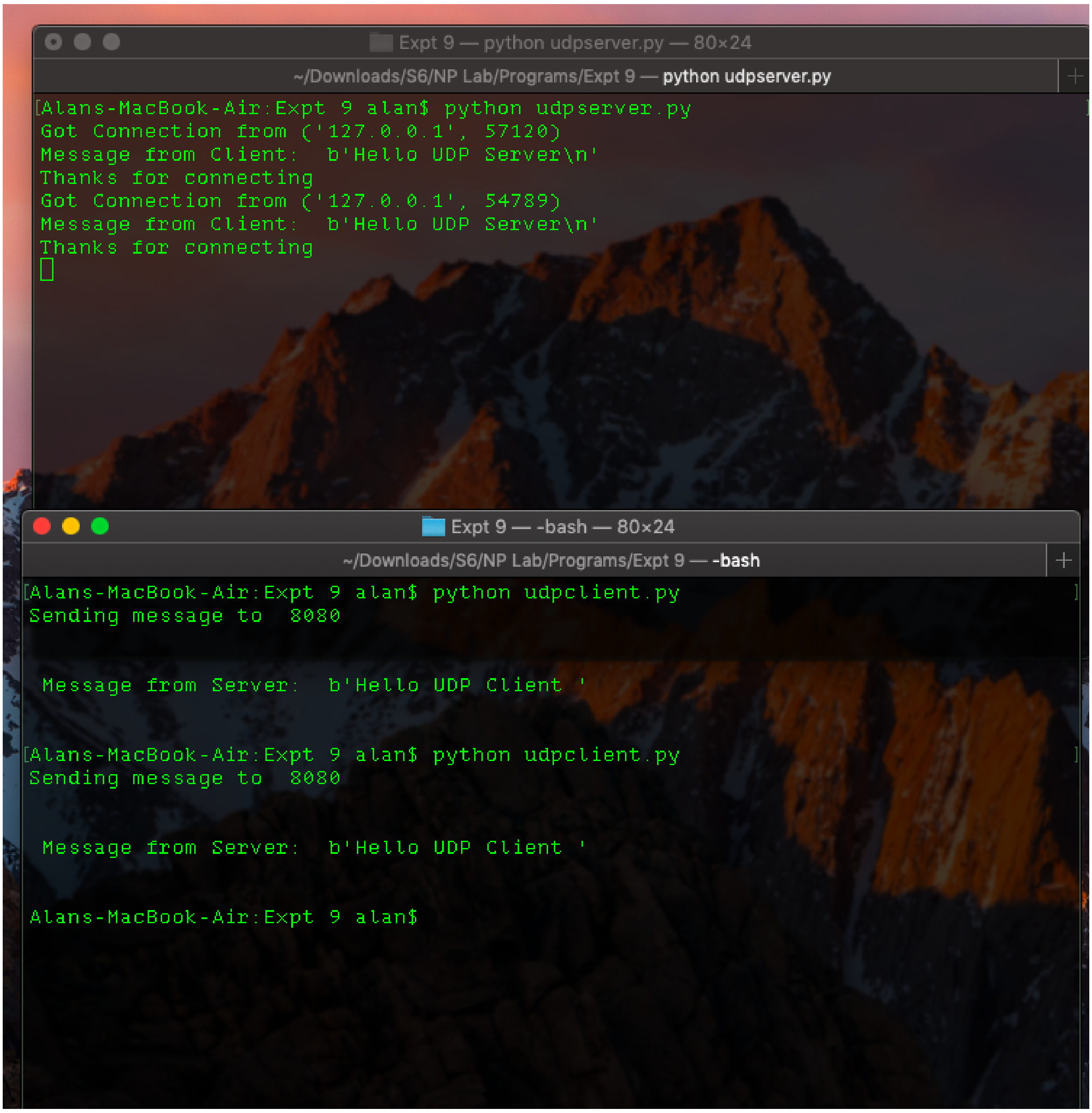
Wi-Fi: en0: <live capture in progress>   Packets: 124 · Displayed: 124 (100.0%)   Profile: Default

1.2.5 Filtering Packets

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type “dns” and you’ll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.

Here we apply UDP to the filter and run a UDP server and client. Communication done between the server and client is captured by wireshark.

2 Output



Expt 9 — python udpserver.py — 80×24
~/Downloads/S6/NP Lab/Programs/Expt 9 — python udpserver.py

```
[Alans-MacBook-Air:Expt 9 alan$ python udpserver.py
 Got Connection from ('127.0.0.1', 57120)
 Message from Client:  b'Hello UDP Server\n'
 Thanks for connecting
 Got Connection from ('127.0.0.1', 54789)
 Message from Client:  b'Hello UDP Server\n'
 Thanks for connecting
```

Expt 9 — -bash — 80×24
~/Downloads/S6/NP Lab/Programs/Expt 9 — -bash

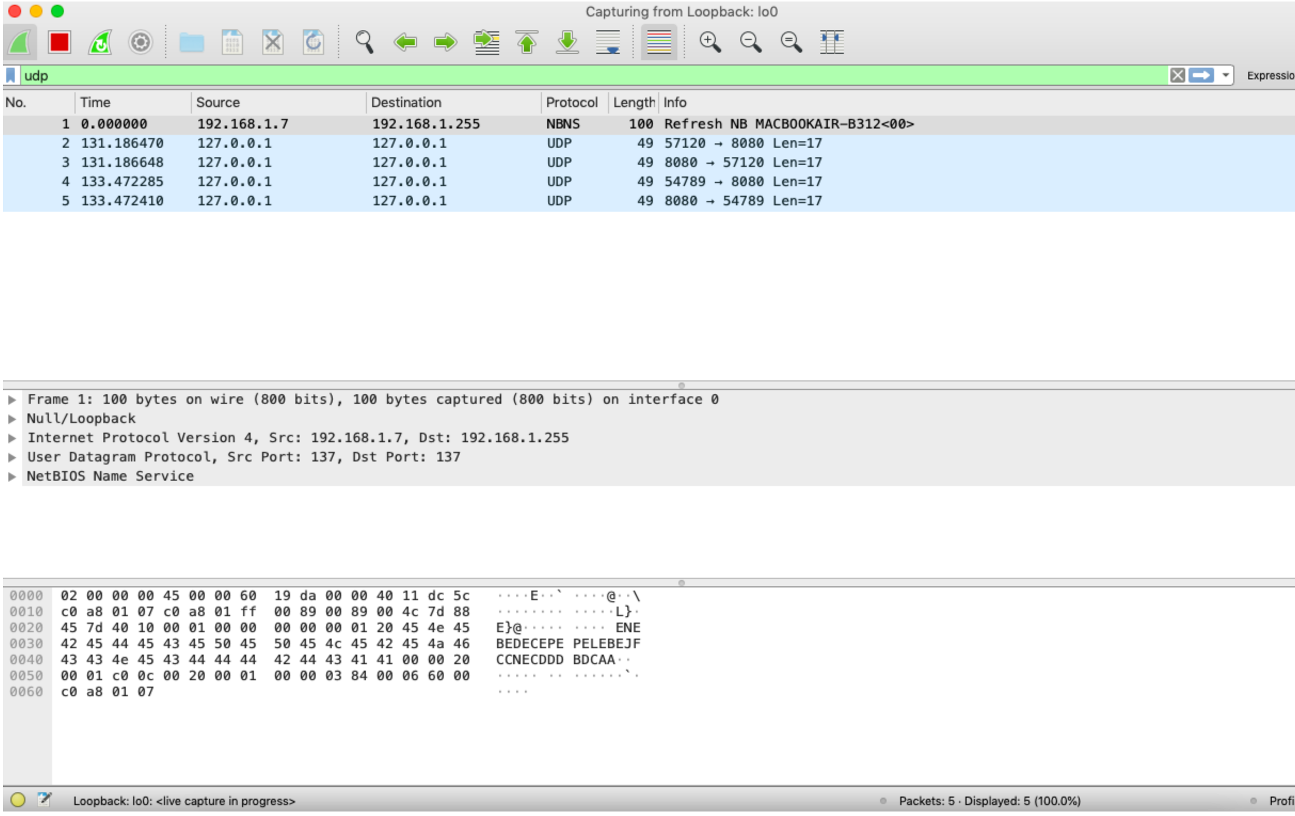
```
[Alans-MacBook-Air:Expt 9 alan$ python udpclient.py
Sending message to  8080


 Message from Server:  b'Hello UDP Client '

[Alans-MacBook-Air:Expt 9 alan$ python udpclient.py
Sending message to  8080


 Message from Server:  b'Hello UDP Client '


Alans-MacBook-Air:Expt 9 alan$
```



Capturing from Loopback: lo0

```
No.   Time         Source          Destination      Protocol  Length  Info
    1 0.000000     192.168.1.7     192.168.1.255    NBNS         100  Refresh NB MACBOOKAIR-B312<00>
    2 131.186470   127.0.0.1       127.0.0.1        UDP           49  57120 → 8080 Len=17
    3 131.186648   127.0.0.1       127.0.0.1        UDP           49  8080 → 57120 Len=17
    4 133.472285   127.0.0.1       127.0.0.1        UDP           49  54789 → 8080 Len=17
    5 133.472410   127.0.0.1       127.0.0.1        UDP           49  8080 → 54789 Len=17
```

```
▶ Frame 1: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface 0
▶ Null/Loopback
▶ Internet Protocol Version 4, Src: 192.168.1.7, Dst: 192.168.1.255
▶ User Datagram Protocol, Src Port: 137, Dst Port: 137
▶ NetBIOS Name Service
```

```
0000  02 00 00 00 45 00 00 60  19 da 00 00 40 11 dc 5c   ····E··` ····@··\
0010  c0 a8 01 07 c0 a8 01 ff  00 89 00 89 00 4c 7d 88   ········ ·····L}·
0020  45 7d 40 10 00 01 00 00  00 00 00 01 20 45 4e 45   E}@····· ···· ENE
0030  42 45 44 45 43 45 50 45  50 45 4c 45 42 45 4a 46   BEDECEPE PELEBEJF
0040  43 43 4e 45 43 44 44 44  42 44 43 41 41 00 00 20   CCNECDDD BDCAA··
0050  00 01 c0 0c 00 20 00 01  00 00 03 84 00 06 60 00   ····· ·· ······`·
0060  c0 a8 01 07                                        ····
```

Loopback: lo0: <live capture in progress>   Packets: 5 · Displayed: 5 (100.0%)   Profi

3 Result

Wireshark was installed and Packet transfer using UDP was observed .