1dv702 Assignment 1

neaguandrei47

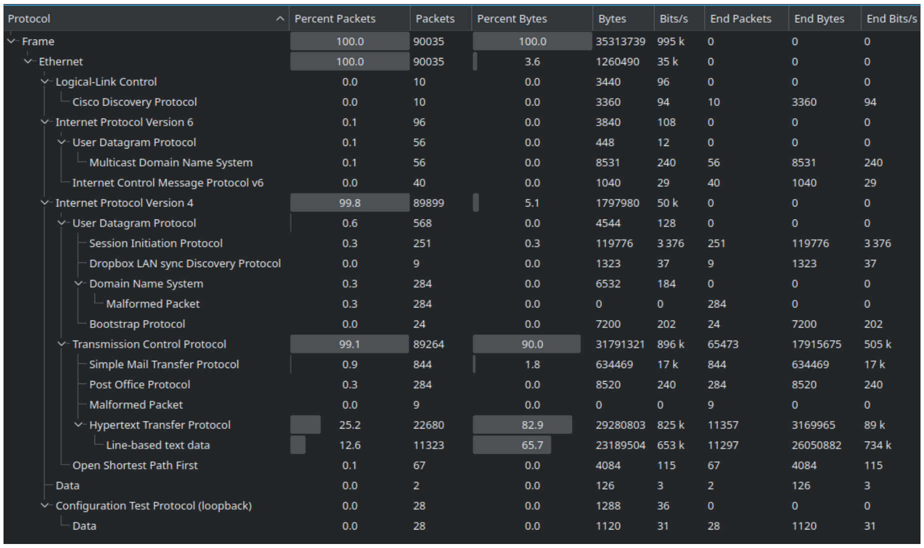
April 2020

1 3.1 Flow Identification

1.1 Bullet point 1

The capture lasted for 04:43 minutes, the OS is Mac OS X 10.8.2, the number of packets captured is 90035, the application used for dumping is Dumpcap 1.12.3 and lastly the average packet size is 392 Bytes.

1.2 Bullet point 2



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	90035	100.0	35313739	995 k	0	0	0
Ethernet	100.0	90035	3.6	1260490	35 k	0	0	0
Logical-Link Control	0.0	10	0.0	3440	96	0	0	0
Cisco Discovery Protocol	0.0	10	0.0	3360	94	10	3360	94
Internet Protocol Version 6	0.1	96	0.0	3840	108	0	0	0
User Datagram Protocol	0.1	56	0.0	448	12	0	0	0
Multicast Domain Name System	0.1	56	0.0	8531	240	56	8531	240
Internet Control Message Protocol v6	0.0	40	0.0	1040	29	40	1040	29
Internet Protocol Version 4	99.8	89899	5.1	1797980	50 k	0	0	0
User Datagram Protocol	0.6	568	0.0	4544	128	0	0	0
Session Initiation Protocol	0.3	251	0.3	119776	3 376	251	119776	3 376
Dropbox LAN sync Discovery Protocol	0.0	9	0.0	1323	37	9	1323	37
Domain Name System	0.3	284	0.0	6532	184	0	0	0
Malformed Packet	0.3	284	0.0	0	0	284	0	0
Bootstrap Protocol	0.0	24	0.0	7200	202	24	7200	202
Transmission Control Protocol	99.1	89264	90.0	31791321	896 k	65473	17915675	505 k
Simple Mail Transfer Protocol	0.9	844	1.8	634469	17 k	844	634469	17 k
Post Office Protocol	0.3	284	0.0	8520	240	284	8520	240
Malformed Packet	0.0	9	0.0	0	0	9	0	0
Hypertext Transfer Protocol	25.2	22680	82.9	29280803	825 k	11357	3169965	89 k
Line-based text data	12.6	11323	65.7	23189504	653 k	11297	26050882	734 k
Open Shortest Path First	0.1	67	0.0	4084	115	67	4084	115
Data	0.0	2	0.0	126	3	2	126	3
Configuration Test Protocol (loopback)	0.0	28	0.0	1288	36	0	0	0
Data	0.0	28	0.0	1120	31	28	1120	31

1.3 Bullet point 3

In total there are 5819 flows.

- Flow - F1. Destination IP 172.20.88.1 . TCP conversations, port 25 for SMTP. The sources are many IP addresses with a /16 subnet mask.

- Flow - F2. Destination IP 172.20.88.1 . TCP conversations, port 110 for POP3. The sources are many IP addresses with a /16 subnet mask.
- Flow - F3. Destination IP 172.20.88.1 . UDP conversations, port 53 for DNS with one IP 0.0.0.0 .
- Flow - F4. Destination IP 172.20.88.6 . TCP conversations, port 80 for HTTP with many IP addresses with a /24 subnet mask.
- Flow - F5. Source IP 10.7.5.1 is talking to 10.7.5.12 and 172.20.88.7. UDP conversations, port 5060 for SIP/SDP .
- Flow - F6. Destination IP address 194.174.88.52 . TCP conversations, port 8880 , this is a web server. It is talking to 10.100.1.5, 10.100.1.6 and 10.100.1.7 .

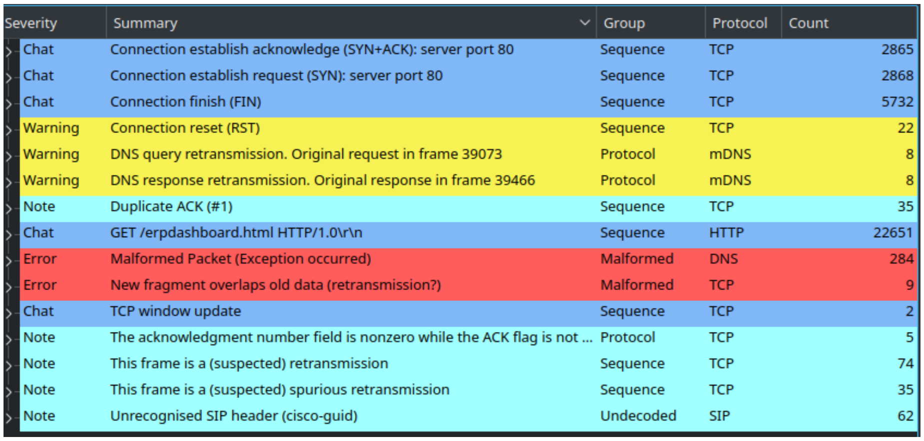
The reason why I picked these profiles is because these are the top applications. I omitted the FTP flow, on port 21 with the IP 172.20.88.5 because the traffic on this capture is very low. However if there is more FTP traffic on another capture it can be profiled later.

1.4 Bullet point 4

The following profiles were made because all of this is significant traffic. Coincidentally all the profiles use TCP.

1. Profile - P1. F1 and F2 fall under the same profile. The profile for Email.
2. Profile - P2. F4 has requests /erpdashboard/, this is very significant since it is one of the 2 main flows.
3. Profile - P3. F6, the smallest flow, it has requests about /currencyexchange/ and /stockmarket/.

1.5 Bullet point 5

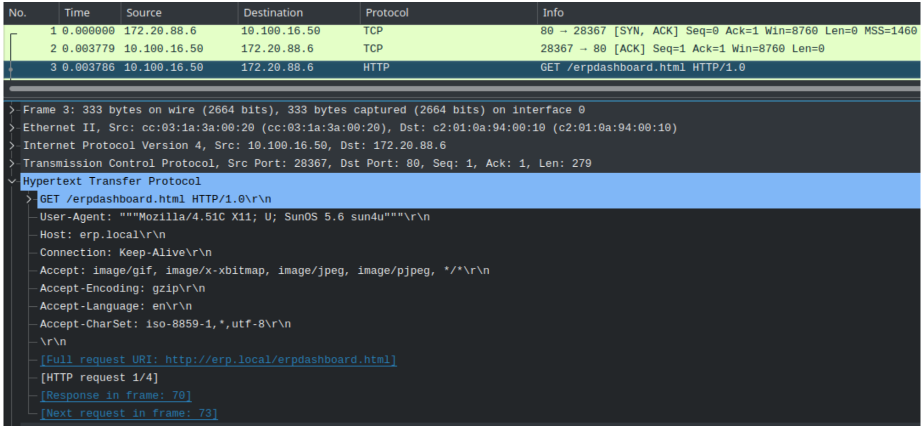


Severity	Summary	Group	Protocol	Count
Chat	Connection establish acknowledge (SYN+ACK): server port 80	Sequence	TCP	2865
Chat	Connection establish request (SYN): server port 80	Sequence	TCP	2868
Chat	Connection finish (FIN)	Sequence	TCP	5732
Warning	Connection reset (RST)	Sequence	TCP	22
Warning	DNS query retransmission. Original request in frame 39073	Protocol	mDNS	8
Warning	DNS response retransmission. Original response in frame 39466	Protocol	mDNS	8
Note	Duplicate ACK (#1)	Sequence	TCP	35
Chat	GET /erpdashboard.html HTTP/1.0\r\n	Sequence	HTTP	22651
Error	Malformed Packet (Exception occurred)	Malformed	DNS	284
Error	New fragment overlaps old data (retransmission?)	Malformed	TCP	9
Chat	TCP window update	Sequence	TCP	2
Note	The acknowledgment number field is nonzero while the ACK flag is not ...	Protocol	TCP	5
Note	This frame is a (suspected) retransmission	Sequence	TCP	74
Note	This frame is a (suspected) spurious retransmission	Sequence	TCP	35
Note	Unrecognised SIP header (cisco-guid)	Undecoded	SIP	62

Wireshark keeps track of any anomalies and other items of interest it finds in a capture file and shows them in the Expert Information dialog. The goal is to give you a better idea of uncommon or notable network behaviour and to let novice and expert users find network problems faster than manually scanning through the packet list. Blue is information, yellow are warnings, small errors, cyan are notable events, and red are serious errors.

1.6 Bullet point 6

Figure 1: HTTP enabled

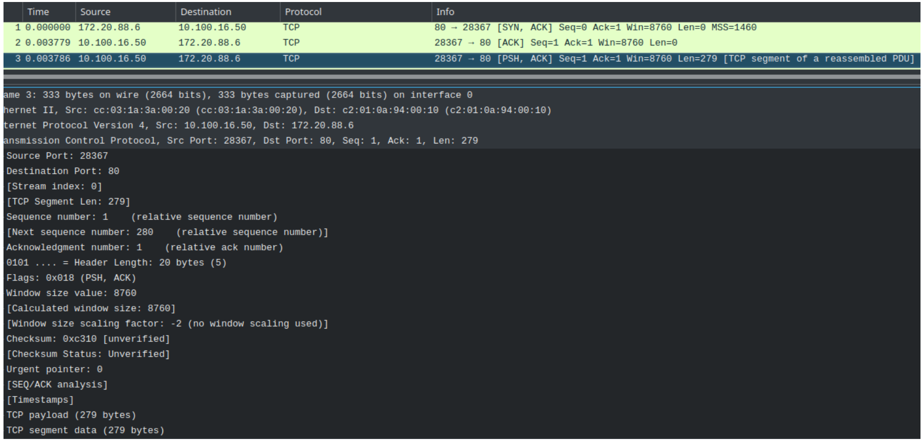


No.	Time	Source	Destination	Protocol	Info
1	0.000000	172.20.88.6	10.100.16.50	TCP	80 → 28367 [SYN, ACK] Seq=0 Ack=1 Win=8760 Len=0 MSS=1460
2	0.003779	10.100.16.50	172.20.88.6	TCP	28367 → 80 [ACK] Seq=1 Ack=1 Win=8760 Len=0
3	0.003786	10.100.16.50	172.20.88.6	HTTP	GET /erpdashboard.html HTTP/1.0

```
> Frame 3: 333 bytes on wire (2664 bits), 333 bytes captured (2664 bits) on interface 0
> Ethernet II, Src: cc:03:1a:3a:00:20 (cc:03:1a:3a:00:20), Dst: c2:01:0a:94:00:10 (c2:01:0a:94:00:10)
> Internet Protocol Version 4, Src: 10.100.16.50, Dst: 172.20.88.6
> Transmission Control Protocol, Src Port: 28367, Dst Port: 80, Seq: 1, Ack: 1, Len: 279
v Hypertext Transfer Protocol
    > GET /erpdashboard.html HTTP/1.0\r\n
      User-Agent: """Mozilla/4.51C X11; U; SunOS 5.6 sun4u"""\r\n
      Host: erp.local\r\n
      Connection: Keep-Alive\r\n
      Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*\r\n
      Accept-Encoding: gzip\r\n
      Accept-Language: en\r\n
      Accept-CharSet: iso-8859-1,*,utf-8\r\n
      \r\n
      [Full request URI: http://erp.local/erpdashboard.html]
      [HTTP request 1/4]
      [Response in frame: 70]
      [Next request in frame: 73]
```

After HTTP was disabled Wireshark does not recognize the HTTP protocol anymore, HTTP details are filtered and only the TCP traffic remains. Dissector is simply a protocol parser. Wireshark contains dozens of protocol dissectors for the most popular network protocols. With this you can analyze packet captures.

Figure 2: HTTP disabled



```
     Time      Source          Destination     Protocol             Info
   1 0.000000  172.20.88.6     10.100.16.50    TCP                  80 → 28367 [SYN, ACK] Seq=0 Ack=1 Win=8760 Len=0 MSS=1460
   2 0.003779  10.100.16.50    172.20.88.6     TCP                  28367 → 80 [ACK] Seq=1 Ack=1 Win=8760 Len=0
   3 0.003786  10.100.16.50    172.20.88.6     TCP                  28367 → 80 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=279 [TCP segment of a reassembled PDU]
```

```
ame 3: 333 bytes on wire (2664 bits), 333 bytes captured (2664 bits) on interface 0
hernet II, Src: cc:03:1a:3a:00:20 (cc:03:1a:3a:00:20), Dst: c2:01:0a:94:00:10 (c2:01:0a:94:00:10)
ternet Protocol Version 4, Src: 10.100.16.50, Dst: 172.20.88.6
ansmission Control Protocol, Src Port: 28367, Dst Port: 80, Seq: 1, Ack: 1, Len: 279
 Source Port: 28367
 Destination Port: 80
 [Stream index: 0]
 [TCP Segment Len: 279]
 Sequence number: 1    (relative sequence number)
 [Next sequence number: 280    (relative sequence number)]
 Acknowledgment number: 1    (relative ack number)
 0101 .... = Header Length: 20 bytes (5)
 Flags: 0x018 (PSH, ACK)
 Window size value: 8760
 [Calculated window size: 8760]
 [Window size scaling factor: -2 (no window scaling used)]
 Checksum: 0xc310 [unverified]
 [Checksum Status: Unverified]
 Urgent pointer: 0
 [SEQ/ACK analysis]
 [Timestamps]
 TCP payload (279 bytes)
 TCP segment data (279 bytes)
```

I disabled all protocols and The only thing that remains is the number of packets and the duration. This is why dissectors are important.

1.7 Bullet point 7

In bullet point 6 we disabled HTTP. If I try to decode as HTTP on a TCP packet with the port 8880 there is no option to choose HTTP. If I enable back protocol HTTP then Wireshark automatically recognizes the packet protocol and if I decode as HTTP nothing happens. Decode as is needed in case Wireshark is not able to understand a protocol and we want to try to see if we can manually set it. This is also used if the network is unusual and the data has an offset, you can make it more readable using decode as.

2 3.2 Flow Engineering

2.1 Bullet point 1

Here I describe only the profiles I created in subsection 1.4.

- P1 - This profile is composite because here we have 2 different applications, F1 and F2, two different flows going to the server.
- P2 - This profile is individual because there is only one flow to the server and also it is unidirectional.
- P3 - This profile is individual, the same as P2.

2.2 Bullet point 2 and 3

Here I will talk about profiles that I have created in subsection 1.4. I applied a filter for each profile and then I exported all the filtered packets to another .pcapng file and then I used Captured file proprieties option to get the wanted data. I already wrote the directionality here without realizing that this was asked in the next bullet point.

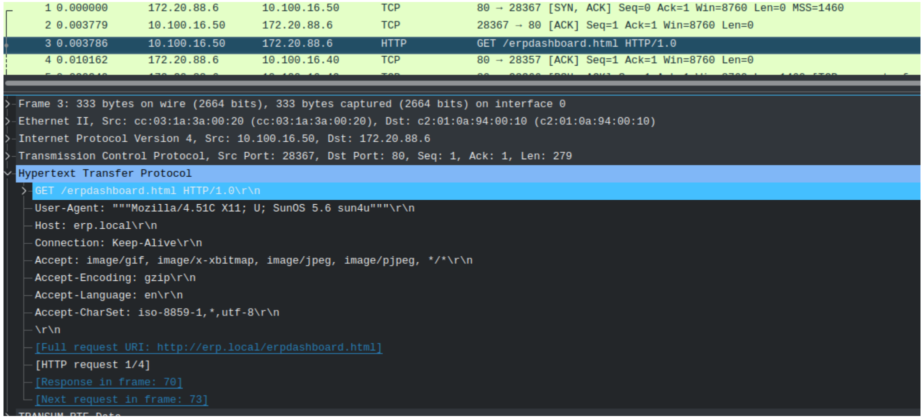
- P1 - Here all the packets go downstream meaning the direction towards the destination, to the server. Client-Server model. Here the server is the data sink and the clients are the data sources. 1436 packets sent in P1 with 723853 bytes transmitted.

- P2 - Here the flow is bidirectional, asymmetric. 99% of the packets go downstream and less than 1% go upstream. Client-Server model. The server is the data sink and the clients are the data sources. 87813 packets in P2 with 34327313 bytes transmitted. This is the biggest flow.

- P3 - Here the flows are again bidirectional, the same amount of packages go downstream and upstream but the number of bytes going upstream is 5 times the size of bytes going downstream. Client-Server model. This makes the client the data sink and the server the data generator. 285 packets in P3 with 94335 bytes transmitted.

2.3 Bullet point 4

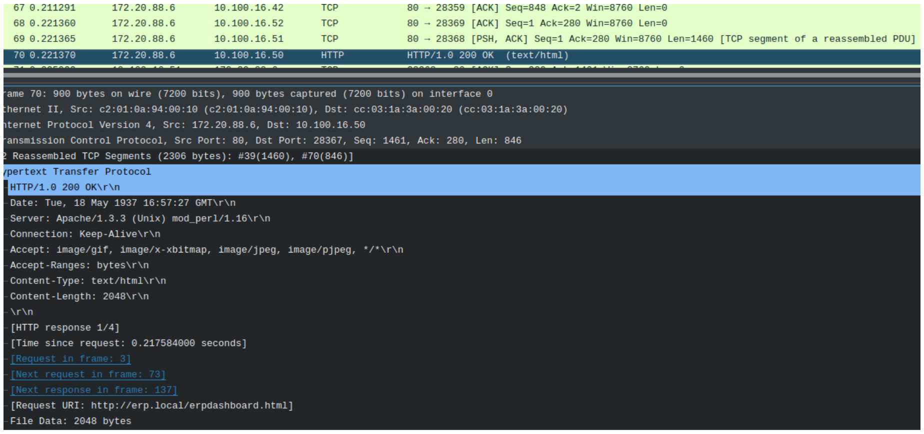
P2 or F4 is the flow where you can monitor traffic of the ERP application. here clients request data from the server using the HTTP protocol.



```
1 0.000000   172.20.88.6    10.100.16.50   TCP    80 → 28367 [SYN, ACK] Seq=0 Ack=1 Win=8760 Len=0 MSS=1460
2 0.003779   10.100.16.50   172.20.88.6    TCP    28367 → 80 [ACK] Seq=1 Ack=1 Win=8760 Len=0
3 0.003786   10.100.16.50   172.20.88.6    HTTP   GET /erpdashboard.html HTTP/1.0
4 0.010162   172.20.88.6    10.100.16.40   TCP    80 → 28357 [ACK] Seq=1 Ack=1 Win=8760 Len=0

> Frame 3: 333 bytes on wire (2664 bits), 333 bytes captured (2664 bits) on interface 0
> Ethernet II, Src: cc:03:1a:3a:00:20 (cc:03:1a:3a:00:20), Dst: c2:01:0a:94:00:10 (c2:01:0a:94:00:10)
> Internet Protocol Version 4, Src: 10.100.16.50, Dst: 172.20.88.6
> Transmission Control Protocol, Src Port: 28367, Dst Port: 80, Seq: 1, Ack: 1, Len: 279
v Hypertext Transfer Protocol
  > GET /erpdashboard.html HTTP/1.0\r\n
    User-Agent: """Mozilla/4.51C X11; U; SunOS 5.6 sun4u"""\r\n
    Host: erp.local\r\n
    Connection: Keep-Alive\r\n
    Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*\r\n
    Accept-Encoding: gzip\r\n
    Accept-Language: en\r\n
    Accept-CharSet: iso-8859-1,*,utf-8\r\n
    \r\n
    [Full request URI: http://erp.local/erpdashboard.html]
    [HTTP request 1/4]
    [Response in frame: 70]
    [Next request in frame: 73]
> TRANSUM RTE Data
```

Here a client requests erpdashboard.html,which is a web page, we can see more optional headers in the picture.



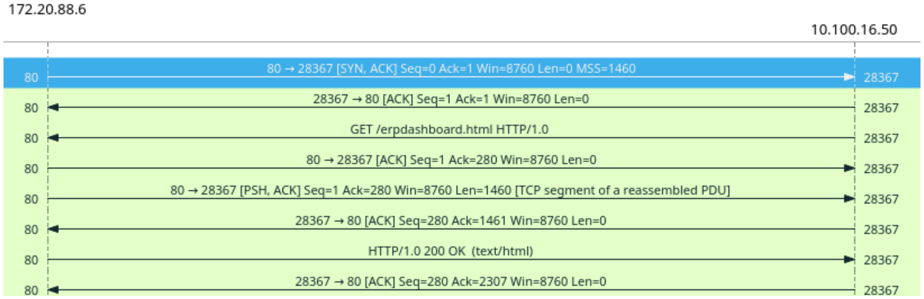
```
 67 0.211291      172.20.88.6      10.100.16.42      TCP          80 → 28359 [ACK] Seq=848 Ack=2 Win=8760 Len=0
 68 0.221360      172.20.88.6      10.100.16.52      TCP          80 → 28369 [ACK] Seq=1 Ack=280 Win=8760 Len=0
 69 0.221365      172.20.88.6      10.100.16.51      TCP          80 → 28368 [PSH, ACK] Seq=1 Ack=280 Win=8760 Len=1460 [TCP segment of a reassembled PDU]
 70 0.221370      172.20.88.6      10.100.16.50      HTTP         HTTP/1.0 200 OK  (text/html)
```

```
rame 70: 900 bytes on wire (7200 bits), 900 bytes captured (7200 bits) on interface 0
thernet II, Src: c2:01:0a:94:00:10 (c2:01:0a:94:00:10), Dst: cc:03:1a:3a:00:20 (cc:03:1a:3a:00:20)
nternet Protocol Version 4, Src: 172.20.88.6, Dst: 10.100.16.50
ransmission Control Protocol, Src Port: 80, Dst Port: 28367, Seq: 1461, Ack: 280, Len: 846
2 Reassembled TCP Segments (2306 bytes): #39(1460), #70(846)]
ypertext Transfer Protocol
 HTTP/1.0 200 OK\r\n
 Date: Tue, 18 May 1937 16:57:27 GMT\r\n
 Server: Apache/1.3.3 (Unix) mod_perl/1.16\r\n
 Connection: Keep-Alive\r\n
 Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*\r\n
 Accept-Ranges: bytes\r\n
 Content-Type: text/html\r\n
 Content-Length: 2048\r\n
 \r\n
 [HTTP response 1/4]
 [Time since request: 0.217584000 seconds]
 [Request in frame: 3]
 [Next request in frame: 73]
 [Next response in frame: 137]
 [Request URI: http://erp.local/erpdashboard.html]
 File Data: 2048 bytes
```

Here the server sends a response 200 OK. We can see that there is about 200 milliseconds of delay between the request and the response.

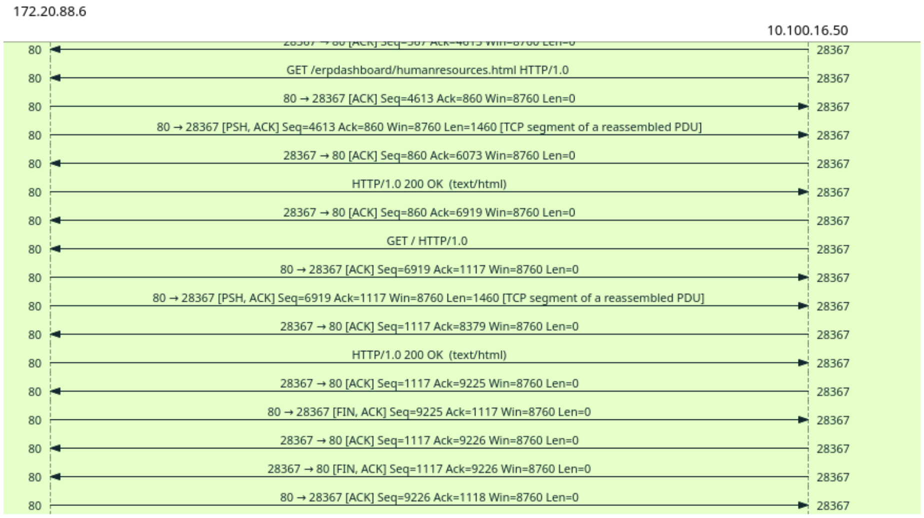
2.4 Bullet point 5

Here I applied a conversation filter to show only the previous flow, the one in section 2.3. In this case, the start of the flow can be identified by the 3 way handshake of the TCP protocol and a filter that shows only the interaction between the the source IP and the sink IP.



```
80 → 28367 [SYN, ACK] Seq=0 Ack=1 Win=8760 Len=0 MSS=1460
28367 → 80 [ACK] Seq=1 Ack=1 Win=8760 Len=0
GET /erpdashboard.html HTTP/1.0
80 → 28367 [ACK] Seq=1 Ack=280 Win=8760 Len=0
80 → 28367 [PSH, ACK] Seq=1 Ack=280 Win=8760 Len=1460 [TCP segment of a reassembled PDU]
28367 → 80 [ACK] Seq=280 Ack=1461 Win=8760 Len=0
HTTP/1.0 200 OK  (text/html)
28367 → 80 [ACK] Seq=280 Ack=2307 Win=8760 Len=0
```

In this image we can see the start of the flow. We can observe the 3 way handshake at the beginning, we can see the source and the destination IP, here the source is the server and we can observe the same requests and responses as previously discussed.

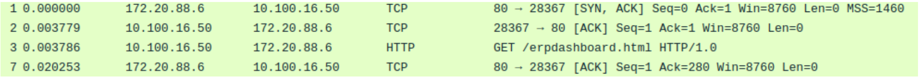


```
172.20.88.6
                                                                          10.100.16.50
80    28367 → 80 [ACK] Seq=567 Ack=4613 Win=8760 Len=0                                28367
80    GET /erpdashboard/humanresources.html HTTP/1.0                                  28367
80    80 → 28367 [ACK] Seq=4613 Ack=860 Win=8760 Len=0                                28367
80    80 → 28367 [PSH, ACK] Seq=4613 Ack=860 Win=8760 Len=1460 [TCP segment of a reassembled PDU]   28367
80    28367 → 80 [ACK] Seq=860 Ack=6073 Win=8760 Len=0                                28367
80    HTTP/1.0 200 OK  (text/html)                                                    28367
80    28367 → 80 [ACK] Seq=860 Ack=6919 Win=8760 Len=0                                28367
80    GET / HTTP/1.0                                                                  28367
80    80 → 28367 [ACK] Seq=6919 Ack=1117 Win=8760 Len=0                               28367
80    80 → 28367 [PSH, ACK] Seq=6919 Ack=1117 Win=8760 Len=1460 [TCP segment of a reassembled PDU]  28367
80    28367 → 80 [ACK] Seq=1117 Ack=8379 Win=8760 Len=0                               28367
80    HTTP/1.0 200 OK  (text/html)                                                    28367
80    28367 → 80 [ACK] Seq=1117 Ack=9225 Win=8760 Len=0                               28367
80    80 → 28367 [FIN, ACK] Seq=9225 Ack=1117 Win=8760 Len=0                          28367
80    28367 → 80 [ACK] Seq=1117 Ack=9226 Win=8760 Len=0                               28367
80    28367 → 80 [FIN, ACK] Seq=1117 Ack=9226 Win=8760 Len=0                          28367
80    80 → 28367 [ACK] Seq=9226 Ack=1118 Win=8760 Len=0                               28367
```

Here the flow ends, we can see the FIN flag.

2.5 Bullet point 6

Figure 3: First lines



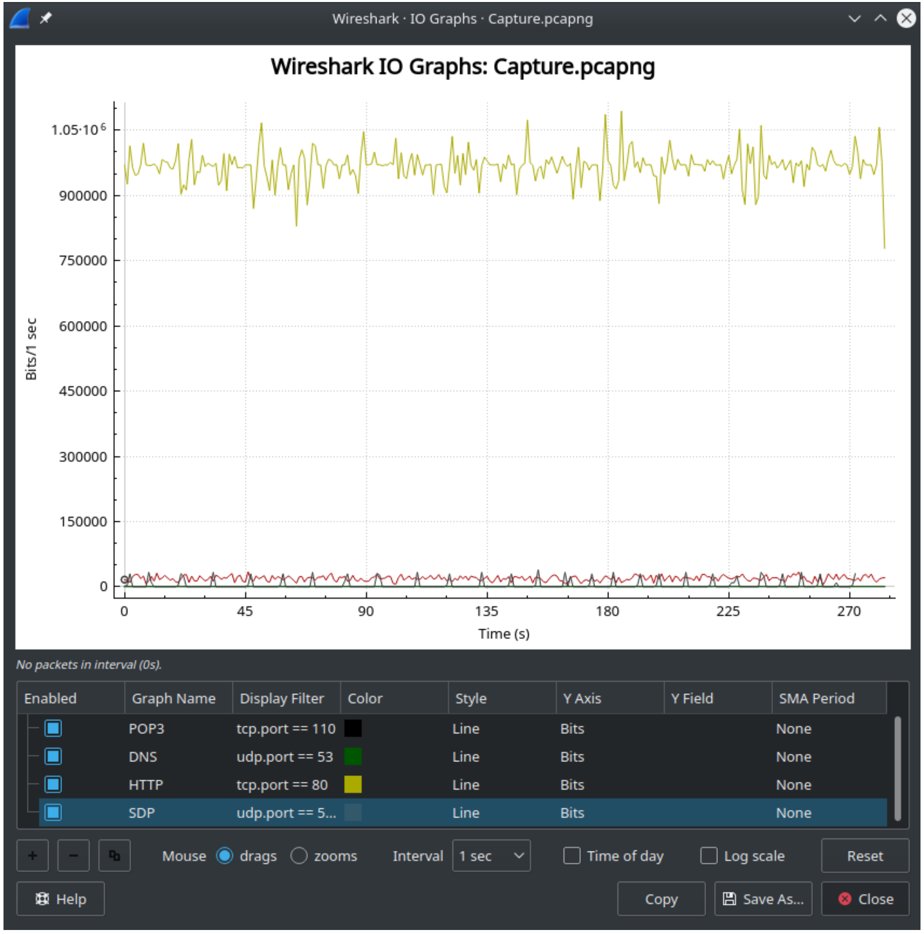
```
1 0.000000     172.20.88.6     10.100.16.50    TCP         80 → 28367 [SYN, ACK] Seq=0 Ack=1 Win=8760 Len=0 MSS=1460
2 0.003779     10.100.16.50    172.20.88.6     TCP         28367 → 80 [ACK] Seq=1 Ack=1 Win=8760 Len=0
3 0.003786     10.100.16.50    172.20.88.6     HTTP        GET /erpdashboard.html HTTP/1.0
7 0.020253     172.20.88.6     10.100.16.50    TCP         80 → 28367 [ACK] Seq=1 Ack=280 Win=8760 Len=0
```

I have selected these 4 lines because this is the easiest for me to explain.The server has an IP of 172.20.88.6 and the client has the IP 10.100.16.50 . The server sends a SYN, ACK to the client, the client responds with an ACK and then the client requests a web page erpdashboard.html. Lastly the server responds with an ACK.

2.6 Bullet point 7

Here I included bits for the Y axis because before I mostly analyzed the packets before and I needed another perspective. If time of day is enabled it helps you to measure the bursts of data and plan accordingly for the peaks, it mostly helps during the day.

Figure 4: IO Graph



No packets in interval (0s).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SMA Period
☑	POP3	tcp.port == 110	⬛	Line	Bits		None
☑	DNS	udp.port == 53	🟩	Line	Bits		None
☑	HTTP	tcp.port == 80	🟨	Line	Bits		None
☑	SDP	udp.port == 5...	■	Line	Bits		None

Mouse ● drags ○ zooms Interval 1 sec □ Time of day □ Log scale