Wireshark assignment due 14/4/2021

**Introduction / Methodology**

In wireshark we use a lot of filters, there is 2 types of filters, we have display filters and capture filters, and we use these filters to single out information we are looking for. One of these filters is DNS (domain name system), we use dns to associate a name (such as a website or service) and use it to communicate with computers or services connected to a device such as our example Dell\_45:49 or network. We also have other filters that are commonly used mainly-packet related such as UDP and TCP. UDP is a friend of DNS because it can help with dns queries, but it is mainly used for delivering a quicker and more efficient stream of information/data by ignoring error checking and UDP gives applications a way to communicate a sorted and error-free stream off information packets over the network. TCP in this case is communicating with Dell\_45:49 and is sharing packets and application data in different sequences and acknowledging each other. TCP can also send back data as it was received, this will be explained better later-on.

Wireshark is a packet sniffer tool, we use Wireshark as a way of sniffing out these packets so we can see these packets because packets are important, packets are a formatted unit of data that is passed through a network. A packet consists of information and user data and can so much information in just a small size 64kb. In Wireshark we can find packets anywhere, but if you ever need to find a certain set or specific individual packet then you can use the filter frame.number == or you could sort packets by a destination or ip address using ip.dst == or ip.addr ==

In this case I used the filter frame.number == to identify the location of the packets 295, 302, 560, 4154, 9525 and find the information needed on them to try and give a detailed description of the packets.

**Notable info before evaluating packet information:**

**DNS/Standard Query:**

So, A DNS or a standard query known by both names is a request for information sent from a or multiple dns client to a dns server. Typically, A DNS Query goes from Client > Server, asking for the IP associated with a complete domain name. If an IP address is needed to be found the client will query dns servers to get the IP if it is known by its domain name. Typically operates using UDP (Port 53)

**Results of each packet:**

**Filter: frame.number == 560:** frame/packet 560 is a dns packet, its capture, and frame length captured is 712 bits, and the frames number is 560, the interfaces name is device npf. It’s from the address destination “all-hsrp-routers\_1f"” (00:00:0c:07:ac:1f) and the source of it is the computer dell\_45:49:bb and its type was IPv4. Its source address is from the IP address 172.18.9.246 and destination address is from the ip address172.24.5.50 using the source port of 49451 to the destination port of 53 with a length of 55 and an unverified checksum status, its UDP payload is 47 bytes.

**Filter: frame.number == 302**: 74 bytes on wire, 74 bytes captured on interface\Device\NPF\_{47ED5301-4DF6-4715-90D5-96F167EF9D47}. The packet arrived on Feb 18, 2019 14:24:54 GMT Standard Time. The Frame Length was 74 bytes, and the Capture Length was 74 bytes. Its destination was for the computer Address: Dell\_45:49:bb (18:db:f2:45:49:bb) from the source: Cisco\_33:d8:00 (00:14:f1:33:d8:00) its type was: IPv4 and Its time to live was 53. Its protocol was ICMP. Its Source Address was: 109.199.112.160 and its Destination Address was: 172.18.9.246. Type: reply.

**Filter: frame.number == 295**: 74 bytes on wire, 74 bytes captured (on interface \Device\NPF\_{47ED5301-4DF6-4715-90D5-96F167EF9D47}, Its Arrival Time: Feb 18, 2019 14:24:54 GMT, Frame Length: 74 bytes and Capture Length: 74 bytes, Protocol: ICMP, Source: 172.18.9.246, Destination: 109.199.112.160, Time to Live: 128, Type: IPv4, Request. Checksum status: Good.

**Filter: frame.number == 4154**: 192 bytes on wire, 192 bytes captured, time: 23.82, Source Address: 172.24.5.50, Destination Address: 172.18.9.246, Protocol: UDP (17), Protocol: DNS, Length: 192, Info: Standard query response from the website rotten tomatoes: static.rottentomatoes.com. Time to Live: 126, checksum status: Unverified, Type: IPv4, Frame Length: 192 bytes, Capture Length: 192 bytes, Arrival Time: Feb 18, 2019 14:25:11 GMT, Source Port: 53, Destination Port: 58781, UDP payload was 150 bytes. Flags: Standard query response, No error.

**Filter: frame.number == 9525:** 60 bytes on wire, 60 bytes captured. Arrival Time: Feb 18, 2019 14:25 GMT. Frame Length: 60 bytes, Capture Length: 60 bytes, Protocols: UDP. Source: Apple\_b3:75:a4, Type: IPv4, Destination: Broadcast. Total Length: 42, Time to Live: 64. Source Address: 172.18.9.45, Destination Address: 255.255.255.255. The apple device was not trying to directly communicate with a specific device but was creating a broadcast to its destination. Source Port: 52232, Destination Port: 2007, Data: 14 bytes.

**Describe the protocols and port numbers used in devices where the most packets were sent and received:**

There were 2 main protocols that were sending and receiving the most packets, out of which were receiving the most it was indefinitely TCP (Transmission control protocol). The second protocol was UDP (user datagram protocol). The difference between both protocols UDP and TCP is that TCP is a connection-oriented protocol, whereas udp is a connectionless protocol. TCP is slower compared to UDP, simply because TCP does not ignore handshake protocols meanwhile udp does, udp uses no handshake protocols.

TCPs Most popular ports were 51319, 51338 and 51312 and all used port b 443 which is the standard port for https traffic. They sent a combined total of 741 Packets and Received a total of 1737 packets with a duration of 8.9.

UDPs Most popular ports were 1985, 1985, 137, Ports B used the same ports as ports A. They sent a combined total of 89 packets and received a total of 89 packets. The port 1985 is used by the UDP and TCP protocol and is commonly used for attackers to cause a denial of service via udp packets to the HSRP when support is not enabled.

**What filter will return 1197 packets?**

The filter TCP was used to send back the 1197 packets because TCP can guarantee that packets will be delivered and processed in the same way that they were sent. The type of traffic being sent from destination address 172.18.9.246 to 54.192.29.82 was application data, the communication between them commenced via src port 443 to 51919 with a sequence of 395084 with an acknowledgement of 362224 and a length of 1460. There were different resembled tcp segments that involved frame 1194, 1196 and 1197 with a resembled tcp length of 2549. The application data had transport layer security using the version TLS 1.2 of a length of 33, the application data protocol used was http2. The advanced Wireshark feature I used to identify the address used was conversations, I observed the 1197 packets being sent from A -> B and B -> A in the IPv4 tab.

**filters which have helped me remove broadcast or unwanted traffic.**

So, to remove broadcast I had to first identify what broadcast was, broadcast at least In my experience always contains the same destination address which is (ff:ff:ff:ff:ff:ff) so I prepared a filter which the outcome of the filter was eth.dst == ff:ff:ff:ff:ff:ff and I added “or” and several other common/basic filters related to broadcast and unwanted traffic such as

**Address resolution protocol** - Arp: used for mapping layer-3 network address to data link addresses.

**Cisco discovery protocol** - Cdp: Network discovery tool, used for assisting admins in identifying local disco devices running low layer transparent protocols.

**Link layer discovery protocol** - Lldp: Layer 2 discovery protocol used to advertise device information to connected peers.

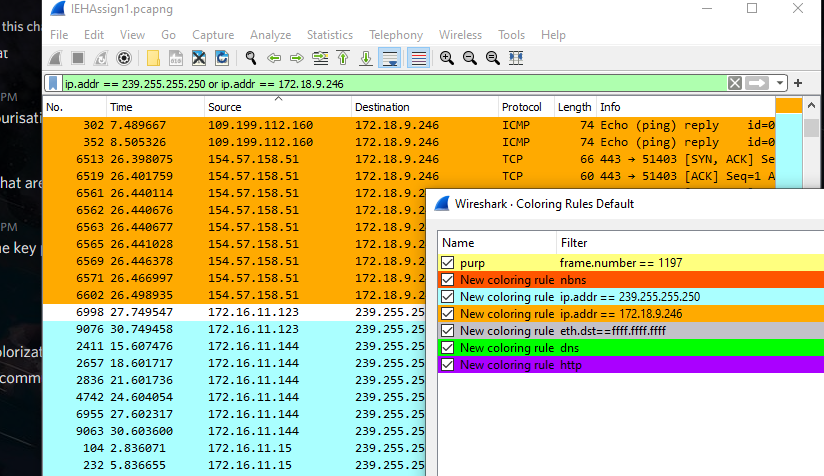
**Spanning tree protocol** - Stp: layer 2 protocol that runs on a bridge or switch, used to ensure that you do not create loops when you have a redundant path in your network, loops can be harmful to your network.

This leaves us with !(eth.dst == ff:ff:ff:ff:ff:ff or arp or cdp or lldp or stp)

This filter cleans up our trace and does not leave us with many broadcast/chatter protocols.

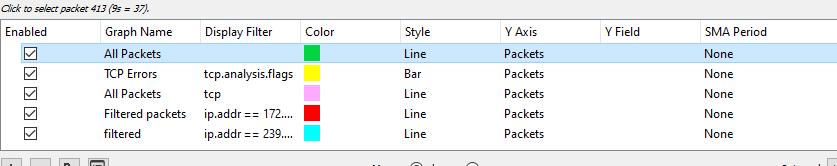
We use filters like these to make it easier for us to look for the information that we need, another way we could do this is using a coloring rule, which helps with identifying information we need rather than information that we do not need a lot easier than manually looking for it.

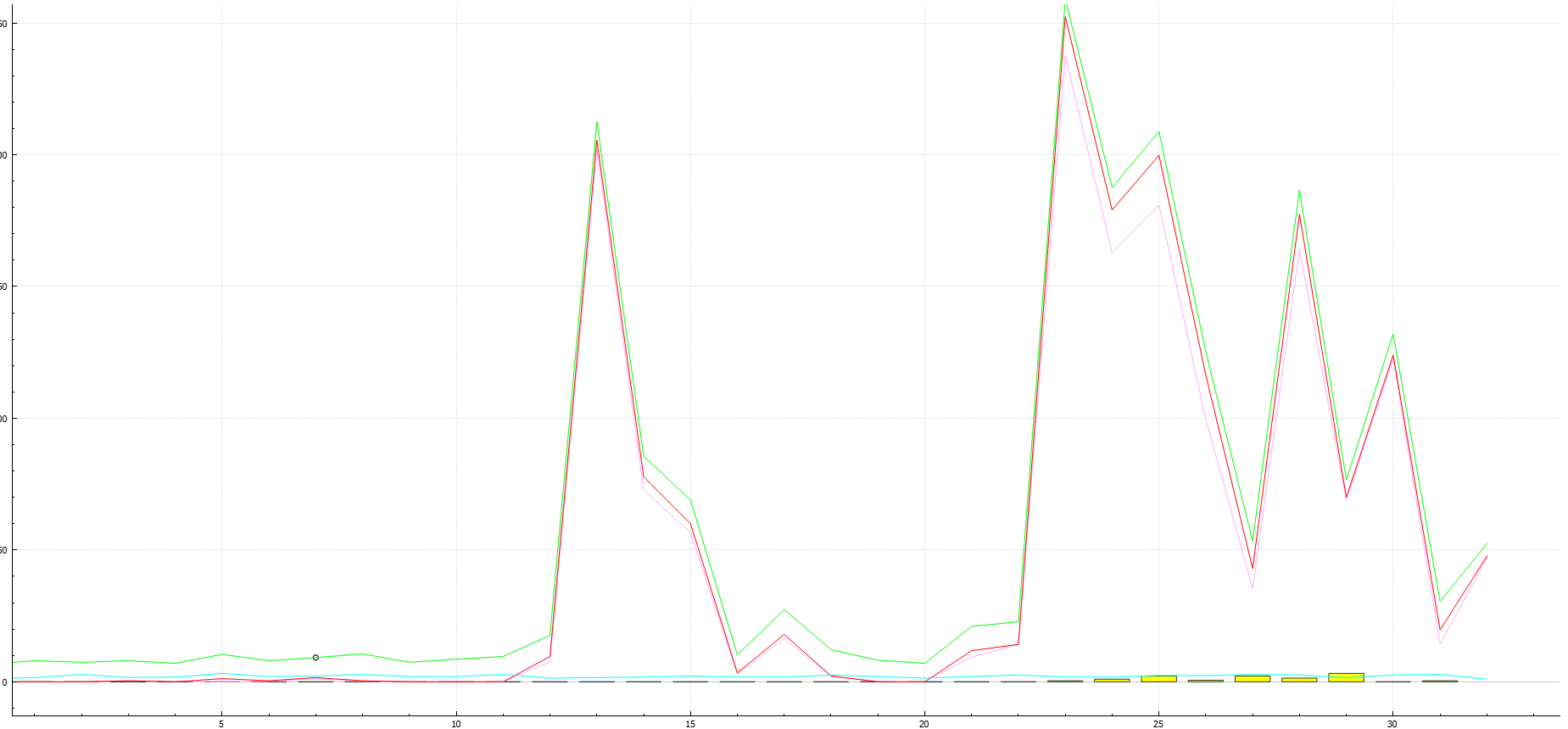
**Two colorization rules to help identify key packets in the capture file:**



The 2 coloring rules I decided to use for identifying key packets were the IP addresses 239.255.255.250 and ip.addr == 172.18.9.246, I chose both of these two filters because they had the most influence on active packets and had some of the highest containing, sending/receiving packets in conversations.

**IO Graph and two filters from above to highlight key traffic.**





**What I learned from researching for / doing this assignment / Discussion of findings/how I felt:**

I learned a lot from doing this assignment, it really helped me learn more about Wireshark and get more comfortable with using it.

I spent a lot of time going back over old notes, lectures and work trying to do this because this was new to me as I have never done this type of work before.

I have learned a lot about filters, how to use them, mixing them without filters, getting rid of unnecessary information, finding information, using conversations, seeing where data comes from, where it goes too, what type of data can be sent through Wireshark.

I had a lot of hardships when it came to finding this, I spent several hours a day starting at my screen, watching YouTube videos, trying to find the information I needed, the right filters, how to use those filters in a way that would actually contribute and help me, trying to find packets that would be useful for me and trying to decipher what information from those packets would be useful to mee which I found very hard as there’s a lot of information that you can get from a packet as where it came from, who it was being sent to, its ports, how long it took, how big the packet was, was type of protocol it was sent from/filtered through such as TCP.

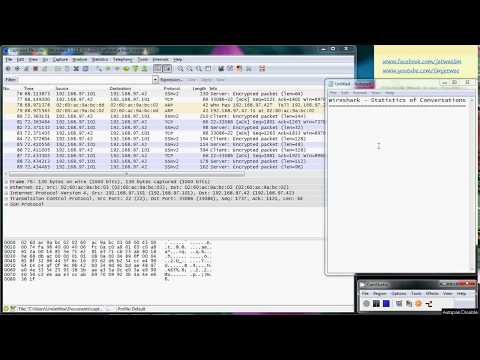
I feel like I was Kinda stuck for time on this assignment, I felt like I struggled a lot and need to prepare more for future assignments as this was hard for me, not only on time but trying to fit close to 2000 words into just 7 questions, but I also could have also simply just answered the questions wrong and not have provided enough information on each as I should have.

**References:**

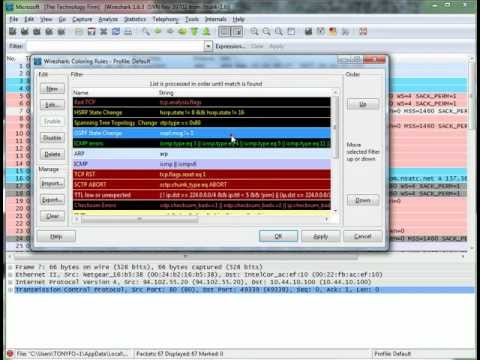
Google for definitions of the filters, such as looking up the meaning of UDP, TCP, Etc.

Videos I used:

[WireShark - Statistics Of Conversations](https://www.youtube.com/watch?v=cXGQSG8ySO4&t=7s)



[Wireshark Colorize](https://www.youtube.com/watch?v=YQGoneW8MSI&t=180s)



[Troubleshooting with Wireshark - Filter Out What You Don't Need](https://youtu.be/jaMob4Pr6dg)



[Wireshark 101: Display Filters and Filter Options, HakTip 122](https://www.youtube.com/watch?v=N-HpD0bUSO4)



I also used many different websites for searching for filters, finding out the difference in the filters, what they do, how they are used and so on. Unfortunately, I do not have these websites saved.

Here is the link to the Turnitin report for the assignment:

**Thank you for reading and I would like to apologize for not getting the full 2000 words on this assignment.**