Assignment 2 – ITC542 Internetworking with TCP/IP

**Question 1**

A network protocol defines how network devices communicate and the rules and conventions of this communication. These protocols describe every interaction between communicating network devices and dictates how each of these interactions are to be completed by each device.  
  
Protocol analysis is the process of capturing and analysing packets being sent over a network with the intent of assessing the nature of the communication. This ability to capture and identify packets allows an IT professional to troubleshoot network communication issues, test network configuration and gather historical information on network performance (Pyles, J., Carrell, J. L., Tittel, E., 2017, p. 24). A common example for using a protocol analyser is to optimise network speeds by finding devices that are producing unnecessary traffic. For example, most printers by default are configured to broadcast using IPX and AppleTalk protocols. If these protocols aren’t being used on your network, the printers are producing unnecessary network traffic. Using a protocol analyser, you would be able to find these broadcast packets and the source of the packets, and reconfigure the printer to stop broadcasting these unused protocols.

Three popular protocol analysers are Wireshark, EtherApe and TCPDump.

**Question 2**

The placement of the protocol analyser is not important on this network since it is a hubbed network, rather than a packet switching network (Pyles et al., 2017, p. 28). In a network environment using hubs, network traffic is forwarded from all ports, therefore the placement of the protocol analyser is not important since it will be able to capture the same data from anywhere on the network. In modern networks using packet switching, a packet is sent on a specific route towards its destination rather than to each and every device, as it does on a hubbed network. On a packet switching network, an analyser would only receive broadcast packets, packets specifically addressed to the analyser, initial packets sent to each device during network startup and multicast packets (Pyles et al., 2017, p. 28).

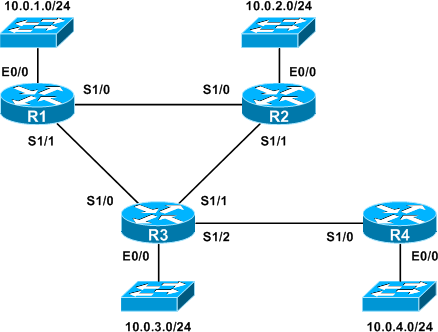
**Question 3**

**Distance Vector Routing**

Distance vector routing is a routing protocol based on the distances between networks and using distance information to find optimal routes. Using this routing protocol, each router shares information to neighbouring routers about the networks that they are directly connected to, as well as networks that they can route to by way of “rumour”. This means that when one router broadcasts its routing table, the table will not make a distinction of which connected router/s it depends on to connect to a particular network, only that it can connect to that network. Routers also calculate cost for the distance between neighbouring routers and use this cost to determine optimal routing.

The diagram below shows 4 routers each connecting to separate networks. Because of “route by rumour”, each of these routers know that they can connect to each of these networks, but they only know the cost of travel from each router to each network and not that the connection to a network is dependent on a specific router being alive. For example, if we assume that each of these hops is a cost of 1, R3 would know that it could connect to 10.0.4.0/24 at a cost of 2 hops and both R1 and R2 would know that they could connect to 10.0.4.0/24 at a cost of 3 hops.

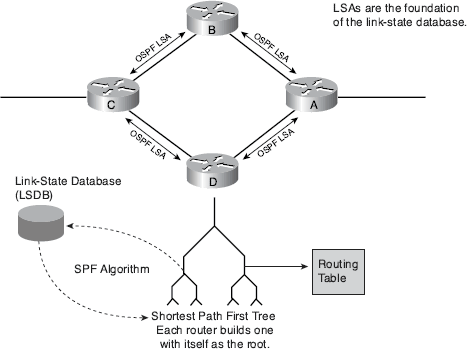
If a packet was coming from 10.0.2.0/24 and heading to 10.0.4.0/24, it would travel to R2 first. R2 would then send the packet to R3, since R3 has a cost of 2 and R1 has a cost of 3. R3 would then send the packet to R4 as it has a cost of 1. If R4 was offline however, R3 would not be able to send the packet to 10.0.4.0/24 via R4. The problem with route by rumour is that the other routers don’t know that R4 was the only route to 10.0.4.0/24. R3 will then send the packet to either R1 or R2 since they both believe that they are able to route to this network. This situation causes a routing loop.



**Link-State Routing**

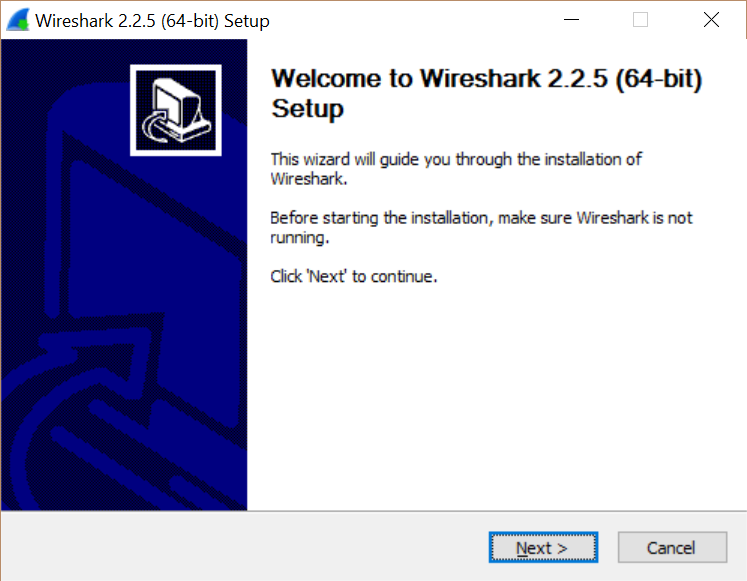
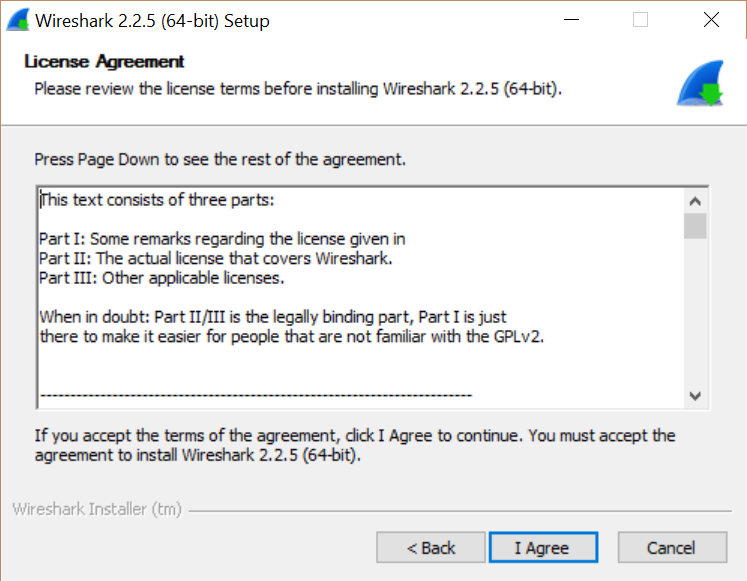
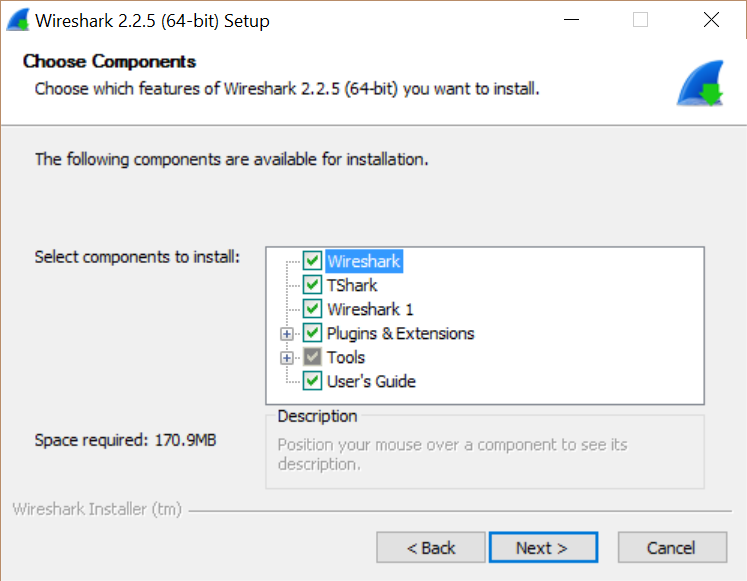
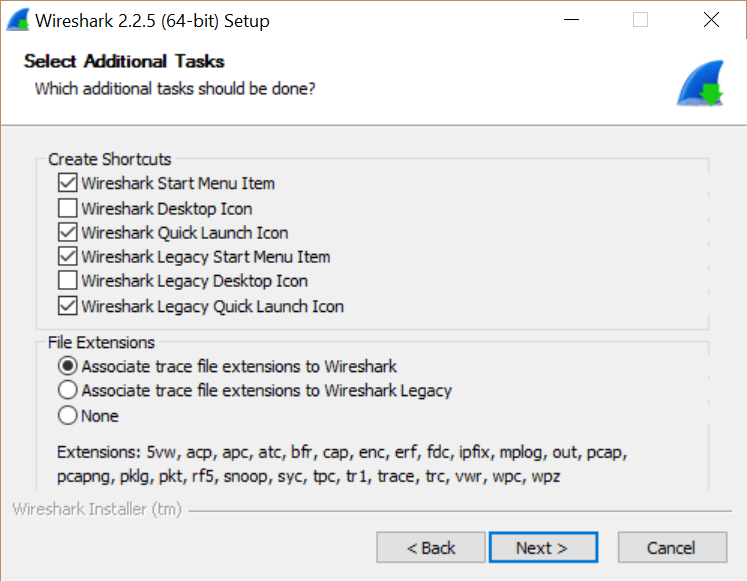
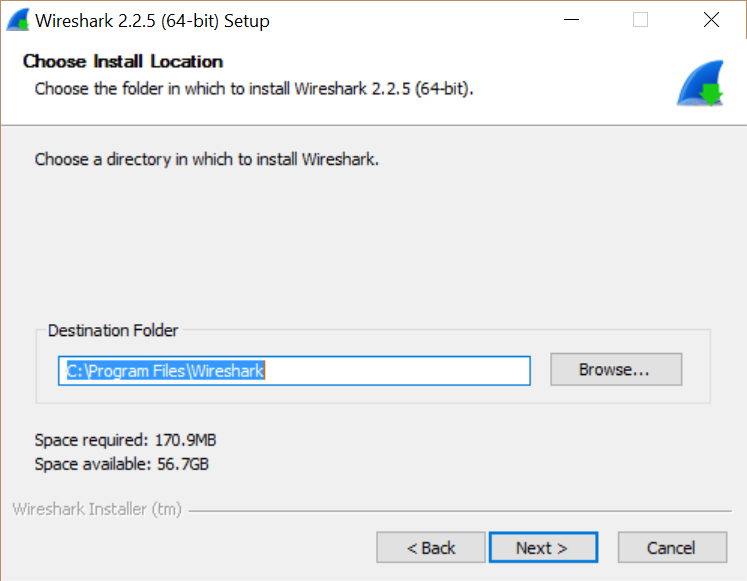
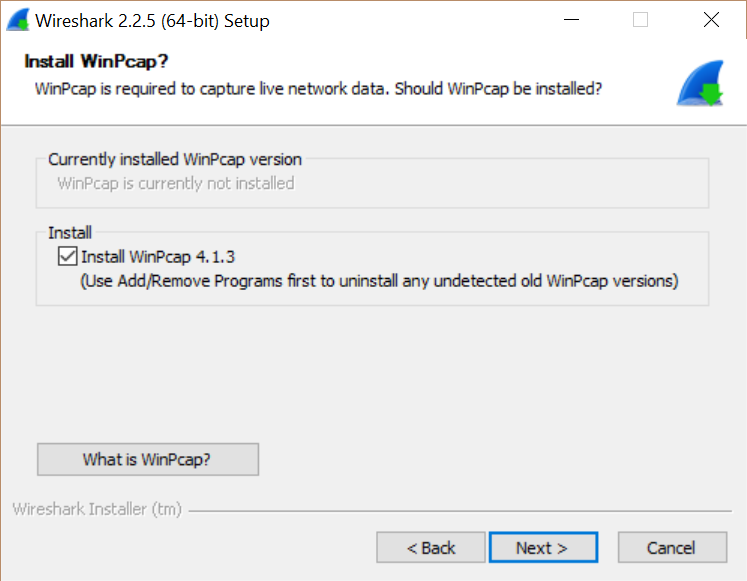
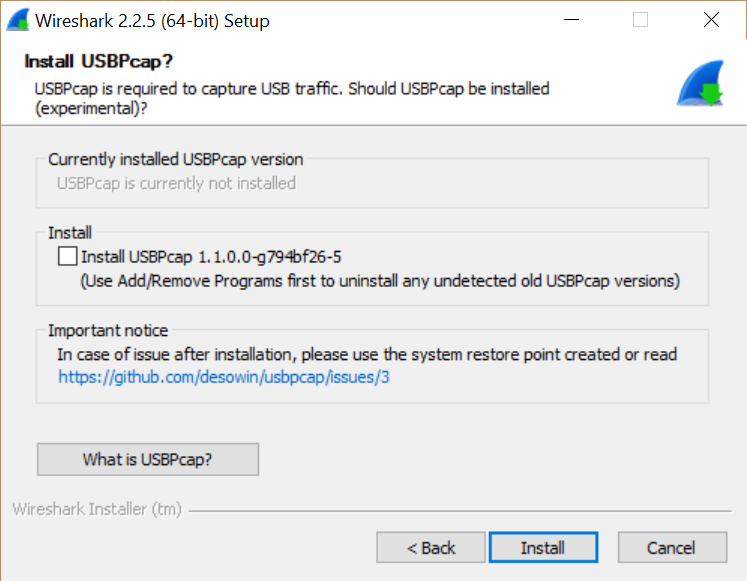
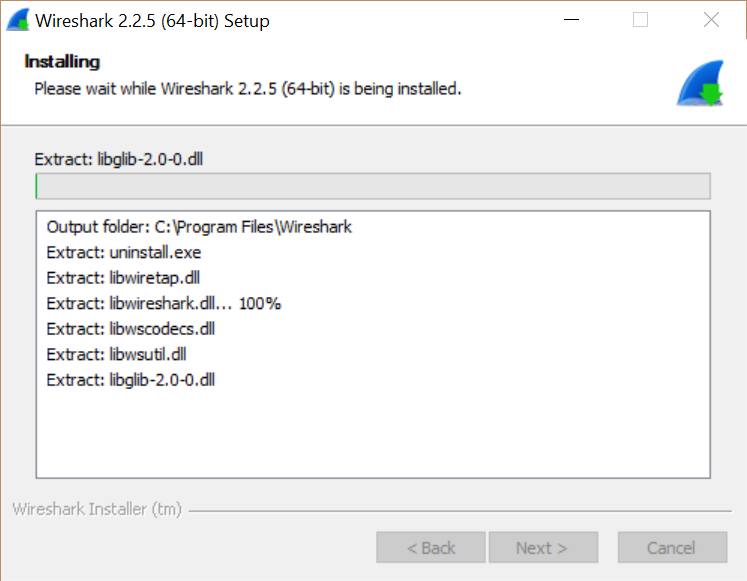
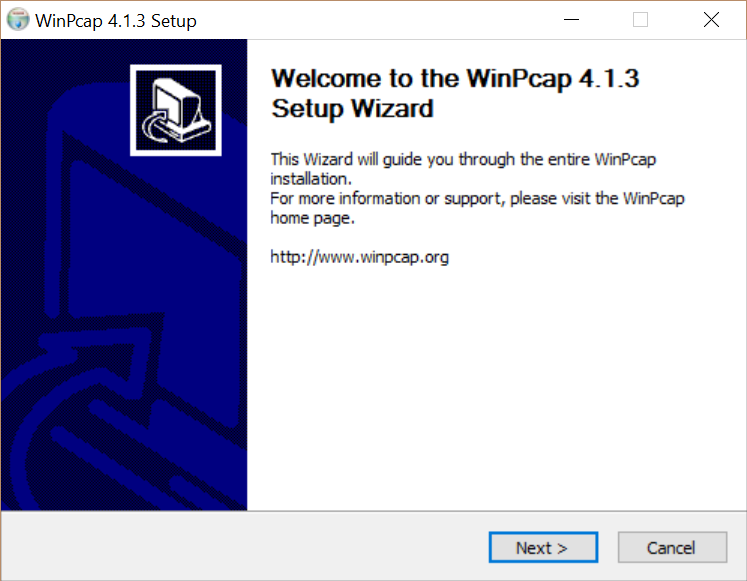
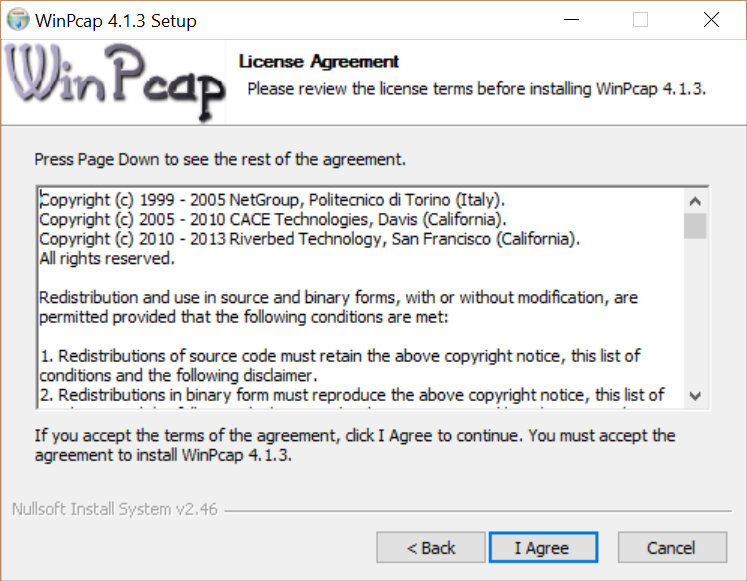
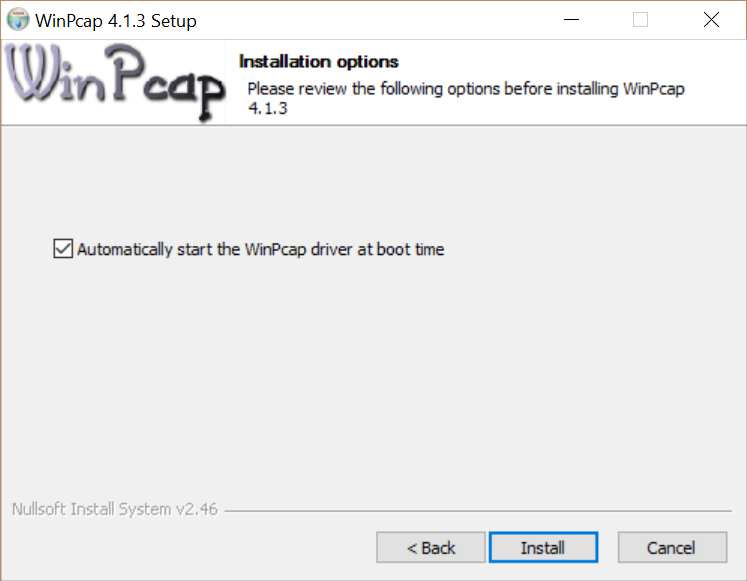
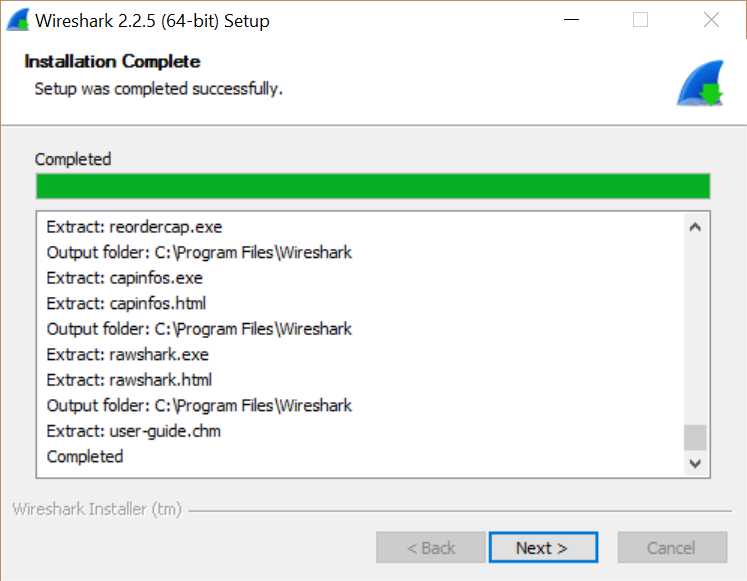
Link-State routing is a routing protocol based on directly connected links and the state of these connections. Unlike distance vector routing, link-state routing protocols do not “route by rumour”; each router’s routing table contains only neighbouring routers that are directly connected. Neighbours are discovered using a process called the Hello process. Once this information is gathered, the router builds and distributes a link-state advertisement (LSA) list containing the neighbour’s details and the cost of travel between routers. As these LSAs are distributed throughout the network, each router gains knowledge of the entire network. Using this information, each router will run an algorithm and build a forwarding table, which is utilised in forwarding packets via the lowest cost route. Below is a diagram which gives an overview of this system.

For example in the diagram below, if router C were to go down the other routers on the network would know that this is the only route to the network on the other side of C. The packet would be dropped and the source notified.

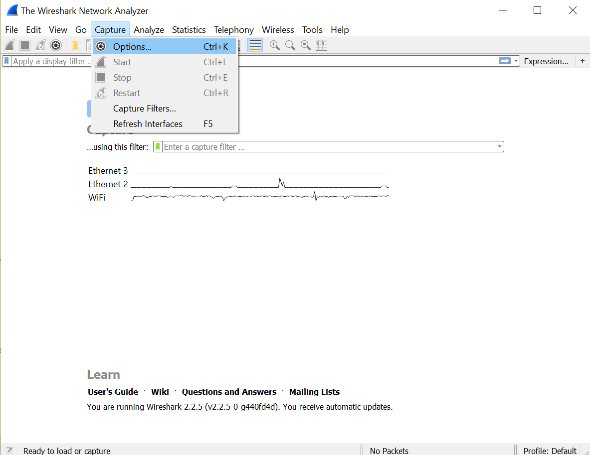
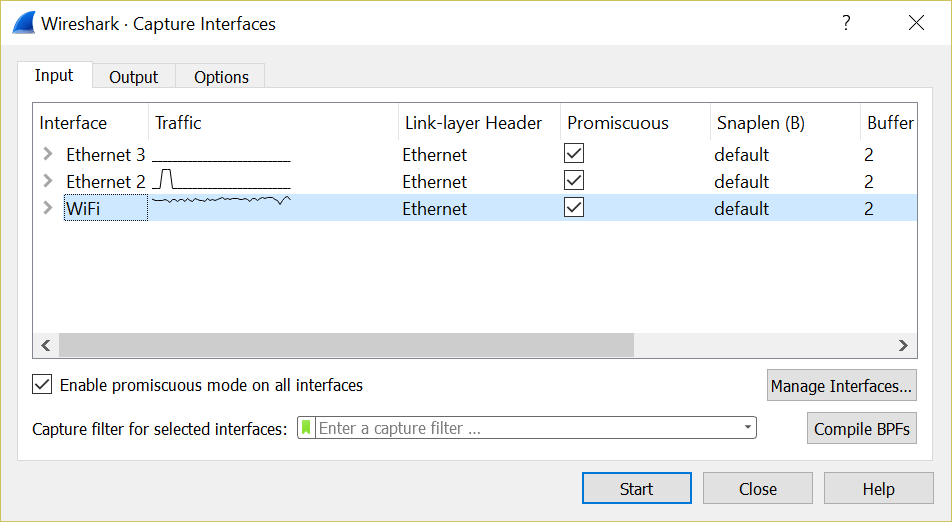
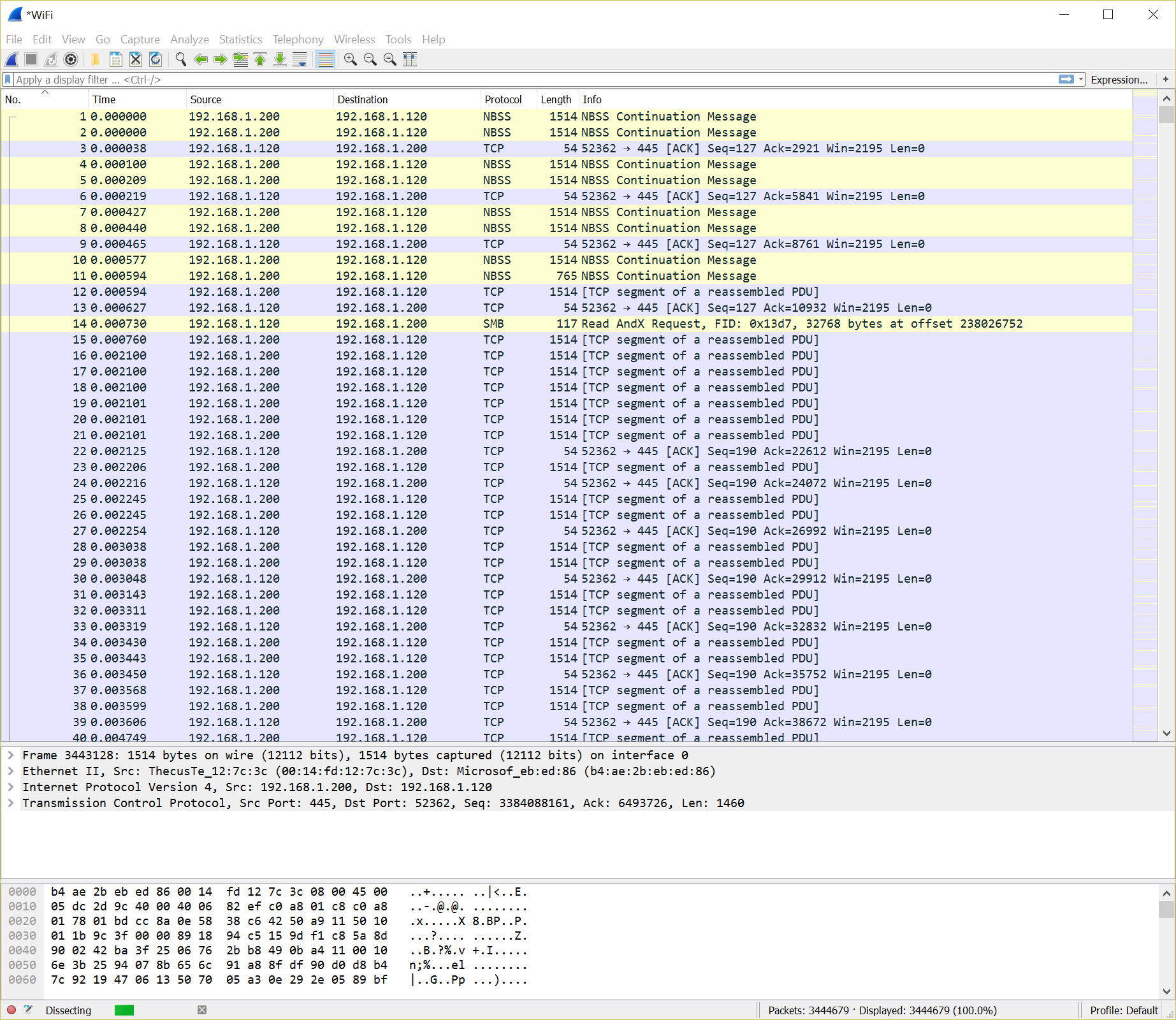
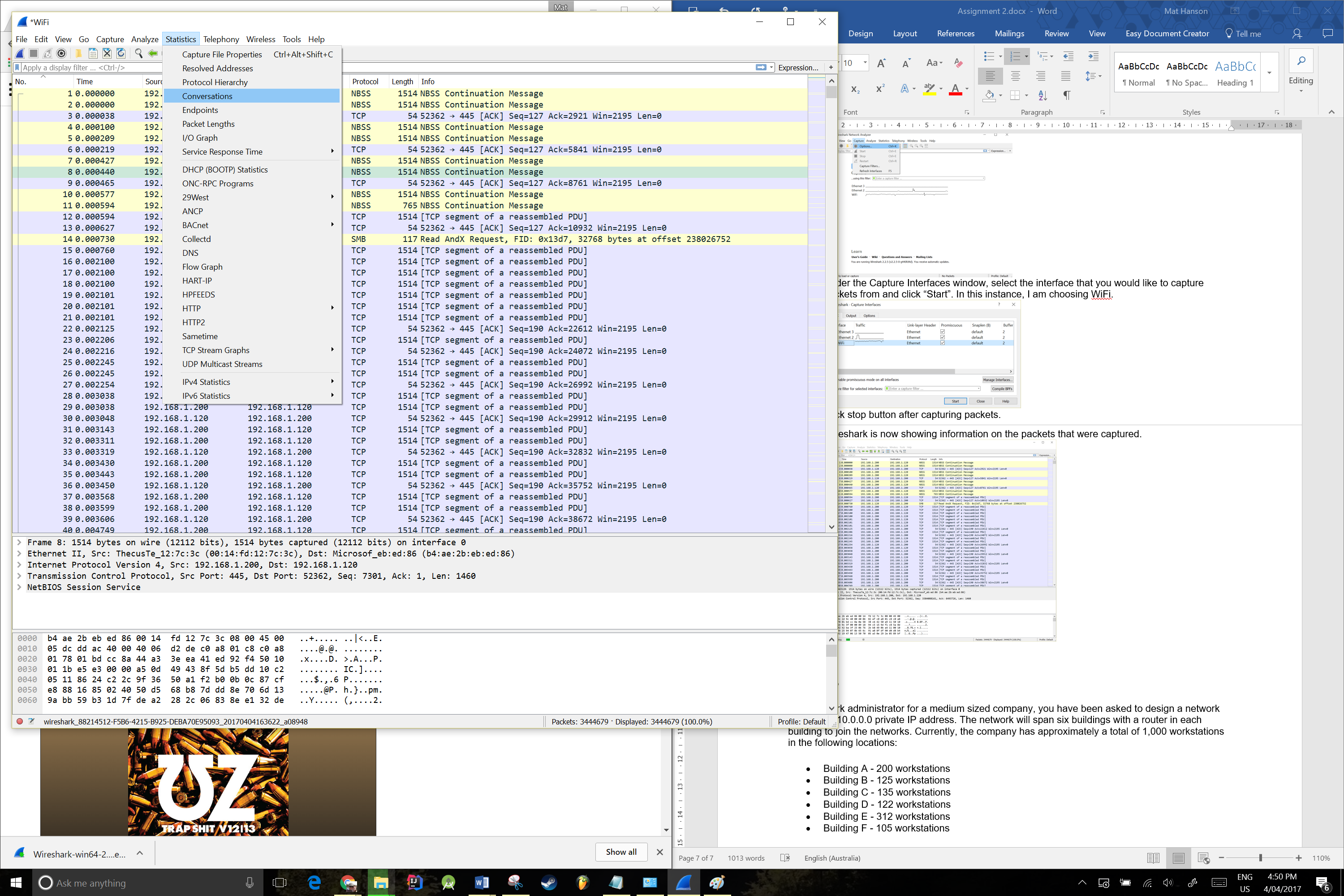
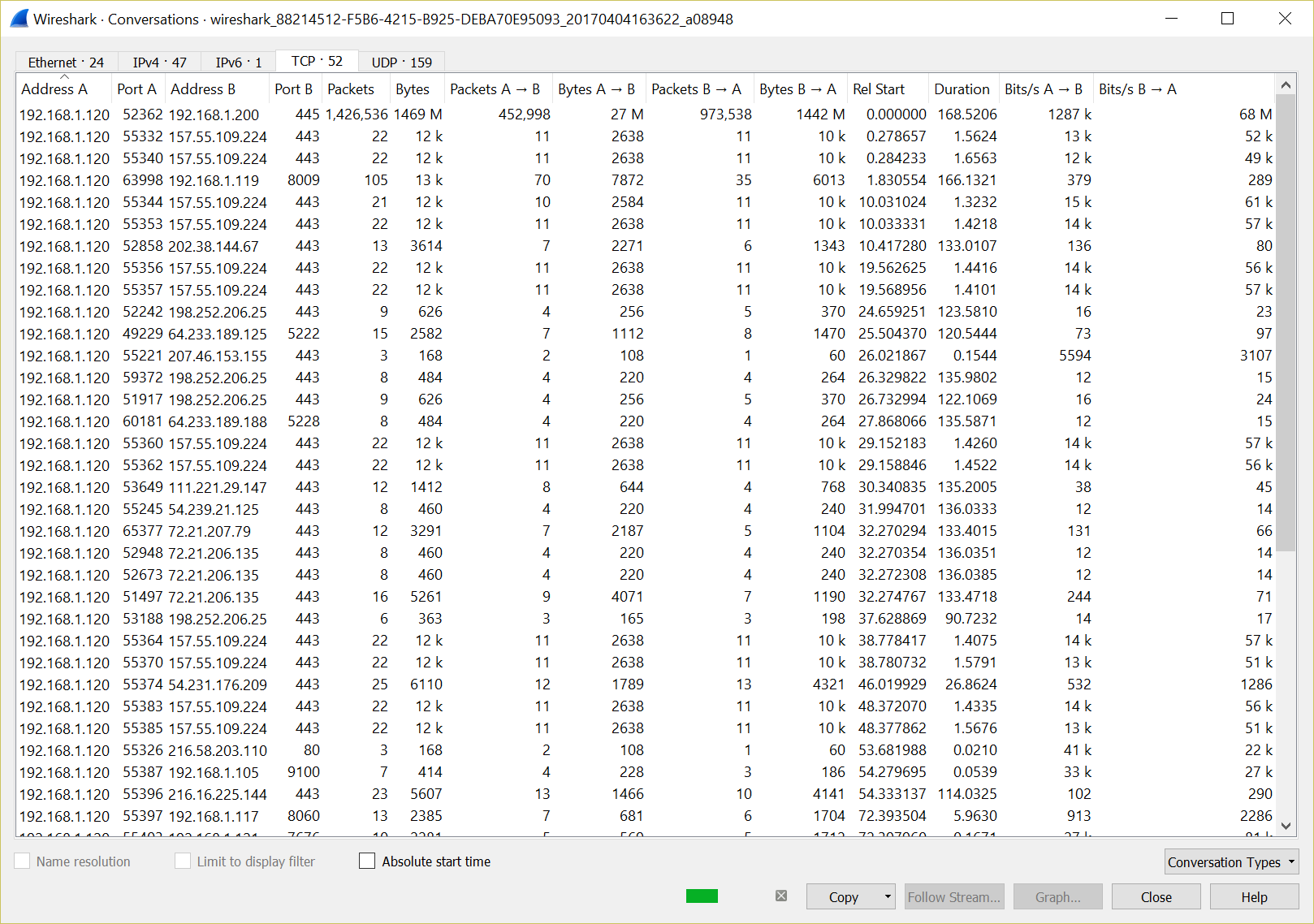
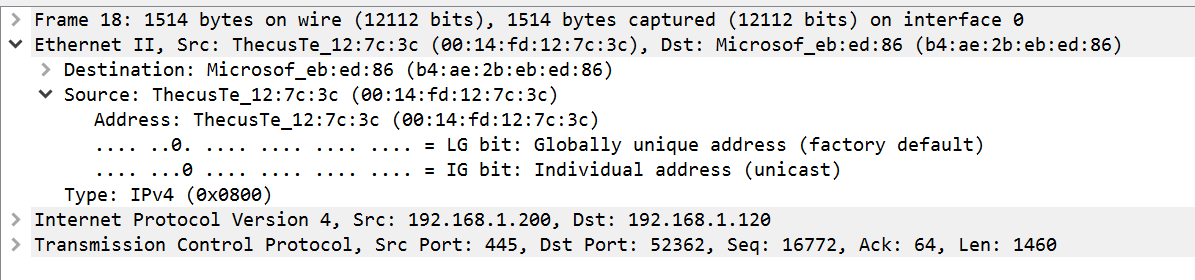
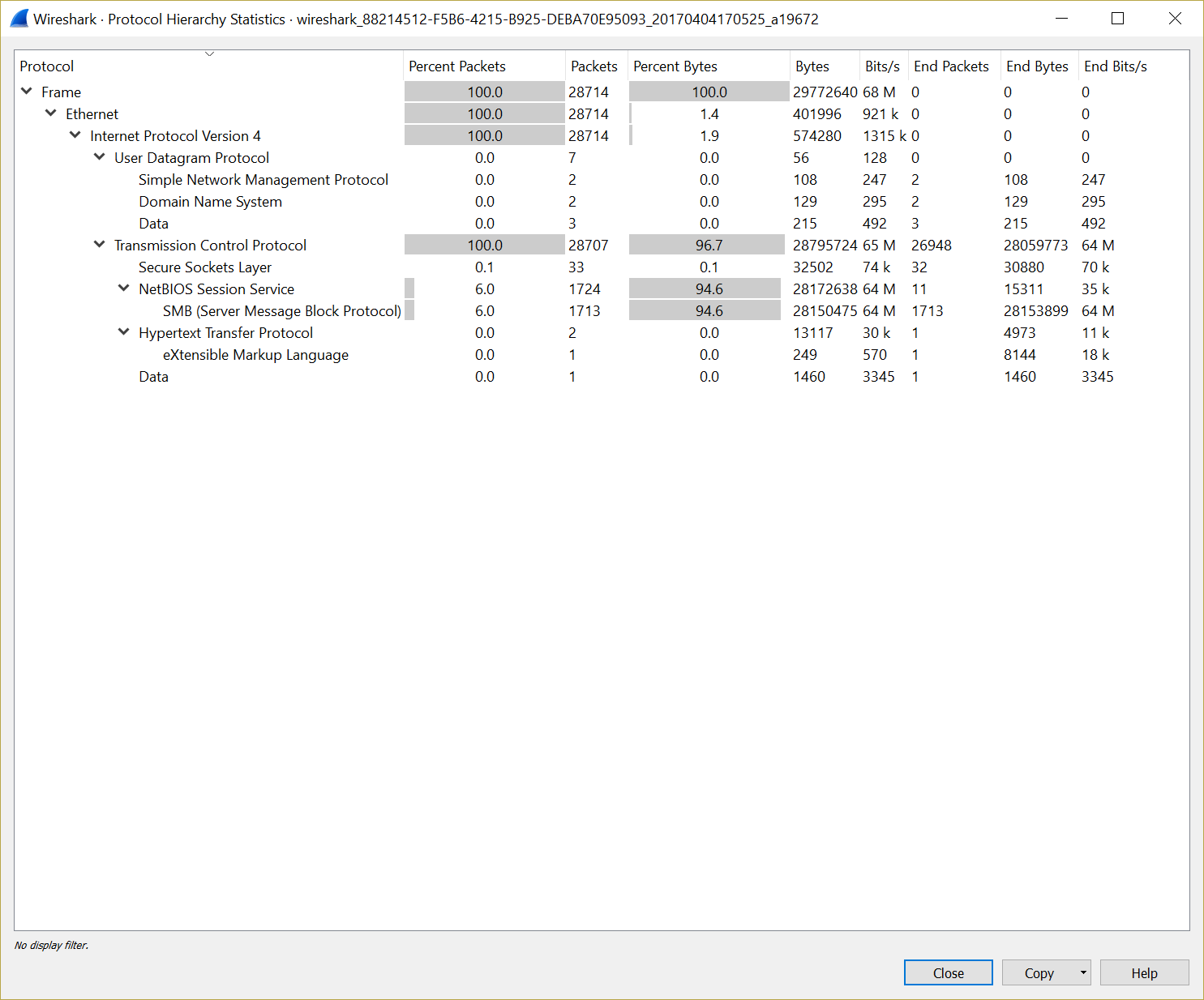
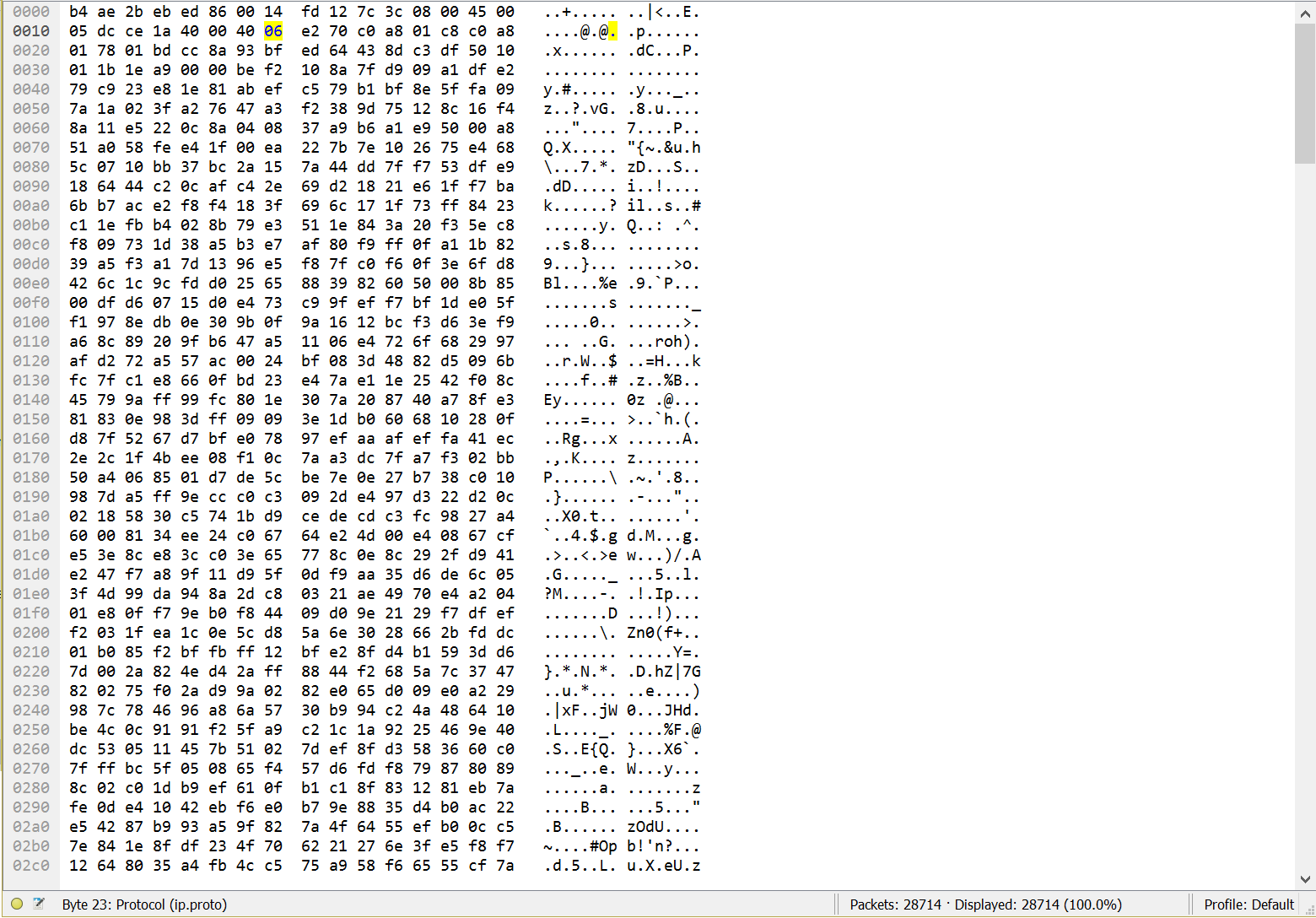
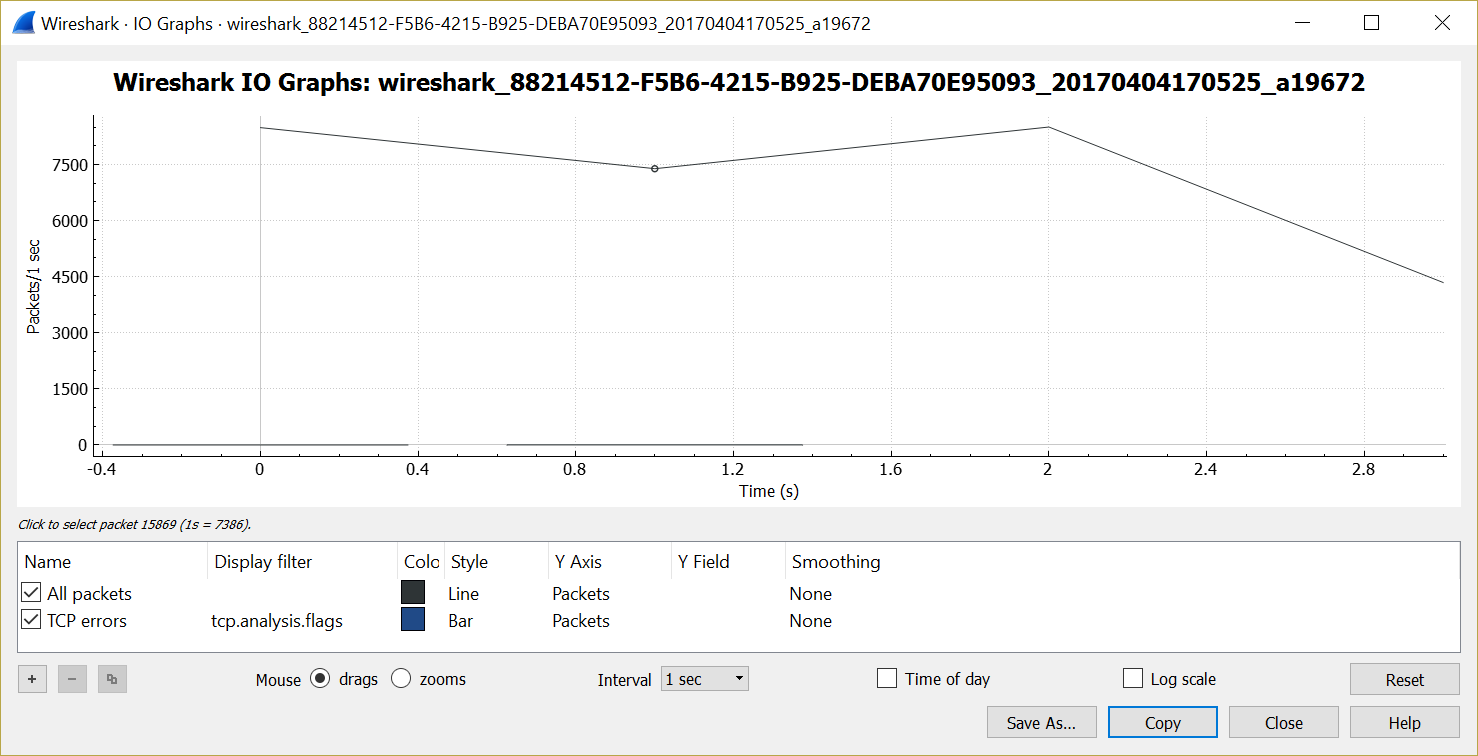
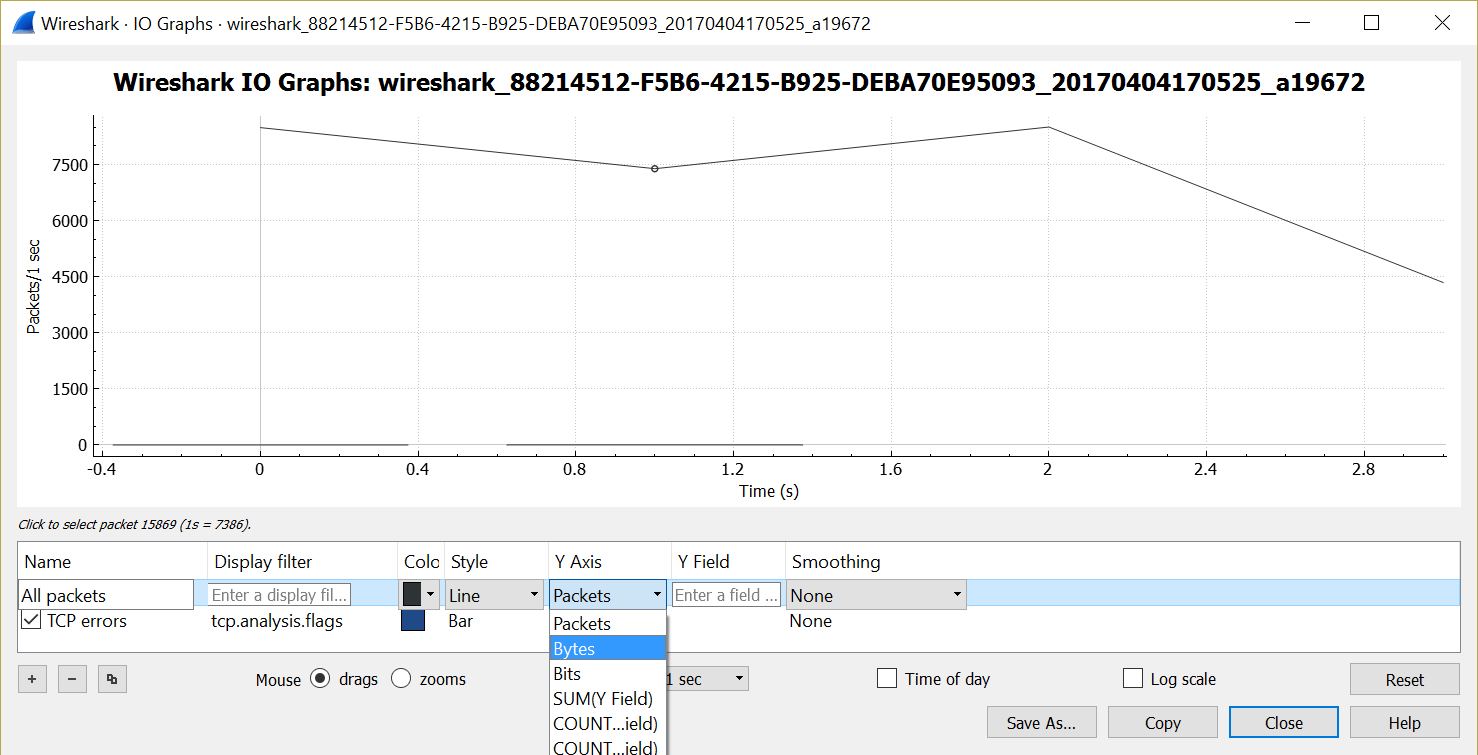
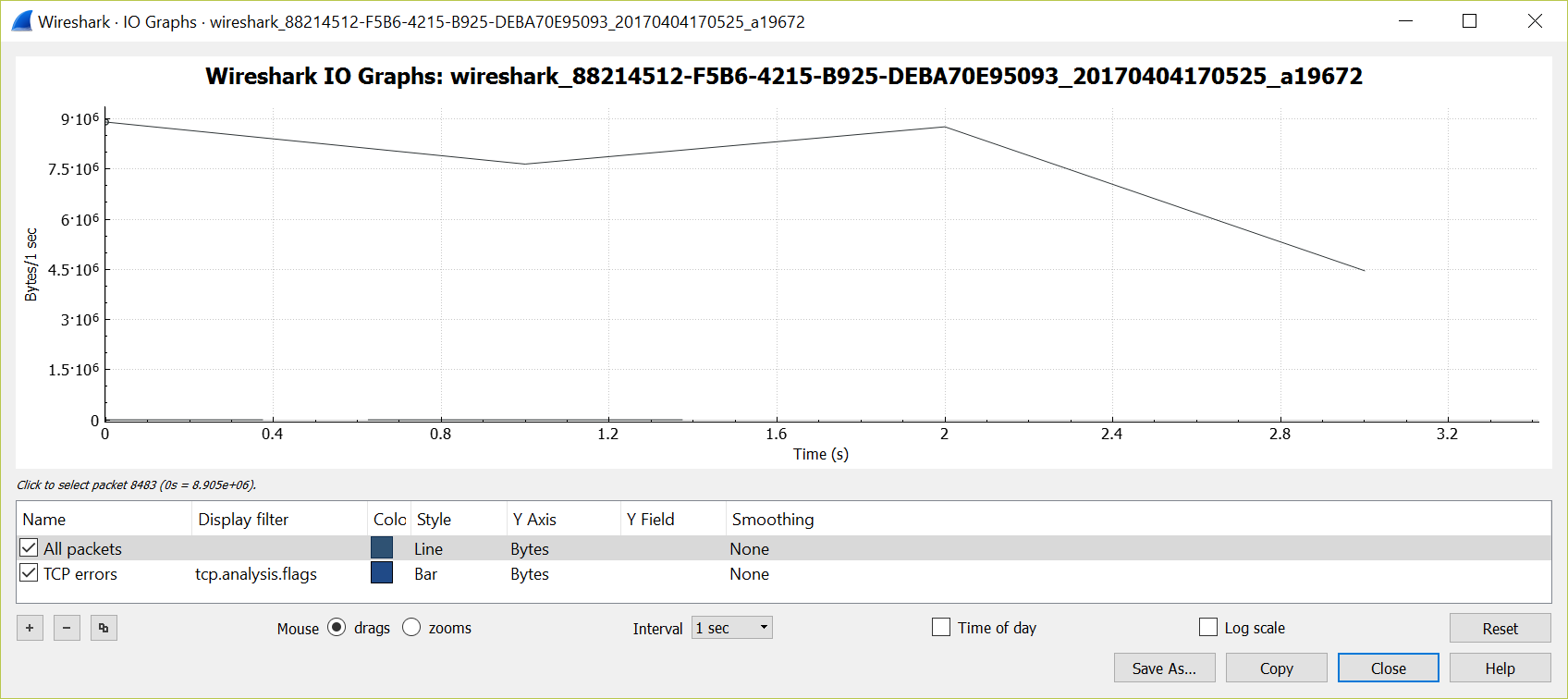
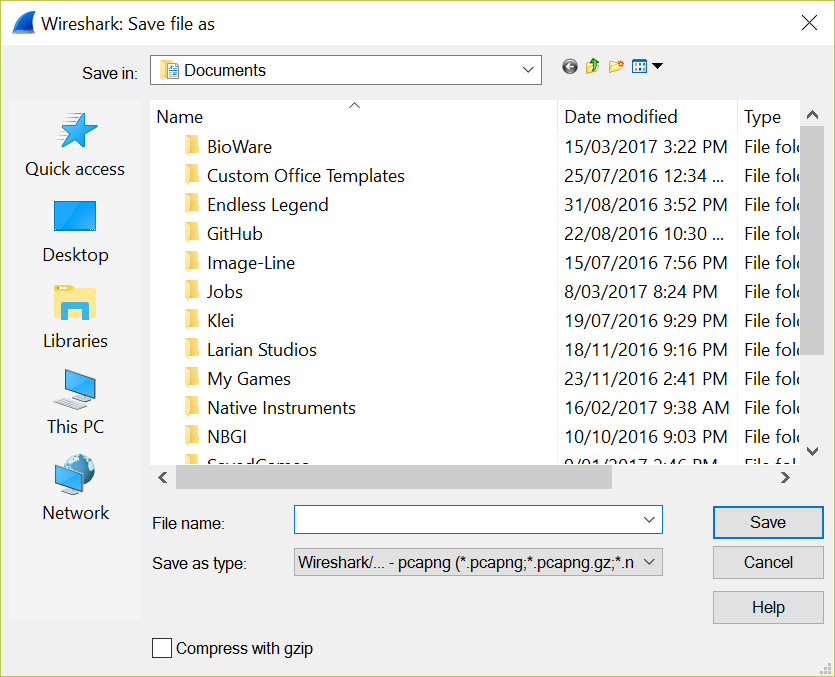


**Question 4**

Hands-On Project 1

1. After downloading the Wireshark installer file from wireshark.org, double click the installer.
2. Click “Next” button.  
   
3. Click “I Agree” button.  
   
4. Leave all the default options checked in the Choose Components screen and click “Next” button.  
   
5. On the Select Additional Tasks screen, leave default options checked and click “Next”.  
   
6. On the Install Location screen, either leave the default location or type in an alternate location, if desired.  
   
7. Wireshark installer will prompt to install WinPcap if it is not already installed. If it is not already installed, select the Install checkbox and click “Next” button.  
   
8. Wireshark installer will prompt to install USBPcap if it is not already installed. If it is not already installed, and you would like to capture USB traffic, select the install checkbox and click the “Install” button. I am choosing to not install USBPcap.  
   
9. The installer will start to install Wireshark.  
   
10. The WinPcap installer will appear, click “Next” button.
11. Click “I Agree” button.  
    
12. Leave the box checked for automatically starting WinPcap driver at boot, and click “Install” button.  
    
13. Click Finish on WinPcap installer.
14. Wireshark installer will complete, click “Next” button.  
    

Hands-On Project 2

1. Open Wireshark.
2. Click on Capture>Options.  
   
3. Under the Capture Interfaces window, select the interface that you would like to capture packets from and click “Start”. In this instance, I am choosing WiFi.  
   
4. Click stop button after capturing packets.
5. Wireshark is now showing information on the packets that were captured.  
   
6. Click Statistics>Conversations to view the conversations identified by Wireshark.  
     
   
7. In the packet details pane, expand the Ethernet II field to identify the MAC address of the device that sent the packet.  
   
8. Click Statistics>Protocol Hierarchy, which will open the Protocol Hierarchy window in order to view the packet size distribution of the packets in the trace buffer. The most common packet in my trace buffer are TCP packets with a packet size of 28,795,724 bytes.  
   
9. Using the packet bytes pane, the current packet is identified as a TCP packet (Protocol field = 0x06).  
   
10. Click on Statistics>IO Graph. This will show a graph illustrating the number of packets captured per second.  
    
11. Change the Y-Axis to bytes instead of packets.  
      
    
12. Click File>Save as to save the captured packets as a trace file.  
    

**Question 5**

Without taking growth into account, the following table shows the most efficient method of subnetting with the given information.

|  |  |  |  |
| --- | --- | --- | --- |
| Network ID | IP Range | Address usage (Current addresses/max) | Location |
| 10.0.0.0/23 | 10.0.0.0 - 10.0.1.255 | 312/510 | Building E |
| 10.0.2.0/24 | 10.0.2.0 - 10.0.2.255 | 200/254 | Building A |
| 10.0.3.0/24 | 10.0.3.0 - 10.0.3.255 | 135/254 | Building C |
| 10.0.4.0/25 | 10.0.4.0 - 10.0.4.127 | 125/126 | Building B |
| 10.0.4.128/25 | 10.0.4.128 - 10.0.4.255 | 122/126 | Building D |
| 10.0.5.0/25 | 10.0.5.0 - 10.0.5.127 | 105/126 | Building F |

The problem with this previous table of subnets is that there are a few subnets which don’t leave many addresses available for future growth. The table below is a more realistic subnet implementation. Building F would be the most vulnerable subnet for host growth with only 21 available addresses but this seems reasonable.

|  |  |  |  |
| --- | --- | --- | --- |
| Network ID | IP Range | Address usage (Current addresses/max) | Location |
| 10.0.0.0/23 | 10.0.0.0 - 10.0.1.255 | 312/510 | Building E |
| 10.0.2.0/24 | 10.0.2.0 - 10.0.2.255 | 200/254 | Building A |
| 10.0.3.0/24 | 10.0.3.0 - 10.0.3.255 | 135/254 | Building C |
| 10.0.4.0/24 | 10.0.4.0 - 10.0.4.255 | 125/254 | Building B |
| 10.0.5.0/24 | 10.0.5.0 - 10.0.5.255 | 122/254 | Building D |
| 10.0.6.0/25 | 10.0.6.0 - 10.0.6.127 | 105/126 | Building F |

**Calculations**

Building E being the largest subnet was the first building to be assigned a subnet, as I decided that it would be best to work from largest to smallest. Building E required 312 addresses and a standard octet can only assign 254 IP addresses –   
28 = 256 – 2 (1 address for network ID and 1 address for broadcast).  
Because of this, one of the network ID bits was used as a host ID bit. This resulted in a /23 subnet, which is able to assign 510 IP addresses –   
29 = 512 – 2.  
Due to the larger hosts section in this subnet, its range is 10.0.0.0 – 10.0.1.255.

Next, Building A required 200 addresses. Due to this address requirement being higher than 27 (128-2), I opted for a standard octet worth of address at 28 (256-2). This allows for a growth of 54 addresses which is reasonable.

Buildings C, B and D were all very close to the maximum address availability of a 7-bit subnet of 126 addresses (27-2). Because of this, the best choice was to assign a standard octet to each of these subnets.

Building F only requires 105 addresses, and due to this I decided to assign 7-bits to the host ID of this subnet. This allows the subnet to hold 126 addresses, leaving 21 addresses free within the subnet, which is reasonable.

**Large Growth**

If the host requirement for each building increased to over 1024, the subnetwork would need to be redesigned. The following table shows an appropriate solution allowing for more than 1024 hosts on each subnetwork.

|  |  |  |  |
| --- | --- | --- | --- |
| Network ID | IP Range | Maximum addresses | Location |
| 10.0.0.0/21 | 10.0.0.0 - 10.0.7.255 | 2046 | Building E |
| 10.0.8.0/21 | 10.0.8.0 - 10.0.15.255 | 2046 | Building A |
| 10.0.16.0/21 | 10.0.16.0 - 10.0.23.255 | 2046 | Building C |
| 10.0.24.0/21 | 10.0.24.0 - 10.0.31.255 | 2046 | Building B |
| 10.0.32.0/21 | 10.0.32.0 - 10.0.39.255 | 2046 | Building D |
| 10.0.40.0/21 | 10.0.40.0 - 10.0.47.255 | 2046 | Building F |

**Calculations**

Since each subnet required over 1024 addresses, the subnets had to be larger than 10-bits as this would only allow for 1022 address. 210 = 1024 – 2 (network ID and broadcast) = 1022.   
Due to this limitation, the host ID section of the subnet mask had to be 211, which gives a total of 2046 available addresses in each subnet (211 = 2048 – 2 = 2046).

**Question 6**

Quality of Service (QoS) is the concept of providing a better service to specific traffic on a network. QoS is a premium service that would ensure a higher level of service for whatever the user requires, this could include low latency, lowest delivered cost, assured delivery or expedited delivery (Pyles et al., 2017, p. 128).

Type of Service (TOS) is an 8-bit field within the IP header which defines a mechanism for assigning priority to IP packets so that routers can choose optimal routes for packets depending on their TOS fields. This field describes certain requirements of the service per packet, including minimal monetary cost, or delay, maximum reliability, throughput, or security (Pyles et al., 2017, p. 183). Using this information, routers are able to act more efficiently when routing data.

IP precedence is contained in the 3 most significant bits of the TOS field. This part of the field is useful when there is a queue of packets needing to be sent from a single-output interface (Pyles et al., 2017, p. 183). Being a 3 bit field, there are 8 available levels of precedence, priority increasing in ascending order. Level 0 is used for standard traffic that does not require any priority, levels 1 to 5 are used for assignable priority, and levels 6 and 7 are reserved for network and internetwork control packets.

Explicit Congestion Notification (ECN) is a field within the IP header which is used as a way for devices to notify each other that a link is currently congested (Pyles et al., 2017, p. 108). The intended outcome for the use of ECN is that the traffic rate will be decreased to compensate for the congestion instead of a router dropping packets. Reducing the number of dropped packets on a network significantly increases overall network performance.

In RFC 2474, the entire TOS field was redefined as Differentiated Services (DS). DS was introduced as a better way to manage network traffic and provide QoS. DS contains the Differentiated Services Code Point (DSCP) value in the 6 most significant bits and the ECN value in the 2 least significant bits. There are 3 sections in the DSCP, the Class Selector, Assured Forwarding and Expedited Forwarding. DSCP allows a router to manage traffic and assign priority to packets according to their own requirements, creating a more reliable network.

References

Pyles, J., Carrell, J. L., Tittel, E. (2017). *Guide to TCP/IP: IPv6 and IPv4, Fifth Edition.* Boston: Cengage Learning.