1. In the tcp\_connection.pcapng file, what is the well-known port used in the first TCP message?

The well-known port used in the first TCP message is the destination port 80. Port 80 is commonly used by HTTP, so it seems like this user was trying to connect to a website.

A screenshot of a computer

Description automatically generated with medium confidence

1. In the tcp.pcapng file, what is the demultiplexing key for TCP stream #1?

The TCP demux key is a combination of the source IP address, the source port, the destination IP address, and the destination port. This information is a 4-tuple with the form (Source Port, Source IP Addr, Destination Port, Destination IP Addr). In this example, the first stream can be found by finding the first instance of a SYN, SYN ACK, ACK between the same source/destination combination. This occurs at time 0.000, with a source IP address of 192.168.200.135 and a destination address of 192.168.200.21. By analyzing the header of that stream, we can see that the source port, associated with the IP address 192.168.200.135, is port 7875. The destination port is 2000, so the demux key is (7875, 192.168.200.135, 2000, 192.168.200.21).



A screenshot of a computer

Description automatically generated

1. If the TCP and IP headers contain no options, what will the MTU be given a MSS value of 492?

The Maximum Transmission Unit (MTU) is the sum of the IP header, TCP header, and Maximum Segment Size (MSS). The TCP and IP headers are both 20 bytes with no options. If we have an MSS of 492, then the MTU would be 492 + 20 (IP header) + 20 (TCP header) = 532 bytes.

1. If RWS equals zero, how will the sender know to resume transmitting?

The Receiver Window Size (RWS) indicates how much available buffer space the receiver has remaining. If the RWS equals zero, that means the receiver’s buffer is full, and the sender needs to wait until transmitting more data. When the RWS gets to zero, the receiver will send an ACK packet to the sender, which lets the sender know that the RWS is zero. The sender will then pause transmission. The receiver will keep sending ACK packets that indicate the RWS is zero until the buffer frees up, in which case the receiver will then ACK back with a RWS > 0. Since the sender has been monitoring the RWS size this entire time, once the ACK gets to the sender with an RWS > 0, the sender knows that it can resume transmission.

1. What TCP fields are used in TCP’s sliding window algorithm?

In order for the sliding window algorithm to work, the fields that each TCP packet must have are the sequence number, the ack number, the advertised window size, and the RWS. The sequence number helps the receiver to order received packets as well as detect any missing packets due to transmission errors. The ack number lets the receiver know which packet they should expect to get next, by keeping track of all the acknowledged packets so far. The sliding window algorithm also needs to know the advertised window size, so that the sender can determine how many packets it can send before it needs an acknowledgement back from the receiver. Finally, the sliding window algorithm needs to know the RWS, so that it can take advantage of congestion control methods to prevent congestion when the receiver’s buffer is full.

1. How does TCP protect against sequence numbers wrapping around?

TCP uses a timestamp mechanism to avoid sequence numbers wrapping around. Since network speed controls how quickly the sequence number would wrap around the given 32 bits, TCP opted to instead have a new 64-bit sequence number field, where the 32 low order bits are the sequence number, like we would expect, and the 32 high order bits are a timestamp. Because time only ever moves forward, the timestamp will continue increasing, so the receiver can use that to distinguish between two packets that might have the same sequence number. The one that came before, according to the time stamp, is the “first” packet, and the one that came later is the “second” packet. This way, the receiver can always tell packets that have the same sequence number apart from one another.

1. Why is the PSH flag set in message no. 24 in tcp.pcapng?

By identifying the bytes in packet 24, we can see that the data segment of this packet contained the following text:

A blue screen with white text

Description automatically generated with low confidence

If we follow the whole TCP stream, we can see that the message sent over this stream was the following:

A screenshot of a computer error

Description automatically generated with medium confidence

This is the end of the stream. We can see that the full message ends at “[Page 6]” and the last bits of data in packet 24 are also “[Page 6]”. This would indicate that the PSH (Push) flag was set to indicate that the sender had sent all their data, or in other words, the PSH flag was set as an indication of a record boundary. The PSH flag in this case is indicating the end-of-record, so now the receiving end can break the TCP stream into records, where the full message sent from the sender is broken cleanly into one single record.

1. For TCP stream #1, what is the window scaling factor for the sender and receiver?

For TCP stream #1, in tcp.pcapng, we can see that the Window Scale is indicated in the options bits of the TCP header. For the sender (source IP 192.168.200.135), we can see that the window scale is 8 (multiply by 256)

A screenshot of a computer program

Description automatically generated with medium confidence

And for the receiver (IP 192.168.200.21), we can see the window scale is set to 7 (multiply by 128)

A screenshot of a computer program

Description automatically generated with medium confidence