ECE 458

Communication Networks

Laboratory Experiment #2 Report

Introduction

The Transmission Control Protocol (TCP) is one of the most widely used Internet protocols. It connects two endpoints, generally a client and a server, to deliver a stable and in-order streaming service. TCP ensures this by identifying whether packets are delayed, damaged, or Lost [1]. In this lab we further investigate and observed the behavior of the following TCP material:

- TCP headers format
- TCP 3-way handshakes
- TCP Reliable data transfer
- TCP Congestion and Flow control
- TCP retransmission

By studying the trace of TCP segments transmitted and received while transferring a 300 KB file from a local computer to a remote web server, we were able to gain a better understanding of TCP's behavior. Using a file named "alice.txt" which is saved on the client computer and via the HTTP POST technique is uploaded to the gaia.cs.umass.edu server. The TCP segments sent/received to/from the client computer are seen using Wireshark and are provided by the lab.

Procedure

The following tools were used to complete Lab 2:

- *alice.txt* file with the server *gaia.cs.umass.edu*
- Trace file *tcp-trace-1.pcap*
- Trace file from a private network *wcp-trace-retransmission.cap*
- Wireshark for observing the traces

Using Wireshark to observe *tcp-trace-1.pcap* the TCP header and port numbers were determined. Using the trace file, the three-way handshake was determined and the SYN segment sent from the client to server and SYN/ACK returned was observed. With the segments 4 to 15, the sequence and acknowledgment numbers, lengths, and sent/received time were recorded and then used to determine the segments acknowledged and the Round Trip Time.

We used our knowledge of two segments in the FIN-ACK sequence to determine the packets used to close the data flow from the server to the client, and vice versa. After, we observed how the congestion window doubled its size in every RTT during the slow start phase, and used this understanding to calculate the congestion window during the different RTT intervals. Finally, with the *tcp-trace-1.pcap* trace we determined why the sender stopped sending segments at the 179th segment, and its receiver's window size.

Lastly, The *wcp-trace-retransmission.cap* trace is used to observe the retransmissions in Wireshark. Segment 12 is the first retransmission, knowing this we determined the segment that identifies such a segment as a retransmission and find the three acknowledgments which triggered fast retransmission. With the various Segments we identify whether the various segment's are fast or timeout retransmissions. [2]

Discussion

4.3.1

1. TCP header content in hexadecimal format: **09 54 00 50 b9 3c 1f 07 00 00 00 00 70 02 40 00 e0 ae 00 00 02 04 05 b4 01 01 04 02**.
2. The TCP port number for the client computer (source) is **2388**, and the server (destination) port is **80**. The port number is determined by the TCP implementation software from a range of port numbers ranging from 1024-65535
3. The maximum header size for TCP is 60 bytes. At offset 0x0020, the value of the byte is 0x70. Then, the header consists of 7×32-bit words, which means 7×4 bytes $= 28$ bytes, which is the same as the Header Length value of 28 bytes.

4.3.2

1. Segment number 1 and 2 are the initial three-way handshakes in the trace file.
2. The sequence number for segment 1 in hexadecimal format is **b9 3c 1f 07**, whereas the sequence number for segment 2 is **b9 3c 1f 08**.
3. The acknowledgement number in the SYN/ACK segment is **b9 3c 1f 08** in hexadecimal. The value of the acknowledgement number is the sequence number of the SYN segment plus 1, which gives an acknowledgment number of b9 3c 1f 08.
4. The sequence number of the third ACK segment in hexadecimal is **b9 3c 1f 08** and the acknowledgment number is **86 61 d8 9b**. The sequence number is determined from the SYN/ACK segment, and the acknowledgement number of the third SYN segment is the sequence number of the next byte expected from the other side.
5. The client and server announce the maximum TCP payload size they're willing to accept is done during the TCP handshake, both devices communicate the size of the packets they are able to receive. The maximum segment size for the three-way handshake is 1460 bytes; this value was chosen because it's the default for the TCP protocol.
6. There is no data sent in the SYN, SYN/ACK messages, but there is data in the ACK message, since the sequence number was incremented.

4.3.3

1.

Packet No.	Data Segments 10.0.1.5 -> 128.119.245.12				ACK Segments 128.119.245.12 -> 10.0.1.5		
	Seq. No./ Relative Seq. No.	Data Length	Time (s)		Ack. No./ Relative Ack. No.	Data Length	Time (s)
4	3107725064 / 1	673	0.000254		2254559387 / 1	673	N/A
5	3107725737 / 674	1460	0.005810		2254559387 / 1	1460	N/A
6	2254559387 / 1	N/A	N/A		3107725737 / 674	0	0.109230

7	3107727197 / 2134	1460	0.000046		2254559387 / 1	1460	N/A
8	3107728657 / 3594	1460	0.000021		2254559387 / 1	1460	N/A
9	2254559387 / 1	N/A	N/A		3107727197 / 2134	0	0.008354
10	3107730117 / 5054	1460	0.000041		2254559387 / 1	1460	N/A
11	3107731577 / 6514	1460	0.000021		2254559387 / 1	1460	N/A
12	2254559387 / 1	N/A	N/A		3107728657 / 3594	0	0.106536
13	3107733037 / 7974	1460	0.000043		2254559387 / 1	1460	N/A
14	3107734497 / 9434	1460	0.000033		2254559387 / 1	1460	N/A
15	2254559387 / 1	N/A	N/A		3107730117 / 5054	0	0.003282

2. Segments 6, 9, 12 and 15 acknowledge the following packets respectively: 4, 5, 7, 8.
3. Segments 4, 5, 7 and 8 have been acknowledged before the 15th segment, and their respective RTT values are the following: 0.11504000, 0.11765100, 0.11497300, 0.11831000.
5. The sequence number doesn't change when the segment contains an acknowledgement for a packet, i.e. stays as 2254559387 / 1. The sequence number doesn't change because there isn't any data being sent to the client. When there is data being sent, the sequence number is calculated by the following formula: $previous\ seq\ \# + data\ length$, i.e. for segment 4, the initial sequence number is 1 and the data length is 673, so the next sequence number would be $1 + 673 = 674$.

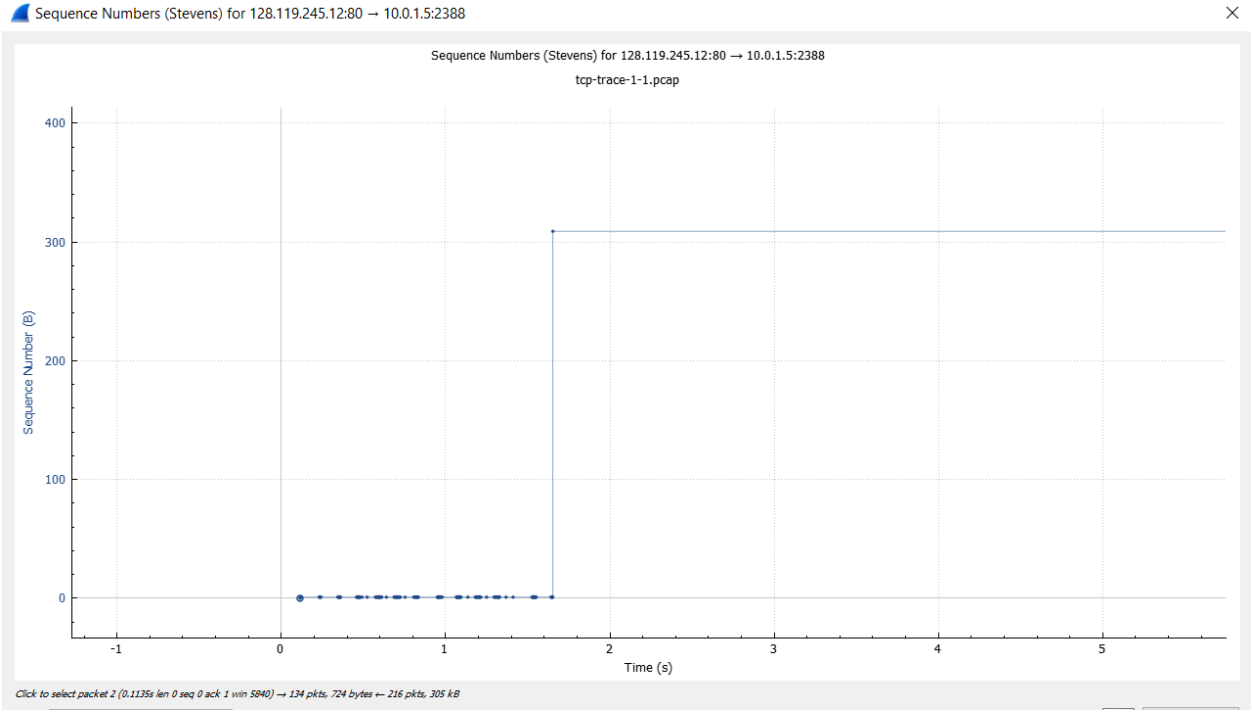
4.3.4

1. Segments 348 and 349 are used to close the data flow from the server to the client. This is determined by finding the [FIN, ACK] and [ACK] messages, which the server will send and wait until the FIN is returned with an ACK.
2. Segment 350 is used to close the data flow from the client to the server. This is determined by the [RST, ACK] message, which is sent by the client doing the active close, since it sent the last ACK.

4.3.5

1. For every two TCP segments sent from the client (10.0.1.5) to the server faia.cs.umass.edu (128.119.245.12) only one TCP is sent vice versa. The TCP data flow has such a pattern because it uses a 3-way handshake to establish a reliable connection.

2. The initial congestion window size (cwnd) is 1. Segment 5 has a cwnd of $2^1 = 2$, segment 8 has a cwnd of $2^2 = 4$, segments 11 and 14 would have a cwnd of 4, since their acknowledgements aren't received until after the 15th packet.
3. The congestion window doubles its size in every RTT in the slow start phase. The initial congestion window size (cwnd) is 1, the congestion window during the second RTT is 2, and the congestion window during the third RTT is 4. The segments corresponding to the aforementioned congestion window sizes would be 4, 6 and 9 respectively. Segments 4 and 5 are in the first RTT, segments 6, 7 and 8 are in the second RTT, and segments 9, 10 and 11 are in the third RTT.
4. We can create a Time Sequence (Stevens) graph via selecting a segment -> Statistics -> TCP Stream graphs -> Time Sequence (Stevens).



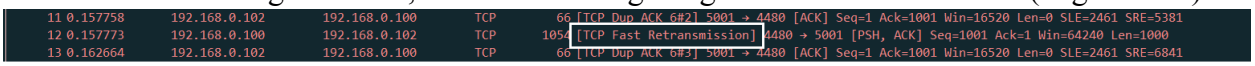
Here we can determine the sender's congestion control change from the slow start phase to the congestion avoidance phase at segment 346 and time 1.652404 s. During the slow start phase, the congestion window increases one MSS with each acknowledgment, and the window size is doubled in every RTT. During congestion avoidance, each acknowledgment increases the congestion window by $MSS^2/congestion\ window\ size$, and so the congestion window size is increased by one MSS every RTT. We know the Slow start phase changes to congestion avoidance phase when the congestion window exceeds the slow-start threshold. [3]
$if\ cwnd\ \leq\ ssthresh$, do slow start
$if\ cwnd > ssthresh$, do congestion avoidance

4.3.6

1. The sender briefly stopped sending more segments, because segment 180 acknowledged the segment in frame 159. The calculated window size is 17520. This can be determined by checking the [Calculated window size:] field under the Transmission Control Protocol section.

4.3.7

1. In Wireshark we can determine that the segment is a retransmission by observing the Segment info. If the segment is a retransmission, it may say TCP Fast Retransmission or TCP Retransmission.
2. To determine if Segment 12 is a fast retransmission we must determine if the duplicated acknowledgments should acknowledge the same acknowledgment number, which is the sequence number of the fast retransmission. The tree acknowledgments which triggered the fast retransmission of segment 12 is contained within Segments 6, 9, 11.
3. Segment 44 is a timeout retransmission. The sequence number in the segment has not been acknowledged three times. In addition, a fast retransmission would be labeled as such in the segment info, see the following image for a fast transmission (Segment 12):



Conclusion

TCP is a connection-oriented protocol that provides reliable delivery for applications that generate large amounts of data. In the case of this lab, we investigated the behaviors of TCP in detail, by analyzing the trace of TCP segments sent and received when transferring a 300 KB file called '*alice.txt*' from a local computer client to a web server.

In this lab we became learned about the following TCP topics in this order:

1. TCP header's format
2. TCP 3-way handshake
3. Reliable data transfer
4. Congestion control algorithm
5. Retransmission scheme

Using the *tcp-trace-1.pcap* trace file we learned about three-way handshakes and how to determine the initial sequence and acknowledgment number. Afterwards, we learned how to determine the sequence and acknowledgment numbers and how to determine the congestion window relative to the RTT.

We learned how to identify the features of a retransmission by observing the *tcp-trace-retransmission.cap* trace file in Wireshark.We also learned how to recognise and differentiate between a fast and a timeout retransmission.

The end result of this lab is a strong understanding of the behavior of TCP between a client computer and web server using Wireshark.

References

[1] “What is Transmission Control Protocol TCP/IP Model?,” *Fortinet*. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/tcp-ip. [Accessed: 21-Feb-2022].

[2] “Laboratory manual for ECE458 Communications Networks - uvic.” [Online]. Available: https://studentweb.uvic.ca/~wenjunyang/ECE458/Lab_manual.pdf.

[3] “TCP slow start graph edit,” *TCP Slow Start Graph - Ask Wireshark*. [Online]. Available: https://ask.wireshark.org/question/5043/tcp-slow-start-graph/.

Feedback

The questions were often difficult to understand without the aid of the lab instructor (e.g. 4.3.3 Question 2).