Department of Electrical and Computer Engineering

ECE 363

Communications Networks

Laboratory 4

Name	Arfaz Hossain
Student no.	V00984826
Date	March 30, 2024
Lab Title	Transmission Control Protocol
Lab Section	B03

Introduction

In this lab, we delve into the intricacies of TCP, the cornerstone of internet communication. Through the analysis of packet traces and protocol interactions, we uncover the mechanisms behind TCP's reliable data transmission, congestion control strategies, and connection management. We explore the TCP header format; we examine its fields and their roles in facilitating efficient communication. We then dissect the three-way handshake process, where connections are established, and observe how TCP adapts to network conditions using congestion control mechanisms. We explore TCP's flow control mechanisms, understanding how receivers manage data and communicate their buffer capacities. Additionally, we investigate TCP's retransmission schemes, crucial for maintaining data integrity and reliability. This lab equips participants with a comprehensive understanding of TCP's principles, essential for effective network communication.

Procedure

We began by opening the provided trace file, `tcp-trace-1.cap`, which captures the TCP traffic involved in a client uploading the contents of *Alice in Wonderland* to a remote server. Within Wireshark, we filter the captured packets to focus solely on TCP segments containing HTTP messages, ensuring our analysis centers on TCP-specific behaviors. Next, we delve into the details of the TCP header, examining sequence numbers, acknowledgments, and payload data to understand the intricacies of data transmission. Our analysis progresses as we scrutinize the three-way handshake process, identifying the SYN, SYN/ACK, and ACK packets exchanged between the client and the server to establish the TCP connection. Subsequently, we observe the transfer of the HTTP POST command and the file `alice.txt`, tracking the sequence of TCP segments involved in the data exchange. As we continue our examination, we pay close attention to the closure of the TCP connection, identifying packets with the FIN bit set to initiate connection termination and the corresponding ACK packets. Throughout the procedure, we summarize key TCP behaviors and protocol interactions, gaining insights into connection establishment, data transfer, and connection closure within the captured trace file.

Discussion

2.1 TCP Header format

1. **Write down the TCP header content in hexadecimal format (in the packet bytes pane). Inspect the TCP header and indicate the value of each field in the header. Annotate the hexadecimal content to explain your answer.**

   Using packet 1, the TCP header contains the following fields:

   - Source port : 0x0954 (2388)
   - Destination port : 0x0050 (80)
   - Sequence No. : 0xb93c1f07 (3107725063)[1]
   - Ack. No. : 0x00000000 (0)
   - Header Length : 0x7 (28 bytes)
   - Flags : 0x002 (SYN)
   - Window size value : 0x4000 (16384)
   - Checksum : 0xe0ae (invalid, disabled)
   - Urgent pointer : 0x0000 (Not urgent)
   - Option 1 : **Kind**: 0x02 (MSS), **Length**: 0x04 (4), **MSS Value**: 0x05b4 (1460)
   - Option 2 : **Type**: 0x01 (NOP)
   - Option 3 : **Type**: 0x01 (NOP)
   - Option 4 : **Kind**: 0x04 (SACK Permitted), **Length**: 0x02 (2)

2. What are TCP port numbers used by the client computer (source) and the server (destination) when transferring the file to gaia.cs.umass.edu? How did the client computer determine the port numbers when it wanted to set up a TCP connection to the server? *The client computer uses port 2388, while the server uses port 80 when transferring files to* `gaia.cs.umass.edu`*. The client determines the port numbers by assigning a non-reserved port and assuming the server is using the well-known port for HTTP traffic.*

3. What is the maximum header length? Given the value of the Header Length field, how to calculate the length of the header in the unit of bytes? Verify your answer using the first TCP segment in the trace file. *The maximum value of a header length is 60 bytes [1]. The field is composed of four bits and corresponds to the number of 32-bit (4 byte) words in the header. The first segment has a Header Length of 7 and this corresponds to the 28-byte header.*

---

4. (Optional) How does TCP calculate the Checksum field? What is the pseudo-header format? Write down the pseudo-header of the flow from the client to the server in hexadecimal format. Verify the Checksum value in the first TCP segment in the trace file.
The checksum algorithm is:

1. Divide the hex value of the header and data into 16-bit groups. A pseudo-header must be created. The checksum field is filled with 0s. Bits are right padded to fill a 16-bit group.
2. Divide the sum by 0xFFFF and add the remainder.
3. Find the 1s complement of this number and use it as the checksum. The pseudo-header for packet 1 is 12 bytes consisting of data from the IP header[2].
   - ⊗ Source IP address, 0x0a000105 (4 bytes)
   - ⊗ Destination address, 0x8077f50c (4 bytes)
   - ⊗ Reserved bits, 0x00 (1 byte)
   - ⊗ Protocol, 0x06 (1 byte, corresponds to TCP)
   - ⊗ TCP Length 0x0028 (2 bytes, corresponds to TCP header and data length)

2.2 TCP Connection Setup

1. Which segments are the initial three-way handshake in the trace file? How do you find them? *The initial handshake occurs in segments 1, 2 and 3. Segments 1 and 2 have SYN flag set, which identify them as connection initiators. Segment 3 is the ACK to the server's SYN in segment 2.*

2. What is the actual initial sequence number in each direction (in hexadecimal format)?

Client → Server: 0xb93c1f07
Server → Client: 0x8661d89a

3. What is the value of the acknowledgement number in the SYN/ACK segment? How did `gaia.cs.umass.edu` **determine that value?** *The ACK number is 0xb93c1f08. It represents the next sequence that it can receive and is one greater than this Client → Server initial sequence number.*

4. What are the values of the sequence number and the acknowledgment number in the third ACK segments in the three-way handshake? How did the client determine these values? *The sequence number is 0xb93c1f08 and is the segment requested by the server in segment 2. The ACK is 0x8661d89b and corresponds to an acknowledgment of the server's SYN request.*

5. How did the client and the server announce the maximum TCP payload size that they were willing to accept? What are the values and why did they choose these values? *Both the client and server announced the maximum TCP payload size they were willing to accept by including a TCP option specifying an MSS of 1460 bytes in their SYN requests. This value corresponds to the optimal Ethernet packet size of 1500 bytes minus two 20-byte headers.*

6. Is there data sent in the SYN, SYN/ACK, and ACK segment? *No.*

---

[1] *Wireshark assigns this a relative value of 0.*

2.3 TCP Data flow

1. Beginning with the 4th segment, what are the sequence number, acknowledgement number, data length, and the time of the segment sent/received from/to the client computer of the 4th, 5th, 6th, . . ., 15th segments in the TCP connection?

Packet No.	Data Segments 10.0.1.5 → 128.119.245.12			ACK Segments 128.119.245.12 → 10.0.1.5		
	(Relative) Seq. No.	Length (b)	Time (s)	(Relative) Seq. No.	Length (b)	Time (s)
04	(1) 0xb93c1f08	673	0.000000	—		
05	(674) 0xb93c21a9	1460	0.005810	—		
06	—			(674) 0xb93c21a9	0	0.115040
07	(2134) 0xb93c275d	1460	0.115086	—		
08	(3594) 0xb93c2d11	1460	0.115107	—		
09	—			(2134) 0xb93c275d	0	0.123461
10	(5054) 0xb93c32c5	1460	0.123502	—		
11	(6514) 0xb93c3879	1460	0.123523	—		
12	—			(3594) 0xb93c2d11	0	0.230059
13	(7974) 0xb93c3e2d	1460	0.230102	—		
14	(9434) 0xb93c43e1	1460	0.230135	—		
15	—			(5054) 0xb93c32c5	0	0.233417

2. What are the segments acknowledged by packet 6, 9, 12, and 15, respectively?

$6 \xrightarrow{\text{ACK}} 4$; $9 \xrightarrow{\text{ACK}} 5$; $12 \xrightarrow{\text{ACK}} 7$; $15 \xrightarrow{\text{ACK}} 8$

3. Given the difference between the time each TCP segment was sent and the time its acknowledgement was received, what is the RTT value for each of the segments which have been acknowledged before the 15th segment?

Segment 4: 0.115040s; **Segment 5**: 0.117651s; **Segment 7**: 0.114973s; **Segment 8**: 0.118310s

2. In the trace file, how did the sequence number of the packets from the server to the client change? Why? *The sequence number did not increment since no data was sent from the server to the client.*

6. (Optional) At the end of the trace file, find the TCP segments used by the server to transfer the congratulation web page to the client computer. How do you determine this? *Segment 347 is an ACK from the client to the server corresponding the TCP segment number 345, indicating that the client received data from the server. The TCP data in segment 345 and HTTP data in segment 346 correspond to the HTTP 200 OK response from the server which contains the congratulations page.*

7. (Optional) Are there any retransmitted segments in the trace file? What do you check for (in the trace) to answer this question? *Retransmissions can be detected with filter:* `tcp.analysis.retransmission`*. This file does not contain any retransmissions.*

2.4 TCP Connection Release

1. Which packets were used to close the data flow from the server to the client? How do you determine this? *Packets 348 and 349 close the server → client connection. 348 is a FIN, which requests to close the connection from the server to the client. The ACK in 349 confirms this closure.*

2. Which packets were used to close the data flow from the client to the server? How do you determine this? *Packet 350, an RST request from the client to the server, was used to close the data flow from the client to the server. This determination is made based on the nature of the RST request, which terminates all existing connections and does not require an acknowledgment (ACK).*

3. (Optional) In the FIN segment, what is the sequence number? In the corresponding ACK segment, what is the acknowledgement number? How did the client determine this number? *The sequence number in the FIN segment*

is 725, while the acknowledgment number in the corresponding ACK segment is 726. The client determined this acknowledgment number by following the standard procedure of incrementing the last received sequence number by one. Nothing noteworthy occurred in this process.

2.5 TCP Congestion Control

1. Examine the 4th to 15th TCP segments and take a reference to the Table in Question 1 of Section 2.3. Can you find a pattern of the number of segments sent from the client and from the server gaia.cs.umass.edu? Why did the TCP data flow have such a pattern? *Two packets are sent from the client before one packet is sent from the server. This pattern corresponds to the exponential growth of the packet transmission rate in the slow start period.*

2. What is the initial size of congestion window? How do you determine this? What is the size of congestion window when segments 5, 8, 11 and 14 were sent out? *The congestion window is equivalent to the number of packets in the receiver buffer. It can be determined by subtracting the number of ACKs from the number of packets sent (assuming no loss).*

Segment 5 sent:	1 segment in receiver buffer.
Segment 8 sent:	2 segments in receiver buffer.
Segment 11 sent:	3 segments in receiver buffer.
Segment 14 sent:	4 segments in receiver buffer.

Note, the packet being sent (5, 8, . . .) is not counted as in the receiver's buffer.

3. In the lecture we have learned that the congestion window doubles its size in every RTT in the slow start phase. Beginning with the 4th packet, what is the size of the congestion window and which packet were inside the congestion window (i.e., these packets could be sent) during the first RTT? What is the size of the congestion window and which packet were inside the congestion window during the second RTT? How about the third RTT? Give the segment numbers.

Right Trip	Segment → ACK	Segments in buffer
1	4 → 6	5
2	5 → 9	7, 8
3	7 → 12	8, 10, 11

4. When did the senders congestion control change from the slow start phase to the congestion avoidance phase? Give the segment number and the time. How do you determine this? *The behavior changes at packet 67, which is 0.475266 s after packet 4. At this point, the client sends out one packet per ACK. This is because the server has ACK'ed the first 29874 b and has a window of 32767 b. Hence, the largest sequence number can receive is 62641. Packet 67 will send data up to segment 61993. The next segment will cover 61994 → 63454, which will overflow the receiver's buffer. Hence, it must wait for an ACK with sequence number 61994 before sending this packet.*

2.6 TCP Flow control

1. Examine the 179th segment in the trace file, why did the sender stop sending more segments? What is the size of receiver window advertised by the receiver at this moment? How do you determine this? *Packet 157 has an ACK for segment 131746 and a window of 32767. Hence, the largest segment it can receive is 164513. Packet 179 contains the highest segment value of 163865. Sending out another packet of 1514 b packet will overflow the receiver window. Hence, the next packet is sent out only after an ACK is received.*

Conclusion

The TCP header provides crucial details for connection establishment, including port numbers and advertised capabilities such as SACK. It also relays the current state of the hosts' buffers via the ACK and window length fields. In the slow start phase, the sender sends two segments for every ACK received. As transmission progresses, the number of packets in the receiver's buffer doubles with each round trip. Once the sender nears filling the receiver's buffer, it shifts to congestion avoidance mode and sends one segment per ACK. When the transfer completes, a FIN request is sent to initiate connection termination, culminating in termination upon sending the RST packet.

Reference

1. Wikipedia Contributors, "Transmission Control Protocol," Wikipedia, Mar. 15, 2019. *https://en.wikipedia.org/wiki/Transmission_Control_Protocol*
2. "How to Calculate IP/TCP/UDP Checksum" GitHub Open-Source Contributors, *https://gist.github.com/drpresq/ecf699f05178dd280ca14a2577e004f2* (accessed Mar. 31, 2024).
3. A. Tanenbaum and D. Wetherall, Computer Networks, 5th ed. Prentice Hall, Oct. 2010.
4. J. F. Kurose and K. W. Ross, Computer Networking: A Top-Down Approach, 5th ed. Addison-Wesley Publishing Company, USA, 2009.