University of Victoria

CENG 460

Computer Communication Networks

Lab 4 - TCP

Instructor:
Dr. Lin Cai

Teaching Assistant:
Amir Andaliby

Tyler Stephen V00812021
A01 - B04

April 4, 2016

1 Introduction

This lab will examine the TCP header fields in depth, as well as several protocol behaviors. The three-way handshake for connection setup is examined, followed by a reliable data flow. The throttling effects of congestion control are observed in the sender's transmission rate. Finally, the connection release behavior is examined.

2 Procedure and discussion

`tcptrace1.cap` contains traffic where a client uploads a the contents of *Alice in Wonderland* to a remote server. An HTTP POST initiates the upload and an HTTP 200 OK response is issued from the server when the upload is complete. The TCP traffic in the file captures the entire conversation, starting with the three-way handshake, then the file transfer and finally the connection closure.

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]
- Acknowledgement 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

[1] Wireshark assigns this a relative value of 0

  - 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 uses port 2388 and the server uses port 80. The client assigns a non-reserved port for use and assumes the server is using the well known port number for HTTP traffic (80).

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 and sum
>   - A pseudo-header must be created (see later)
>   - 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 [2].

The pseudo-header for packet 1 is 12 bytes consisting of data from the IP header:

1. Source IP address, `0x0a000105` (4 bytes)
2. Destination address, `0x8077f50c` (4 bytes)
3. Reserved bits, `0x00` (1 byte)
4. Protocol, `0x06` (1 byte, corresponds to TCP)
5. TCP Length `0x0028` (2 bytes, corresponds to TCP header and data length) [2]

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?

The SYN requests from both the client and server contain a TCP option specifying an MSS of 1460 bytes. This corresponds to the optimal ethernet packet size of 1500 bytes, less two 20 byte headers.

6. Is there data sent in the SYN, SYN/ACK, and ACK segment?

No.

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)
4	(1) `0xb93c1f08`	673	0.000000		—	
5	(674) `0xb93c21a9`	1460	0.005810		—	
6		—		(674) `0xb93c21a9`	0	0.115040
7	(2134) `0xb93c275d`	1460	0.115086		—	
8	(3594) `0xb93c2d11`	1460	0.115107		—	
9		—		(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?

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

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: 0.115040 s
Segment 5: 0.117651 s
Segment 7: 0.114973 s
Segment 8: 0.118310 s

4. (Optional) What is the Estimated RTT value after the receipt of each ACK? Assume that the value of the Estimated RTT is equal to the measured RTT for the first segment, and then is computed using the Estimated RTT equation for all subsequent segments.

—

5. 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) in order 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 is a RST request from the client to the server. This terminates all existing connections and does not need an 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?

FIN sequence number: 725
ACK sequence number: 726
As with all other ACKs, it is one up from the last received sequence. Nothing special happens.

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 segment 5, 8, 11 and 14 were sent out?

> The congestion window is equivalent to the amount 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 segments 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.

Round 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 ACKed the first 29874 b and has a window of 32767 b. Hence, the largest sequence number is 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 a 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.

2.7 Retransmission in TCP (Optional)

1. Segment 12 is the first retransmission. What is it in the segment that identifies the segment as a retransmission? Which segment was segment 12 retransmitted for?

The sequence number in 12 is 1001 (relative). This is the same sequence number as segment 5. Hence, segment 12 is a retransmission of segment 5.

2. Segment 12 is a fast retransmission, which should be triggered by triple- duplicated-acknowledgment. Find the three acknowledgments which triggered the fast retransmission of segment 12.

Segments 6, 9 and 11 contain ACKs for segment 1001.

3. Is segment 44 a fast retransmission or timeout retransmission? How do you determine this?

It's a timeout retransmission. The segment has only been ACKed once, in segment 43. There's also a 1.47 s delay between the ACK in segment 43 and the retransmission in 44. This long delay is indicative of a timeout.

3 Conclusion

The TCP header contains useful information for connection setup — such as the port numbers and advertised capabilities like SACK — and for relaying the current state of the two hosts' buffers via the ACK and window length fields. During the slow start phase, the sender will send two segments for every ACK received. As the transmission continues, the amount of packets

in the receiver's buffer doubles during every round trip. When the sender comes close to filling the receiver's buffer it transitions to congestion avoidance operation and sends one segment per ACK. After the transfer is complete, a FIN request is sent to terminate the connection. The termination occurs when the RST packet is sent.

4 Feedback

1. Use some statistics tools in Wireshark to analyze throughput, etc.
2. Use the conversation analyzer in Wireshark to examine a more complicated trace, perhaps with many hosts.

References

[1] Wikipedia. (2016). Transmission control protocol, [Online]. Available: https://en.wikipedia.org/wiki/Transmission_Control_Protocol#TCP_segment_structure.

[2] roman10.net. (Nov. 27, 2011). How to calculate ip/tcp/udp checksum, [Online]. Available: http://www.roman10.net/2011/11/27/how-to-calculate-iptcpudp-checksumpart-1-theory/.