EP2120 TCP Lab (Lab 2) Report

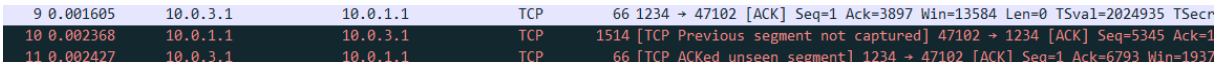
[redacted]

Introduction

In this laboratory assignment, our group of four students utilized two computers, with each computer controlling a Host and a Router through the Linux system. By managing the data transmission between the two Hosts, we simulated various possible scenarios in TCP data transmission to explore the characteristics of TCP. This hands-on approach served to enhance our understanding of TCP.

Measuring TCP with ttcp and Wireshark

Note: There are mistakes in our .pcap file from Wireshark, which fails to capture one of the packet with length 1514 bytes and its corresponding ACK. According to [3], the error we encountered "TCP Previous Segment not Captured" is a sign added by Wireshark. So, the packet is actually received by Host B but is not detected by Wireshark, as shown in the following figure.



Hence, the answers to the questions of this part will not consider this error mentioned above. For example, the total transmitted packets shown in output_5.pcap of ours is 18 packets, but it is actually 20. And in the answers to the following questions, we consider the number of packets to be 20.

(a) How many packets are transmitted in total (count both directions)?

A: 20 packets

(b) What is the range of the sequence numbers used by the sender (Host A)?

A: From Seq=0 to Seq=10002

(c) How many packets do not carry a data payload?

A: 12 packets

(d) What is the total number of bytes transmitted in the recorded transfer? From those, calculate the amount of user data that was transmitted?

A: Total number of bytes transmitted is 11.336 Kbytes, including the handshakes, transmitted segments and ACKs. We suppose the amount of user data transmitted is equal to the number of valid message we intend to send, which is 10*1000=10 Kbytes.

(e) Compare the total amount of data transmitted in the TCP data transfer to that of a UDP data transfer. Which of the protocols is more efficient in terms of overhead? What is the efficiency in percentage for these two protocols?

A: UDP is more efficient in terms of overhead, since UDP segment has smaller length of headers compared to TCP, and no ack exists in the transmission process, which reduces the overhead and thus, increases the efficiency. The efficiency of TCP should be 10/11.336=88.21%. The efficiency of UDP should be 10000/10420=95.97%. One thing is for sure, efficiency of UDP is higher than that of TCP in this given scenario.

TCP connection management

This exercise provides a practical understanding of how TCP connections are established and terminated. In this exercise, we will utilize the Telnet application to initiate and conclude TCP connections.

Connection establishment and termination

(a) Which packets constitute the three-way handshake? Which flags are set in the headers of these packets?

A: The first three TCP packets constitute the three-way handshake. The first packet is sent by host A used to synchronize with flag 0x002 (SYN). The second packet is sent by host B used to synchronize and acknowledge with flag 0x012 (SYN,ACK). The third packet is sent by host A used to acknowledge with flag 0x010 (ACK).

(b) What are the initial sequence numbers used by the client and the server, respectively?

A: The initial sequence number is 0 for both client and server.

(c) Which packet contains the first application data?

A: The fourth packet with 93 contains the application data.

(d) What are the initial window sizes for the client and for the server?

A: The window size is 29200 bytes and 5792 bytes for client and server respectively.

(e) How long does it roughly take to open the TCP connection?

A: Th time roughly take to open the TCP connection is between 0.000736 and 0.001122 seconds.

(a) Which packets are involved in closing the connection?

A: The last three packets are involved in closing TCP connection.

(b) Which flags are set in these packets?

A: The flags are 0x011 (FIN, ACK), 0x010 (FIN, ACK) and 0x010 (ACK) for the first, second and third packet respectively.

Connecting to a non-existing port

(a) How does the server host (Host B) close the connection?

A: Host B sends a TCP SYN (Synchronize) packet to Host A, indicating its desire to establish a connection. Host A, upon receiving the SYN packet, checks if the specified port (the destination port in the packet) is open and accepting connections. If the port does not exist or is not open, Host A will respond with a TCP RST packet. Then the connection is closed.

(b) How long does the process of ending the connection take?

A: About 0.000051 seconds.

Connecting to a non-existing host

(a) How often does the client try to open a connection? Note the time interval between attempts.

A: Client implement exponential backoff when retrying connections to non-existent hosts. This means that the time interval between retry attempts increases exponentially with each attempt.

Host A try to open a connection in 42.5499s, 43.6247, 49.7847, 58.2647, 74.9047, 107.5447 seconds.

(b) D client stop trying to connect at some point? If so, after how many attempts?

A: Clients using exponential backoff typically do not continue retrying indefinitely. They are designed to stop trying to connect after a certain number of attempts or when a maximum retry limit is reached.

Fragmentation in TCP

(a) How many packets did Host A measure and how many packets did Host B measure? Why?

A: Host A measures 54 (50 TCP + 4 ICMP) packets, Host B measures 43 (TCP) packets. The inconsistency is due to some of the packets sent initially by Host A are rejected by Router A since they are larger than MTU of Router A. Host B cannot measure these "large" packets since they are dropped before B. Also, the packets lost and retransmitted (if any) will be measured twice by Host A and only once by Host B. Hence, the measures of A and B are different.

(b) Is the DF flag set in the datagrams? Why?

A: Yes, DF is set to 1 which indicates the datagram cannot be fragmented. Usually in TCP, DF is set to 1 by the sender to prevent fragmentation during transmission, which will cause extra overhead of reassemble and certain latency in the receiver side. When transmission fails due to path MTU, the sender will adjust the size of the datagram sent via TCP according to ICMP messages.

(c) Do you observe fragmentation? If so, where does it occur?

A: Yes. The fragmentations are carried by Host A. Host A divides the datagrams into small ones which are smaller than MTU and transmit them.

(d) Study the ICMP messages recorded at Host A. Which node is the source? What is the type and the code of the messages?

A: The source is 10.0.1.2. Type 3 (Destination Unreachable). Code 4 (Fragmentation Needed)

TCP data transfer

Interactive application - fast link

Type a few characters slowly:

(a) Describe the payload of each packet.

A: For each character typed, 3 packets are observed. The 1st packet is from Host A to Host B and the payload is the character typed. The 2nd packet is the echo from Host B with the same payload character as the 1st packet. The 3rd packet is the Host A's ACK to the 2nd packet without any telnet payload.

(b) Explain why you do not see four packets per typed character.

A: Because of piggybacking technique, when Host B echoes to Host A, the ACK to the 1st packet is contained in the TCP header in the 2nd packet. In this way, Host B has no need to send two separate packets for respectively echo and ack.

(c) When the client receives the echo, it waits a certain time before sending the ACK. Why? How long is the delay?

A: Because of piggybacking technique, Host A waits a short time during which if there happened to be a packet carrying payload from A to B, the acknowledgement feedback can be carried by the way. The delay is about 0.1ms

(d) In the segments that carry characters, what window size is advertised by the telnet client and by the server? Does the window size vary as the connection proceeds?

A: The window size advertised by the client is 237. The window size advertised by the server is 362. These sizes do not vary as the connection proceeds.

Type quickly

(a) Do you observe a difference in the transmission of segment payloads and ACKs?

A: Differences are not observed w.r.t payloads but there is a difference in ACKs. Each packet still carries one single character as before. However, when typing quickly, both Host A and Host B apply piggybacking so that the separate ACK packet is only observed when typing ends.

Bulk transfer - fast link

(a) How often does the receiver send ACKs? Can you see a rule on how TCP sends ACKs?

A: On average, the receiver sends ACKs once per 0.26ms. The rule is TCP sends ACKs with the roughly same speed as segment sending speed.

(b) How many bytes of data does a receiver acknowledge in a typical ACK?

A: 1448 bytes.

(c) How does the window size vary during the session?

A: The send window size remain the same (29312) during the session. However, the rwnd size is small at first (5792) but increases gradually. After rwnd reaches 110160, it remains the same until the end.

(d) Select any ACK packet in the Wireshark trace and note its acknowledgement number. Find the original segment in the Wireshark output. How long did it take from the transmission until it was ACKed?

A: Select the ACK packed with Ack=2449, the corresponding original packet has Seq=1001. The time difference is $0.937400 - 0.936522 = 0.878\text{ms}$.

(e) Does the TCP sender generally transmit the maximum number of bytes as allowed by the receiver?

A: No. The maximum number of bytes allowed by the receiver equals to advertised rwnd size, which is 110160 when transmission is stable. But the number of transmitted bytes during an RTT is roughly 30000, significantly less than 110160.

Interactive application - slow link

(a) How many packets are transferred for each keystroke? Does the number change when you type faster?

A: 3 packets for each keystroke. The number changes when typing faster.

(b) Do you observe delayed acknowledgements?

A: Yes. And the delay is still about 0.1ms when typing slowly.

(c) Do you observe the effect of Nagle's algorithm? How many characters can you see in a segment?

A: Yes. When typing fast, the number of characters in a segment ranges from 4 to 9.

Bulk transfer - slow link

(a) Look at the pattern of segments and ACKs. Did the frequency of ACKs change compared to the bulk transfer on the fast link? How?

A: Yes. The average ACK frequency is about once every 1.23s.

(b) Are the window sizes advertised by the receiver different from those of the previous exercise?

A: Yes. The advertised rwnd size keep increasing during the session from 5792 to 60592, which is significantly smaller than that on fast link.

(c) Does the TCP sender generally transmit the maximum number of bytes as allowed by the receiver?

A: No. During an RTT about 8000 bytes are sent out, less than the advertised rwnd size.

TCP retransmission

(a) How many packets are transmitted at retransmission timeout?

A: 12+20 packets are transmitted. Retransmission take place during the disconnection of the Ethernet cable.

(b) Do the retransmissions end at some point?

A: Yes, Tcp retransmission will end at some point. If a sender sends a packet and does not receive an acknowledgment within a certain time frame, it may assume the packet was lost and retransmit it.

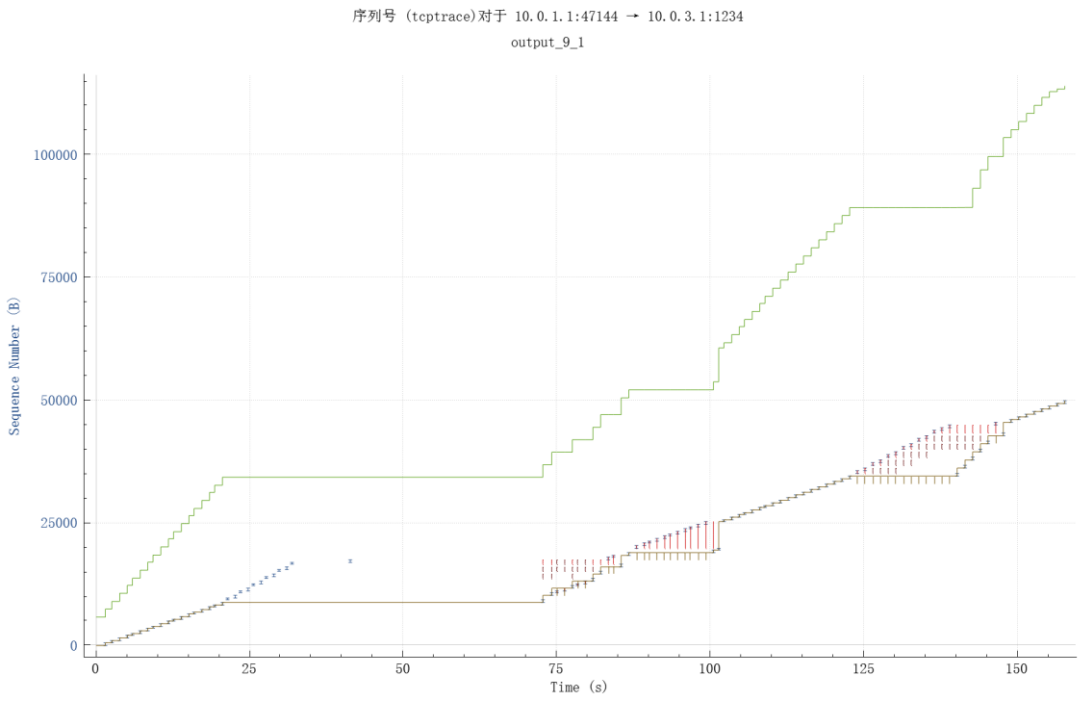


Figure 9_1

TCP congestion control

(a) Try to observe periods when TCP sender is in slow start phase and when the sender switches to congestion avoidance. Verify if the congestion window follows the rule of the slow-start phase.

A: During the slow start phase, the sender starts with a small congestion window (cwnd = 1 segment) and increases it exponentially with each ACK received. As the beginning of figure 10_1 show.

When it detects congestion, the sender transitions to the congestion avoidance phase, typically signaled by a triple duplicate ACK.

(b) Can you find occurrences of fast recovery?

A: We see in the figure that when the ack is repeated three times, the ssthress is set to half of the current cwnd, and the lost message is retransmitted, and the current cwnd = ssthress + 3 * MSS is set.

Since there are still multiple messages being transmitted in the network, the receiving end will continue to send ack messages. Every time the sending end receives an ack, it increases the cwnd by 1 and sends a new packet

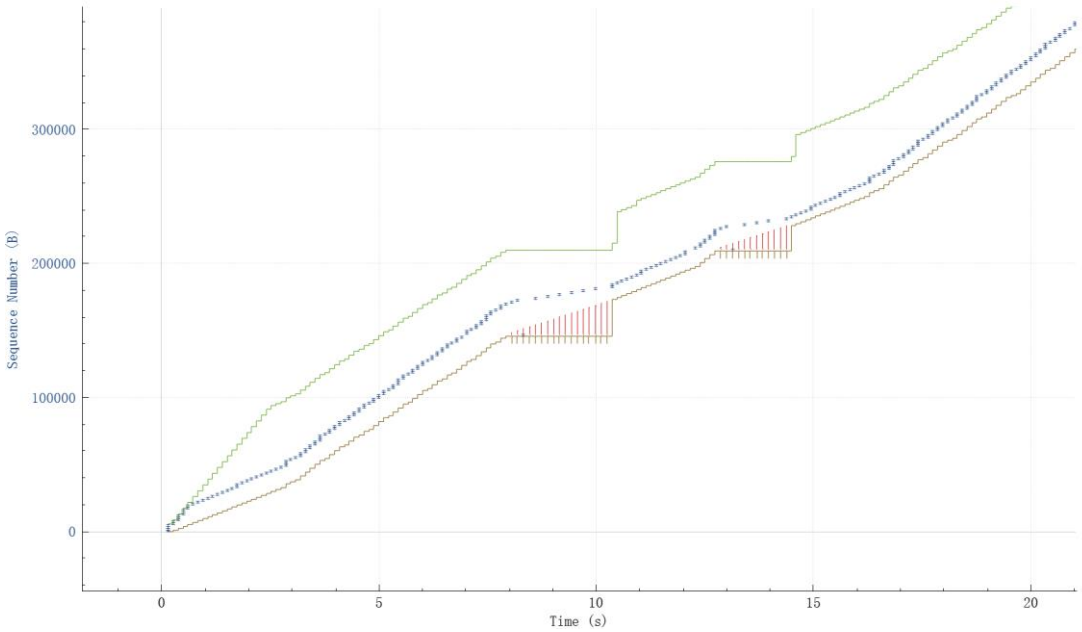


Figure 10_1

References

[1] B. A. Forouzan, *TCP/IP Protocol Suite*. Boston: McGraw-Hill Higher Education, 2010.

[2] X. Xie, *Computer Networking*. Beijing: Publishing House of Electronics Industry, 2017.

[3] R. Sharpe, E. Warnicke, and U. Lamping. Wireshark User's Guide. [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/. [Accessed: Sep. 24, 2023].