19/07/2019

Lecture 04 tutorial: Transport Layer

During tutorials prepare a short report of your activities and show it to your tutor.

Please be aware that the exam question might be directly related to the tutorial questions.

Introduction: How the TCP data transfer works

(Skip over the intro if you feel confident and go straight to PART 1)

The basic principle of the TCP transfer is that

- The sender sends segments **continuously** without waiting for acknowledgment frames from the receiver.
- The receiver acknowledges the received frames **cumulatively,** at its convenience.

Each segment has the sequence number in the TCP header. The **Seq#** is the number of the first byte in the segment.

The acknowledgment frame has the ACK number in the TCP header. The **Ack#** is the number of the last byte received plus 1, that is, the number of the first unacknowledged byte.

If the received frame is acknowledged immediately, then **Ack# = Seq# + Len**

If the sender is sending segments of the long object, e.g. a photo, the ACK frame consist of the TCP header only. The resulting Ethernet frame will have 14+20+20 = 54 bytes

Consider the following frames from the **trainMonash.pcap** Wireshark capture:

101	6.270548	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
102	6.270733	172.16.157.213	130.194.20.181	TCP	54	59705 → 80 [ACK] Seq=325 Ack=35041 Win=65700 Len=0
103	6.285954	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
104	6.285956	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
105	6.285958	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
106	6.285962	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
107	6.285963	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
108	6.285965	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
109	6.285966	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
110	6.285968	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
111	6.285969	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
112	6.285971	130.194.20.181	172.16.157.213	TCP	1514	[TCP segment of a reassembled PDU]
113	6.286227	172.16.157.213	130.194.20.181	TCP	54	59705 → 80 [ACK] Seq=325 Ack=49641 Win=65700 Len=0

Before we discuss details, we expand frames 102 and 103 to get some additional parameters:

Frame 102: 54 bytes
Internet Protocol Version 4, Src: 172.16.157.213, Dst: 130.194.20.181
Transmission Control Prot., Src Port: 59705, Dst Port: 80, Seq: 325, **Ack: 35041**, Len: 0

Frame 103: 1514 bytes
Internet Protocol Version 4, Src: 130.194.20.181, Dst: 172.16.157.213
Transmission Control Prot., Src Port: 80, Dst Port: 59705, **Seq: 35041**, Ack: 325, Len: 1460

In the above example, TCP segments are being sent from a host 130.194.20.181, port 80, to a host 172.16.157.213, port 59705. Each segment Len =1514-54=1460.

Without looking into details of each frame, we note that:

The Ack# in frame 102 was Ack = 35041. It will be a Seq# in the frame 103

After sending 10 frames each consisting a segment of 1460 bytes
the Ack# in frame 113 was Ack = 35041+14600 = 49641.
In addition, we can see that the Seq# in the frame 112 (last un-acknowledged segment) Seq#=48181. Adding the Len=1460, results in the Acq# in frame 113 equal to 46841.

Note that the ACK frames, 102 and 113, are both 54 bytes long (Len=0), that is, they do not carry and data.

In general an ACK frame can be also sent together with data, if exist (piggybacking).

Consider the following frames from the **trainMonash.pcap** Wireshark capture. Pay attention that the frames are from the same "TCP stream", meaning that the pair of the communication ports (the socket) is the same, (61981, 80) in the example

Frame 15: 360 bytes
Internet Protocol Version 4, Src: 172.16.8.1, Dst: 130.194.64.145
Trans. Control Protocol, Src Port: **61981**, Dst Port: 80, [PSH, ACK], Seq: 1, Ack: 1, Len: 306
Hypertext Transfer Protocol: GET /~app/tute/ http/1.1

Frame 18: 1276 bytes
Internet Protocol Version 4, Src: 130.194.64.145, Dst: 172.16.8.1
Trans. Control Protocol, Src Port: 80, Dst Port: **61981**, [PSH, ACK], Seq: 1, Ack: 307, Len: 1222
Hypertext Transfer Protocol: HTTP/1.1 200 OK

In Frame 15, the TCP segment carries an http request GET. The length of the payload, Len[15] = 306. The frame 18, carries the HTTP response consisting of Len[18] = 1222 bytes, and in addition, the Ack for the frame 15. Note that: Ack[18] = Seq[15] + Len[15] = 307.

PART 1: Wireshark example

INSTRUCTIONS: This part will be conducted as an I**ndividual Laboratory Exercise.** Please work through the task and note your observations in your individual report. Estimated time: **1 hour**.

Find the Wireshark file **trainSuzhou.pcap and open it.**

This file was recorded during transmission accessing a web page http://users.monash.edu/~app/tute/ The web page is on a Monash web server and the request was sent from myPC located in Suzhou

Select frame:

15 4.758506 172.16.8.1 130.194.64.145 HTTP 360 GET /~app/tute/ HTTP/1.1

We would like to see only frames related to this HTTP request. To do so, select from the pull-down menu:

Analyze → Follow TCP stream

You do not need to analyse the pop-up window "**Follow TCP Stream**", so after examining its contents you can close it.

Look at the main Wireshark window and observe that you have now **only TCP and HTTP** frames related to downloading the tutorial page "train". Note that there are **only two IP addresses and ports (sockets)** left:

172.16.8.1, port 61981 -- my PC
130.194.64.145, port 80 – the Monash tutorial web server

There are **five** main parts in the Wireshark frames:

1. Opening the connection: frames 11, 13, 14
2. Getting the top level web page: frames 15 and 18
3. Getting the photo of the train: frames 19 to 138
4. Getting the favicon.ico icon : frames 139 to 160
5. Closing the connection: frames 210, 211, 212, 218

- One comment regarding the length of the frame and related payload. The numbers visible in the Wireshark "Length" column refer to the length of the Ethernet frames. We will need the length of the payloads in each TCP segment, so we have to subtract the headers. We have

Lpy = LEth – Heth – Htcp –Hip = LEth – 54 (typically)

- Now, look at slides 23, 28—34, (**32**), to get the ideas about possible situations during TCP transmissions. The top view analysis of the TCP transmissions can be as follows:

  1. The main page requested in frame 15, with seq# = 1 and ack# = 1, is acknowledged in frame 18 with ack# = 1 + 306 = 307 and seq# = 1
  2. The train photo is requested in frame 19. The frame 19 give the ACK to frame 18 and we have seq# = 1+ 306 (frame 15) = 307, and ack# = 1+1222 (frame 18) = 1223.

  Now I would like you to **continue the above transmission story**. Make sure that you can explain every event in the recorded Wireshark file. In particular:

  - Missing segments, e.g. in the frame 33
  - Duplicated ACKs as in the frame
  - Retransmissions as in the frame 86
  - Out of order segments as in the frame 121.

In order to assist you with the task, you can go to the pull-down menu:

Statistics → Flow Graph

Select **TCP flow** and OK. This should produce the plot: "train.pcap – Graph Analysis"

You can save the plot as a text file, e.g. trainFlowGraph.txt.

The last year version of this **flow graph** edited for compactness is available from **Moodle**

The latest version of the Wireshark has more options, so select the one that is most important to you.

In tutorial 5, there are some more questions related to the flow graph. Typically, you can expect questions related to the graph in the class test and the exam.

Typical questions with relation to the flow graph (and underlying Wireshark frames if needed) can be as follows:

1. Why there are so many frames with ACK=643?
2. Why 643?
3. Why and where it is going to change?
4. What is special in frame 18 ?

PART 2: Questions.

INSTRUCTIONS: This part will be conducted as an **Interactive Group Tutorial Exercise**. Please form groups of 4 or 5 students (10 groups total) - Note: these same groups will likely be retained for the Network Security module *unless* swaps between pairs of students are agreed to. Each numbered group should then initially focus on a subset of questions below. (Group numbers will be assigned randomly.) After discussion and formulation of your answers you will then be asked to present your findings to the rest of the class! You may nominate one (or more) members of your group for this informal presentation. In addition, your team may facilitate other students in answering ALL QUESTIONS in their individual reports. Estimated time: **1 hour**.

GROUP 1

Question 1.
Consider a TCP connection between Host A and Host B. Suppose that the TCP segments traveling from Host A to Host B have source port number **x** and destination port number **y**. What are the source and destination port numbers for the segments traveling from Host B to Host A?

Question 2.
Suppose a process in Host C has a UDP socket with port number 6789. Suppose both Host_A and Host_B each send a UDP segment to Host_C with destination port number 6789. Will both of these segments be directed to the same socket at Host_C? If so, how will the process at Host_C know that these two segments originated from two different hosts?

GROUP 2

Question 3.
Why it is that voice and video traffic is often sent over TCP rather than UDP in today's Internet? (Hint: The answer we are looking for has nothing to do with TCP's congestion-control mechanism.)

Question 4.
Suppose that a Web server runs in Host C on port 80. Suppose this Web server uses persistent connections, and is currently receiving requests from two different Hosts, A and B. Are all of the requests being sent through the same socket at Host C? If they are being passed through different sockets, do both of the sockets have port 80? Discuss and explain

GROUP 3 (*)

Question 5.
True or false?

a. Host A is sending Host B a large file over a TCP connection. Assume Host B has no data to send Host A. Host B will not send acknowledgments to Host A because Host B cannot piggyback the acknowledgments on data.

b. The size of the TCP **rwnd** never changes throughout the duration of the connection.
c. Suppose Host A is sending Host B a large file over a TCP connection. The number of unacknowledged bytes that A sends cannot exceed the size of the receive buffer.
d. Suppose Host A is sending a large file to Host B over a TCP connection. If the sequence number for a segment of this connection is m, then the sequence number for the subsequent segment will necessarily be m + 1.
e. The TCP segment has a field in its header for **rwnd.**
f. Suppose that the last **SampleRTT** in a TCP connection is equal to 1 sec. The current value of **TimeoutInterval** for the connection will necessarily be > 1sec.
g. Suppose Host A sends one segment with sequence number 38 and 4 bytes of data over a TCP connection to Host B. In this same segment the acknowledgment number is necessarily 42.

GROUP 4

Question 6.
Suppose Host A sends two TCP segments back to back to Host B over a TCP connection. The first segment has sequence number 90; the second has sequence number 110.

a. How much data is in the first segment?
b. Suppose that the first segment is lost but the second segment arrives at B. In the acknowledgment that Host B sends to Host A, what will be the acknowledgment number?

Question 7.
True or false? Consider congestion control in TCP. When the timer expires at the sender, the value of **ssthresh** is set to one half of its previous value.

GROUP 5

Question 8.
Suppose two TCP connections are present over some bottleneck link of rate *R* bps. Both connections have a huge file to send (in the same direction over the bottleneck link). The transmissions of the files start at the same time. What transmission rate would TCP like to give to each of the connections?

Question 9.
Consider Figure 3.5. What are the source and destination port values in the segments flowing from the server back to the clients' processes? What are the IP addresses of the network-layer datagrams carrying the transport-layer segments?

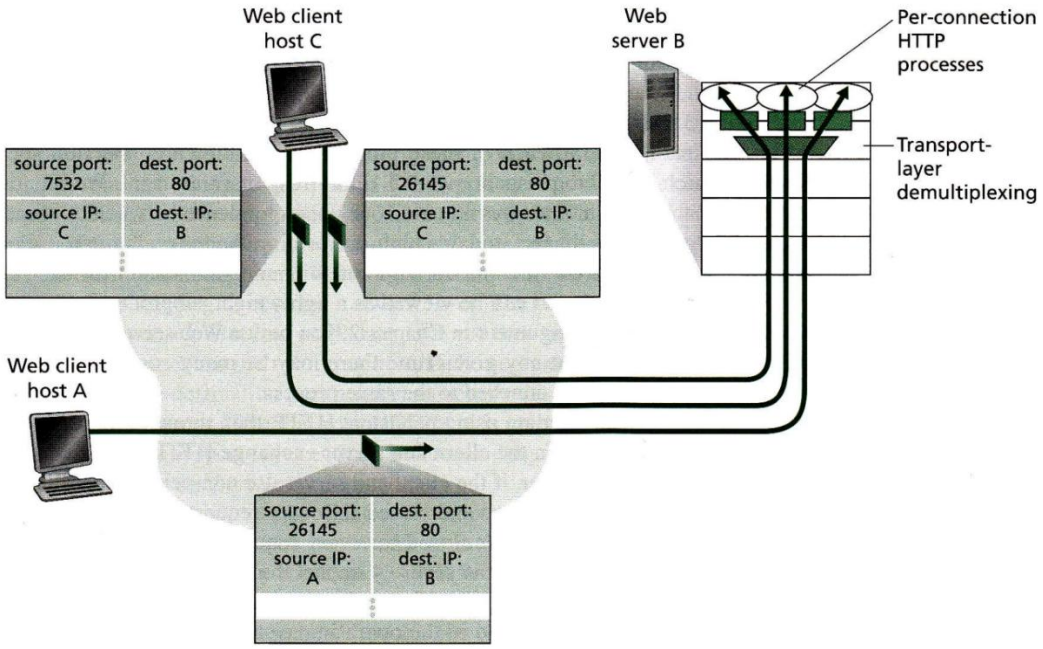


Figure 3.5 ♦ Two clients, using the same destination port number (80) to communicate with the same Web server application

GROUP 6

Question 10.
Consider transferring an enormous file of *L* bytes from Host A to Host B. Assume an MSS (maximum segment size) of 536 bytes.

a. What is the maximum value of ***L*** such that TCP sequence numbers are not exhausted? Recall that the TCP sequence number field has 4 bytes.

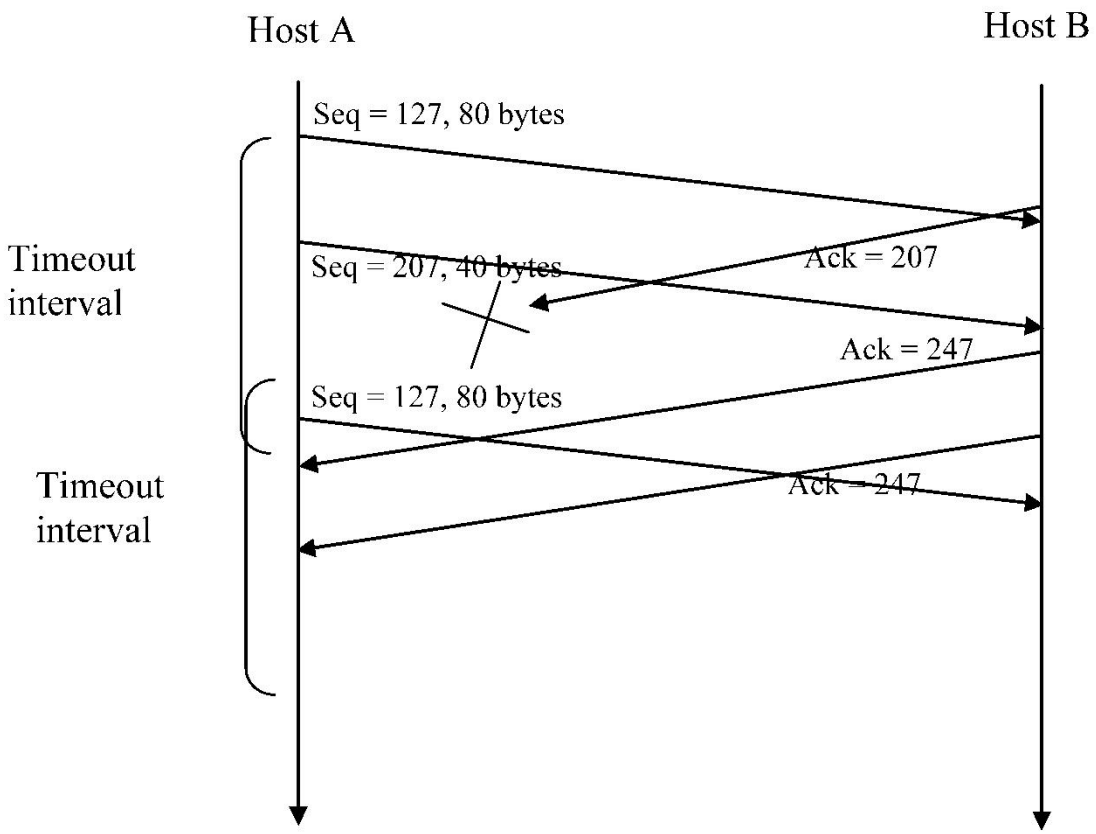
b. For the ***L*** you obtain in (a), find how long it takes to transmit the file. Assume that a total of 66 bytes of transport, network, and data-link header are added to each segment before the resulting packet is sent out over a 155 Mbps link. Ignore flow control and congestion control so A can pump out the segments back to back and continuously.

GROUP 7 (*)

Question 11.
Host A and B are communicating over a TCP connection, and Host B has already received from A all bytes up through byte 126. Suppose Host A then sends two segments to Host B back-to-back. The first and the second segments contain 80 and 40 bytes of data, respectively. In the first segment, the sequence number is 127, the source port number is 302, and the destination port number is 80. Host B sends an acknowledgment whenever it receives a segment from Host A.

a. In the second segment sent from Host A to B, what are the sequence number, source port number, and destination port number?

b. If the first segment arrives before the second segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number, the source port number, and the destination port number?
c. If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?
d. Suppose the two segments sent by A arrive in order at B. The first acknowledgment is lost and the second acknowledgment arrives after the first timeout interval. Draw a timing diagram, showing these segments and all other segments and acknowledgments sent. (Assume there is no additional packet loss.) For each segment in your figure, provide the sequence number and the number of bytes of data; for each acknowledgment that you add, provide the acknowledgment number.
e.



GROUP 8

Question 12.
Host A and B are directly connected with a 100 Mbps link. There is one TCP connection between the two hosts, and Host A is sending to Host B an enormous file over this connection. Host A can send its application data into its TCP socket at a rate as high as 120 Mbps but Host B can read out of its TCP receive buffer at a maximum rate of 50 Mbps. Describe the effect of TCP flow control.

Question 13.

In Slide 30, we discussed TCP's estimation of RTT. Why do you think TCP avoids measuring the **SampleRTT** for retransmitted segments?

GROUP 9 (*)

Question 14.
Suppose that the five measured **SampleRTT** values (see Slides 26, …) are 106ms, 120ms, 140ms, 90ms, and 115ms. Compute the **EstimatedRTT** after each of these **SampleMT** values is obtained, using a value of $\alpha = 0.125$ and assuming that the value of **EstimatedRTT** was 100ms just before the first of these five samples were obtained. Compute also the **DevRTT** after each sample is obtained, assuming a value of $\beta = 0.25$ and assuming the value of **DevRTT** was 5ms just before the first of these five samples was obtained. Last, compute the TCP **TimeoutInterval** after each of these

(use MATLAB, Excel, … and plot the results)

GROUP 10

Question 15.
In Slide 39, we saw that TCP waits until it has received three duplicate ACKs before performing a fast retransmit. Why do you think the TCP designers chose not to perform a fast retransmit after the first duplicate ACK for a segment is received?

(*) These questions have a comparatively high difficulty level