EL5373 Internet Architecture and Protocols

Homework 4 - Solutions

Question 1

An IP Datagram arrives at a Router, the relevant fields of whose IP Header are given below.

Header Length	5
Total Length	3300
ID	1
More Fragments	0
Fragment Offset	0
TTL	5

Assume that the MTU of the link on every interface of the router is 1100 byte, and that the Don't Fragment bit of the packet is not set, answer the following questions.

a. Corresponding to this datagram, how many datagrams are sent on the outgoing link?

b. For each of these datagrams forwarded on the outgoing link, list the values of the same fields that are specified for the incoming datagram.

Solution

a. Four datagrams will be sent corresponding to this incoming datagram.

b. Using the MTU to be 1100 byte (as given in the question), given below are the value of the fields:

First Fragment:

Header Length	5
Total Length	1100
ID	1
More Fragments	1
Fragment Offset	0
TTL	4

Second Fragment:

Header Length	5
Total Length	1100
ID	1
More Fragments	1
Fragment Offset	135
TTL	4

Third Fragment:

Header Length	5
Total Length	1100
ID	1
More Fragments	1
Fragment Offset	270
TTL	4

Fourth Fragment:

Header Length	5
Total Length	60
ID	1
More Fragments	0
Fragment Offset	405
TTL	4

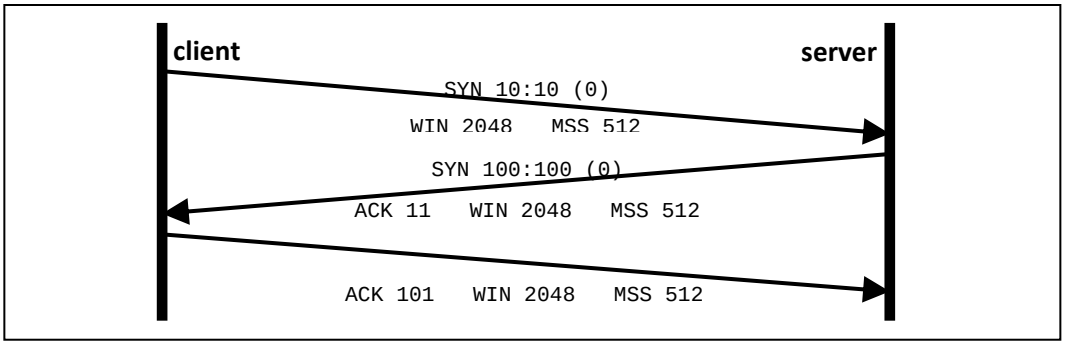
Question 2

A. Show the segment exchange involved in opening a TCP Connection assuming that the client performs an active open using an ISN of 10 and the ISN at the server is 100. Use an MSS of 512 byte and a receive window size of 2048 byte at each end. Use the notation used in the Text Book and indicate window size on messages.

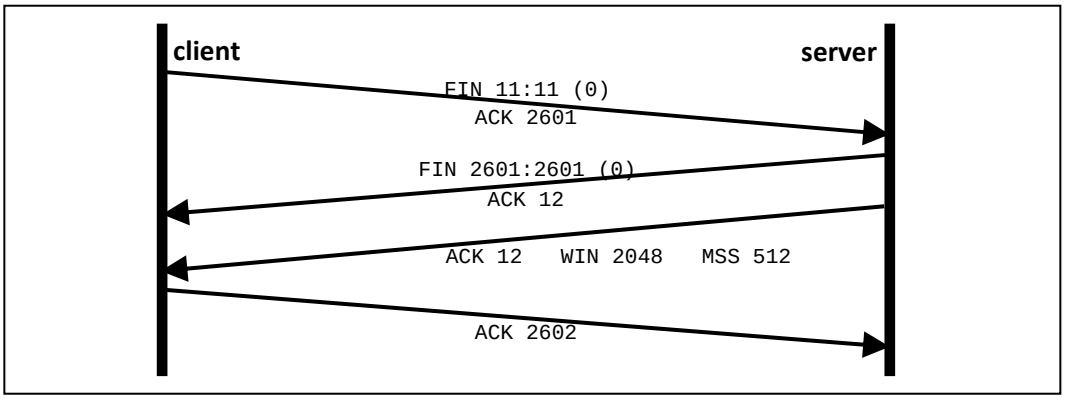
B. Assume that the server sends 2500 byte of data to the client, but the client does not send any data to the server. Show the message exchange to close the TCP Connection with the client-end performing the active close. Again, use the notation in the Text Book.

Solution

A. Segment exchange shown below:



B. Message exchange shown below:



Question 3

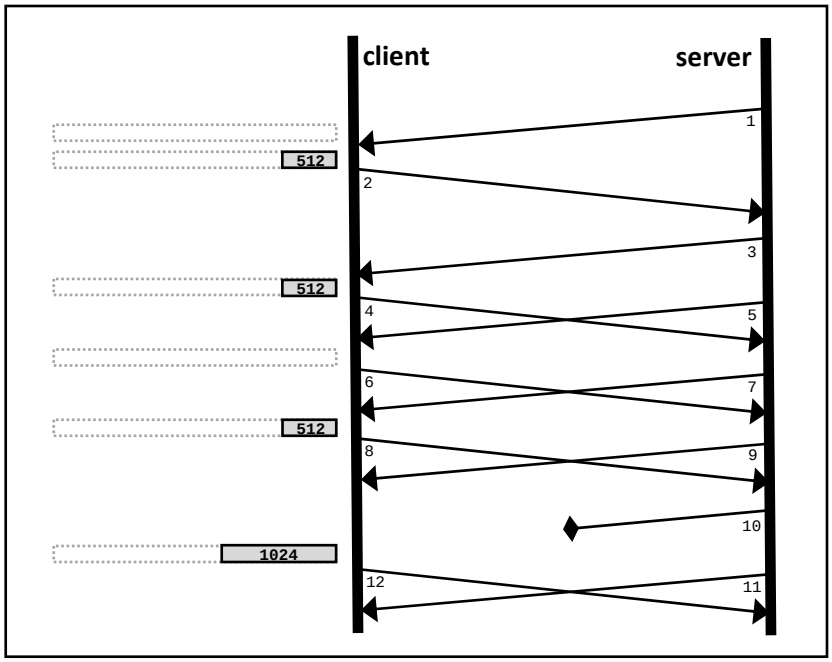
Consider a TCP Connection opened using an ISN of 100 at the client end and that of 50 at the server end. Use an MSS of 512 byte and a receive window size of 2560 byte at both ends.

A. The server sends a 3650 byte stream to the client. The client does not have any data to send to the server. For each of the segments shown in the figure on the next page, provide the sequence number, acknowledgement number and window size carried in the TCP Header. Transmission of segment 11 is not a retransmission but contains new data.

B. Assume *ssthresh* is 2048 byte at the start of the data exchange shown in the figure on the next page. Show *cwnd* at the server at different time instants in the table given on the next page.

C. Assume that the first RTT measurement is 1s and RTT increases by 1s for each subsequent RTT measurement. Given $\alpha = \frac{1}{4}$ and $\beta = \frac{1}{8}$, show the RTO values at the server at the start and when various segments are received.

D. If the retransmission timer times out at the server after it receives segment 12, and it resends the segment it sent earlier as segment 10, what are the new values of *cwnd* and *ssthresh*?



Time	*cwnd* at server (byte)	RTO at server (s)
When segment 1 is sent		
After segment 2 is received		
After segment 4 is received		
After segment 6 is received		
After segment 8 is received		
After segment 12 is received		

Solution

A. Note that the status of the receiving window at the client at different points of time is shown in the figure with the Question. This status depends on the rate at which the application layer pulls the received packets up. This information is used to compute the advertised Window Size in the segments originating from the client. The status of the receiving window at the server is not shown because it is already known that the server does not receive any data in the process, and thus, its receiving window is always empty. The values asked for the segments are given below:

Segment	Sequence Number	ACK Number	Window Size
1	51	101	2560
2	–	563	2048
3	563	101	2560
4	–	1075	2048
5	1075	101	2560
6	–	1587	2560
7	1587	101	2560
8	–	2099	2048
9	2099	101	2560
10	2611	101	2560
11	3123	101	2560
12	–	2611	1536

B. The value of *cwnd* keeps increasing by 512 byte as long as it is less than *ssthresh*. Since this happens only for the last packet, no new calculations are necessary. However the equation used for this calculation is given below.

$$cwnd = cwnd + \frac{segsize \times segsize}{cwnd} + \frac{segsize}{8}$$

Thus, the values are:

Time	***cwnd* at server (byte)**
When segment 1 is sent	512
After segment 2 is received	1024
After segment 4 is received	1536
After segment 6 is received	2048
After segment 8 is received	2560
After segment 12 is received	2726

C. Using the equations given in the slides, the following values are obtained. Keep in mind that since there are no statistics when we send out the first packet, the timeout duration is arbitrarily assumed to be 3.00 seconds.

Time	RTO at server (s)
When segment 1 is sent	3.00
After segment 2 is received	3.50
After segment 4 is received	4.53
After segment 8 is received	5.91

It is assumed above that the value of G is smaller than 2s.

D. The value of *cwnd* drops to **512 byte**. The value of *ssthresh* is half the minimum of the advertised window and *cwnd* before timeout (but at least as long as two segments). This is half of 1536 byte which is 768 byte. But since this is less than two segment size, the value of *ssthresh* is set to **1024 byte**.

Question 4

A TCP Application Client performs a small 10 byte data write followed by a small 15 byte data write and then waits for a reply from the server end. Assume 250×10^{-3}s of RTT. What is the maximum delay (time gap between start of transmission and receiving of acknowledgements) expected at the client end if the TCP layer sends this data

a. in two separate TCP segments, the first with a 10 byte payload and the second with a 15 byte payload?
b. in one TCP segment that has a 25 byte payload?

Solution

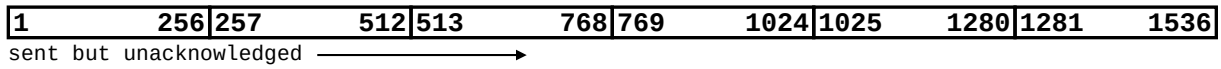
Depending on whether Delayed Acknowledgement is enabled at the server, it may delay sending back the ACK. This delay would be maximum when the Delayed Acknowledgement timer at the server just went off before the packet from the client arrived thus making the server wait for an entire timer cycle before being able to send out the ACK. However, all this need not be considered in this case as the RTT includes all possible delays resulting from TCP. In other words, RTT is the maximum time gap between the transmission of a packet and the reception of its ACK.

If the client has Nagle's algorithm disabled, then the two data writes must be sent in separate TCP segments. However, if the client has Nagle's Algorithm enabled, then it is possible that the two data chunks are combined into a single TCP segment while waiting in the buffer. These possibilities lead to the two cases considered in the question.

a. This case may occur either because Nagle's algorithm is disabled at the client, or because even though the client has Nagle's algorithm enabled, the second data chunk did not arrive in the buffer when it was time to send the first, and hence the two data chunks could not be combined into a single TCP segment. As a result, the client sends the second packet only after it receives the ACK for the first packet. Thus the maximum delay would be twice the RTT which is 500ms.
b. This case can occur only when Nagle's algorithm is enabled at the client and both chunks of data arrive in the buffer before the first could be sent. Thus, as both the payloads will be send in the same packet, the maximum delay will be only one RTT which is 250ms.

Question 5

Given a congestion window of 768 byte at the sender, if an ACK from the receiver to the sender carries the ACK Number 513 and a window size of 1024 byte, then which of the following segments can the sender send before it receives another ACK? The sender window before the ACK is received is given below.



Solution

It is clear from the sender window that the value of MSS is 256 byte. To answer this question, two different cases need to be considered.

Case I: The Sender is in Slow Start Phase
Since ACK513 is received, and the *cwnd* is 768 byte before the ACK is received, the new *cwnd* becomes 1024 byte, which is same as the advertised window. This means four packets each 256 byte long can be sent. Thus, all the packets can be sent before the sender received the next ACK.

Case II: The Sender is in Congestion Avoidance Phase
The new value of *cwnd* will be given by

$$cwnd = cwnd + \frac{mss \times mss}{cwnd} + \frac{mss}{8} = 768 + \frac{256^2}{768} + \frac{256}{8} = 885.3 \approx 885byte$$

Since this is smaller than the advertised window, only three more packets can be sent. Thus, the last packet cannot be sent before the next ACK is received.