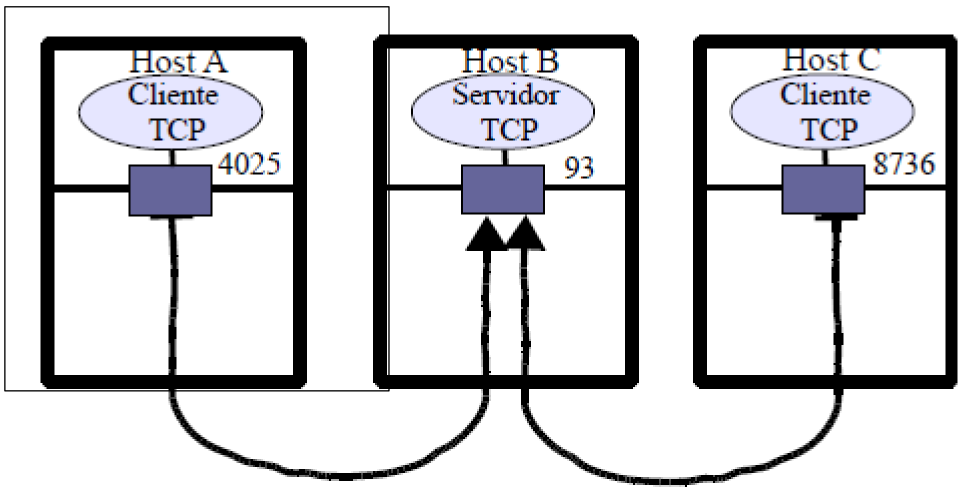
# Chapter 4 - Transport Layer

1. Suppose we have 3 hosts: A, B and C. A and C. Hosts A and C have two clients using ports 4025 and 8736 respectively. In the host B we have a TCP server that offers its service on port 93 (as shown in the figure).



The client A connects to the server B and then the client C connects to server B too.

a) How can the server B differentiate the two connections, if both are managed through a single port (93) ?.
b) If Client C gets as local port 4025, can server B differentiate the connections?
c) If a UDP server that offers service in the port 93 is added in host B, would it produce any problem? Justify your answer.
d) What is the difference(s) between concurrent servers versus iterative servers from the point of view of clients?

**Solution:**

a) From the point of view of the server application, each client is being served by a different socket, even though both sockets share the same port (93). From TCP's point of view, the connections are clearly differentiated since a TCP connection is identified by four parameters: source IP address, destination IP address, source port and destination port. In that case, there is no ambiguity, since client IP addresses and ports of the clients are different.
b) There is also no ambiguity since the client IP address is different.
c) The fact that a UDP service is offered in the same port does not cause any problem, since the packages of data are demultiplexed depending on the type of transport protocol at the IP level, before inspect the port numbers.

d) From the client's point of view, the response time of a concurrent server since the client requests the service until it establishes connection with the server is always fixed and bounded, regardless of how many clients are accessing the service. In non-concurrent (sequential) servers a client must wait for all clients who are queued before him to get the service from the server.

2. TCP flow control based on sliding window, has a window field in TCP header (available buffer at the other end) which limits the injection segments in the connection. The maximum size that can be indicated is 64 KB. This limitation, could affect the performance of TCP when high-speed networks are used (eg .: Gigabit Ethernet ~ 1Gbps) with RTTs of 2 ms.?

**<u>Solution:</u>**

At 1 Gbps (=$10^9$ bps) the transmission time of 64KB will be 64 $10^3$ * 8 /$10^9$ = 0.000512 seconds, that is, 0.512 ms. The sender will take this time to transmit its entire transmission window (64KB), while the first acknowledgment will arrive, as soon, 2 ms later. This means that after the transmission of the window, it will have to wait 2-0.512 = 1.488 ms before the first ACK arrives. Therefore, only 25% (≈0.512/2=0.256) of the time is being transmitted by the line.

Another way to look at it is as follows:

Optimal operation in a sliding window protocol is achieved when it is not necessary to wait for the reception of an acknowledgment, that is, after the transmission of the first segment of the window, the acknowledgment of that segment will arrive after a RTT (Round-Trip Time), during the transmission of the last segment of the window (from this window size the benefits won't improve, since we will have obtained 100% utilization).

In the proposed case, if the RTT is 2 ms, the size of the window should be

Vtx = 2 ms x 1 Gbps = 2 x 10-3 x 109 = 2 x 106 bits = 250KB

This means that before the first acknowledgment arrives, we would have time to transmit 250 KB. Since the maximum window in TCP is 64KB, we can confirm loss of benefits due to protocol limitations TCP.

3. Find the five errors of the next TCP header, knowing that it corresponds to the first segment of establishing a connection made by a standard HTTP client to the corresponding standard server. The options field is not empty. Justify your answer.

| | |
|---|---|
| TCP source port: | 120 |
| TCP destination port: | 80 |
| Sequence number: | 1400 |
| Acknowledgment number: | 0 |
| Header length: | 5 |
| Reserved: | 0 |
| Flags: | URG = 0 RST = 0 ACK = 1 PSH = 0 SYN = 1 END = 0 |

| | |
|---|---|
| Window size (in decimal) | 86535 |
| Checksum (in hexadecimal) | 9FB0 |
| Options (in hexadecimal bytes) | 02- 04 - 05 – B4 (MSS = 1460)<br>04 – 02 (Enable selective acknowledgment) |
| | |
| Padding Bytes: | Without padding bytes |

**Solution:**

**TCP Source port** is 120, lower than 1024. Ports below 1024 are reserved for servers. Thus it is an error.

**Header length** indicates the length of the TCP header expressed in 4-bytes words. A header not using the optional TCP field has a Header length of 5 (representing 20 bytes), however in the exercise TCP header there are optional TCP fields, so the header length will be 7.

However, the optional TCP fields only occupy 6 bytes, which is one 4-bytes word and a half. As Header length is expressed in 4-bytes words, 2-byte padding bytes should be added. Therefore, the **Padding Bytes field** should be 2.

If flag SYN is enabled, it means that this is the first segment sent from source to destination, so **ACK flag** can't be enabled.

**Window size** field takes 2-bytes of the header, so that window size value is limited to a maximum of 65,535 bytes. However, the window size field of the exercise TCP header is 86,535, higher than the maximum 65,535, thus, it is an error.

4. Suppose a process P (on host A1) wants to establish a TCP connection with process Q (on host E1):
   (A) Assuming that datagrams are not lost or there are not retransmissions, indicate the segments that would be exchanged both TCPs in the following situations:

   a) Q performs a passive aperture, and P an active aperture (assume that NSI (P) = 0, WIN (P) = 20, NSI (Q) = 13500, WIN (Q) = 30).
   b) Next P sends 100 octets to its TCP entity to send them to Q.
   c) Process P then tells your TCP to close the connection. Suppose that Q is also willing to close when it receives the P closure segment.

   (B) Same as section (A), but assuming that the Internet loses one segment of every 3 segments send from TCP of process P. The segments that the TCP of process Q sends do not lose. The retransmission timer (RTO) of P has a value equal to two RTT's. NOTE: NSI (P) represents the initial sequence number of process P, while WIN (P) indicates the initial window size that the TCP of the process P chooses. They are not using delayed acknowledgments, and the MSS = 100 bytes. Fill out the table below with your answer.

| Process | Sequence # | Flags | Acknowledgments # | Data<br>(indicating initial and final byte) |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|

**Solution:**

A.a)

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| P | 0 | SYN | ------------- | ---------- |
| Q | 13500 | SYN,ACK | 1 | ---------- |
| P | 1 | ACK | 13501 | ---------- |

A.b)

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| P | 1 | ACK | 13501 | 1..30 |
| Q | 13501 | ACK | 31 | ---------- |
| P | 31 | ACK | 13501 | 31..60 |
| Q | 13501 | ACK | 61 | ---------- |
| P | 61 | ACK | 13501 | 61..90 |
| Q | 13501 | ACK | 91 | ---------- |
| P | 91 | ACK | 13501 | 91..100 |
| Q | 13501 | ACK | 101 | ---------- |

A.c)

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| P | 101 | FIN, ACK | 13501 | ---------- |
| Q | 13501 | FIN,ACK | 102 | ---------- |
| P | 102 | ACK | 13502 | ---------- |

B.a)

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| P | 0 | SYN | ------------- | ---------- |
| Q | 13500 | SYN,ACK | 1 | ---------- |
| P | 1 | ACK | 13501 | ---------- |

B.b)

| Process | Sequence # | Flags | Acknowledgments # | Data<br>(indicating initial and final byte) |
|---|---|---|---|---|
| P | 1 | ACK | 13501 | 1..30 |
| This segment is lost, after RTO ms a timer will timeout | | | | |
| P | 1 | ACK | 13501 | 1..30 |
| Q | 13501 | ACK | 31 | ---------- |
| P | 31 | ACK | 13501 | 31..60 |
| Q | 13501 | ACK | 61 | ---------- |
| P | 61 | ACK | 13501 | 61..90 |
| This segment is lost, after RTO ms a timer will timeout | | | | |
| P | 61 | ACK | 13501 | 61..90 |
| Q | 13501 | ACK | 91 | ---------- |
| P | 91 | ACK | 13501 | 91..100 |
| Q | 13501 | ACK | 101 | ---------- |

B.c)

| Process | Sequence # | Flags | Acknowledgments # | Data<br>(indicating initial and final byte) |
|---|---|---|---|---|
| P | 101 | FIN,<br>ACK | 13501 | ---------- |
| This segment is lost, after RTO ms a timer will timeout | | | | |
| P | 101 | FIN,<br>ACK | 13501 | ---------- |

| | | | | |
|---|---|---|---|---|
| Q | 13501 | FIN,ACK | 102 | ---------- |
| P | 102 | ACK | 13502 | ---------- |

5. A sender has sent segments 1 to 50. Each of them has 512 bytes of data. The sender receives an ACK with value 15873 (31x512 = 15872), and then 3 duplicate ACKs with value 15873.
    a) Based on this information, what segment (s) can the sender assume that have been lost? And which ones can you consider to have been received correctly?
    b) The same sender retransmits the segment allegedly lost and receives an ACK with value 18433 (36x512 = 18432). Can the sender assume that any of the 50 segments sent initially has been lost? If so, which one(s)? Which segments can you consider correctly received?

**Solution:**

a) The first ACK received with value 15873 indicates that the first 31 segments have arrived well. The three duplicate ACKs with the value 15873 indicate that the next segment has not arrived in order, so we can assume that segment 32 (with bytes 15873 to 16384) has been lost. Regarding the segments after 32 we can't assume anything, except that three of them have arrived, but we do not know which ones.

b) After receiving this new ACK, we know that segment 32 was lost and that 33, 34, 35 and 36 arrived correctly, since acknowledgement has been received for them. For segment 37 we have not received any acknowledgement, so we can assume that it has been lost. For segments after 37 we do not have any information.

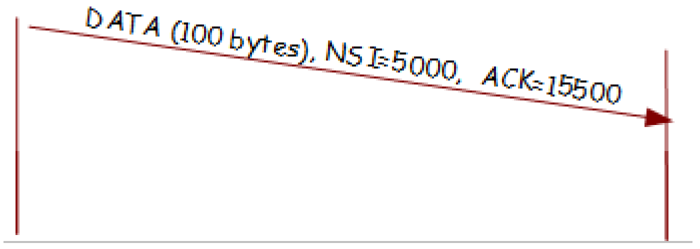
6. A user wants to connect to the web server on a computer that does not actually have any web server running, so the connection can't be established. We will call "computer A" to the host on which the user is running the browser, and "computer B" to the host on which the web server is supposed to be running. Fill out the table below, specifying the sequence of segments that will take place between the two computers in the previous assumption.

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| A | 51 | SYN | | |
| B | | | | |

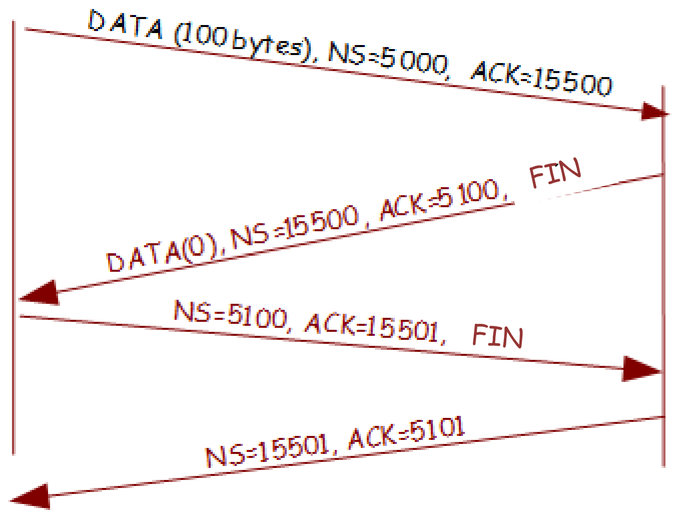
**Solution:**

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| A | 51 | SYN | | |
| B | **xxxxxxx** | **RST ACK** | **52** | |

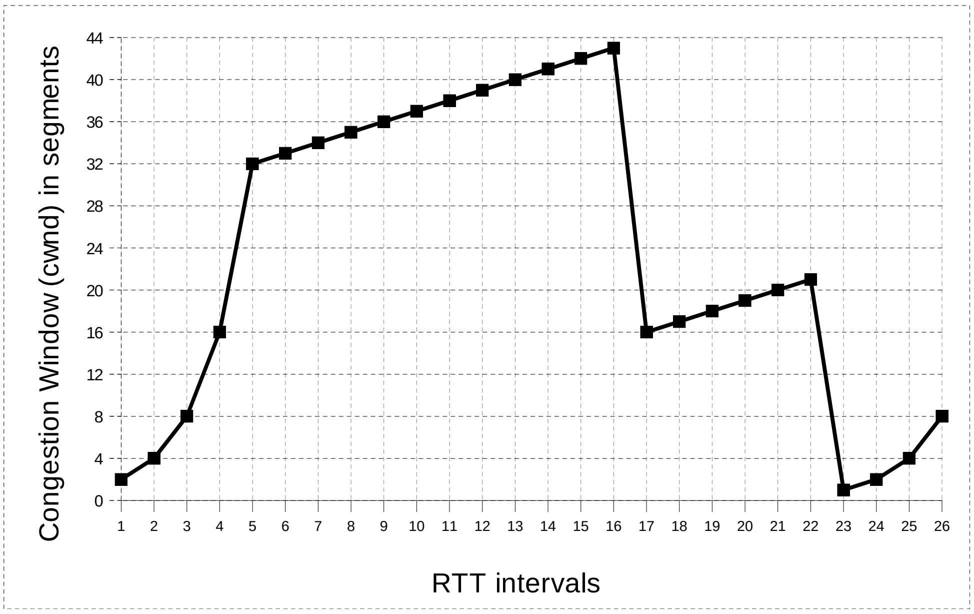
7. Assuming no errors occur and neither end sends more data, complete the following sequence of TCP segments until the connection is closed at both ends. Indicates for each segment: activated flags, ACK field value (only if the ACK flag is activated) and segment number of the segment.



**Solution:**


DATA (100 bytes), NS=5000, ACK=15500
DATA(0), NS=15500, ACK=5100, FIN
NS=5100, ACK=15501, FIN
NS=15501, ACK=5101

8. Consider the following representation of the size of the congestion window as a function of time. Based on the TCP congestion control, answer the following questions: (When the window size is asked, indicate it in segments).



a) Identify the intervals in which the slow-start mechanism acts.

b) Identify the intervals in which the congestion avoidance mechanism acts.
c) After RTT 16, how is segment loss detected? And after the RTT 22?
d) What is the value of the threshold (slow-start threshold) in RTT 2?
e) What is the value of the threshold (slow-start threshold) in the RTT 18?
f) What is the slow-start threshold in RTT 24?
g) During which RTT segment 70 is sent?
h) Assuming that after the RTT 26 the loss of a packet is detected by the reception of three duplicate ACKs, what will be the congestion window and threshold values?

**Solution:**

a) From RTT 1 to 5 and from 23 to 25.
b) From RTT 5 to 16, and from 17 to 22.
c) The RTT 16 detects the reception of three duplicate ACKs. A timer timeout on the RTT 22.
d) The value in RTT 2 is 32 segments.
e) The value in RTT 18 is 16 segments.
f) The value in RTT 24 is 10.5 segments.
g) In RTT 5 we have sent 2 + 4 + 8 + 16 + 32 = 62 segments, then the 70 will be sent in the RTT 6
h) In the RTT 26, 8 segments are sent. The transmission window, therefore, will have size 8. When the three duplicate acknowledgments are received, the threshold an the congestion windows will change following the formulas seen in class: Threshold = Transmission Window / 2 = 8/2 = 4; Congestion Window = Threshold = 4.

9. The process **pA** in computer A and the process **pB** in computer B want to establish a TCP connection. Throughout the connection, pA and pB will declare in their segments that the window sizes will remain constant with the following values: WIN (pA) = WIN (pB) = 300 bytes. Assuming no packets are lost or retransmissions occur, fill out the table below with the segments exchanged between pA and pB in the following situations:

   a) Establishment of connection between pA and pB (pA performs an active aperture and pB a passive aperture). Assume that during this process the initial sequence numbers exchanged are: NSI (pA) = 1000 and NSI (pB) = 5000.

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| | | | | |

   b) Then, pA sends 700 bytes to pB. In this section we will assume that during the connection establishment phase, both processes have agreed to exchange segments of maximum size 100 bytes (MSS = 100 bytes). Whenever possible pA sends segments of the MSS size. A policy of delayed acknowledgments is applied. The data transfer must follow the slow-start protocol. There is no loss or change of order of any segment.

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| | | | | |

**Solution:**

a)

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| $p_A$ | 1000 | SYN | ------------- | ---------- |
| $p_B$ | 5000 | SYN,ACK | 1001 | ---------- |
| $p_A$ | 1001 | ACK | 5001 | ---------- |

b)

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| $p_A$ | 1001 | ACK | 5001 | 1001..1100 |
| $p_A$ | 1101 | ACK | 5001 | 1101..1200 |
| $p_B$ | 5001 | ACK | 1201 | ---------- |
| $p_A$ | 1201 | ACK | 5001 | 1201..1300 |
| $p_A$ | 1301 | ACK | 5001 | 1301..1400 |
| $p_A$ | 1401 | ACK | 5001 | 1401..1500 |
| $p_B$ | 5001 | ACK | 1401 | ---------- |
| $p_A$ | 1501 | ACK | 5001 | 1501..1600 |
| $p_A$ | 1601 | ACK | 5001 | 1601..1700 |
| $p_B$ | 5001 | ACK | 1501 | ---------- |
| $p_B$ | 5001 | ACK | 1701 | ---------- |

10. A client and a server communicate over the TCP protocol. The client application sends a 30-byte request to the server. The server response is a 1476-byte message, after which the connection will be closed. We know that the MSS that use the two ends is 512 bytes, ISN (C) = 7,000, ISN (S) = 15,000 (ISN is the initial sequence number), WIN (C) = WIN (S)=2048, For this exercise, the initial size of the congestion window is two segments (2 * MSS bytes). Both extremes use delayed acknowledgments. Describe the evolution of the TCP connection, from the establishment to the closing of the connection. Fill in the table below with your answer.

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| | | | | |

**Solution:**

| Process | Sequence # | Flags | Acknowledgments # | Data (indicating initial and final byte) |
|---|---|---|---|---|
| C | 7000 | SYN | --- | --- |
| S | 15000 | SYN, ACK | 7001 | --- |
| C | 7001 | ACK | 15001 | --- |
| C | 7001 | ACK | 15001 | 1..30 (7001..7030) |
| S | 15001 | ACK | 7031 | 1..512 (15001..15512) |
| S | 15513 | ACK | 7031 | 513..1024 (15513..16024) |
| C | 7031 | ACK | 16025 | --- |
| S | 16025 | ACK | 7031 | 1025..1476 (16025..16476) |
| C | 7031 | ACK | 16477 | --- |
| S | 16477 | FIN, ACK | 7031 | --- |
| C | 7031 | FIN, ACK | 16478 | --- |
| S | 16478 | ACK | 7032 | --- |