HY-335a Computer Networking

TCP Theory

TA: Paul Zikas
zikas@csd.uoc.gr

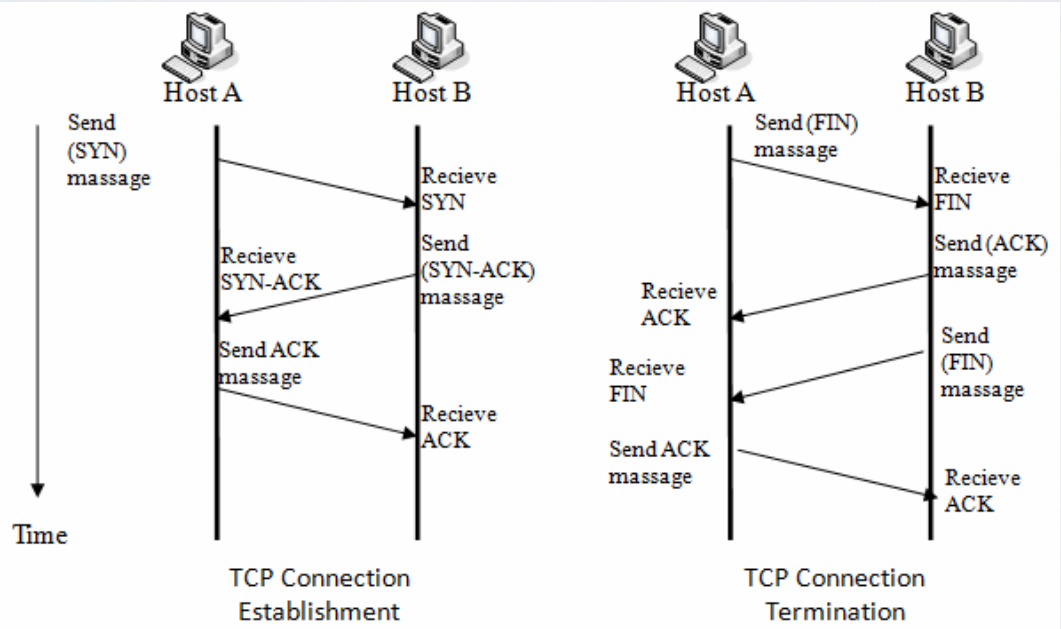


Figure 2.1. TCP session establishment and termination

Overview

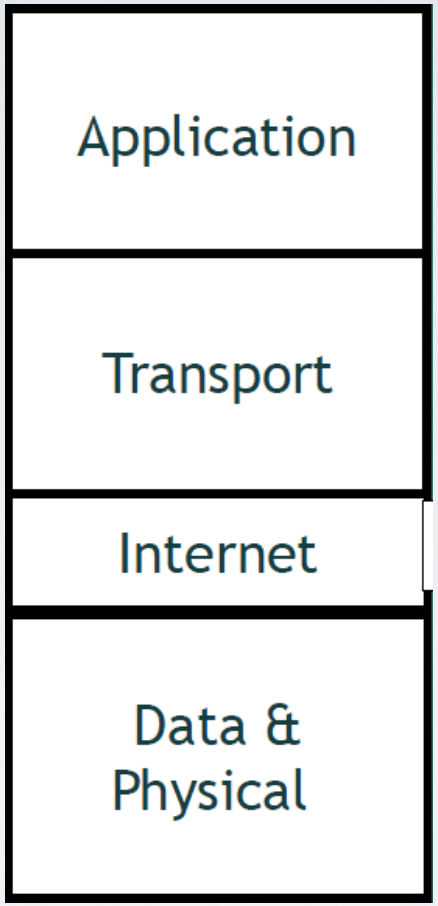
1. Layering the internet
2. Transport Layer
3. TCP theory and usage
4. Assignments
5. Conclusion

Layering the internet

HTTP. FTP, SMTP, BitTorrent

TCP UDP

Ethernet, WiFi



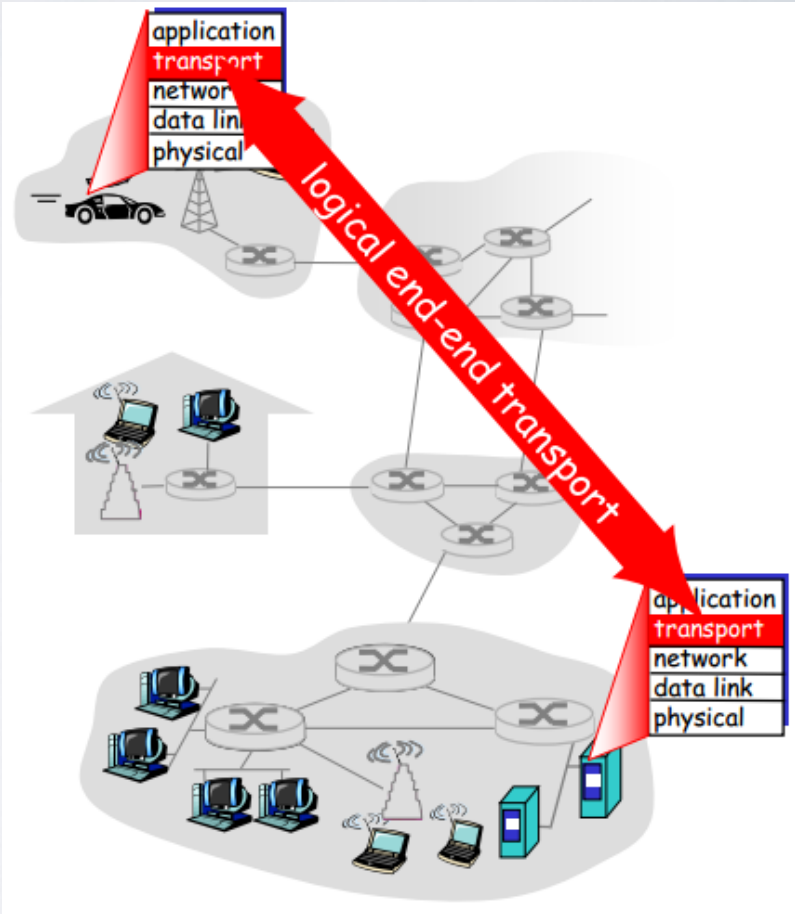
Reliability, integrity, packet ordering

Forwarding and routing

Transport Layer

The transport layer is a conceptual division of methods in the layered architecture of protocols in the network stack

>The protocols of this layer provide host-to-host communication services for applications



Services:

- connection-oriented communication
- Reliability
- Flow control
- Multiplexing

TCP

Advantages:

- TCP always guarantees three things 1) your data reaches its destination, 2) it reaches there in time and 3) it reaches there without duplication.
- It automatically breaks up data into packets for you.

Disadvantages

- Can not be used for broadcast or multicast transmission (or not?)
- It is slower in functioning than UDP

UDP (for comparison)

Advantages:

- Broadcast and multicast connections are available with UDP which is not the case with TCP.
- It does not restrict you to connection based communication model (handshake)
- Much faster than TCP

Disadvantages

- There are no guarantees with UDP. It is possible that a packet may not be delivered, or delivered twice, or delivered not in time.
- You have to manually break the data into packets

TCP – The basics

TCP socket has 4 fields

1. Source IP address
2. Source port number
3. Destination IP address
4. Destination port number

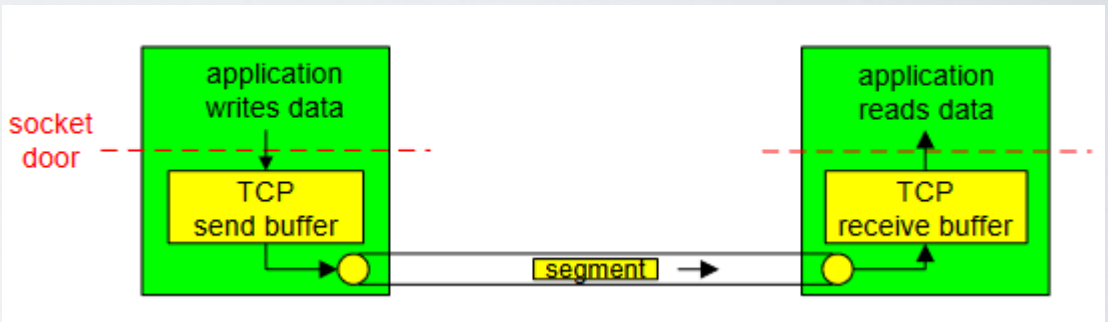
Socket: Endpoint of a two-way communication link between two programs running on the net.

The recipient uses all 4 fields to forward the segment to the correct socket.

- A web server can support multiple TCP sockets at the same time
- Web servers have a separate socket for each client

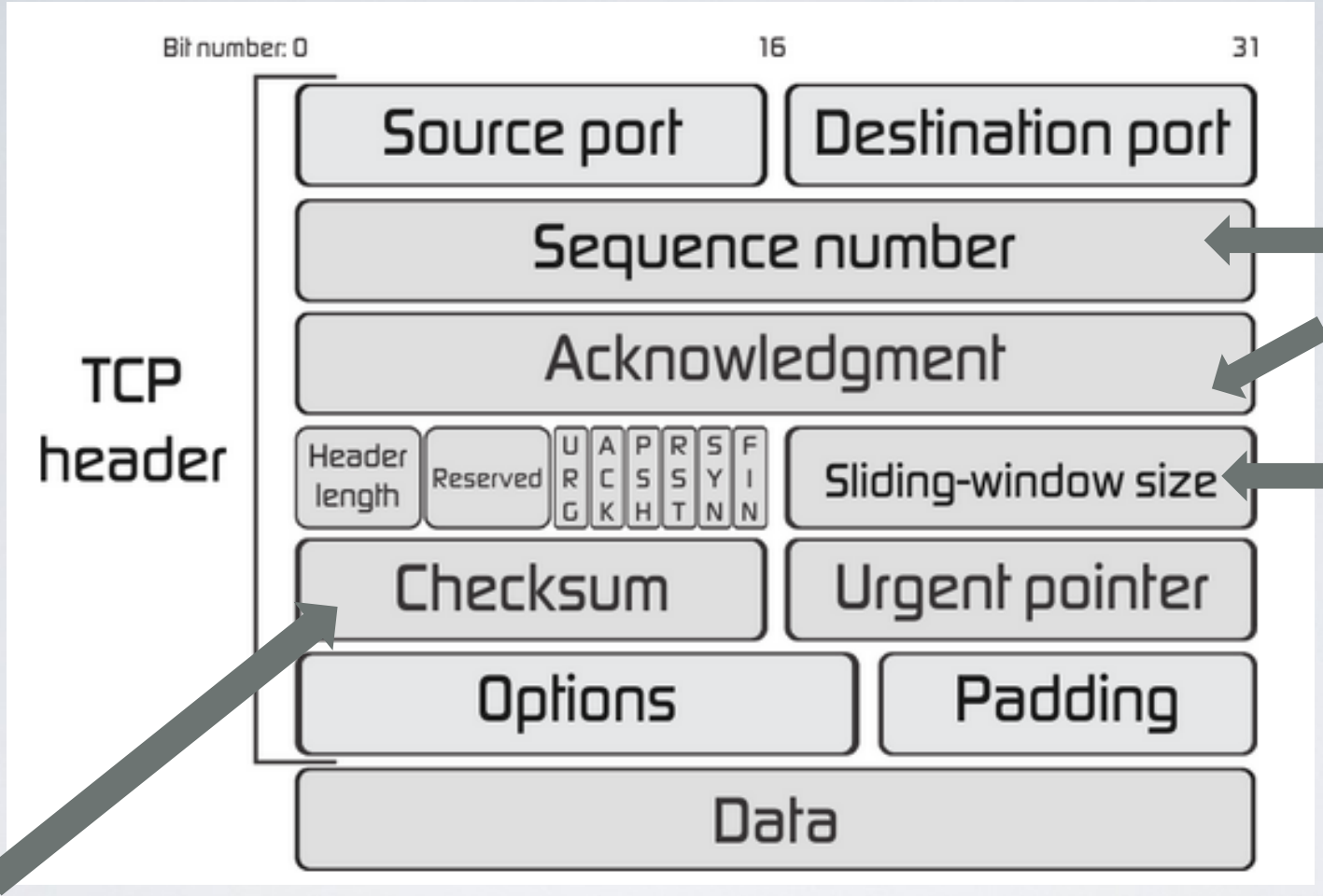
TCP – The basics

- End-to-end support
  (one sender one recipient – no multicasting)
- Packets arrived **in order**
- **Pipelined:** Send successive requests, over the same persistent connection, without waiting for the answer
  + avoids latency of the connection
- Send and receive buffers
- Sends data to both directions
- **Handshake**
- Congestion avoidance
  + avoids overloading the recipient with packets



TCP segment

32 bits



Counting based on the bytes not from the segments

Number of bytes to recieve

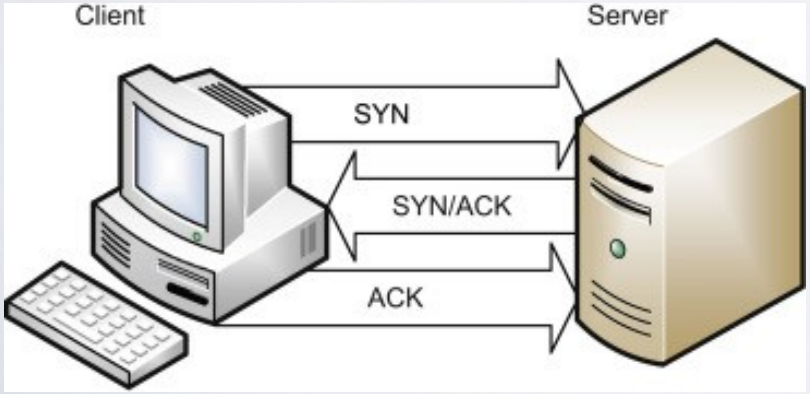
Same as UDP

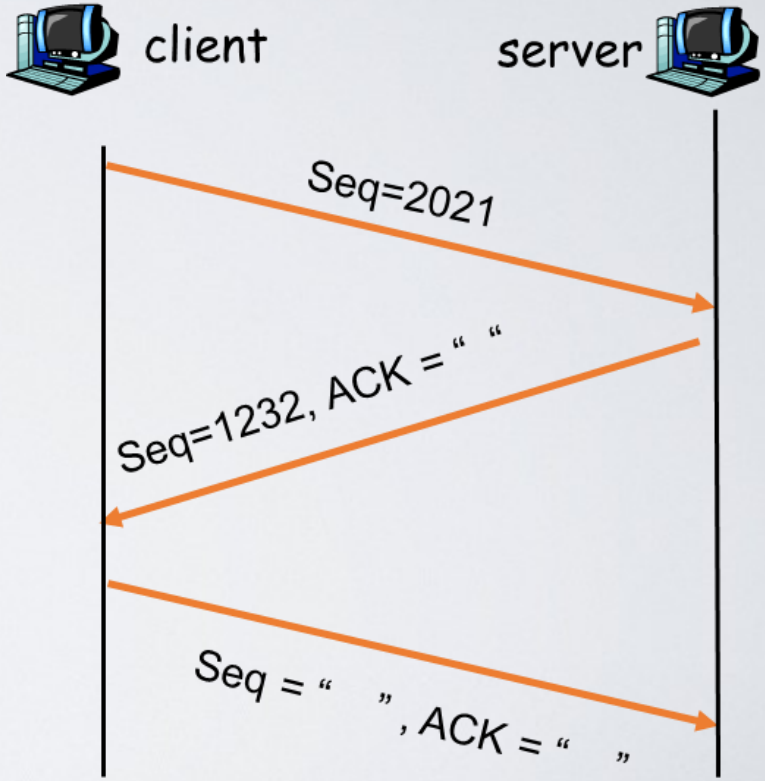
Three way handshake

Step 1: Client host sends a TCP SYN segment to server
- Identifies the initial sequence number
- No data

Step 2: Server receives the SYN and sends back to host SYN ACK
- Identifies the initial sequence number

Step 3: Client recienves the SYN ACK and sends the ACK to server along with the first data



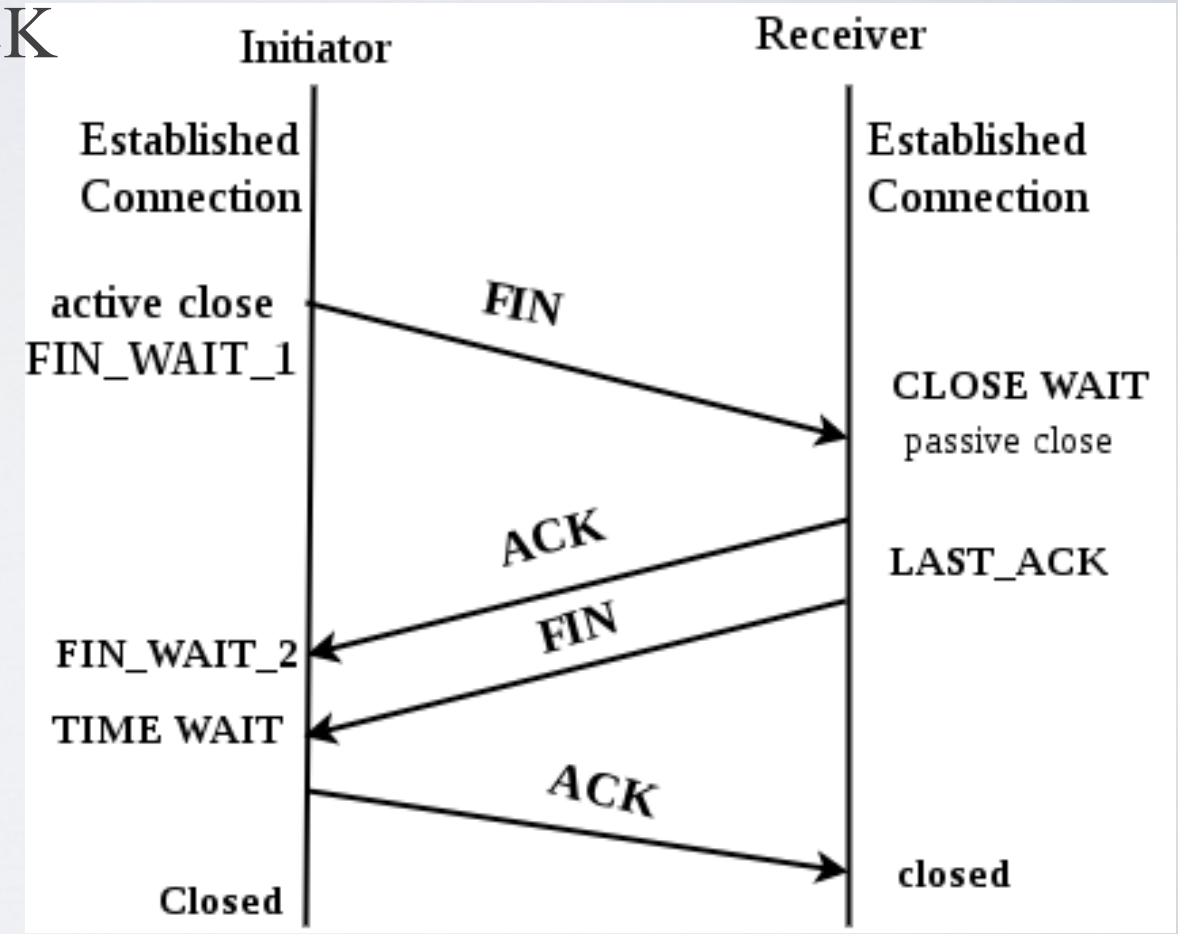


Socket closed

Step 1: Client Hosts sends TCP FIN to server
Step 2: Server sends to Host an ACK
Step 3: along with a FIN
Step 4: Host sends another ACK

"I always ACK before closing the socket"

```
clientSocket.close();
```

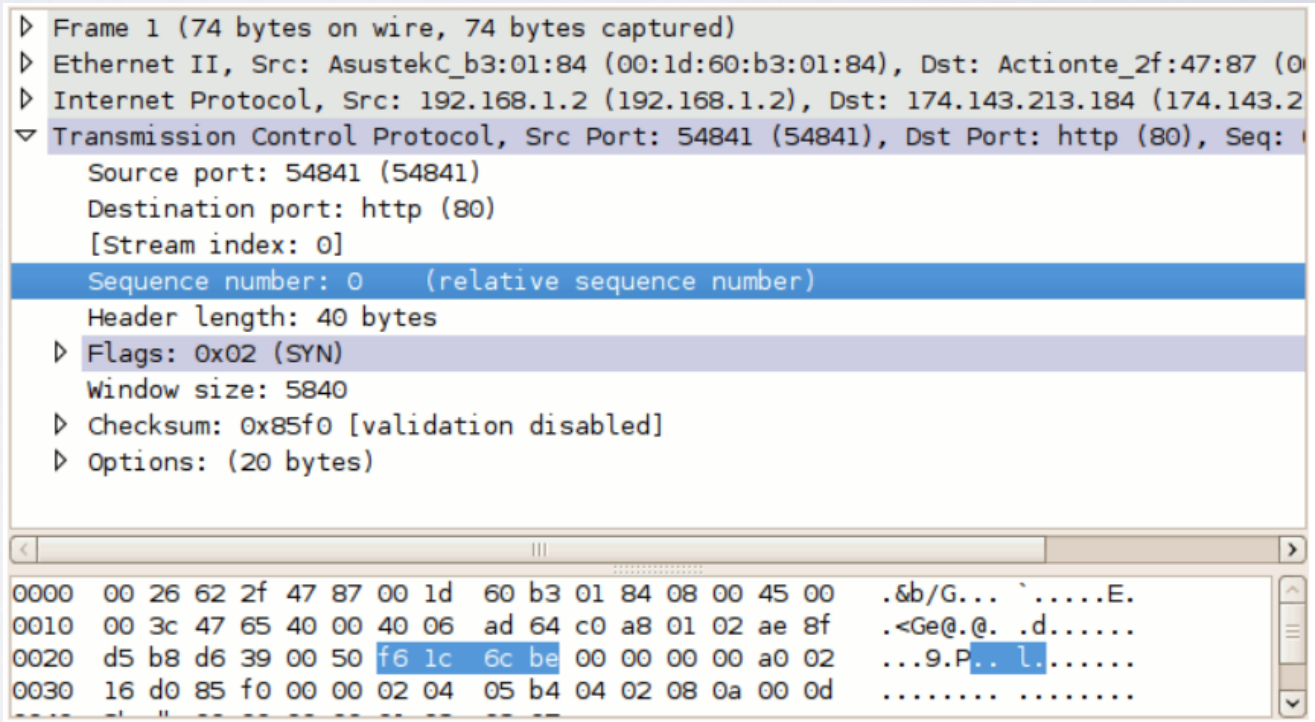


Sequence number

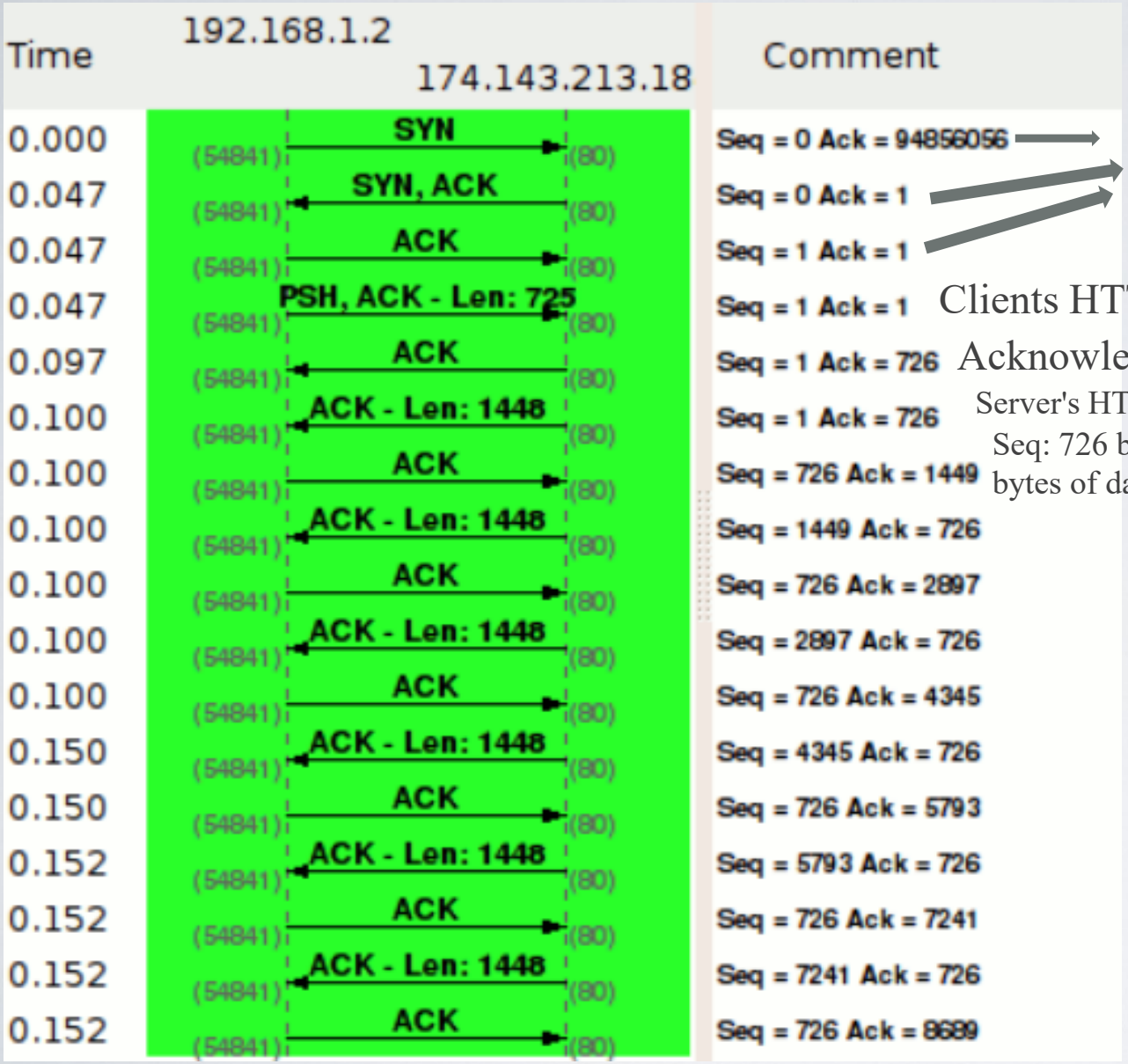
The client on **either side** of a TCP session maintains a 32-bit sequence number it uses to keep track of how much data it has sent.
(Seq and ACK both keep the packets in correct line)

* When a host initiates a TCP session, its initial sequence number is effectively random; it may be any value between 0 and 4,294,967,295.

The acknowledgement number to inform the sending host that the transmitted data was received successfully



Sequence numbers and ACKs example



Three way handshake

Clients HTTP request (Ack still 1 cause no data)

Acknowledge the data sent by the client in packet 4

Server's HTTP response (Ack still 1 cause no data arrived yet)

Seq: 726 because of the last packet, ACK: Having received 1448 bytes of data from the server, the client increases its ack to 1449.

TCP Congestion Control

TCP uses a **congestion window** and a **congestion policy** that avoid congestion
If the network is busy it cannot deliver the data on time , it must tell the sender to slow down

Congestion policy in TCP:

1. **Slow Start Phase**: starts slowly increment is exponential to threshold
2. **Congestion Avoidance Phase**: After reaching the threshold increment is by 1
3. **Congestion Detection Phase**: Sender goes back to Slow start phase or Congestion avoidance phase.

Window size: How much data (in bytes) the receiving device is willing to receive at any point in time

TCP Congestion Control

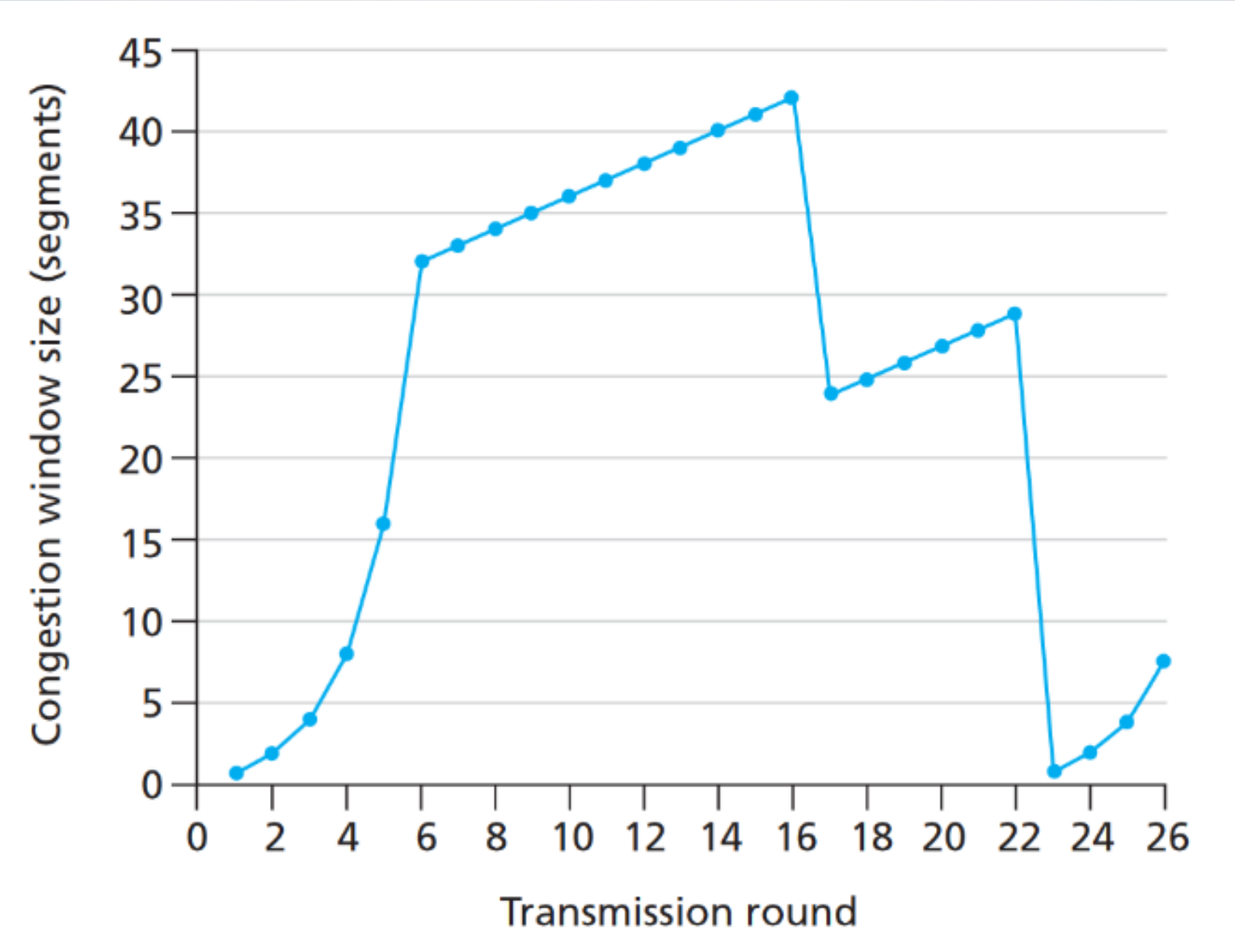
Slow Start Phase : exponential increment – In this phase after every RTT the congestion window size increments exponentially.

Congestion Avoidance Phase : additive increment – This phase starts after the threshold value also denoted as ssthresh. The size of cwnd(congestion window) increases additive. After each RTT cwnd = cwnd + 1

If congestion occurs, the congestion window size is decreased.
Retransmission can occur in one of two cases: when the RTO timer times out or when three duplicate ACKs are received.

Case 1 : Retransmission due to Timeout – In this case congestion possibility is high.
(a) ssthresh is reduced to half of the current window size.
(b) set cwnd = 1
(c) start with slow start phase again.
Case 2 : Retransmission due to 3 Acknowledgement Duplicates – In this case congestion possibility is less.
(a) ssthresh value reduces to half of the current window size.
(b) set cwnd= ssthresh
(c) start with congestion avoidance phase

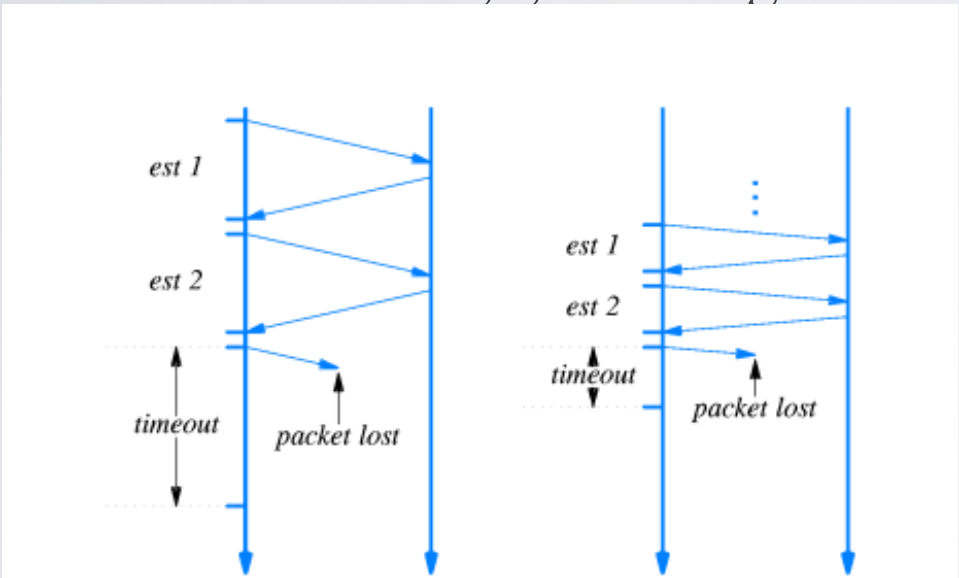
Example (part of 2nd assignment)

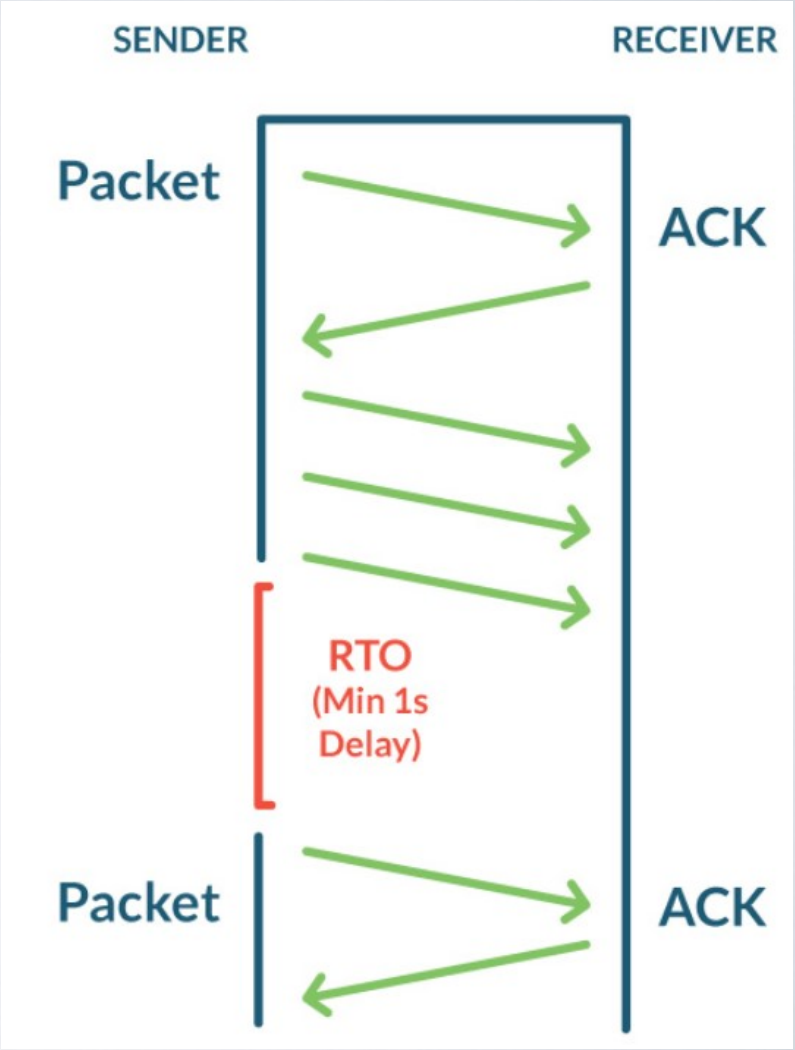


RTT and Timeout

RTT: the amount of time it takes for a signal to be sent plus the amount of time it takes for an acknowledgement of that signal to be received

RTO: When the sender is missing too many acknowledgments and decides to take a time out. After that sends 1,2,3 messages and so on

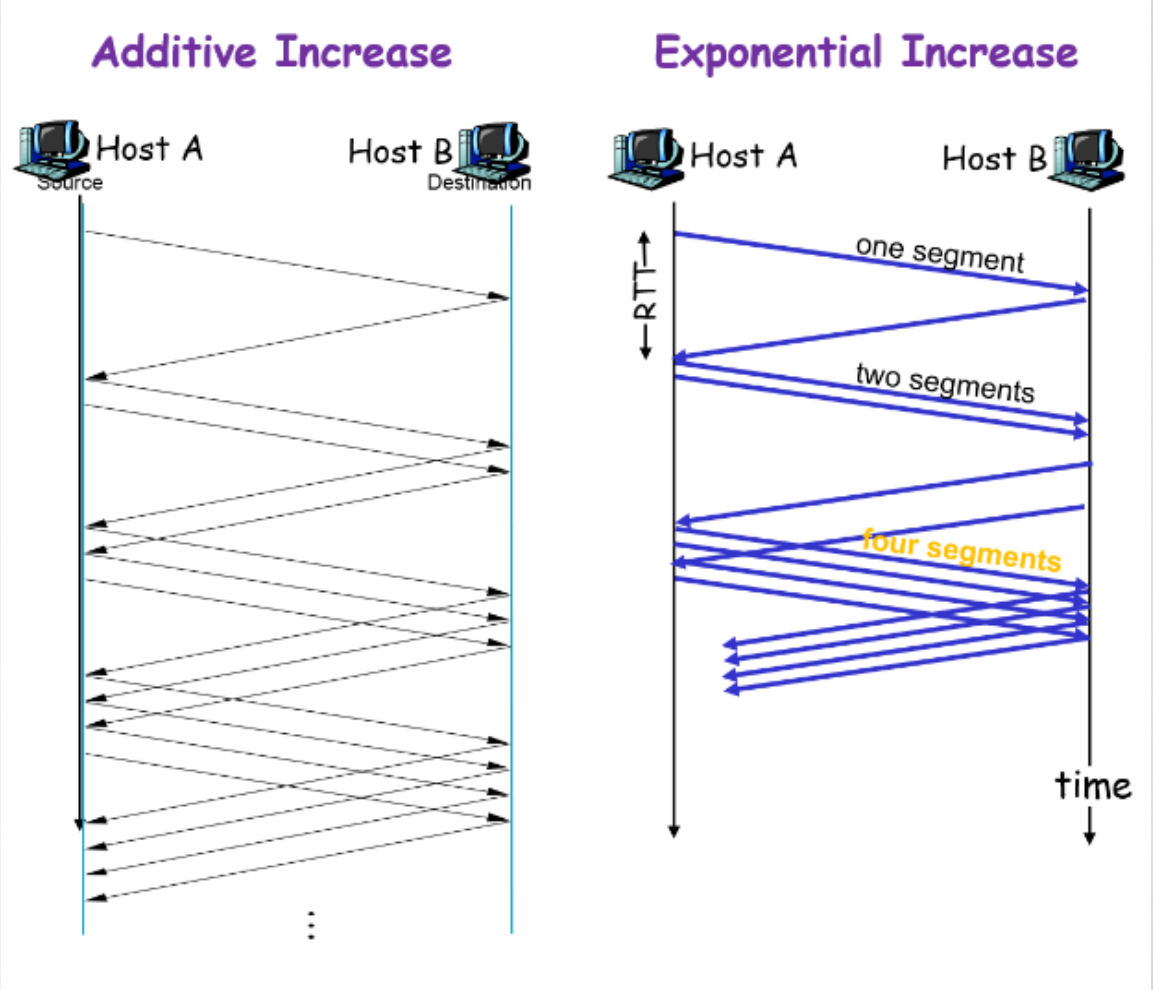




Timeout for how long: Basically the timeout is indicated by a retransmission of a packet that is now considered lost.

Question 1

Είναι το ίδιο να αυξάνεται το congestion window κατά μία μονάδα μετά τη λήψη από κάθε ACK πακέτου με το να αυξάνεται σε κάθε RTT;Αν δεν είναι το ίδιο σε ποια περίπτωση επιτυγχάνεται μεγαλύτερο data-rate στην TCP σύνδεσή;Δώστε σχήμα.



Answer 1

Αύξηση του παραθύρου συμφόρησης (congestion window) κατά μία μονάδα σε κάθε RTT σημαίνει γραμμική αύξησή του παραθύρου στον χρόνο (αριστερά στο σχήμα). Αντίθετα, αύξηση του παραθύρου κατά μία μονάδα με την λήψη κάθε ACK πακέτου ισοδυναμεί με διπλασιασμό του παραθύρου σε κάθε RTT, δηλαδή εκθετική αύξησή του στον χρόνο (δεξιά στο σχήμα). Στην δεύτερη περίπτωση το μέγεθος του παραθύρου αυξάνεται πολύ γρήγορα, επιτυγχάνοντας σύντομα μεγάλο data rate.

Στο πρωτόκολλο TCP υλοποιούνται και οι δύο αλγόριθμοι
Στην φάση της **αργής εκκίνησης** το μέγεθος του παραθύρου συμφόρησης τίθεται σε 1 MSS (MaximumSegmentSize) και στην **συνέχεια αυξάνεται εκθετικά στον χρόνο**.
Στην φάση της **αποφυγής συμφόρησης** το μέγεθος του **παραθύρου αυξάνεται γραμμικά** στον χρόνο μέχρι να συμβεί κάποιο συμβάν απώλειας πακέτου.

Question 2

Κατά τη διάρκεια μίας TCP ροής όταν συμβεί timeout τι επιπτώσεις θα υπάρχουν στο μέγεθος της πληροφορίας που θα στείλει αμέσως μετά ο TCP transmitter;

Αν συμβούν 2 duplicate ACKs αντί για timeout τι γίνεται σε αυτή την περίπτωσή;

Answer 2

Ο TCP transmitter εκτιμά την συμφόρηση ανιχνεύοντας events απώλειας πακέτων. Ένα πακέτο θεωρείται χαμένο αν:
1) εκπνεύσει ο χρονομετρητής (timeout event)
2) ληφθούν3 ίδια ACKs (3 duplicate ACKs event).

1) ο αποστολέας εισέρχεται στην φάση αργής εκκίνησης, δηλαδή θέτει το παράθυρο συμφόρησης σε 1 MSS και μετά αυξάνει το παράθυρο εκθετικά, μέχρι να φτάσει στο μισό της τιμής που είχε πριν από το συμβάν λήξης χρόνου (threshold= CongWin/2). Έπειτα, ο αποστολέας εισέρχεται στην φάση αποφυγής συμφόρησης και το παράθυρο αυξάνεται γραμμικά.

2) ο μετρητής των duplicate ACKs θα πάρει την τιμή 2, χωρίς όμως να πυροδοτεί κανένα από τους δύο παραπάνω τύπους event απώλειας πακέτου. Έτσι, ο αλγόριθμος αργής εκκίνησης θα συνεχίσει να εκτελείται κανονικά. Στην περίπτωση όμως που ληφθεί και τρίτο duplicate ACK, ο αποστολέας θα υποδιπλασιάσει το παράθυρο συμφόρησης, και στη συνέχεια θα εισέλθει στην φάση αποφυγής συμφόρησης, αυξάνοντας το παράθυρο γραμμικά.

Question 3

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 second segments contain 70 and 50 bytes of data, respectively. 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.

Question 3

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

Answer:
In the second segment from Host A to B, the sequence number is 197, source port number is 302 and destination port number is 80.

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?

Answer:
The acknowledgement number is 197, the source port number is 80 and the destination port number is 302.

Question 3

If the second segment arrives before the first segment, in the acknowledgment of the first arriving segment, what is the acknowledgment number?

Answer:
The acknowledgement number is 127, indicating that it is still waiting for bytes 127 and onwards.

Extra Material

TCP
https://packetlife.net/blog/2010/jun/7/understanding-tcp-sequence-acknowledgment-numbers/

Congestion Control
https://www.geeksforgeeks.org/tcp-congestion-control/

TCP Window
https://www.networkcomputing.com/data-centers/network-analysis-tcp-window-size

RTT and RTO
https://www.extrahop.com/company/blog/2016/retransmission-timeouts-rtos-application-performance-degradation/

Summary

- Layering the internet
- Transport Layer
- TCP (and UDP for comparison)
- TCP segment
- Three way handshake
- How to close a socket
- Sequence numbers
- Congestion control
- RTT and Timeouts
- Q&A

Questions