Computación en Internet I

Andrés A. Aristizábal P.
aaaristizabal@icesi.edu.co

Departamento de Tecnologías de Información y Comunicaciones

2023-1

Agenda

1. **TCP**
   - Introduction
   - TCP segment structure
   - Reliable Data Transfer
   - TCP Flow control
   - TCP Connection management
   - Socket Programming with TCP
2. **Exercises**

Agenda del día

What is the TCP?

What is the TCP?

- The Internet's transport-layer, connection-oriented, reliable transport protocol.

What is the TCP?

- The Internet's transport-layer, connection-oriented, reliable transport protocol.

What are its main characteristics?

What is the TCP?

- The Internet's transport-layer, connection-oriented, reliable transport protocol.

What are its main characteristics?

- Point-to-Point.
  - One sender, one receiver.
- Reliable, in-order byte stream.
  - No message boundaries
- Full duplex data.
  - Bi-directional data flow in same connection.
  - MSS: maximum segment size.

What are its main characteristics?

What are its main characteristics?

- Cumulative ACKs.
- Pipelining.
  - Sending multiple data units without waiting for an acknowledgment for the first frame sent.
  - TCP congestion and flow control set window size.
- Connection-oriented.
  - Handshaking (exchange of control messages) initializes sender, receiver state before data exchange.
- Flow controlled.
  - Sender will not overwhelm receiver

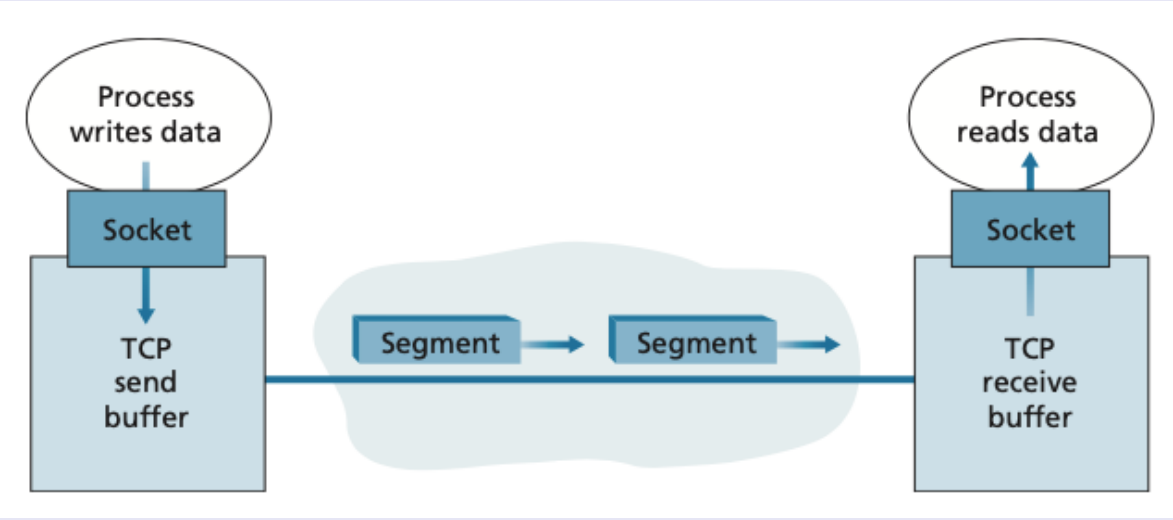
How does a TCP connection is established?

How does a TCP connection is established?

- A process running in one host wants to initiate a connection with another process in another host.
  - Client process - Server process.
- The client application process first informs the client transport layer that it wants to establish a connection to a process in the server.
- The client application process first informs the client transport layer that it wants to establish a connection
- A three-way handshake is performed.
  - The client first sends a special TCP segment.
  - The server responds with a second special TCP segment.
  - Finally the client responds again with a third special segment.
  - The first two segments carry no payload, that is, no application-layer data.
  - The third of these segments may carry a payload.

Then, how do two application processes exchange data?

Then, how do two application processes exchange data?



- TCP directs this data to the connection's send buffer.
- TCP will grab chunks of data from the send buffer and pass the data to the network layer.
  - Data is limited by the maximum segment size (MSS).
- TCP pairs each chunk of client data with a TCP header forming TCP segments.
- The segments are passed down to the network layer.
- There they are separately encapsulated within network-layer IP datagrams.
- The IP datagrams are then sent into the network.

Agenda del día

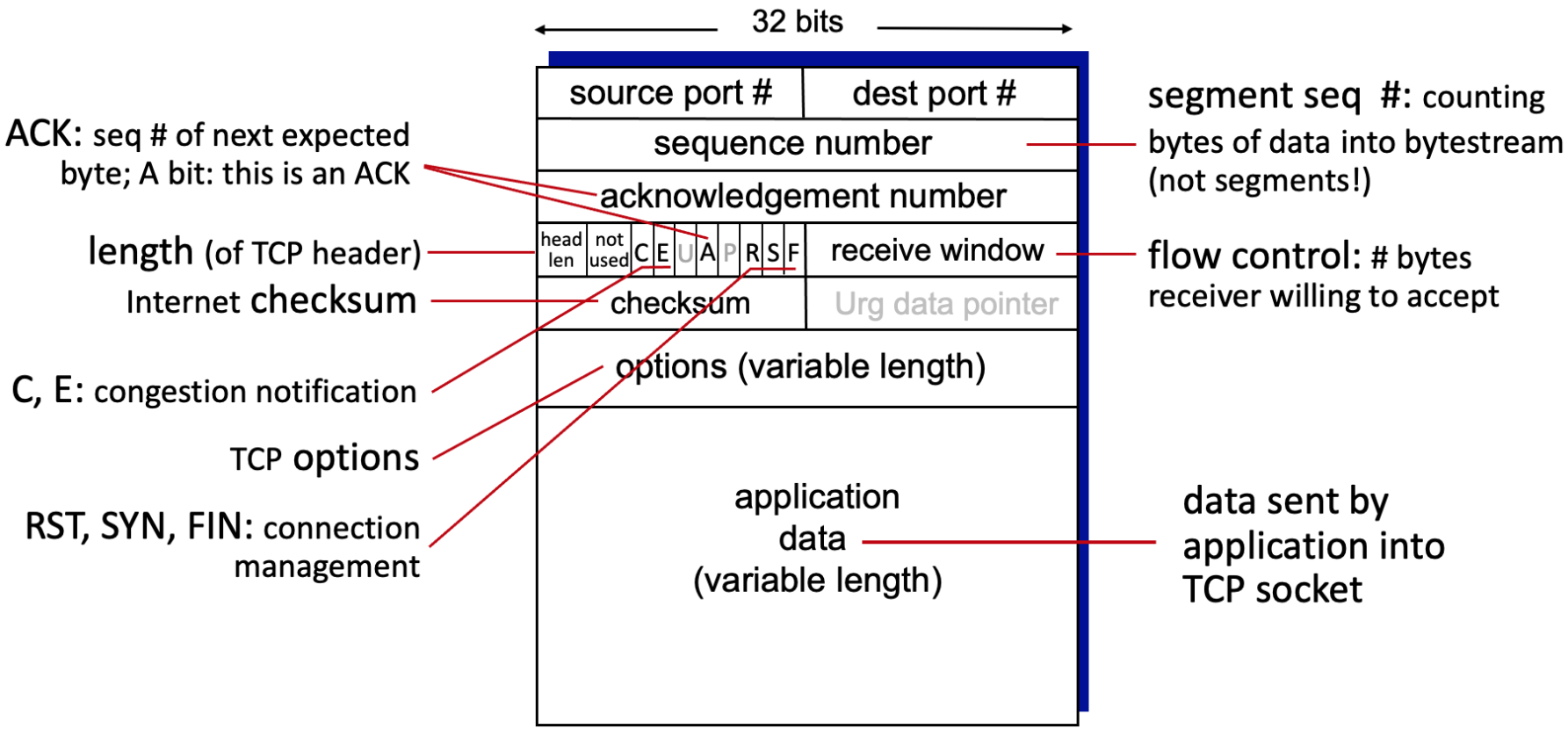
1. **TCP**
   - Introduction
   - TCP segment structure
   - Reliable Data Transfer
   - TCP Flow control
   - TCP Connection management
   - Socket Programming with TCP
2. **Exercises**

TCP segment structure

How is the TCP segment structure?

TCP segment structure

How is the TCP segment structure?



TCP segment structure

What is the purpose of TCP sequence numbers and ACKs?

TCP segment structure

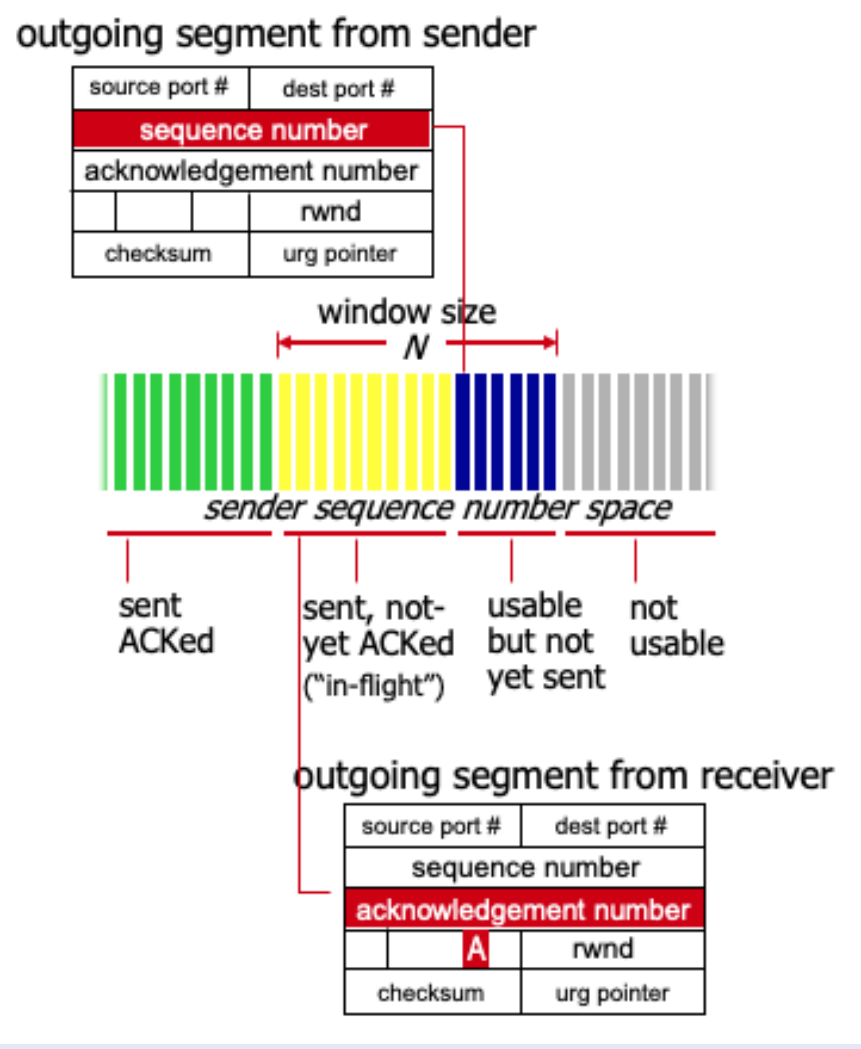
What is the purpose of TCP sequence numbers and ACKs?

- They are a critical part of TCP's reliable data transfer service.
- Sequence numbers
  - Byte stream number of first byte in segment's data.
  - The host implicitly numbers each byte in the data stream.
  - Each sequence number is inserted in the sequence number field.
- Acknowledgements
  - Sequence number of next byte expected from other side.
  - Cumulative ACK.

TCP segment structure

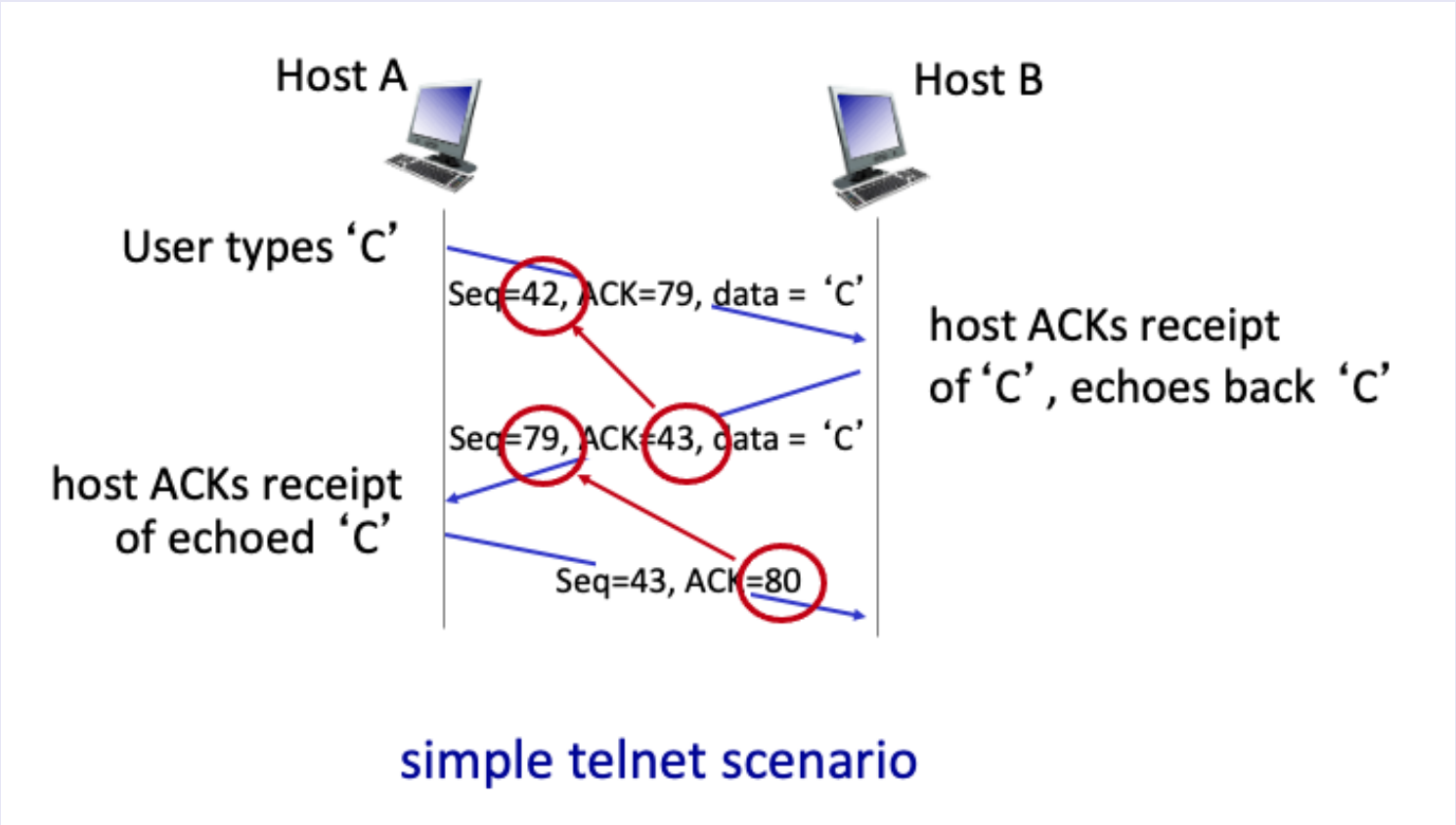
How the are outgoing segments?

How the are outgoing segments?



How about a Telnet example?

How about a Telnet example?



simple telnet scenario

TCP segment structure

How does TCP recover from lost segments?

TCP segment structure

How does TCP recover from lost segments?

- By using a timeout/retransmit mechanism.

TCP segment structure

How does TCP recover from lost segments?

- By using a timeout/retransmit mechanism.

How to set a TCP timeout value?

TCP segment structure

How does TCP recover from lost segments?

- By using a timeout/retransmit mechanism.

How to set a TCP timeout value?

- Longer than RTT (time from when a segment is sent until it is acknowledged).
- Too short: premature timeout, unnecessary retransmissions.
- Too long: slow reaction to segment loss.

TCP segment structure

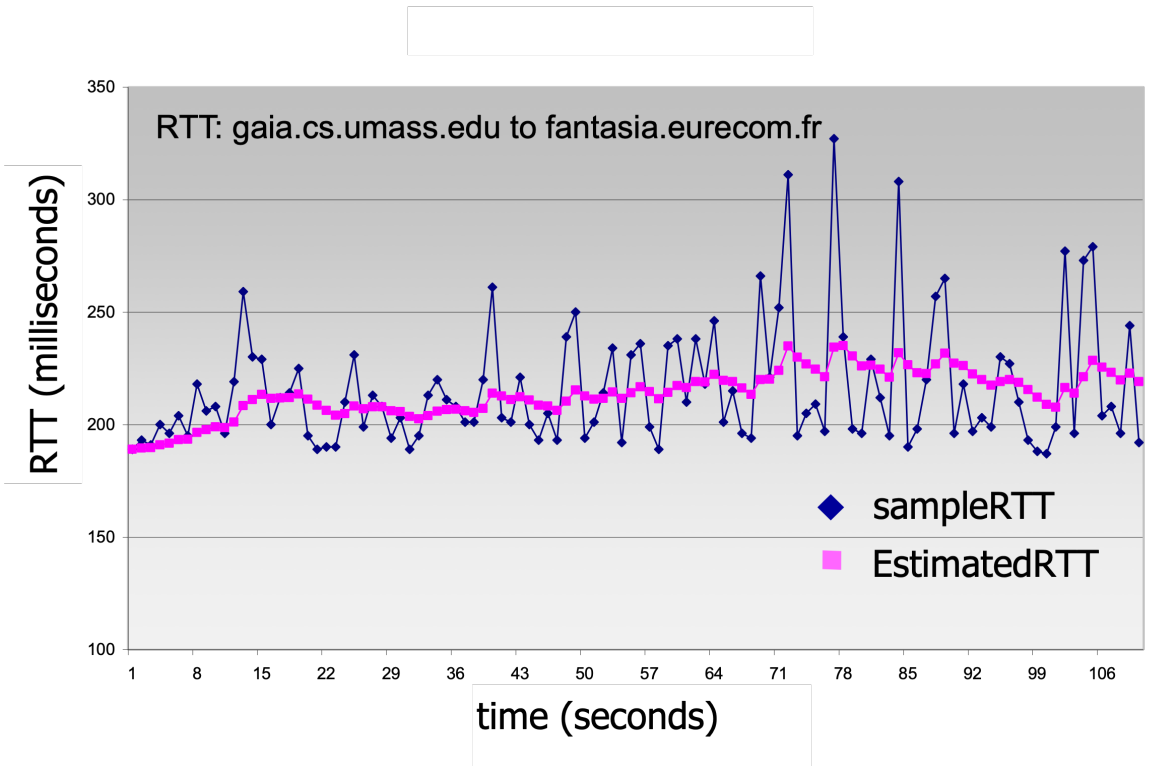
How to estimate RTT?

How to estimate RTT?

- SampleRTT: measured time from segment transmission until ACK receipt.
  - Ignore retrasmissions.
  - Only one SampleRTT measurement at a time.
- SampleRTT will vary.
  - Want estimated RTT smoother.
  - Average several recent measurements, not just current SampleRTT.

How to estimate RTT?

- EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT.
- exponential weighted moving average (EWMA).
- Influence of past sample decreases exponentially fast
- Typical value: $\alpha = 1/8 = 0.125$.



How to find the correct timeout interval?

- Timeout interval: EstimatedRTT plus **safety margin.**
  - Large variation in EstimatedRTT: want a larger safety margin.
- TimeoutInterval = EstimatedRTT + 4*DevRTT.
- DevRTT: EWMA of SampleRTT deviation from EstimatedRTT.
  - DevRTT = (1-β)*DevRTT + β*|SampleRTT-EstimatedRTT|.
  - Typically, β = 0.25.

Agenda del día

1. **TCP**
   - Introduction
   - TCP segment structure
   - Reliable Data Transfer
   - TCP Flow control
   - TCP Connection management
   - Socket Programming with TCP
2. **Exercises**

How does a TCP sender recover from lost segments?

How does a TCP sender recover from lost segments?

- Event: data received from application
  - Create segment with sequence number.
  - Sequence number is a byte-stream number of first data byte in segment.
  - Start timer if not already running.
    - Think of timer as for oldest unACKed segment.
    - Expiration interval: TimeOutInterval.
- Event: timeout.
  - Retransmit segment that caused timeout.
  - Restart timer.
- Event: ACK received.
  - If ACK acknowledges previously unACKed segments.
    - Update what is known to be ACKed.
    - Start timer if there are still unACKed segments.

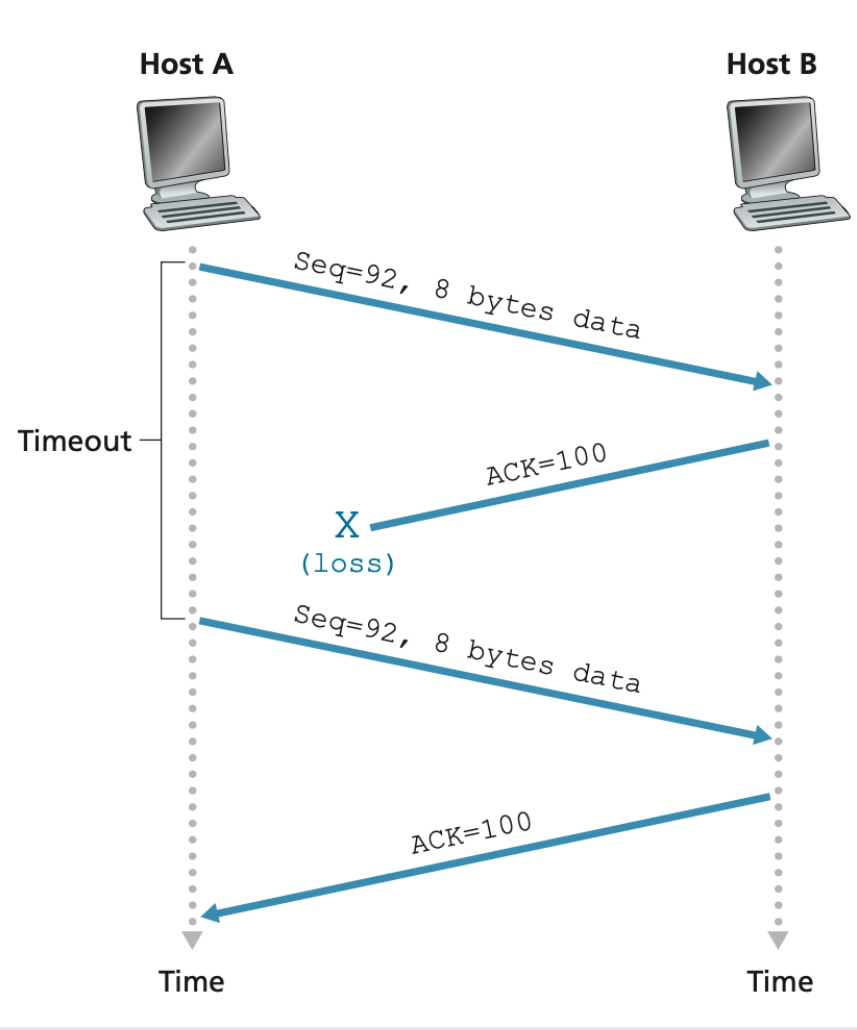
How is the TCP ACK generation?

How is the TCP ACK generation?

Event	TCP Receiver action
in-order segment arrival, no gaps, everything else already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
in-order segment arrival, no gaps, one delayed ACK pending	immediately send single cumulative ACK
out-of-order segment arrival higher-than-expect seq. # gap detected	send duplicate ACK, indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate ACK if segment starts at lower end of gap

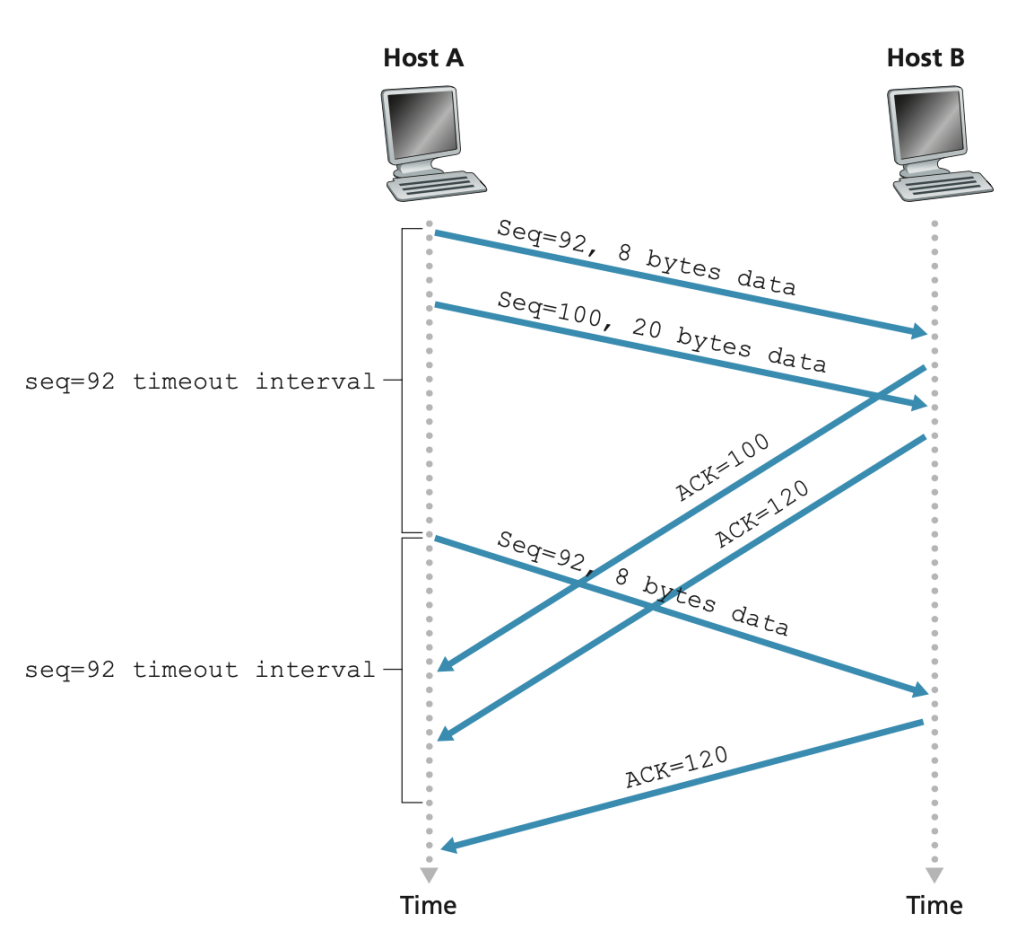
Retransmission due to a lost acknowledgment

Retransmission due to a lost acknowledgment



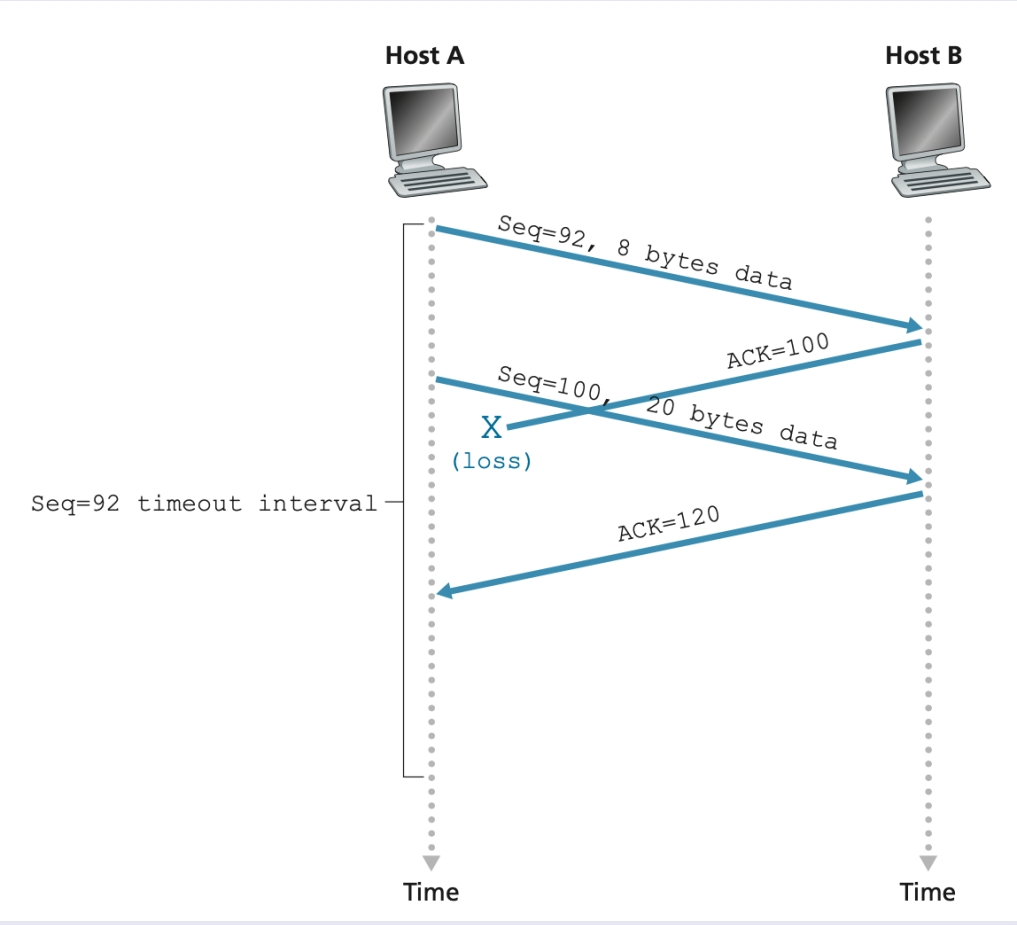
Segment 100 not retransmitted

Segment 100 not retransmitted



A cumulative acknowledgment avoids retransmission of the first segment

A cumulative acknowledgment avoids retransmission of the first segment



What are the problems of the timeout/retransmit mechanism?

What are the problems of the timeout/retransmit mechanism?

- In timeout-triggered retransmissions the timeout period can be relatively long.
- Long timeout period forces the sender to delay resending the lost packet (increasing end-to-end delay).

What are the problems of the timeout/retransmit mechanism?

- In timeout-triggered retransmissions the timeout period can be relatively long.
- Long timeout period forces the sender to delay resending the lost packet (increasing end-to-end delay).

What can be done about it?

What are the problems of the timeout/retransmit mechanism?

- In timeout-triggered retransmissions the timeout period can be relatively long.
- Long timeout period forces the sender to delay resending the lost packet (increasing end-to-end delay).

What can be done about it?

- Fast retransmit.
  - The sender can often detect packet loss well before the timeout event occurs by noting so-called duplicate ACKs.
  - A duplicate ACK is an ACK that reacknowledges a segment for which the sender has already received an earlier acknowledgment.

An example of fast retransmit

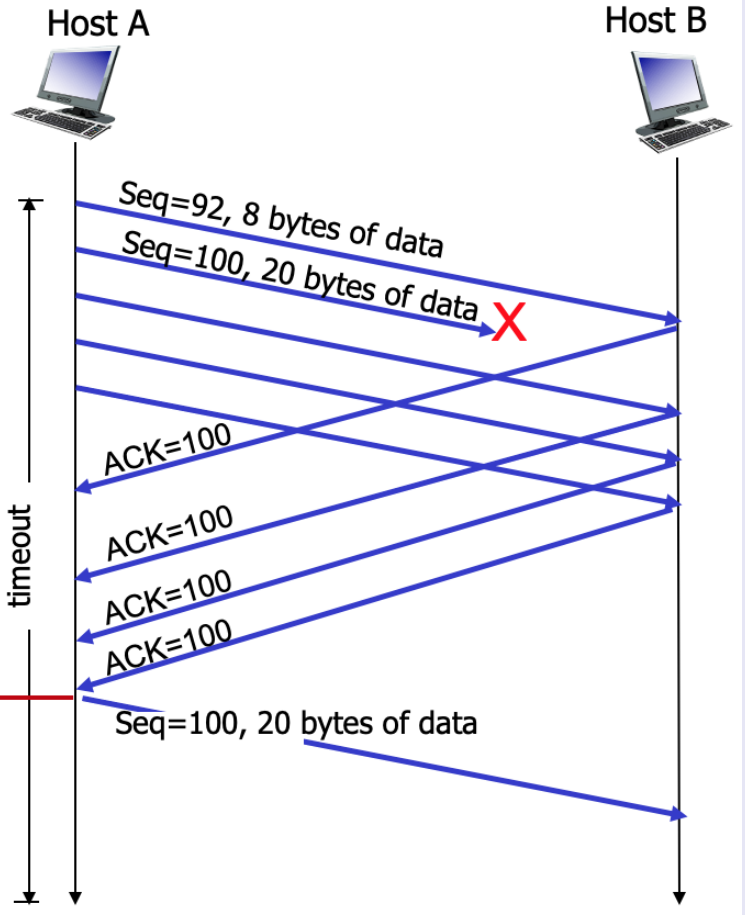
An example of fast retransmit

TCP fast retransmit

if sender receives 3 additional ACKs for same data ("triple duplicate ACKs"), resend unACKed segment with smallest seq #

- likely that unACKed segment lost, so don't wait for timeout

Receipt of three duplicate ACKs indicates 3 segments received after a missing segment – lost segment is likely. So retransmit!



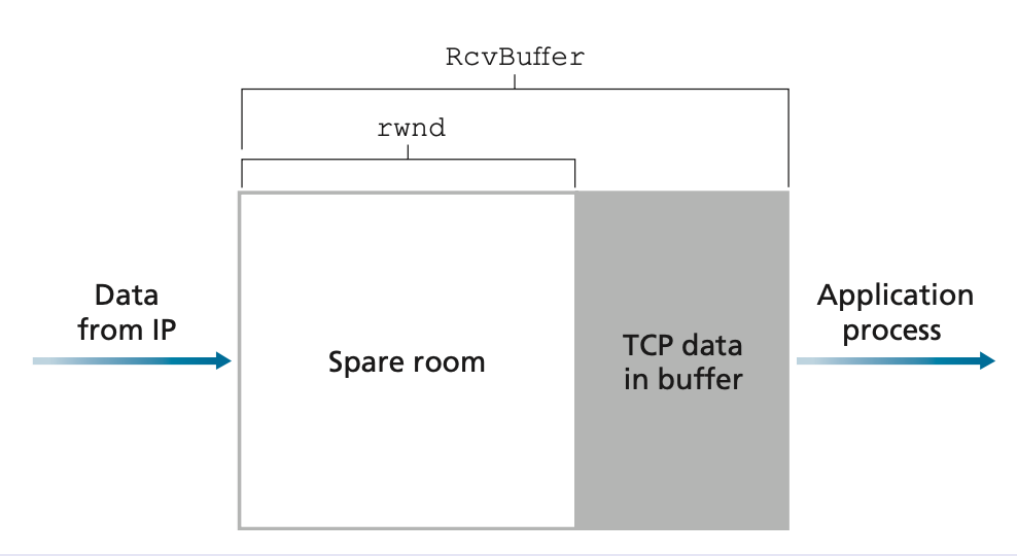
Agenda del día

1. **TCP**
   - Introduction
   - TCP segment structure
   - Reliable Data Transfer
   - TCP Flow control
   - TCP Connection management
   - Socket Programming with TCP
2. **Exercises**

TCP Flow control

How does TCP flow control work?

How does TCP flow control work?



- The receive side of TCP connection has a receive buffer.
- App process may be slow at reading from buffer.
- Receiver controls sender, so sender won't overflow receiver's buffer by transmitting too much, too fast.
- TCP receiver advertises free buffer space in rwnd field in TCP header.
  - The receiver informs the sender how much free buffer space there is
  - The sender is limited to send no more than this amount of data.
  - This goes from receiver to the sender in the receiver window in the TCP header.
  - The value will change as the amount of free buffer space fluctuates over time.

Agenda del día

1. **TCP**
   - Introduction
   - TCP segment structure
   - Reliable Data Transfer
   - TCP Flow control
   - TCP Connection management
   - Socket Programming with TCP
2. **Exercises**

TCP Connection management

How does TCP establish a connection?

TCP Connection management

How does TCP establish a connection?

- TCP sender, receiver establish connection before exchanging data segments.
  - Initialize TCP variables.
    - Sequence numbers.
    - Buffers, flow control info (e.g. RcvWindow).
  - Client: connection initiator.
    - Socket clientSocket = new Socket(hostname,portnumber);
  - Server: contacted by client.
    - Socket connectionSocket = welcomeSocket.accept();

TCP Connection management

How does the three way handshake work?

How does the three way handshake work?

Three way handshake:

Step 1: client host sends TCP SYN segment to server

- specifies initial seq #
- no data

Step 2: server host receives SYN, replies with SYNACK segment

- server allocates buffers
- specifies server initial seq. #

Step 3: client receives SYNACK, replies with ACK segment, which may contain data

How does TCP close a connection?

How does TCP close a connection?

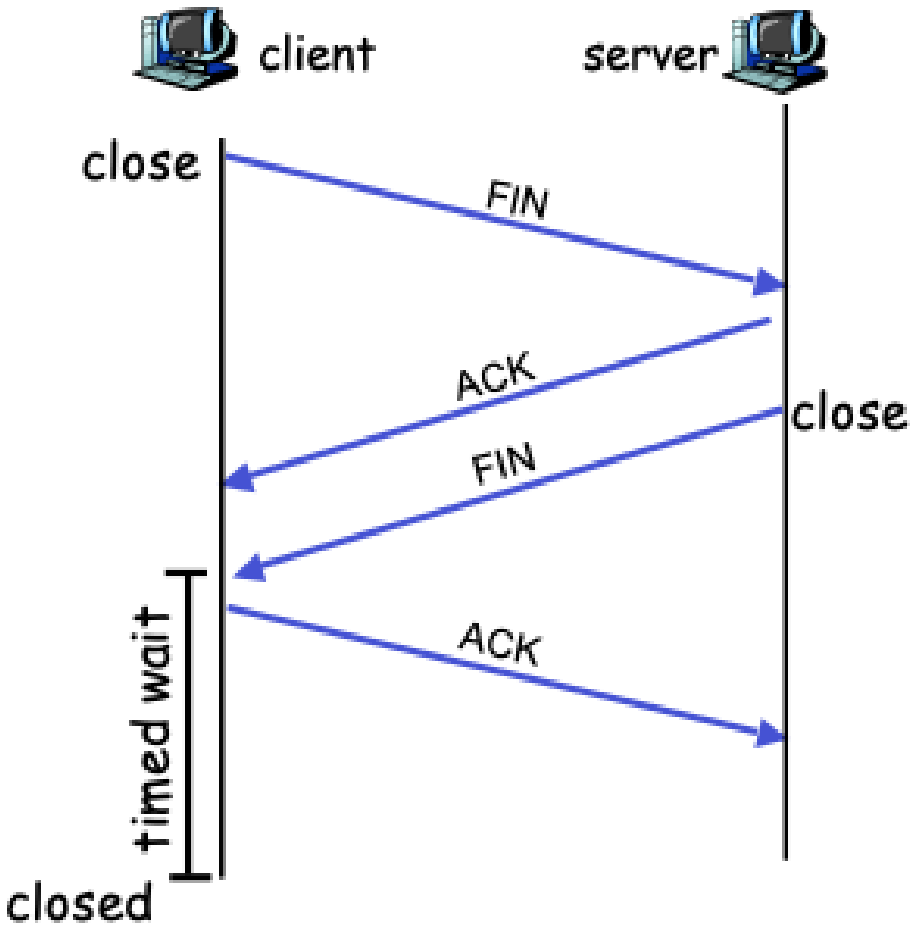
Closing a connection:

client closes socket:

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

Step 1: client end system sends TCP FIN control segment to server

Step 2: server receives FIN, replies with ACK. Closes connection, sends FIN.



Agenda del día

1. **TCP**
   - Introduction
   - TCP segment structure
   - Reliable Data Transfer
   - TCP Flow control
   - TCP Connection management
   - Socket Programming with TCP
2. **Exercises**

How would the code for the client side of the application be?

How would the code for the client side of the application be?

```java
import java.io.*;
import java.net.*;

class TCPClient {
  public static void main(String argv[]) throws Exception {
    String sentence;
    String modifiedSentence;
    BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
    Socket clientSocket = new Socket("127.0.0.1", 6789);
    DataOutputStream outToServer = new DataOutputStream(clientSocket.getOutputStream());
    BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
    sentence = inFromUser.readLine();
    outToServer.writeBytes(sentence + '\n');
    modifiedSentence = inFromServer.readLine();
    System.out.println("FROM SERVER: " + modifiedSentence);
    clientSocket.close();
  }
}
```

How would the code for the server side of the application be?

How would the code for the server side of the application be?

```java
import java.io.*;
import java.net.*;

class TCPServer {
  public static void main(String argv[]) throws Exception {
    String clientSentence;
    String capitalizedSentence;
    ServerSocket welcomeSocket = new ServerSocket(6789);
    while(true) {
      Socket connectionSocket = welcomeSocket.accept();
      BufferedReader inFromClient = new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
      DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());
      clientSentence = inFromClient.readLine();
      capitalizedSentence = clientSentence.toUpperCase() + '\n';
      outToClient.writeBytes(capitalizedSentence);
    }
  }
}
```

Exercises

1. Write a TCP client/server system in which the client program sends an integer number and the server program returns its square.
2. Write a Simple Calculator via TCP.