Redes de Computadores

The Transport Layer

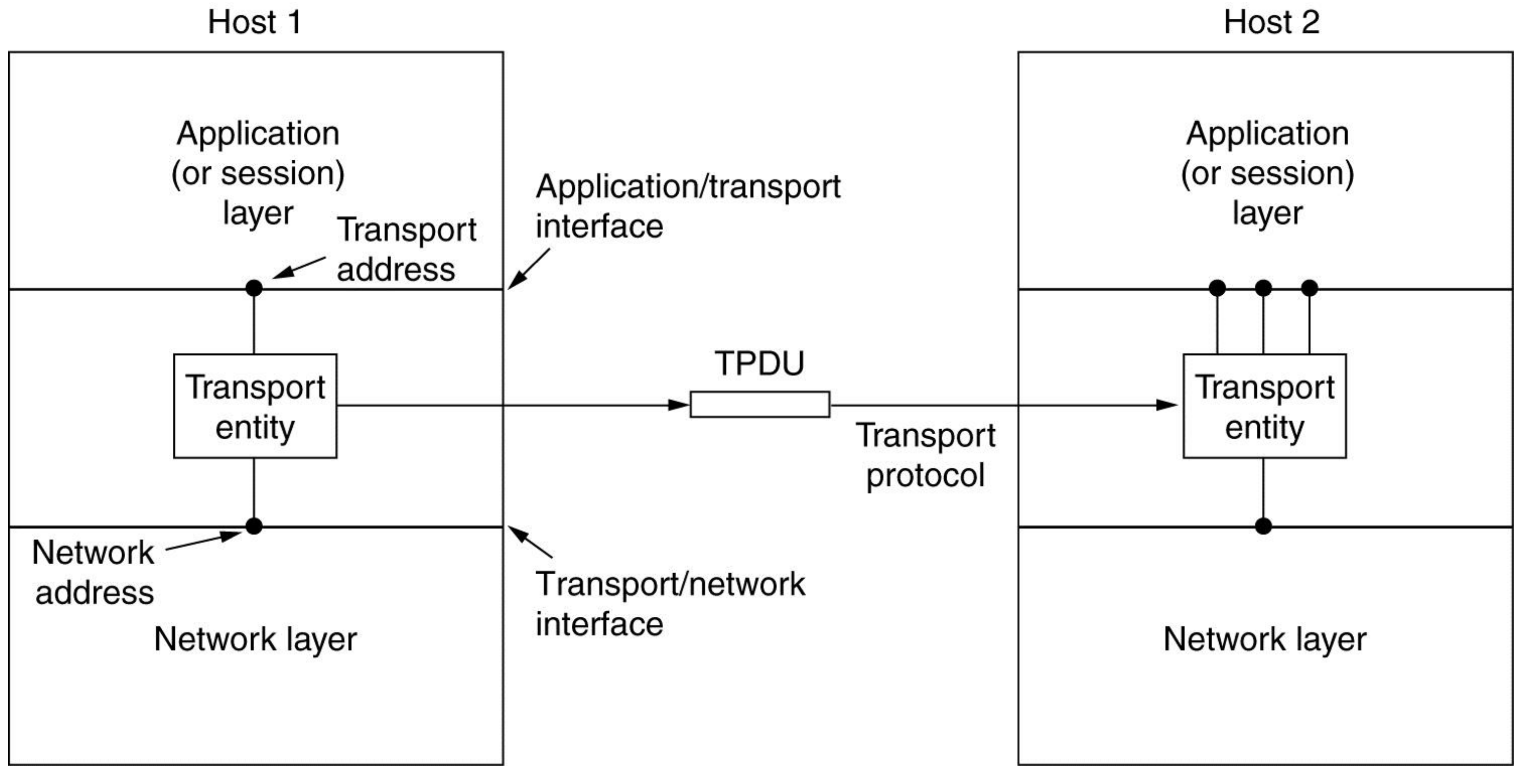
Manuel P. Ricardo

Faculdade de Engenharia da Universidade do Porto

- *What are the services provided by the Transport Layer?*
- *What are the transport protocols in the TCP/IP stack?*
- *What are the differences between UDP and TCP?*
- *How is the connection established in TCP?*
- *What is the difference between flow control and congestion control?*
- *How does TCP implement flow control?*
- *What mechanisms does TCP adopt to prevent network congestion control?*
- *Why is it the congestion control mechanism implemented by TCP so important for the behaviour of the Internet?*

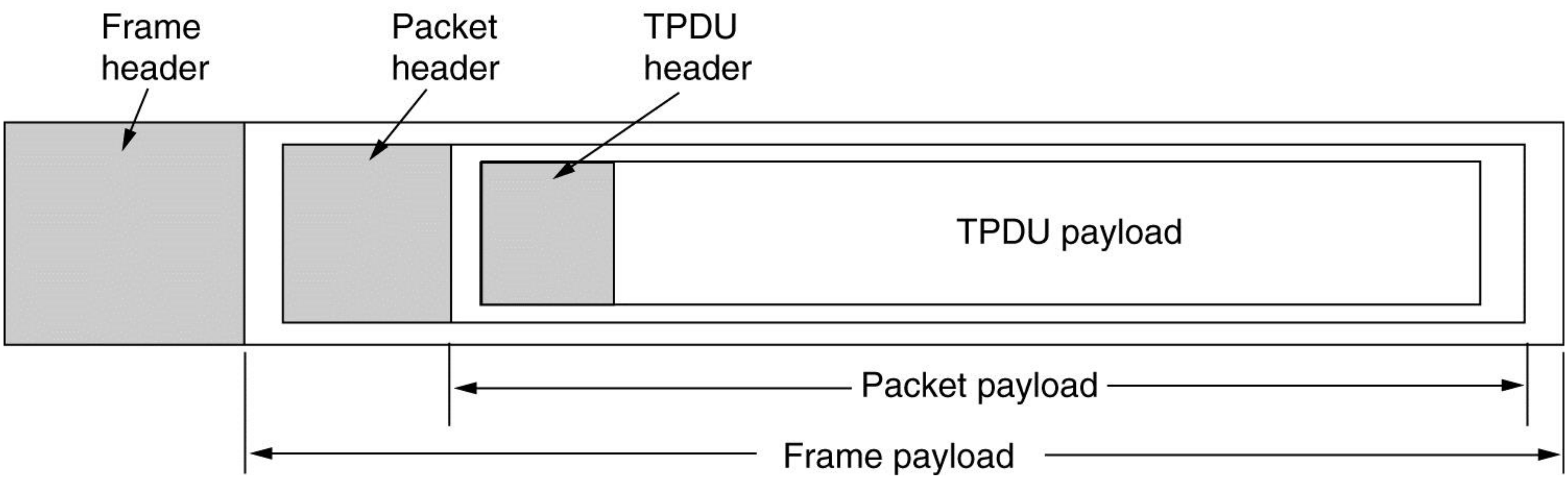
Services Provided to the Upper Layers

The network, transport, and application layers

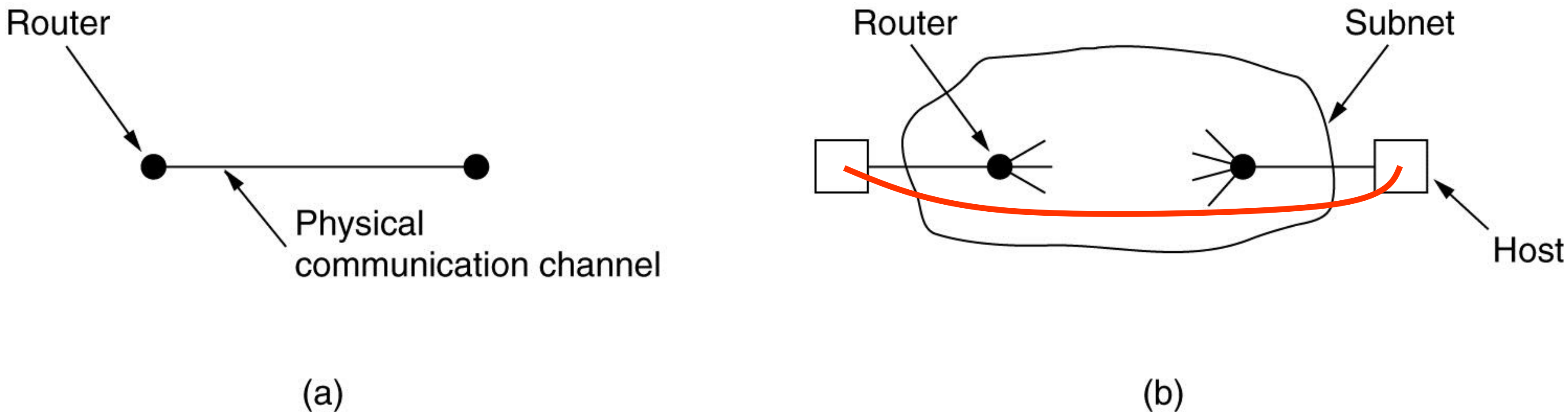


Transport Service Primitives

The nesting of TPDUs, packets, and frames



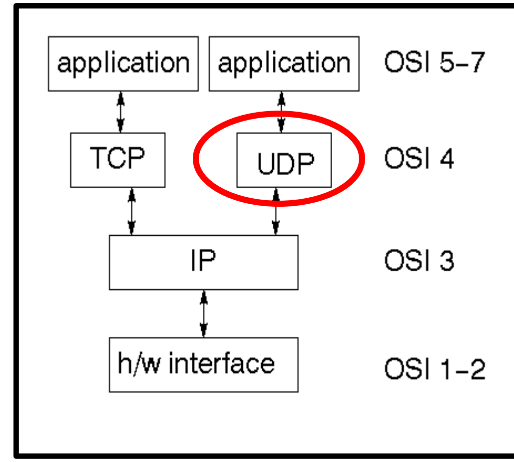
Transport Protocol



(a) Environment of the data link layer

(b) Environment of the transport layer

UDP - User Datagram Protocol (UDP)



- Datagram oriented
  - » Unreliable ➔ no error control mechanism
  - » Connectionless
- Allows applications

  to interface directly to IP

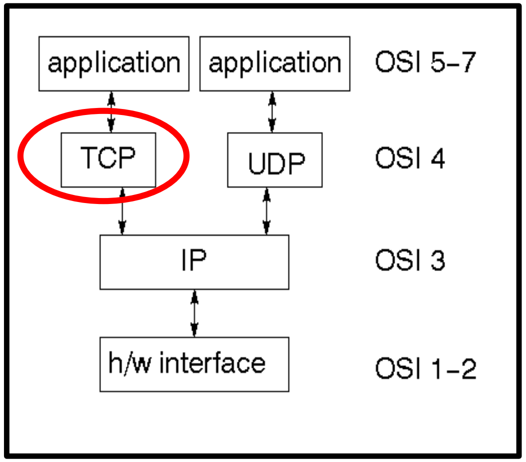
  with minimal additional protocol overhead
- UDP header
  - » Port numbers identify sending and receiving processes
  - » UDP length = length of packet in bytes
  - » Checksum covers header and data; optional

32 Bits

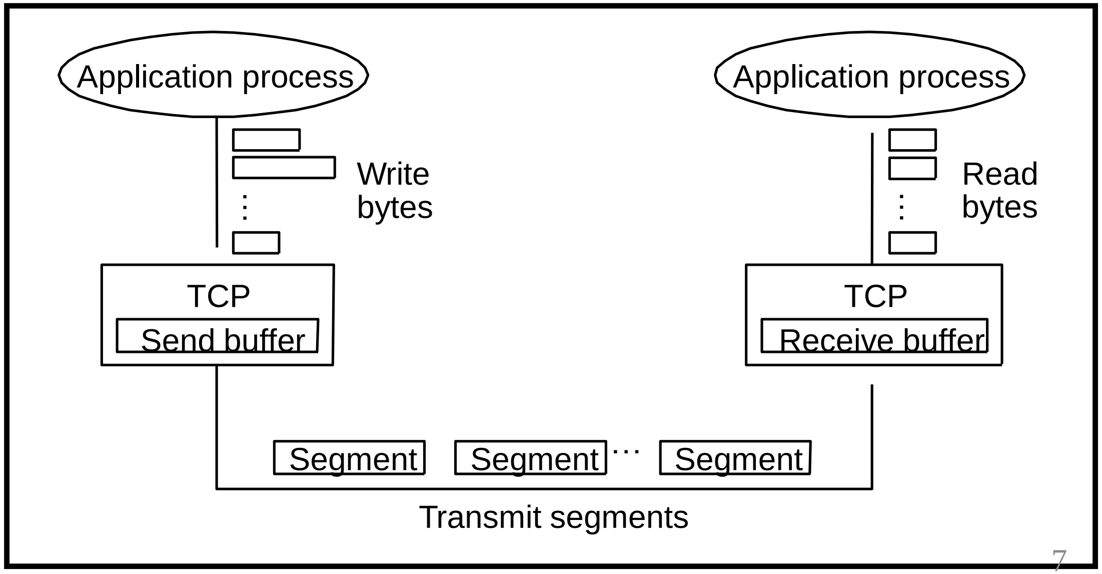
Source port	Destination port
UDP length	UDP checksum

TCP – Transmission Control Protocol

- Connection oriented
- Full-duplex
- Byte stream



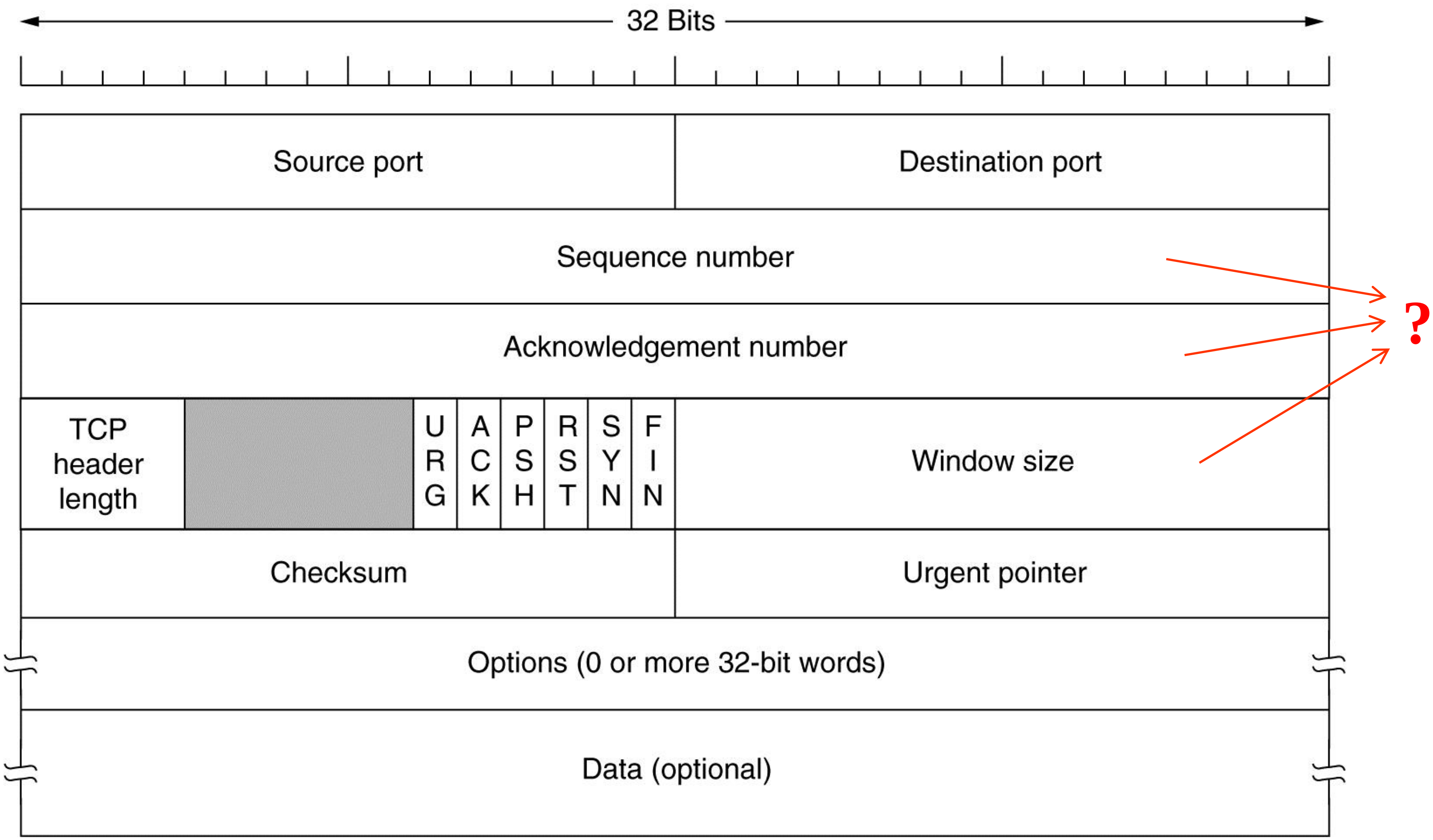
- Flow control
  - » Reliability
  - » ARQ mechanism
  - » Avoids receiver's congestion
- Congestion control
  - » Avoids network's congestion



Basic TCP Operation

- Sender
  - Application data is broken in segments
  - TCP uses timer while waiting for an ACK of every segment sent
  - Un-ACKed segments are retransmitted
- Receiver
  - Errors detected using a checksum
  - Correctly received data is acknowledged
  - Segments reassembled in proper order
  - Duplicated segments discarded
- Window based flow control

The TCP Segment Header

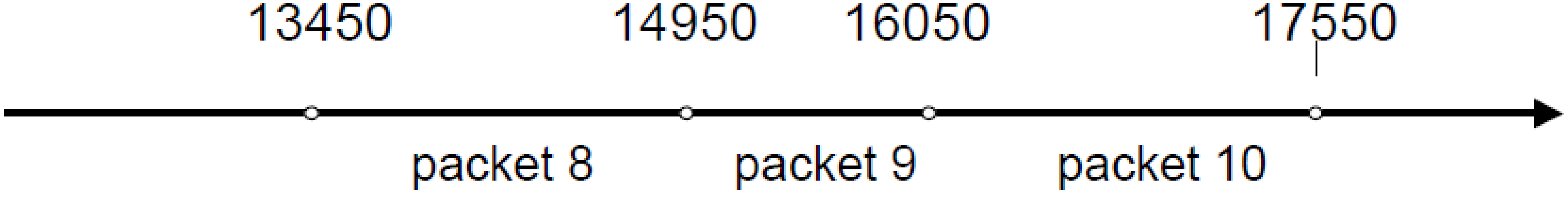


TCP Header

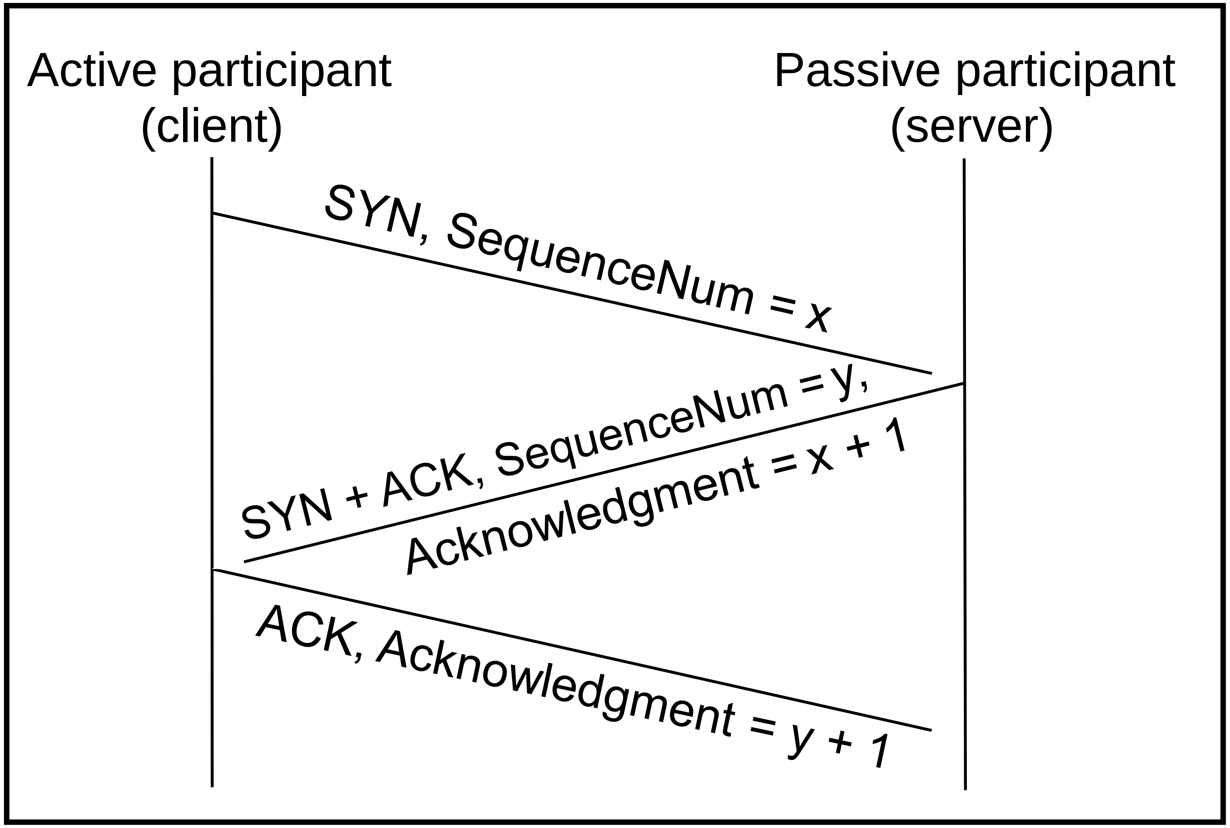
- Ports number are the same as for UDP
- 32 bit SeqNumber uniquely identifies the application data contained in the TCP segment
  - » SeqNumber is in bytes
  - » It identifies the first byte of data
- 32 bit AckNumber is used for piggybacking ACKs
  - » AckNumber indicates the next byte the receiver is expecting
  - » Implicit ACK for all of the bytes up to that point
- Window size
  - » Used for flow control (ARQ) and congestion control
    - Sender cannot have more than a window of bytes in the network
  - » Specified in bytes
    - Window scaling used to increase the window size in high speed networks
- Checksum covers the header and data

Sequence Numbers in TCP

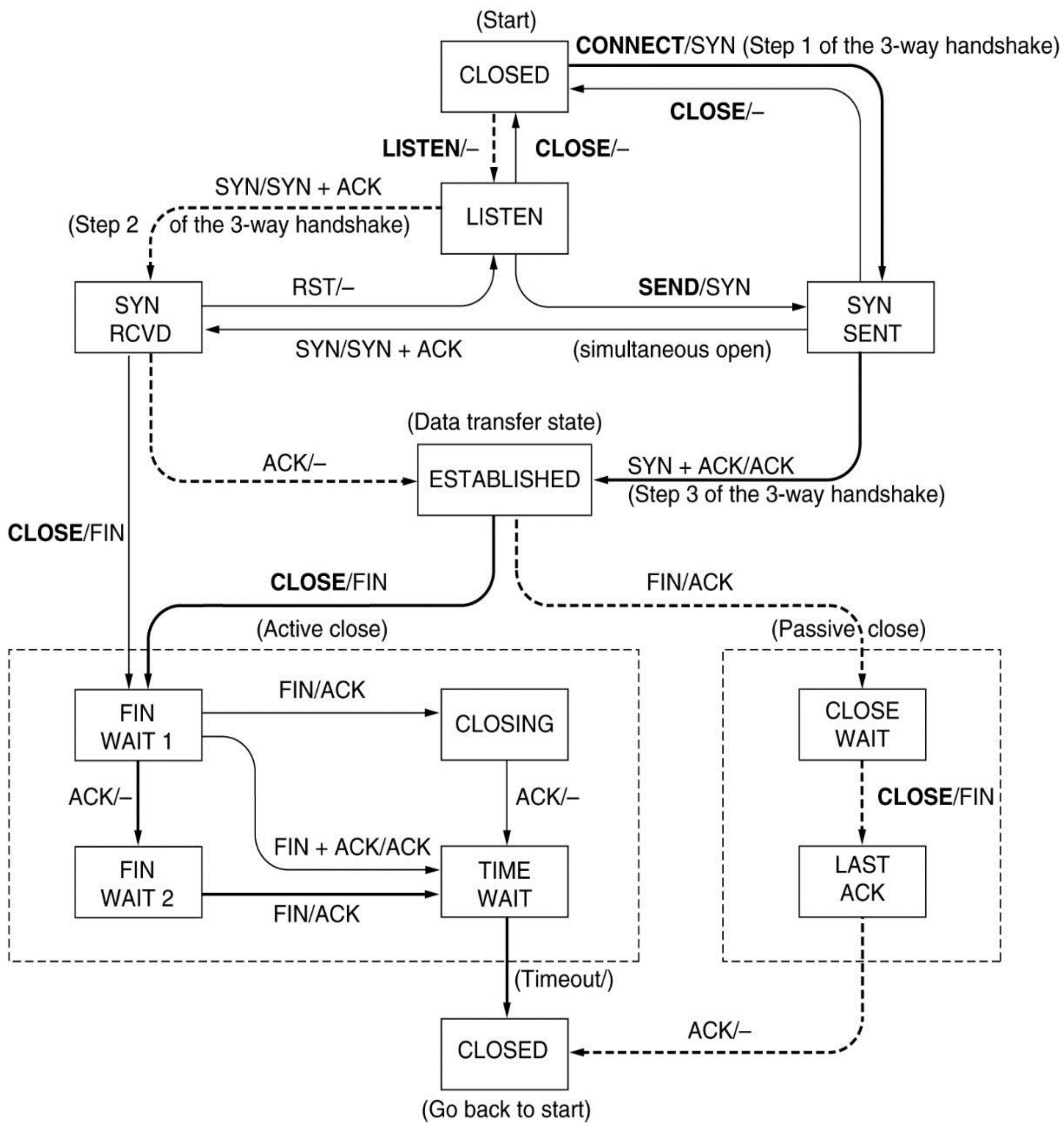
- TCP regards data as a *byte-stream*
  - each byte in stream is numbered sequentially
- TCP breaks byte stream into segments
  - size limited by the Maximum Segment Size (MSS)
- Each packet has a sequence number
  - sequence number of the 1^{st} byte of data transported by the segment
- TCP connection is duplex
  - data in each direction has different sequence numbers



Connection Establishment

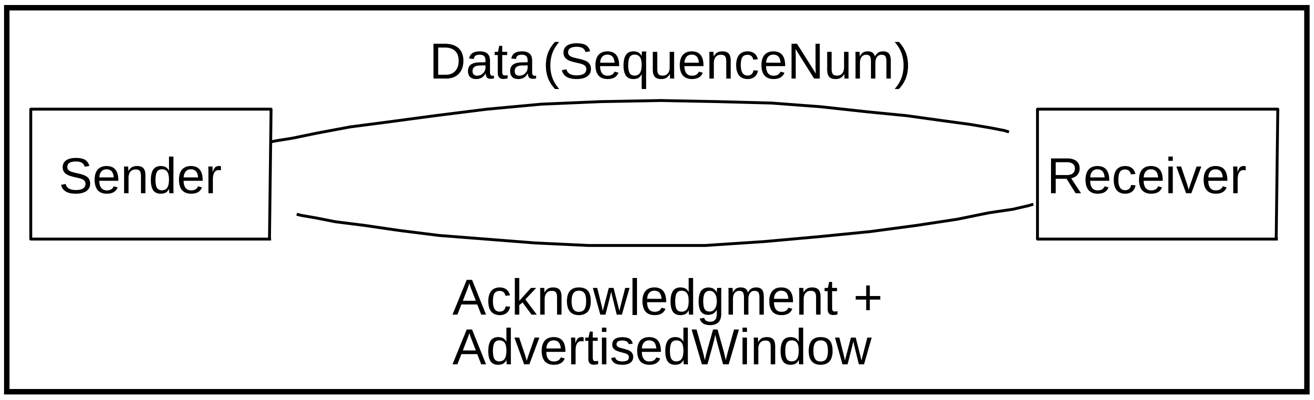


TCP Connection Management

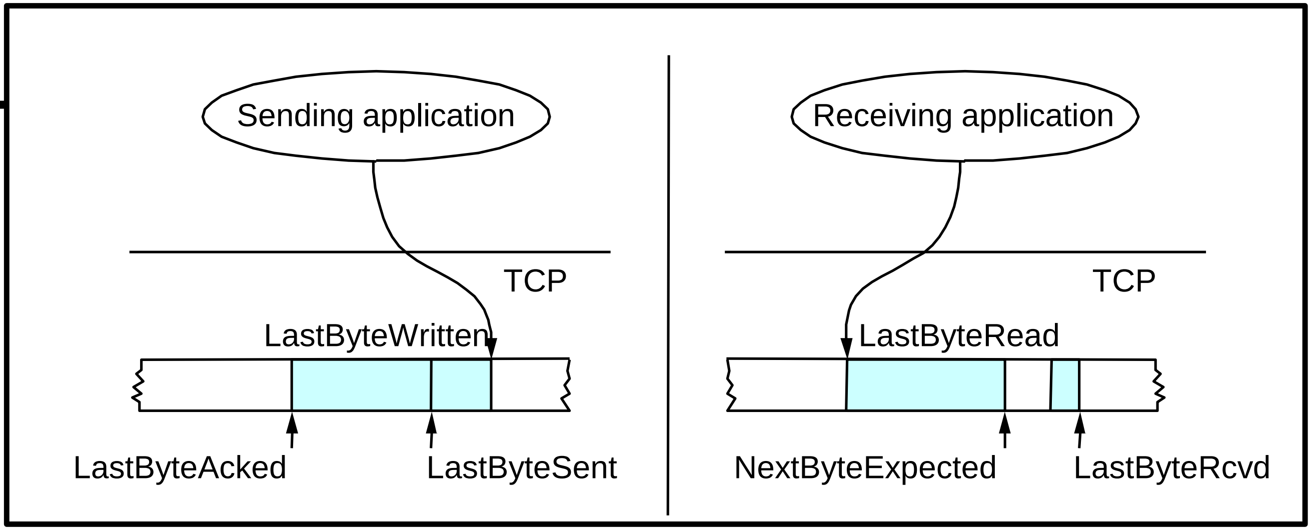


Retransmissions in TCP – A variation of Go-Back-N

- Sliding window
  - » Ack contains a single sequence number
  - » acknowledges all bytes with a lower sequence number
  - » duplicate ACKs sent when out-of-order packet received
- Sender retransmits a single packet at a time
  - » optimistic assumption → only one packet is lost
- Error control based on byte sequences, not packets



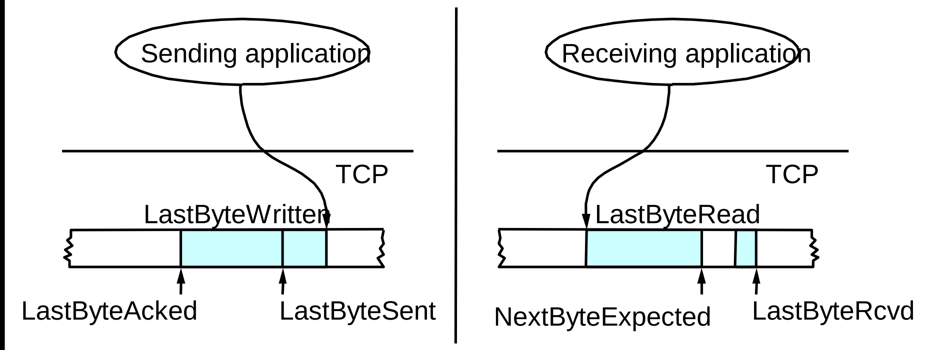
Sliding Window



» Sender
  - **`LastByteAcked < = LastByteSent`**
  - **`LastByteSent < = LastByteWritten`**
  - Buffers bytes between **`LastByteAcked`** and **`LastByteWritten`**

» Receiver
  - **`LastByteRead <  NextByteExpected`**
  - **`NextByteExpected < = LastByteRcvd +1`**
  - Buffers bytes between **`LastByteRead`** e **`LastByteRcvd`**

Flow Control



- Buffer length
  - Sender → **`MaxSendBuffer`**
  - Receiver → **`MaxRcvBuffer`**
- Receiver

  `LastByteRcvd - LastByteRead < = MaxRcvBuffer`

  `AdvertisedWindow = MaxRcvBuffer -(LastByteRcvd - LastByteRead)`

- Sender

  `LastByteWritten - LastByteAcked < = MaxSendBuffer`

  `LastByteSent - LastByteAcked < = AdvertisedWindow`

  `EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAcked)`

- Sending application blocks if it needs to write y bytes and

  `(LastByteWritten - LastByteAcked) + y > MaxSenderBuffer`

- **ACK** sent when a segment is received

- TCP works in the Internet. How to determine a reasonable value for the retransmission timer?

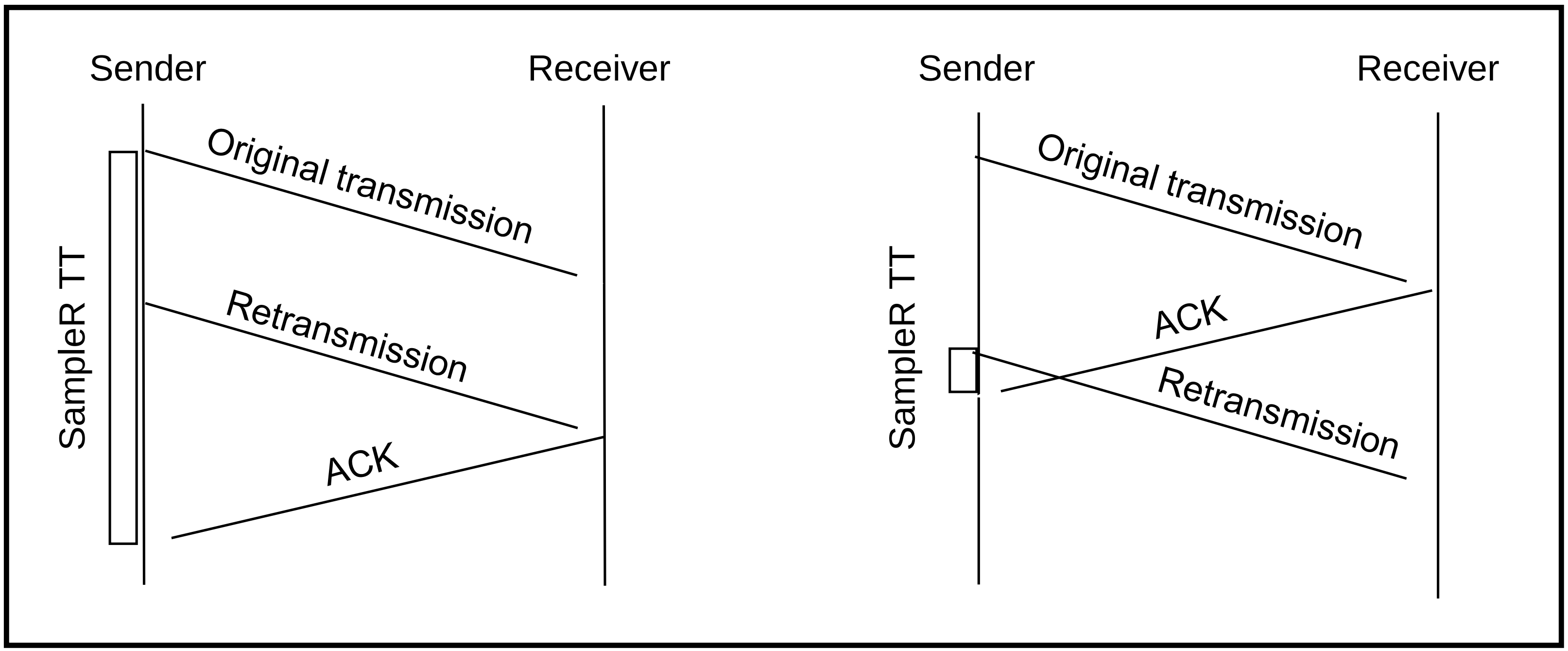
Adaptive Retransmission (Original Algorithm)

- RTT → Round Trip Time
- **`sampleRTT`** measured for each **`segment/ACK`** pair
- Average **`RTT`**
  - » **`RTT = a x RTT + (1-a) x sampleRTT`**

    `a in [0.8, 0.9]`
- **`TimeOut = 2 x RTT`**

Karn/Partridge Algorithm

- `sampleRTT` not measured in retransmission
- Timeout doubled for each retransmission



Selective ACK

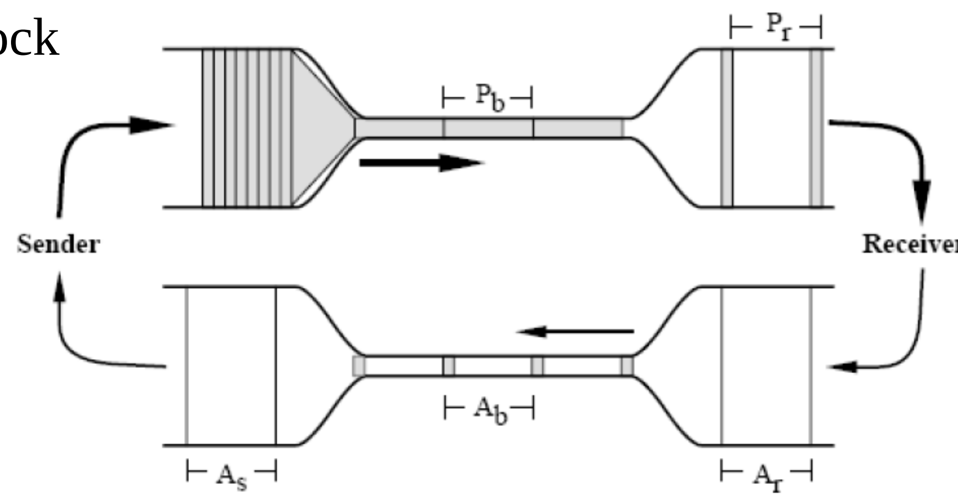
- Option for selective ACKs (SACK) also widely deployed

- Selective acknowledgement (SACK)
  - adds a bitmask of packets received
  - implemented as a TCP option

- When to retransmit?
  - packets may experience different delays
  - still need to deal with reordering
  - wait for out of order by 3 packets

TCP – Congestion Control

- Main idea
  - » each source determines its capacity
  - » based on criteria enabling
    - – flow fairness
    - – efficiency
- Received **`ACKs`** regulate packet transmission
  - ➔ they are the source clock



Additive Increase/Multiplicative Decrease

- Changes in channel capacity → adjustment of transmission rate
- New variable per connection → **`CongestionWindow`**
  - » limits the amount of traffic in transit
    - – **`MaxWin = MIN(CongestionWindow, AdvertisedWindow)`**
    - – **`EffWin = MaxWin - (LastByteSent - LastByteAcked)`**
- Objective
  - » If network congestion decreases → **`CongestionWindow`** increases
  - » If network congestion increases → **`CongestionWindow`** decreases
- **`Bitrate (byte/s) → CongestionWindow/RTT`**

Additive Increase/Multiplicative Decrease

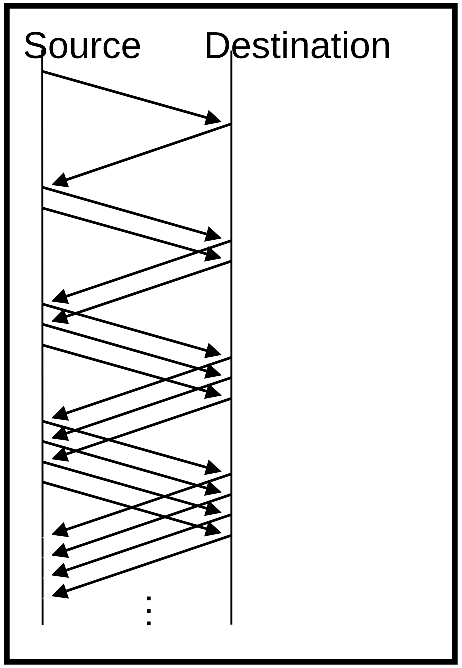
How does the source know if/when the network is in congestion?

➔ By timeout

- » timeout occurrence ➔ loss of packet
- » packet loss ➔ buffer in router is full ➔ congestion

Additive Increase/Multiplicative Decrease

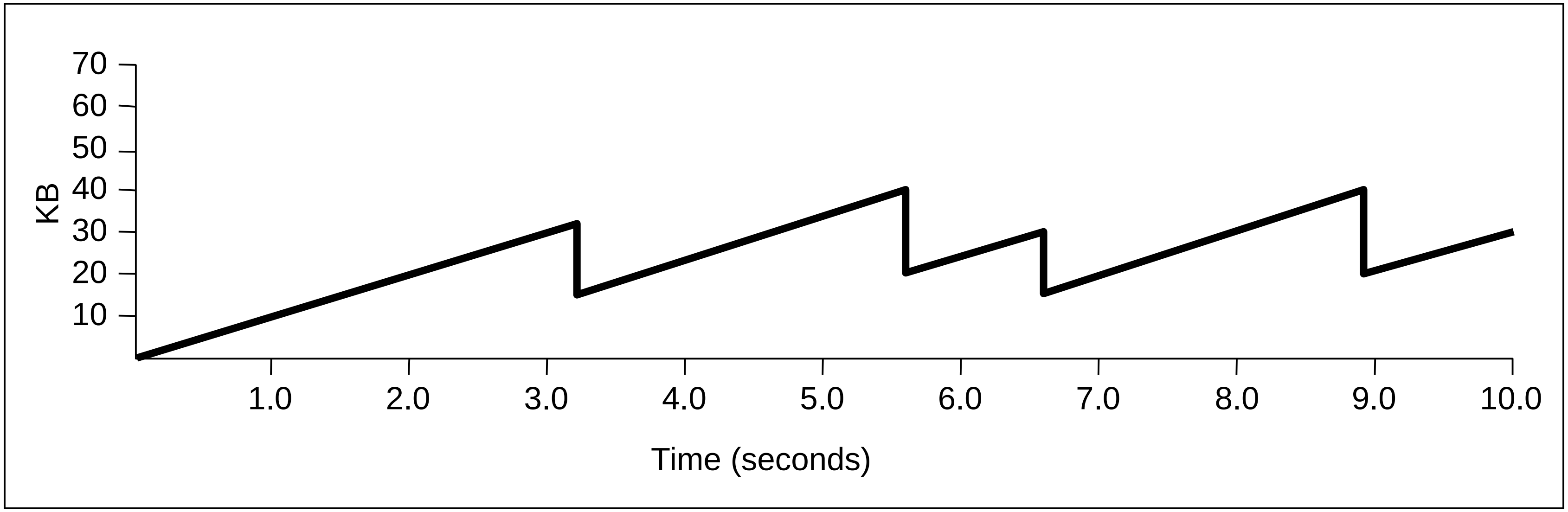
- Algorithm
  - » increases **`CongestionWindow`** by **`1`** segment
    - for each **RTT** (Round Trip Time) ➔ additive increase
  - » divide **`CongestionWindow`** by **2**
    - when there is a packet loss ➔ multiplicative decrease



- In practice,
  - » Increases by ACK received
  - » **`Increment= MSS * (MSS / CongestionWindow)`**
  - » **`CongestionWindow += Increment`**
  - » **`MSS → Maximum Segment Size`**

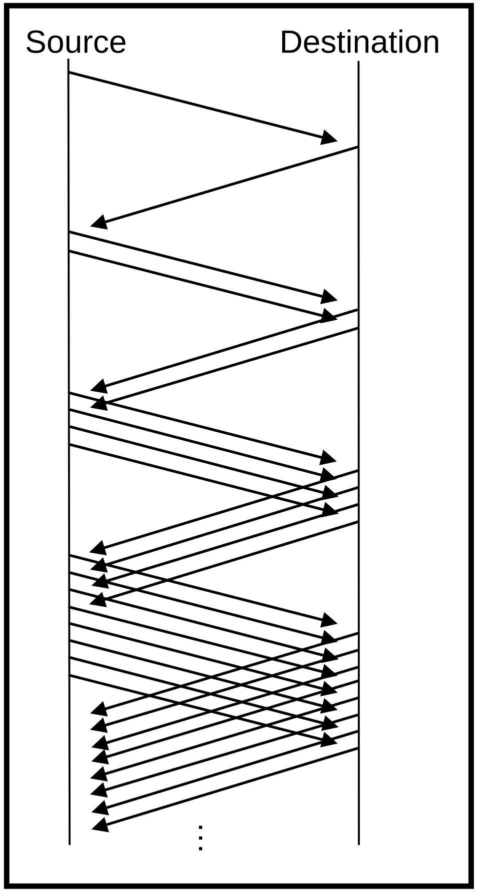
Additive Increase/Multiplicative Decrease

Saw-tooth behavior



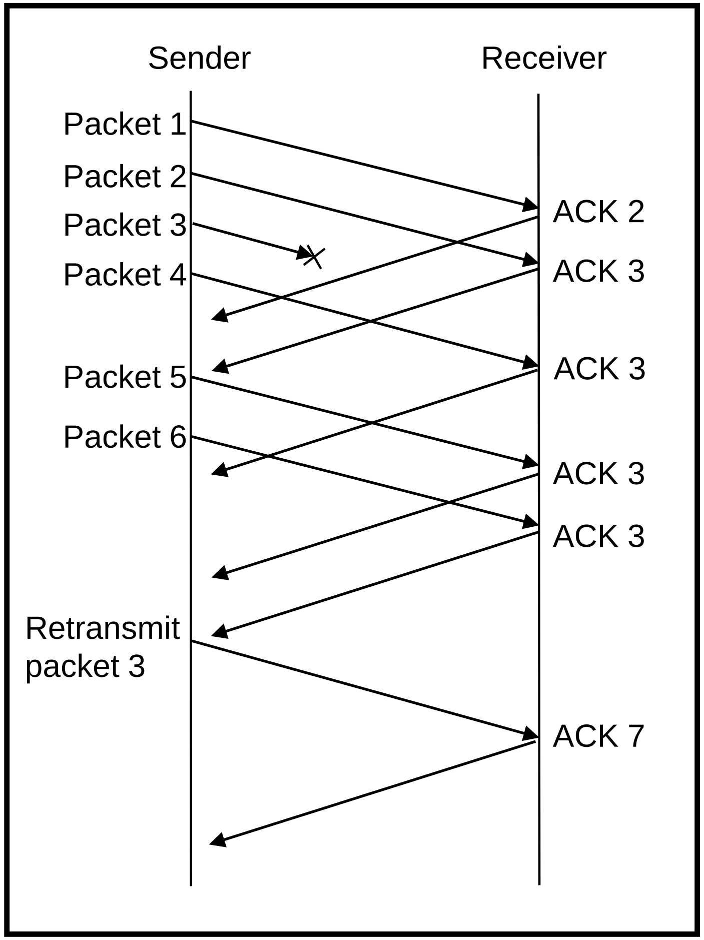
Slow Start ☺

- Objective
  - » determine the available capacity
- Behaviour
  - » start by **`CongestionWindow = 1 segment`**
  - » double **`CongestionWindow by each RTT`**



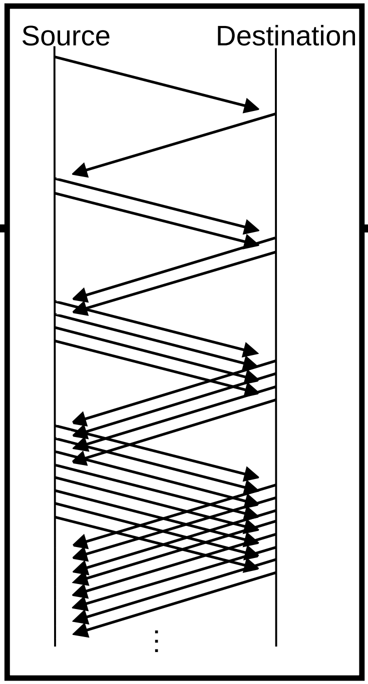
Fast Retransmission, Fast Recovery

- Problem
  - » if TCP timeout is large
    - → long inactivity period
- Solution
  - » fast retransmission
    - → after 3 repeated ACKs



TCP – Slow Start

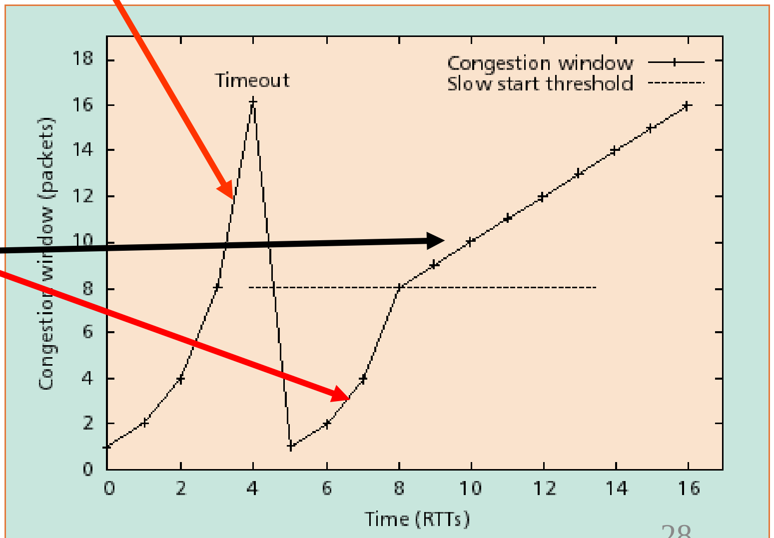
- ***Slow Start***
  - » Sender starts with **`CongestionWindow=1sgm`**
  - » Doubles **`CongestionWindow`** by **`RTT`**



- When a segment loss is detected, by timeout
  - » **`threshold = ½ congestionWindow(*)`**
  - » **`CongestionWindow=1 sgm`**

    (router gets time to empty queues)
  - » Lost packet is retransmitted
  - » ***Slow start*** while

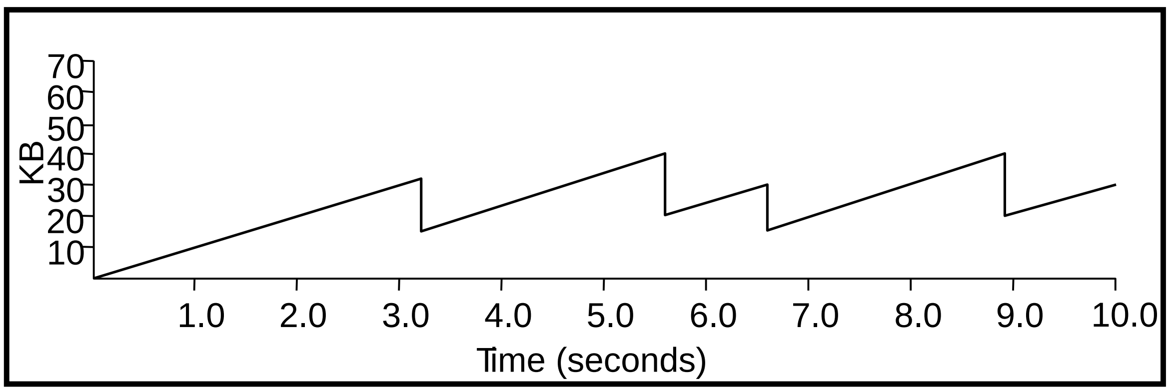
    `congWindow<threshold`
  - » Then → ***Congestion Avoidance*** phase

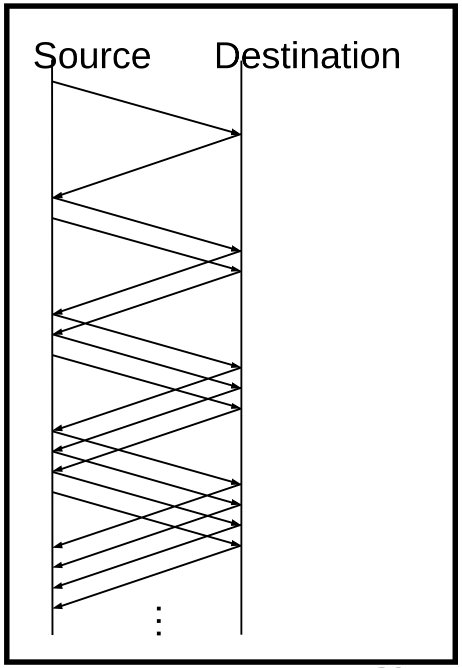


(*) - in fact FlightSize, the amount of outstanding data

Congestion Avoidance

- ***Congestion Avoidance*** (additive increase)
  - » increments **`congestionWindow`** by 1 sgm, per RTT
- Detection of segment loss, by reception of 3 duplicated ACKs
  - » Assumes packet is lost,
    - – Not by severe congestion, because following segments have arrived
  - » Retransmits lost packet
  - » **`CongestionWindow=CongestionWindow/2`**
  - » ***Congestion Avoidance*** phase



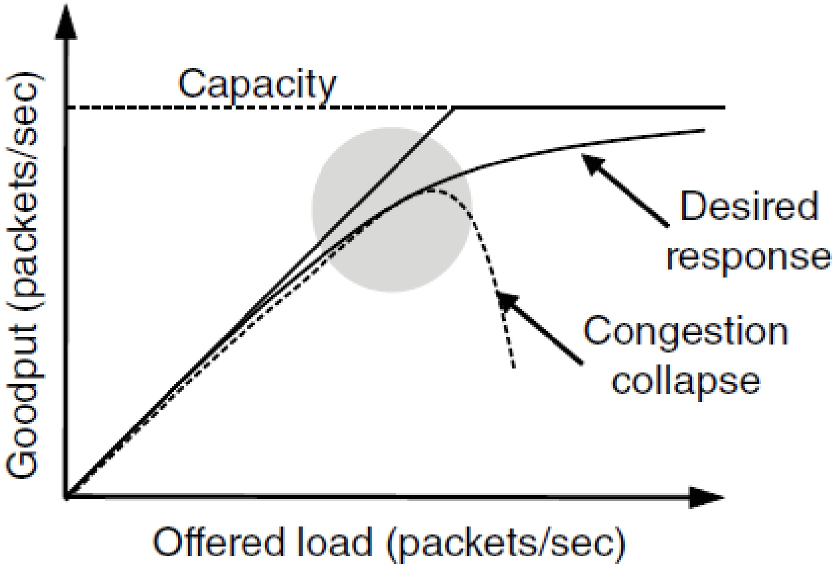


TCP – Congestion Control

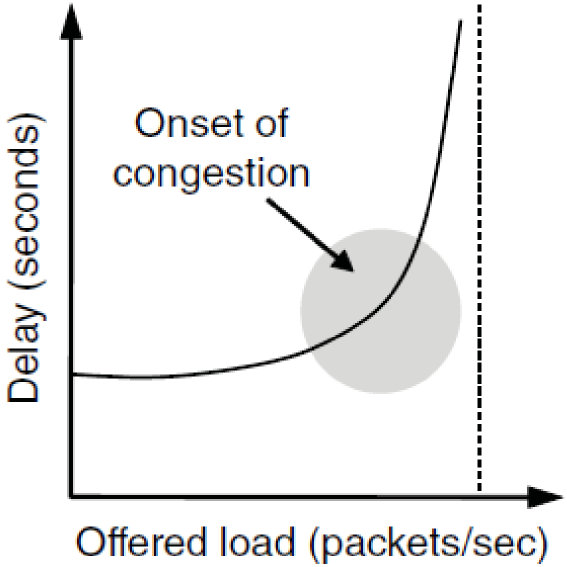
- In reality, a bit more complex
- RFC 2581, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”

Desirable Bandwidth Allocation

Efficient use of bandwidth gives high goodput, low delay



Goodput rises more slowly than load when congestion sets in

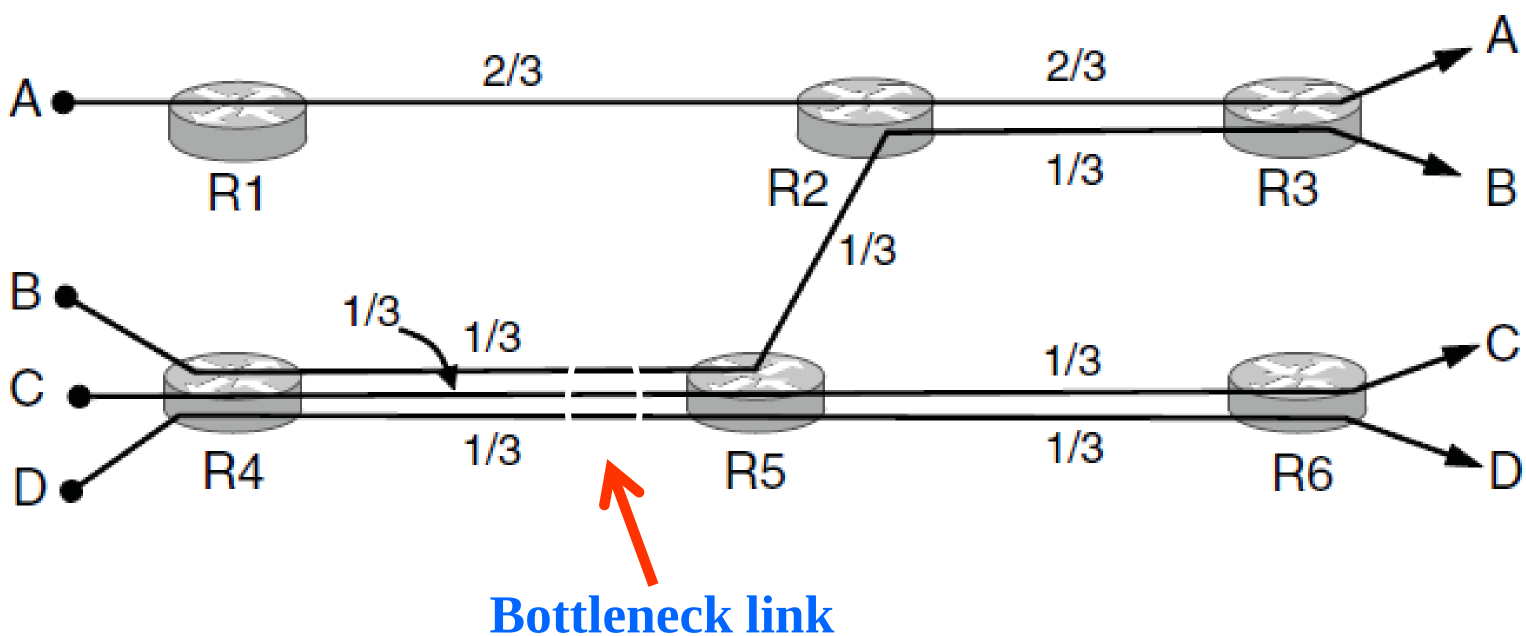


Delay begins to rise sharply when congestion sets in

Desirable Bandwidth Allocation – Max-min fairness

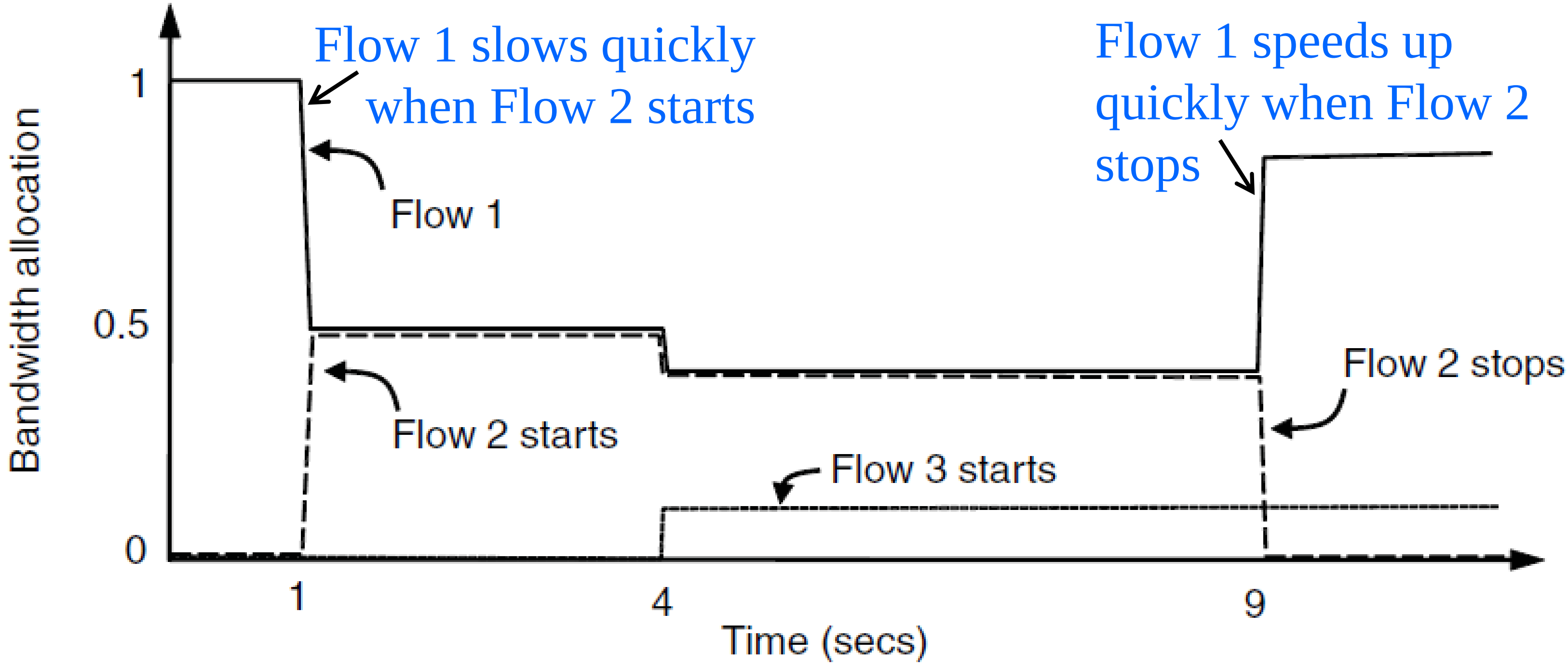
Fair use gives bandwidth to all flows (no starvation)

» **Max-min fairness** gives equal shares of bottleneck



Desirable Bandwidth Allocation – Bitrates along the time

Bitrates must converge quickly when traffic patterns change



Homework

1. Review slides

2. Read from Tanenbaum
   » Chapter 6 – The Transport Layer

3. Answer questions at moodle