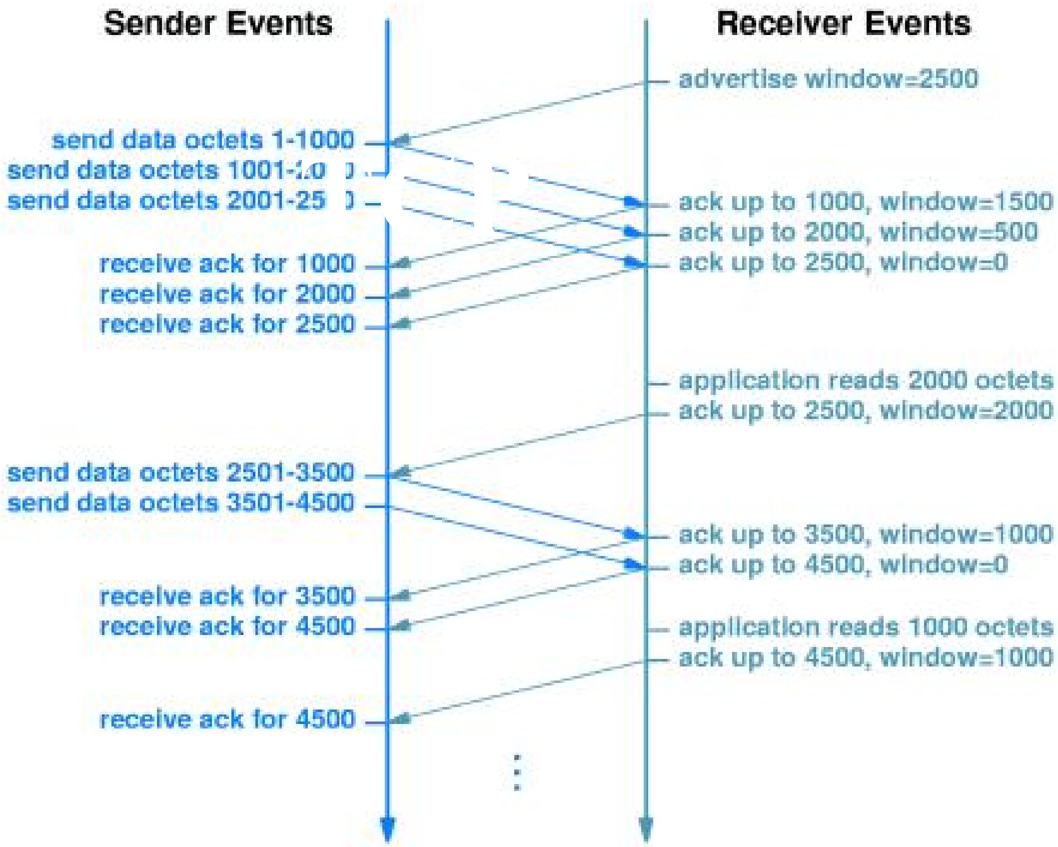
TCP



TCP flow control

Transport Protocols

Why?

Basic Features

Ports

UDP

TCP

Headers

TCP Connection

TCP Flow Control

Transport: Why?

- IP provides a weak, but efficient service model (*best-effort*)
  - Packet can be delayed, dropped, reordered, duplicated
  - Packets have limited size (why?)
- IP packets are addressed to a host
  - How to decide which application gets which packets?
- How should hosts send packets into the network?
  - Too fast is bad; too slow is not efficient

Transport: Basic Features

Can provide more reliability, in order delivery

Supports message of arbitrary length

Provides a way to decide which packets go to which applications (*multiplexing/demultiplexing*)

Governs when hosts should send data

Port, Socket and Socket pair

- Software defined port-to-port (end-to-end) process supported by OS
- Port
  - The point at which an application attaches to the network
  - Required due to multifunction of communication
  - Supports applications
  - Some preassigned, others available on demand

Port, Socket and Socket pair

- *Well known port* (0-1023): everyone agrees which services run on these port
  - e.g., ssh:22, telnet:23, SMTP:25, http:80
  - on UNIX, must be root to gain access to these ports (why?)
- ephemeral ports (most 1024-65535): given to clients
  - e.g., chatclient gets one of these

Port, Socket and Socket pair

Socket

- A specific combination of IP address and a port number

Socket pair

- 4-tuple consisting of the client IP address, client port number, server IP address and server port number
- Uniquely identifies each application level connection in Internet

UDP

- User Datagram Protocol
- Minimalistic transport protocol
- Same best-effort service as IP
- Messages of up to 64KB
- Provides multiplexing/demultiplexing to IP
- Does not provide connection control
- Advantage over TCP: does not increase end-to-end delay over IP
- Application example: video/audio streaming
- RTP implementations are built on the UDP

TCP

Transmission Control Protocol

Reliable, in-order delivery

Messages can be of arbitrary length

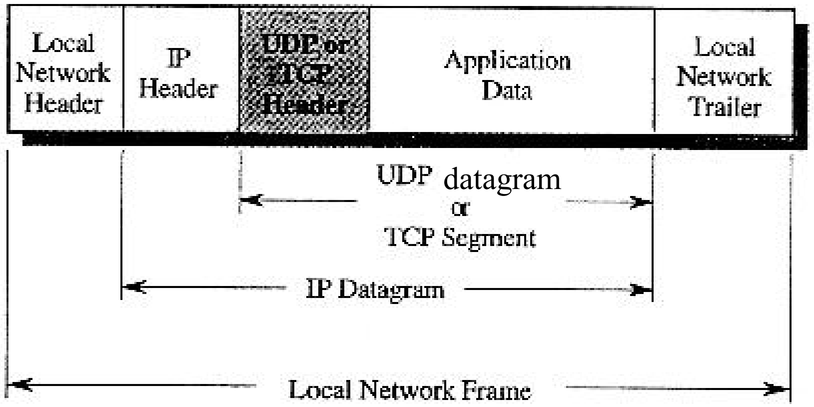
Provides multiplexing/demultiplexing to IP

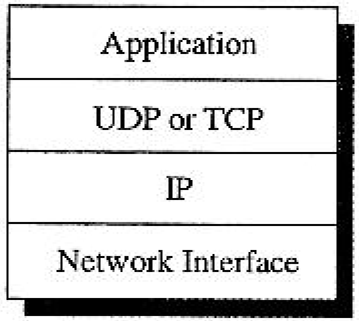
Provides congestion control and avoidance

Increases end-to-end delay over IP

Applications: file transfer, chat

TCP/UDP and IP Header Positions



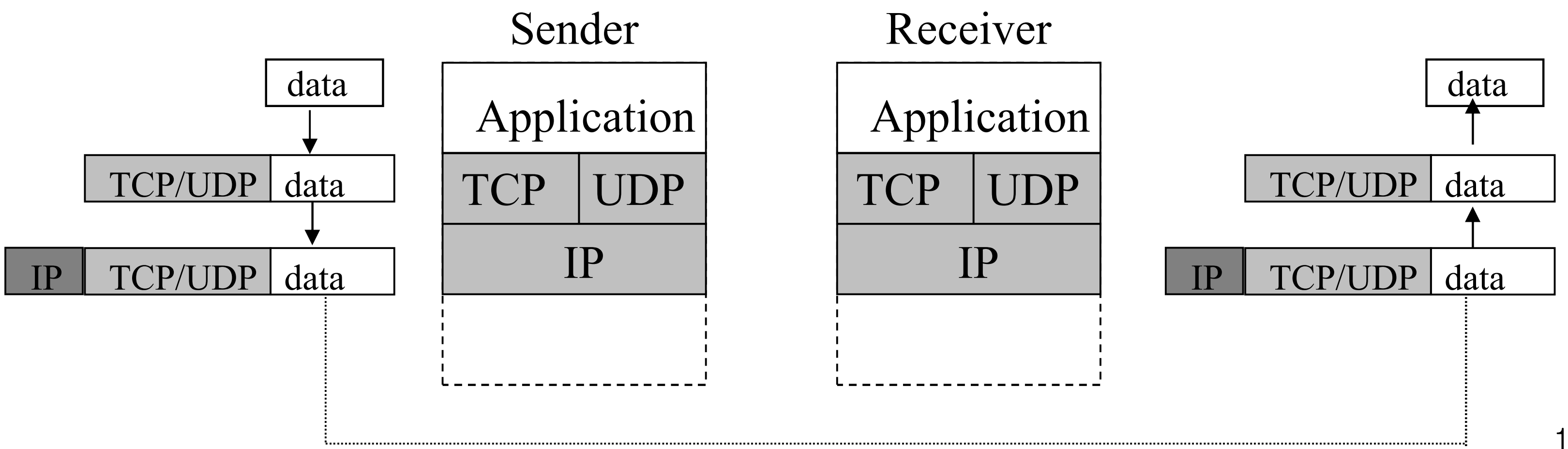


Header

IP header: used for IP routing, fragmentation, error detection

UDP header: used for multiplexing/demultiplexing, error detection

TCP header: used for multiplexing/demultiplexing, flow and congestion control



Transport: UDP

Service:

Send datagram from (IP1, Port 1) to (IP2, Port 2)

Service is unreliable, but error detection possible

Header:

0 — 16 — 31

Source port	Destination port
UDP length	UDP checksum
Payload (variable)	

UDP length is UDP packet length

(including UDP header and payload, but not IP header)

Optional UDP checksum is over UDP packet + pseudoheader

→ why have UDP checksum in addition to IP checksum?

→ why not have just the UDP checksum?

→ why is the UDP checksum optional?

Transport: TCP

Service

Header

Connections

3-Way Handshake

Sliding Window Flow Control

TCP: Service

Start a connection

Reliable byte stream delivery

From (IP1, TCP Port 1) to (IP2, TCP Port 2)

Indication if connection fails: Reset

Terminate connection

The TCP Header

0	4	10	16 … 31
Source Port			Destination Port
Sequence Number			
ACK Number			
Data Offset	Reserved (zero)	Flags	AdvertisedWindow
Checksum			Urgent Pointer
Options (if present), pad, data			

URG	ACK	PSH	RST	SYN	FIN

Flags Detail

TCP Header Fields

Source and Destination port: port numbers at sender/receiver of this segment

Sequence number: sequence number of the first data byte (each byte has sequence number) of this segment in transfer from sender to receiver

ACK number: next sequence number the sender of this segment expects to see from peer on reverse-direction data flow

TCP Header Fields

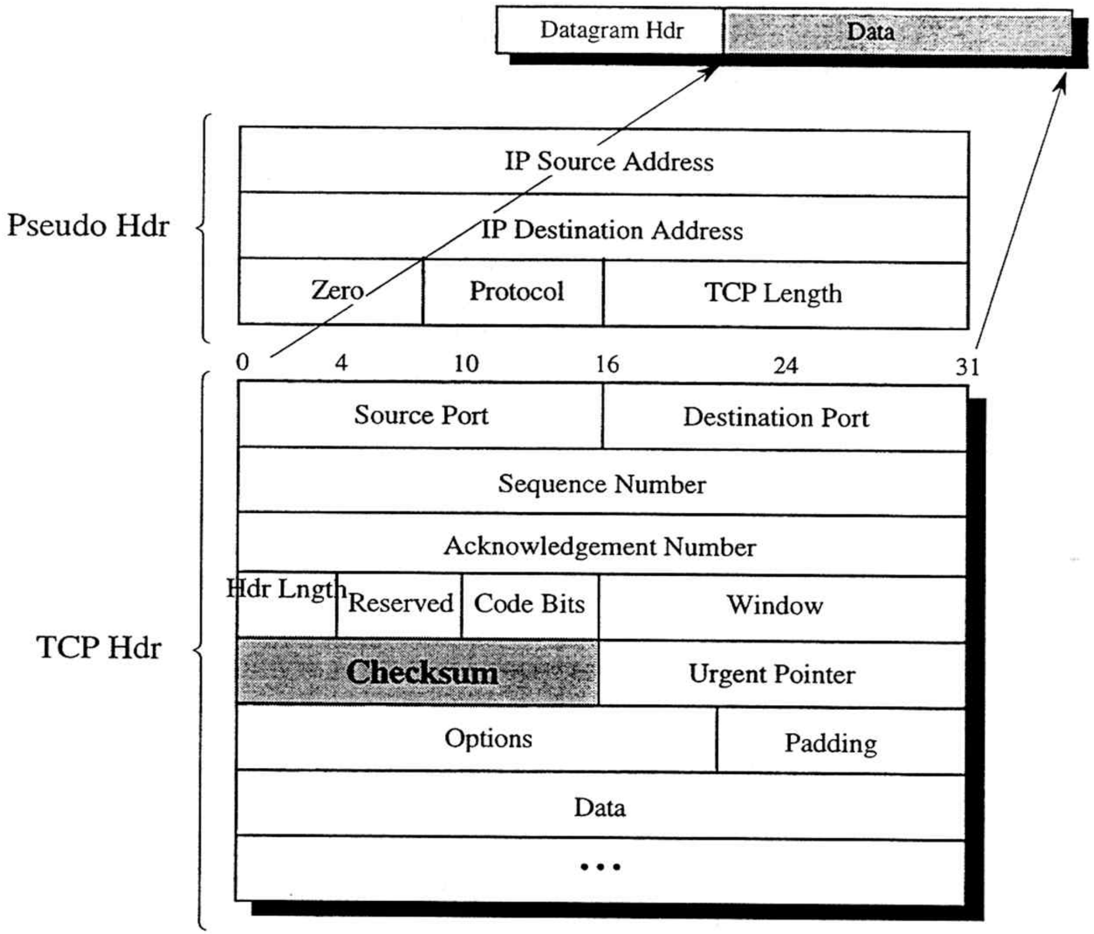
Data offset: number of 32-bit words comprising the TCP header (allows for variable-length header to contain TCP options; similar to the way IP works)

Window: buffer size in bytes available at the sender to receive data from peer on reverse-direction data flow

Checksum: pseudoheader checksum (required in TCP)

Urgent pointer: offset of last byte of urgent data

Pseudo Headers and the Checksum Field



TCP Header Fields - Pseudo Headers and Checksum Field

Before transmitting either a UDP or TCP header, a "pseudo header" is created that is placed on top of the UDP or TCP header as shown.

The Checksum computes its value for both the Pseudo Header and the TCP segment.

Just the TCP segment is transmitted.

The receiver extracts the IP Source and Destination Addresses from the IP datagram, rebuilds the Pseudo Header, and runs the Checksum.

Why? Great way to test reliable destination results.

TCP Header Fields

flags for control information:

- URG: segment contains urgent data
- ACK: ACK field is valid (normally "on")
- PSH: TCP push function: can be used to preserve the message boundary
- RST: abort current connection and reset connection immediately
- SYN: synchronize sequence numbers (open connection)
- FIN: close connection

TCP Connections

TCP is a connection-oriented transport protocol which runs on top of a connectionless datagram network layer

TCP connections are bi-directional

TCP connections do not preserve *message boundaries* (*byte-stream protocol*)

Sender application writes: 8 + 2 + 20

Receiver application reads: 5 + 5 + 5 + 5 + 5 + 5

A TCP connection consists of two connected TCP endpoints

TCP Connections

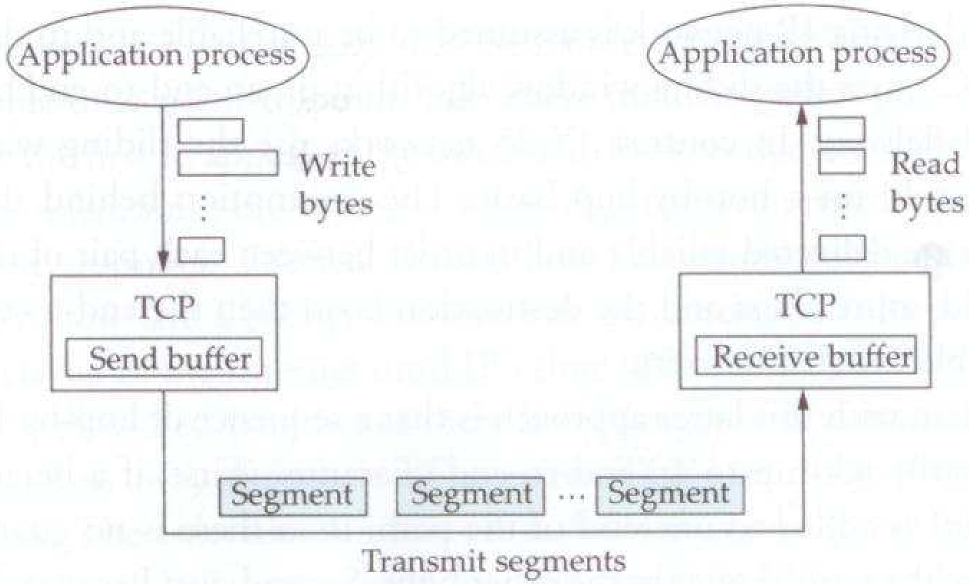


Figure 5.3 How TCP manages a byte stream.

TCP Connections

TCP uses port numbers similar to UDP, but the port space is disjoint from UDP

Each TCP endpoint is identified by:

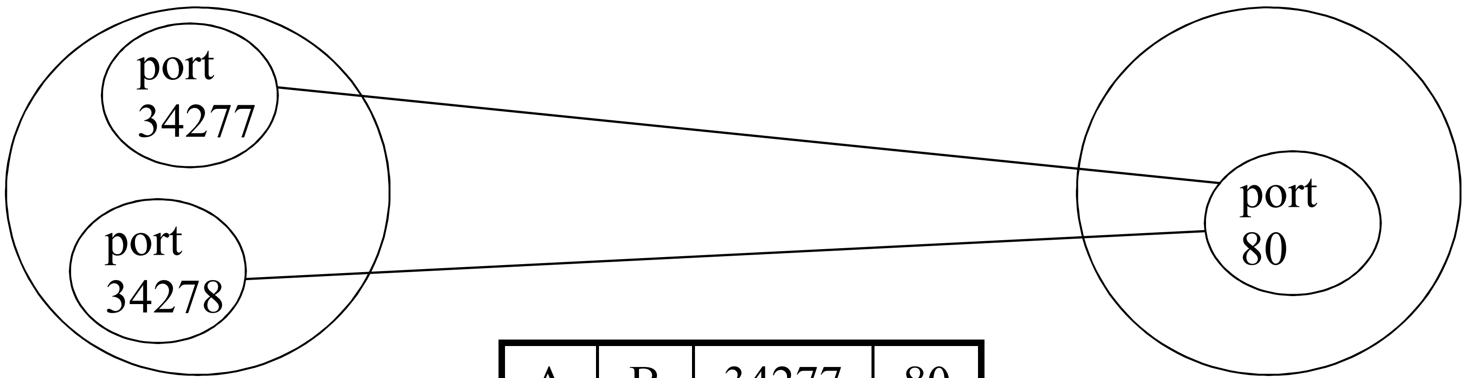
(IP address, port number)

So, an entire connection is identified by the 4-tuple:

(IP1, port1; IP2, port2)

TCP Connections

TCP connection 4-tuple provides the ability for a server to provide service to multiple clients:



IP Address A

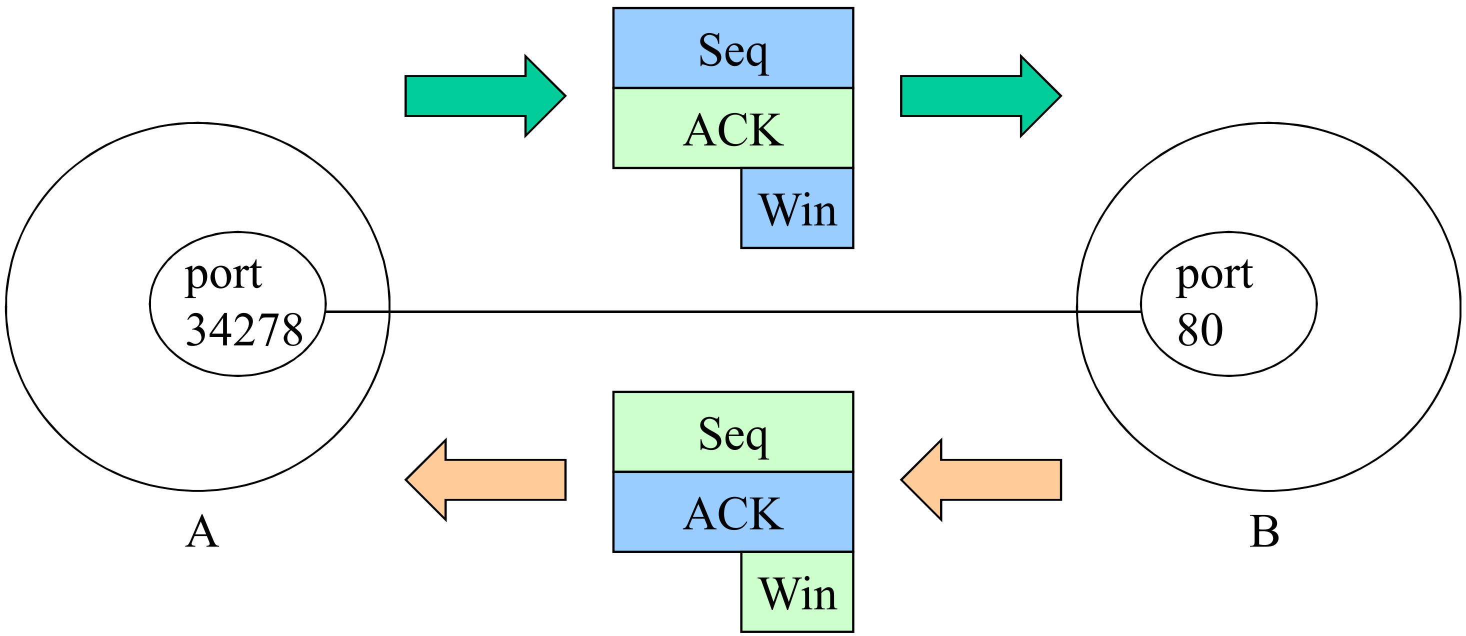
IP Address B

A	B	34277	80
A	B	34278	80

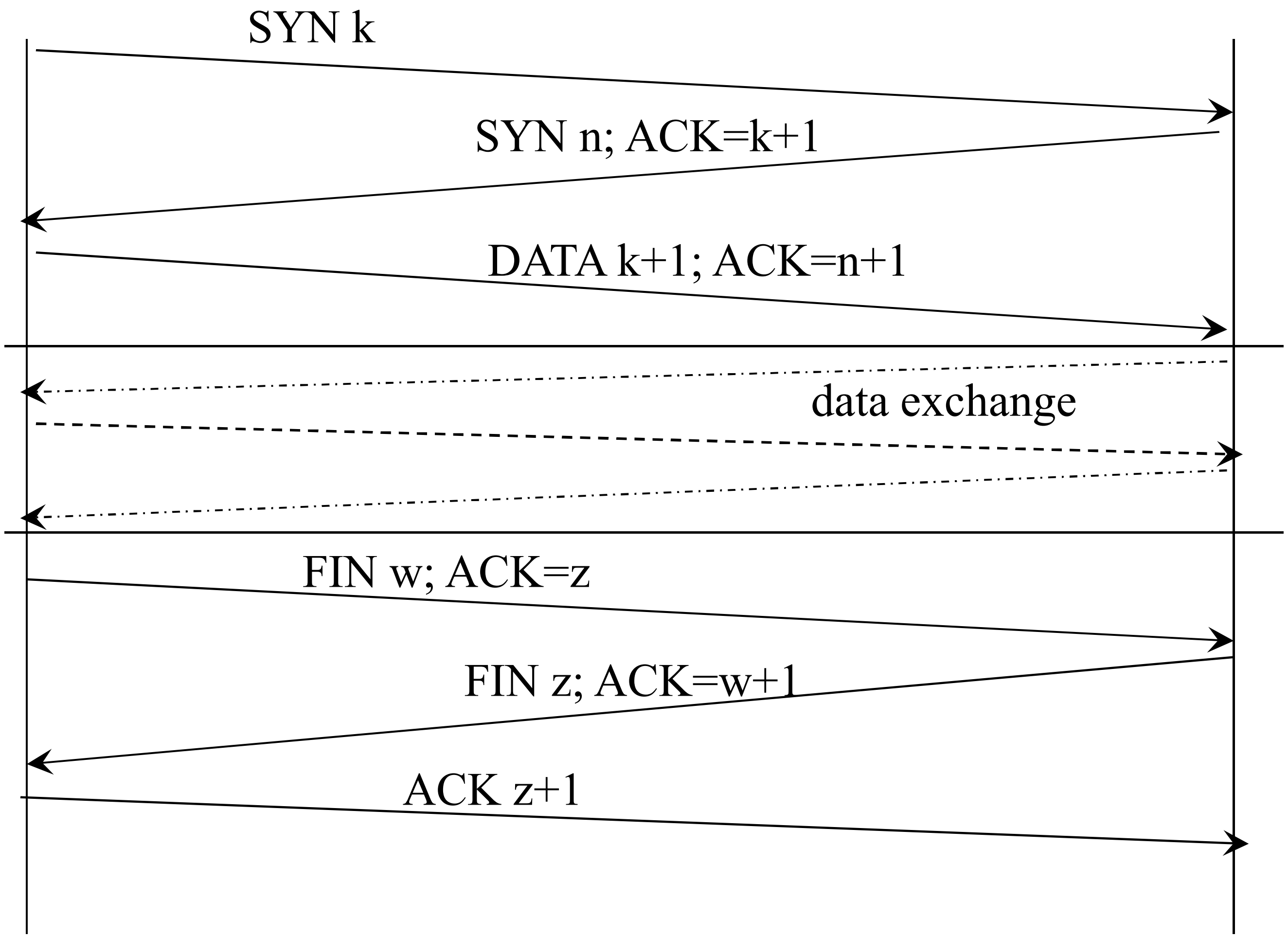
Connections

Steady State Operation

Typical operation looks like this:



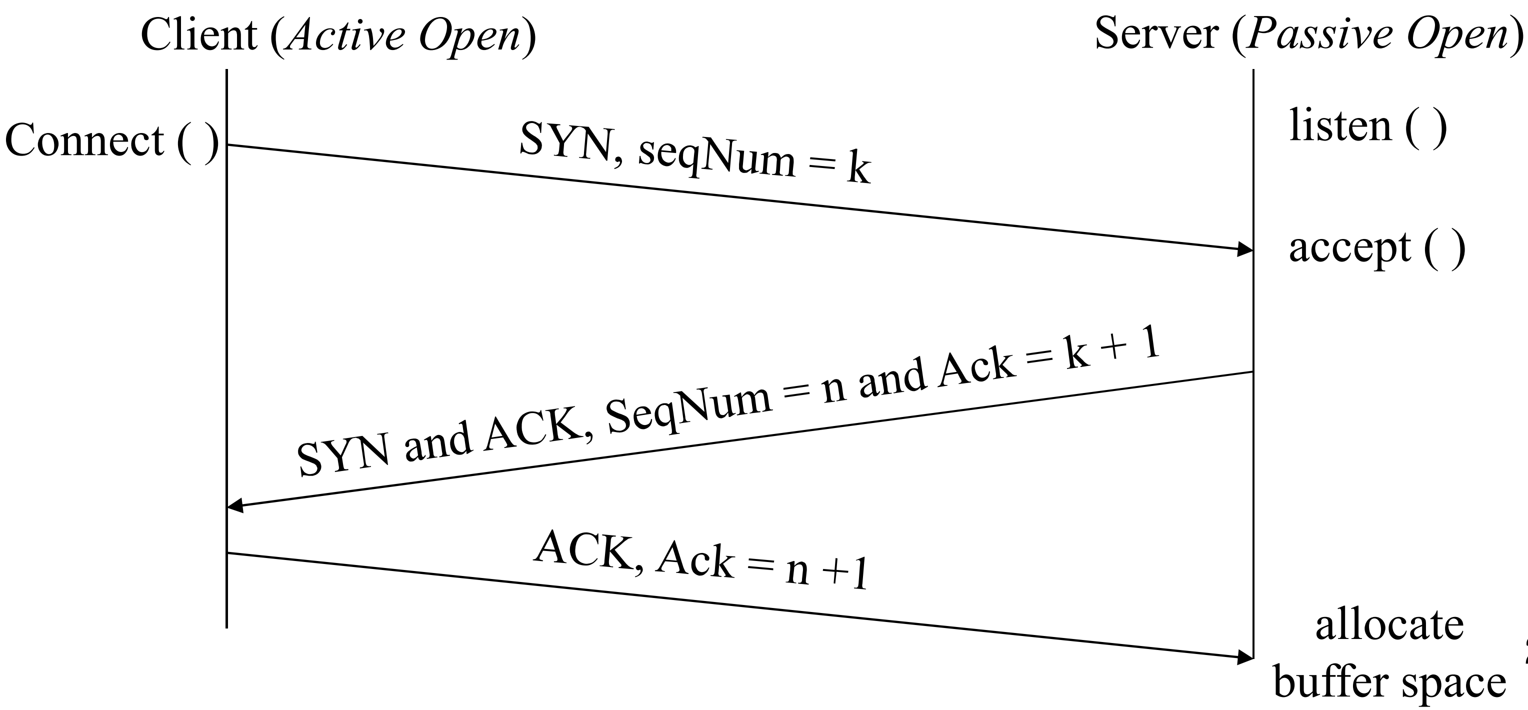
TCP: Steps/3-way handshake



3WH: Description

Goal: agree on a set of parameters: the start sequence number for each side

Starting sequence numbers are random



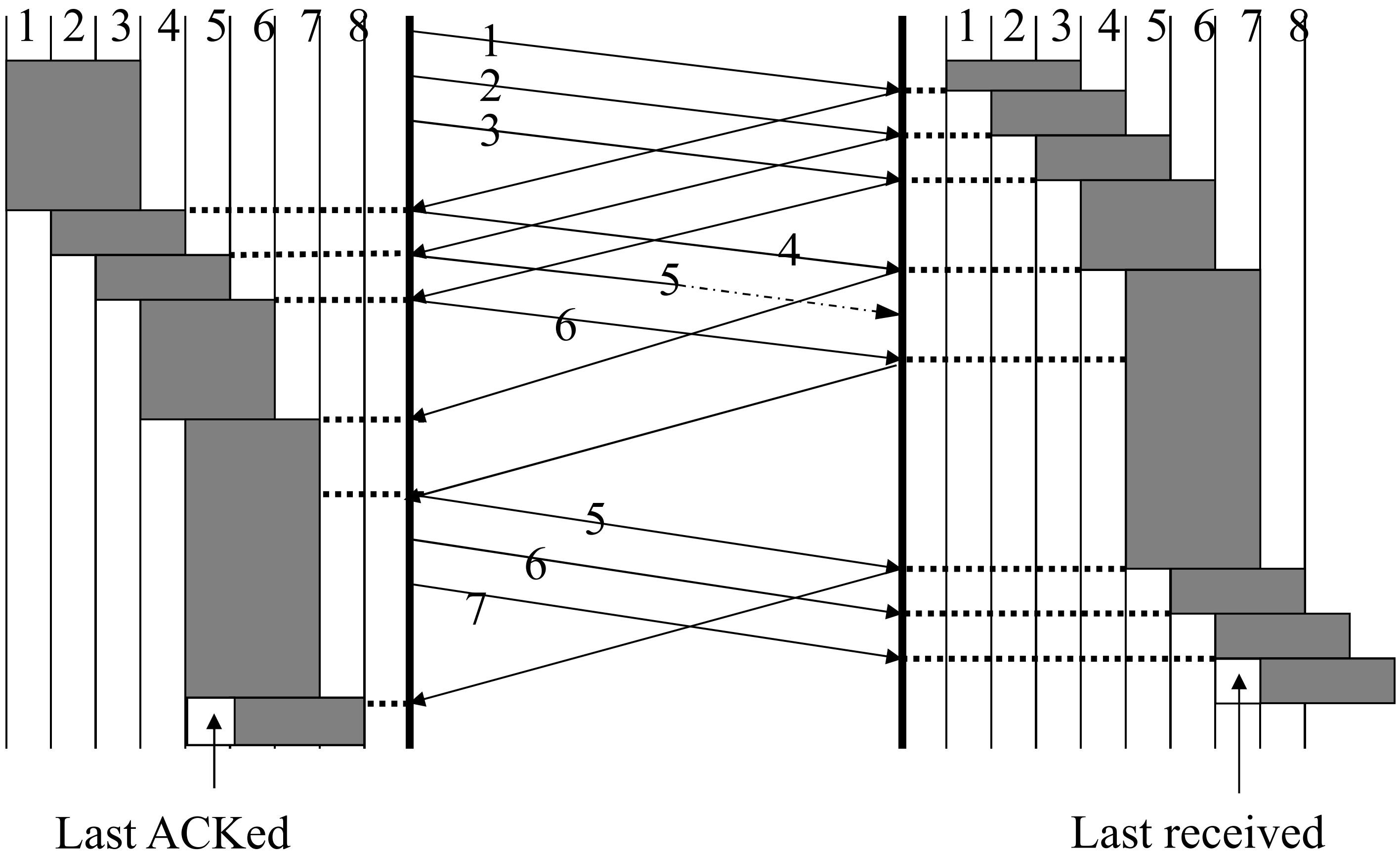
3WH: Rationale

Three-way handshake adds 1 RTT delay

Why?

- Congestion control: SYN (20+20 byte) acts as cheap probe
- Protects against delayed packets from other connection (would confuse receiver)

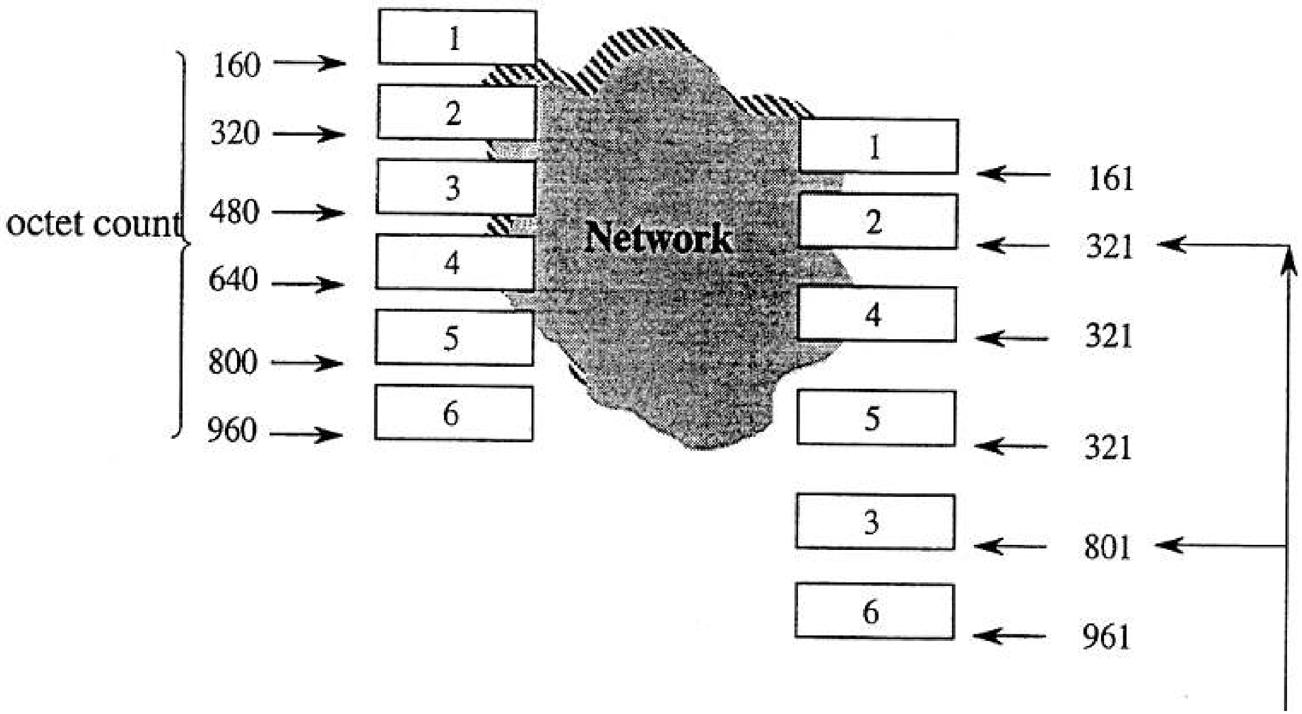
Sliding Window Flow Control: Go-Back-n



TCP Ack

Transmit Site
At Launch Time

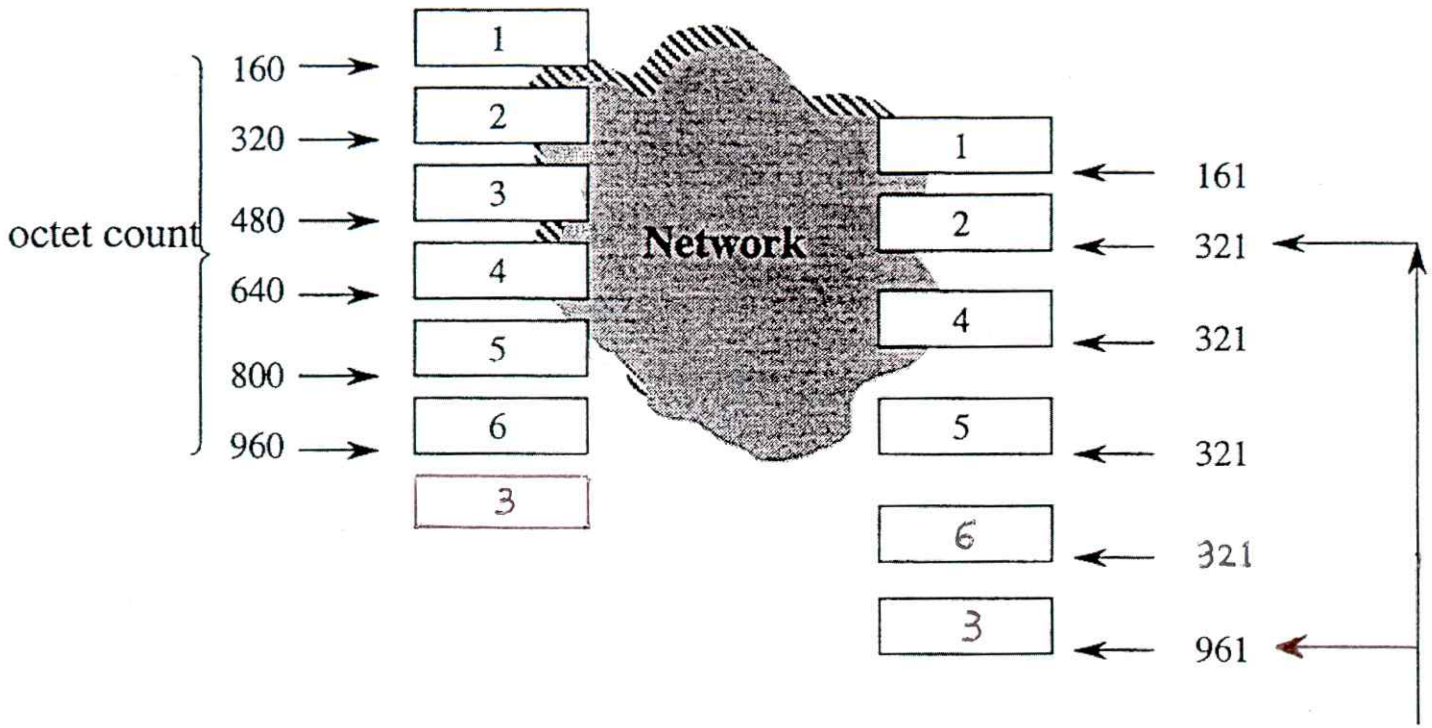
Receive Site
Some Seconds Later



TCP Ack and Fast Retransmissions

Transmit Site at Launch Time

Receive Site Some Seconds Later



TCP Acknowledgement and Retransmissions

TCP transmits byte oriented streams of binary messages.

The receiver collects data bytes from arriving segments and reconstructs an exact copy of the stream being sent. Remember, segments can be **damaged**, **lost** and/or arrive out of order.

Cumulative Ack: The receiver always acknowledges the longest contiguous prefix of the stream that has been received correctly.

Acknowledgements always specify the sequence number of the next byte that the receiver expects to receive.

The receiver uses the sequence numbers to order retransmission.

Timer associated with each segment: If the timer expires before the segment is Acked, the sender must retransmit.

TCP Summary

TCP provides reliable, in order delivery

TCP connection with (IP1, port1; IP2, port2)

Socket pair uniquely identifies each application level connection in Internet

Sliding window based flow control:

- SeqNum, Ack, AdvWindow
- Cumulative Ack
- Retransmit time expired segment