Chapter 3 outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

- segment structure
- reliable data transfer
- flow control
- connection management

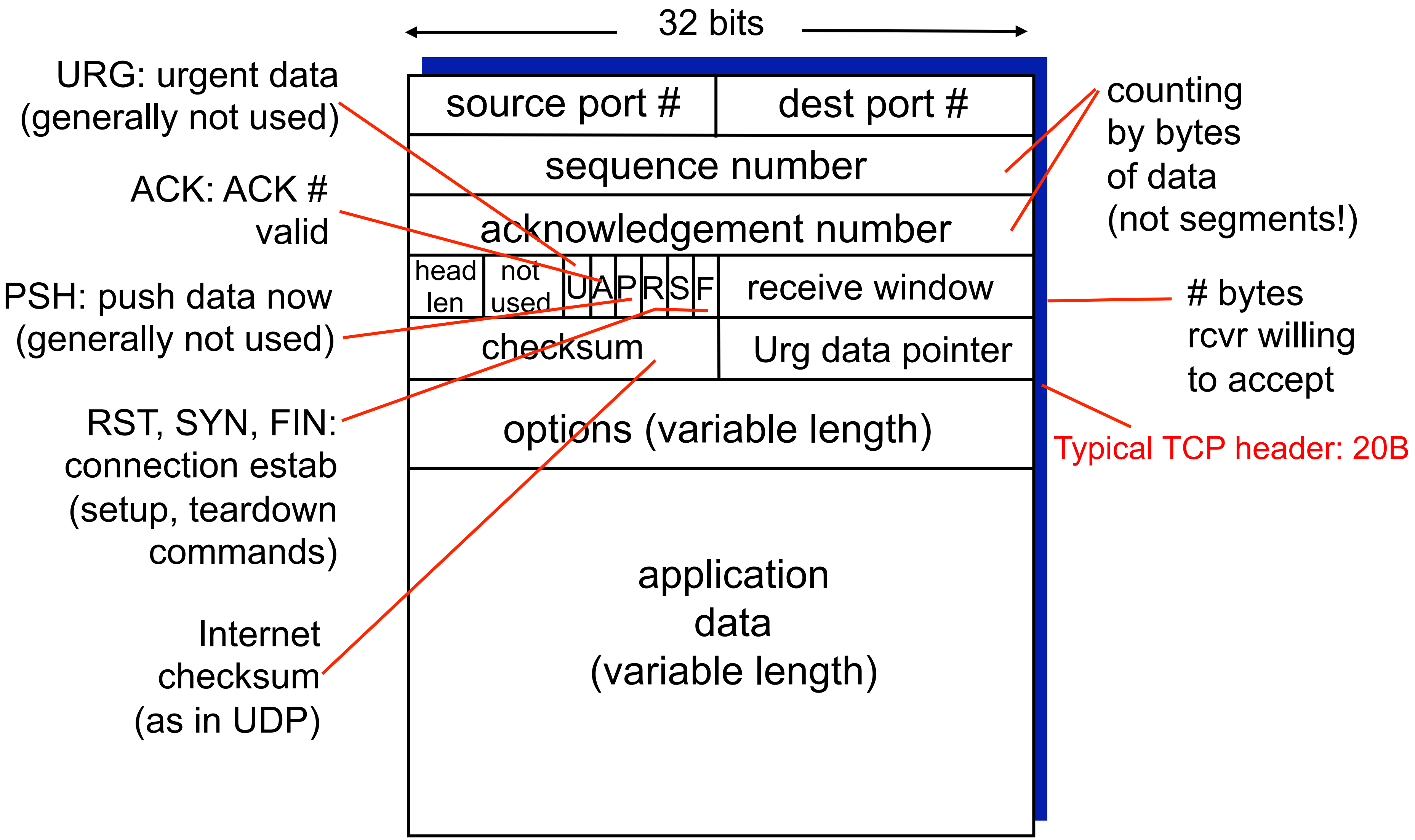
3.6 principles of congestion control

3.7 TCP congestion control

TCP: Overview RFCs: 793,1122,1323, 2018, 2581

- point-to-point:
  - one sender, one receiver
- "connection"-oriented:
  - handshaking (exchange of control msgs) inits sender, receiver state before data exchange
  - not a circuit
- **TCP views data as an unstructured, ordered, byte stream**
  - delivers these bytes reliably and in order"
  - no "message boundaries"
- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- pipelined:
  - TCP congestion and flow control set window size
- flow controlled:
  - sender will not overwhelm receiver

TCP segment structure



32 bits

source port # | dest port #

sequence number

acknowledgement number

head len | not used | U | A | P | R | S | F | receive window

checksum | Urg data pointer

options (variable length)

application data (variable length)

URG: urgent data (generally not used)

ACK: ACK # valid

PSH: push data now (generally not used)

RST, SYN, FIN: connection estab (setup, teardown commands)

Internet checksum (as in UDP)

counting by bytes of data (not segments!)

# bytes rcvr willing to accept

Typical TCP header: 20B

TCP seq. numbers, ACKs

sequence numbers:

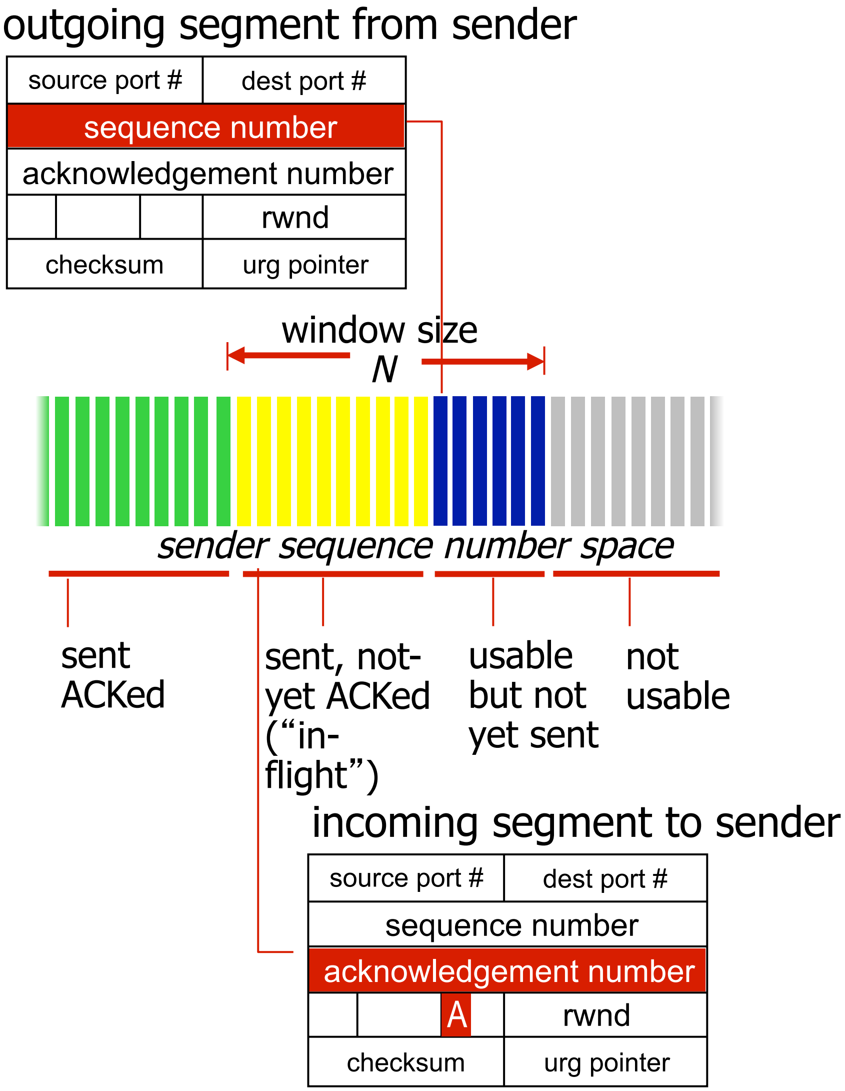
- byte stream "number" of first byte in segment's data
- 32-bit seq
- randomly chosen upon initialization (not 0!)
- one per direction

acknowledgements:

- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments

- A: TCP spec doesn't say, - up to implementor
- In practice: typically buffers them and wait to fill up gaps.



outgoing segment from sender

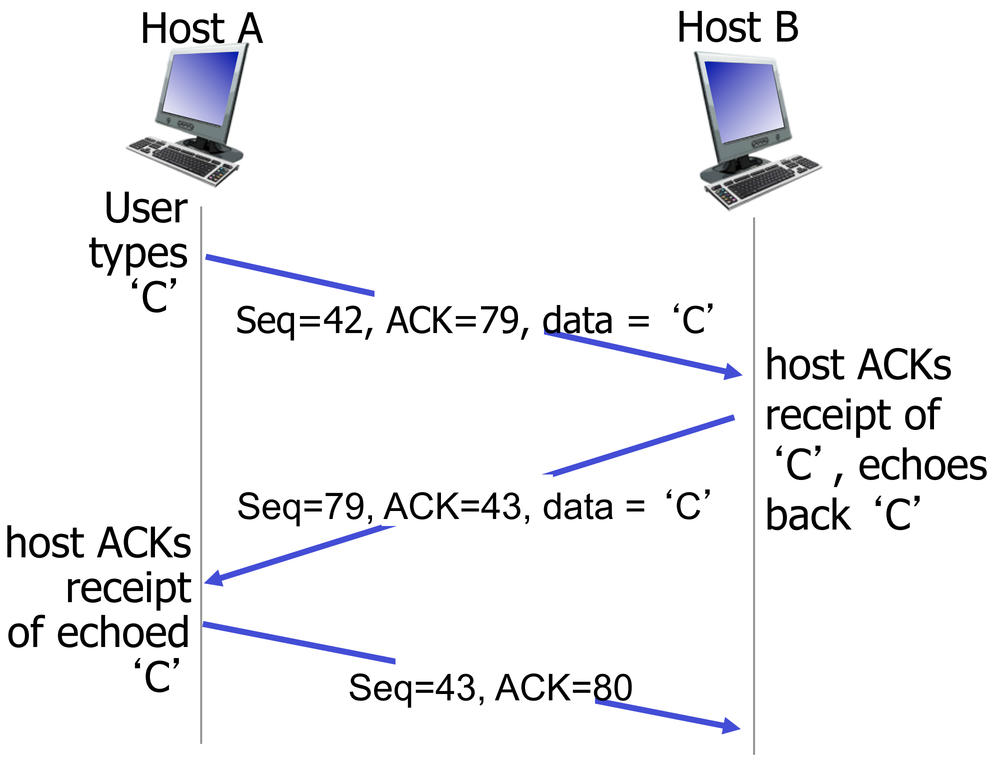
source port #	dest port #
sequence number	
acknowledgement number	
	rwnd
checksum	urg pointer

window size
N
sender sequence number space
sent ACKed
sent, not-yet ACKed ("in-flight")
usable but not yet sent
not usable

incoming segment to sender

source port #	dest port #
sequence number	
acknowledgement number	
A	rwnd
checksum	urg pointer

TCP seq. numbers, ACKs


Host A
Host B
User types 'C'
Seq=42, ACK=79, data = 'C'
host ACKs receipt of 'C', echoes back 'C'
Seq=79, ACK=43, data = 'C'
host ACKs receipt of echoed 'C'
Seq=43, ACK=80

simple telnet application scenario

ACK "piggybacked" on the server-to-client data packet