CS 372 Lecture #18

Reliable data transfer with TCP

- efficiency
  - stop-and-wait
  - pipelining
  - sliding-window

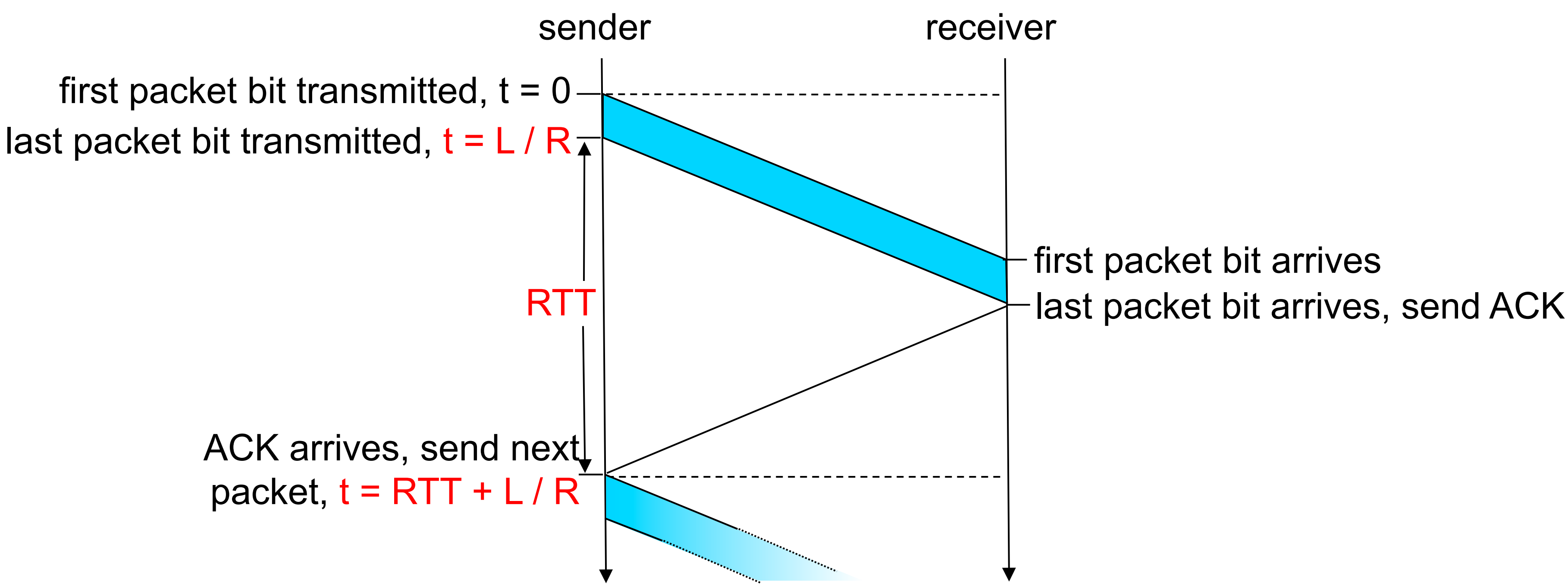
Note: Many of the lecture slides are based on presentations that accompany *Computer Networking: A Top Down Approach,* 6th edition, by Jim Kurose & Keith Ross, Addison-Wesley, 2013.

Efficiency considerations

- TCP's acknowledgement model is great for reliability …
- … but with what tradeoffs?
  - – Segment overhead
  - – Each packet requires at least one RTT
    - send segment, wait for ACK
  - – May increase queuing delay
  - – Increases network congestion

Performance of "stop-and-wait"

- Example: R=1 Gbps, 15 ms end-to-end propagation delay, L=1000Byte packet:

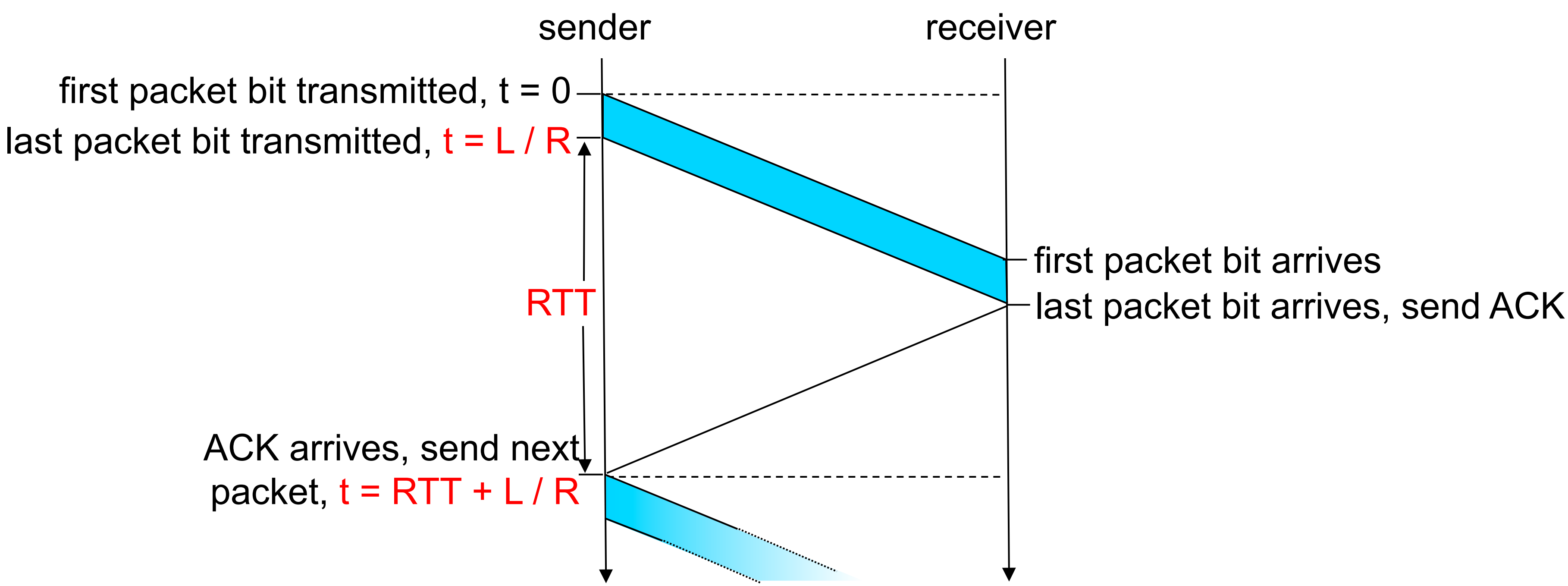


$$\text{transmission delay} = \frac{L}{R} = \frac{8\text{x}10^3 \text{ bits}}{10^9 \text{ bps}} = 0.008 \text{ ms}$$

round-trip time (RTT) = 30 ms

Performance of "stop-and-wait"

- Example: R=1 Gbps, 15 ms end-to-end propagation delay, L=1000Byte packet:

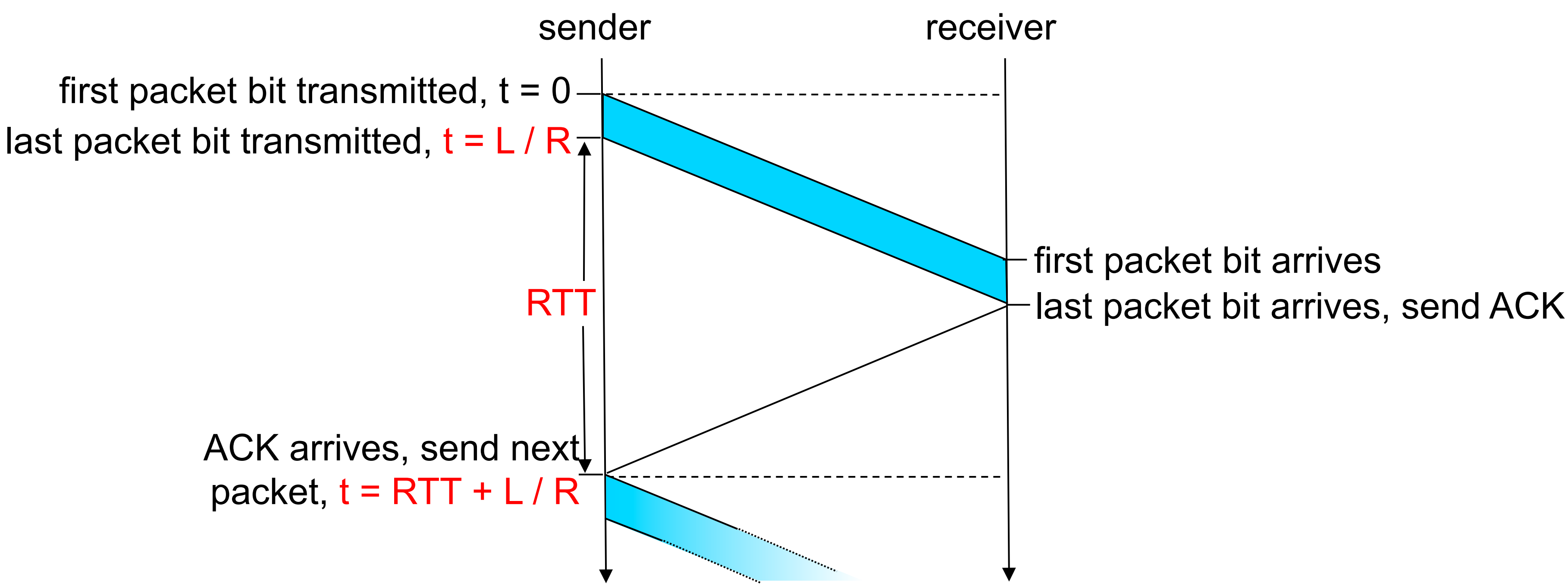


U_{sender}: utilization = fraction of time sender is busy sending

$$U_{sender} = \frac{L / R}{RTT + L / R} = \frac{.008}{30.008} = 0.00027$$

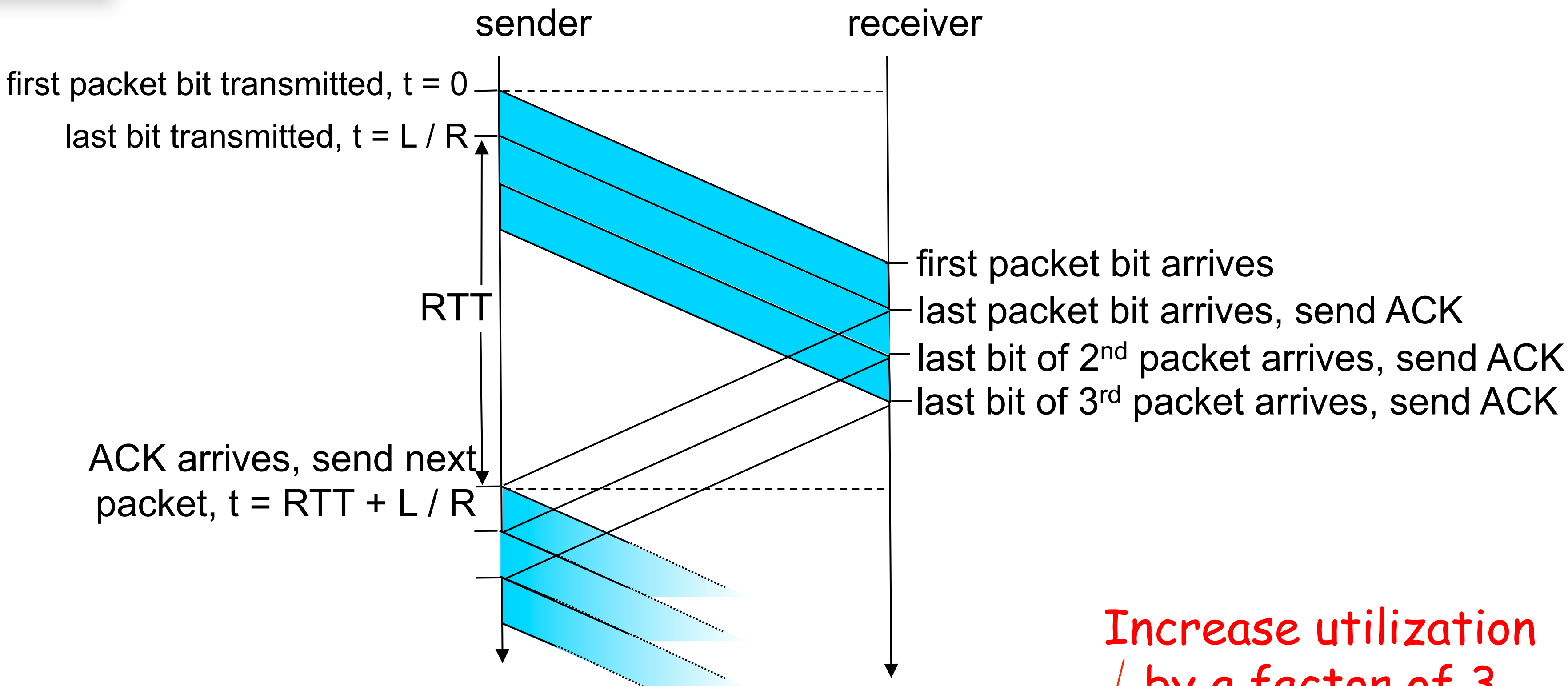
Performance of "stop-and-wait"

- Example: R=1 Gbps, 15 ms end-to-end propagation delay, L=1000Byte packet:



- 1KB packet every 30 msec -> 33Kbps throughput over **1 Gbps** link !!!
- "stop-and-wait" protocol limits use of physical resources.

Pipelining: increased utilization

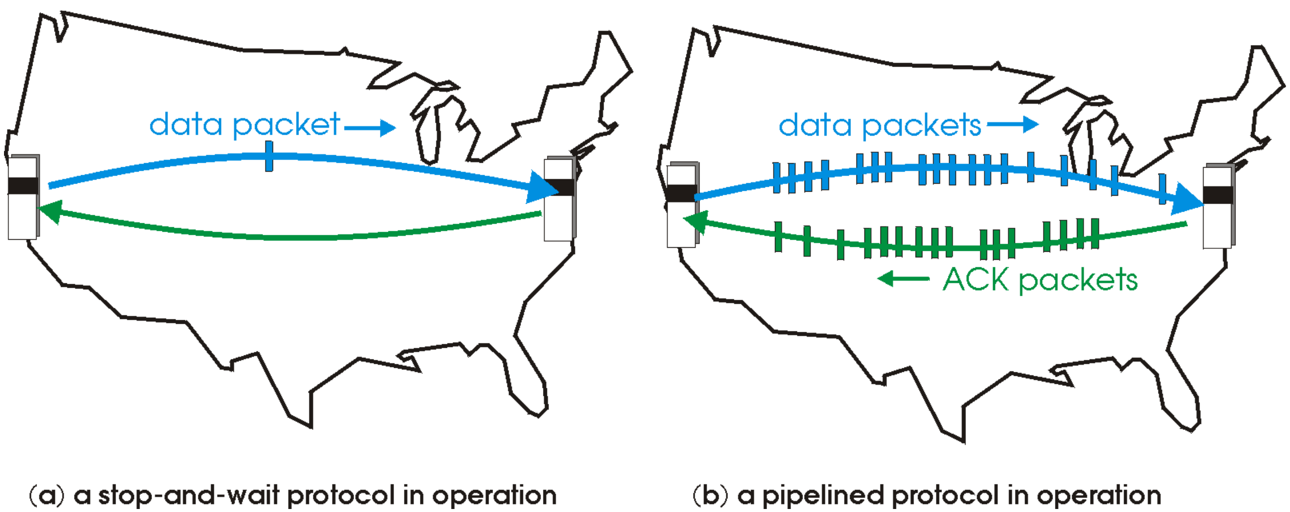


Increase utilization by a factor of 3

$$U_{sender} = \frac{3 * L / R}{RTT + L / R} = \frac{.024}{30.008} = 0.0008$$

Pipelined protocols

Pipelining: Sender transmits multiple packets.
Packets "in-flight" have yet to be acknowledged

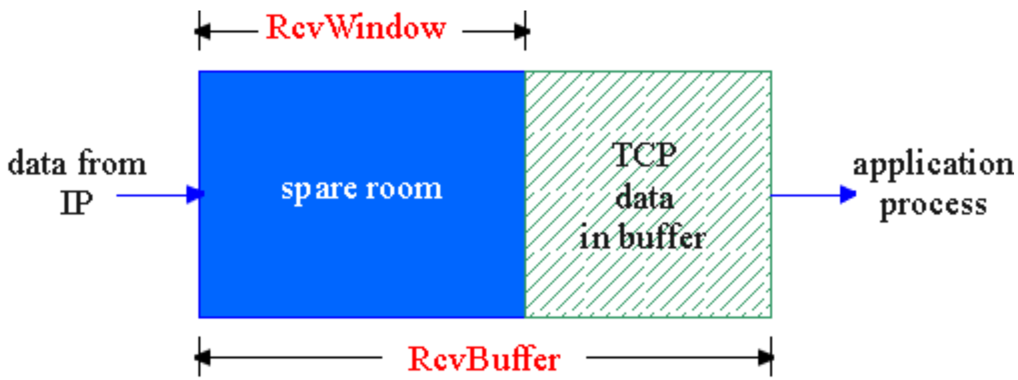


Problem: receiver might not be able to handle that many packets as fast as they arrive.

Problem: packets might arrive out-of-order

TCP flow control

- receive side of TCP connection has a receive buffer:



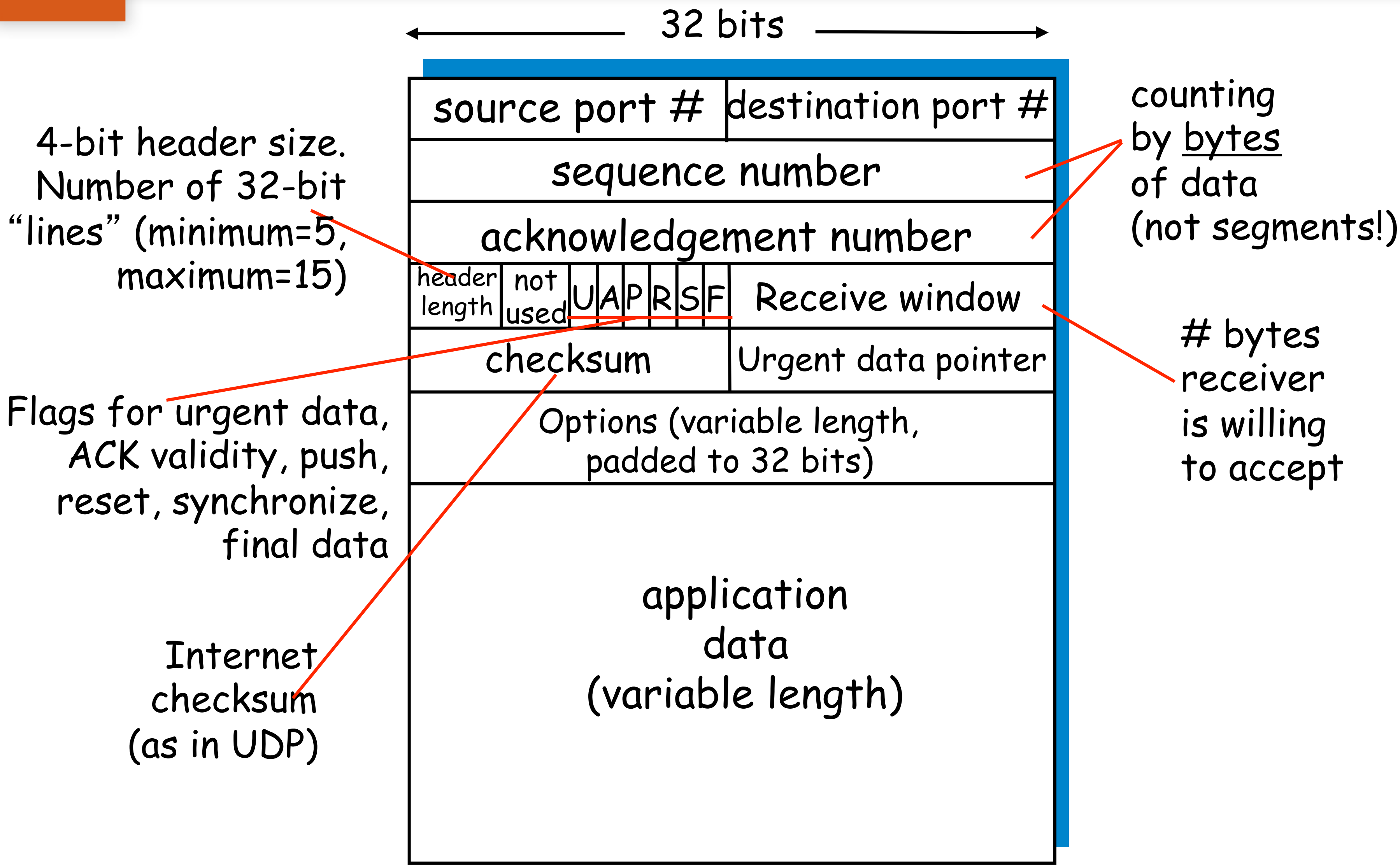
- receiver application layer process reading from buffer may be slow

flow control

sender won't overflow receiver's buffer by transmitting too much, too fast

- speed-matching service: matching the send rate to the receiving application's drain rate
- See animations on textbook's website.

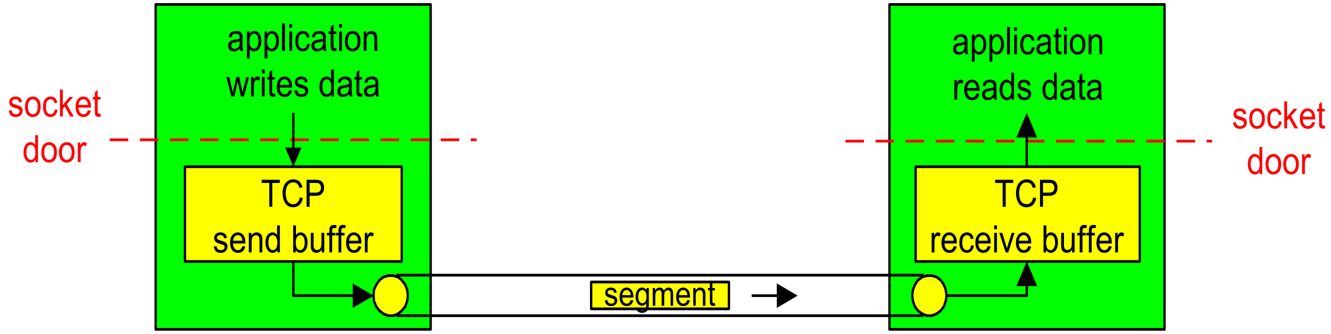
TCP segment structure



TCP flow control

- TCP uses sliding window for flow control

1. Sender transmits one segment
2. Receiver specifies window size (*window advertisement*) in ACK header "receive window"
   – Specifies how many *bytes* in the data stream can be sent
3. Sender limits unACKed data to "receive window"
   – guarantees receive buffer doesn't overflow



Summary Lecture #18

- Efficiency
  - stop-and-wait
  - pipelining
- flow control
- “receive window”
- sliding-window protocol

- Next hurdle:
  - What happens if there are errors in the pipeline?