IOWA

CS3640

Transport Layer (3): Congestion Control

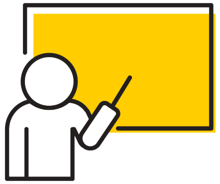
Prof. Supreeth Shastri
Computer Science
The University of Iowa

Midterm Format and Question

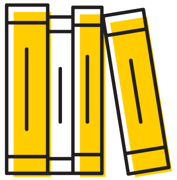
Category	**Example questions and topics**	**Weight**
Networking **Principles**	*End-to-end argument; Routing and forwarding; Protocol layering*	25%
Networking **Protocols**	*TCP vs. UDP; HTTP headers and extensions; Designing CDNs*	25%
Networking **Problems**	*Understanding delays; Mitigating congestion; Security challenges*	25%
Network **Programs**	*Explain how traceroute works; Socket programming; Video Streaming*	25%

*There will be an optional **bonus question** carrying 10% extra points (expect it to be challenging)*

Preparations and logistics



*Revisit the **lectures and slides**:*
https://shastri.info/teaching/cs3640



*Read the **textbook**:*
Kurose-Ross chapters 1-3

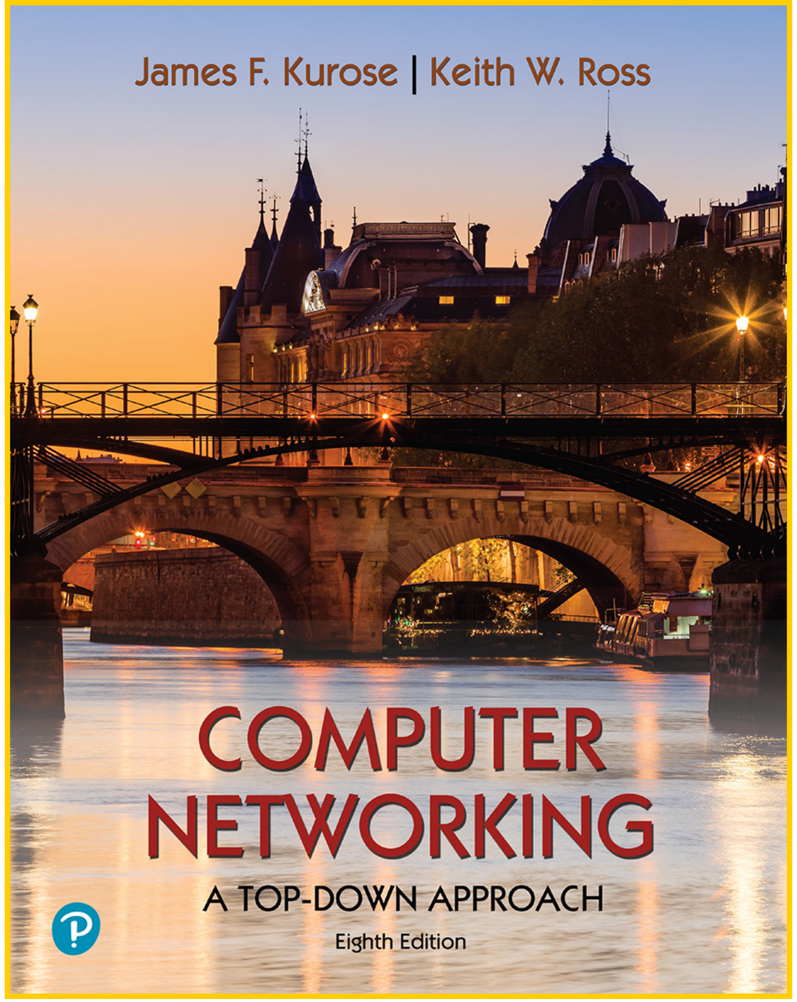
*Midterm **schedule**:*
*3/10 Thursday at 6:30PM in **3655 SC***

It is a 1-hour pen-and-paper exam (closed book, closed notes, closed electronics)

Lecture goals

continued discussion of reliable data transfer, followed by congestion control

- *Pipelined RDT protocols*
- *Congestion control*



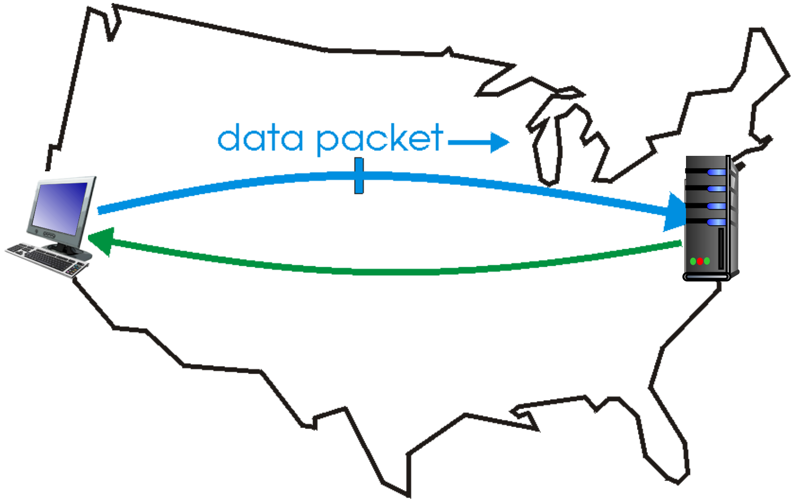
Chapter 3.4, 3.6

IOWA

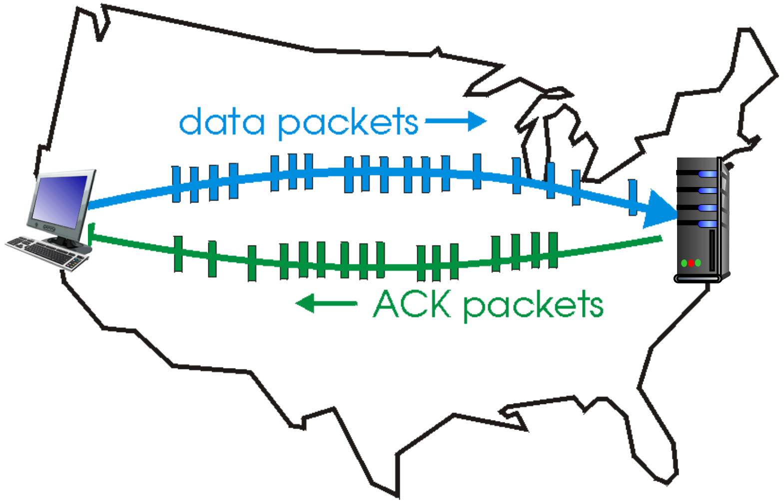
Reliable Data Transfer: techniques and mechanisms

Checksum	detect bit errors in packet
ACK	report reception of a packet correctly
NAK	report error(s) in a received packet
Sequence numbers	detect any missing packets
Timers	detect and recover from packet loss
Pipelining	increase channel utilization

Increased Utilization w/ Pipelining



(a) a stop-and-wait protocol in operation

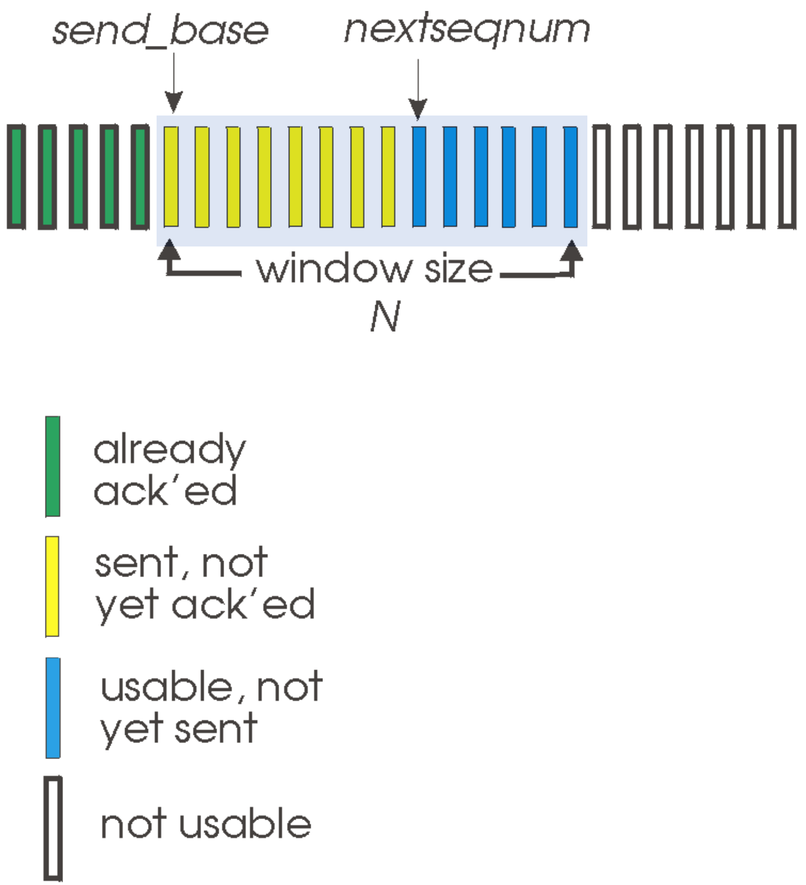


(b) a pipelined protocol in operation

Pipelined transfer of packets, where several could be in-transit and unacknowledged

- change-1: increase the range of sequence# in RDT protocols
- change-2: add buffer capability to both sender and receiver sides
- two variants: *Go-Back-N* and *Selective Repeat*

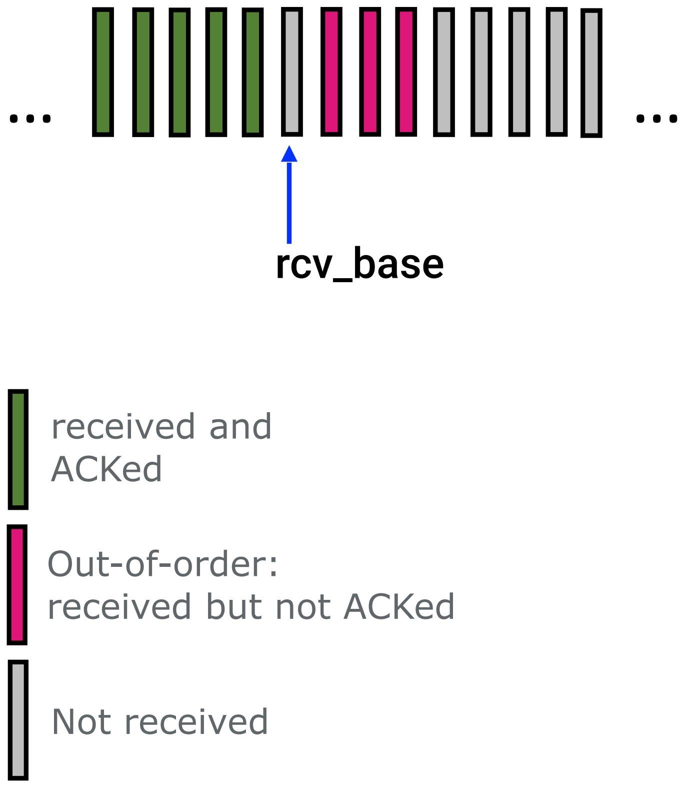
Go-Back-N (GBN) Protocol



Sender

- **Packet window**: sender defines a window of up to N consecutive transmitted but not yet ACKed packets
- **Cumulative ACK**: if the sender receives ACK(k), then it considers all packet up to k to be ACKed i.e., it moves the window forward to k+1
- **Timer**: sender always maintains a timer for the oldest not yet ACKed packet
- **Retransmissions**: Upon timer interrupt, sender retransmits the first not yet ACKed packet followed by all higher seq# packets in window

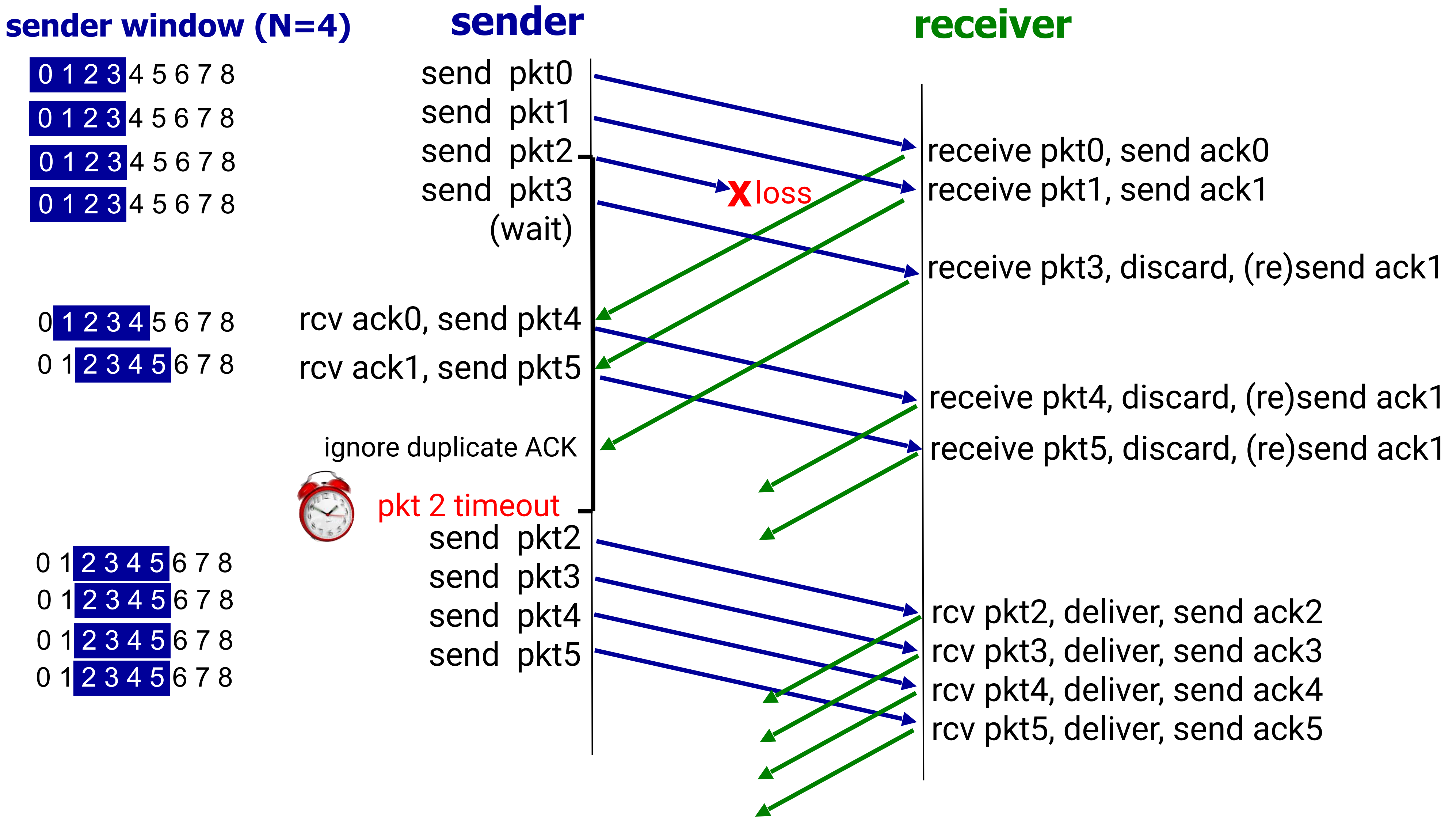
Go-Back-N (GBN) Protocol



Receiver

- **Out-of-order packets**: are typically discarded
- Discarding results in lower channel utilization, whereas buffering introduces extra state management
- **Acknowledging**: always send ACK for any correctly-received packet; however, the ACK will always carry the seq# of the highest in-order packet
- This ACK scheme may generate duplicate ACKs

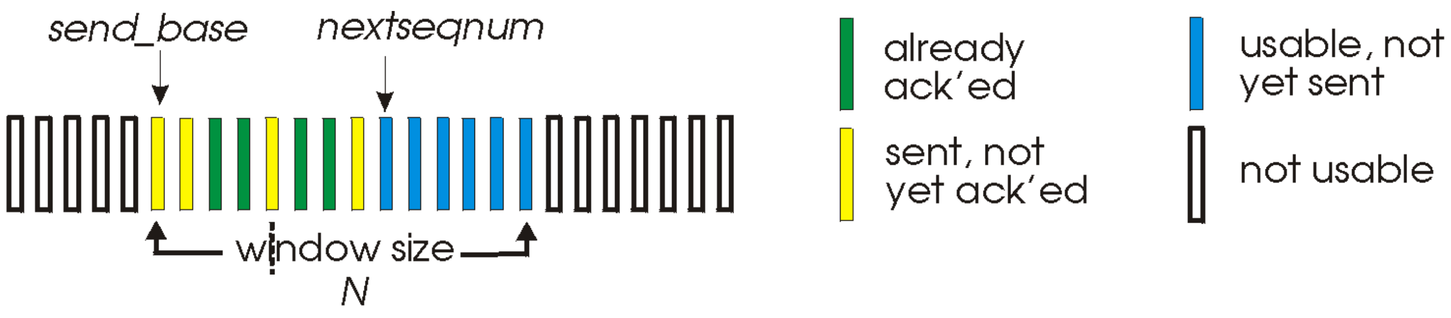
Go-Back-N in action



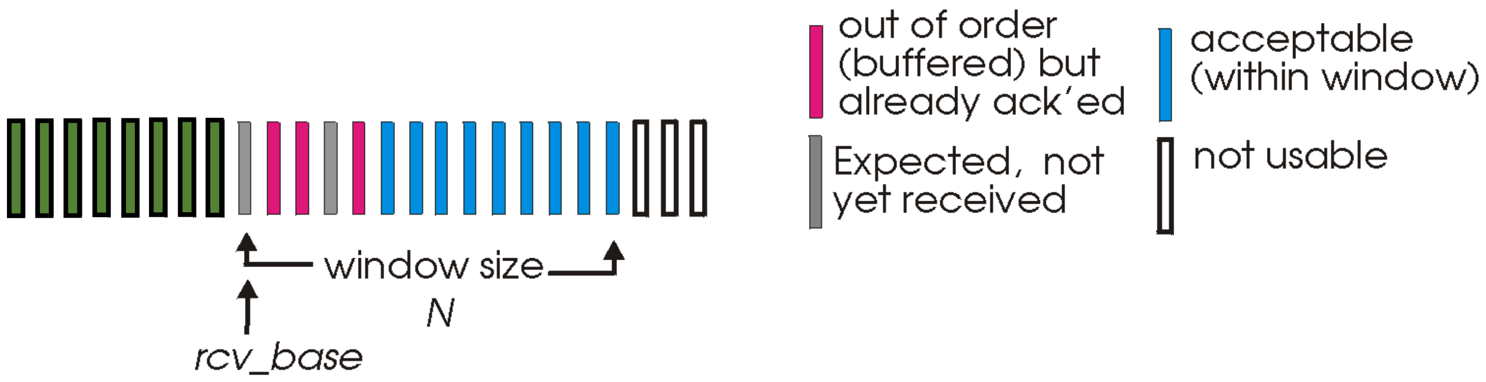
Selective Repeat (SR) Protocol

	Go-Back-N	Selective Repeat
ACKs	**Cumulative** i.e., ACK(k) will ACK all packets up to and including #k	**Individual** i.e., ACK(k) just ACKs packet #k
Out of order packets	Receiver discards all out of order packets	Buffers out-of-order packet for later delivery
Buffer size	Sender buffer = N; receiver buffer = 1	Sender buffer = N; Receiver buffer = N
Sender timer	Set for only the oldest unacknowledged packet	Set for every transmitted packet

Selective Repeat: sender and receiver windows



(a) sender view of sequence numbers



(b) receiver view of sequence numbers

Selective Repeat: sender and receiver

sender

data from above:

- if next available seq # in window, then send packet

timeout(n):

- resend packet n, restart timer

ACK(n) in [sendbase, sendbase+N]:

- mark packet n as received
- remove the timer for packet n
- if n smallest unACKed packet, advance window base to next unACKed seq #

receiver

packet n in [rcvbase, rcvbase+N]

- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order packets), advance window to next not-yet-received packet

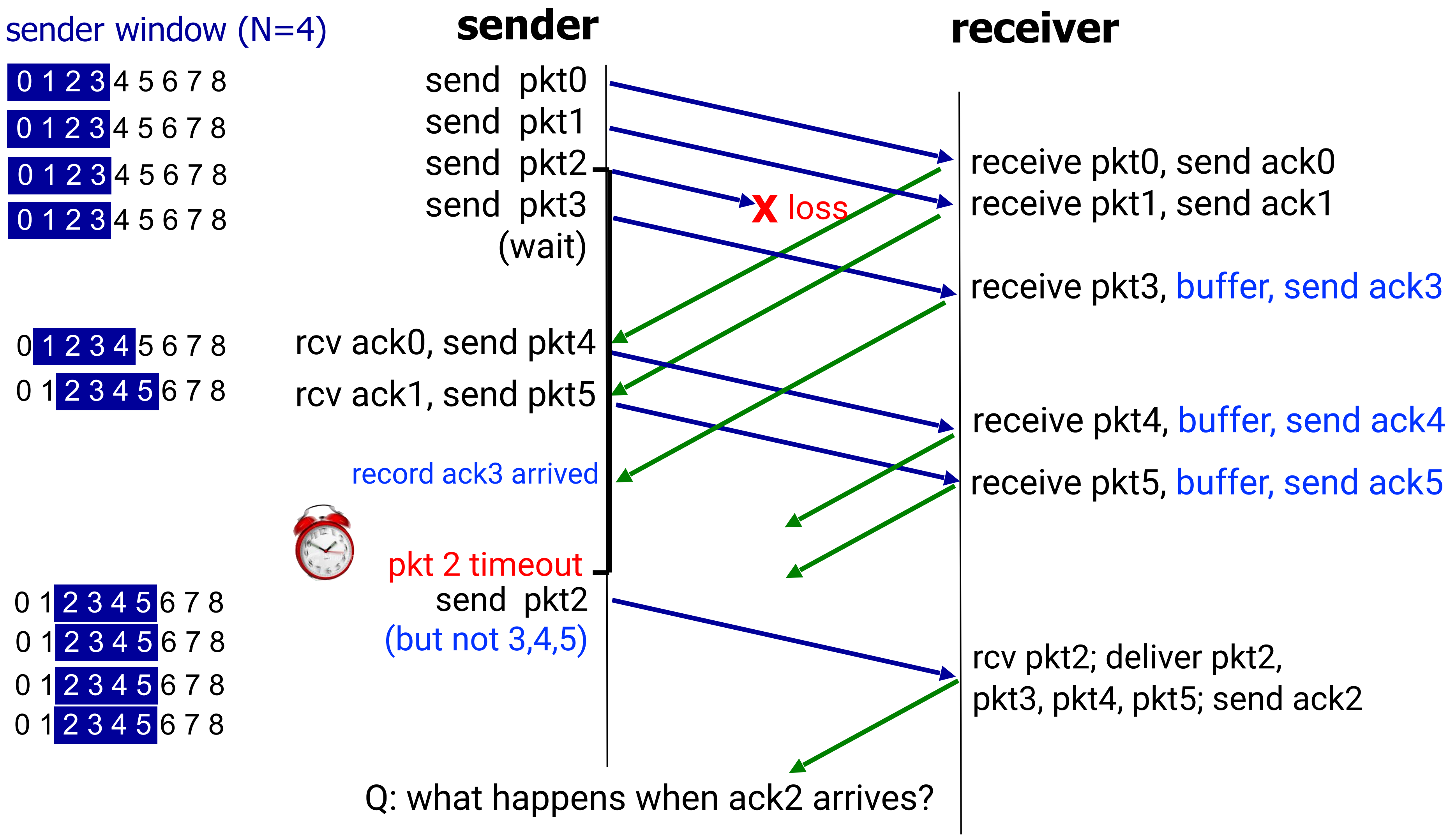
packet n in [rcvbase-N, rcvbase-1]

- ACK(n)

otherwise:

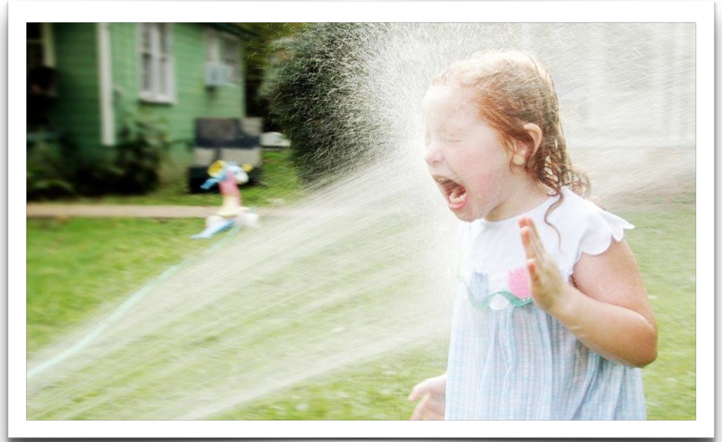
- ignore

Selective Repeat in action



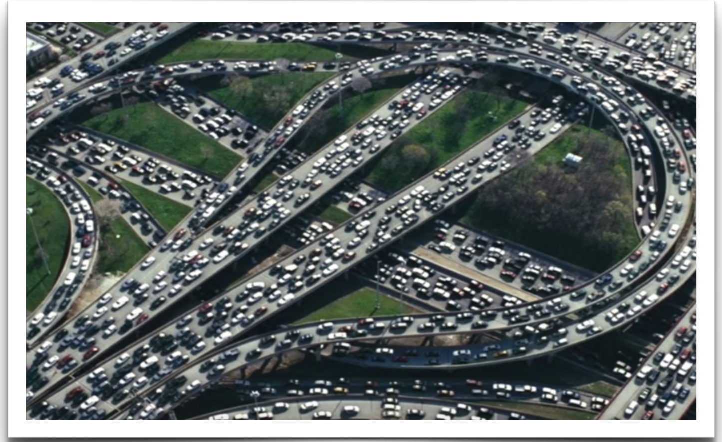
Congestion Control

Distinguishing **Flow Control** from **Congestion Control**



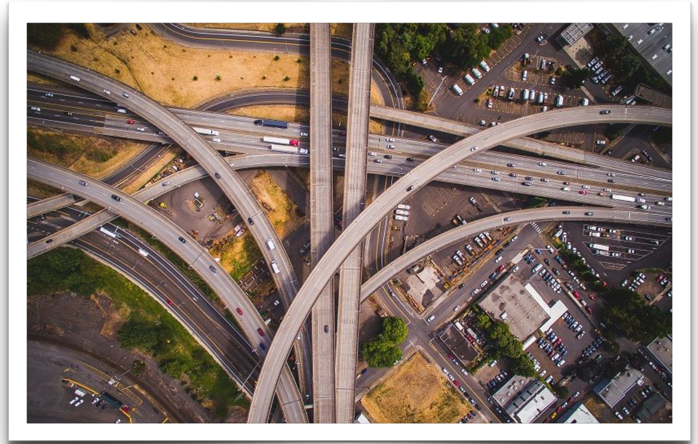
flow control

one sender too fast for one receiver

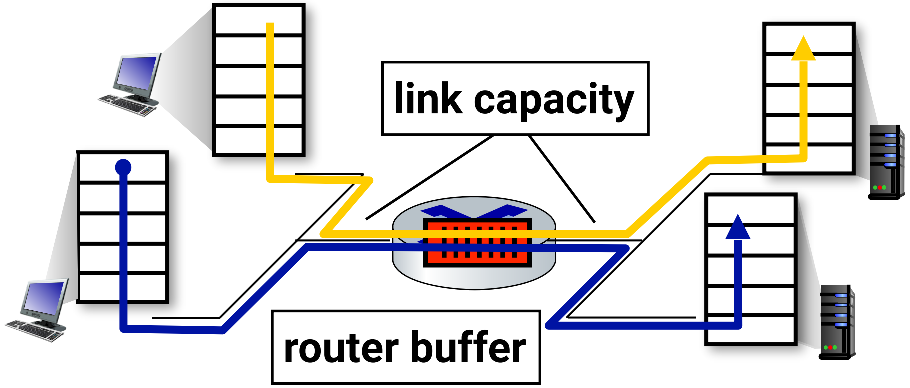


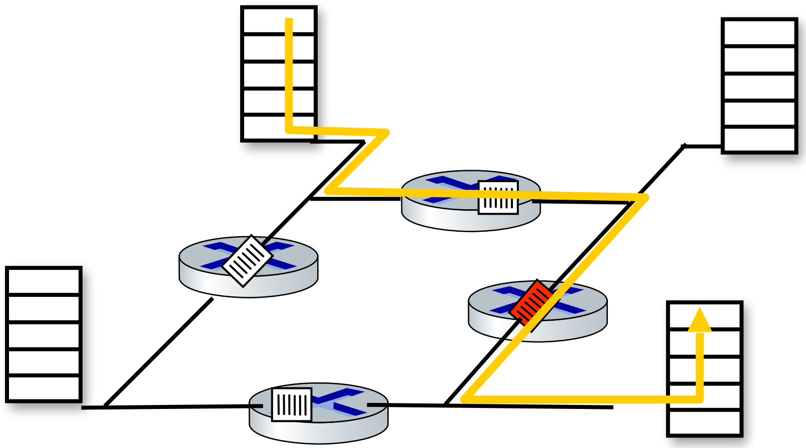
congestion control

many senders, too fast for the network to handle



Causes and Implications of Congestion





1. When the router’s link capacity gets saturated

- packets experience queuing delays
- senders and receivers experience lower throughput

2. When the router’s buffer overflows

- some packets get dropped, causing sender to retransmit
- some packets get delayed, causing sender to retransmit duplicate packets
- both cases lead to lower throughput and higher latency

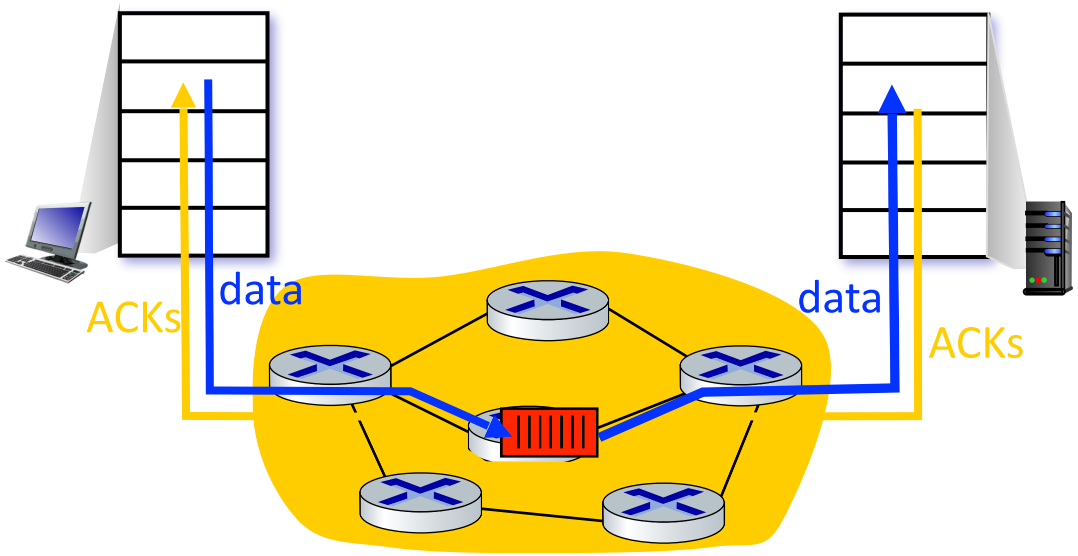
3. When an upstream router drops a packet

- transmission capacity used up by the packet so far is wasted
- this reduces the utilization of the overall network

Approaches towards congestion control

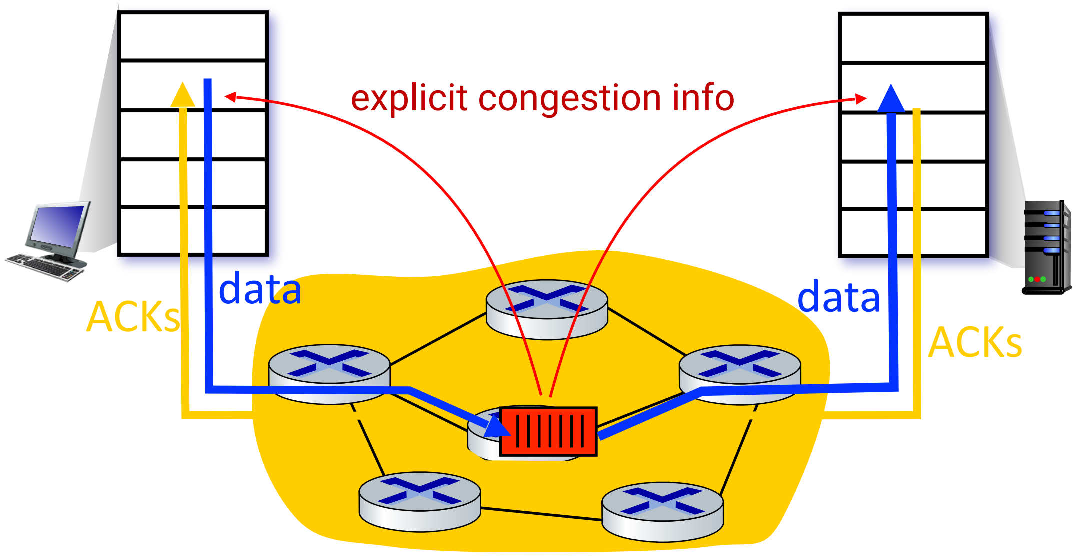
End-to-end (reactive)

- no explicit feedback from network layer
- congestion is ***inferred*** from observed loss and delay (via timeouts, duplicate ACKs, or RTT measurements)
- this is the approach taken by the original TCP



Network-assisted (proactive)

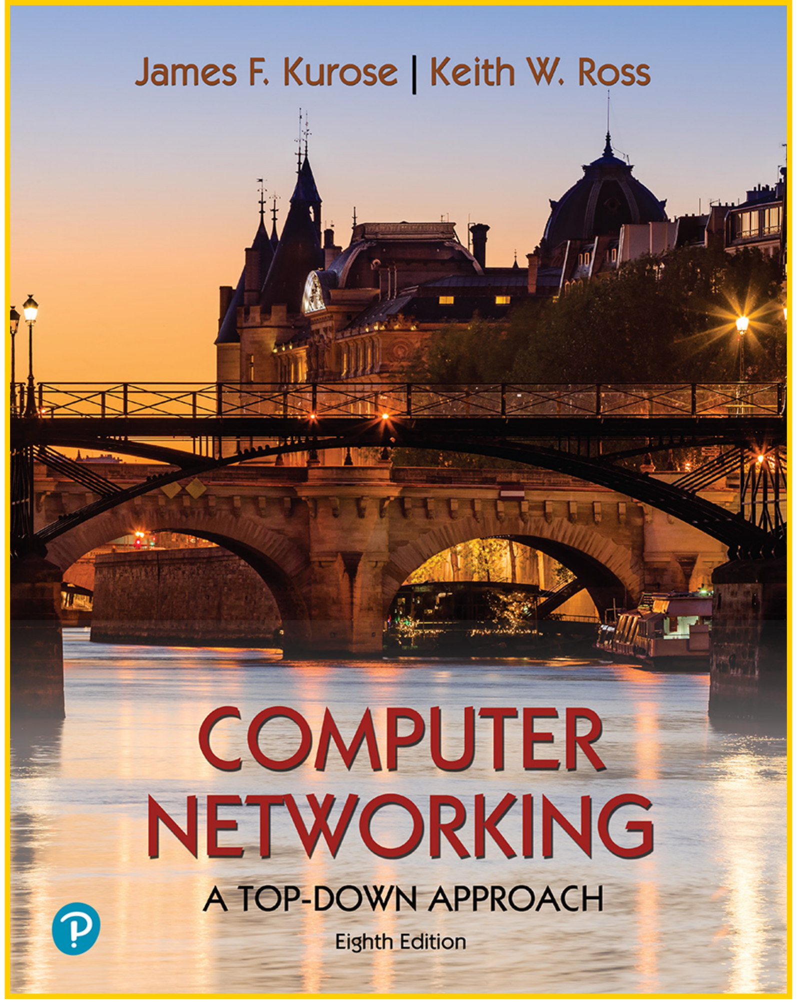
- a congested router provides ***direct*** feedback to all hosts with flows passing through it
- router acts proactively to indicate its congestion level or to explicitly set sending rate
- RFC3168 (in 2001) added explicit congestion notification (ECN) to TCP and IP



Next lecture

from principles to practice: design and operation of TCP

- *Protocol structure*
- *Connection management*
- *Reliable data transfer*
- *Flow and congestion control*



Chapters 3.5, 3.7

Spot Quiz (ICON)