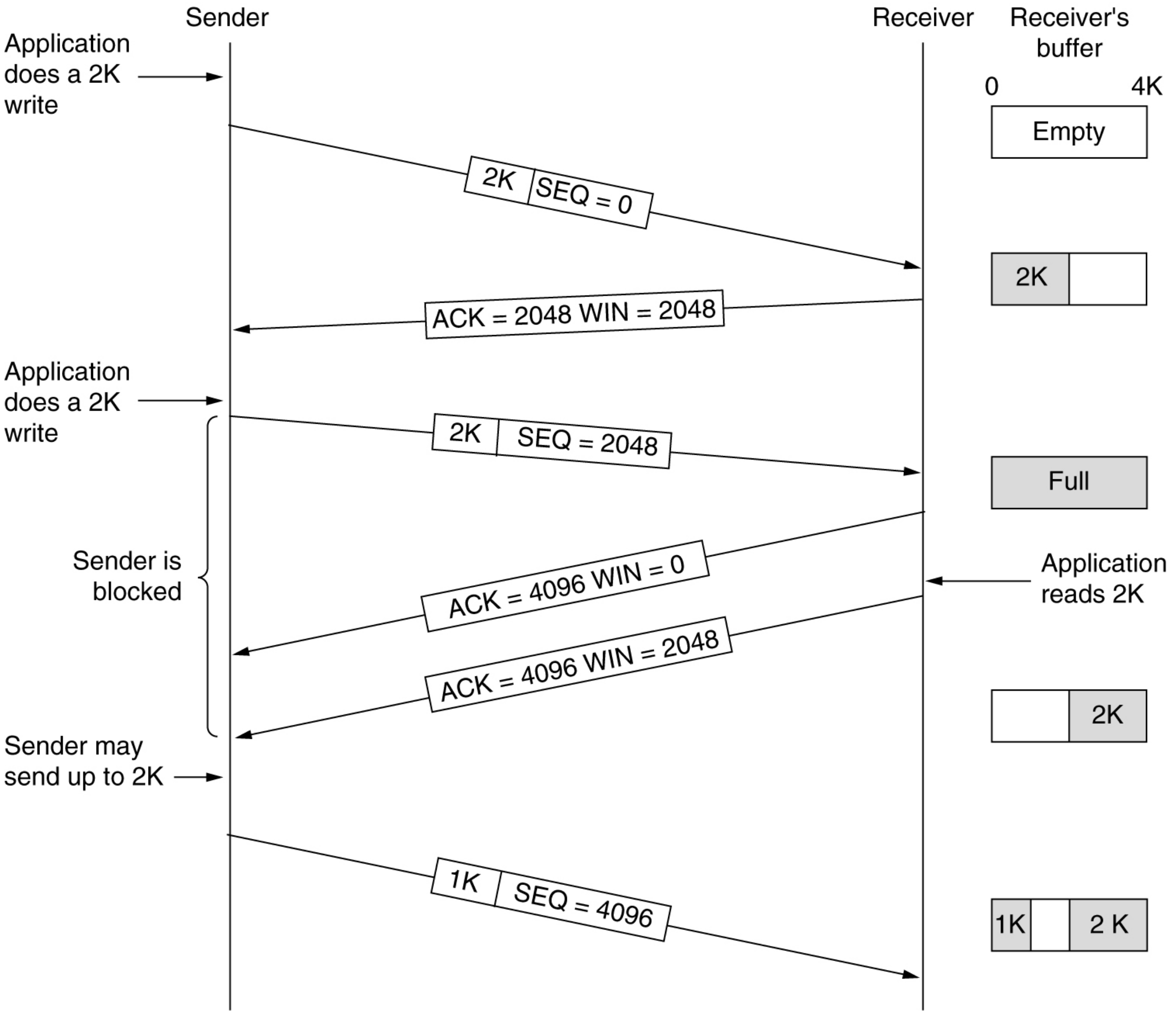
CSc 450/550
Computer Networks
Error Control

Jianping Pan

Summer 2007

Review: TCP flow control

- Purpose
  - to avoid overflow
- Mechanism
  - sliding window
  - variable window



6/14/06

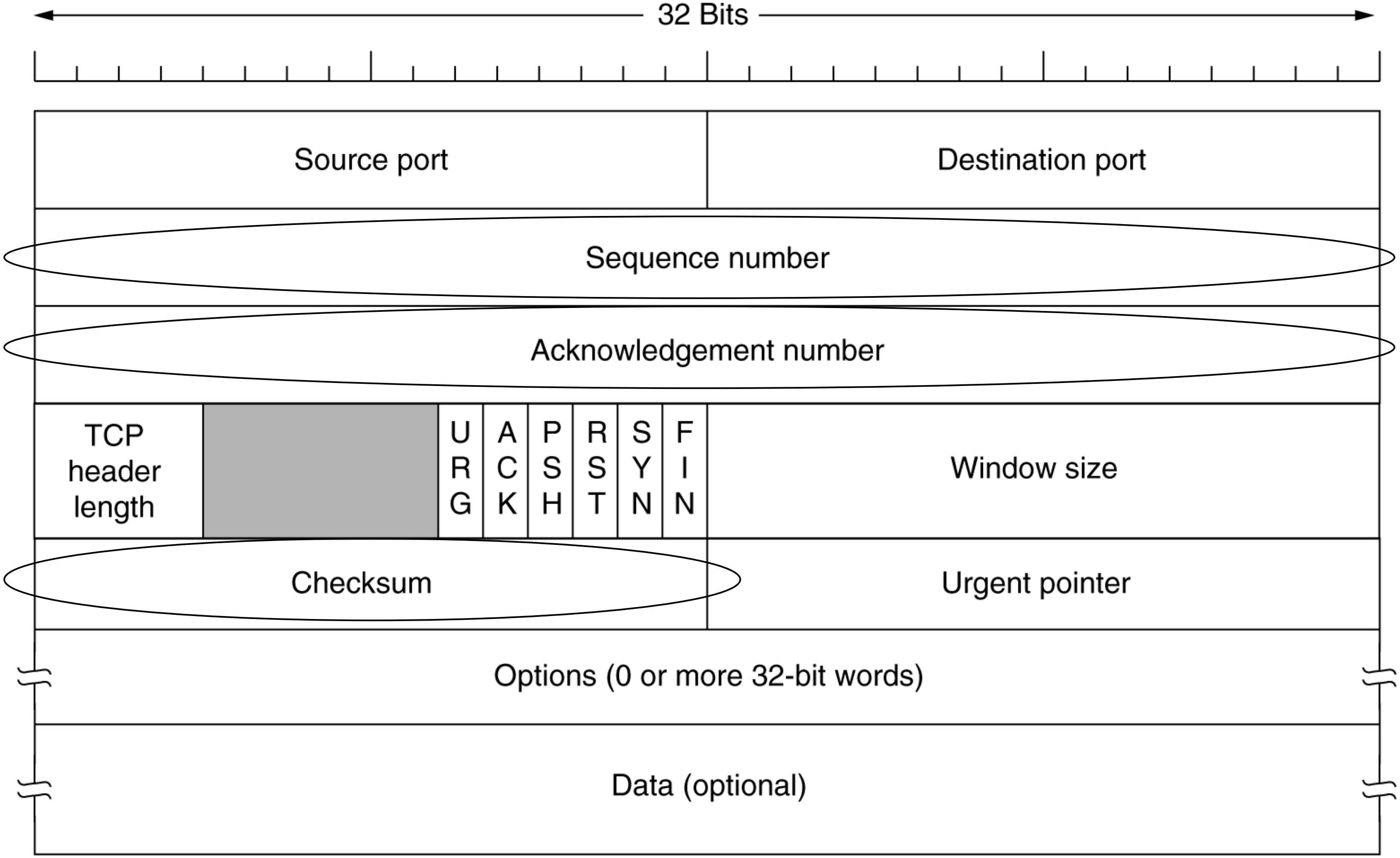
Q: seq, ack, win?

Error control

- Service provided by TCP
  - connection-oriented, reliable data transfer
- Service provided by IP
  - connectionless, unreliable packet delivery
  - packets may get
    - lost
    - duplicated
    - corrupted
    - reordered

Q: why?

TCP packet header



32 Bits

Source port

Destination port

Sequence number

Acknowledgement number

TCP header length

URG

ACK

PSH

RST

SYN

FIN

Window size

Checksum

Urgent pointer

Options (0 or more 32-bit words)

Data (optional)

What can go wrong?

- IP packet delivery
  - lost
    - *transmission error* or network congestion
  - duplicated
    - deleted by referring to sequence number; done
  - corrupted
    - arrived but in “bad shape”
  - reordered
    - rearranged by referring to sequence number; done

Error detection

- Corrupted packets
  - detected by TCP checksum
    - action: drop!
- Lost packets
  - how do you tell if something ***is*** already lost?
  - TCP sender
    - timer for acknowledgment
  - TCP receiver (cumulative acknowledgment)
    - duplicate acknowledgment

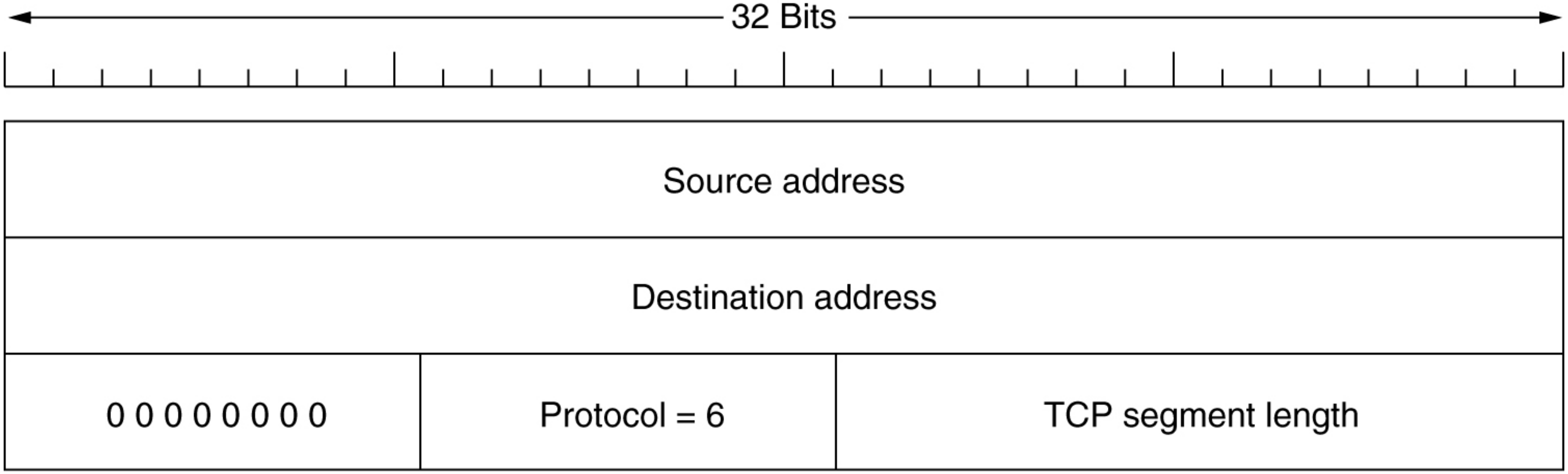
TCP/IP checksum

- Algorithm: 16-bit one compliment of one's compliment sum with carry
  - 16-bit: padding when necessary
    - cover: TCP header, payload, pseudo header
  - calculate: pad, sum, carry, compliment => checksum
  - verify: sum with checksum, carry, compliment => 0?
- Examples

		1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
		1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

IP pseudo header

- TCP checksum also covers IP pseudo header
  - to detect mis-delivered packets by IP layer
    - include: IP addresses, protocol ID, segment length



32 Bits

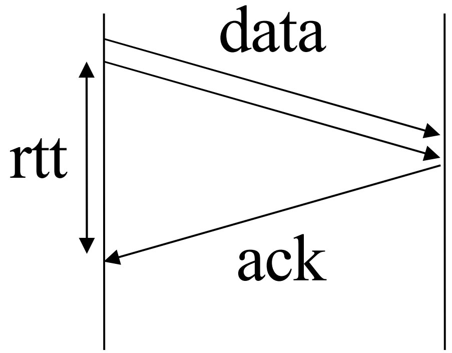
Source address

Destination address

0 0 0 0 0 0 0 0 | Protocol = 6 | TCP segment length

Q: why pseudo header?

TCP sender timer

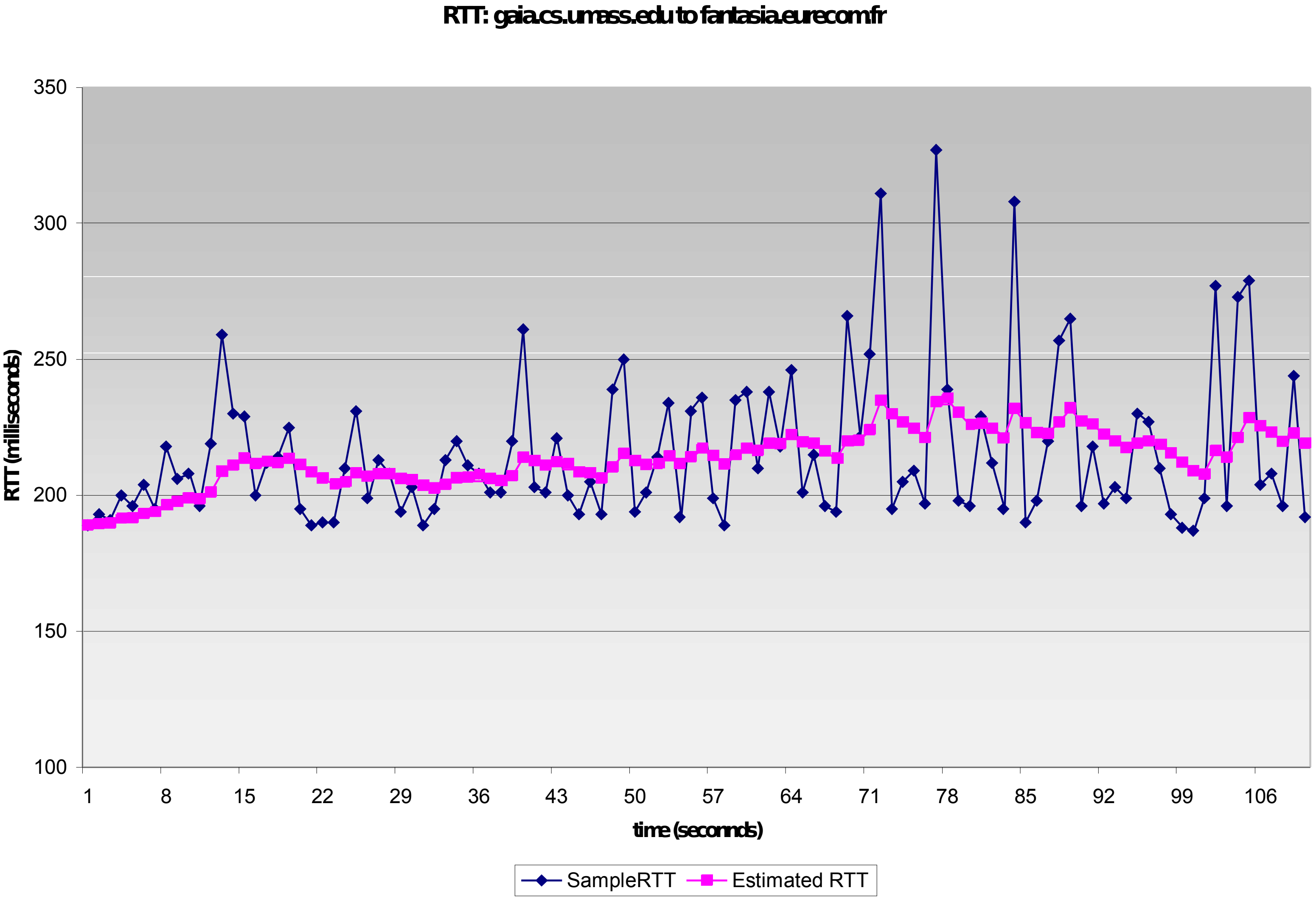


- TCP sender
  - start a timer when sending out a packet
    - in reality: one timer per a window of packets
  - on acknowledgment “covering” this packet
    - cancel the timer and setup another one
  - if timer timeouts: *indicate* packet may be lost
- Timeout value
  - too soon: unnecessary transmission
  - too late: “slow response”

TCP round-trip time

- RTT measurement and calculation
  - RTT sample
    - time from sending a packet to receiving its ack
      - coarse-grained: 500 ms in BSD
    - ignore retransmitted packets
  - smoothed RTT (SRTT)
    - exponentially weighted moving average (EWMA)
    - $\mathrm{SRTT}_{i+1} = \mathrm{SRTT}_i + a\,(\mathrm{RTT}\text{-}\mathrm{SRTT}_i)$
    - $a = 1/8$

EWMA example



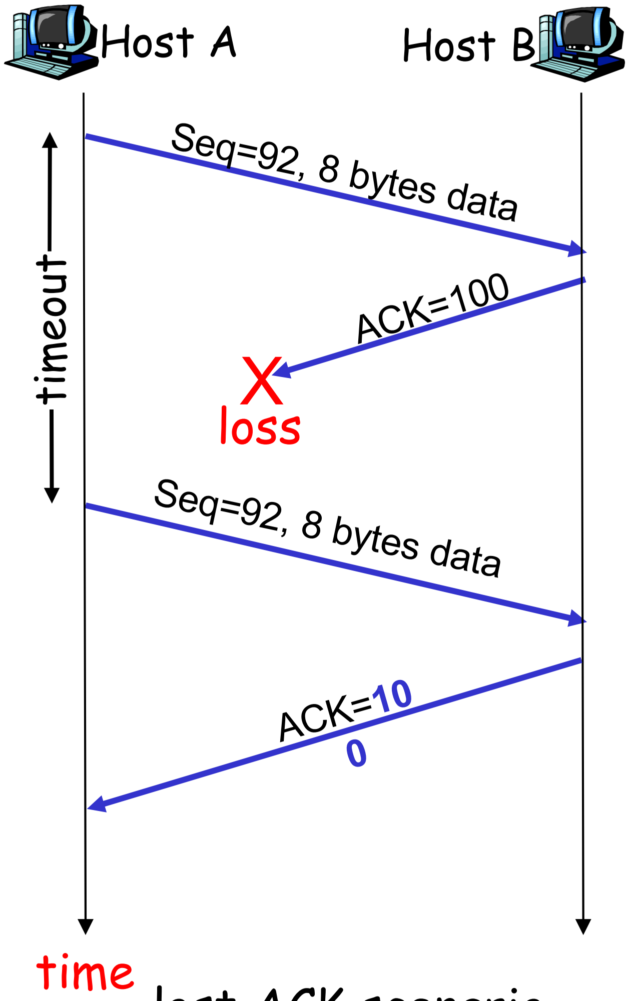
TCP timeout value

- RTO calculation based on SRTT
  - RTT variance (RTTV)
    - $RTTV_{i+1} = RTTV_i + b(|RTT - SRTT_i| - RTTV_i)$
    - $b = 1/4$
  - RTO
    - RTO = d (SRTT + c RTTV)
    - c: initially 2, now 4
    - d: backoff factor
      - initially 1, doubled when timeout until reaching the maximum
  - initial SRTT, RTTV and minimum RTO

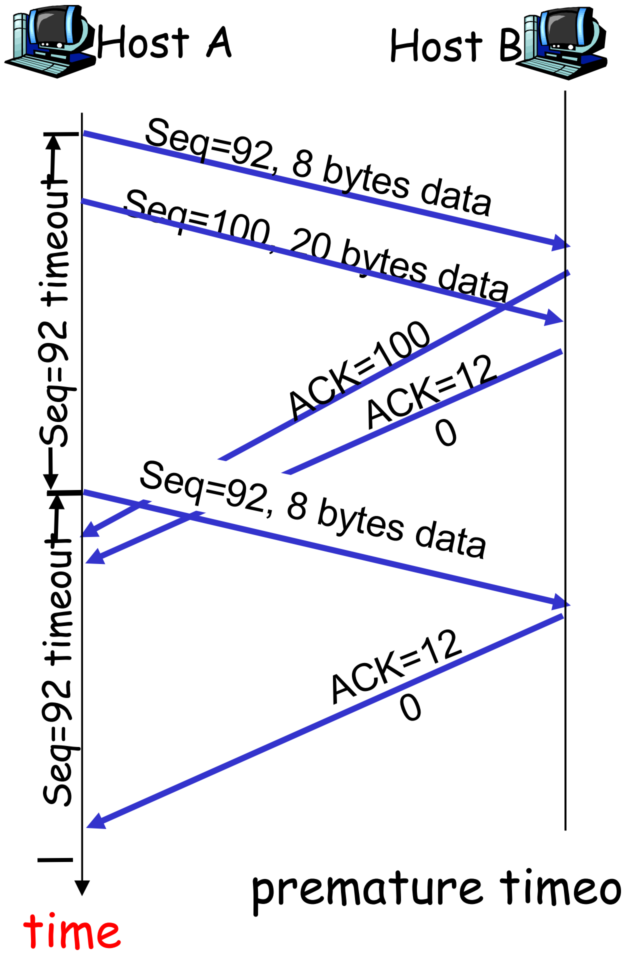
TCP sender (simplified with no flow control and congestion control)

```
send_next = InitialSeqNum
ack = InitialSeqNum

 loop (forever) {
   switch(event)

   event: data received from application above
        create TCP segment with sequence number send_next
        if (timer currently not running)
             start timer with timer's seqno = send_next
        pass segment to IP
        send_next = send_next + length(data)

    event: timer timeout
        retransmit not-yet-acknowledged segment with
               smallest sequence number
        start timer with the resent seqno

    event: ACK received, with ACK field value of y
        if (y > ack) {
             ack = y
             cancel timer with timer's seqno < y
            if (timer not running && there are currently not-yet-acknowledged segments)
                   start timer
            }
 }  /* end of loop forever */
```

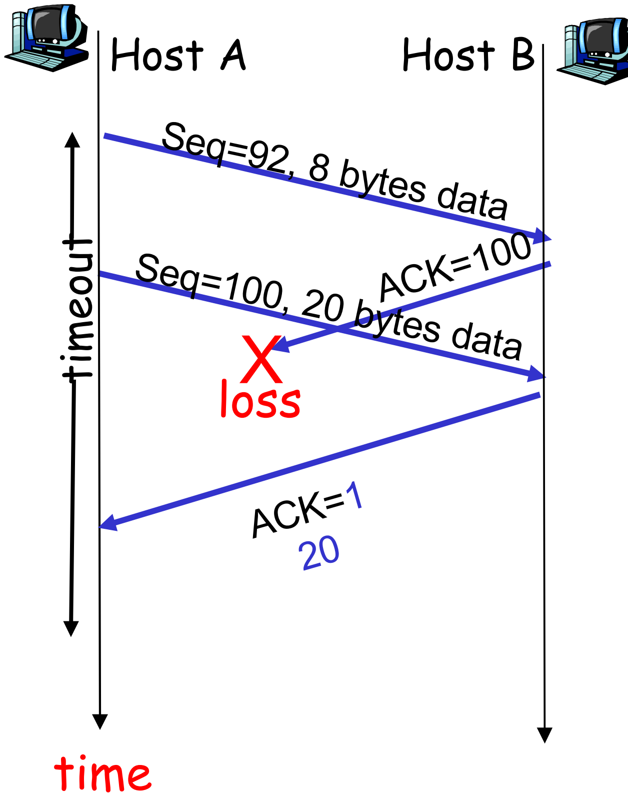
TCP: retransmission scenarios



lost ACK scenario



premature timeout



Cumulative ACK

TCP ACK generation [RFC 1122, RFC 2581]

Event at Receiver	TCP Receiver action
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
Arrival of in-order segment with expected seq #. One other segment has ACK waiting	Immediately send single cumulative ACK, ACKing both in-order segments
Arrival of out-of-order segment higher-than-expect seq. # . Gap detected	Immediately send duplicate ACK, indicating seq. # of next expected byte
Arrival of segment that partially or completely fills gap	Immediate send ACK, provided that segment starts at lower end of gap (accept out-or-order)

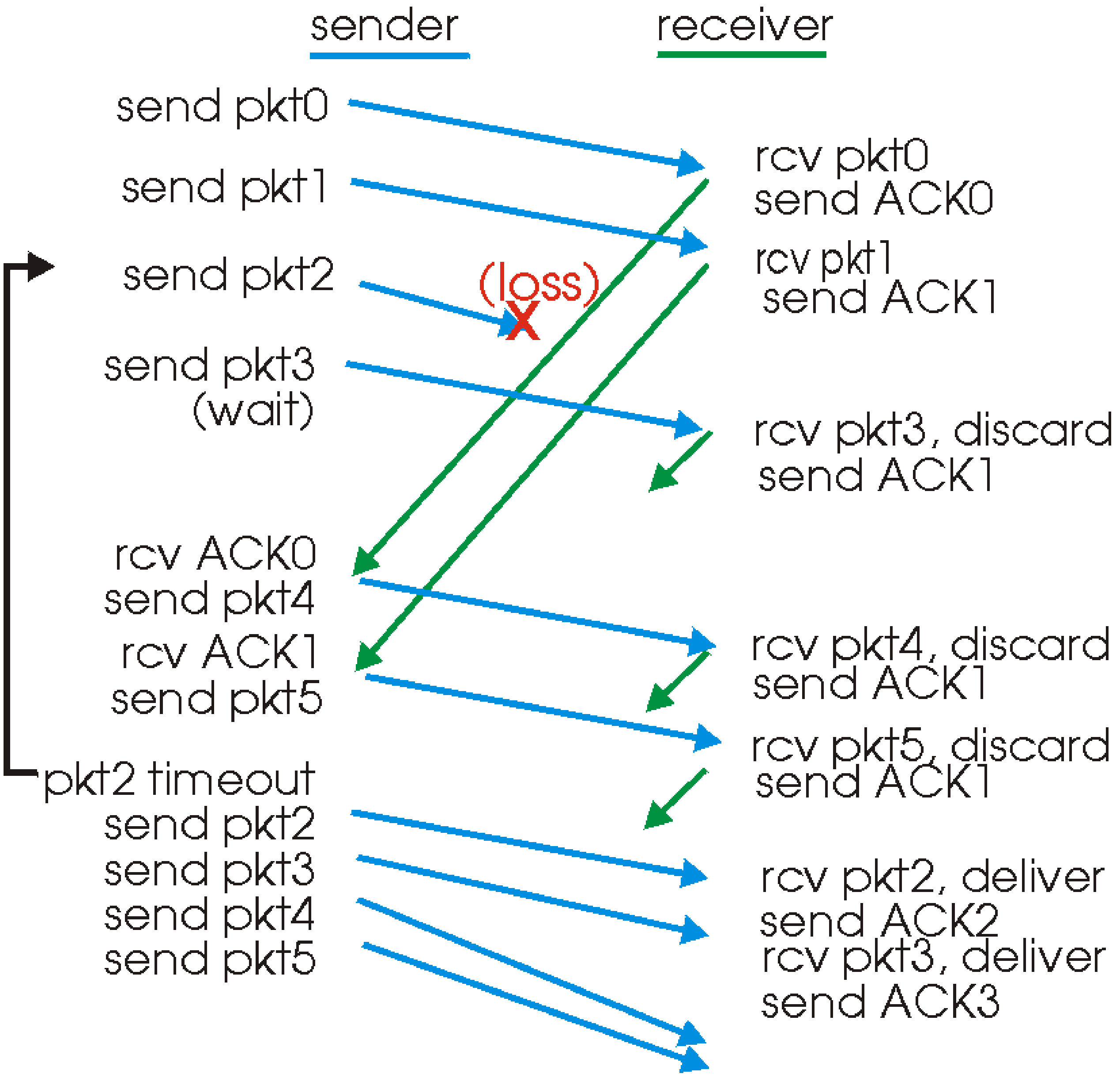
Duplicate acknowledgment

- TCP acknowledgment
  - cumulative acknowledgment
  - example
    - rcv: [0, 500),[500, 1000),[1500, 2000),[2000, 2500)
    - ack: 500,1000,1000 (1st dupack),1000 (2nd dupack)
- Enough duplicate acknowledgments
  - *indicate* packet loss may have occurred
    - ack: 500, 1000, 1000, 1000, 1000 (3rd dupack)
    - packet [1000,1500) is considered lost

Error recovery

- End-to-end retransmission
  - go-back-N (GBN)
    - retransmit from ackno and upward
  - selective retransmission
    - only retransmit those “known” to be lost
- TCP’s error recovery
  - mostly GBN
    - receiver can buffer out-of-order packets
  - explore further: TCP selective acknowledgment

GBN in action (N = 4)



Q: how does SR look like?

This lecture

- TCP error control
  - purpose
  - mechanisms
    - detection
    - recovery
- Explore further
  - TCP selective acknowledgment (SACK)
  - http://www.icir.org/floyd/

One more message...

- NSERC USRA opportunities
  - http://www.cs.uvic.ca/~pan/usra/
  - get a taste of doing research
    - answer unanswered questions and improve answered ones!
  - possible projects
    - First, you can always propose your own projects...
    - Multimedia Streaming over Multipath Networks
    - Scalable Network measurement on UVicNet
    - A Network Testbed for Service Provider Networks
    - and more...

Next lectures

- TCP congestion control