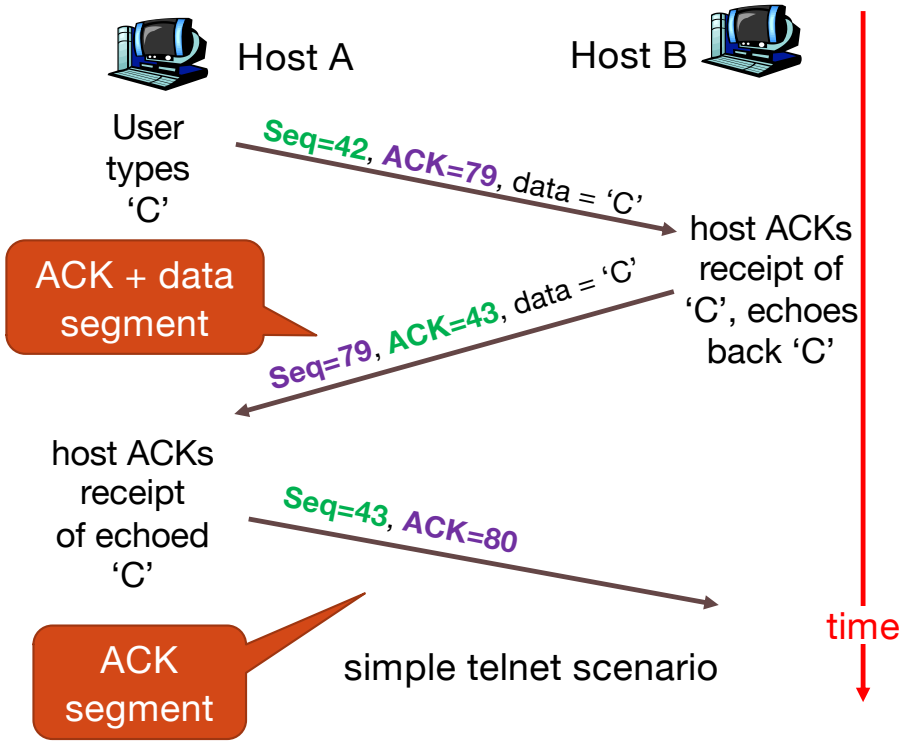
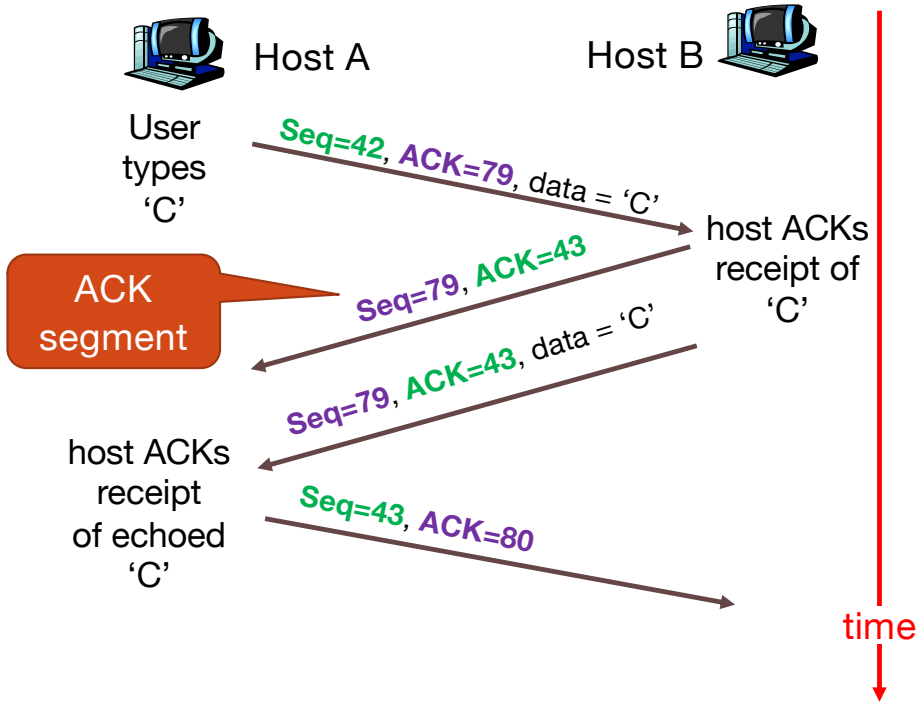
CPSC 317
COMPUTER NETWORKING

2023W2: Transport – Day 7 - TCP

TCP ACKS


Host A
Host B
User types 'C'
Seq=42, ACK=79, data = 'C'
host ACKs receipt of 'C', echoes back 'C'
ACK + data segment
Seq=79, ACK=43, data = 'C'
host ACKs receipt of echoed 'C'
Seq=43, ACK=80
ACK segment
simple telnet scenario
time


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

LEARNING GOALS

TCP

- Explain how TCP handles lost data and lost ACKs
- Explain TCP's "fast retransmit" mechanism
- Describe and analyze TCP's congestion detection and control mechanisms

READING

- Reading: 3.5.4, 3.7 before 3.7.1

TCP RETRANSMISSION

Retransmission based on timeout

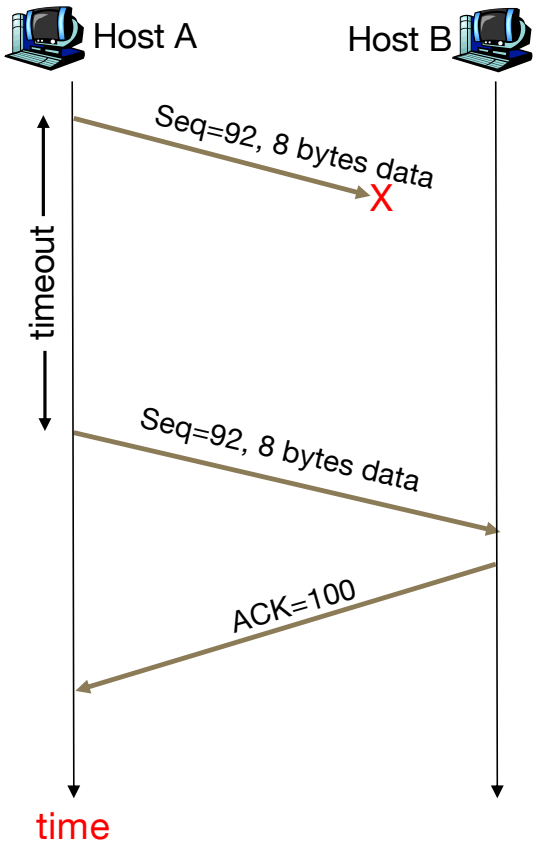
- A timer is set for the first unacknowledged segment
- When the timer expires we retransmit
- Retransmit just one segment (like Selective-Repeat)

Fast retransmission

- Four or more ACKs with same ack number trigger a retransmission without a timeout

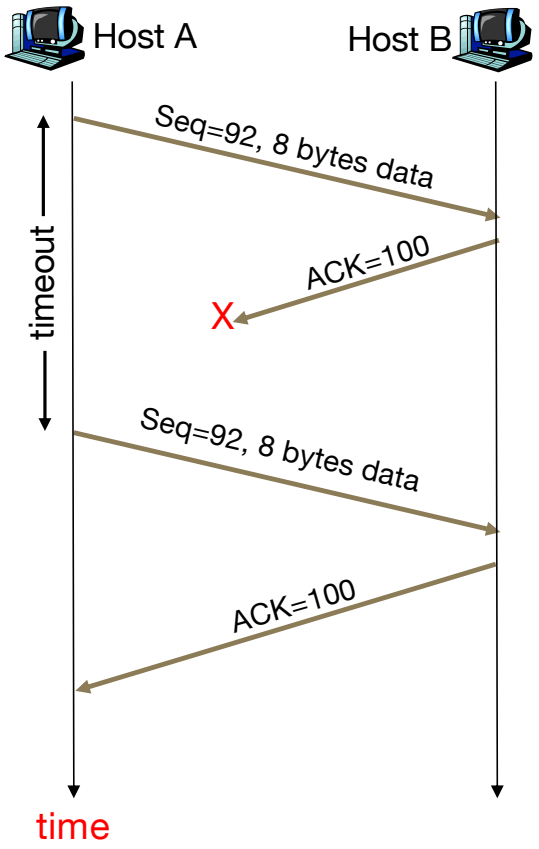
TCP RETRANSMISSION SCENARIOS

Lost data scenario



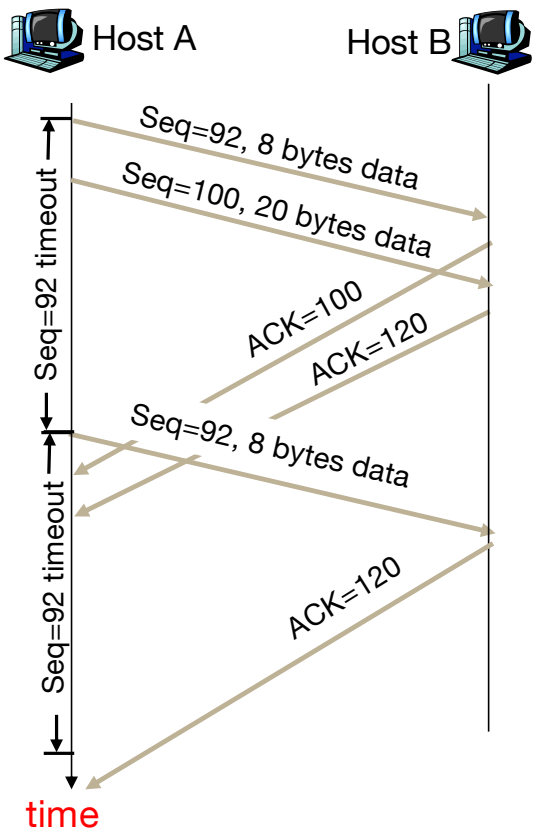
TCP RETRANSMISSION SCENARIOS

Lost ACK scenario



TCP RETRANSMISSION SCENARIOS

Premature timeout

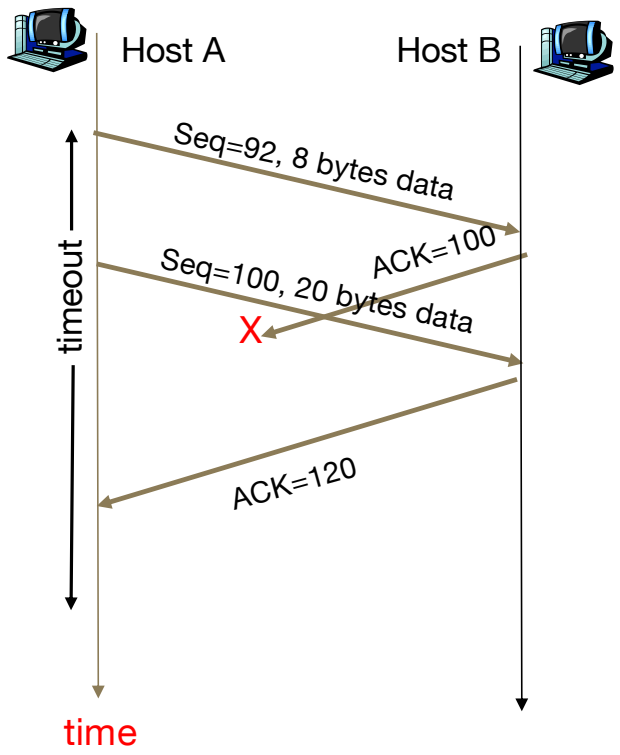


Second segment not resent

ACK's both segments even though only the first one was re-sent

TCP RETRANSMISSION SCENARIOS

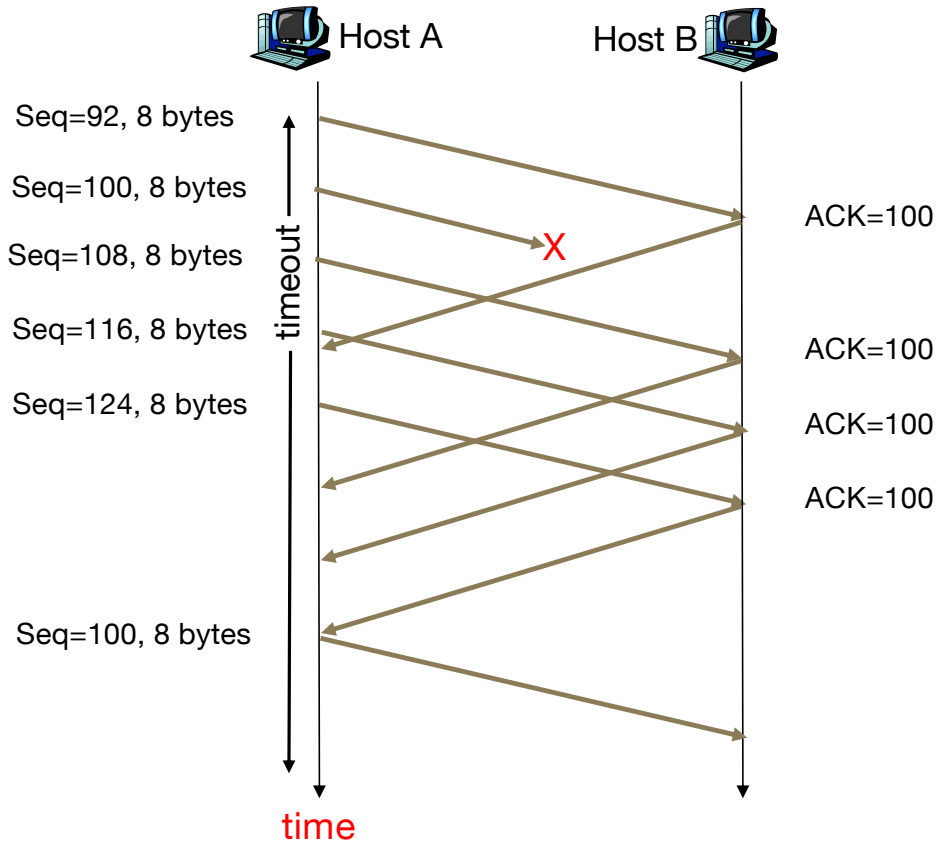
Cumulative ACK scenario



Data is not resent

TCP RETRANSMISSION SCENARIOS

Early retransmission
(triple duplicate ACK)



CLICKER QUESTION

How does TCP handle lost data packets?

A. About like GBN
B. About like SR
C. Better than both GBN and SR
D. Worse than both GBN and SR

CLICKER QUESTION

How does TCP handle lost ACK packets?

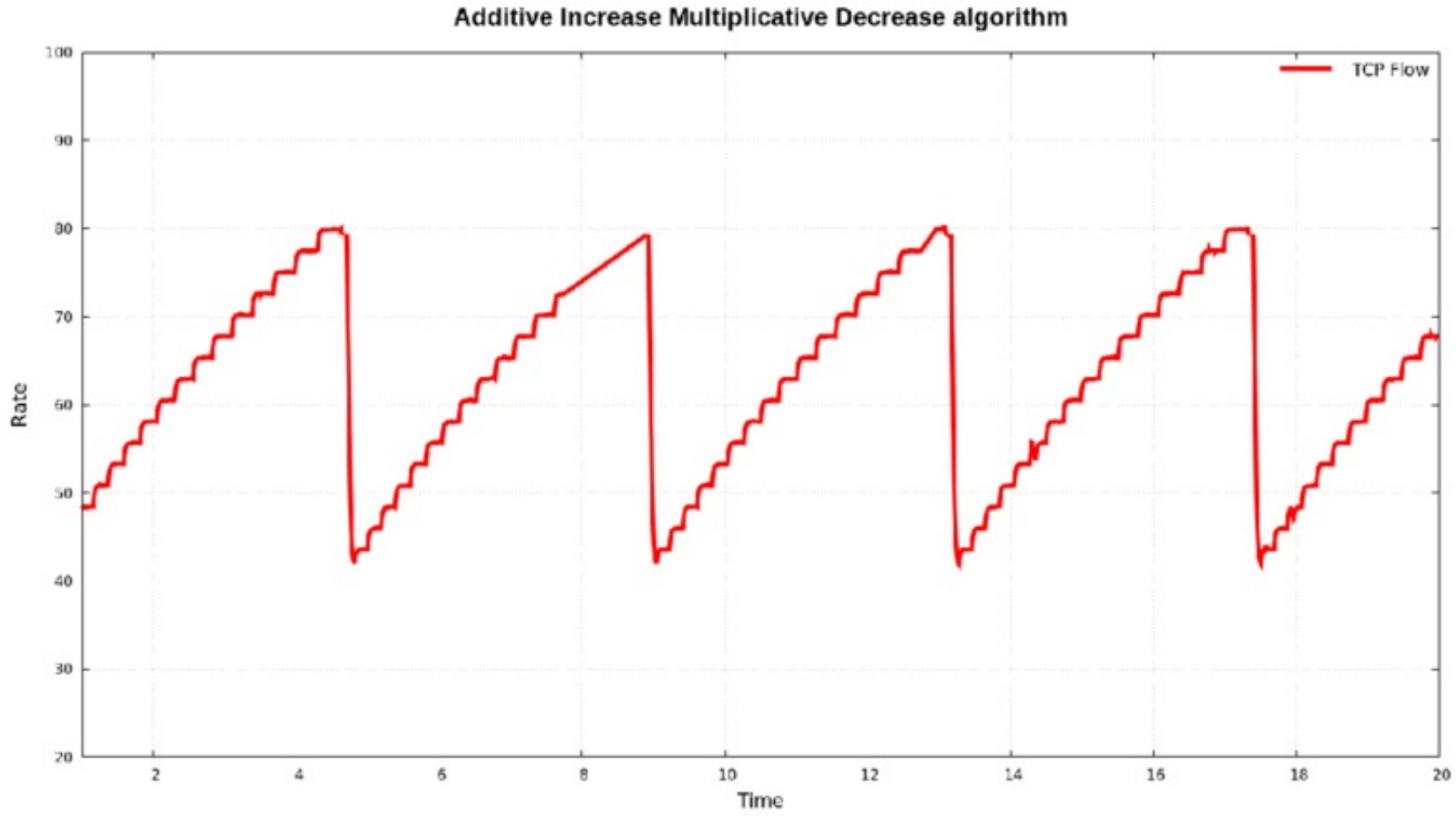
A. About like GBN
B. About like SR
C. Better than both GBN and SR
D. Worse than both GBN and SR

CONGESTION DETECTION

- Congestion is detected via:
  - Timeouts
  - Multiple (4) identical ACKs (triple duplicate ACK)
- Sender adapts its congestion window
- Lack of congestion is detected via:
  - Sending a window full of data without detecting congestion

CONGESTION MANAGEMENT

- TCP is very conservative
  - It never wants to overwhelm the network with data
- At the first sign of congestion it slows down a lot
  - Cuts congestion window in half
  - Multiplicative decrease
- When it appears that congestion has eased it increases slowly
  - Adds 1 segment to congestion window
  - Additive increase


Additive Increase Multiplicative Decrease algorithm
TCP Flow
Rate
Time
100
90
80
70
60
50
40
30
20
2
4
6
8
10
12
14
16
18
20

SLOW START

- We start with a congestion window of 1 segment
- Increase by 1 each time a segment is ACKed
  - Which is equivalent to doubling it each time we send a window full of data with no congestion detected
- Stop doubling when we detect congestion

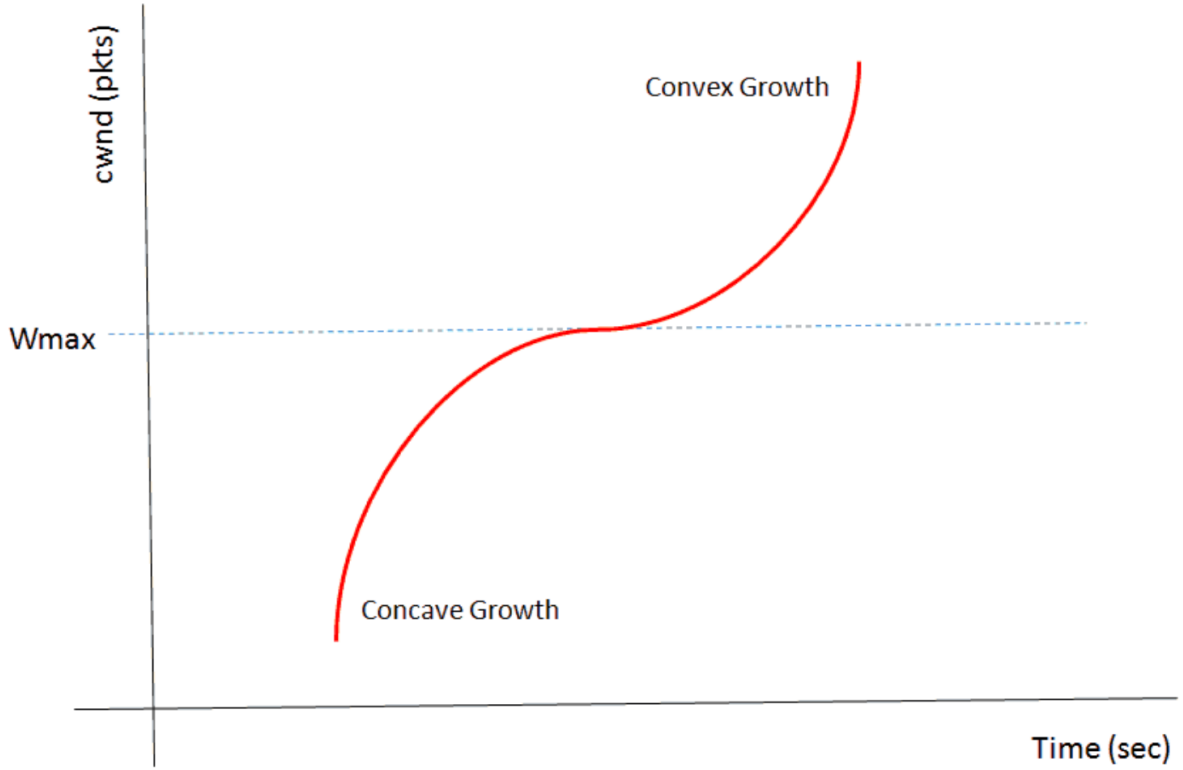
TCP CONGESTION MANAGEMENT IS EVOLVING

- Many versions:
  - Tahoe
    - Very aggressive in response to congestion
    - Drop the congestion window to 1 and enter slow start
  - Reno and New Reno
    - Less aggressive
    - Cut the congestion window in half
  - Vegas – congestion avoiding
    - Use increasing RTTs as an earlier signal of approaching congestion
    - Slow down when RTTs increase
    - Speed up when RTTs decrease

TCP CONGESTION MANAGEMENT IS EVOLVING

- Many versions:
  - BIC
    - Increases more rapidly when recovering from congestion
    - Binary search between the congestion window and the max window size
  - CUBIC
    - The congestion window isn't changed every RTT
    - Instead it is a cubic function of the time since the last congestion detection

CUBIC



IN-CLASS ACTIVITY

- Explore TCP congestion management
  - and review TCP's lost data/ACK behaviour
- ICA47