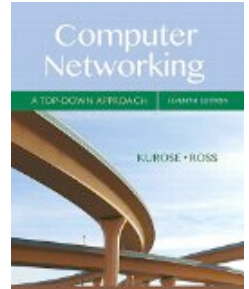# COMP 375: Lecture 22

- **News & Notes:**
  - Midterm #1 grades released before lab
  - Quiz #5 in class today
  - Project #3 due @ 10PM
  - Project #4 due Monday, April 9
- **Reading (Wed, March. 21)**
  - Review Sections 3.6 and 3.7

Quiz #5

- Closed book, closed notes.
- Happy "National Let's Laugh Day"

Section 3.5

RELIABLE TRANSPORT WITH TCP

Over the next two classes, we'll look to answer the following questions about TCP.

- How is pipelining handled?
- **How should we choose timeout values?**
- What are connections created and destroyed?
- How many segments should be pipelined?

TCP uses the equation below to calculate estimated RTT after each new sample.

EstimatedRTT = (1 − α) * EstimatedRTT + α * SampleRTT

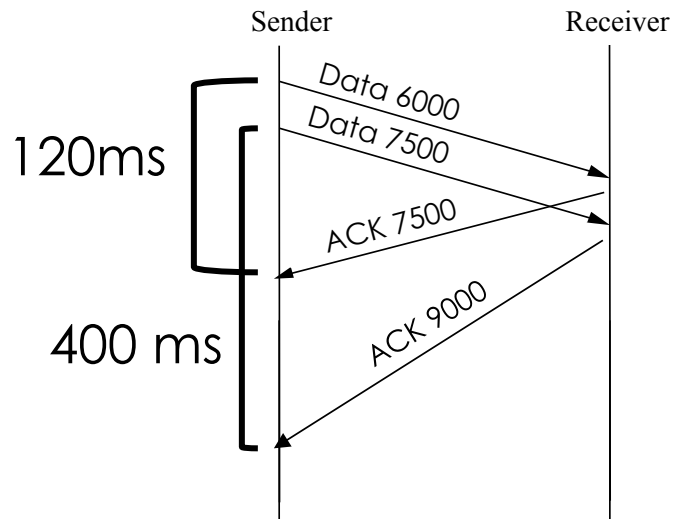*Which samples are given more weight?*

(Note: SampleRTT is the most recent RTT time.)

A. Newer samples given more weight.

B. Older samples given more weight.

C. All samples have the same weight.

D. It depends on α

TCP uses **E**xponentially **W**eighted **M**oving **A**verage (EWMA) to estimate RTT.

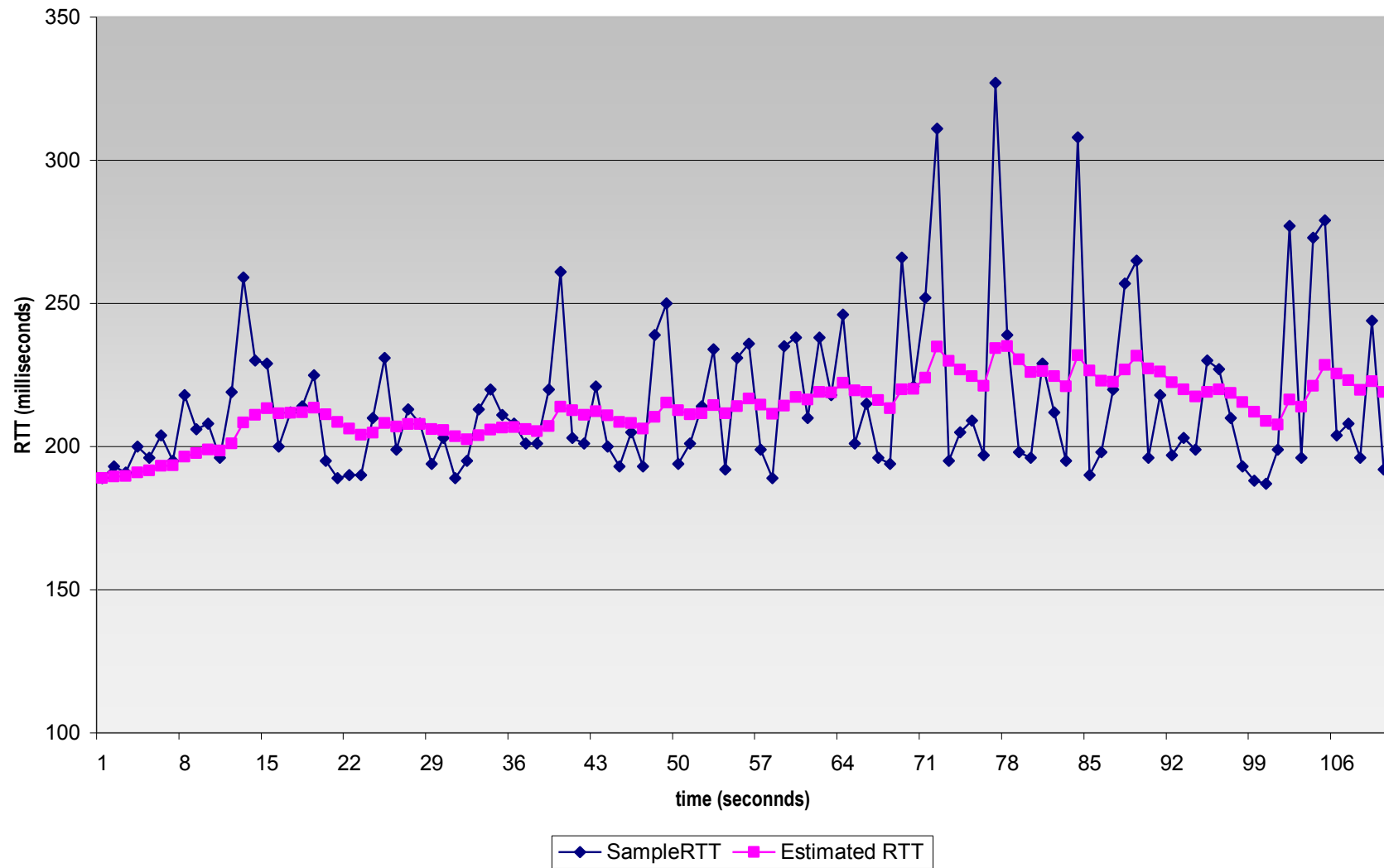Assume we start with EstimatedRTT = 64 ms and
DevRTT = 8 ms. Use α = 1/8, β = 1/4.

EstimatedRTT = $(1 - \alpha)$ * EstimatedRTT + α * SampleRTT

DevRTT = $(1 - \beta)$ * DevRTT + β * | (SampleRTT – EstimatedRTT) |

Estimated RTT varies much less than the actual, sampled RTT.

TCP's timeout is set based on estimated RTT and the deviation.

```
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

"safety margin"

Over the next two classes, we'll look to answer the following questions about TCP.

- How is pipelining handled?

- How should we choose timeout values?

- **How are connections created and destroyed?**

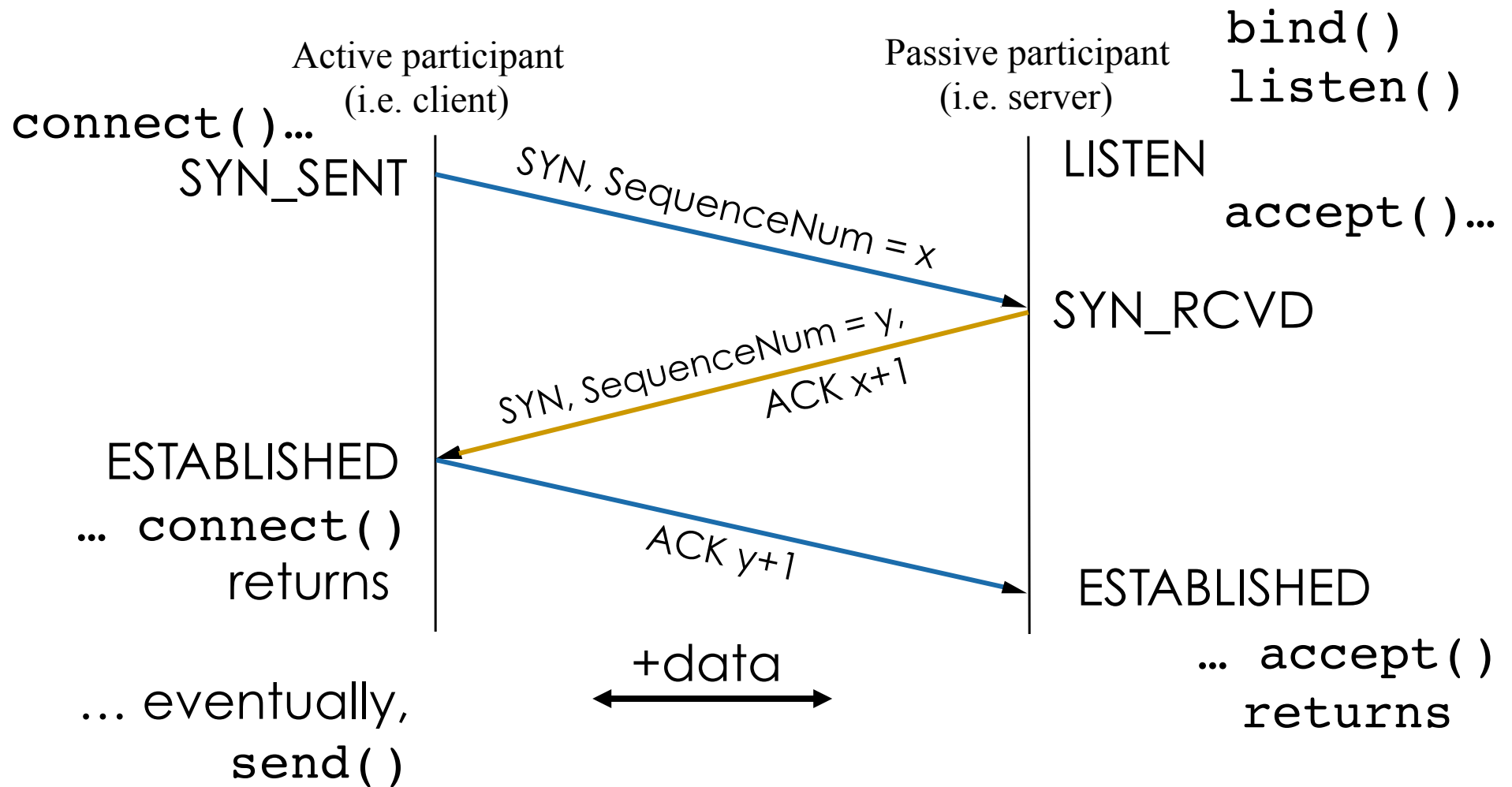- How many segments should be in-flight?

TCP Connections

Which of the following statements are *TRUE* about a TCP connection?

1. It requires stored state at two hosts.

2. It requires stored state within the network.

3. It establishes a path between two hosts.

A.	1
B.	2
C.	1 and 3
D.	2 and 3
E.	1, 2, and 3

TCP **connection setup** provides an opportunity to exchange initial state.

Active participant
(i.e. client)

Passive participant
(i.e. server)

`bind()`
`listen()`

`connect()`...
SYN_SENT

LISTEN

`accept()`...

SYN, SequenceNum = x

SYN_RCVD

SYN, SequenceNum = y,
ACK x+1

ESTABLISHED
... `connect()`
returns

ACK y+1

ESTABLISHED

... `accept()`
returns

+data

... eventually,
`send()`

When the initial sequence number is known, it is easy to forge a connection.

Attacker

Target Server

(From: Forged IP of Trusted Client)
SYN

(From: Forged IP of Trusted Client)
ACK (Guess the ISN of server)

Evil commands

SYN
ACK

**Trusted
Client**
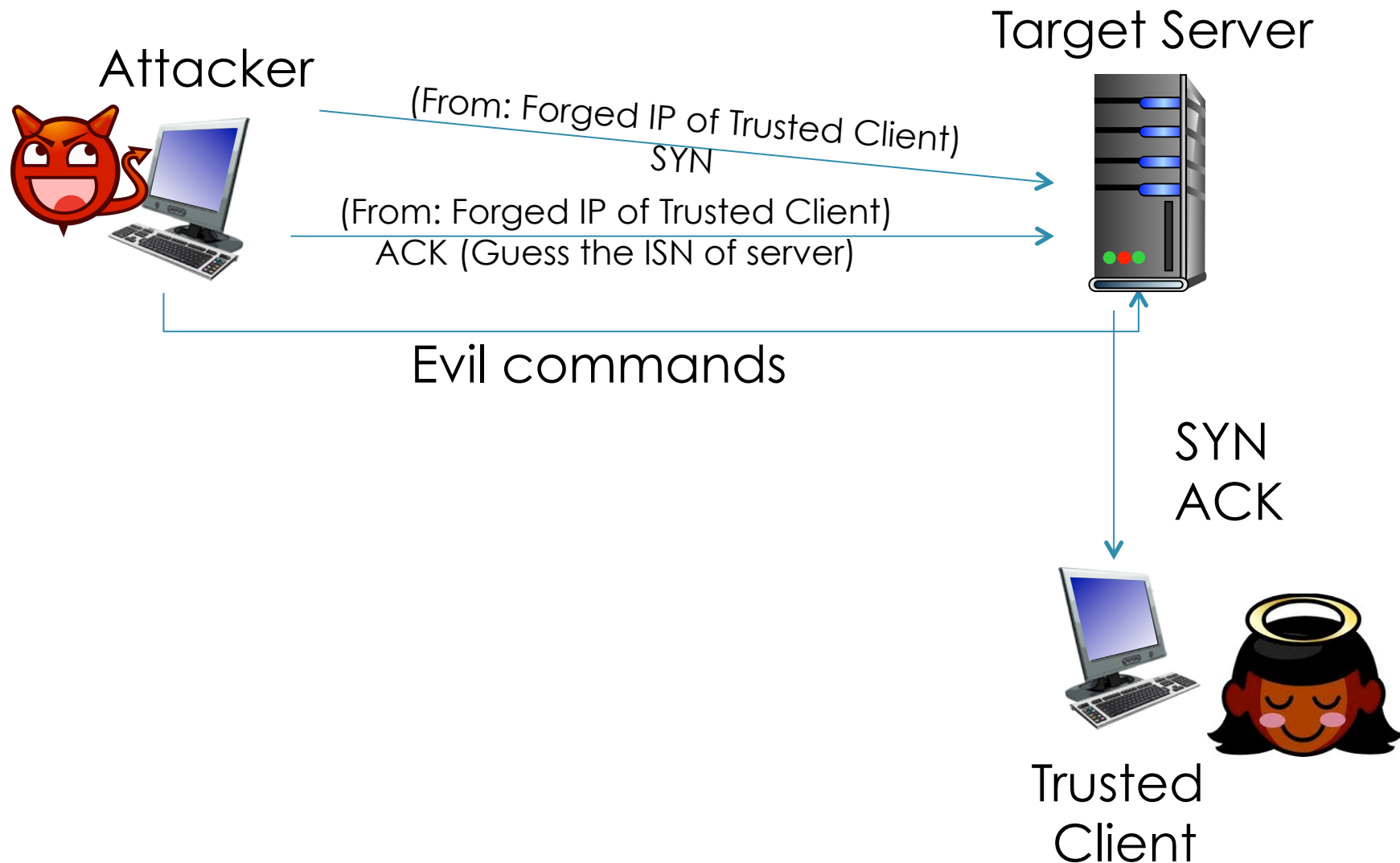
TCP **connection teardown** cleans up state, can be initialized by either host.

Initiator | Receiver

ESTABLISHED
connection

close() *active close*
FIN_WAIT_1

FIN →

ESTABLISHED
connection

CLOSE_WAIT
passive close

ACK

FIN_WAIT_2

FIN

LAST_ACK close()

TIME_WAIT

ACK

CLOSED

CLOSED