Goal: predict the RTT  Lecture 17

EWMA: Exponentially weighted Moving Average

$S_1 \; S_2 \; S_3 \; S_4 \; \ldots \; \ldots \; S_n$

Simple average: $\dfrac{\sum_{i=1}^{n} S_i}{n}$

Moving Average: $\dfrac{\sum_{i=1}^{w} S_i}{n}$  w is the size of window

EWMA: $S_{av}^{n} = \alpha S_{av}^{n-1} + (1-\alpha) S_n \quad 0 < \alpha < 1$

($S_{av}^{n-1}$: history; $S_n$: new sample)

$$S_{av}^{1} = \alpha \cdot S_{av}^{I} + (1-\alpha) S_1$$

$S_{av}^{I}$: initial value of the history

$$S_{av}^{2} = \alpha S_{av}^{1} + (1-\alpha) S_2$$

$$= \alpha\left[\alpha S_{av}^{I} + (1-\alpha) S_1\right] + (1-\alpha) S_2$$

$$= \alpha^2 S_{av}^{I} + \alpha(1-\alpha) S_1 + (1-\alpha) S_2$$

$$S_{av}^{3} = \alpha^3 S_{av}^{I} + \alpha^2 (1-\alpha) S_1 + \alpha(1-\alpha) S_2 + (1-\alpha) S_3$$

$$S_{av}^{n} = \alpha^n S_{av}^{I} + \alpha^{n-1} (1-\alpha) S_1 + \ldots$$

$0 < \alpha < 1$

↓ contribution of $S_1$ is decaying exponential as we get more samples

**EWMA** : $\alpha^n(1-\alpha)S_1 + \alpha^{n-1}(1-\alpha)S_2 + \cdots + (1-\alpha)S_n$

1) We have samples of RTT
   - timestamp segments
   - Find the time when the ACK is received

   → $RTT_n$

Smothed RTT : SRTT

$$SRTT_n = (1-\alpha)\, SRTT_{n-1} + \alpha\, RTT_n$$

$SRTT_{n-1}$ ← history; $\alpha\, RTT_n$ ← current sample

$\alpha$ is close to 1 ⇒ more importance to the current sample

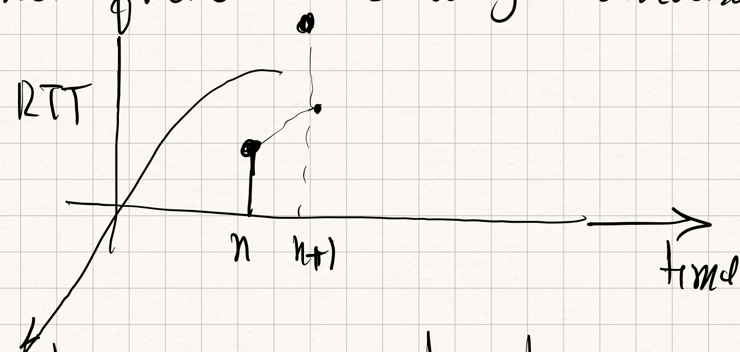
$\alpha$ is close to 0 ⇒ more importance to the history

→ <u>Whenever an RTT sample</u> is received we calculate the SRTT

⇒ set the timeout proportional to SRTT

In early 1980s this was implemented but it <u>did not</u> work

SRTT does not quite track large variations in RTT




large changes occur due to queueing delays

$devRTT_n$ : deviation in the RTT when the $n^{th}$ sample is received

$$devRTT_n = (1-\beta)\, devRTT_{n-1} + \beta\,(RTT_n - SRTT_n)$$

- $(1-\beta)$, $\beta$ → Exponential averaging
- $devRTT_{n-1}$ → history of deviation
- $(RTT_n - SRTT_n)$ → current sample of the deviation

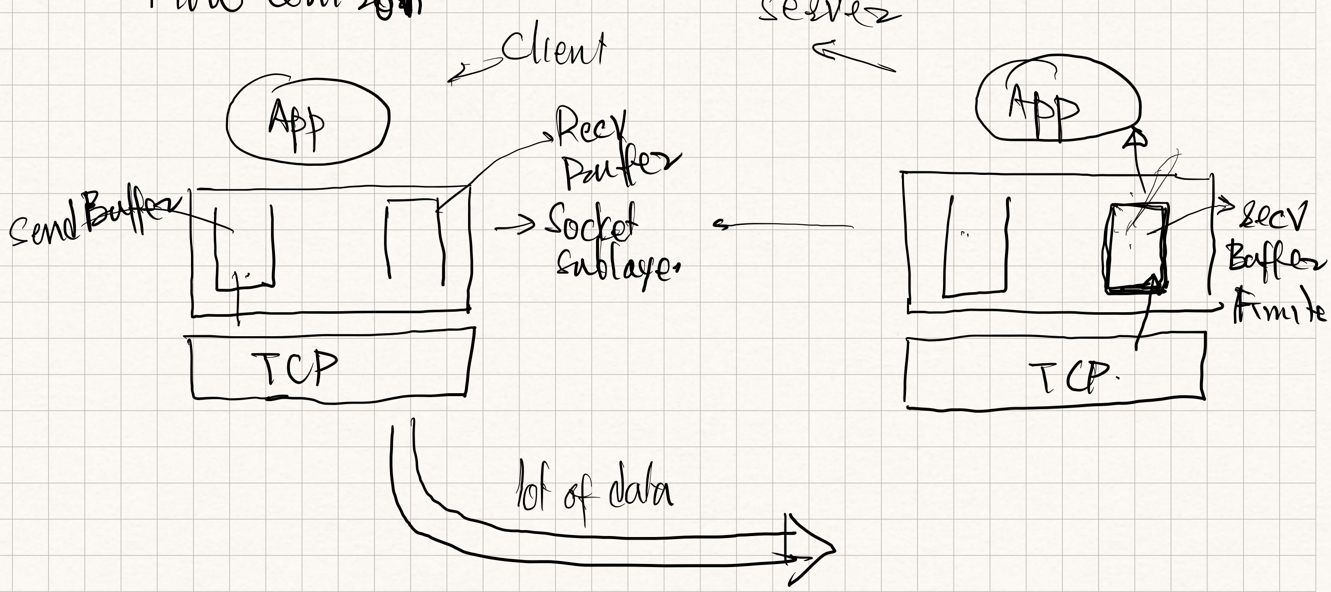
Timeout value: $SRTT_n + 4 \cdot devRTT_n$
(RTO)

$\alpha$ is small ~ 0.125 1/8
$\beta$ is large ~ 0.875 (?)

## TCP: Transmission Control Protocol

Congestion Control

Flow Control



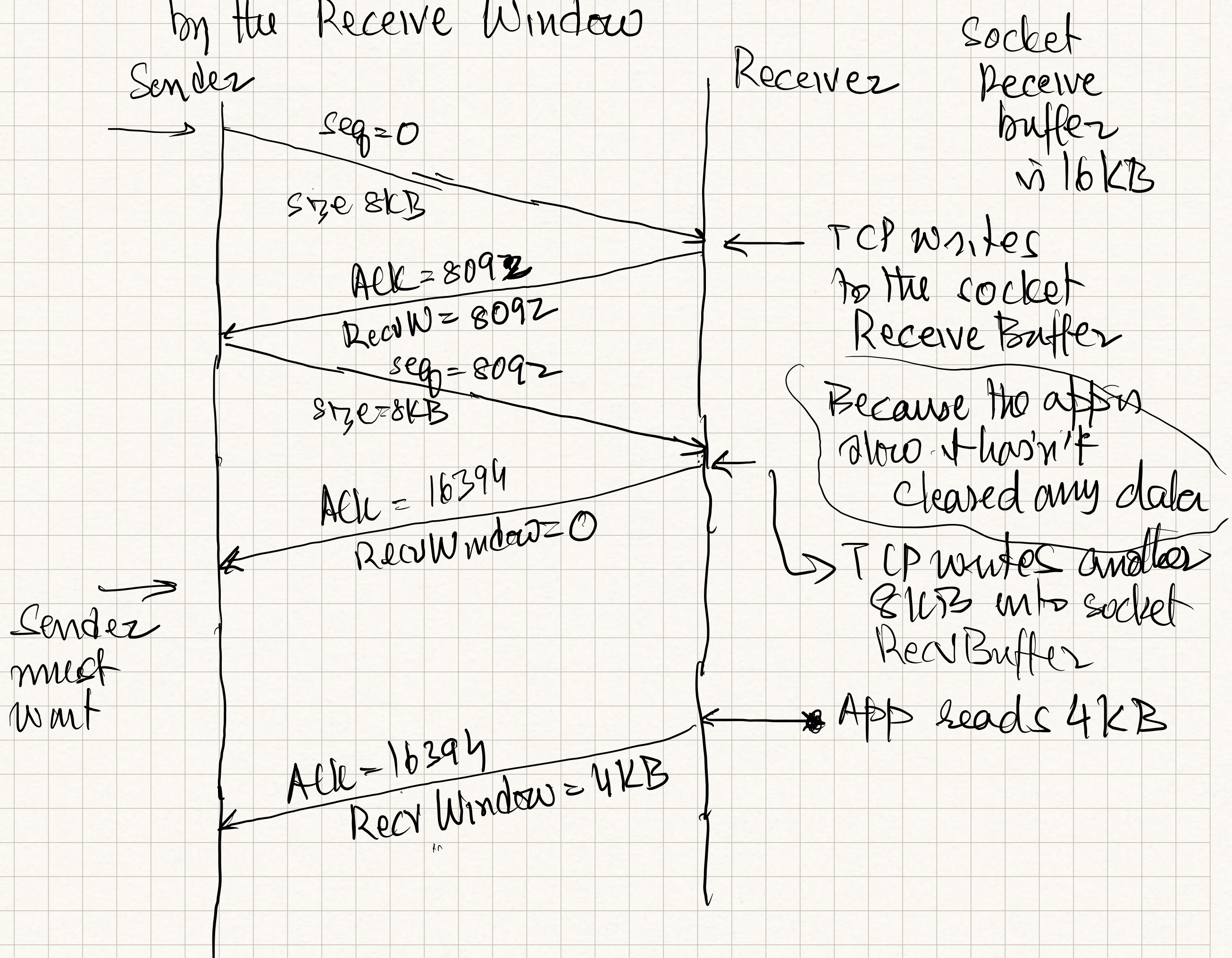

Client side is fast
Network is fast
Server side App is slow

Flow control is a method to **pace the sender**
⇒ **end-to-end** between the TCP sender and TCP receiver

## Method:

TCP header
Receive Window

- When the receiver sends ACK it also sends the remaining space in the socket Receive buffer to the sender → in the Receive Window field
- The sender's window size is capped by by the Receive Window



## Congestion Control

→ "Flow control" between the sender and the network

→ How fast should the sender send?

⇒ Depends on congestion in the network

⇒ How is congestion in the network detected?

Internet is a distributed system
We donot have a central controller that can tell each sender what rate to send data

Each sender must determine on its own

(A): How is congestion detected?

- loss based
- delay based

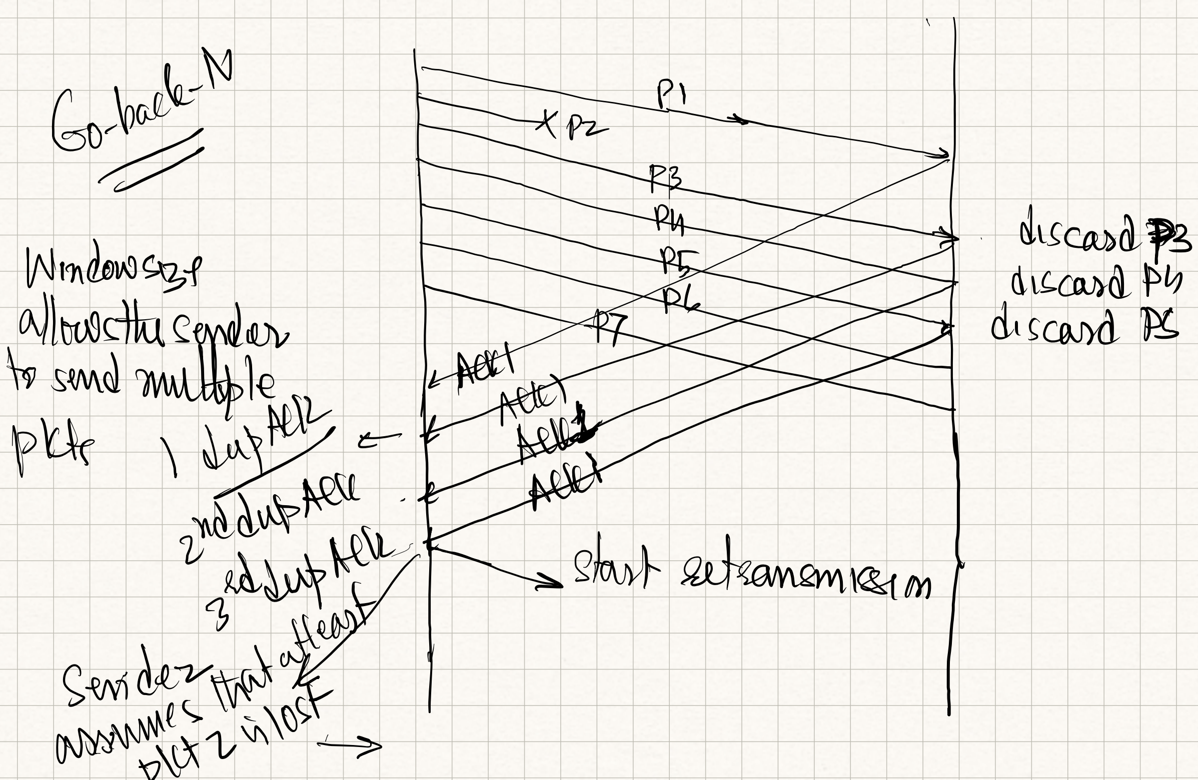
**loss based:** If a pkt is lost then the sender determines that the network is congested

how do we know that a pkt is lost?

⇒ when we have a timeout

Typically, timeouts are large. And there is another method to determine packet loss

**Go-back-N**




3 dupACKs is another way to determine pkt loss

pkt loss is a signal that the network
is congested

→ delay based signals of congestion

If we know the base RTT

increase beyond the base RTT is
a signal that the network is congested