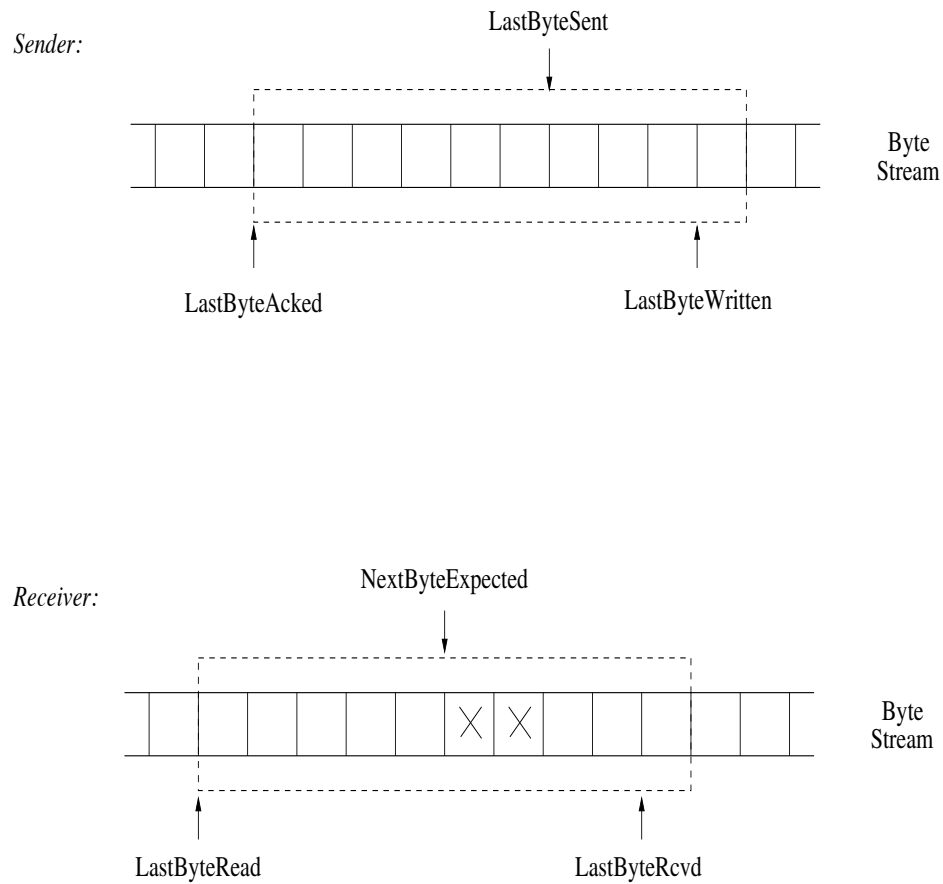


TCP's sliding window protocol:



- sliding window with 1-byte data unit
 - sender and receiver buffers
- `MaxSendBuffer`, `MaxRcvBuffer`

Recall: receiver's feedback includes **AdvertisedWindow**

→ conveys space available at receiver buffer

Limit how much sender can send:

$$\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAcked})$$

Sender may choose to send even less: self-throttling

→ called congestion control

→ **CongestionWindow**

EffectiveWindow update procedure:

$$\text{EffectiveWindow} = \text{MaxWindow} - (\text{LastByteSent} - \text{LastByteAcked})$$

where

$$\text{MaxWindow} = \min\{\text{AdvertisedWindow}, \text{CongestionWindow}\}$$

How to set CongestionWindow?

→ TCP congestion control

Issues:

How to let sender know of change in receiver window size after `AdvertisedWindow` becomes 0?

- trigger ACK event on receiver side when `AdvertisedWindow` becomes positive
- sender periodically sends 1-byte probing packet

→ design choice: smart sender/dumb receiver

→ same situation for congestion control

Silly window syndrome: Assuming receiver buffer is full, what if application reads one byte at a time with long pauses?

- can cause excessive 1-byte traffic
- if `AdvertisedWindow < MSS` then set
`AdvertisedWindow ← 0`

Do not want to send too many 1 B payload packets.

Nagle's method:

- rule: connection can have only one such unacknowledged packet outstanding
- while waiting for ACK, incoming bytes are accumulated (i.e., buffered)

→ compromise between real-time constraints and efficiency

→ facilitates interactive applications

RTT estimation:

→ important to not underestimate nor overestimate

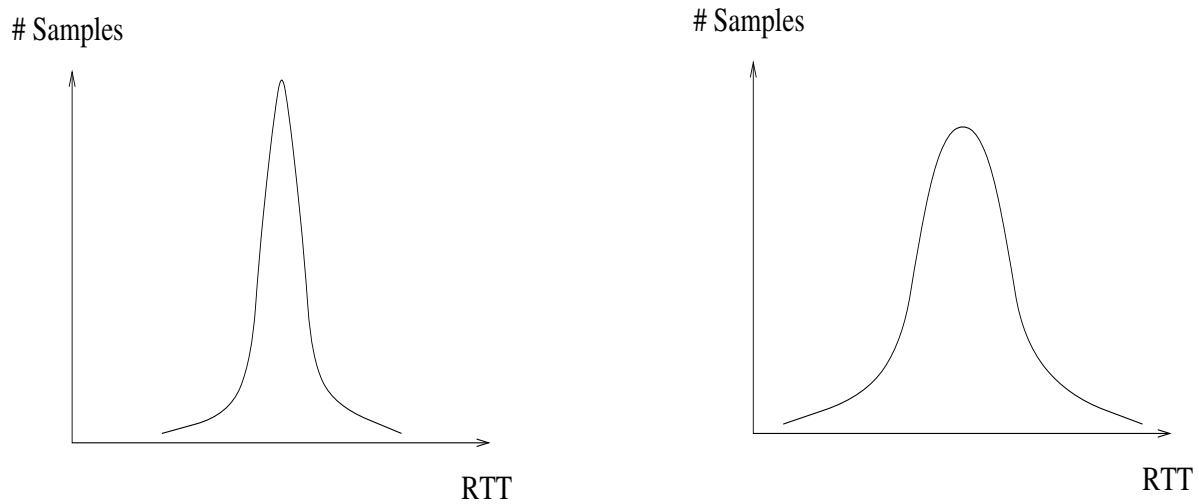
Karn/Partridge: maintain running average with precautions

$$\text{EstimateRTT} \leftarrow \alpha \cdot \text{EstimateRTT} + \beta \cdot \text{SampleRTT}$$

- **SampleRTT** computed by sender using timer
- $\alpha + \beta = 1$; $0.8 \leq \alpha \leq 0.9$, $0.1 \leq \beta \leq 0.2$
- **TimeOut** $\leftarrow 2 \times \text{EstimateRTT}$ or
TimeOut $\leftarrow 2 \times \text{TimeOut}$ (if retransmit)

→ need to be careful when estimating **SampleRTT**

Issue of variance:



→ need to account for variance

→ real-world: more messy

Jacobson/Karels:

- $\text{Difference} = \text{SampleRTT} - \text{EstimatedRTT}$
- $\text{EstimatedRTT} = \text{EstimatedRTT} + \delta \times \text{Difference}$
- $\text{Deviation} = \text{Deviation} + \delta \times (|\text{Difference}| - \text{Deviation})$

Here $0 < \delta < 1$.

Then

- $\text{TimeOut} = \mu \times \text{EstimatedRTT} + \phi \times \text{Deviation}$

where $\mu = 1, \phi = 4$.