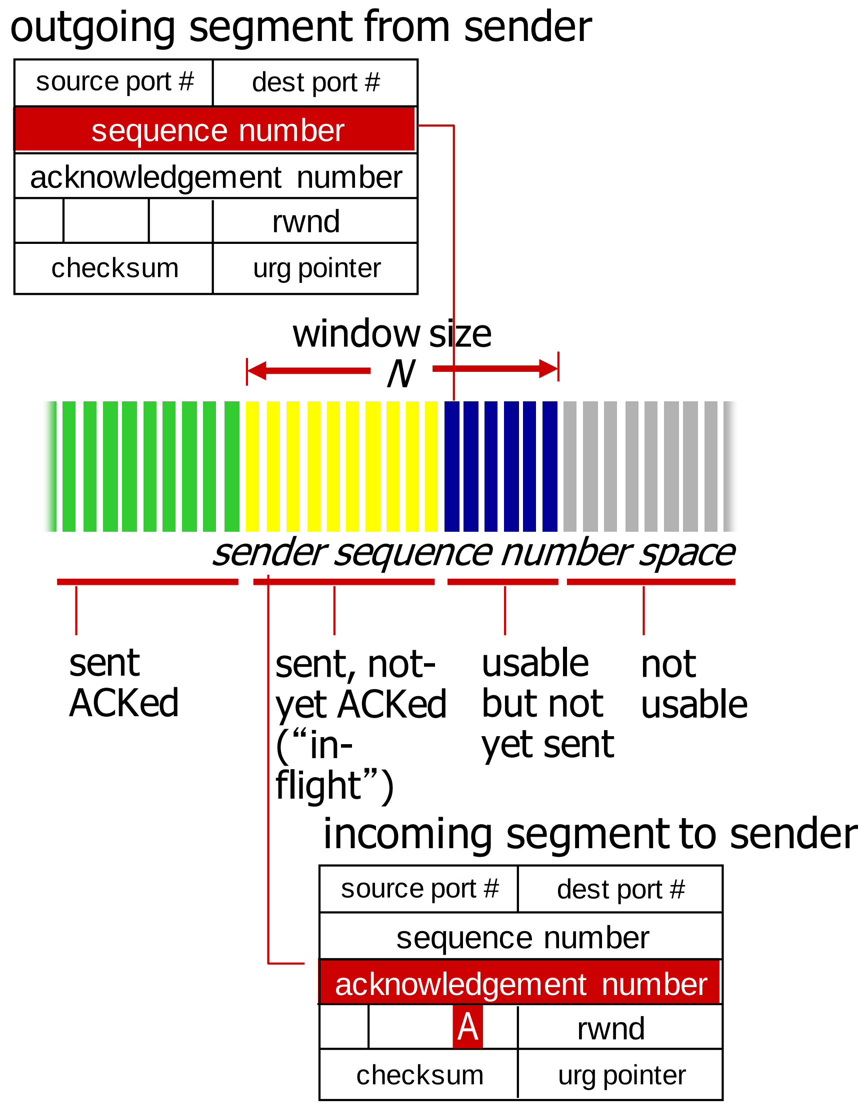
TCP seq. numbers, ACKs

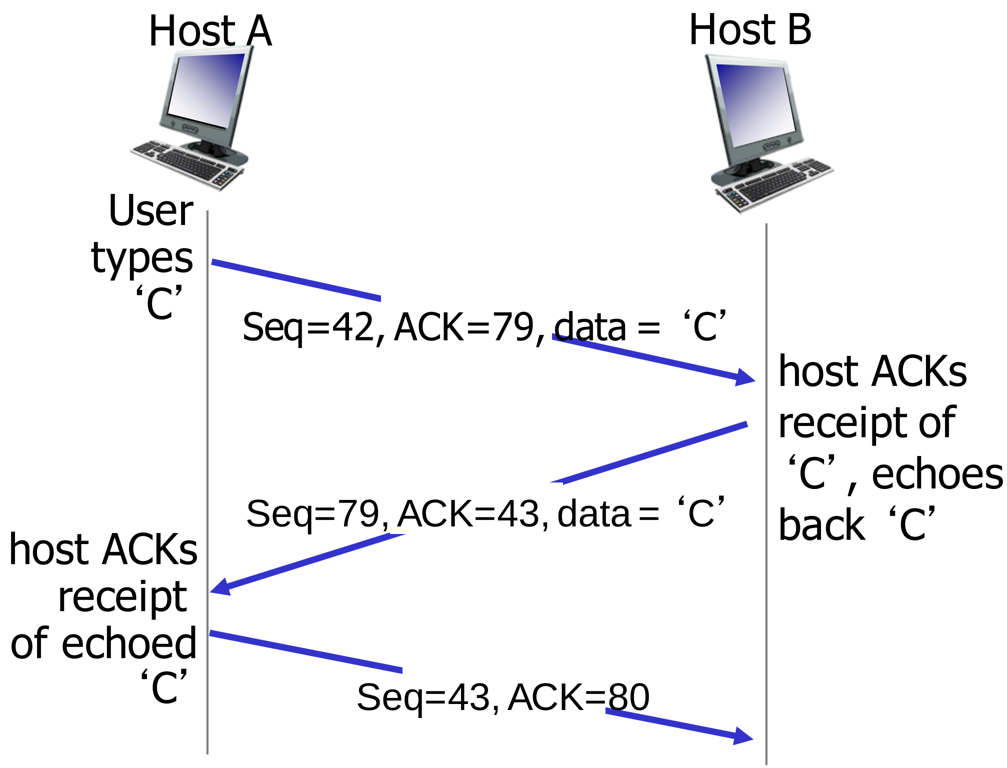
sequence numbers:
- byte stream "number" of first byte in segment's data

acknowledgements:
- seq # of next byte expected from other side
- cumulative ACK

Q: how receiver handles out-of-order segments
- A: TCP spec doesn't say, - up to implementor



TCP seq. numbers, ACKs



simple telnet scenario

TCP round trip time, timeout

- timeout interval: **`EstimatedRTT`** plus "safety margin"
  - large variation in **`EstimatedRTT`** -> larger safety margin
- estimate SampleRTT deviation from EstimatedRTT:

```
DevRTT = (1-β)*DevRTT +
                β*|SampleRTT-EstimatedRTT|

                (typically, β = 0.25)
```

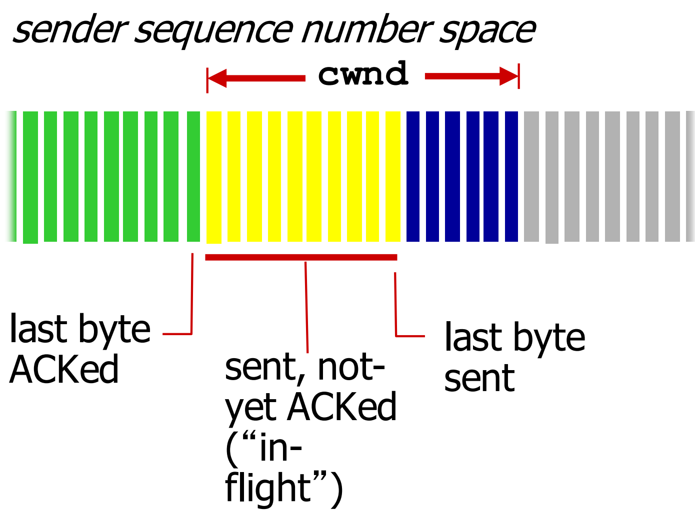
```
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

estimated RTT

"safety margin"

Retransmissions excluded from TimeoutInterval computation

TCP Congestion Control: details



- sender limits transmission:

$$\texttt{LastByteSent-LastByteAcked} \leq \texttt{cwnd}$$

- **cwnd** is dynamic, function of perceived network congestion

TCP sending rate:

- *roughly:* send cwnd bytes, wait RTT for ACKS, then send more bytes

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

Additive Increase/Multiplicative Decrease

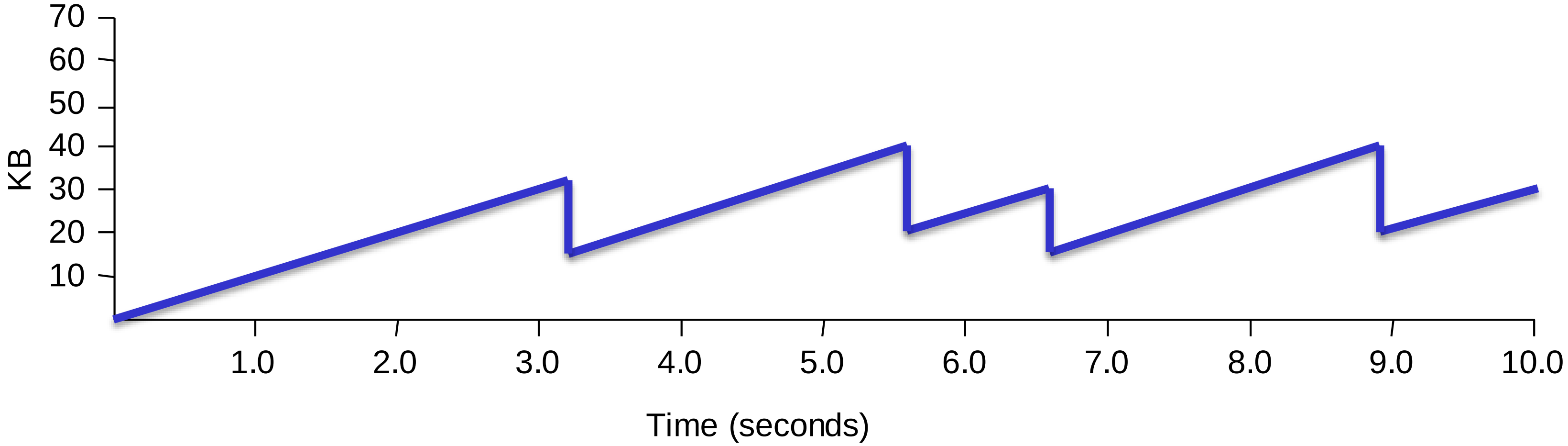
- Objective: adjust to changes in the available capacity
- New state variable per connection: **`CongestionWindow`**
  - limits how much data source has in transit

```
MaxWin = MIN(CongestionWindow, AdvertisedWindow)
EffWin = MaxWin - (LastByteSent - LastByteAcked)
```

- Idea:
  - increase **`CongestionWindow`** when congestion goes down
  - decrease **`CongestionWindow`** when congestion goes up

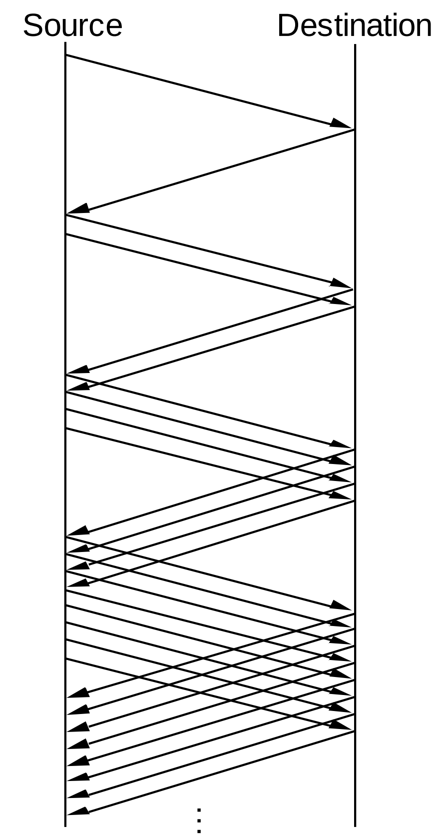
AIMD (cont)

- Trace: sawtooth shape behavior



Slow Start

- Objective: quickly determine the available capacity in the first
- Idea:
  - begin with **`CongestionWindow`** = 1 pckt
  - double **`CongestionWindow`** each RTT (increment by 1 packet for each ACK)
  - This is exponential increase to probe for available bandwidth
  - <u>Up to half of cwnd may get lost</u>
- Used…
  - when first starting connection
  - when connection goes dead waiting for timeout
- SSTHRESH (slow start threshold) indicates when to begin additive increase phase



SSTHRESH and CWND

- SSTHRESH typically very large on connection setup
- Set to one half of `CongestionWindow` on packet loss
  - So, SSTHRESH goes through multiplicative decrease for each packet loss
  - If loss is indicated by timeout, set `CongestionWindow = 1`
    - SSTHRESH and `CongestionWindow` always >= 1 MSS
- After loss, when new data is ACKed, increase CWND
  - Manner depends on whether we're in slow start or congestion avoidance

Congestion Control Functionality

Until cwnd ≤ slow_start_threshold

- Slow Start phase (exponential growth)
  - Each returning ACK, a new pckt is transmitted
    - cwnd -> cwnd + 1
  - Every RTT
    - cwnd -> 2 cwnd

When cwnd > slow_start_threshold

- Congestion avoidance phase (linear growth)
  - Each returning ACK, a new pckt is transmitted
    - cwnd -> cwnd + (1/cwnd)
  - Every RTT
    - cwnd -> cwnd + 1

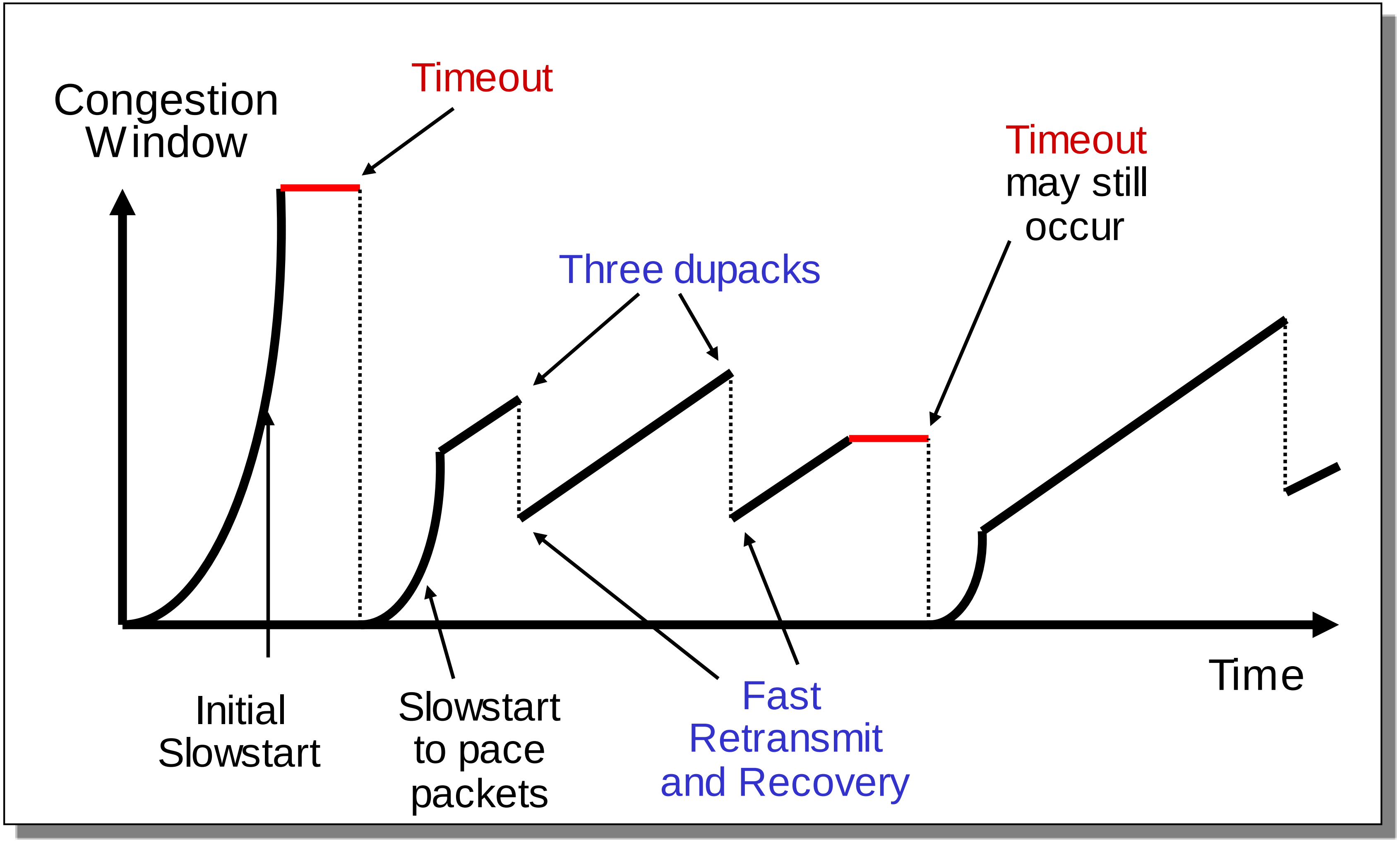
Loss recovery

- Two ways to detect losses
  - Time outs
  - Three dupacks

- With timeout expiration
  - ssthresh = cwnd / 2
  - cwnd = 1 (so, restart in slow start phase)

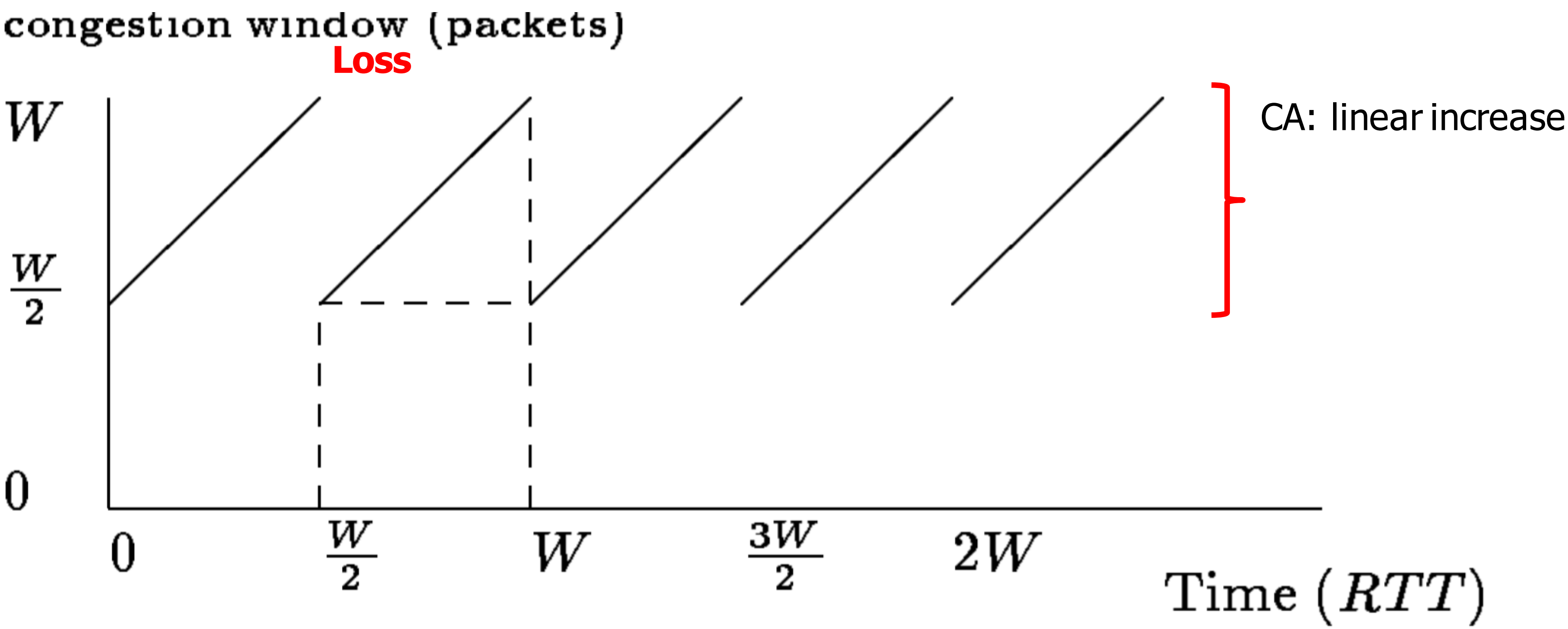
- With three dupacks
  - ssthresh = cwnd / 2
  - cwnd = cwnd / 2 (so, restart in cong. avoidance phase)

TCP Saw Tooth

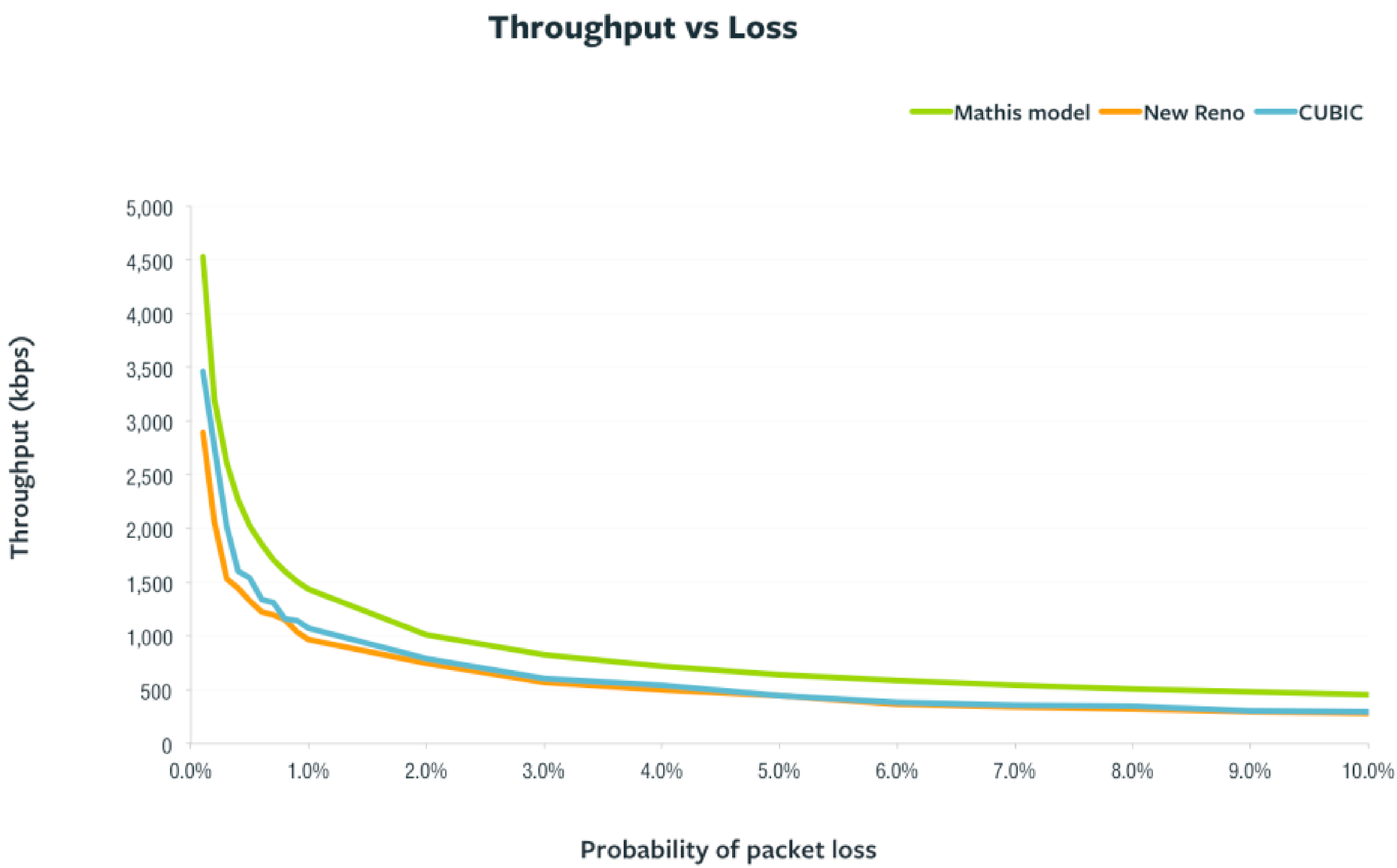


Mathis et. al, 1997 – Macroscopic TCP Throughput Estimation

W → max cwnd
p → periodic loss prob. at end. cycle
MSS → Maximum segment size



Model Validation



Approaches towards congestion control

two broad approaches towards congestion control:

end-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

network-assisted congestion control:

- routers provide feedback to end systems
  - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
  - explicit rate for sender to send at

Case study: ATM ABR congestion control

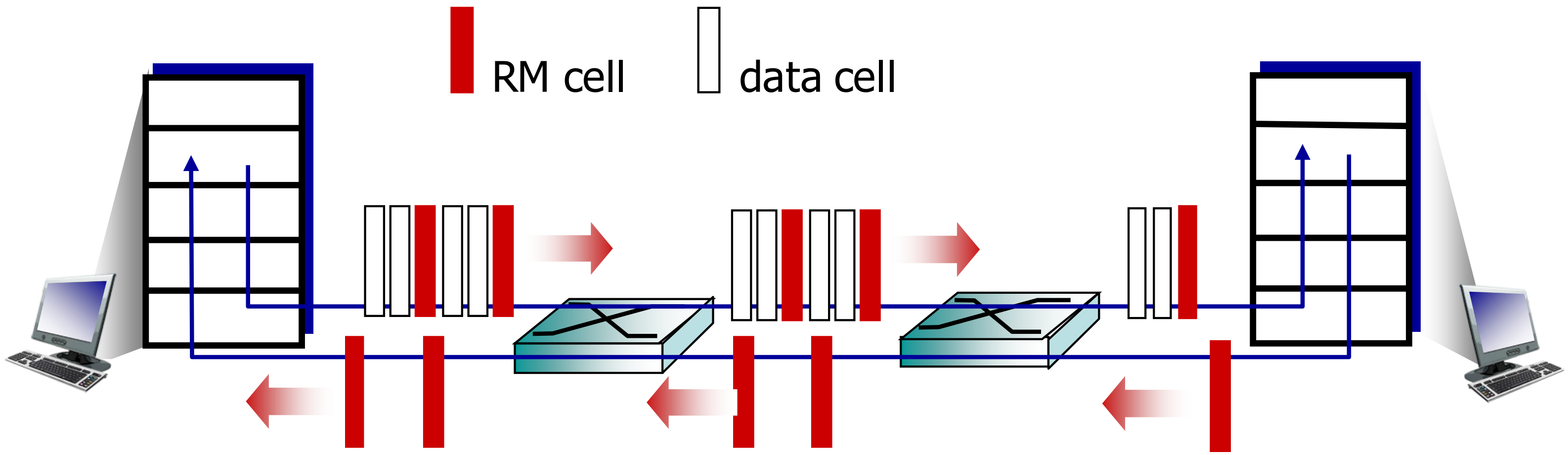
ABR: available bit rate:

- "elastic service"
- if sender's path "underloaded":
  - sender should use available bandwidth
- if sender's path congested:
  - sender throttled to minimum guaranteed rate

RM (resource management) cells:

- sent by sender, interspersed with data cells
- bits in RM cell set by switches ("*network-assisted*")
  - *NI bit:* no increase in rate (mild congestion)
  - *CI bit:* congestion indication
- RM cells returned to sender by receiver, with bits intact

Case study: ATM ABR congestion control



- two-byte ER (explicit rate) field in RM cell
  - congested switch may lower ER value in cell
  - senders' send rate thus max supportable rate on path
- EFCI bit in data cells: set to 1 in congested switch
  - if data cell preceding RM cell has EFCI set, receiver sets CI bit in returned RM cell