VU VRIJE UNIVERSITEIT AMSTERDAM

The Transport Layer

(Part II)

Balakrishnan Chandrasekaran

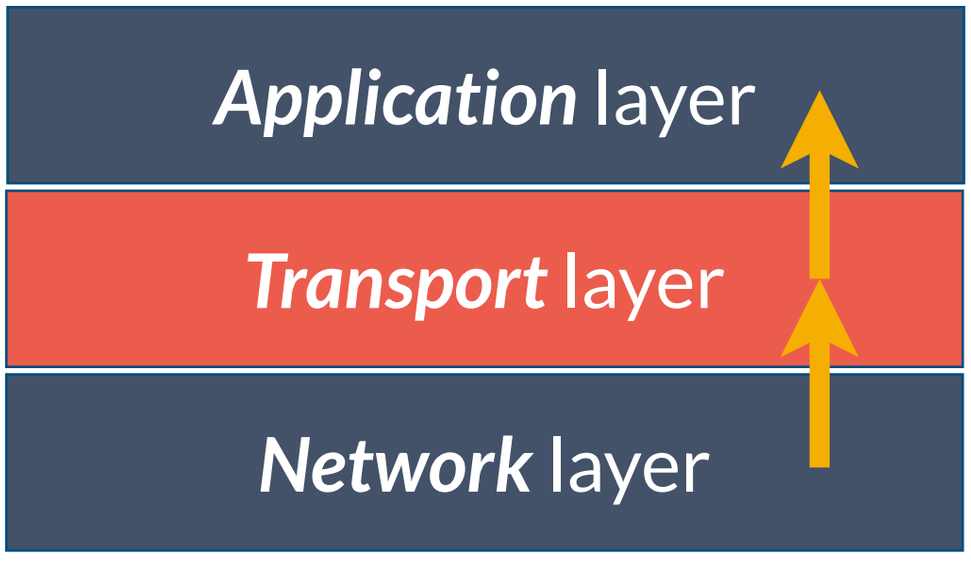
(Slides borrow content from lectures of Jennifer Rexford, Anja Feldmann, Bruce M. Maggs, and Jesse Donkervliet.)

Objectives

- Stream of Bytes Service
- RTT Estimation
- Flow Control
- Congestion Control

Recap

Transport layer



Offers various functionalities to the application

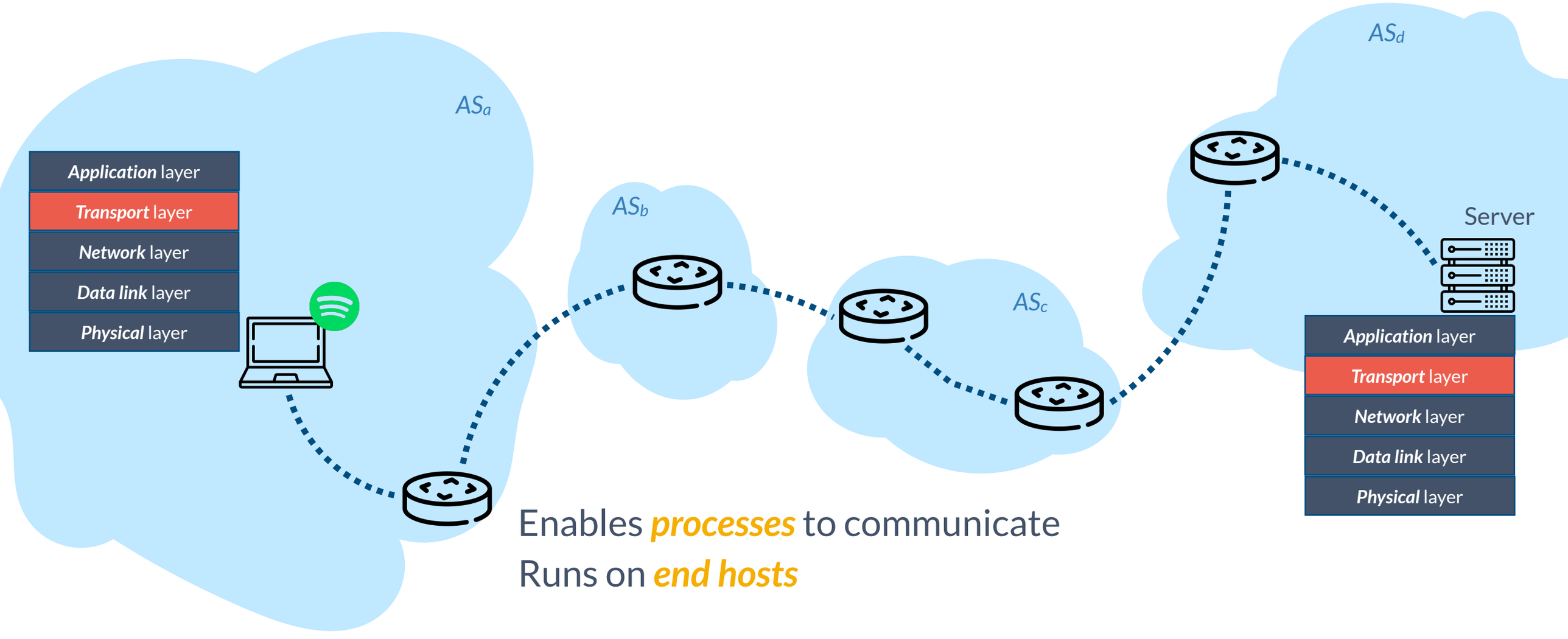
(reliable data transfer, congestion control, flow control, …)

Relies on *services* exposed by the network

(Network layer provides the path for the transport.)

Recap

End-to-End protocol



Enables ***processes*** to communicate

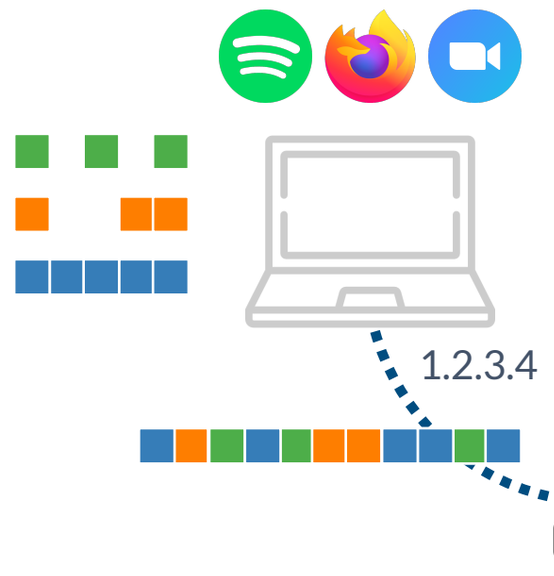
Runs on ***end hosts***

Recap

(De)Multiplexing traffic

- Different applications may generate different traffic workloads
  - Browsing: typically *bursty*

- *Statistical multiplexing*
  - Maximize *utilization* by multiplexing different applications' traffic



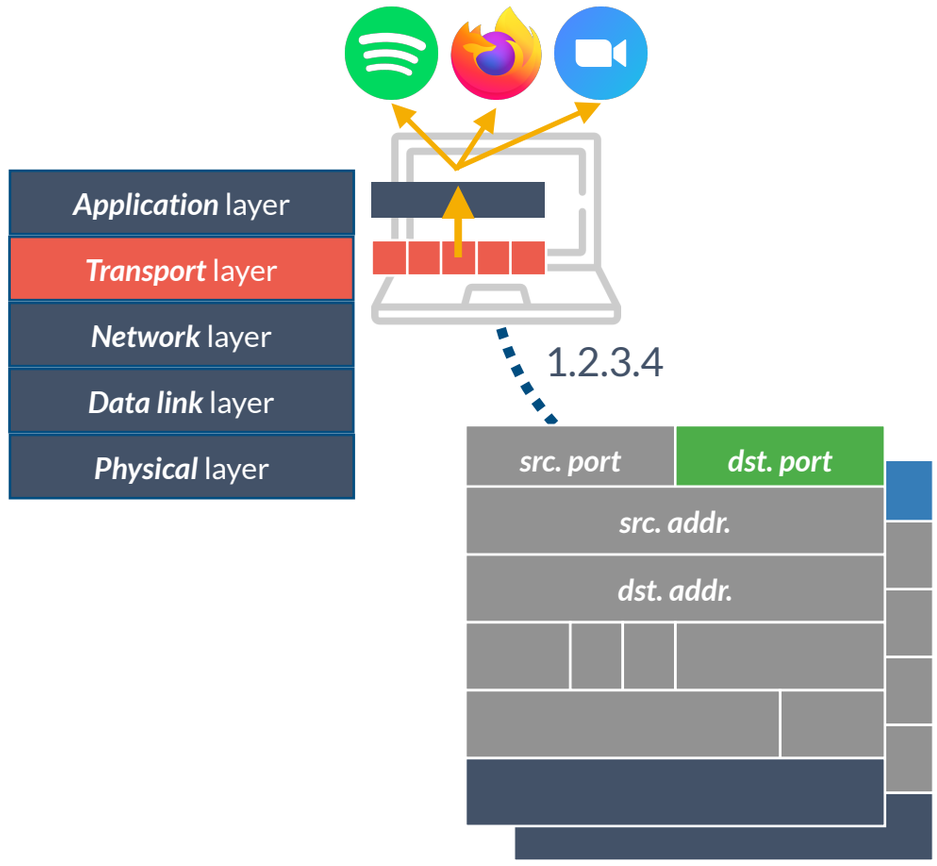
Recap

(De)Multiplexing traffic

- All applications running on the end host have the *same* IP address
  - Who should receive reassembled segments?
  - Transport layer needs more than IP addresses to identify *end points*

- Host receives IP *datagrams*, each of which
  - has a *source* and *destination* IP address
  - carries *one* transport *segment*

- Each *segment* has a *source* and *destination port* number

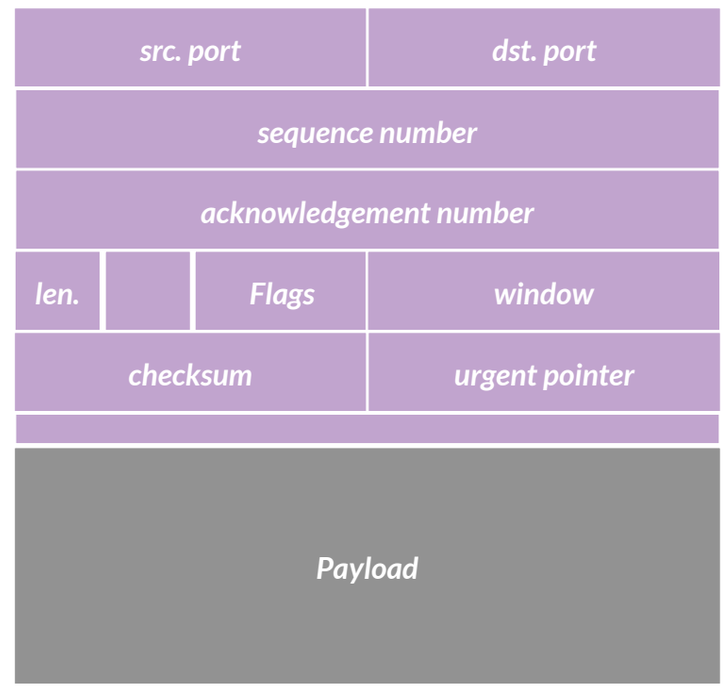


IP addresses and port numbers determine which application gets which segments.

Recap

Reliable data transfer

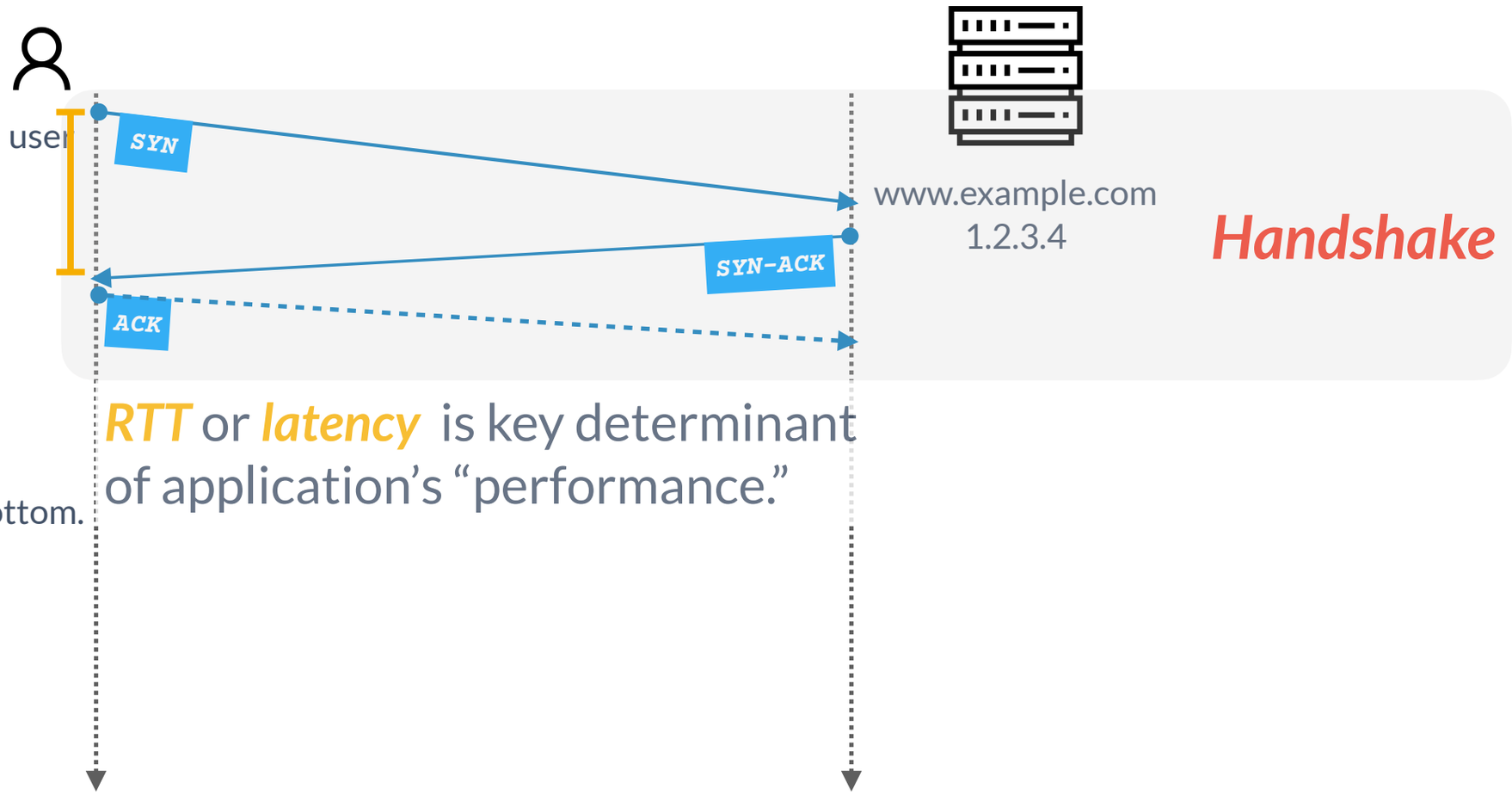
- *Transmission Control Protocol (TCP)*
  - Source and destination ports
  - Sequence and acknowledgement numbers
  - At least 20-bytes of header
- Rich protocol
  - *Connection-oriented*
  - *Stream-of-bytes* service
  - *Reliable*, *in-order* delivery
  - *Flow* control
  - *Congestion* control



Recap

TCP “Handshake”

Round-trip time (RTT)

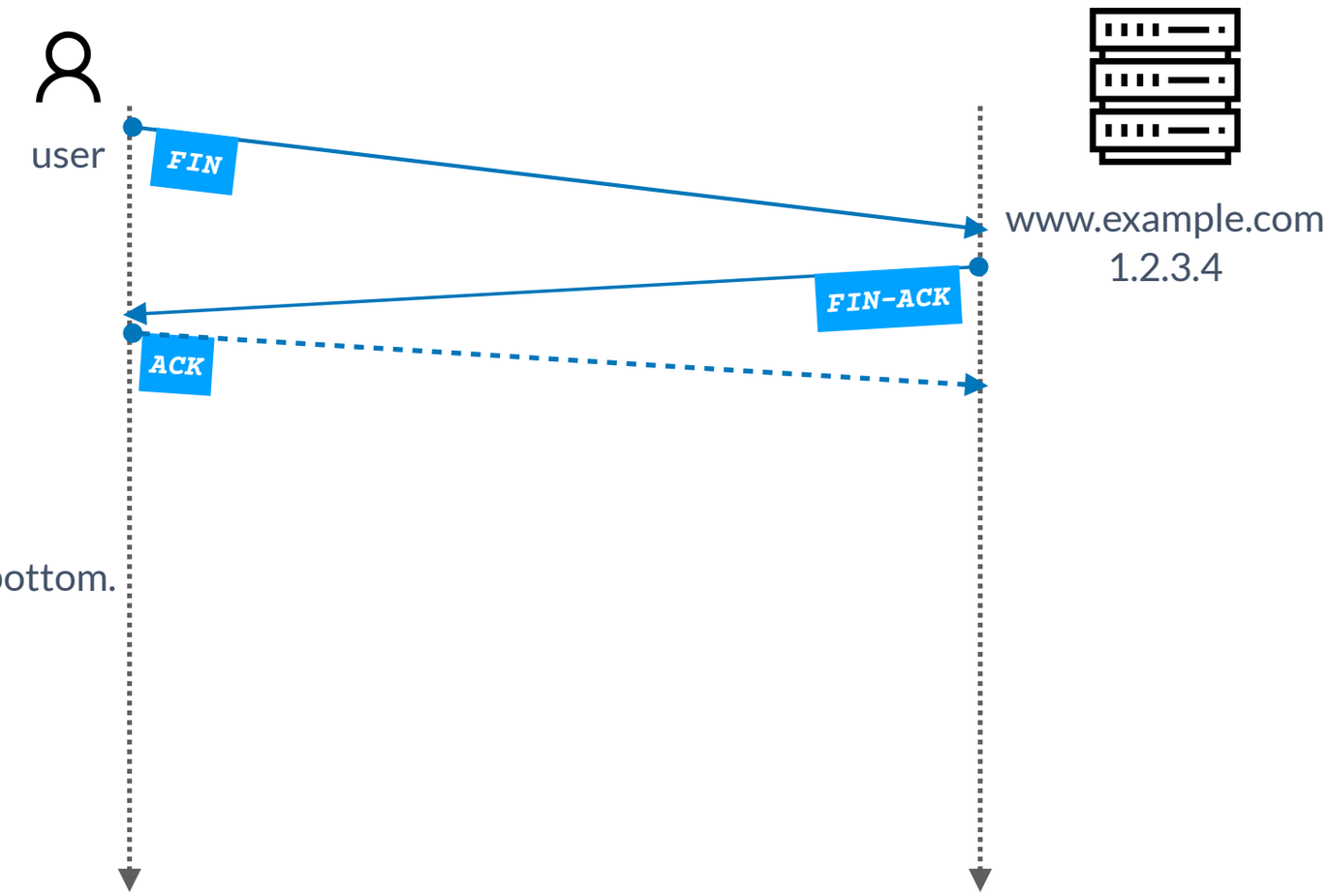


Time moves from top to bottom.

RTT or ***latency*** is key determinant of application’s “performance.”

Recap

Connection “tear down”

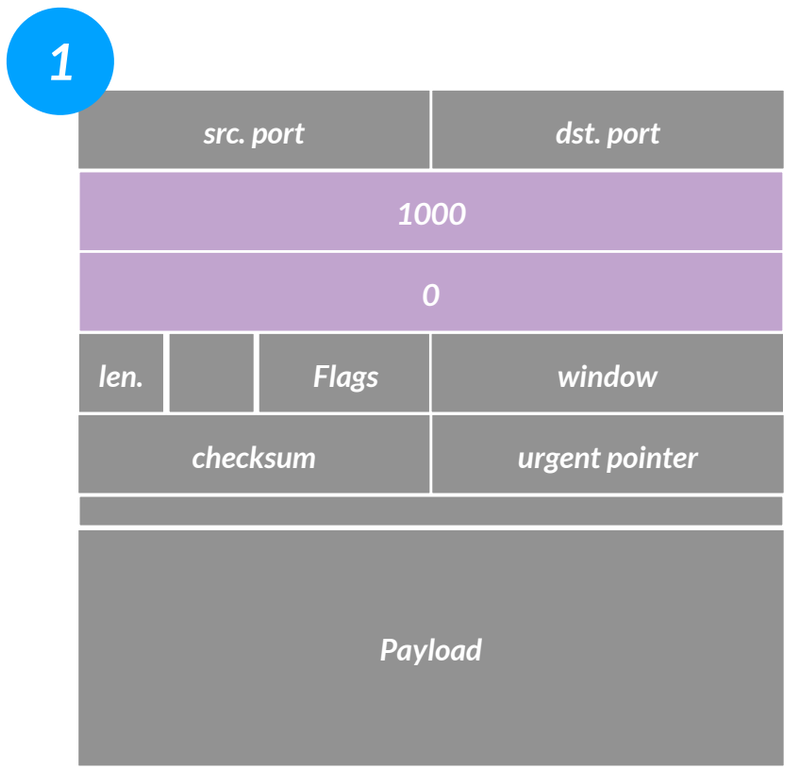


Time moves from top to bottom.

Recap

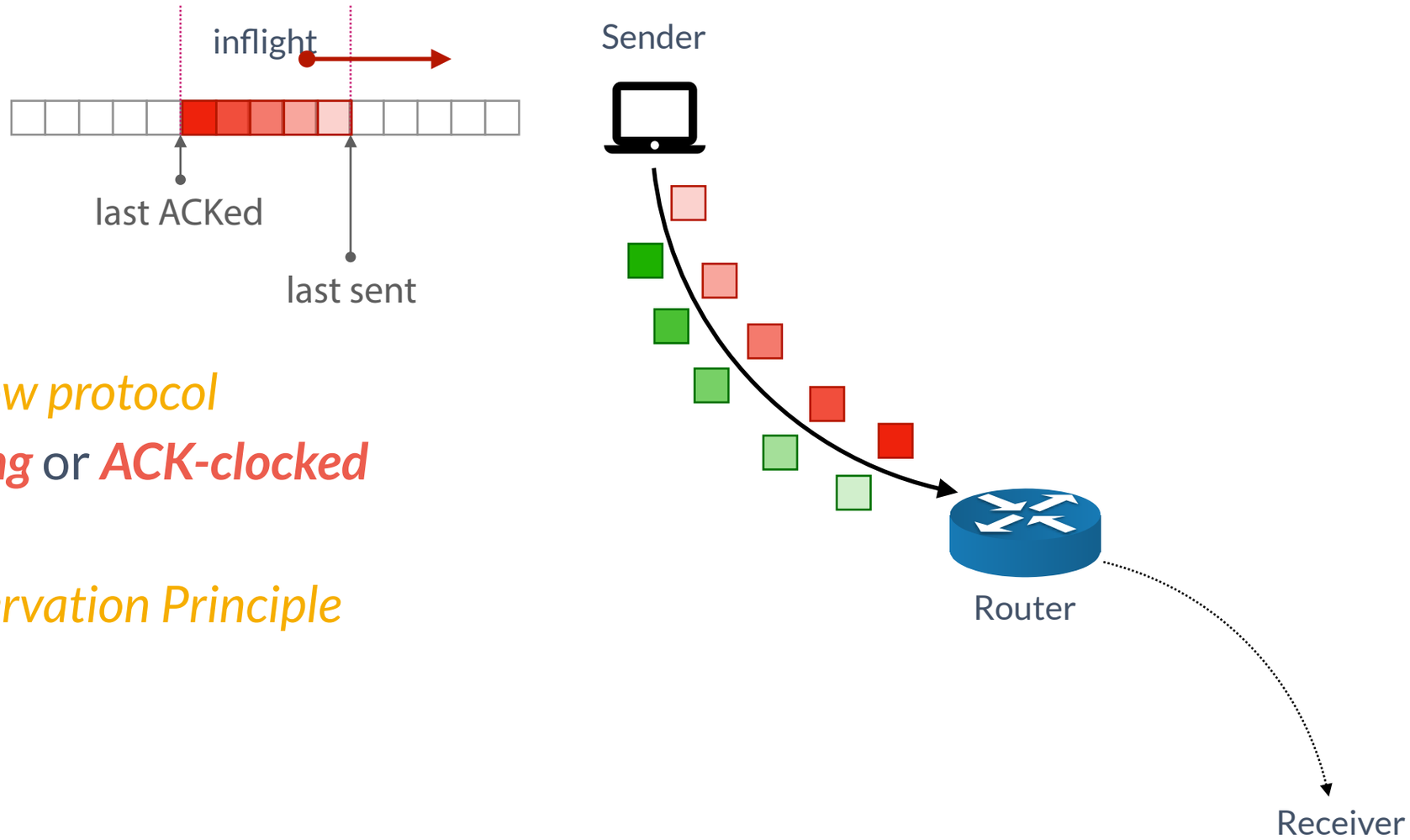
Stream of Bytes Service

- SYN
  - Sender's *initial* SEQ number
  - Pick a *random* number
  - Set *acknowledgement (ACK)* number to *zero*
- SYN-ACK
  - Receiver's *random* initial SEQ number
  - Set ACK number to *Sender's SEQ number + 1*



Recap

Sliding Window

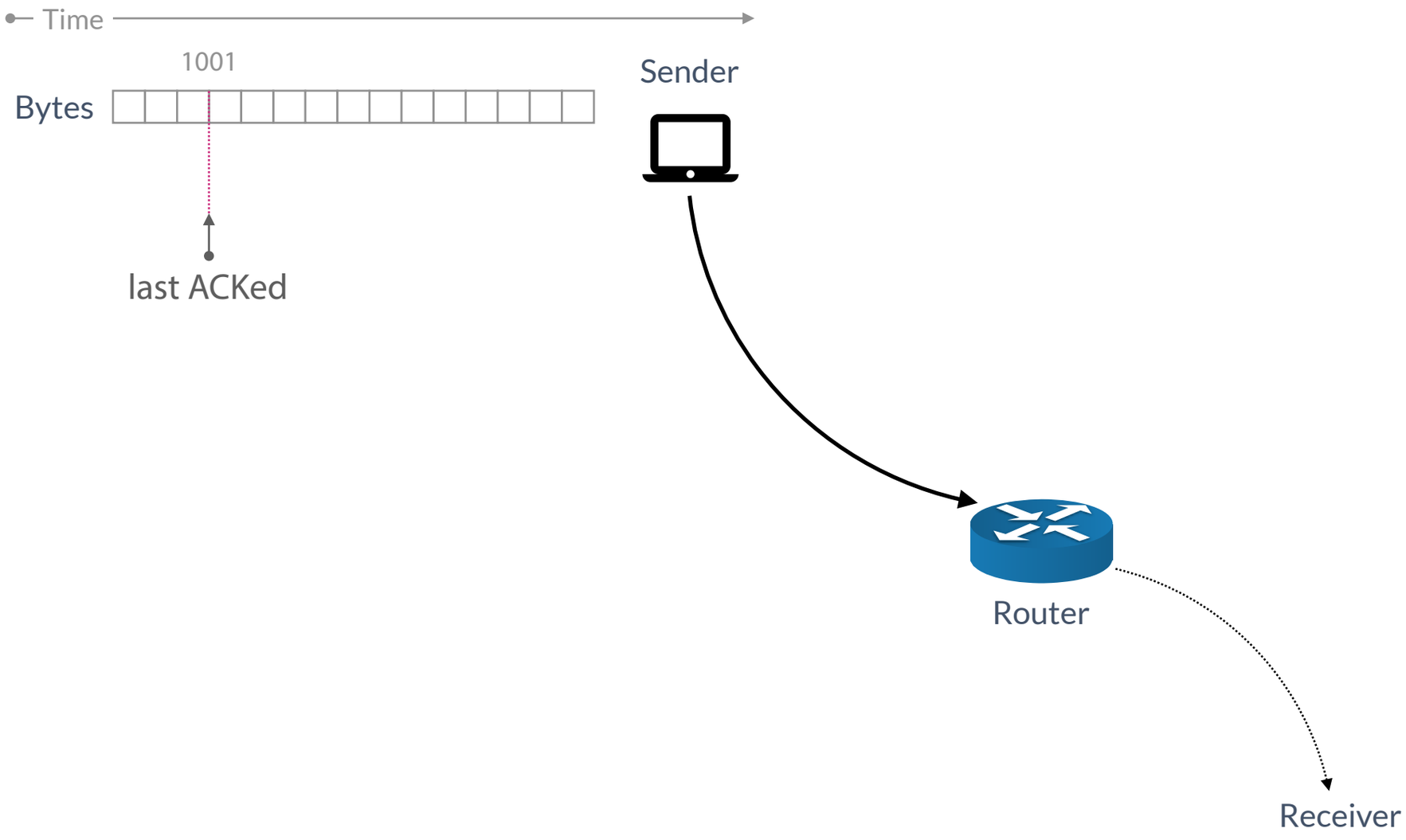


Sliding window protocol

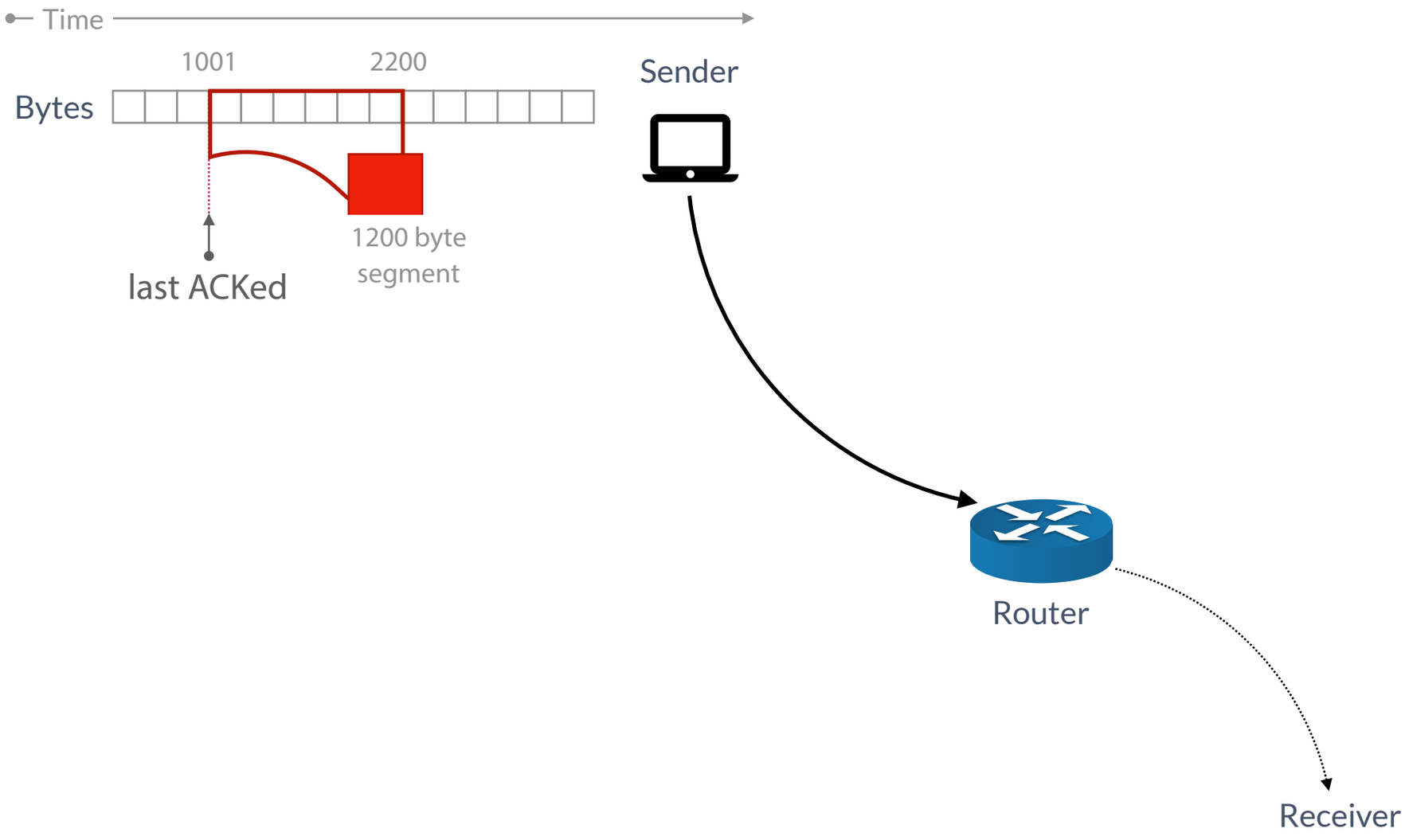
Self-clocking or ***ACK-clocked***

Packet Conservation Principle

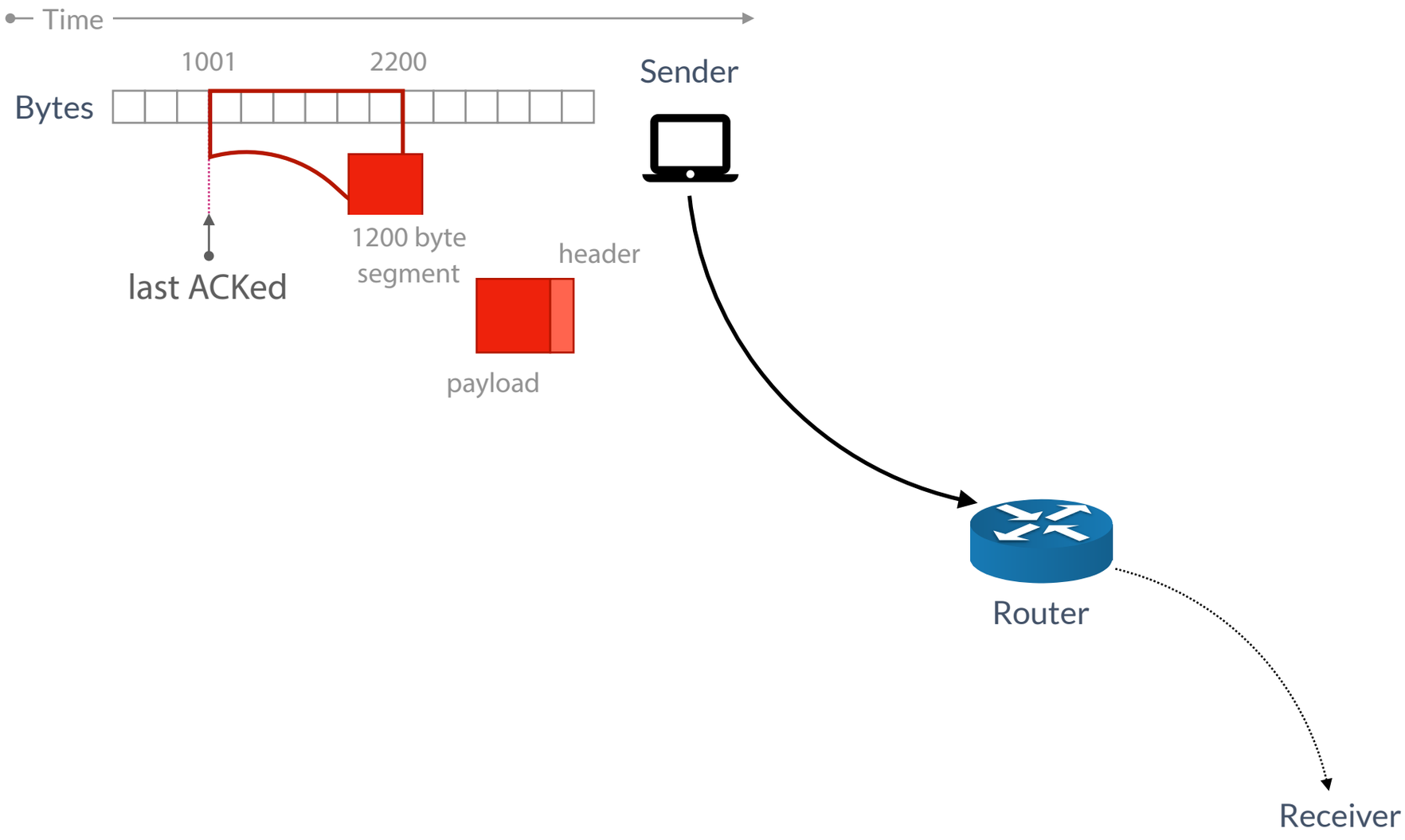
Stream of Bytes Service



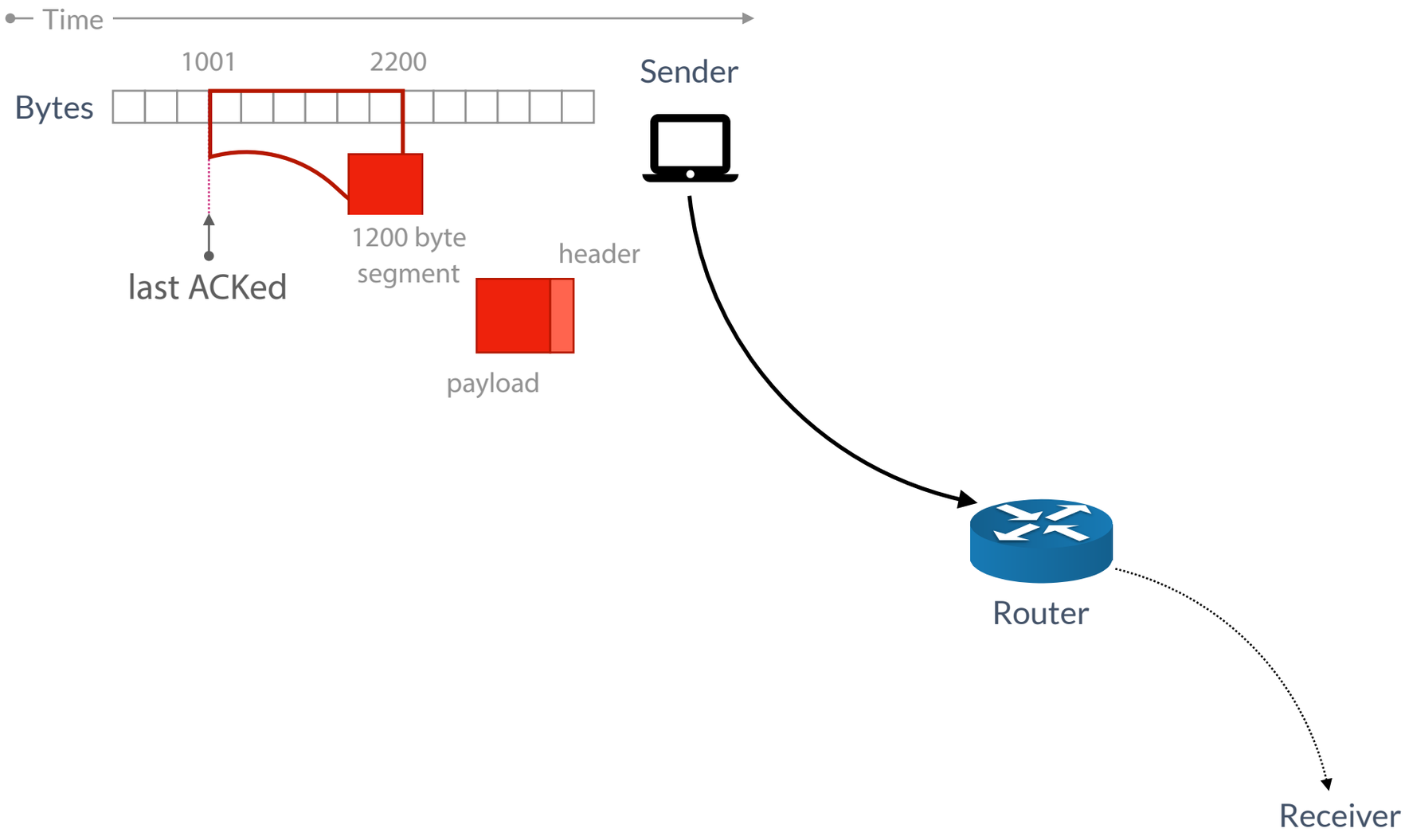
Stream of Bytes Service



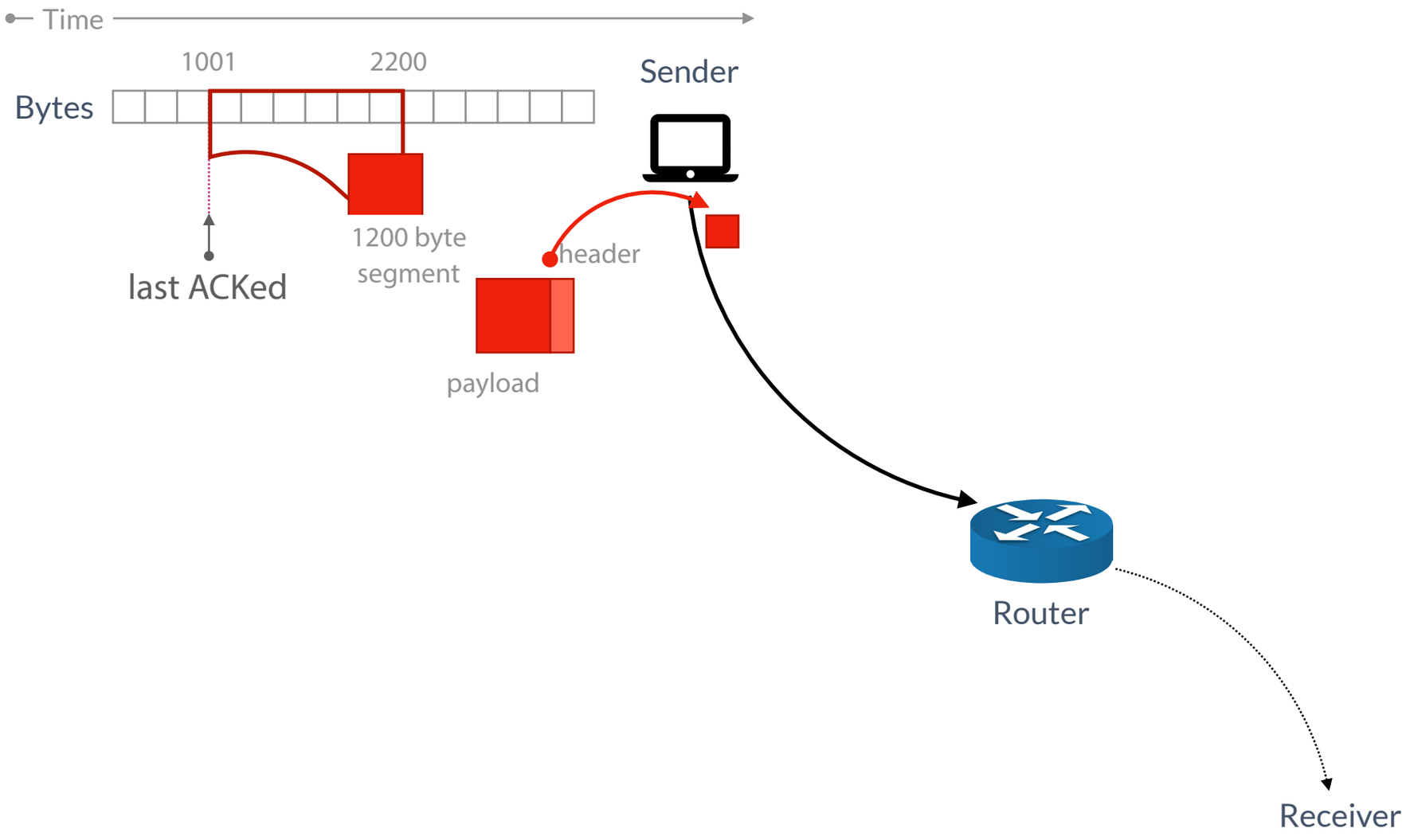
Stream of Bytes Service



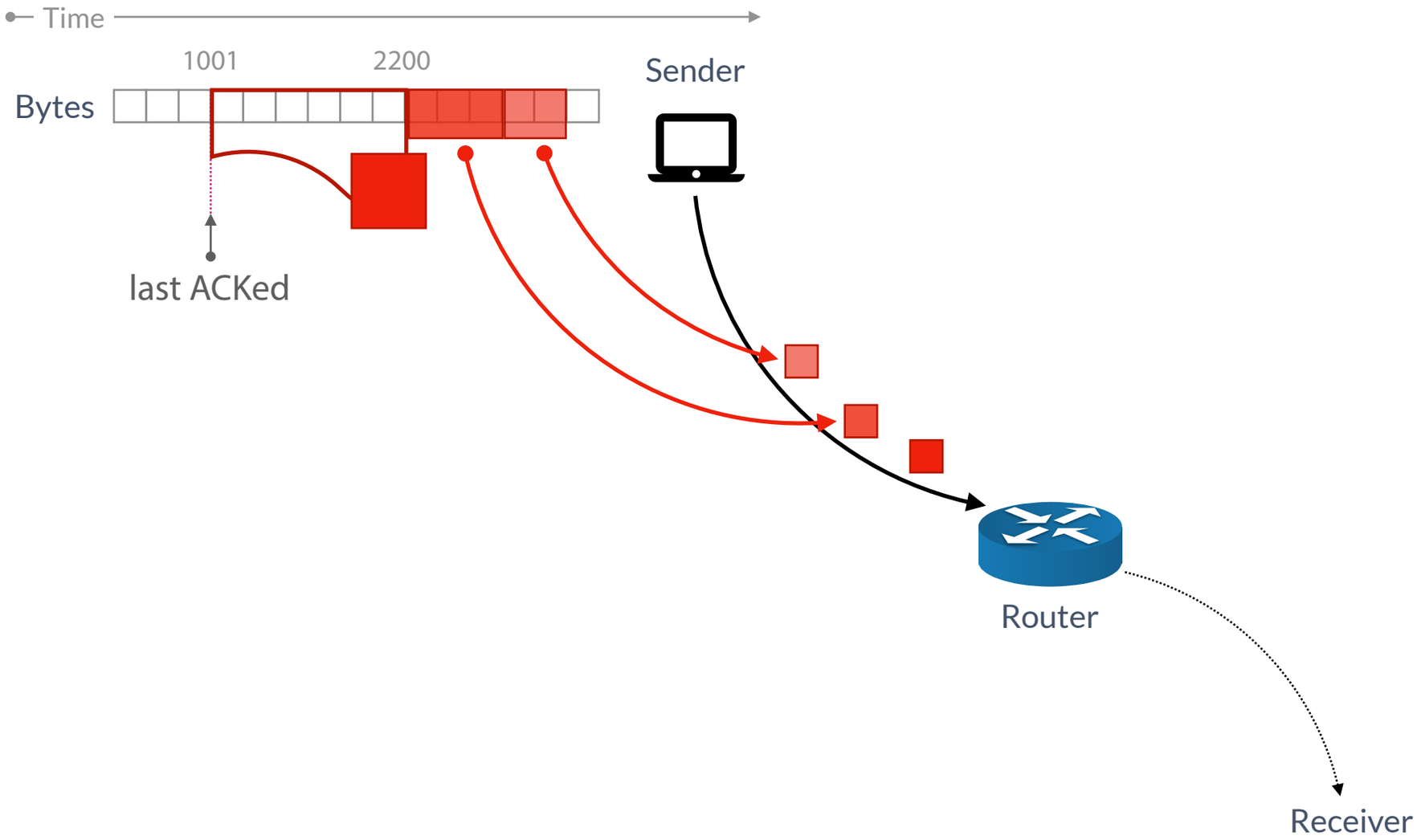
Stream of Bytes Service



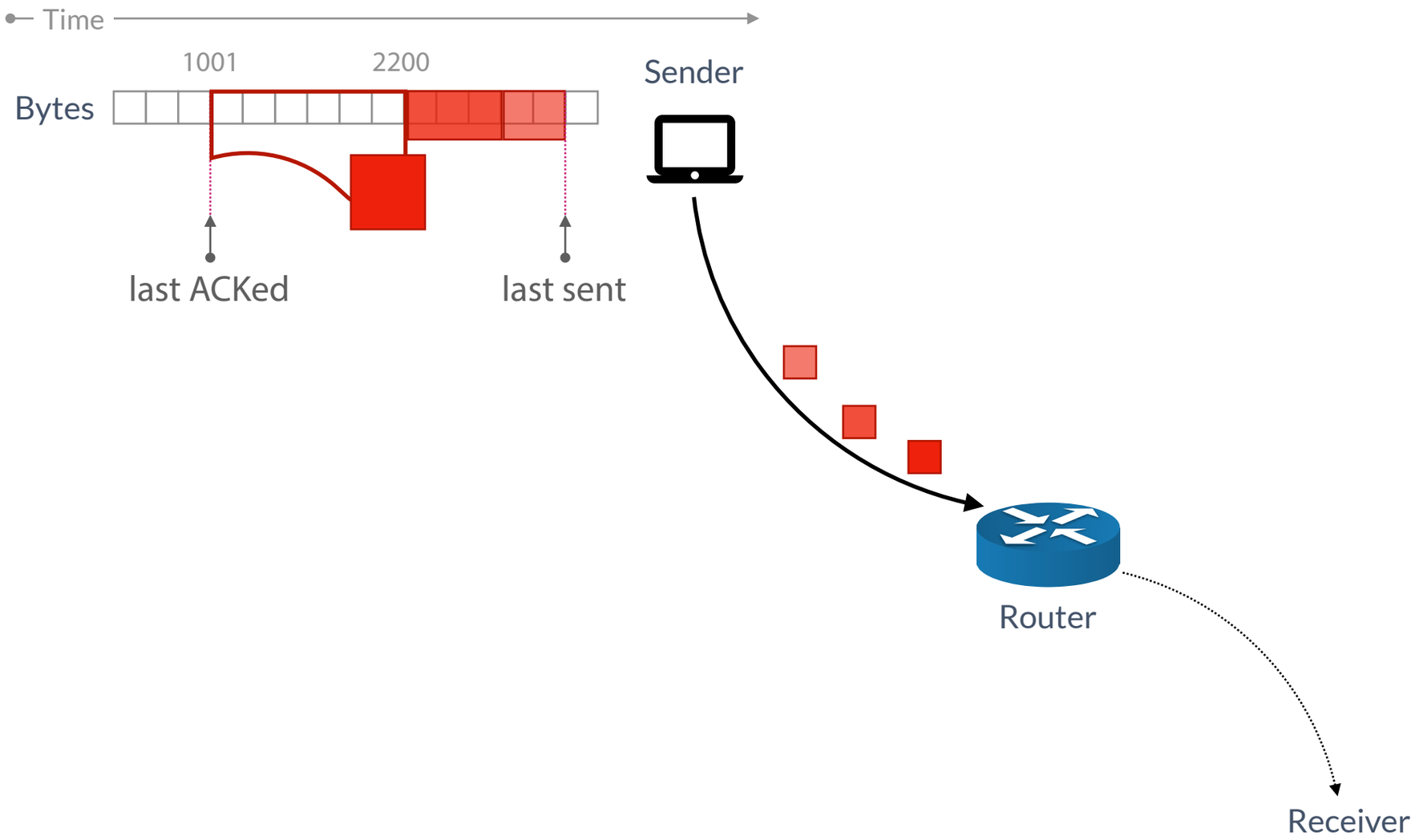
Stream of Bytes Service



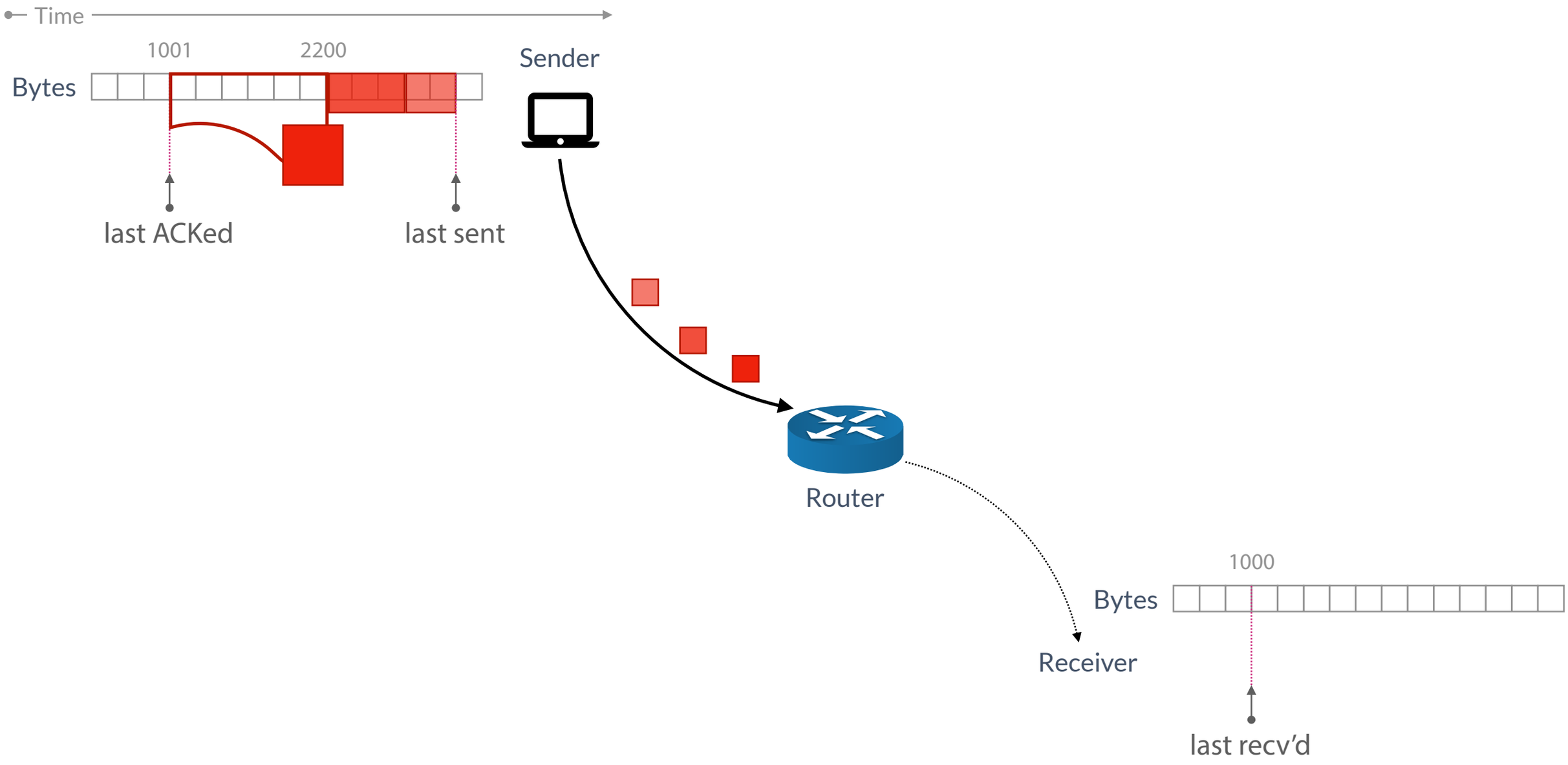
Stream of Bytes Service



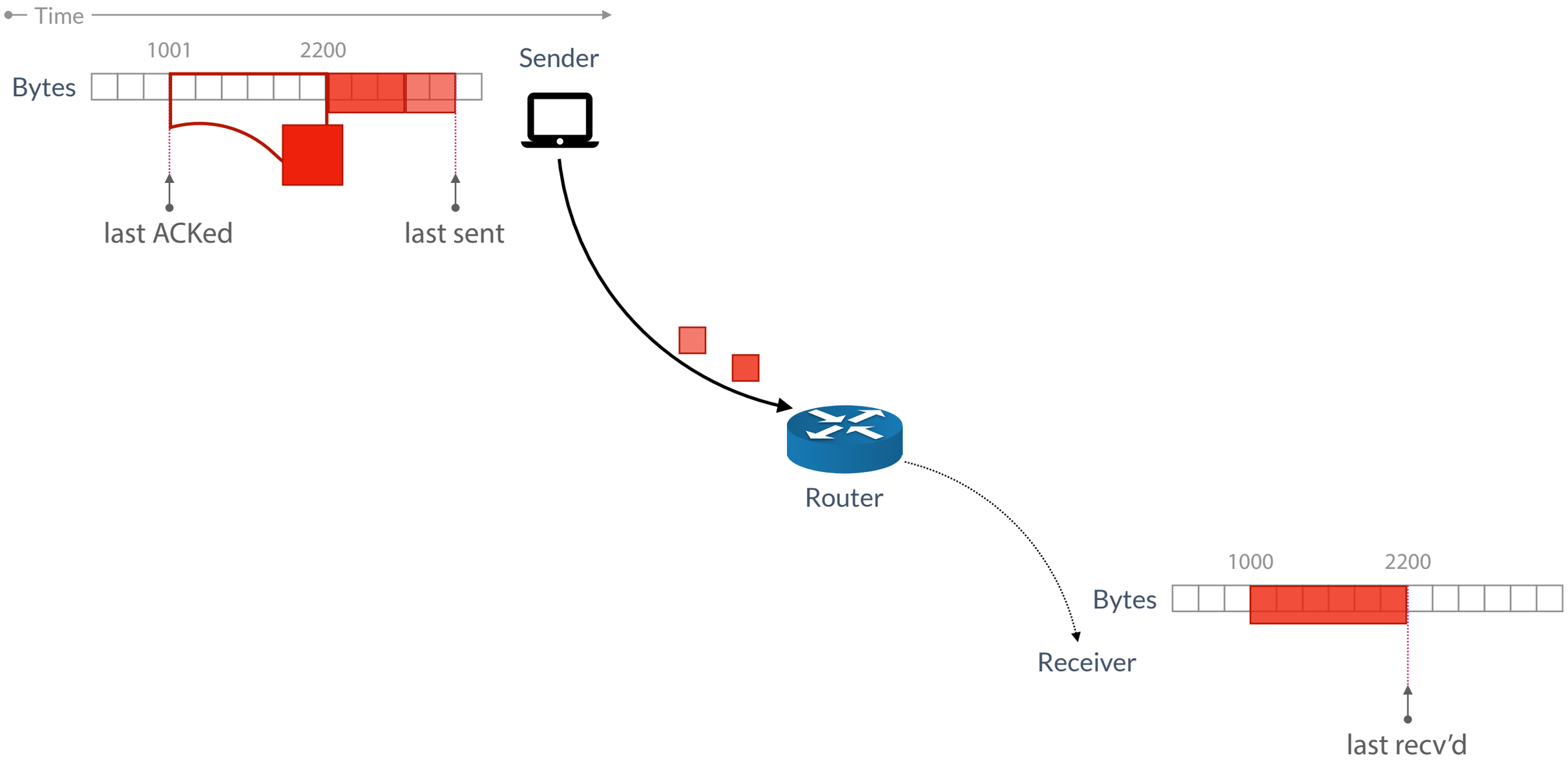
Stream of Bytes Service



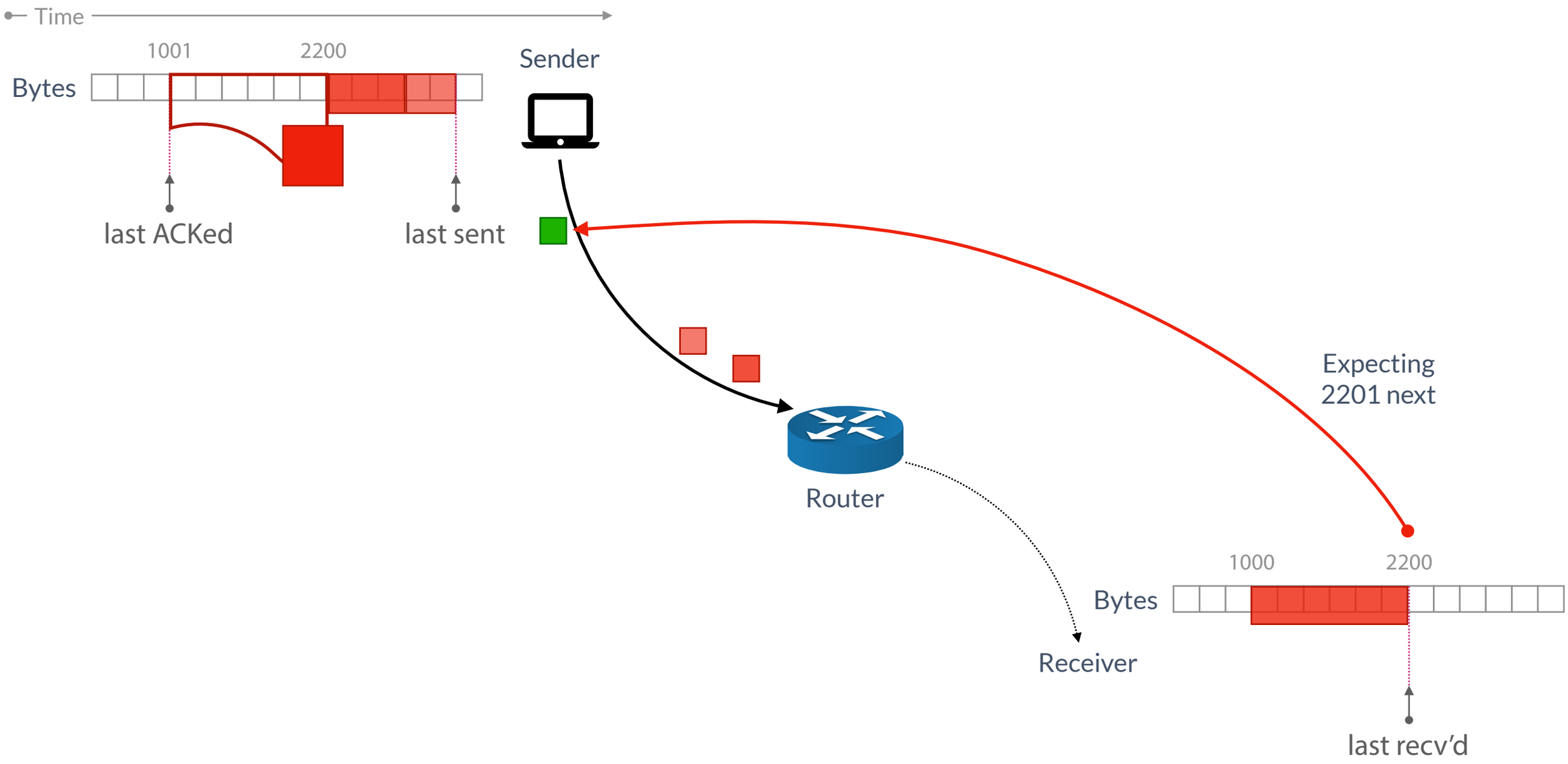
Stream of Bytes Service



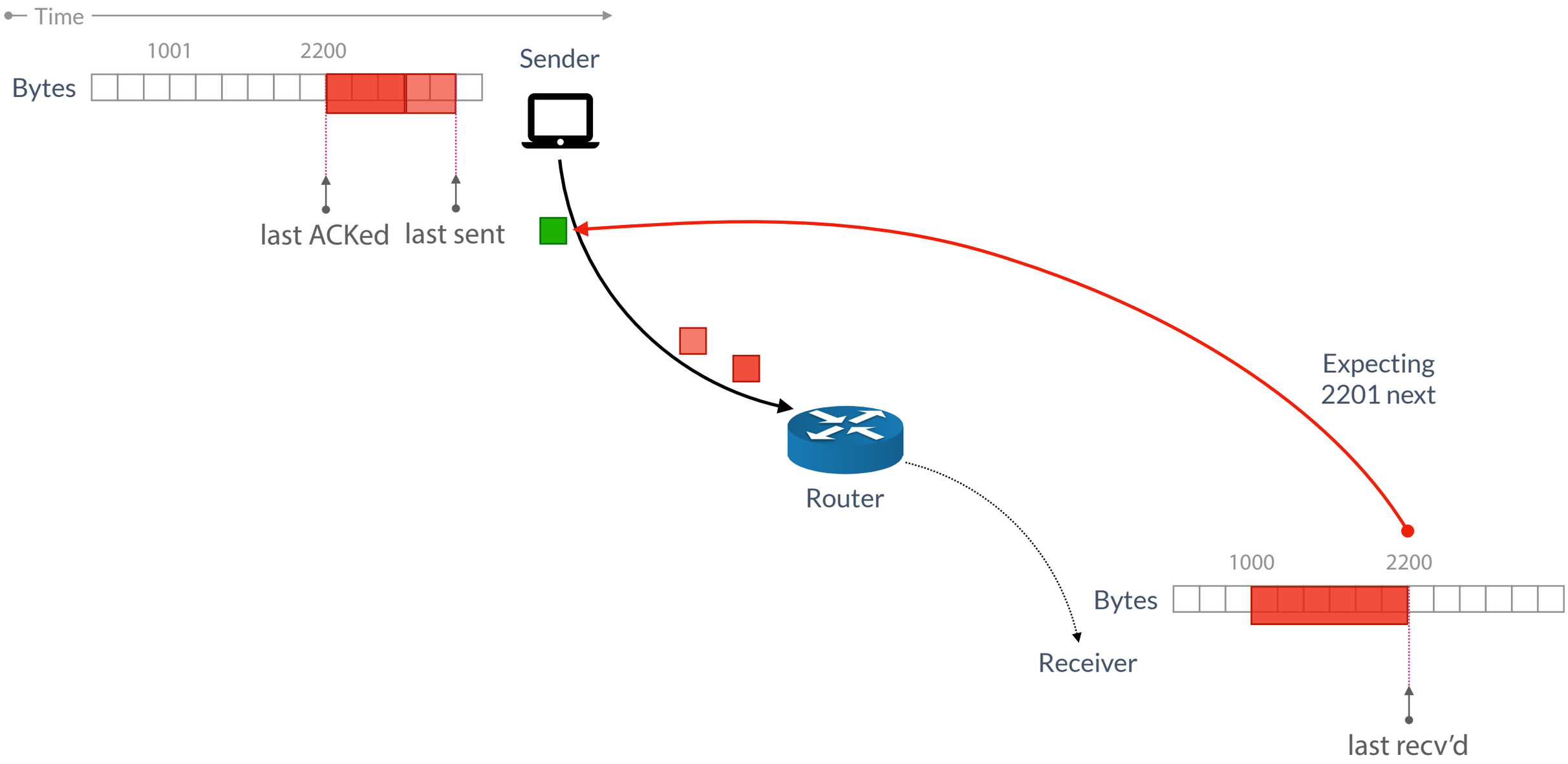
Stream of Bytes Service



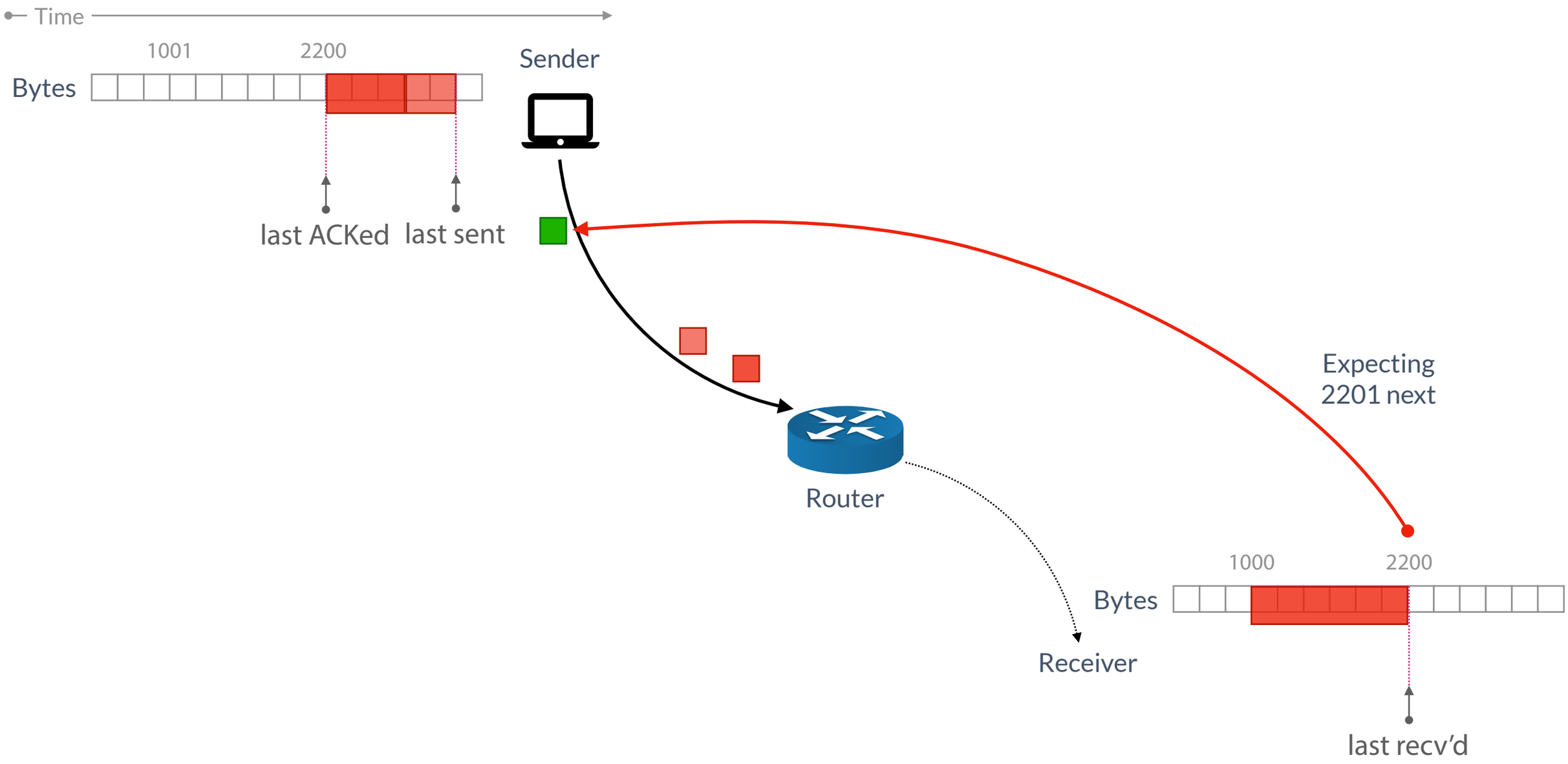
Stream of Bytes Service



Stream of Bytes Service



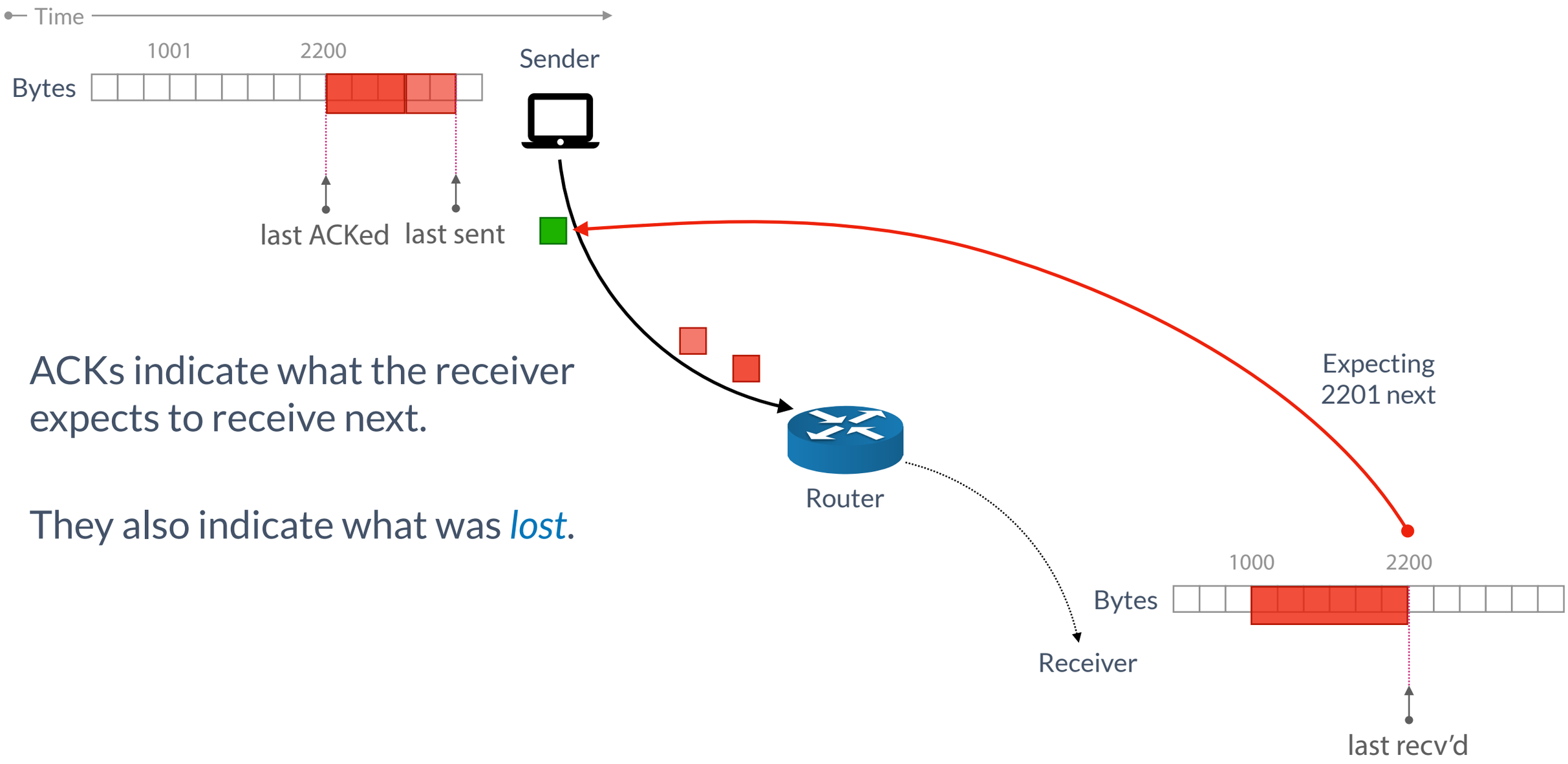
Stream of Bytes Service



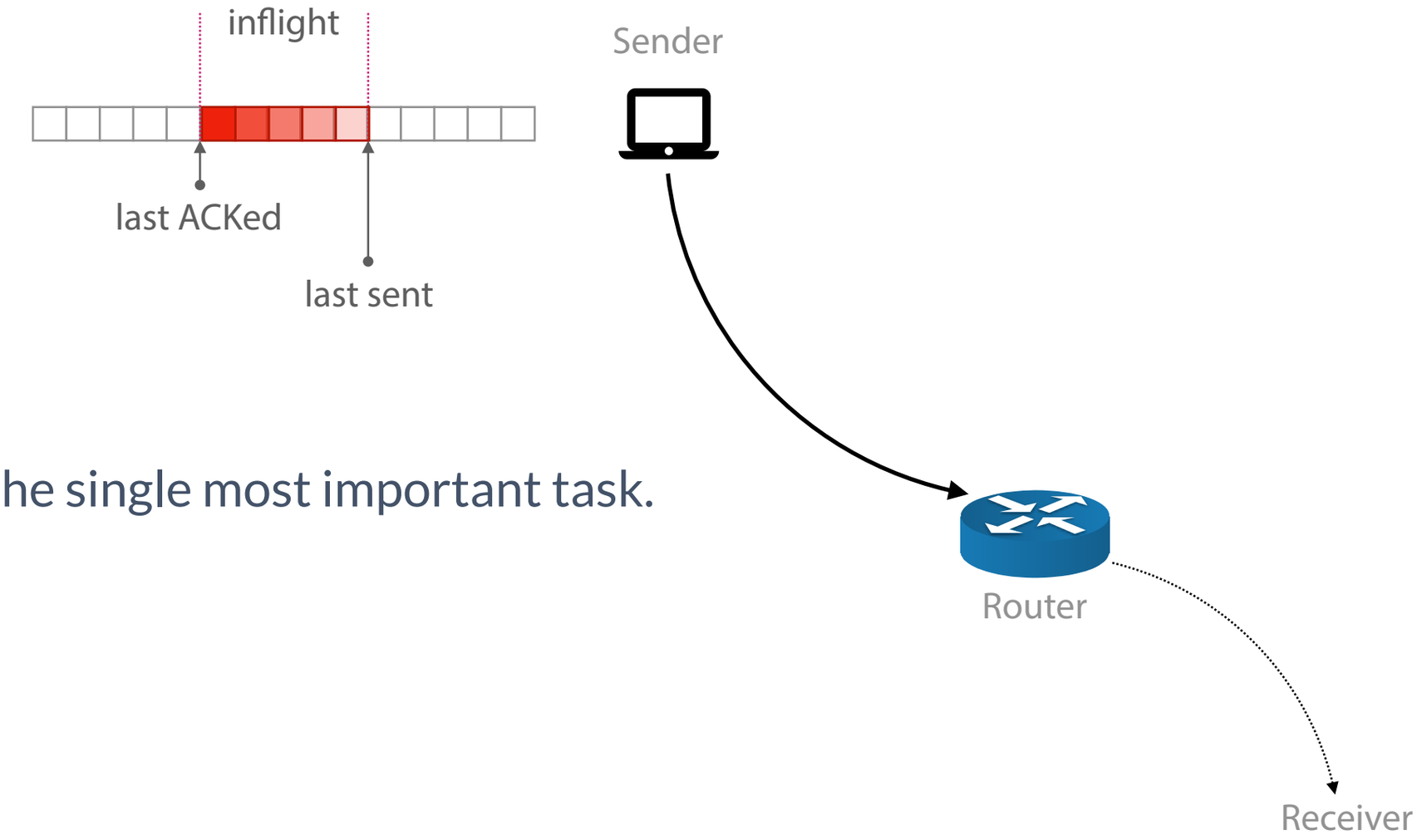
Stream of Bytes Service

ACKs indicate what the receiver expects to receive next.

They also indicate what was *lost*.

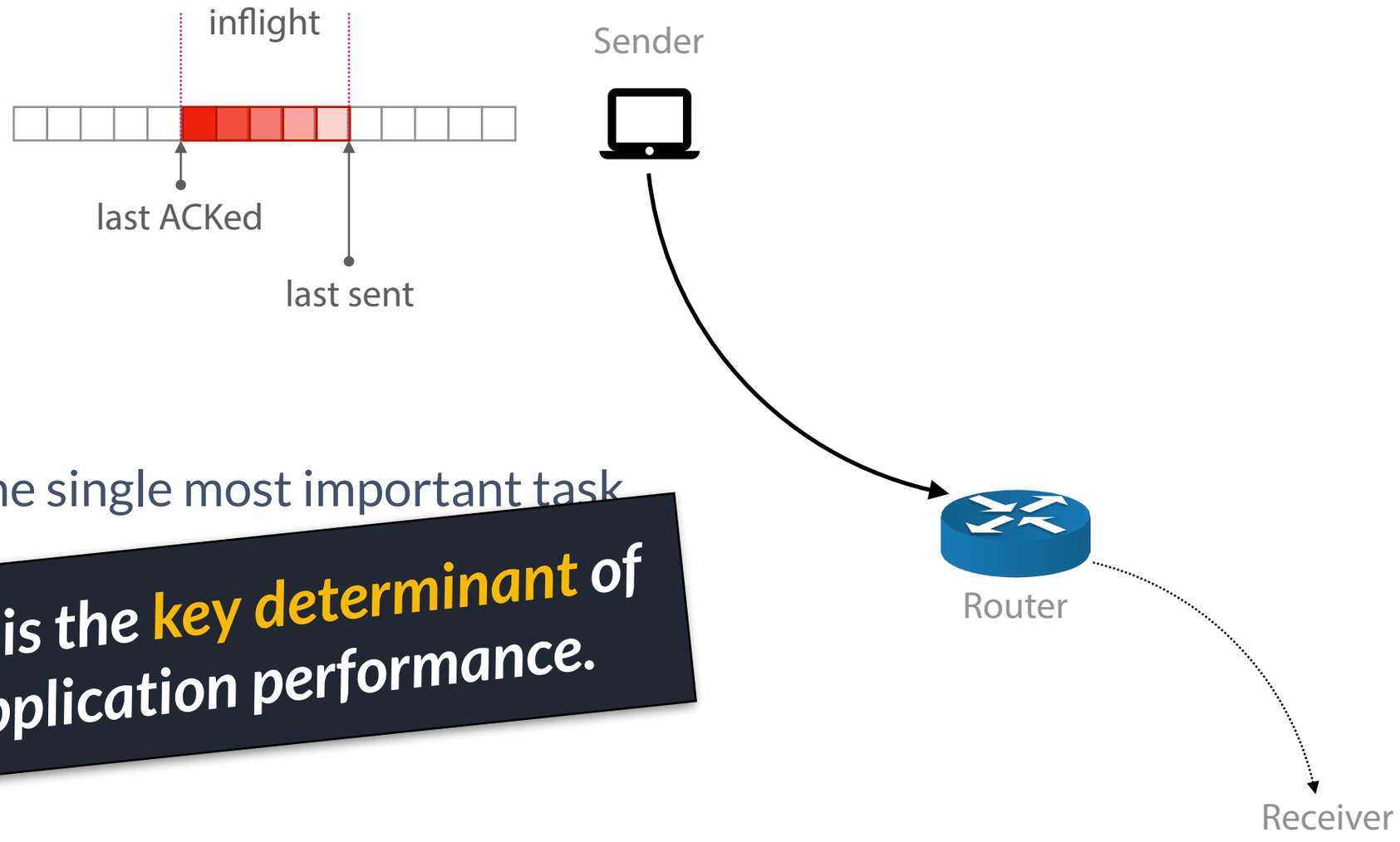


Round-trip time (RTT)



Estimating RTT is the single most important task.

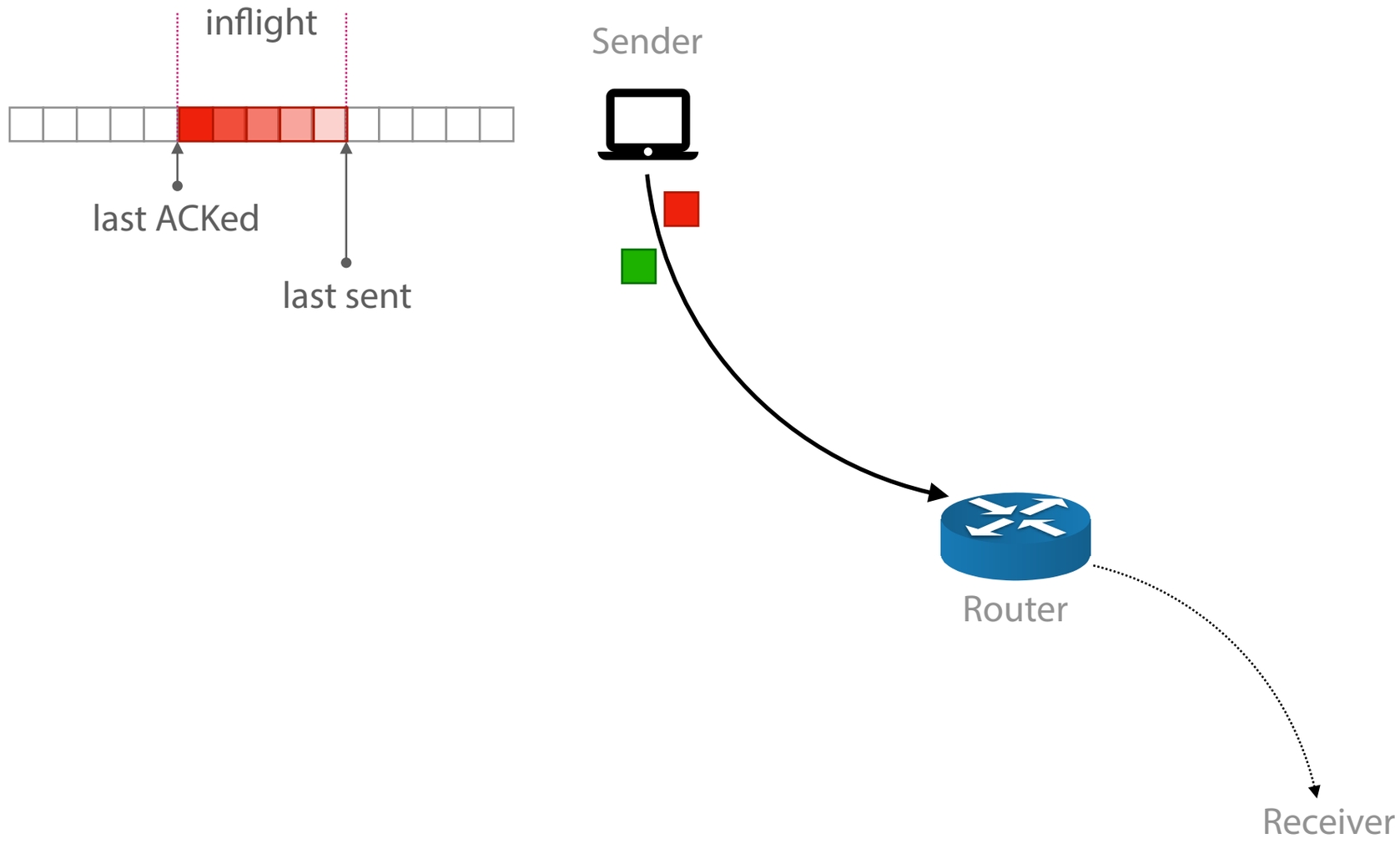
Round-trip time (RTT)



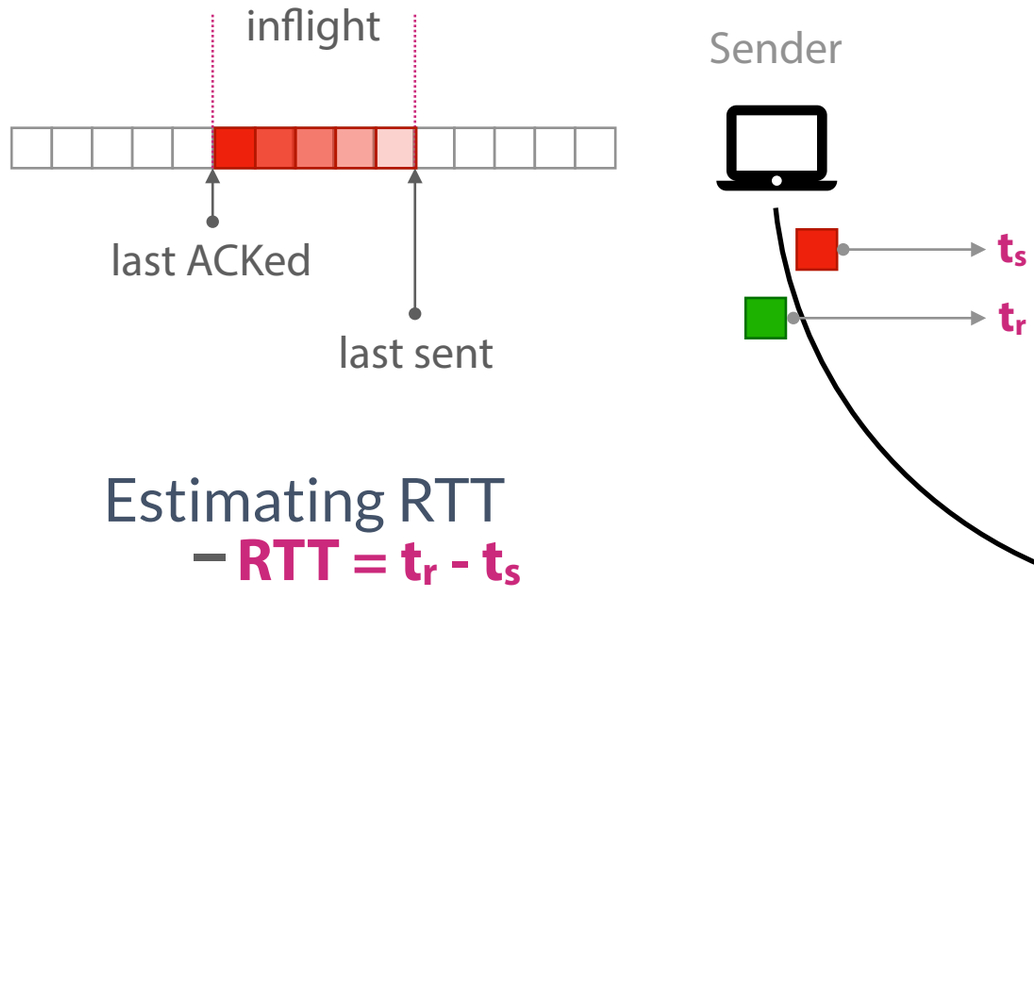
Estimating RTT is the single most important task

RTT is the *key determinant* of *application performance.*

Round-trip time (RTT)



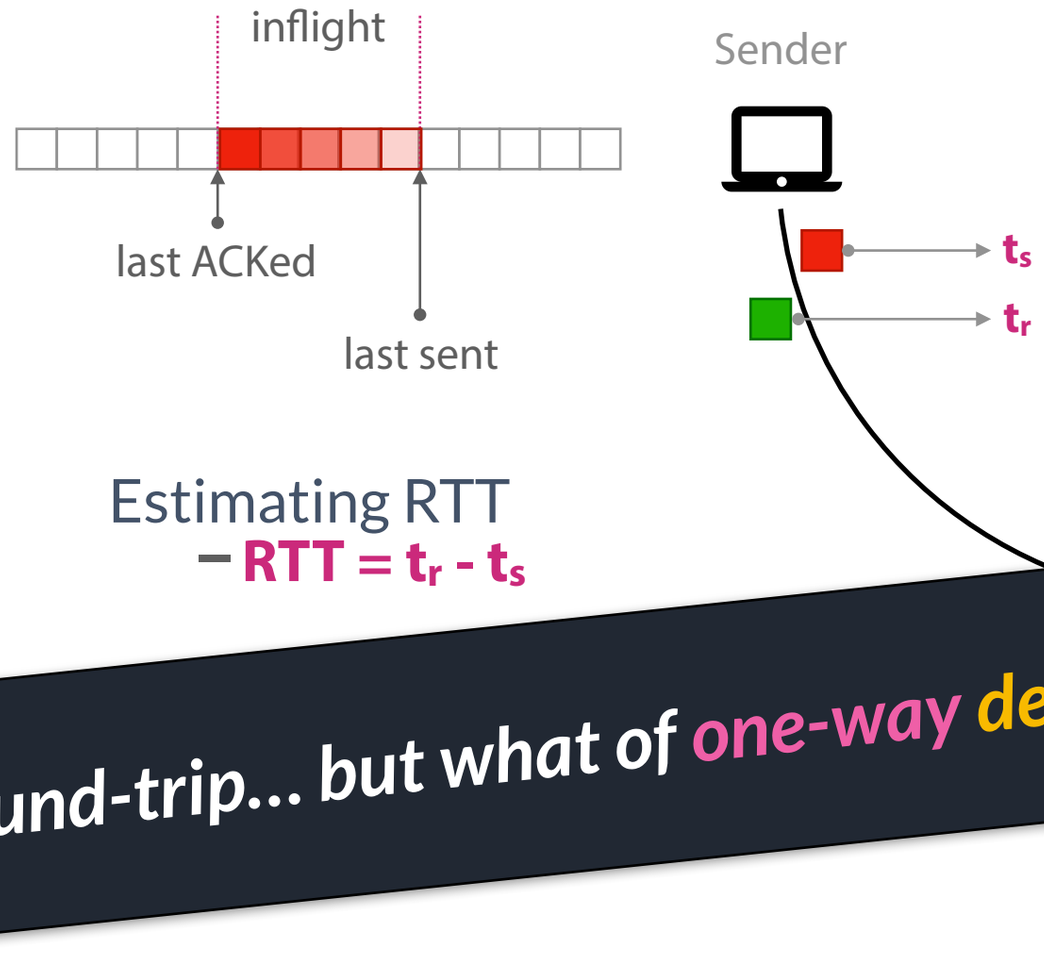
Round-trip time (RTT)



Estimating RTT

- **$RTT = t_r - t_s$**

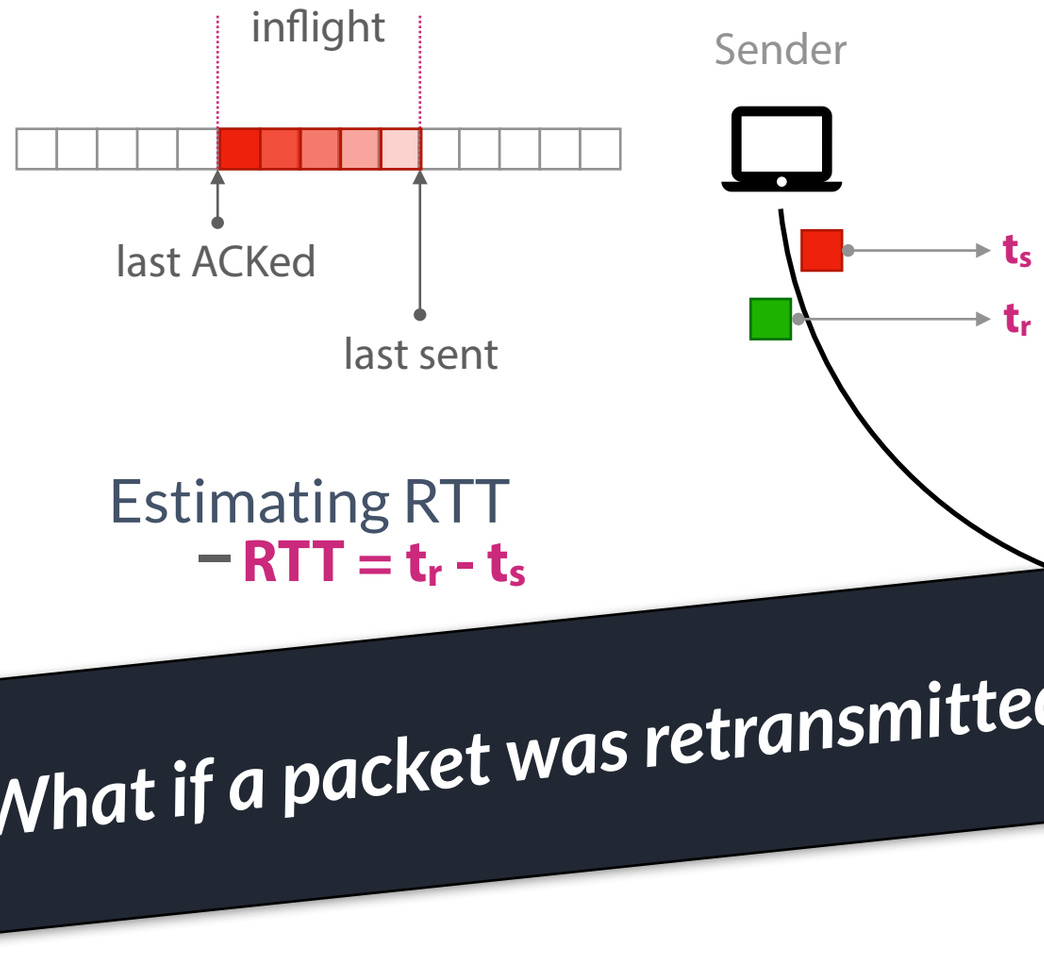
Round-trip time (RTT)



Estimating RTT
- **RTT = t_r - t_s**

Round-trip… but what of one-way delay?

Round-trip time (RTT)

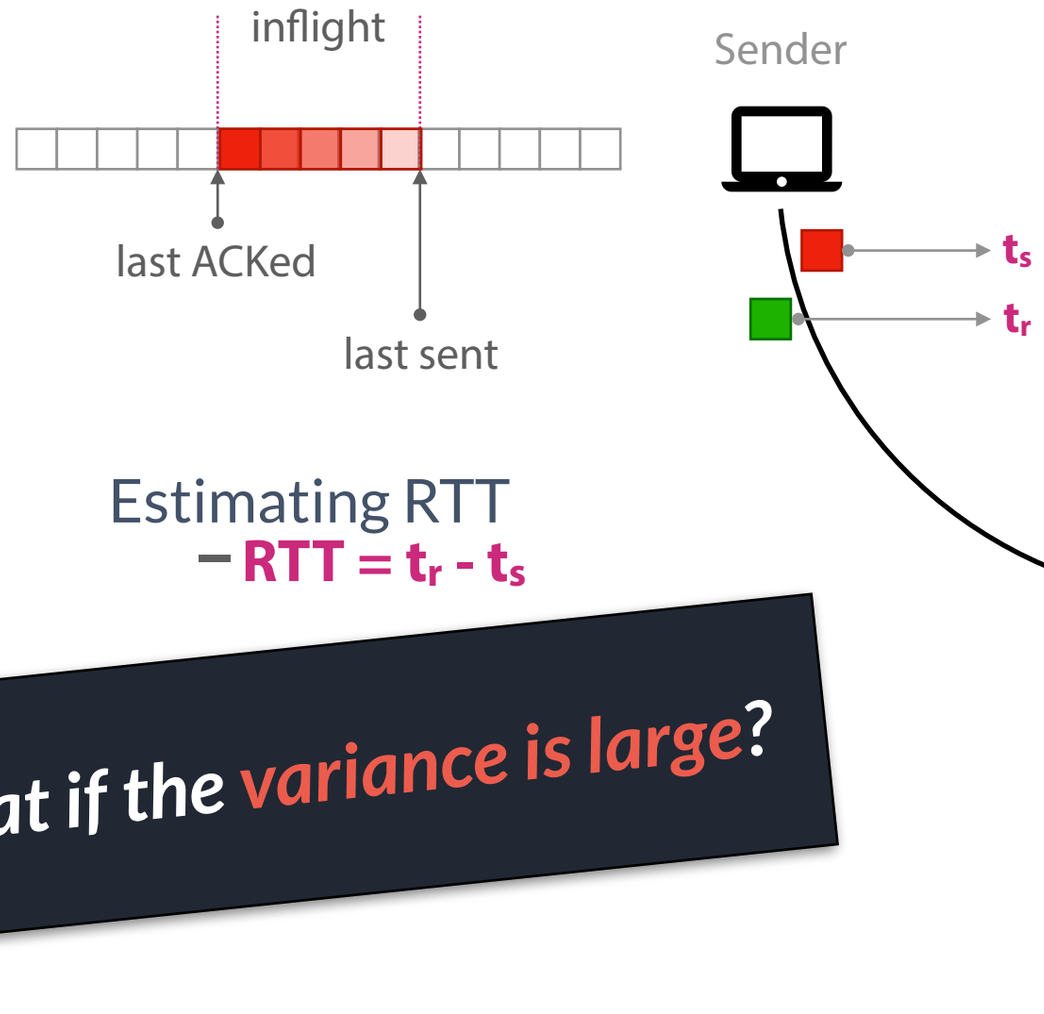


Estimating RTT

- **$RTT = t_r - t_s$**

What if a packet was retransmitted?

Round-trip time (RTT)

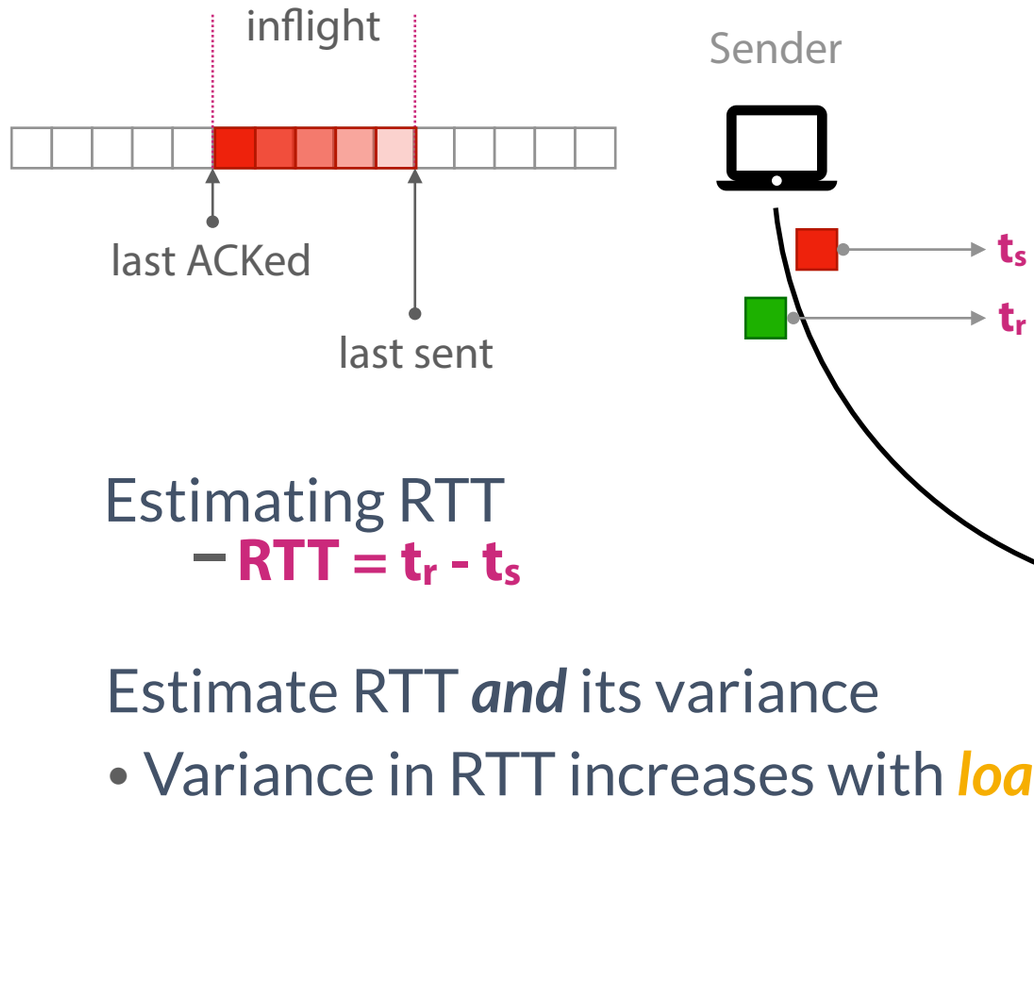


Estimating RTT

- **RTT = t_r - t_s**

What if the *variance is large*?

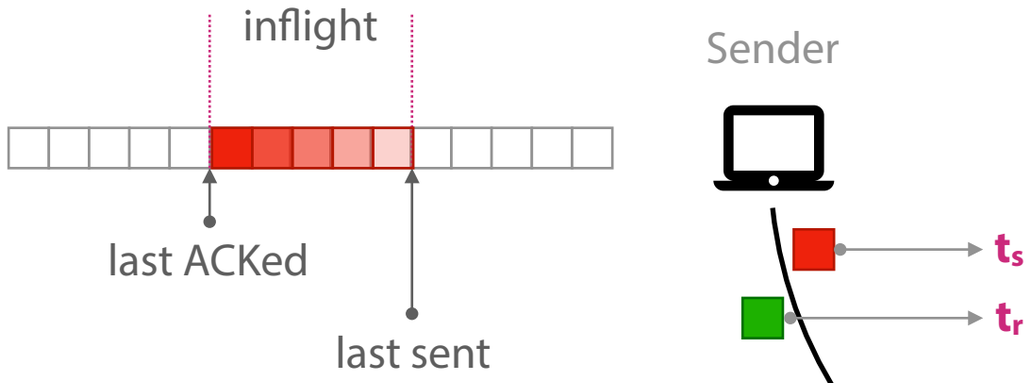
Round-trip time (RTT)

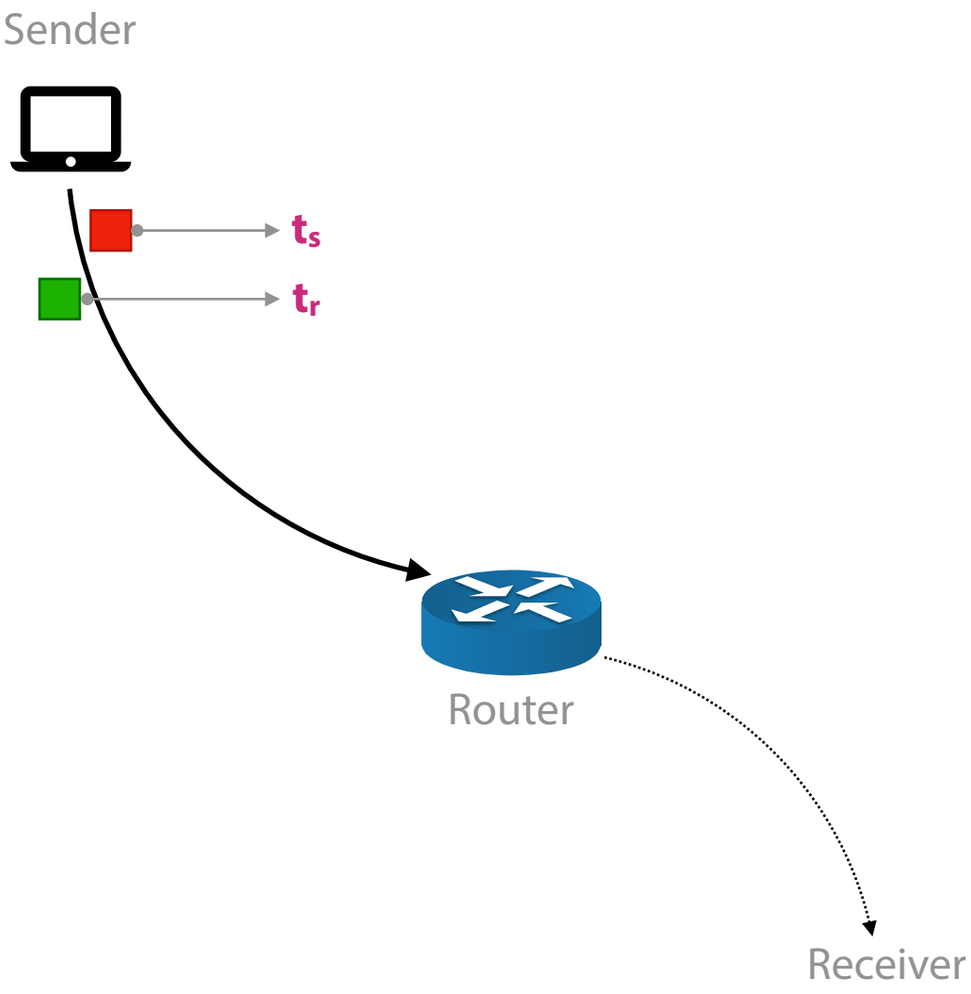


Estimating RTT
- **RTT = t_r - t_s**

Estimate RTT ***and*** its variance
- Variance in RTT increases with ***load***

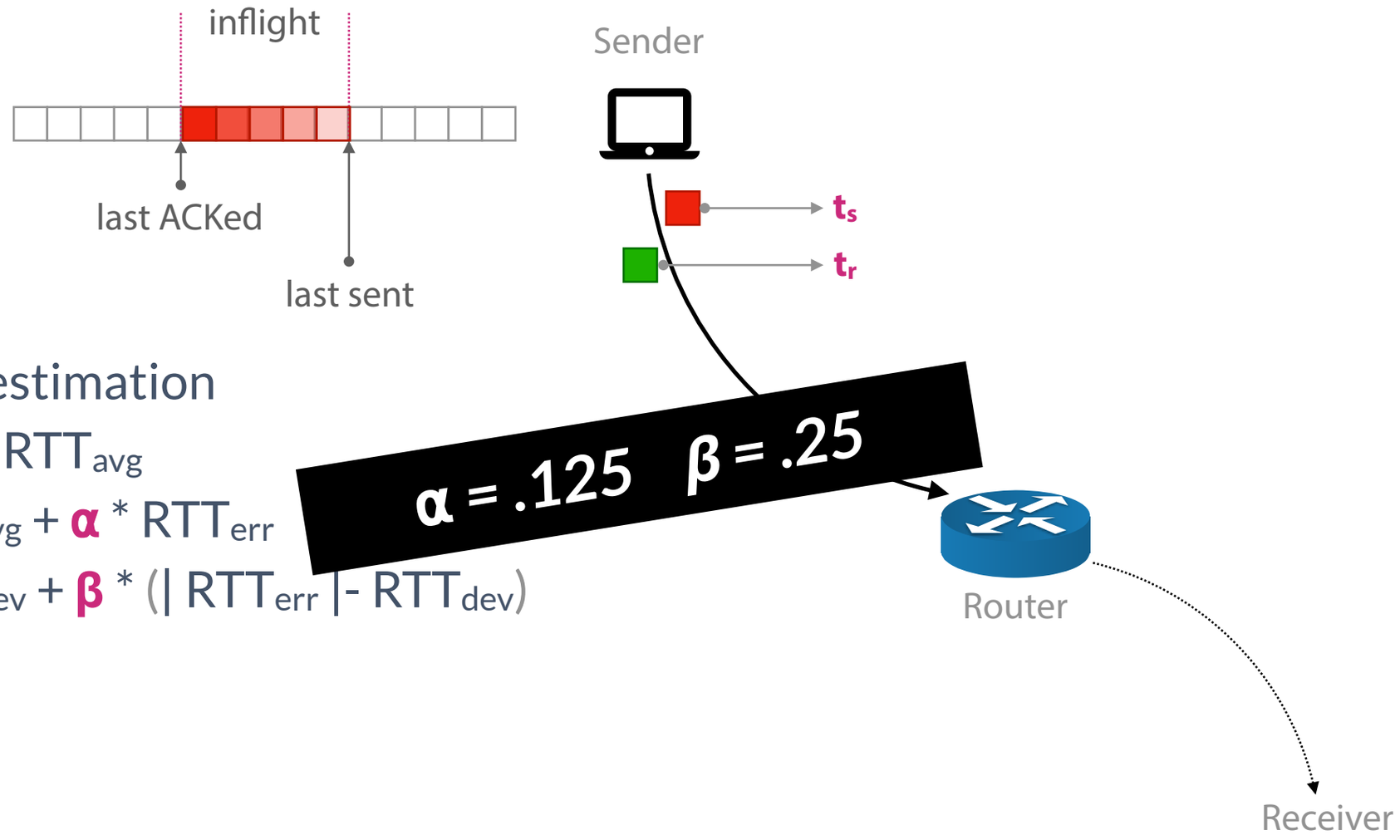
Round-trip time (RTT)





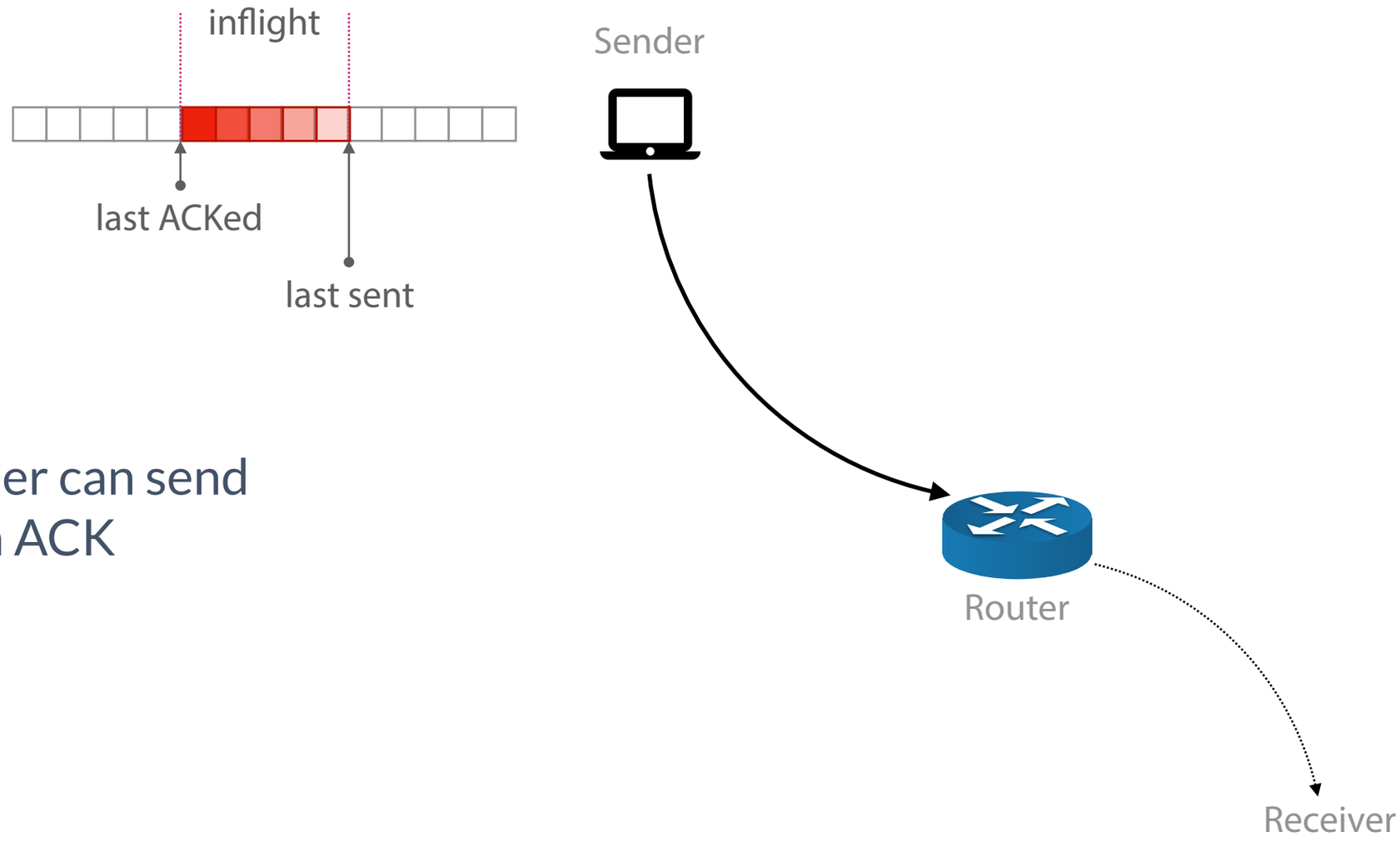
- **Round-trip** time estimation
  - $RTT_{err} = RTT - RTT_{avg}$
  - $RTT_{avg} = RTT_{avg} + \alpha * RTT_{err}$
  - $RTT_{dev} = RTT_{dev} + \beta * (|RTT_{err}| - RTT_{dev})$

Round-trip time (RTT)



- **Round-trip** time estimation
  - $RTT_{err} = RTT - RTT_{avg}$
  - $RTT_{avg} = RTT_{avg} + \alpha * RTT_{err}$
  - $RTT_{dev} = RTT_{dev} + \beta * (|RTT_{err}| - RTT_{dev})$

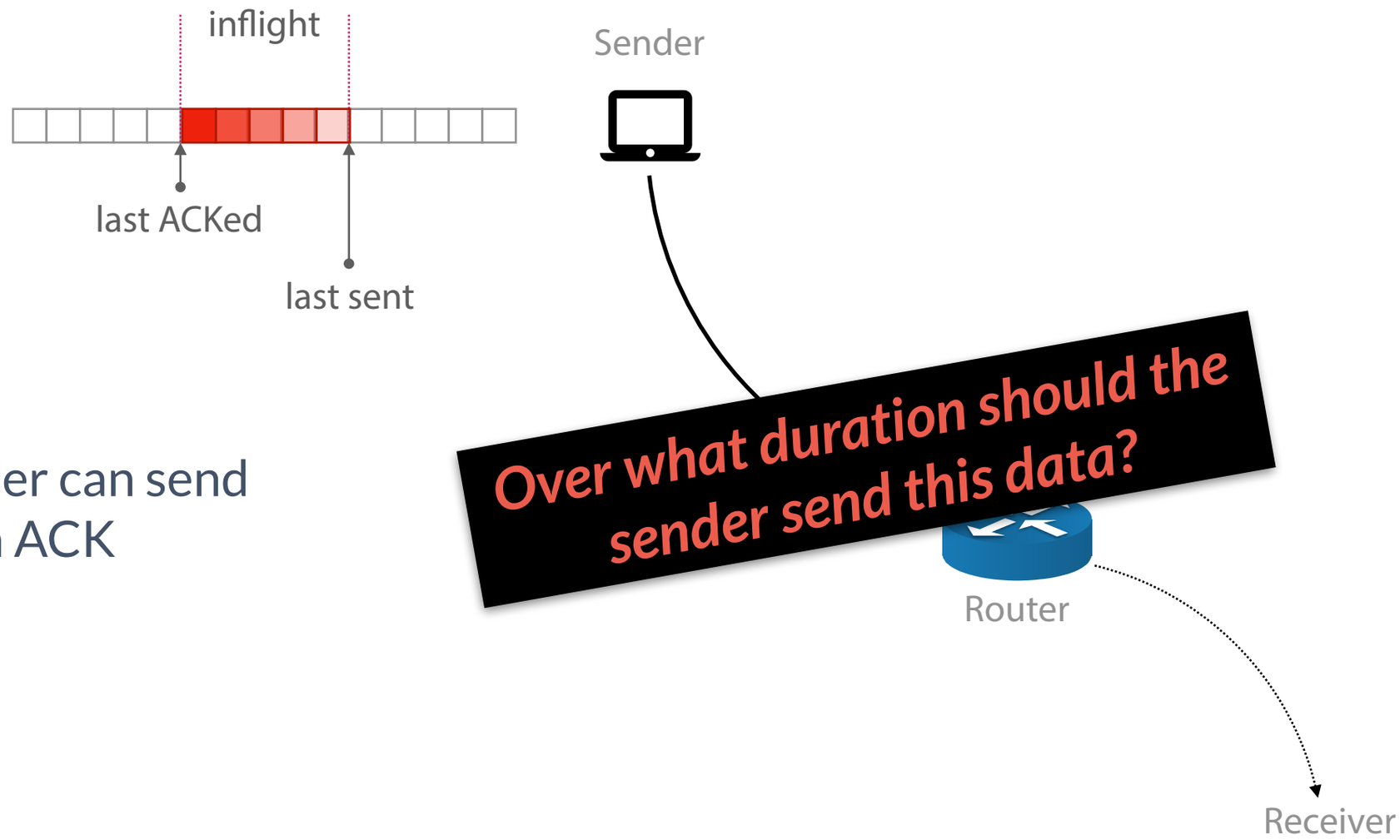
Flow Control



inflight

amount of data a sender can send without waiting for an ACK

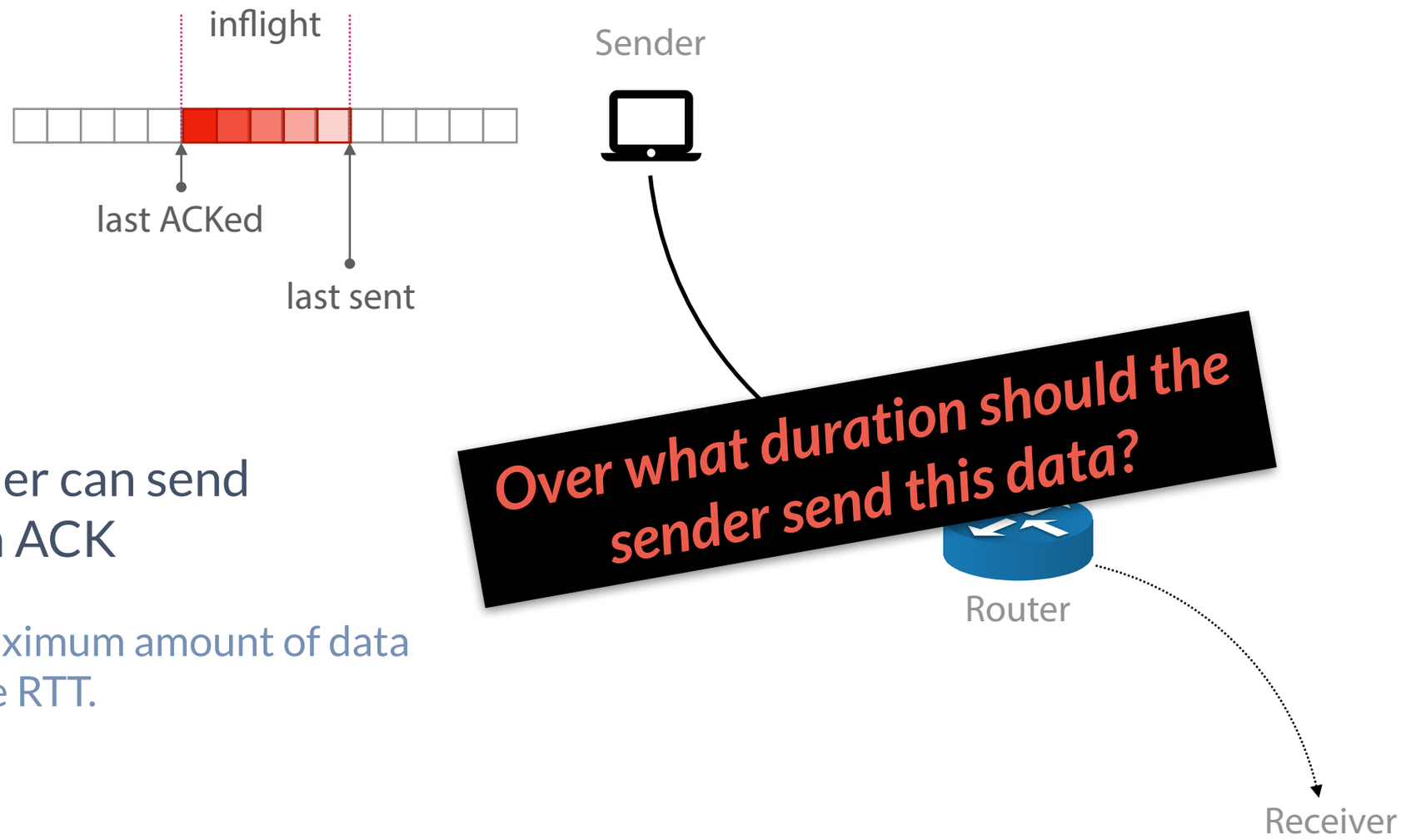
Flow Control



inflight

amount of data a sender can send without waiting for an ACK

Flow Control

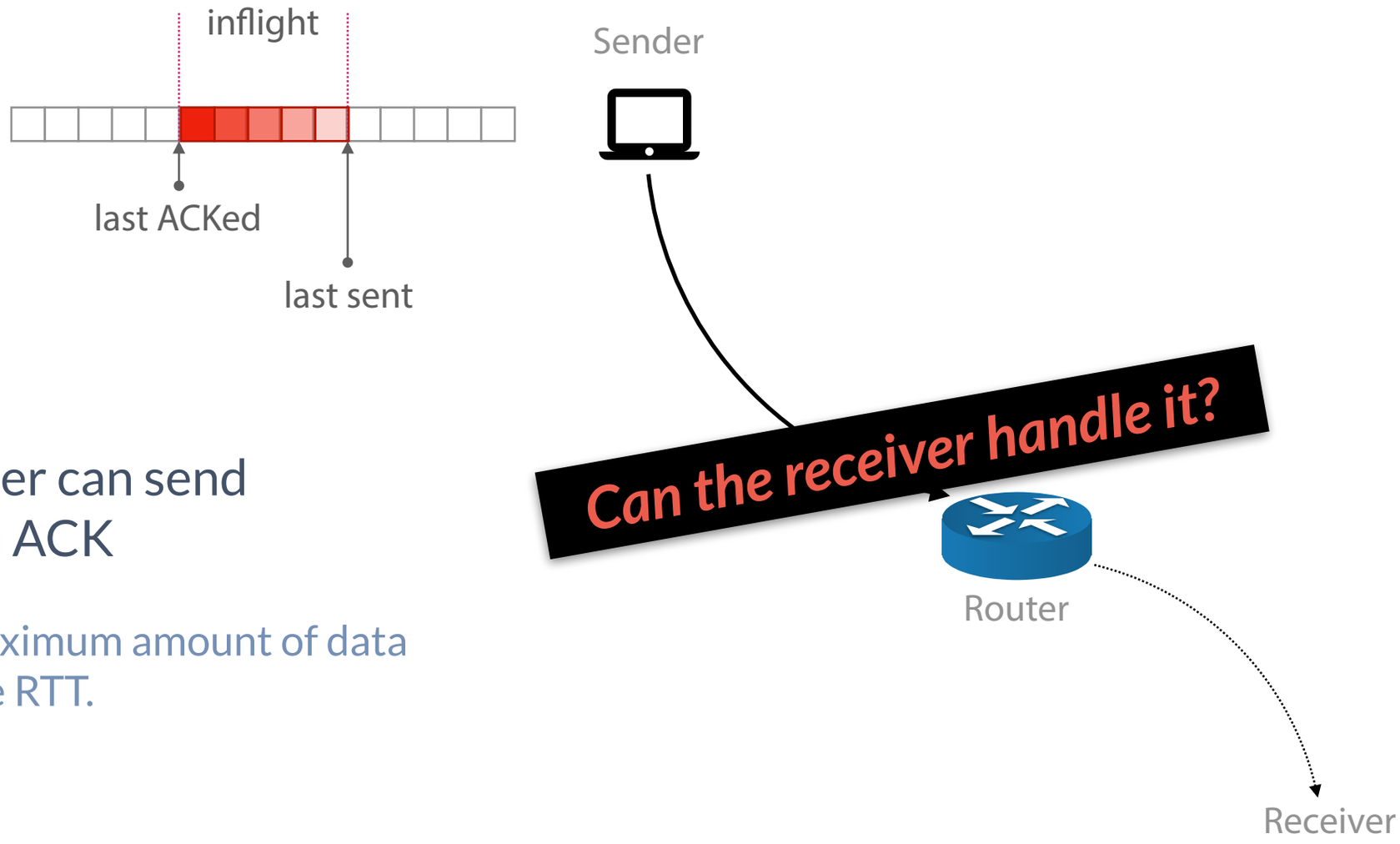


inflight

amount of data a sender can send without waiting for an ACK

Assume that this is the maximum amount of data the sender can send in one RTT.

Flow Control



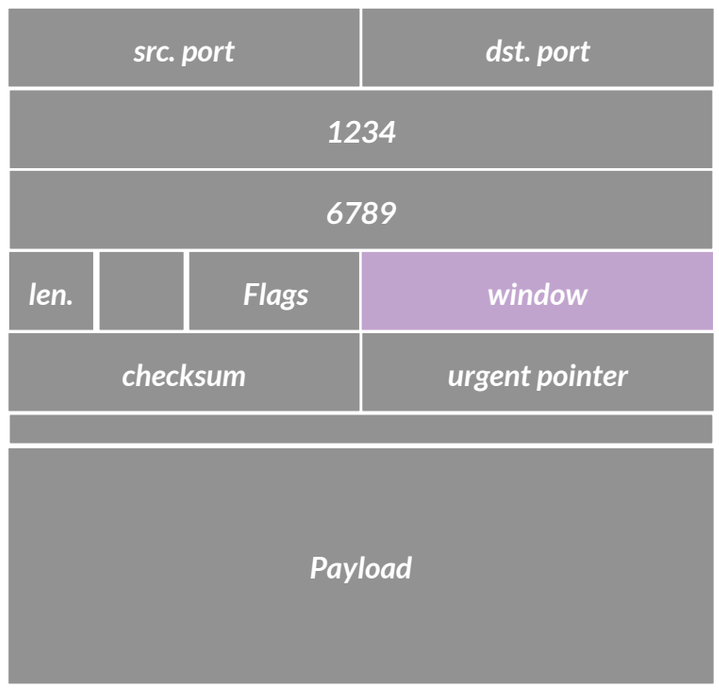
inflight

amount of data a sender can send without waiting for an ACK

Assume that this is the maximum amount of data the sender can send in one RTT.

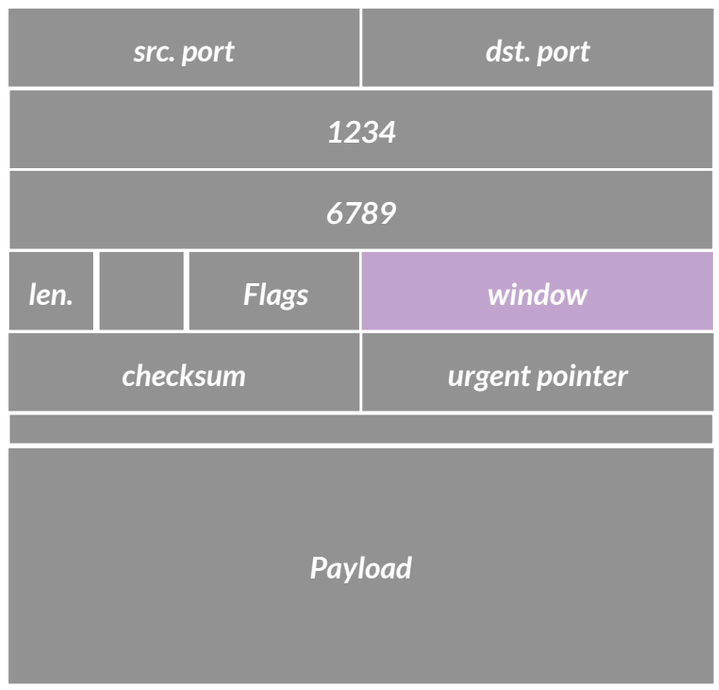
Flow Control

- *Window*
  - Receiver announces its receive window size in this field
  - 16 bits



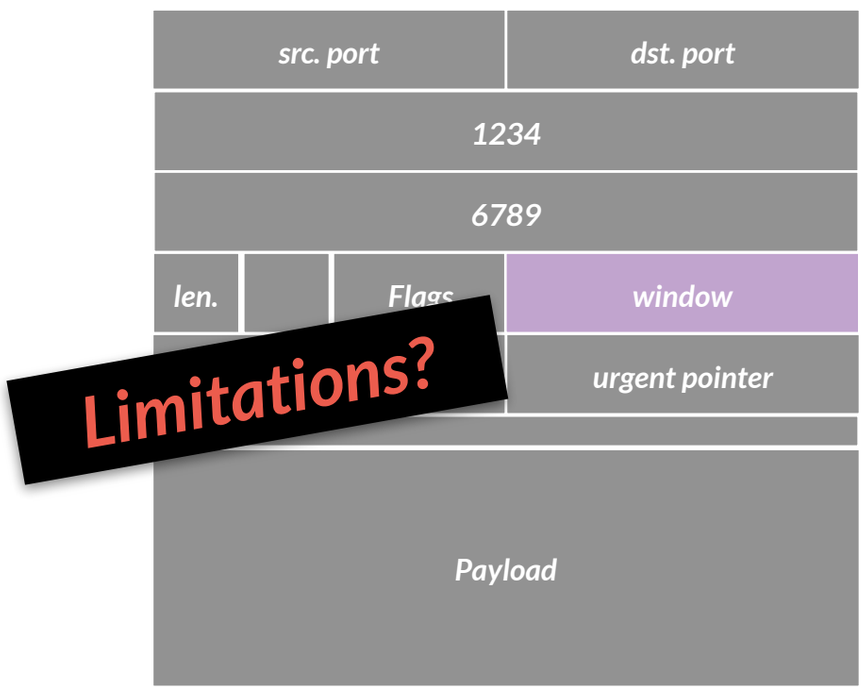
Flow Control

- *Window*
  - Receiver announces its receive window size in this field
  - 16 bits
  - *Every* segment (from *handshake*) the receiver *advertises* its (current) window



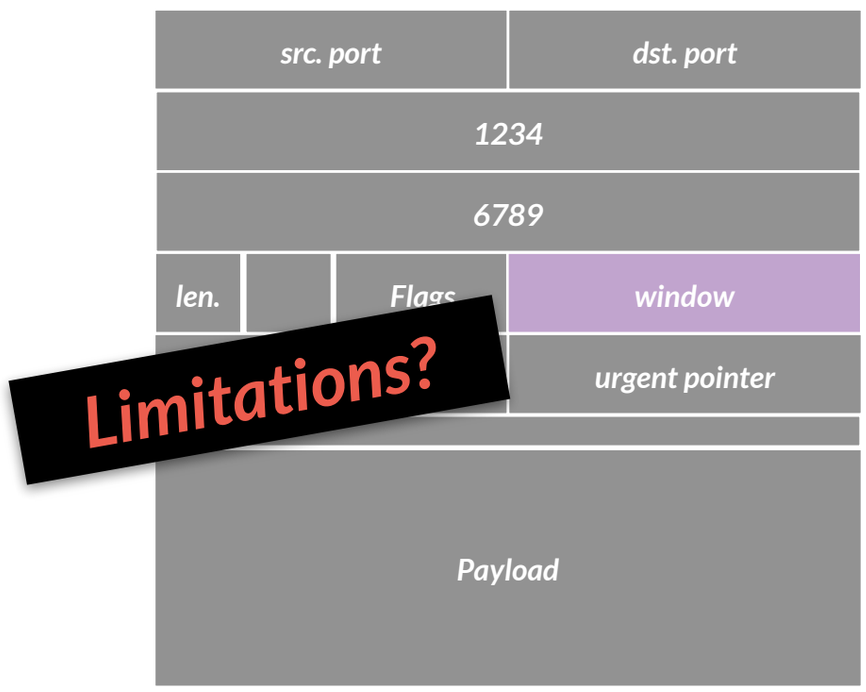
Flow Control

- *Window*
  - Receiver announces its receive window size in this field
  - 16 bits
  - *Every* segment (from *handshake*) the receiver *advertises* its (current) window



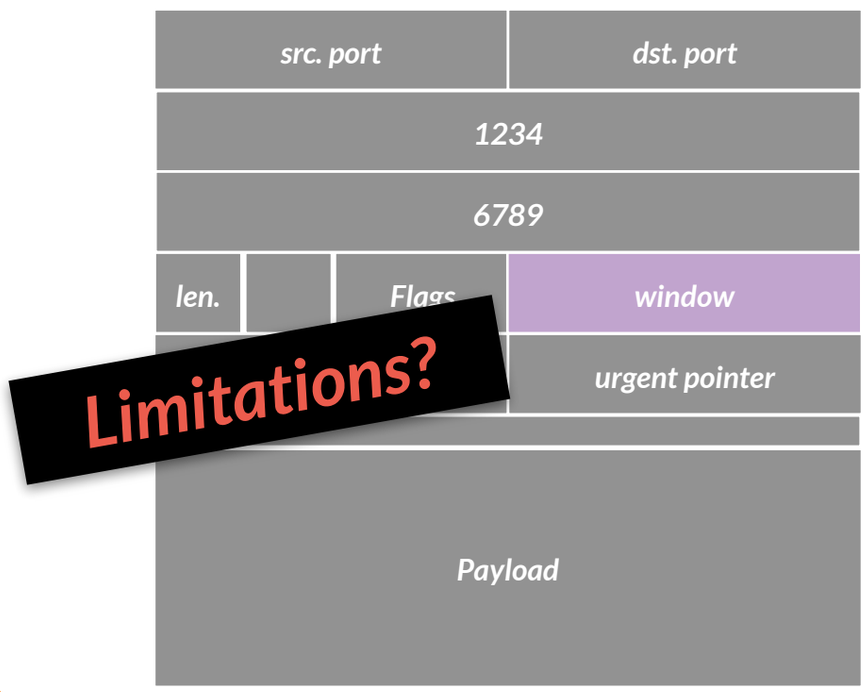
Flow Control

- *Window*
  - Receiver announces its receive window size in this field
  - 16 bits
  - *Every* segment (from *handshake*) the receiver *advertises* its (current) window
  - Maximum value is 65,535 bytes



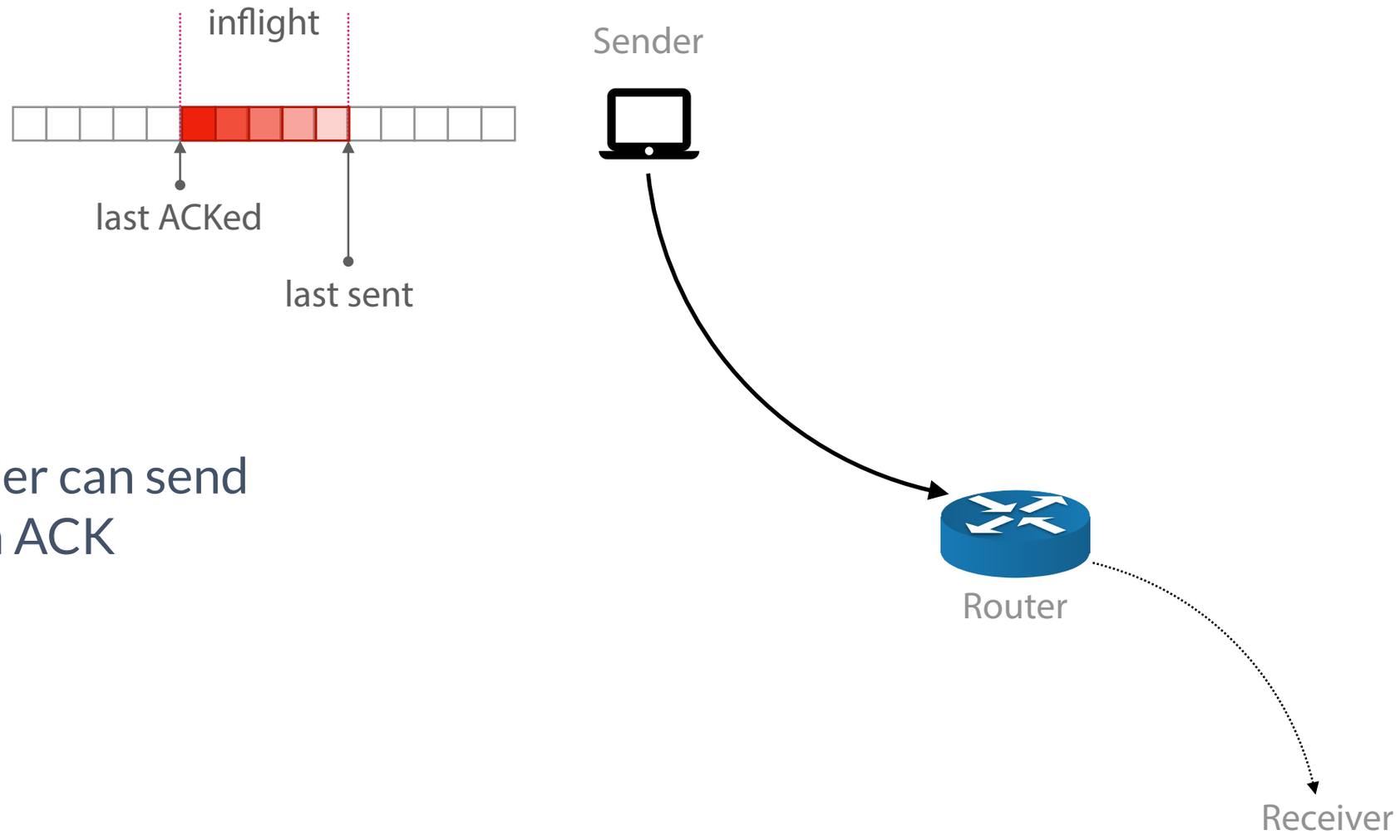
Flow Control

- *Window*
  - Receiver announces its receive window size in this field
  - 16 bits
  - *Every* segment (from *handshake*) the receiver *advertises* its (current) window
  - Maximum value is 65,535 bytes



- *Window scale factor*
  - a TCP **option**; left-shift multiplier
  - Enlarge window size to ~1 GB

Congestion Control



inflight

amount of data a sender can send without waiting for an ACK


CARPOOLS ONLY
2 OR MORE PERSONS
PER VEHICLE
Slauson Ave
Sepulveda Bl
1 MILE

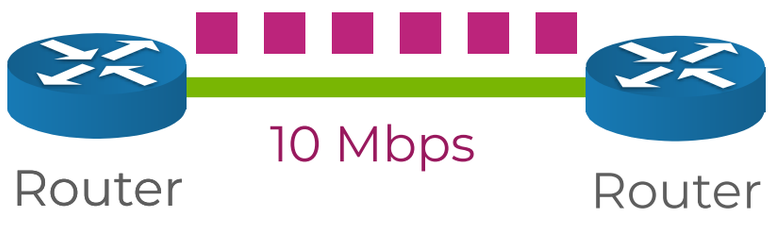

CARPOOLS ONLY
Slauson Ave
Sepulveda Bl
1 MILE

[www.fleetowner.com, secure.i.telegraph.co.uk, abbtakk.tv, www.itv.com]

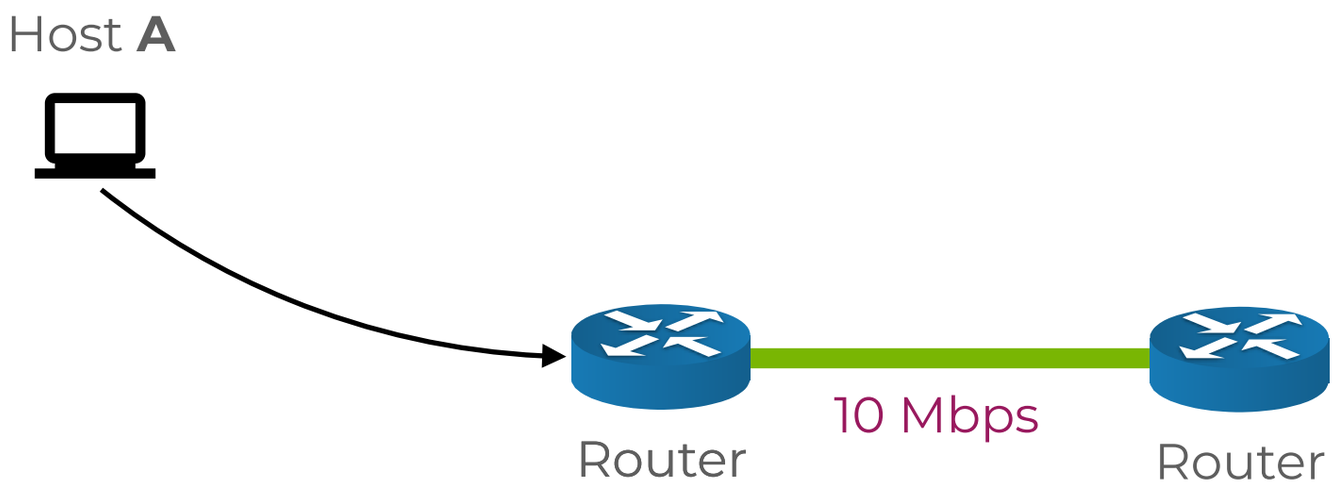
Router

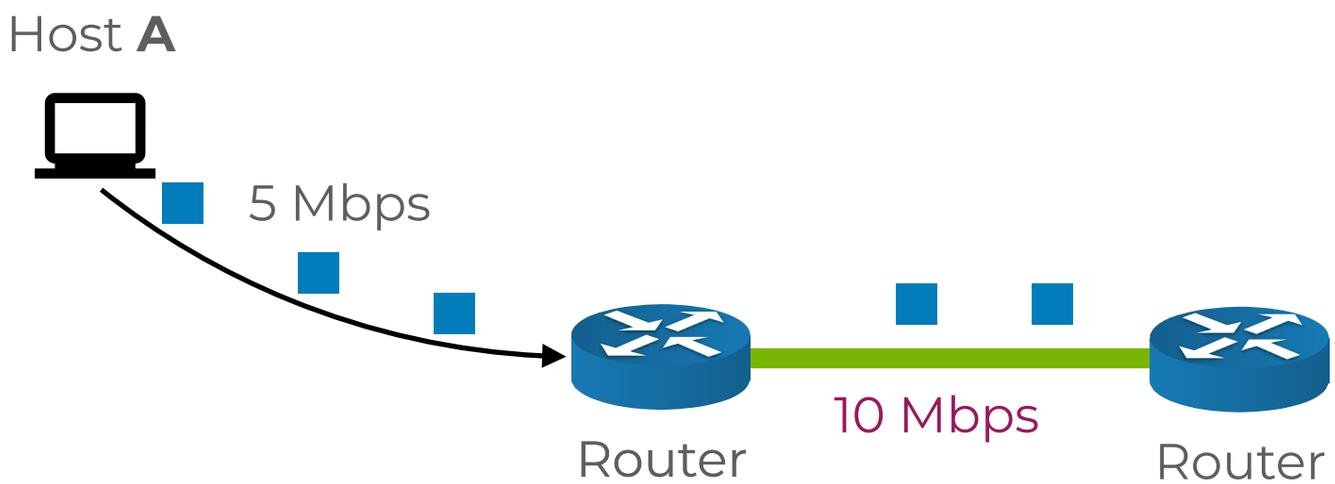
Router

System has finite **capacity**.



System has finite **capacity**.

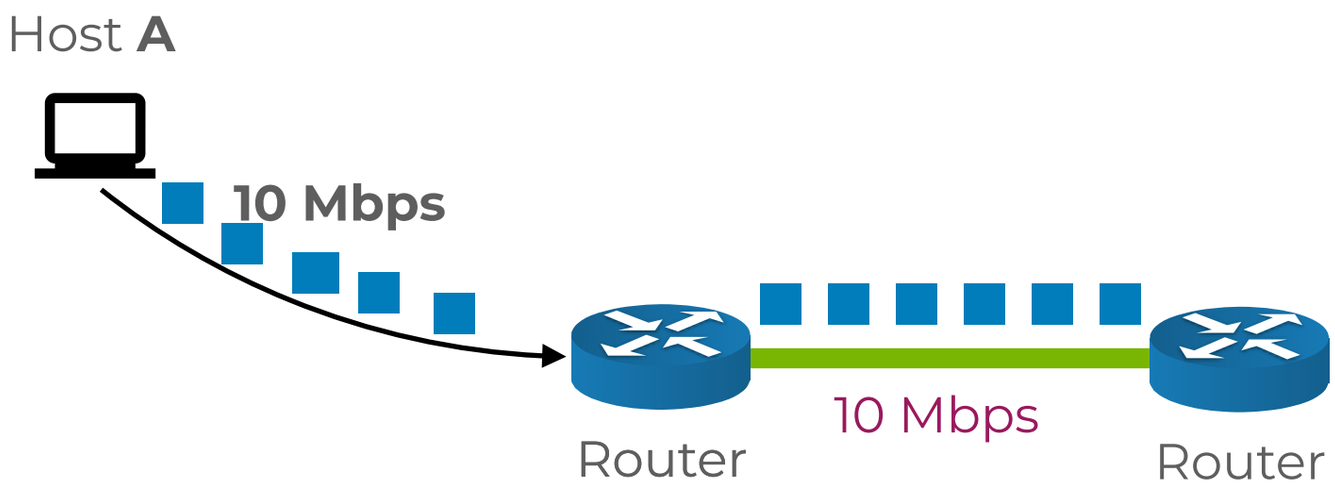




System has finite **capacity**.

Offered **load** is at or below capacity.

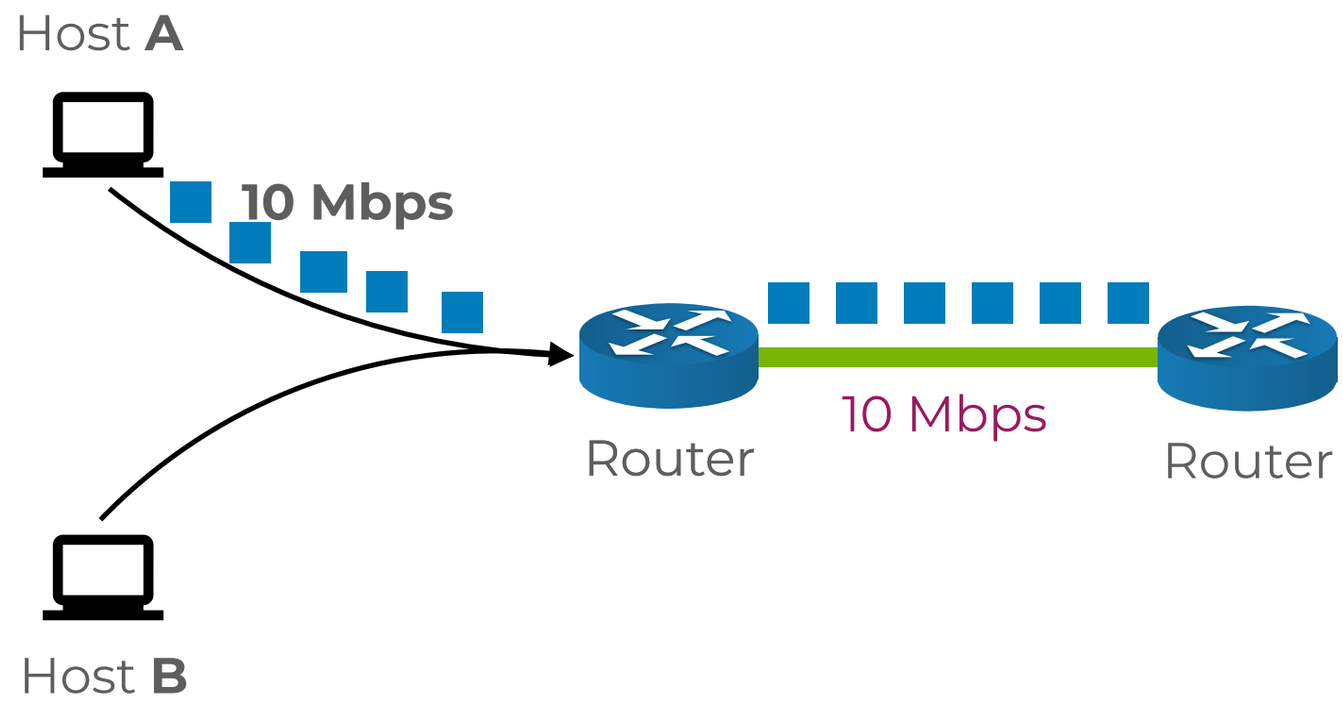
Assume UDP, or TCP with no congestion control



System has finite **capacity**.

Offered **load** is at or below capacity.

Assume UDP, or TCP with no congestion control

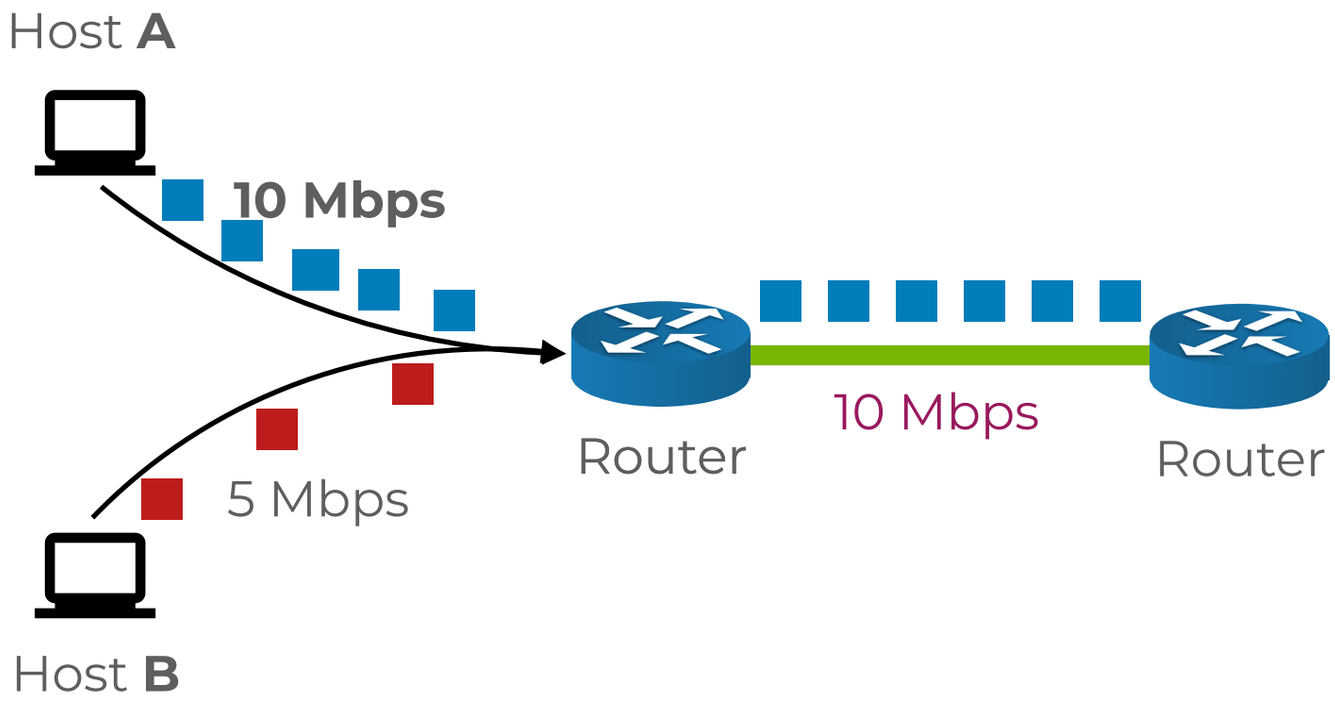


System has finite **capacity**.

Offered **load** is at or below capacity.

Assume UDP, or TCP with no congestion control

Congestion

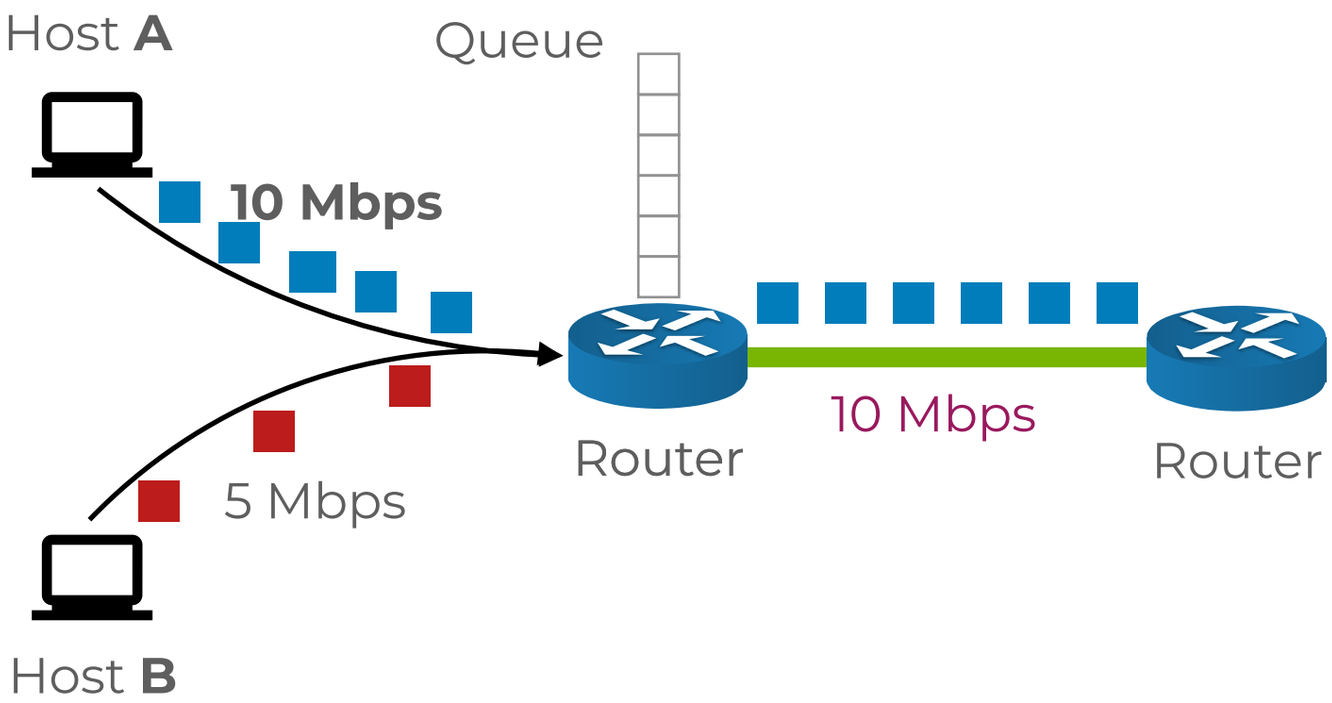


System has finite **capacity**.

Offered **load** exceeds capacity.

Assume UDP, or TCP with no congestion control

Congestion

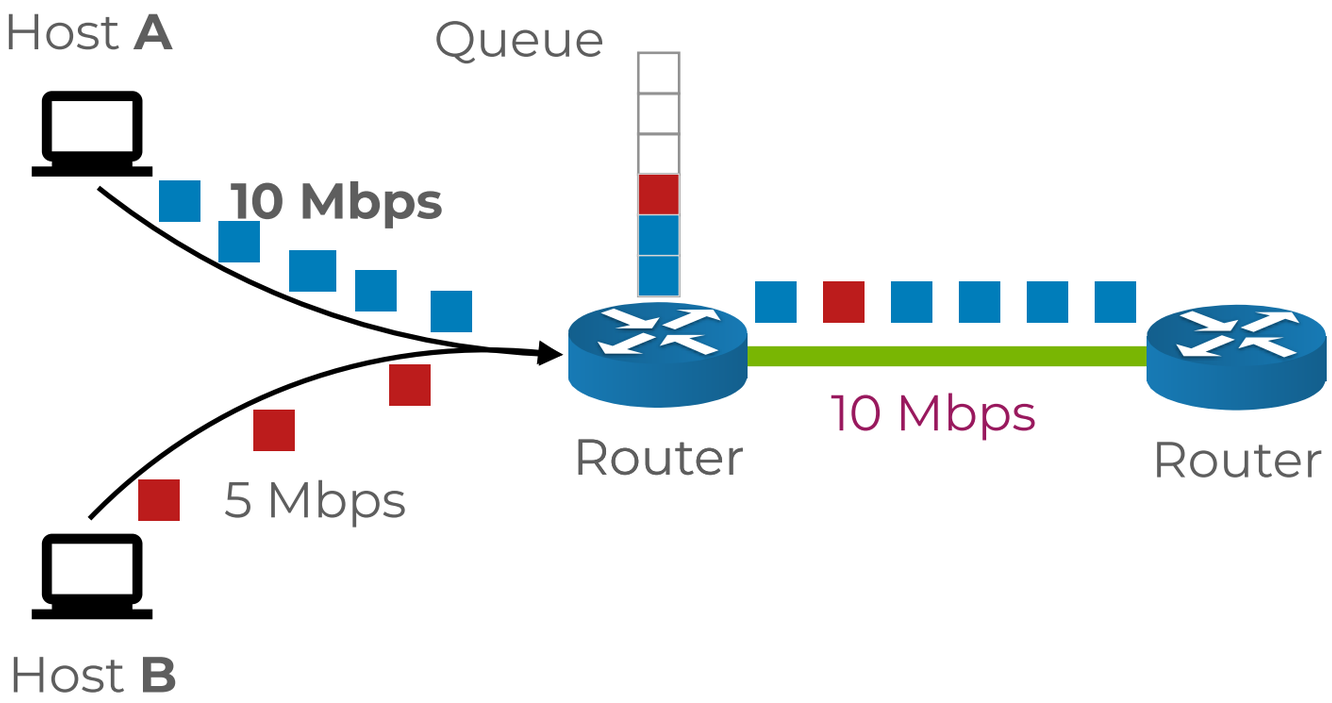


System has finite **capacity**.

Offered **load** exceeds capacity.

Assume UDP, or TCP with no congestion control

Congestion

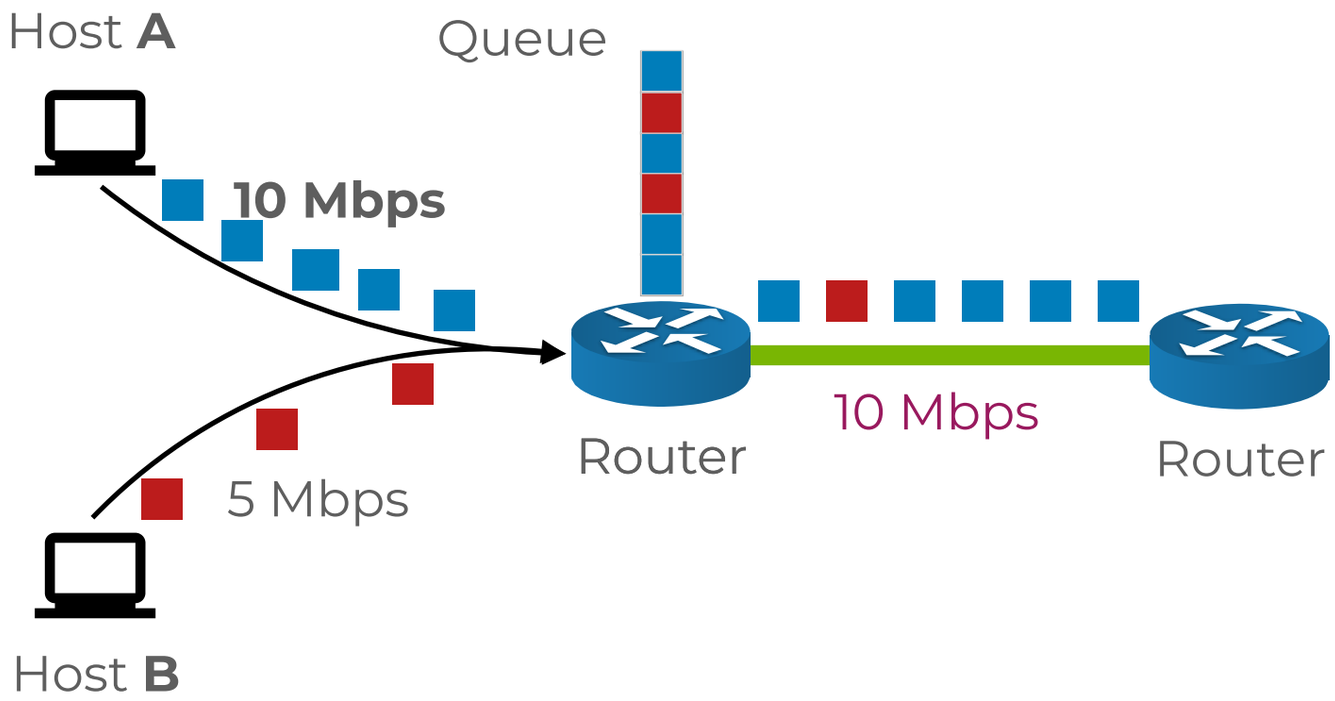


System has finite **capacity**.

Offered **load** exceeds capacity.

Assume UDP, or TCP with no congestion control

Congestion

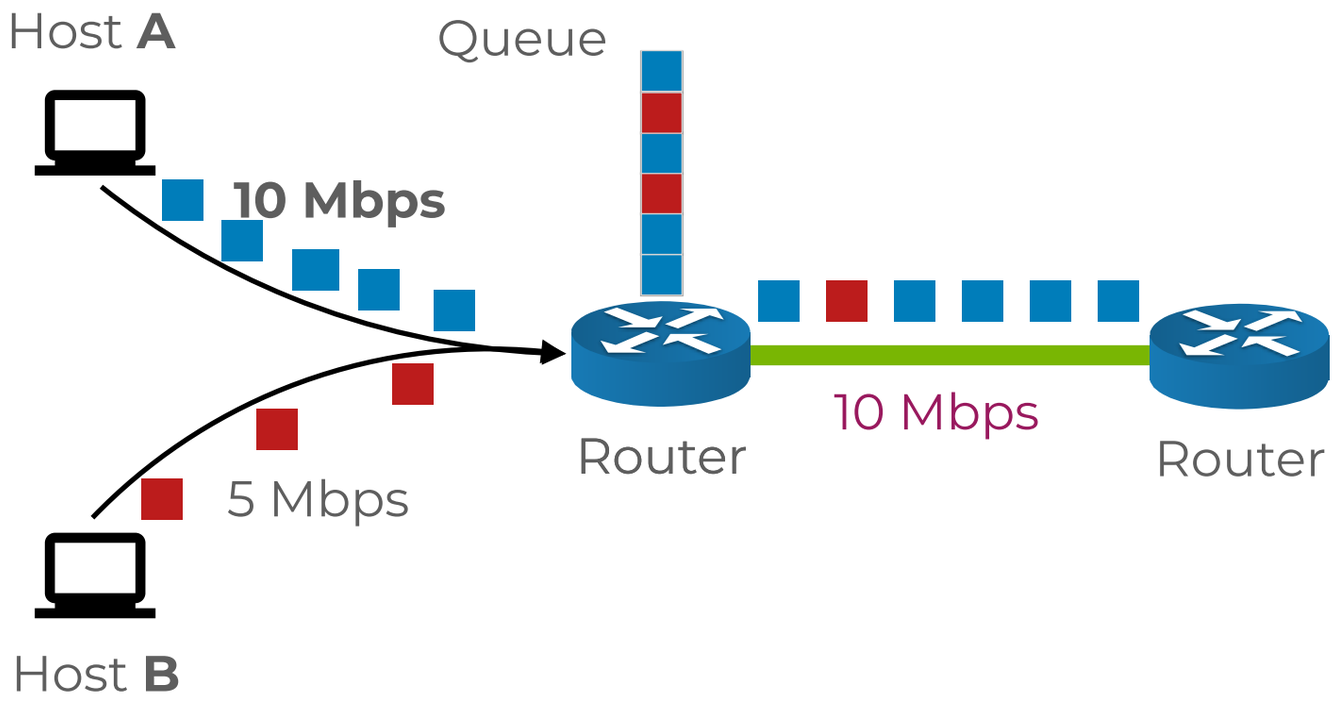


System has finite **capacity**.

Offered **load** exceeds capacity.

Assume UDP, or TCP with no congestion control

Congestion



System has finite **capacity**.

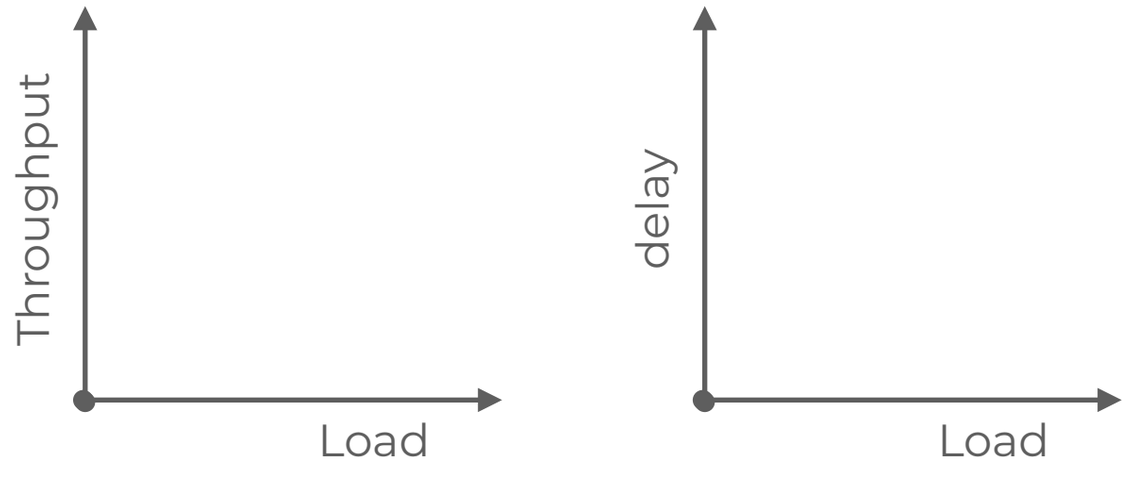
Offered **load** exceeds capacity.

Assume UDP, or TCP with no congestion control

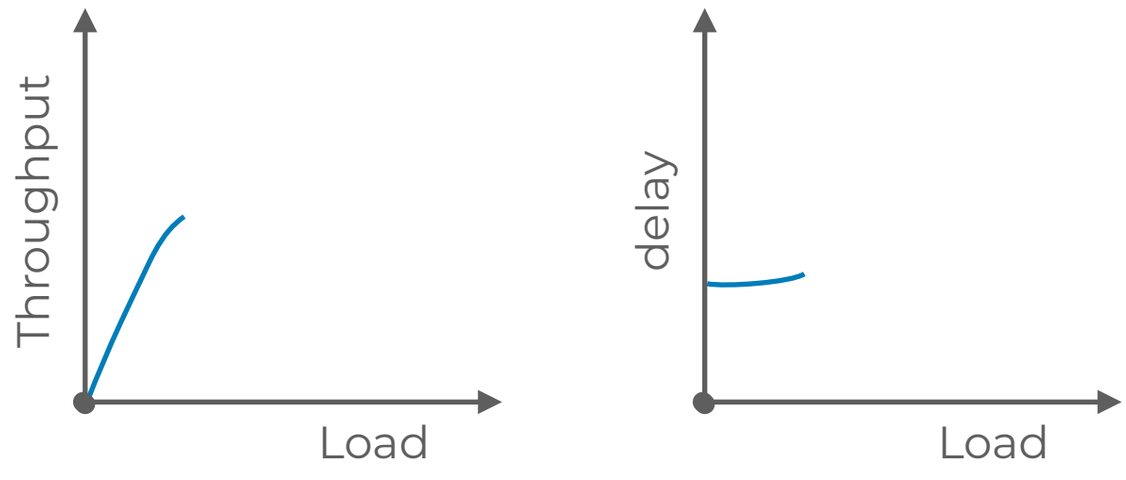
System performance

System performance

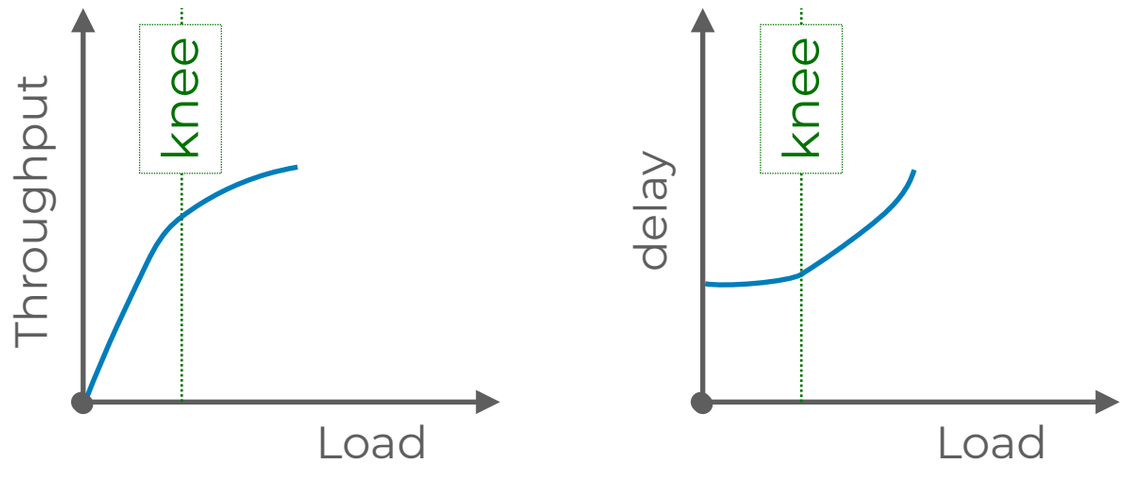
System performance



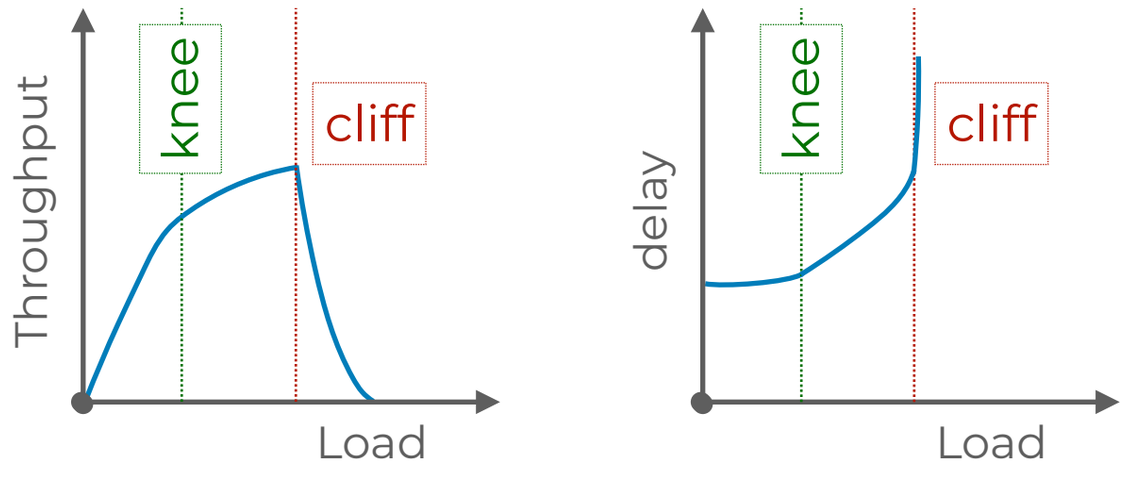
System performance



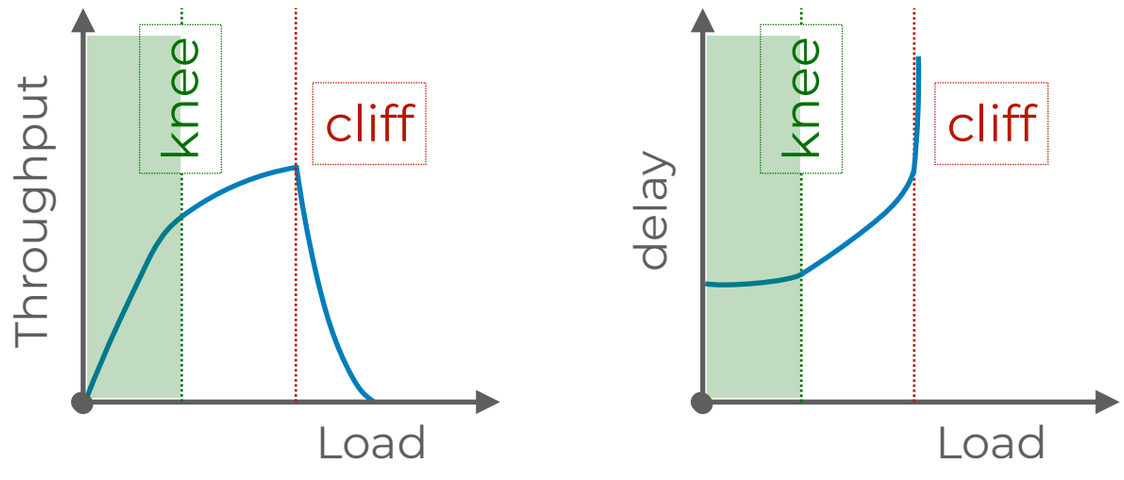
System performance



System performance

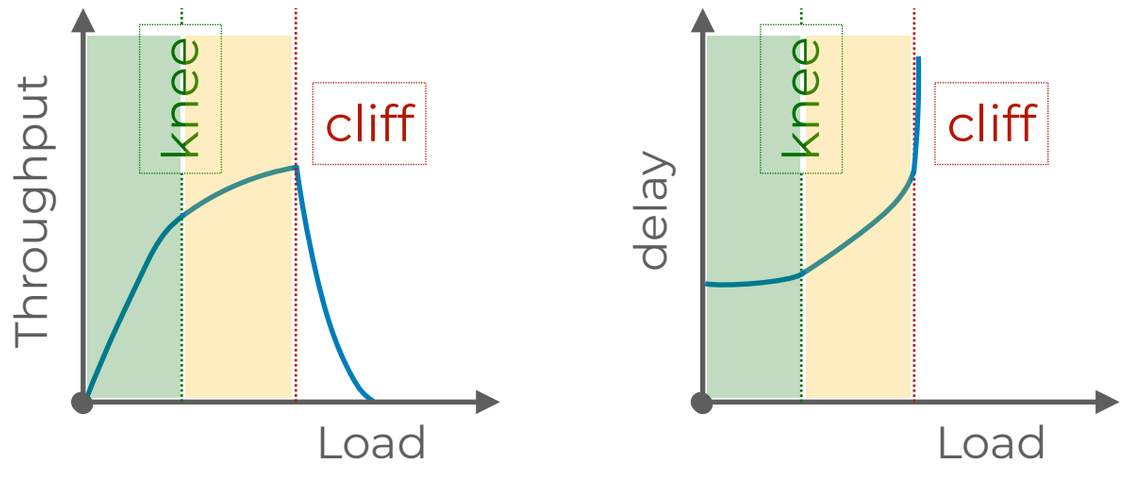


System performance



» **Congestion avoidance** Stay left of “knee”

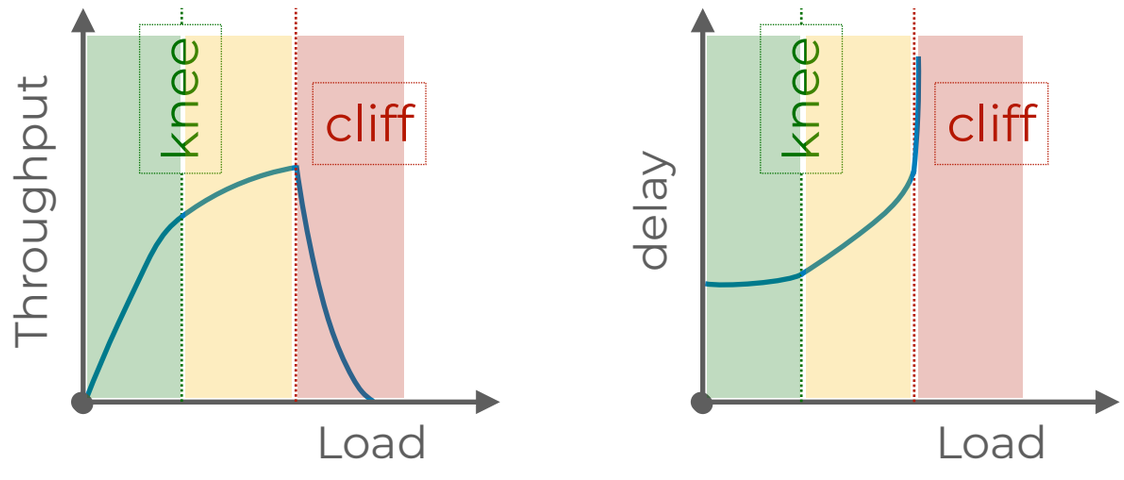
System performance



» **Congestion avoidance** Stay left of “knee”

» **Congestion control** Stay left of “cliff”

System performance



» **Congestion avoidance** Stay left of “knee”

» **Congestion control** Stay left of “cliff”

» **Congestion collapse** Avoid at all costs

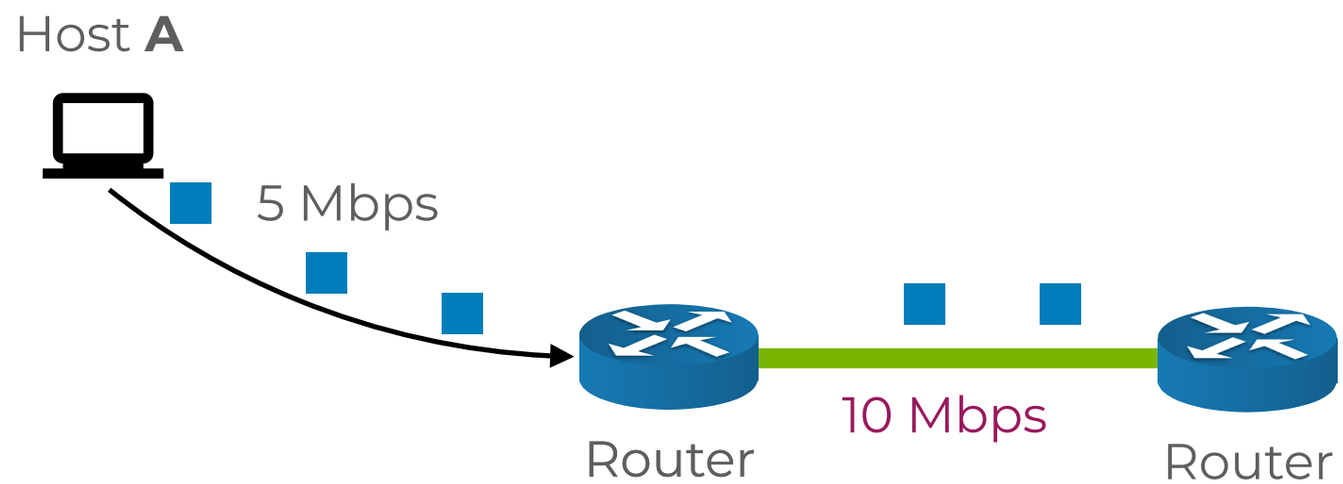
Did you know?

- Congestion collapse is real, and was first observed in 1986 in NSFnet.
- TCP initially did not have congestion control!

Congestion Control: Challenges & Goals

- What are the key challenges and objectives for congestion control?

Why is it a hard problem?

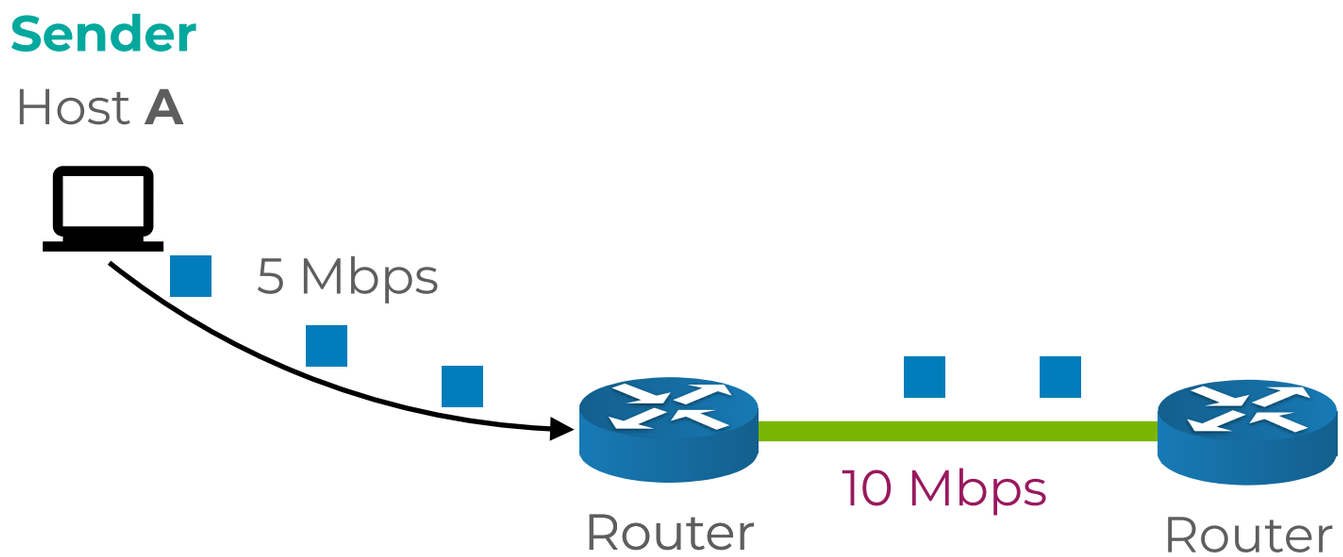


System has finite **capacity**.

Offered **load** is at or below capacity.

VU VRIJE UNIVERSITEIT AMSTERDAM

Why is it a hard problem?

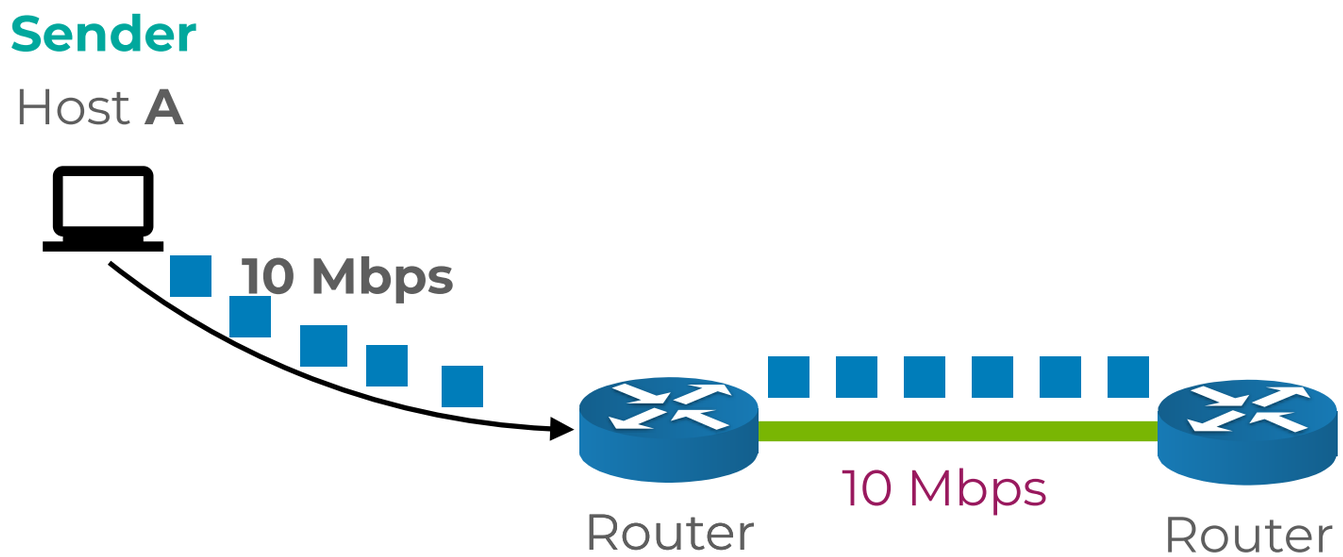


System has finite **capacity**.

Offered **load** is at or below capacity.

VU VRIJE UNIVERSITEIT AMSTERDAM

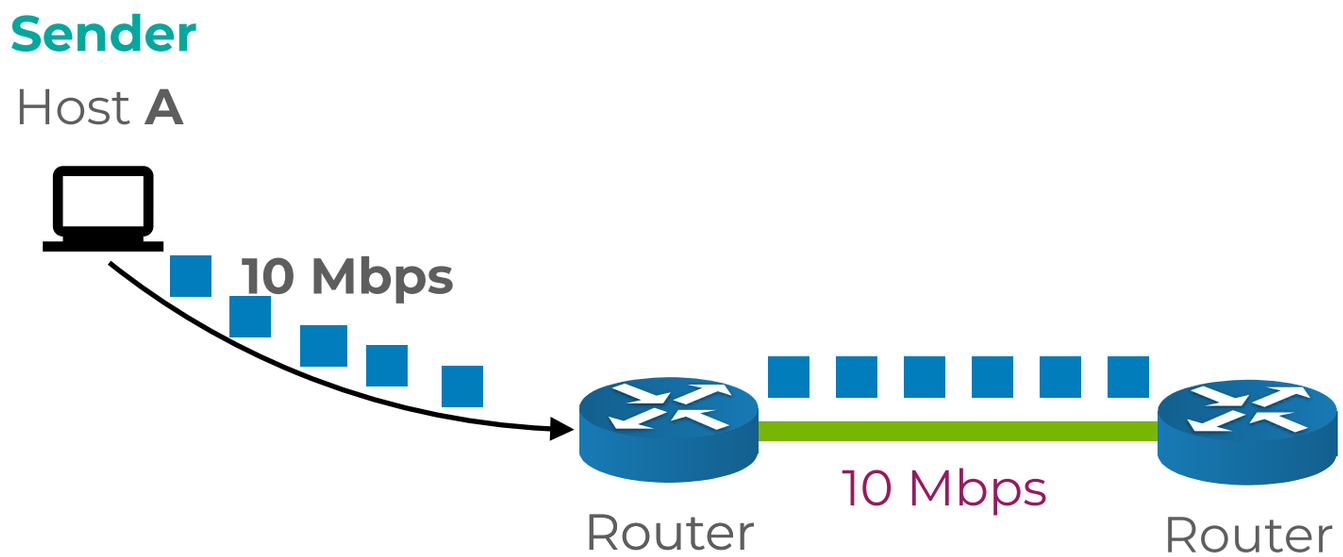
Why is it a hard problem?



System has finite **capacity**.

Offered **load** is at or below capacity.

Why is it a hard problem?

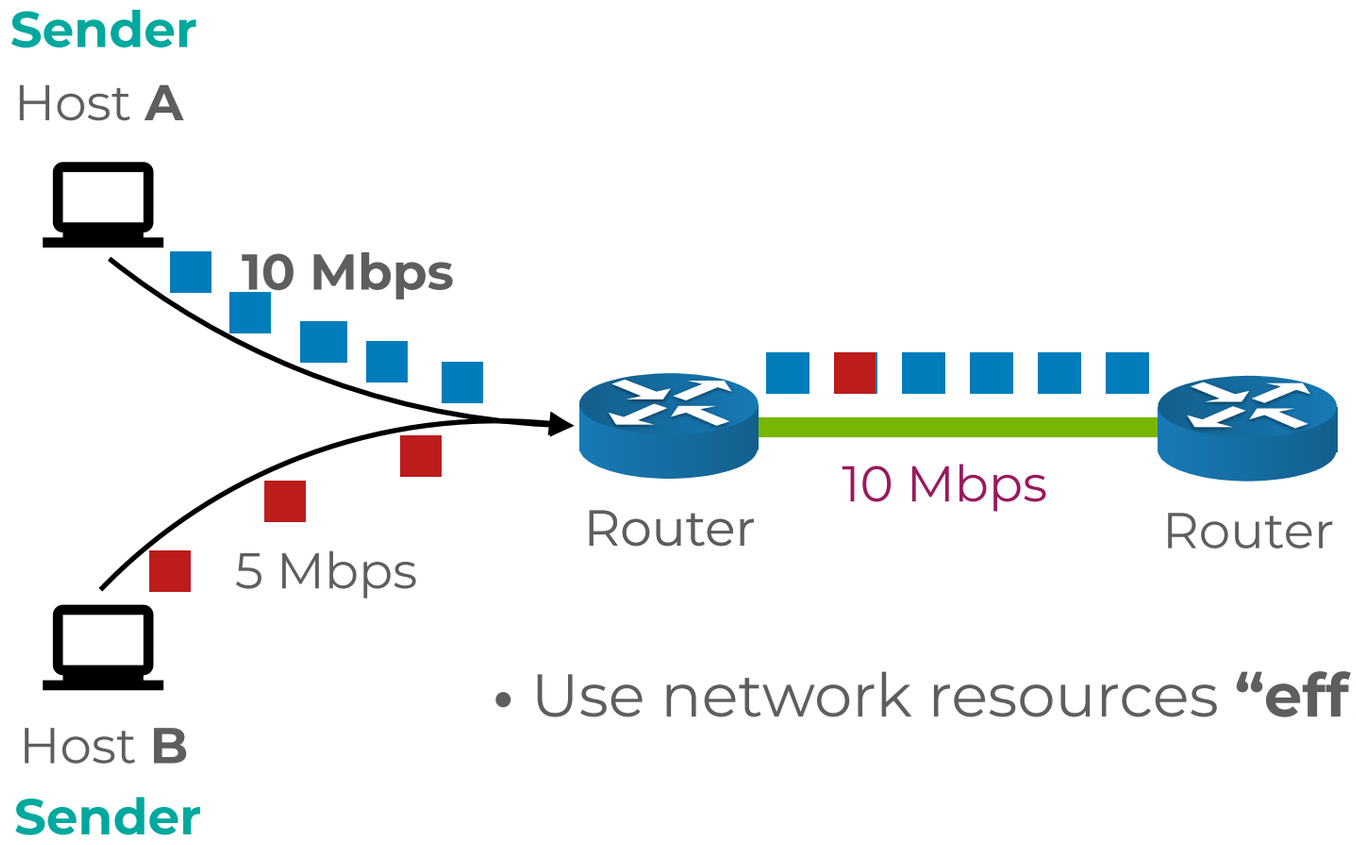


System has finite **capacity**.

Offered **load** is at or below capacity.

- Use network resources **“efficiently”**

Why is it a hard problem?

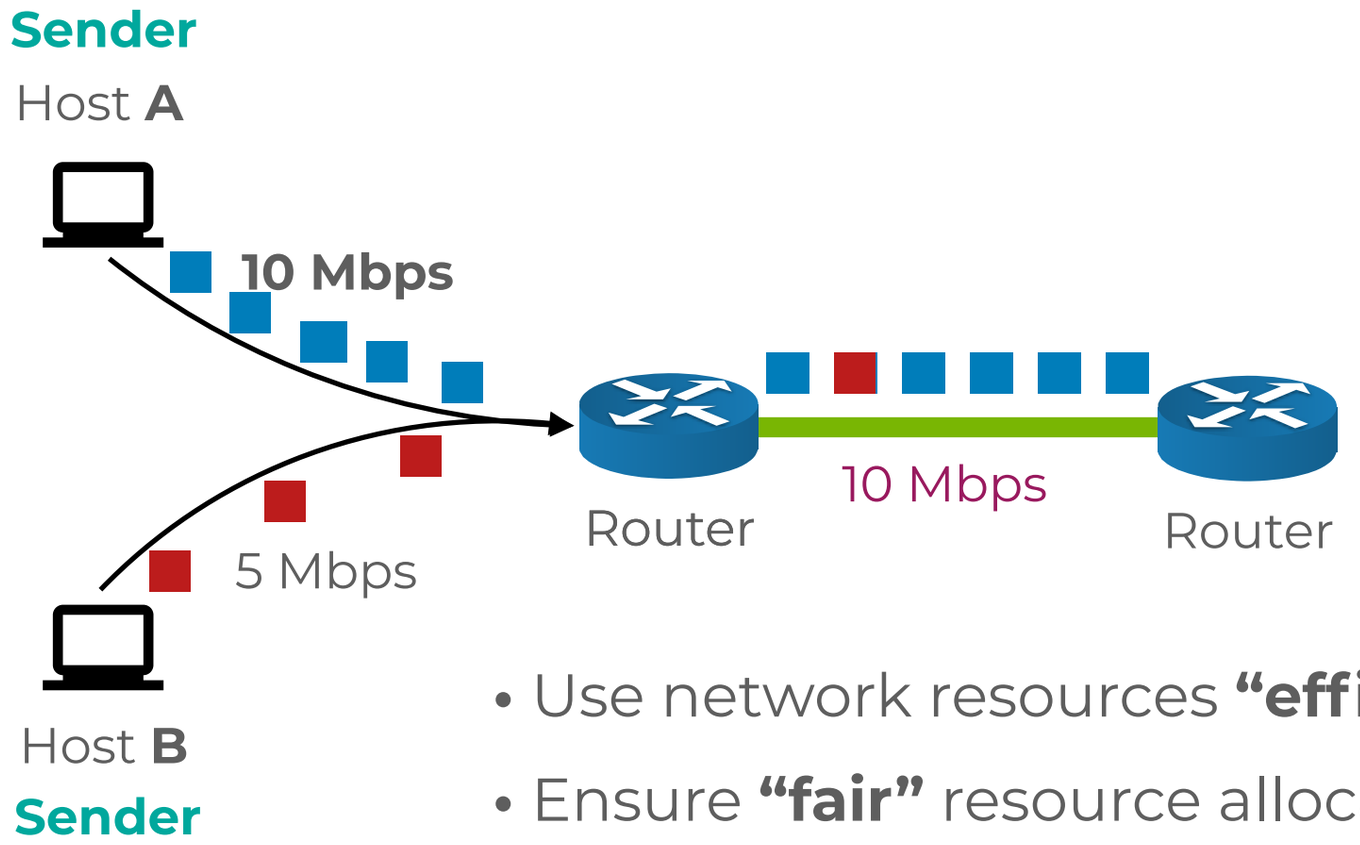


System has finite **capacity**.

Offered **load** exceeds capacity.

- Use network resources **“efficiently”**

Why is it a hard problem?

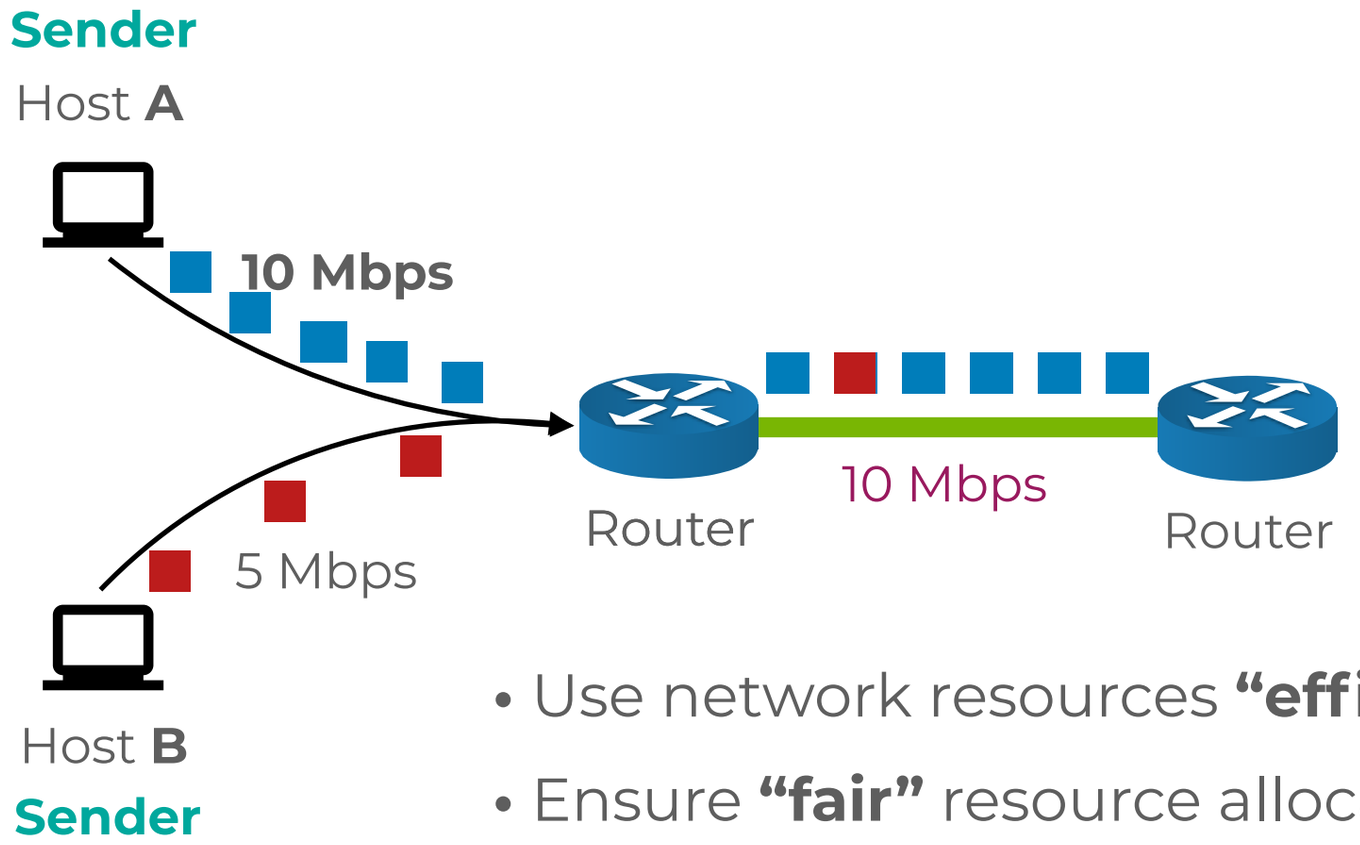


System has finite **capacity**.

Offered **load** exceeds capacity.

- Use network resources **“efficiently”**
- Ensure **“fair”** resource allocation

Why is it a hard problem?

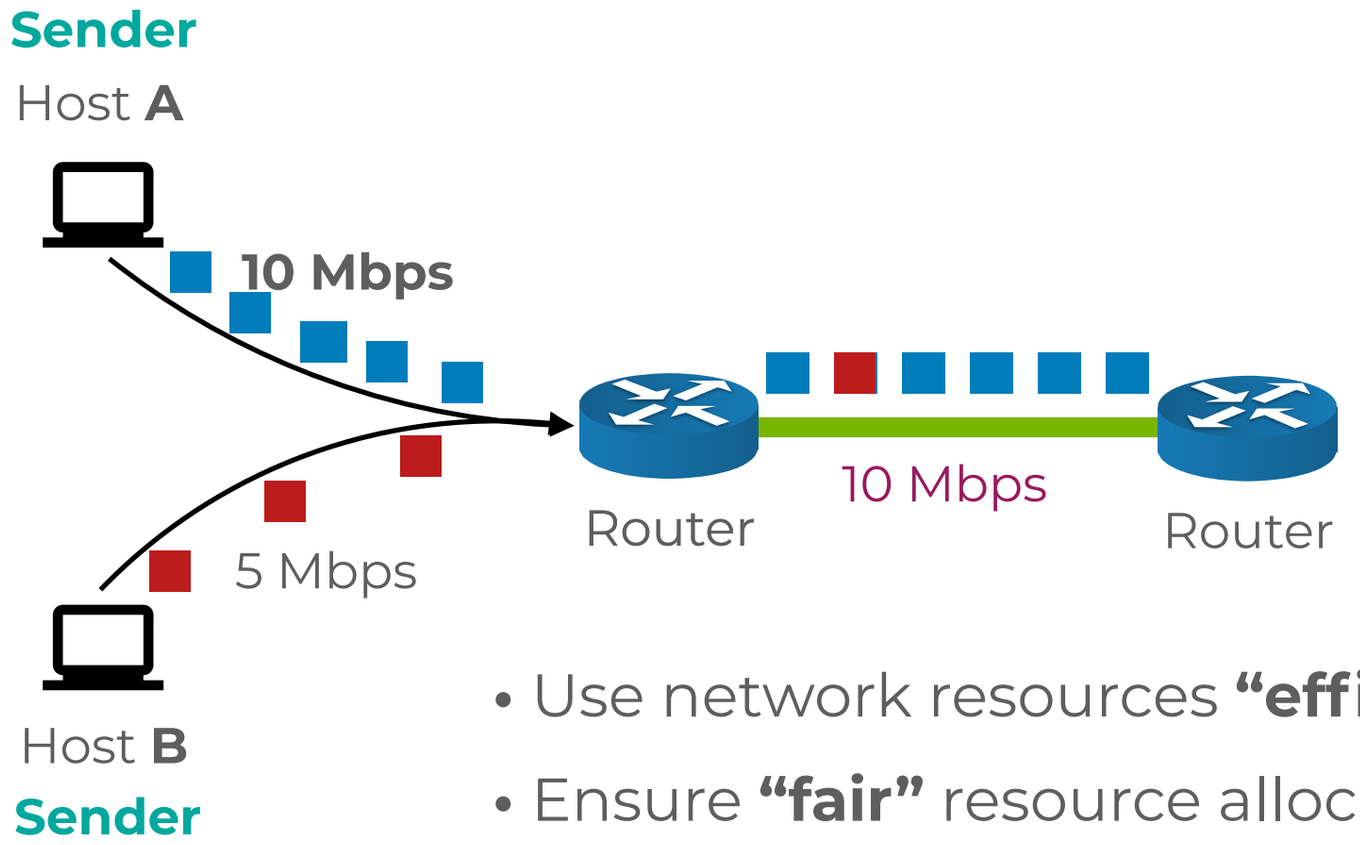


System has finite **capacity**.

Offered **load** exceeds capacity.

- Use network resources **“efficiently”**
- Ensure **“fair”** resource allocation
- Adapt **“quickly”**

Why is it a hard problem?

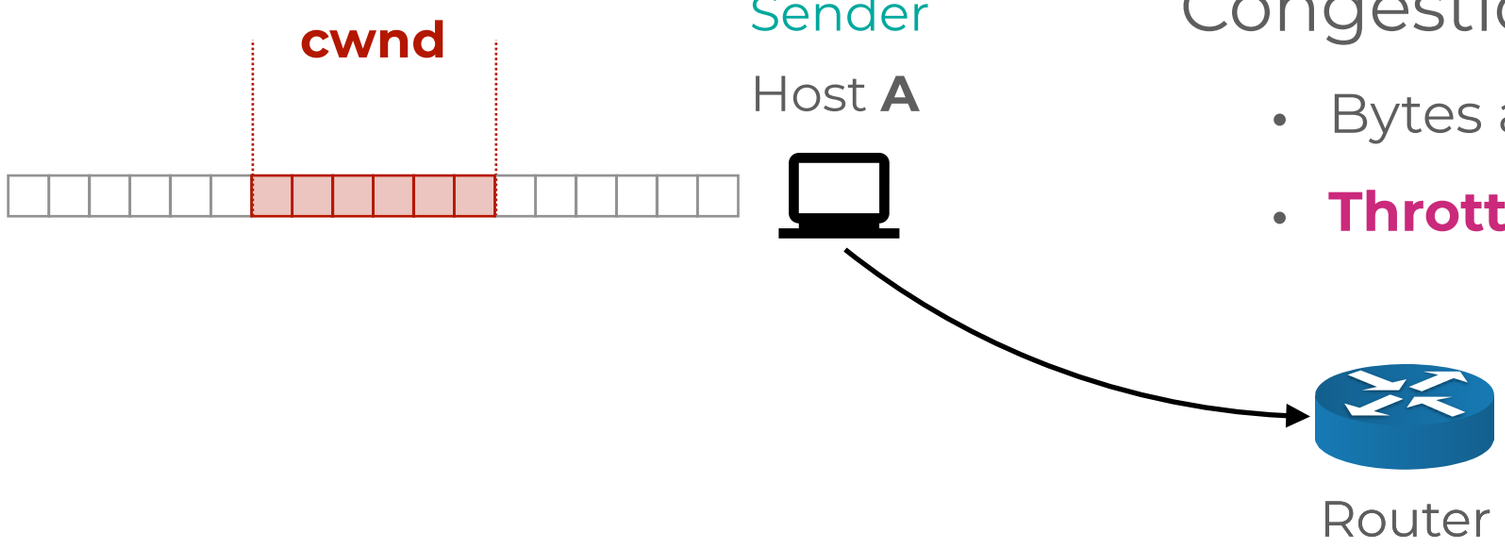


System has finite **capacity**.

Offered **load** exceeds capacity.

- Use network resources **“efficiently”**
- Ensure **“fair”** resource allocation
- Adapt **“quickly”**
- No **“explicit coordination”**

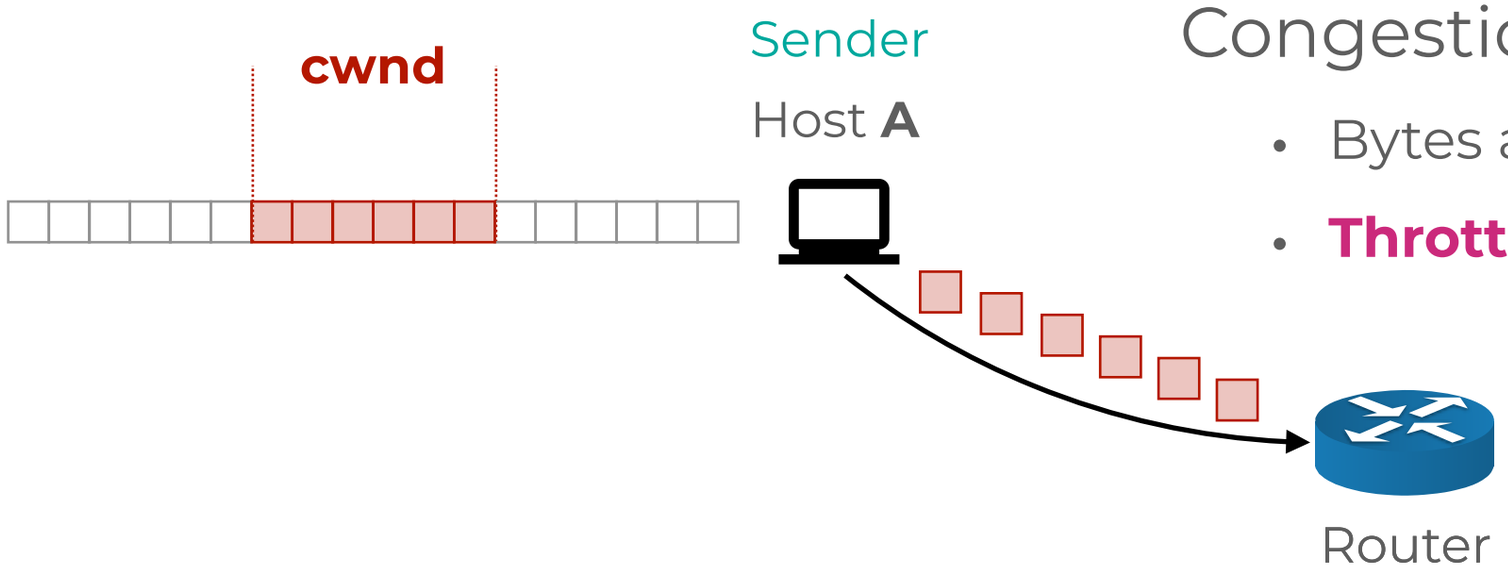
Window-based control



Congestion window (cwnd)

- Bytes awaiting acknowledgement
- **Throttles** sender

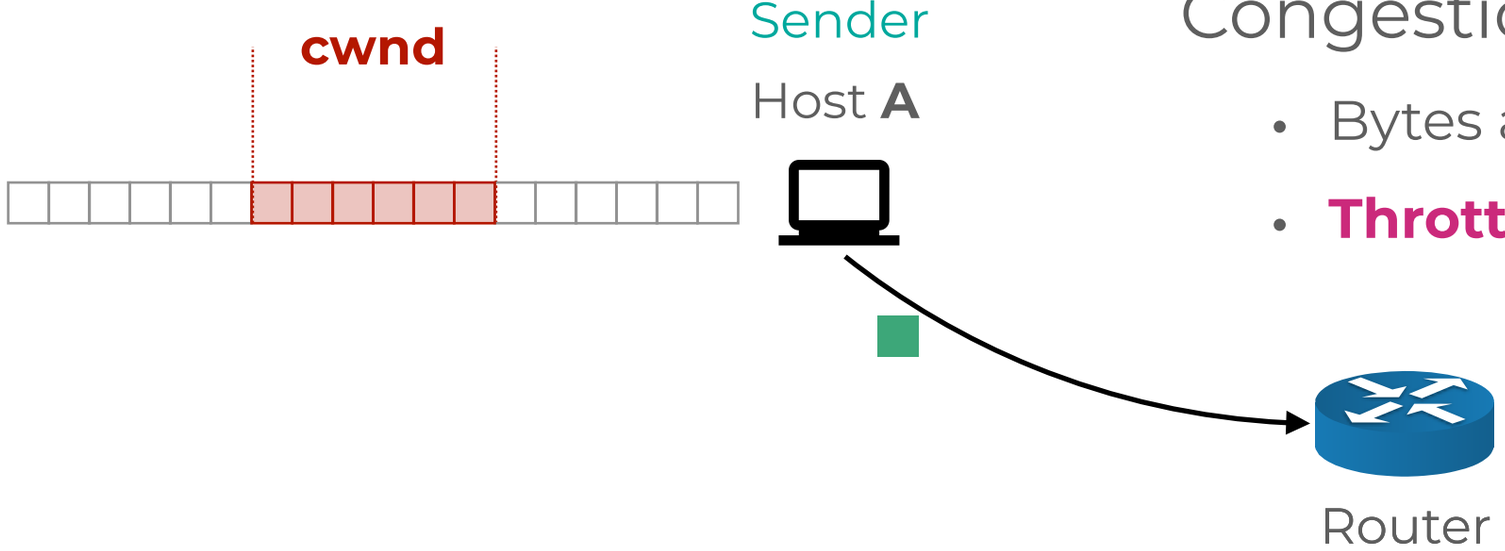
Window-based control



Congestion window (cwnd)

- Bytes awaiting acknowledgement
- **Throttles** sender

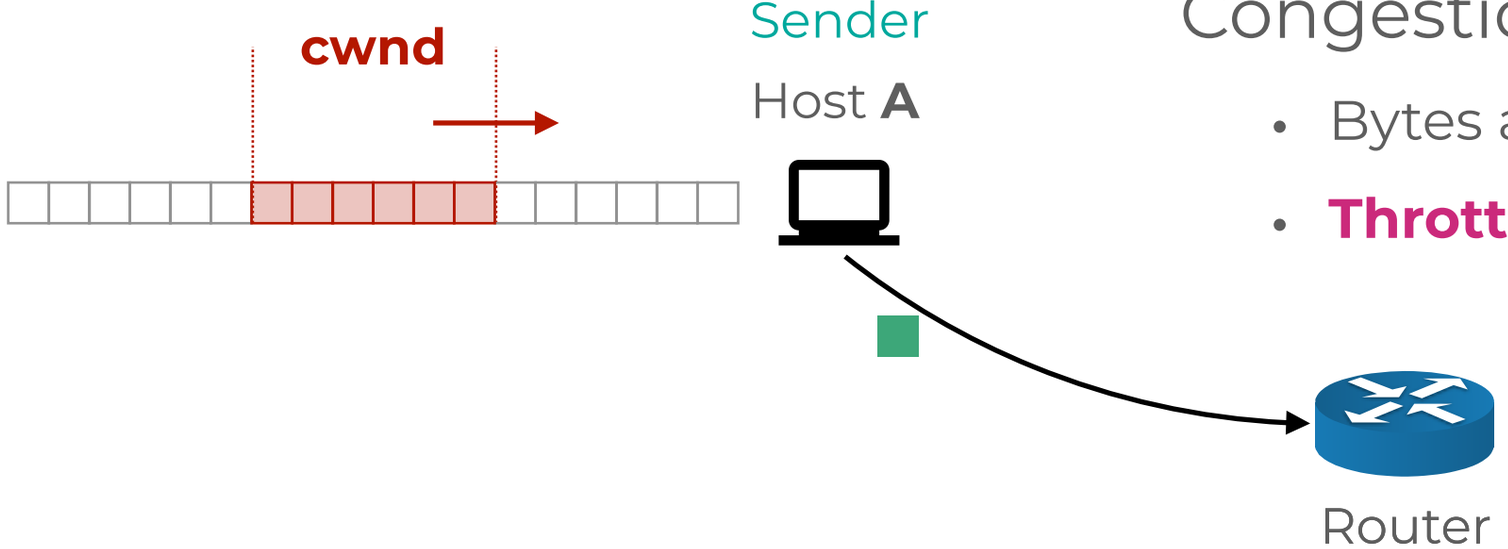
Window-based control



Congestion window (cwnd)

- Bytes awaiting acknowledgement
- **Throttles** sender

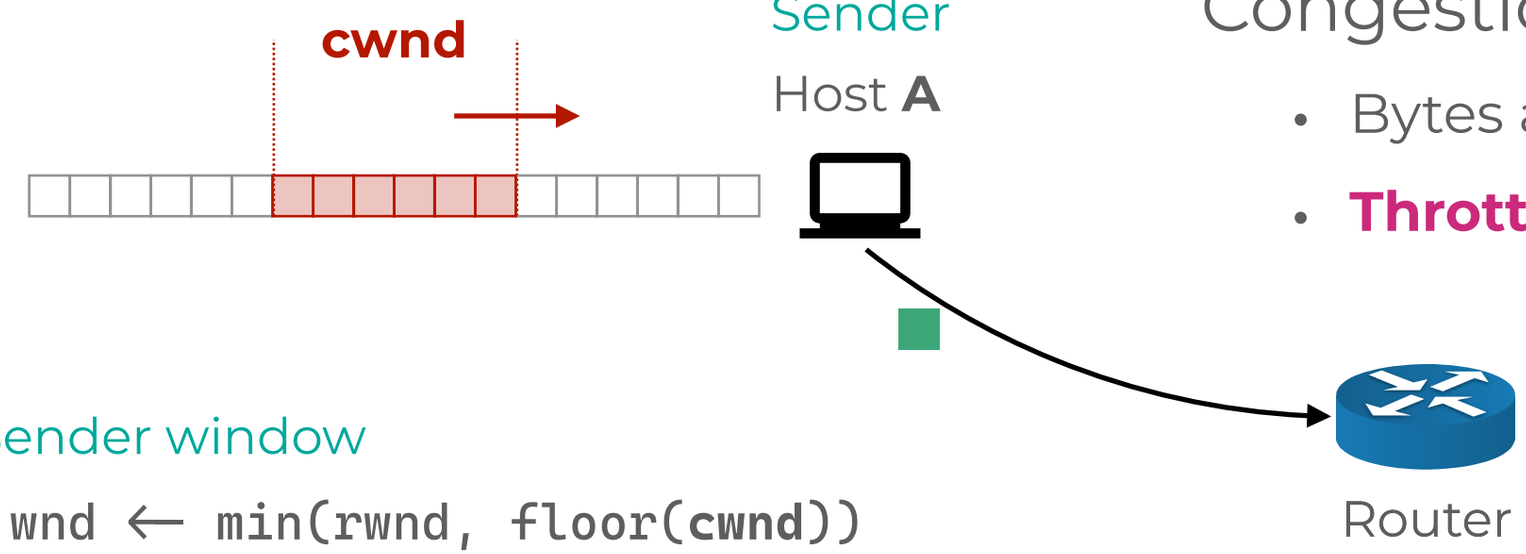
Window-based control



Congestion window (cwnd)

- Bytes awaiting acknowledgement
- **Throttles** sender

Window-based control



Congestion window (cwnd)

- Bytes awaiting acknowledgement
- **Throttles** sender

Sender window

```
wnd ← min(rwnd, floor(cwnd))
```

Window-based control

Congestion window (cwnd)

- Bytes awaiting acknowledgement
- **Throttles** sender

```
wnd ← min(rwnd, floor(cwnd))
```

If there’s no congestion

- **Increase** cwnd

On detecting congestion

- **Decrease** cwnd

Window-based control

Congestion window (cwnd)

- Bytes awaiting acknowledgement
- **Throttles** sender

If there's no congestion

- **Increase** cwnd

On detecting congestion

- **Decrease** cwnd

```
wnd ← min(rwnd, floor(cwnd))
```

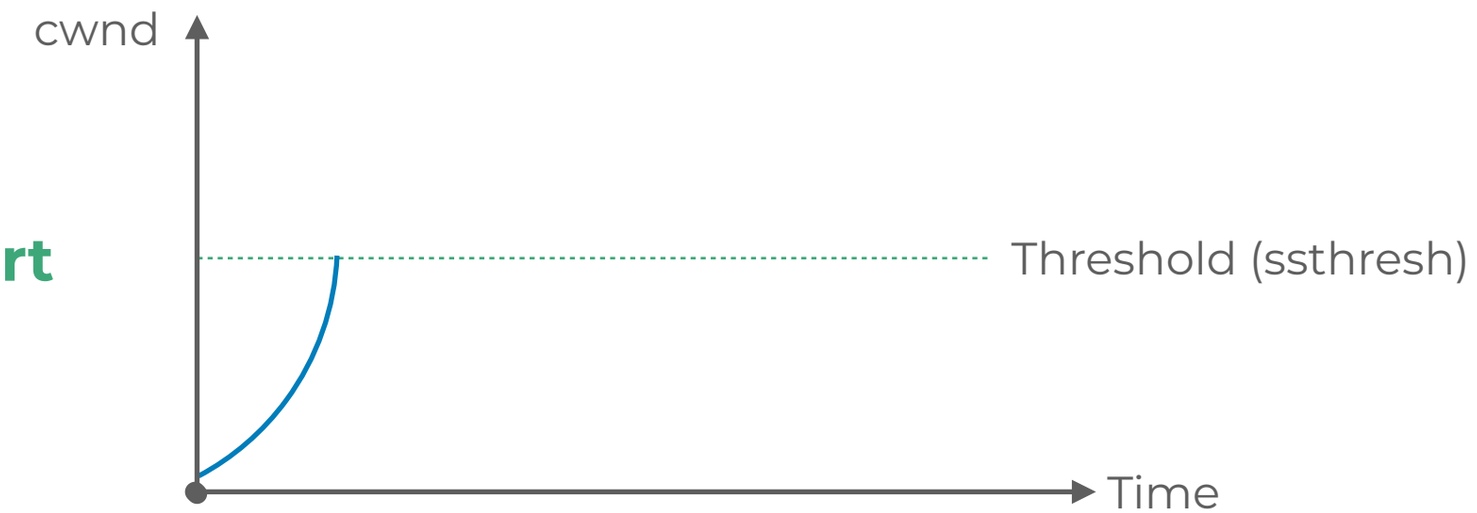
Adaptation

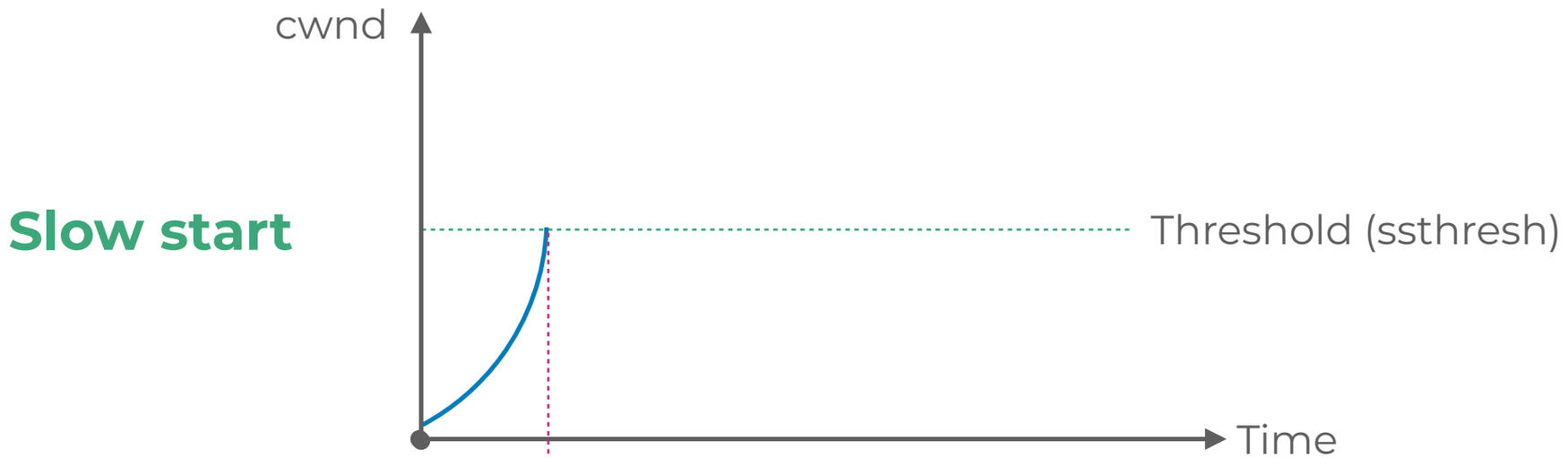

Time

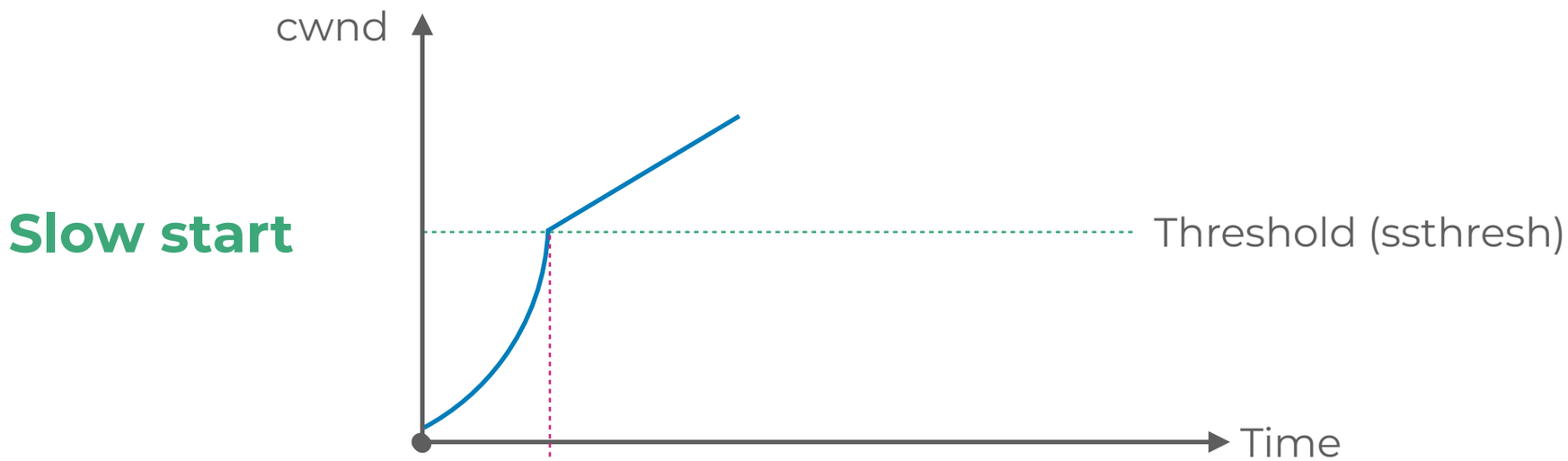

cwnd
Time

Slow start

Slow start

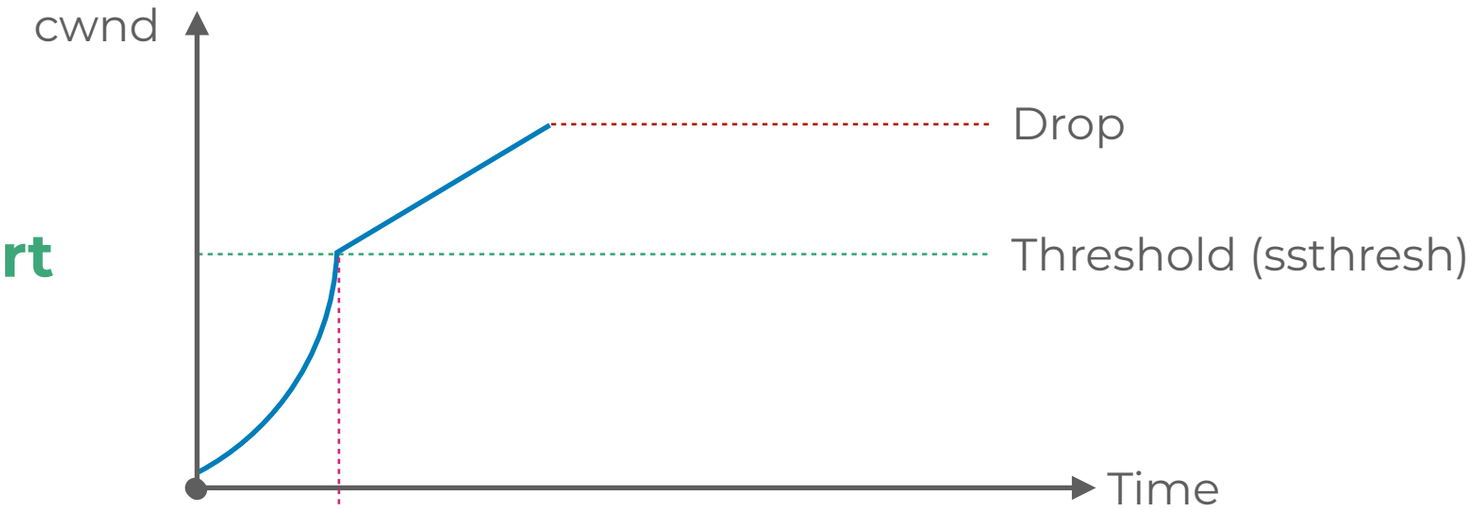



cwnd
Slow start
Threshold (ssthresh)
Time
Quickly probe path bandwidth



Quickly probe path bandwidth

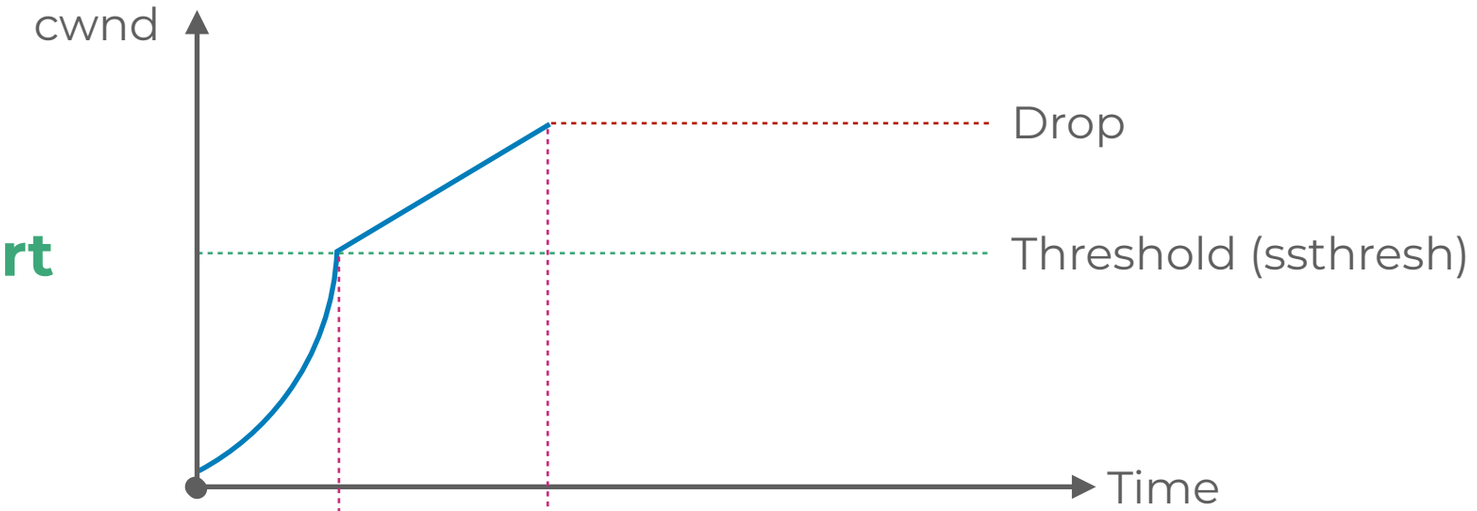
Congestion avoidance



Slow start

Quickly probe path bandwidth

Congestion avoidance

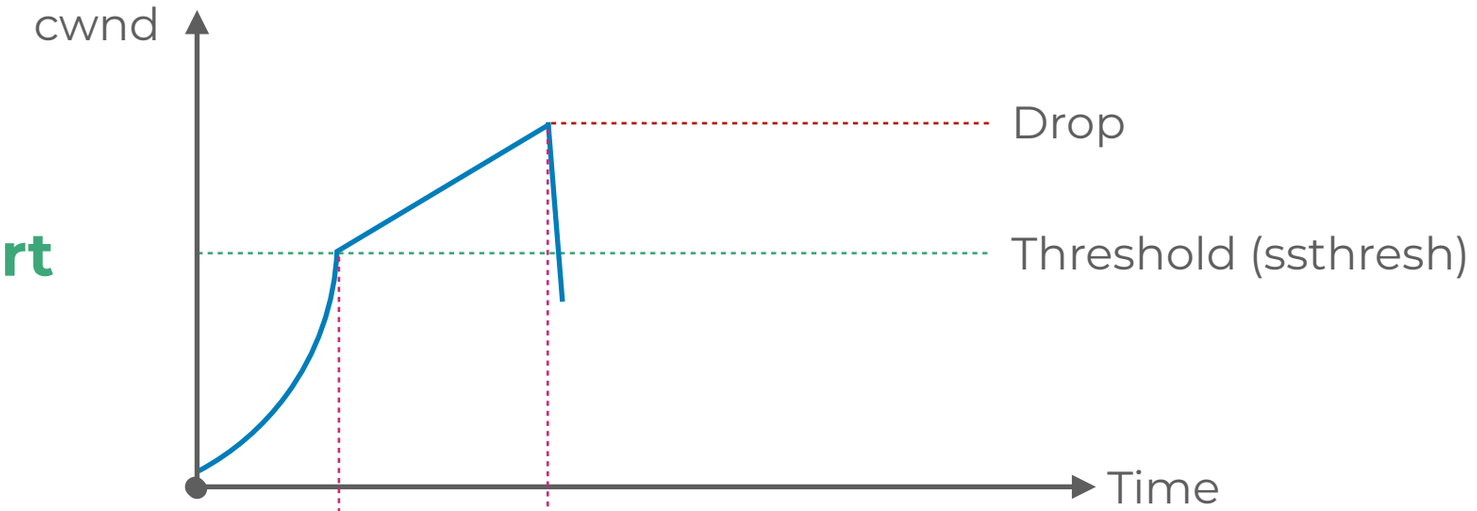


Slow start

Quickly probe path bandwidth

Cautiously probe available bandwidth

Congestion avoidance

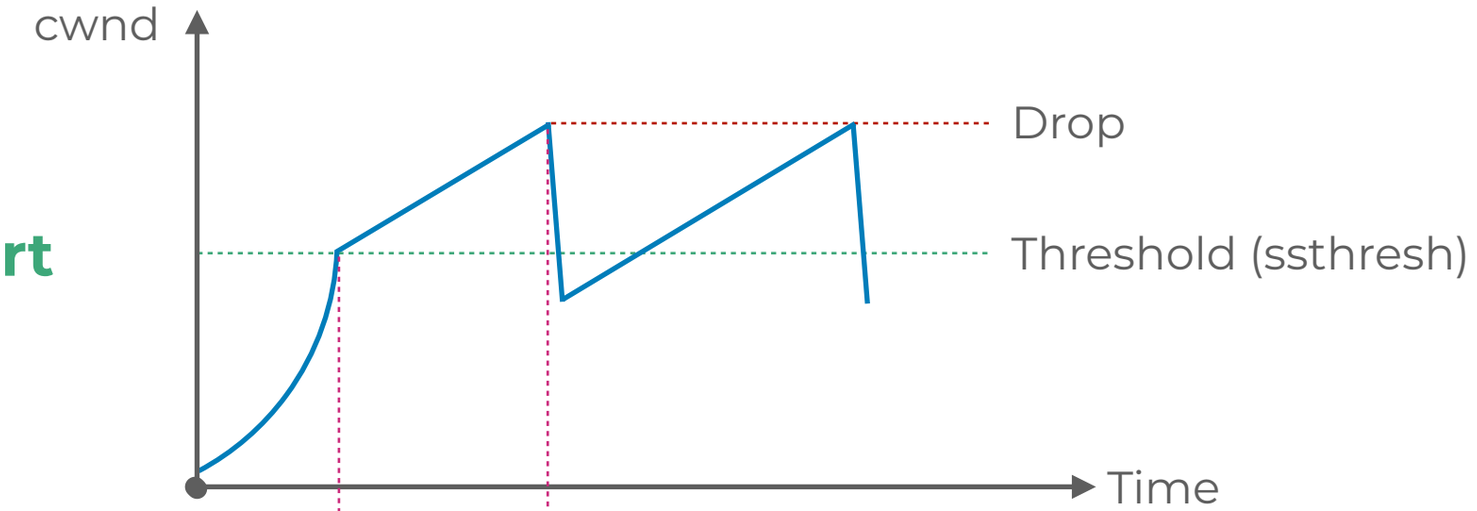


Slow start

Quickly probe path bandwidth

Cautiously probe available bandwidth

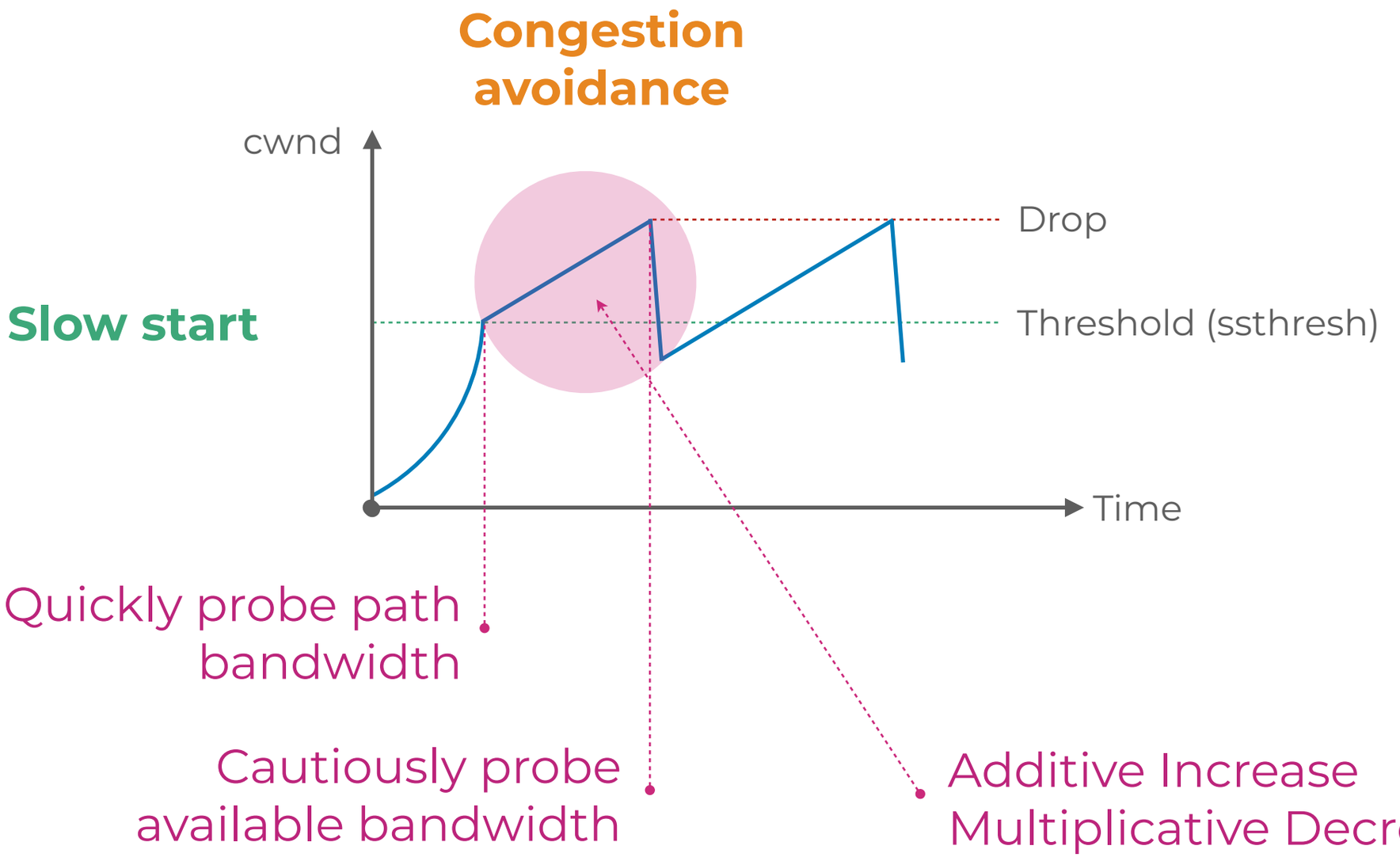
Congestion avoidance



Slow start

Quickly probe path bandwidth

Cautiously probe available bandwidth


Congestion avoidance
cwnd
Drop
Slow start
Threshold (ssthresh)
Time
Quickly probe path bandwidth
Cautiously probe available bandwidth
Additive Increase Multiplicative Decrease