CS 43: Computer Networks

TCP Connections, Flow Control, Congestion Control

November 2, 2026



Reading Quiz
(graded for correctness)

Put devices away

Note: Reading quiz slides will have an orange border.

End of Reading Quiz!

Announcements

- Posted mid-semester grades in Moodle gradebook.
- OH on Monday will be full of protocol meetings. If you'd like to talk to me today, please send me an email.
- This week my availability is very limited because of protocol meetings and advising week.

Practical Reliability Questions

- What does connection establishment look like?
- How do we choose sequence numbers?
- How should we choose timeout values?
- How do the sender and receiver keep track of outstanding pipelined segments?
- How many segments should be pipelined?

Timeouts

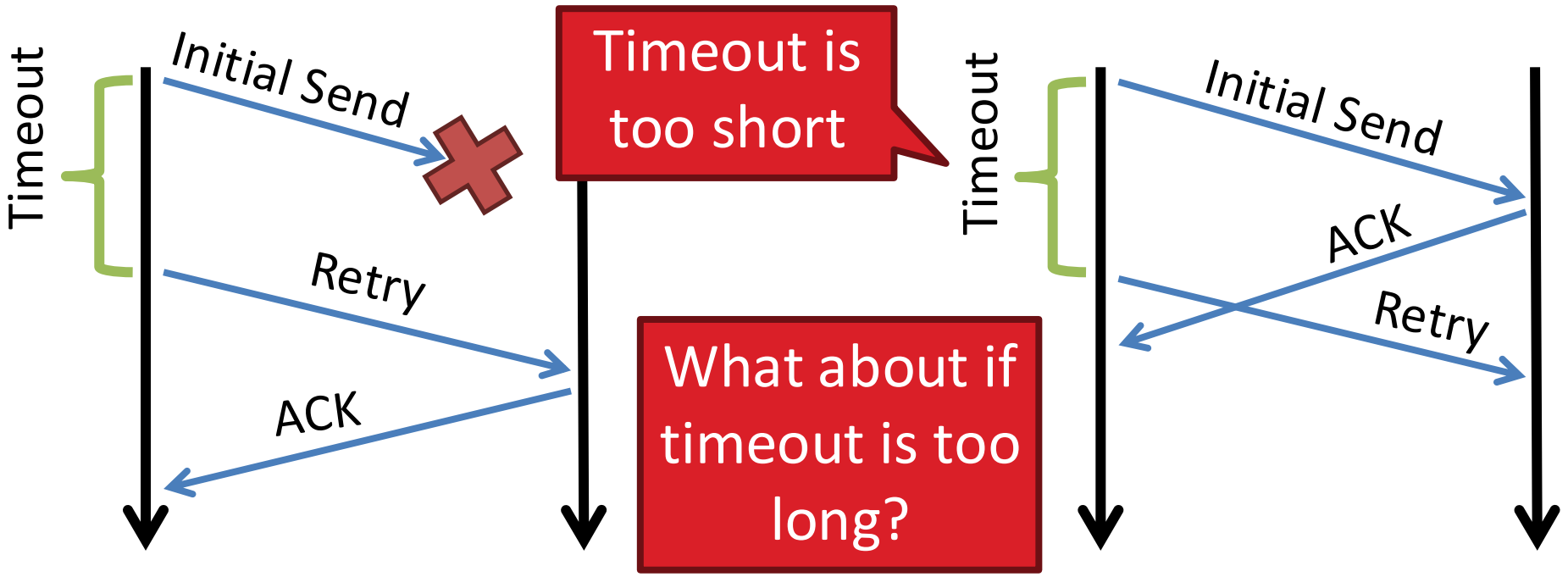
- How long should we wait before timing out and retransmitting a segment?

- Too short: needless retransmissions
- Too long: slow reaction to losses

- Should be (a little bit) longer than the RTT

Retransmission Timeouts

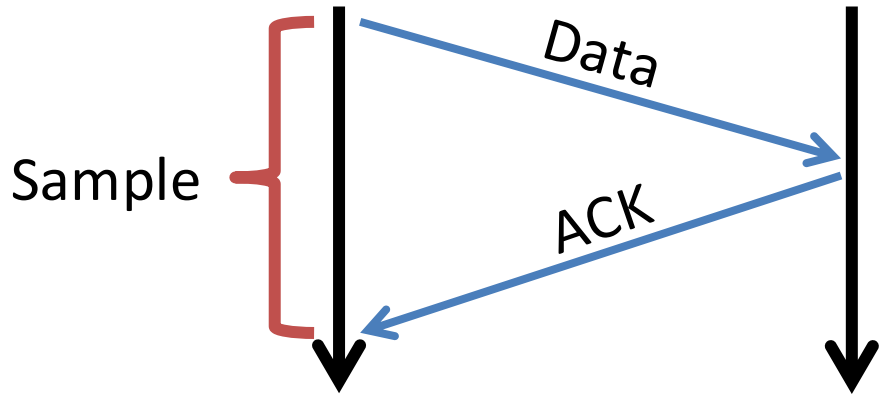
- Problem: time-out is linked to round trip time



Estimating RTT

- Problem: RTT changes over time
  - Routers buffer packets in queues
  - Queue lengths vary
  - Receiver may have varying load
- Sender takes measurements
  - Use statistics to decide future timeouts for sends
  - Estimate RTT and variance
- Apply “smoothing” to account for changes

Round Trip Time Estimation:
Exponentially Weighted Moving Average (EWMA)



EstimatedRTT = (1 – a) * EstimatedRTT + a * SampleRTT

- a is usually 1/8.

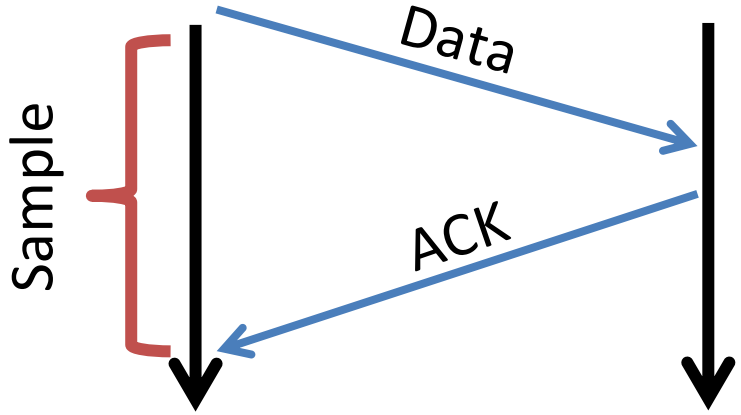
In words current estimate is a blend of:

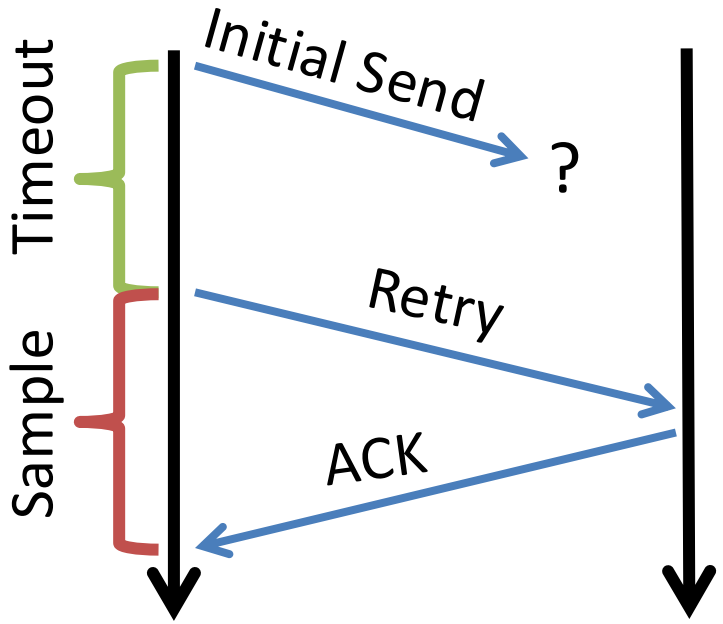
- 7/8 of the previous estimate
- 1/8 of the new sample.

DevRTT = (1 – B) * DevRTT + B * | SampleRTT – EstimatedRTT |

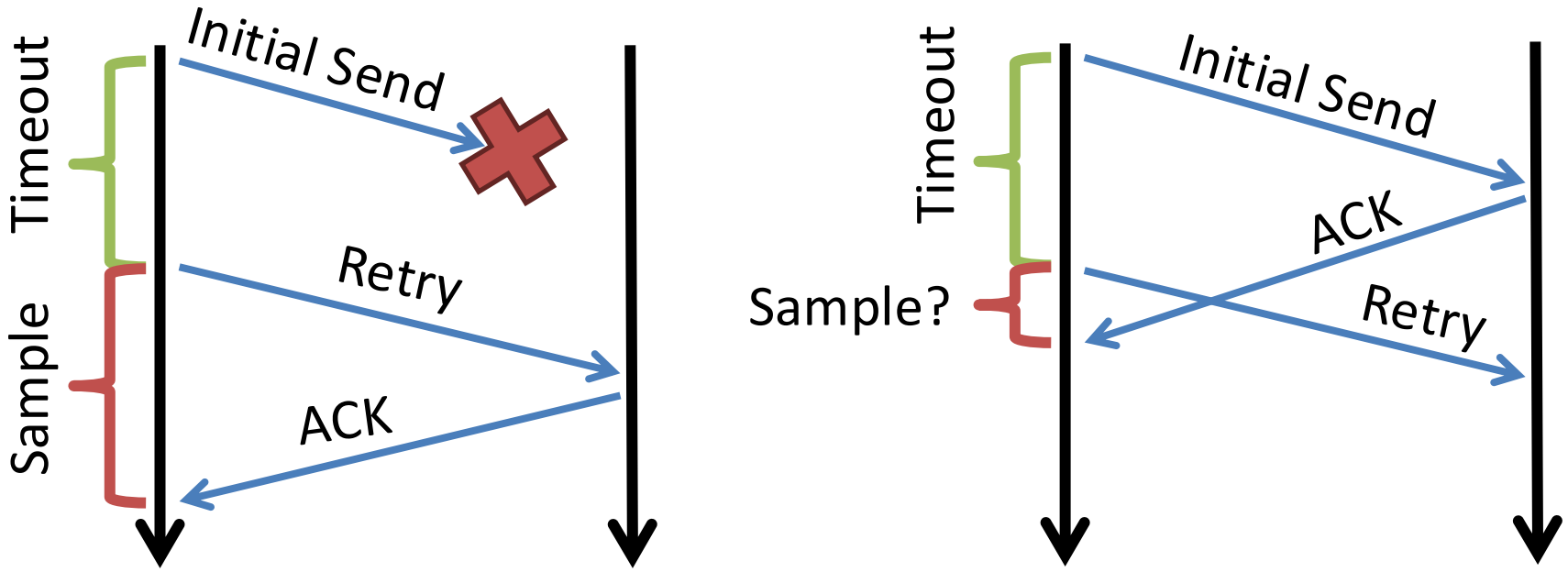
- B is usually 1/4

Round Trip Time Estimation:
Exponentially Weighted Moving Average (EWMA)





RTT Sample Ambiguity



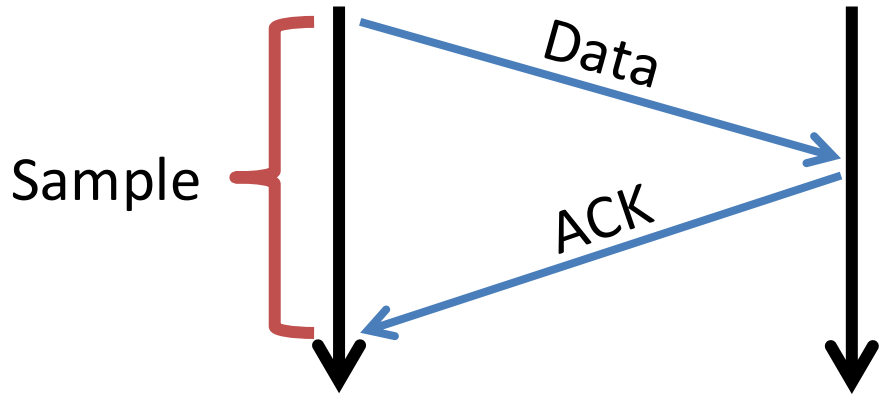
Ignore samples for retransmitted segments

Estimating RTT

- For each segment that did not require a retransmit (ACK heard without a timeout)
  - Consider the time between segment sent and ACK received to be a sample of the current RTT
  - Use that, along with previous history, to update the current RTT estimate

- Exponentially Weighted Moving Average (EWMA)

Round Trip Time Estimation:
Exponentially Weighted Moving Average (EWMA)



EstimatedRTT = (1 – a) * EstimatedRTT + a * SampleRTT

- a is usually 1/8.

In words current estimate is a blend of:

- 7/8 of the previous estimate
- 1/8 of the new sample.

DevRTT = (1 – B) * DevRTT + B * | SampleRTT – EstimatedRTT |

- B is usually 1/4

TCP Timeout Value

`TimeoutInterval = EstimatedRTT + 4*DevRTT`



estimated RTT

“safety margin”

Practical Reliability Questions

- What does connection establishment look like?
- How do we choose sequence numbers?
- How should we choose timeout values?
- How do the sender and receiver keep track of outstanding pipelined segments?
- How many segments should be pipelined?

Sliding window

- How many bytes to pipeline?
- How big do we make that window?
  - Too small: link is under-utilized
  - Too large: congestion, packets dropped
  - Other concerns: fairness

Discussion: Why do we need rate control ?

A. to help the global network (core routers, and other end-hosts)
B. to help the receiver
C. to help the sender
D. some other reason

Shared high-level goal: don’t waste capacity by sending something that is likely to be dropped.

Rate Control

Flow Control

- Don’t send so fast that we overload the receiver.
- Rate directly negotiated between one pair of hosts (the sender and receiver).

Congestion Control

- Don’t send so fast that we overload the network.
- Rate inferred by sender in response to “congestion events.”

Shared high-level goal: don’t waste capacity by sending something that is likely to be dropped.

Flow Control

- Don’t send so fast that we overload the receiver.
- Rate directly negotiated between one pair of hosts (the sender and receiver).

Flow Control

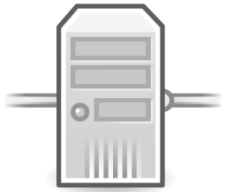
Problem: Sender can send at a high rate. Network can deliver at a high rate. The receiver is drowning in data.

- Example scenarios:





Flow Control



Fast server



Low-power device



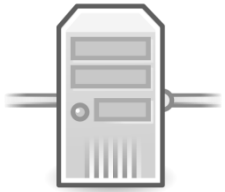
Finite socket buffer space at the receiver.

Flow Control



Finite socket buffer space at the receiver.

Flow Control



Fast server



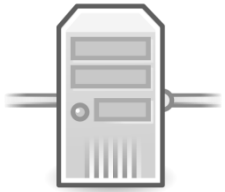
App calls recv()

Low-power device



Finite socket buffer
space at the receiver.

Flow Control



Fast server



App calls recv()



Low-power device



Finite socket buffer
space at the receiver.

Flow Control

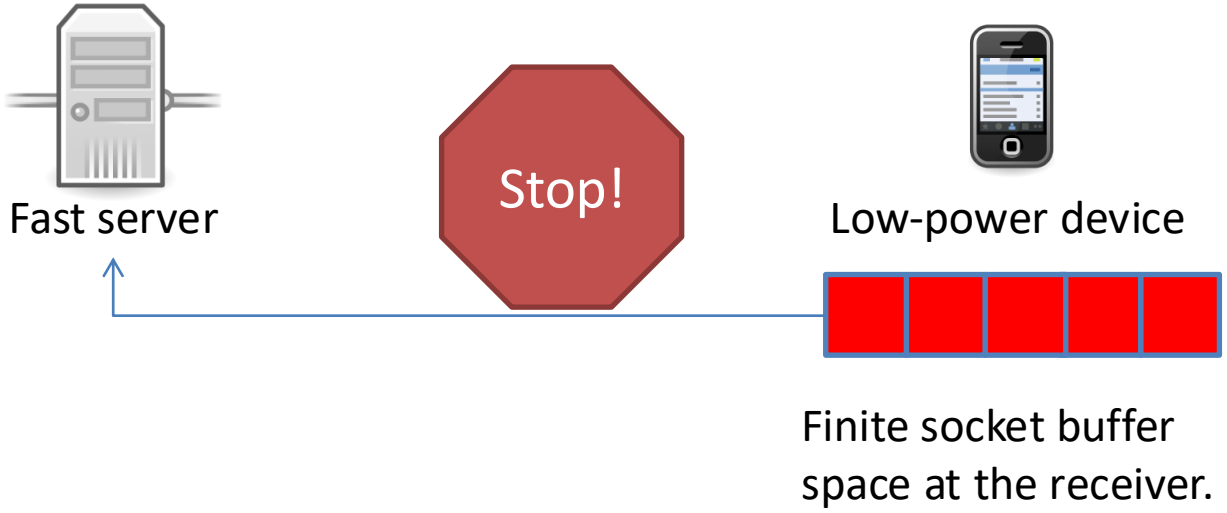


Finite socket buffer space at the receiver.

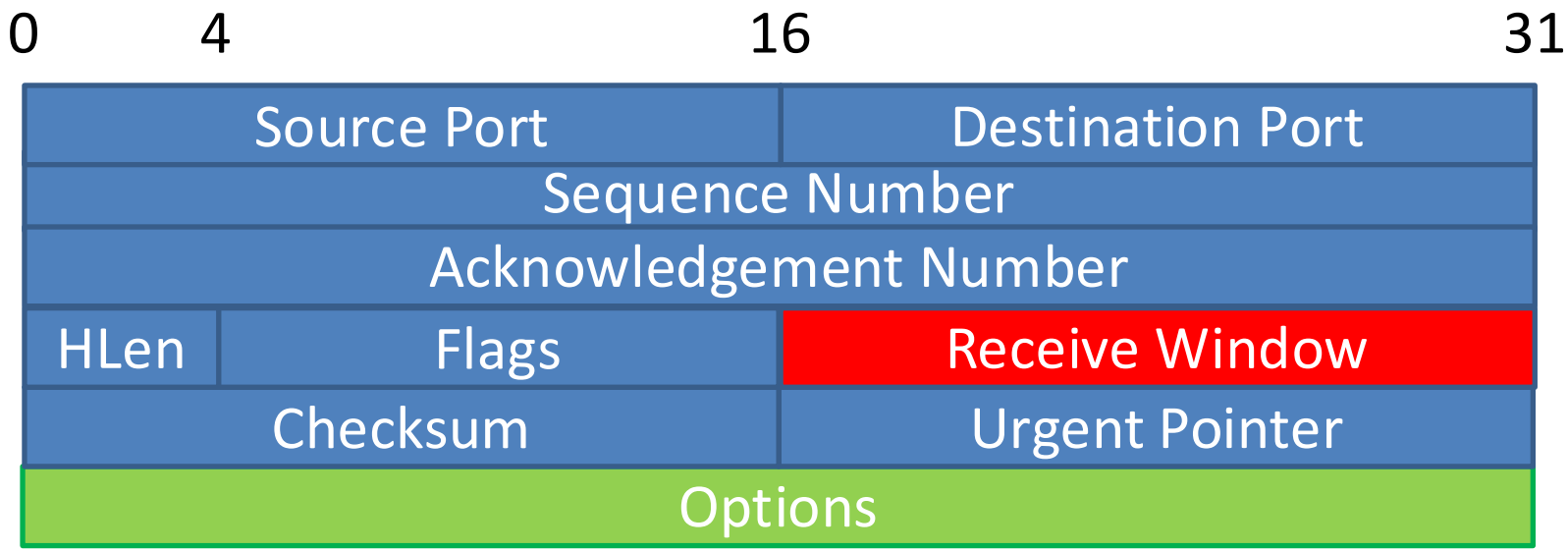
Flow Control



Flow Control

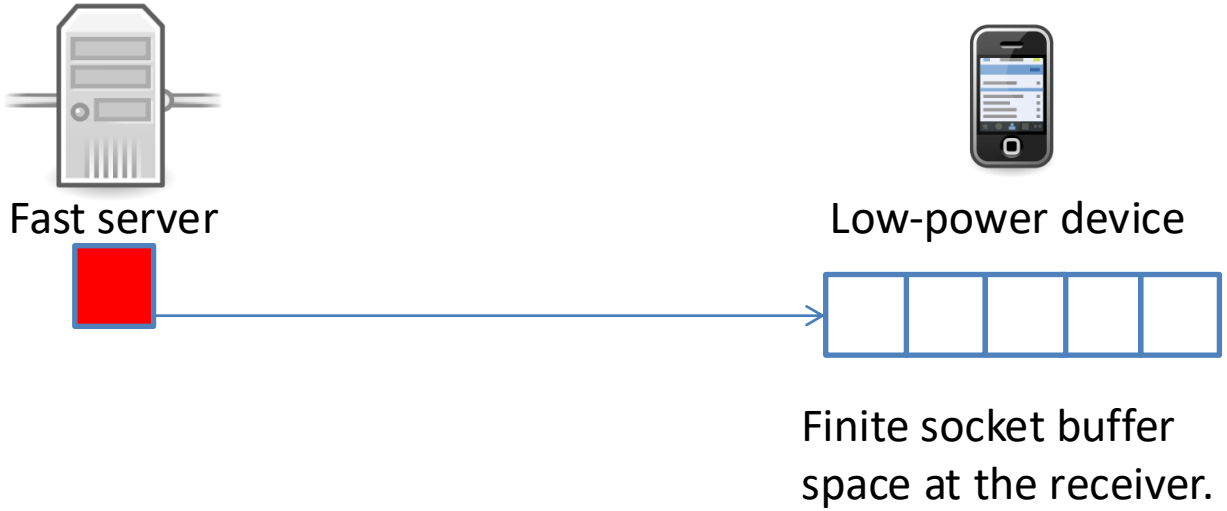


TCP Receive Window (rwnd)



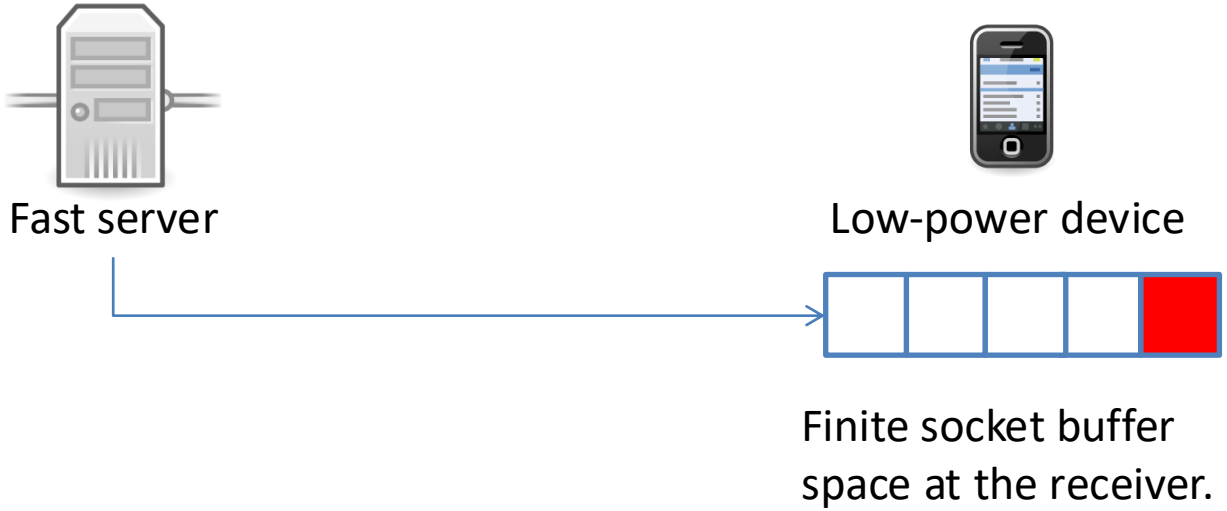
Flow Control

- Sender never sends more than rwnd.



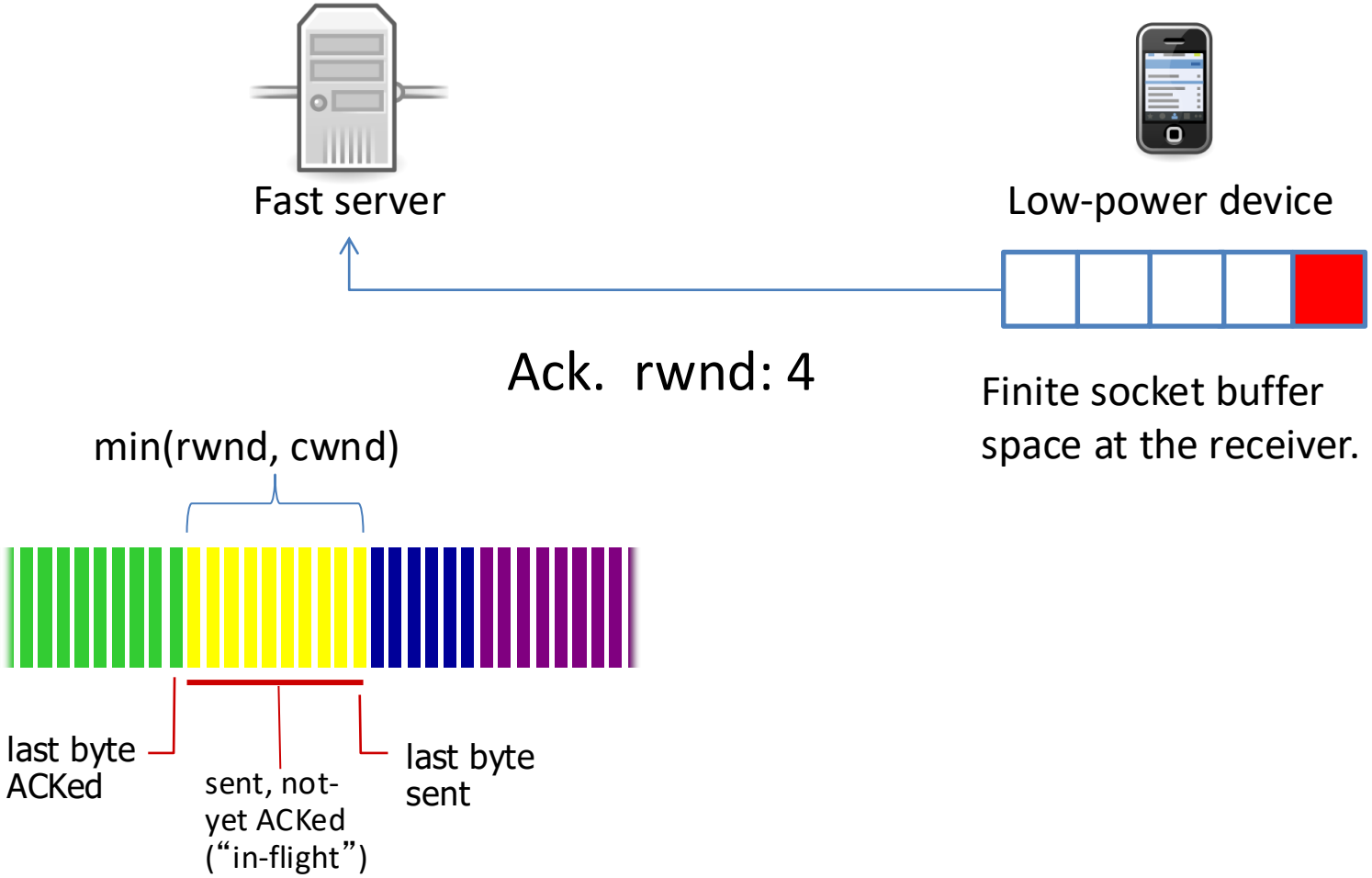
Flow Control

- Sender never sends more than rwnd.



Flow Control

- Sender never sends more than rwnd.

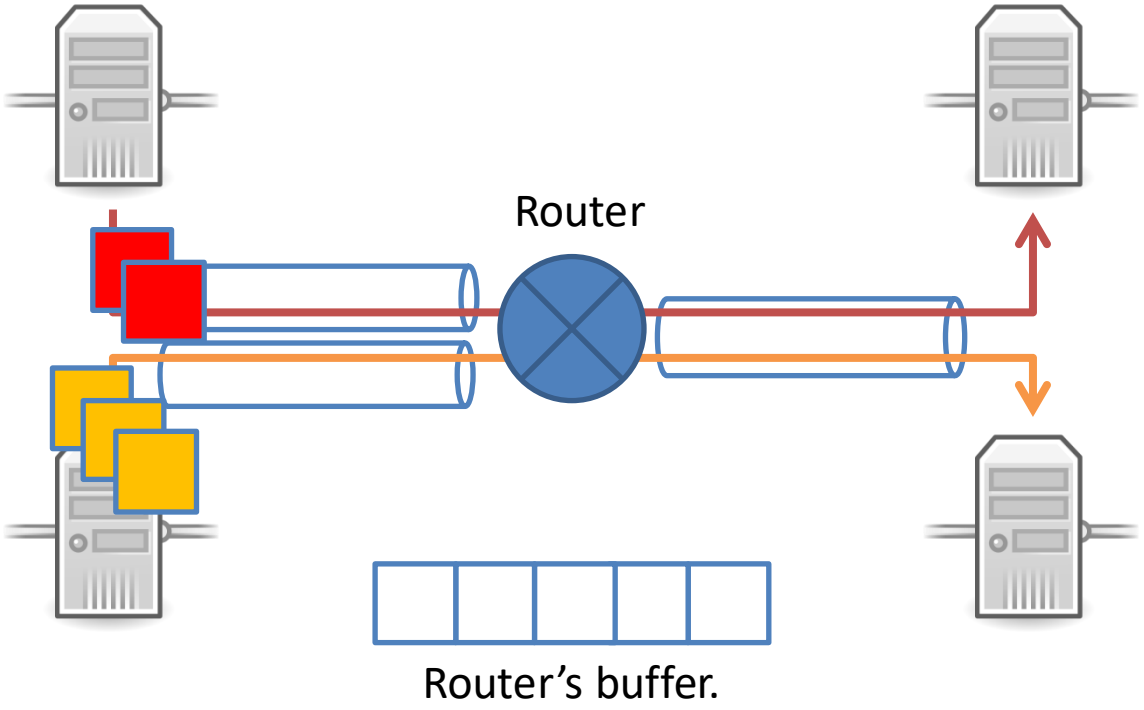


Congestion

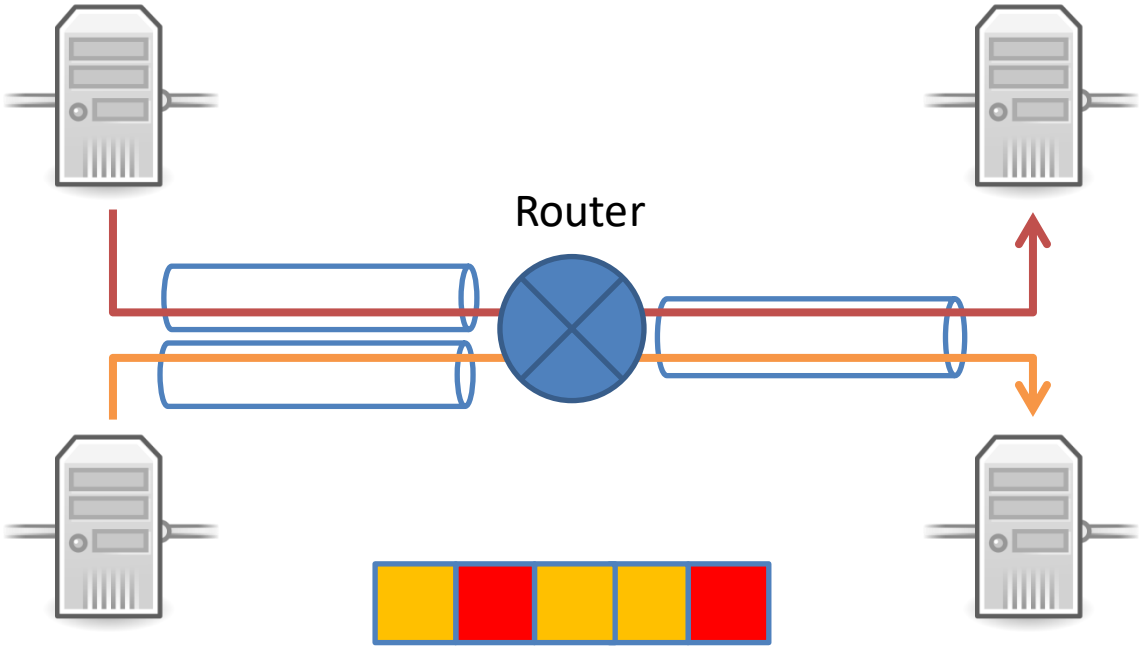
- Flow control is (relatively) easy. The receiver knows how much space it has.

- What about the network devices?

Congestion

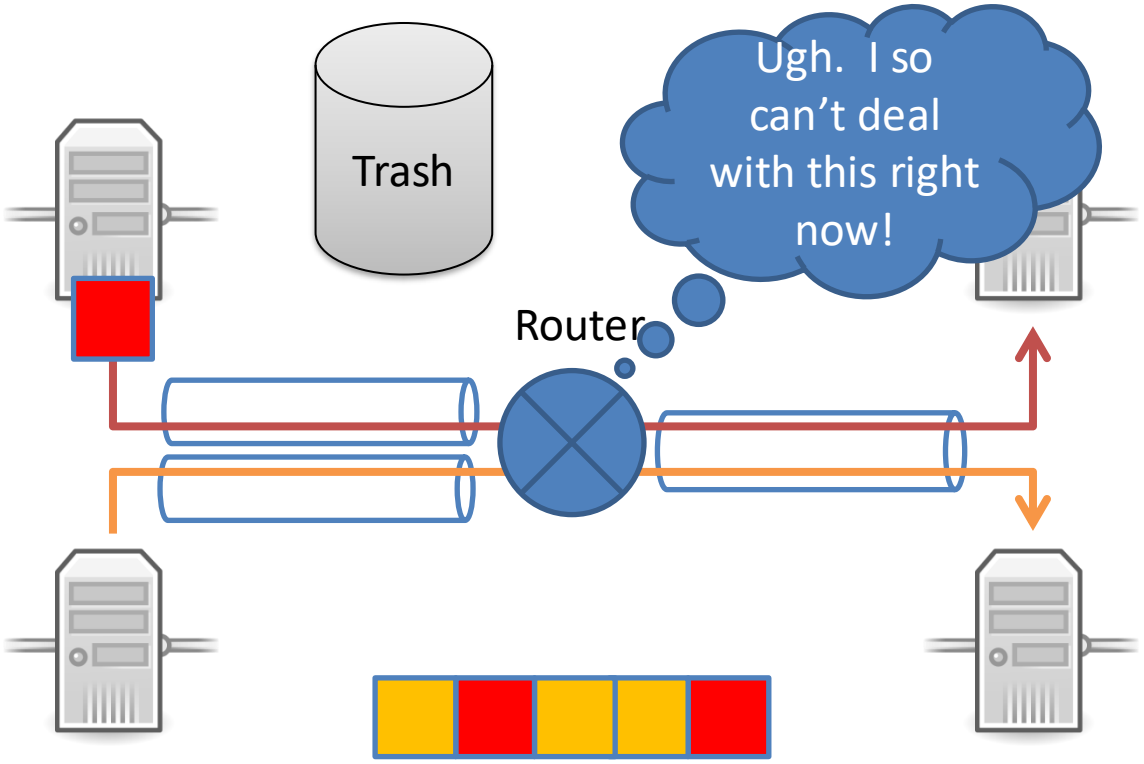


Congestion



Incoming rate is faster than outgoing link can support.

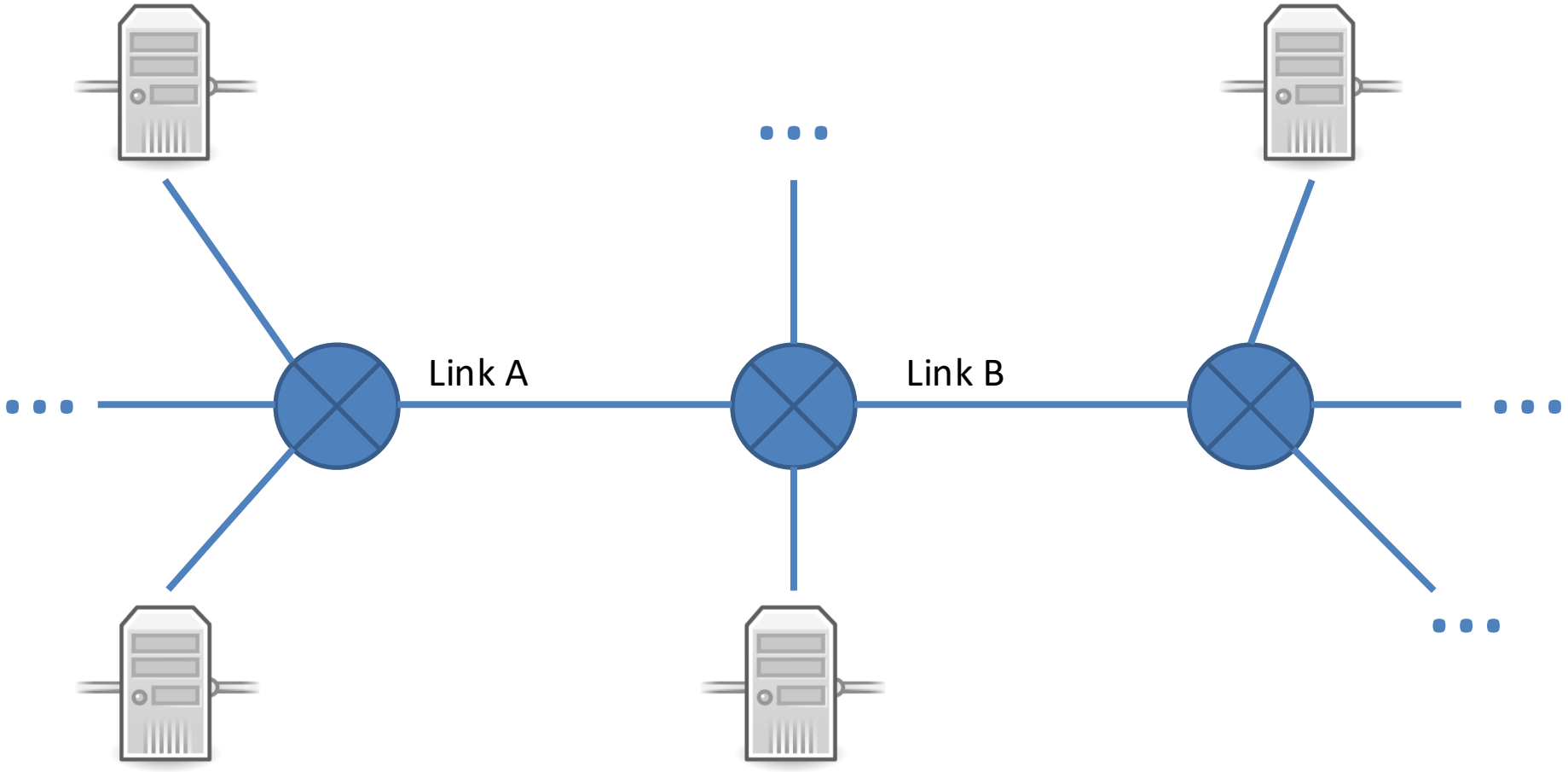
Congestion



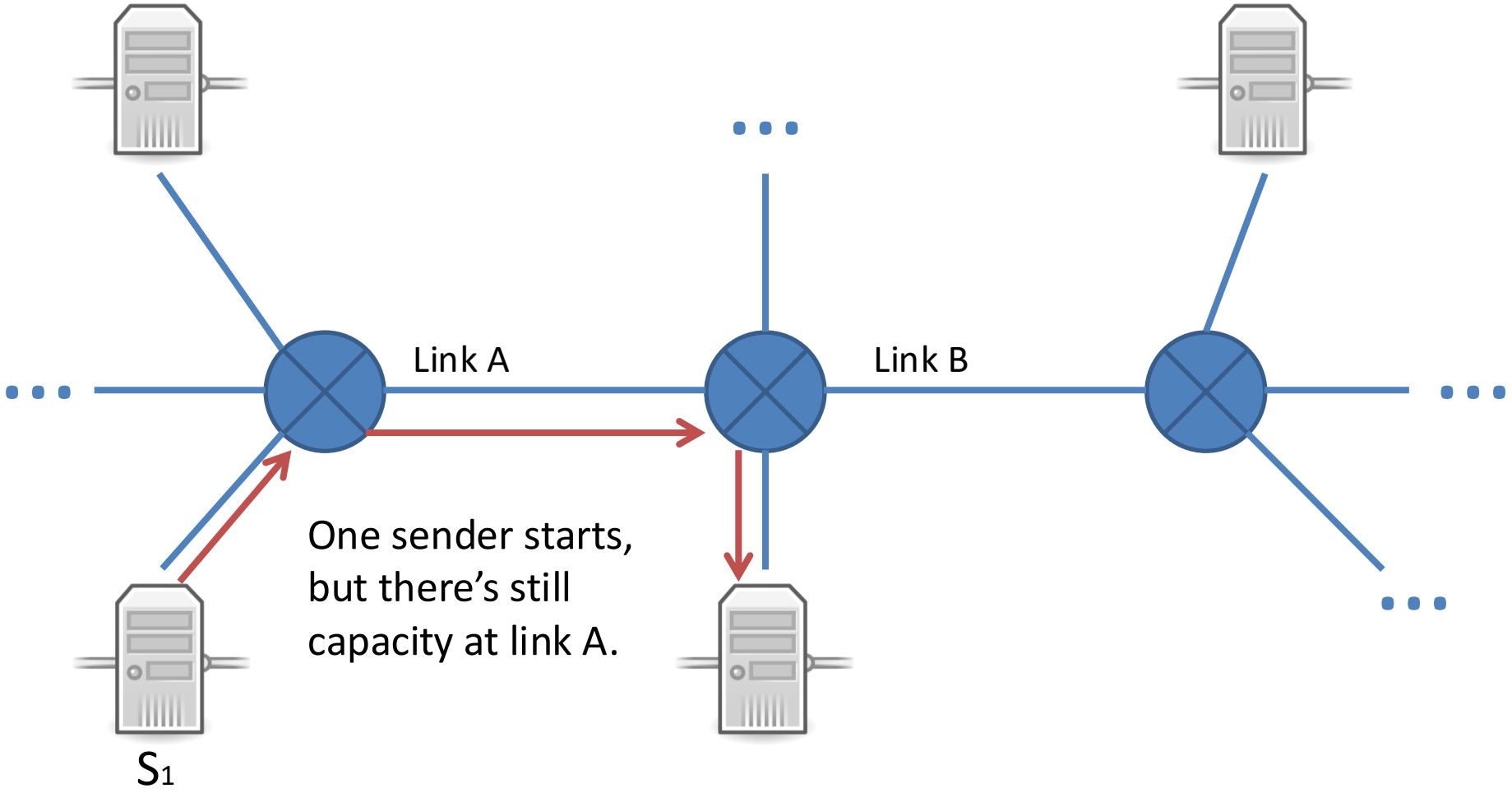
Router's buffer.

Incoming rate is faster than outgoing link can support.

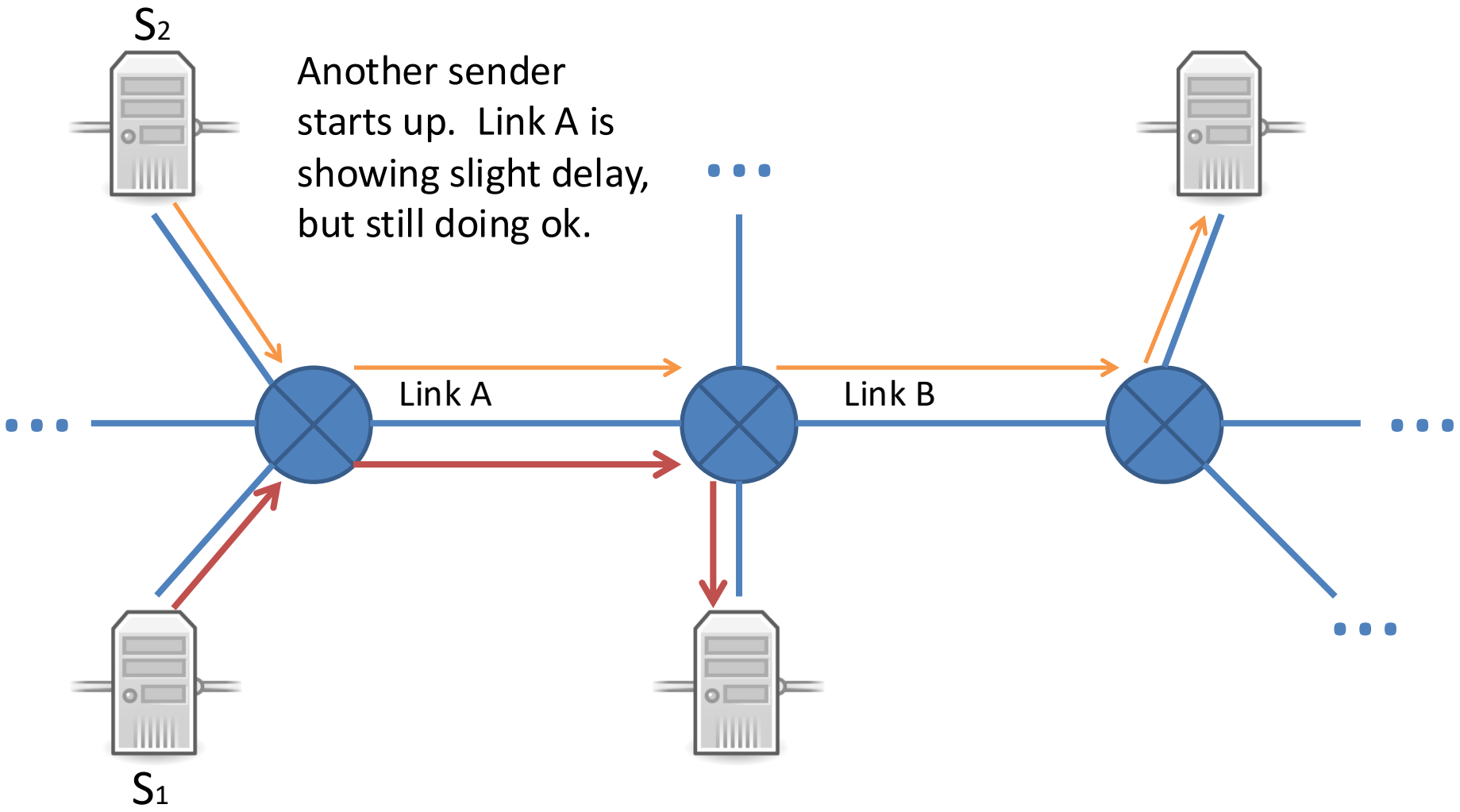
Congestion Collapse



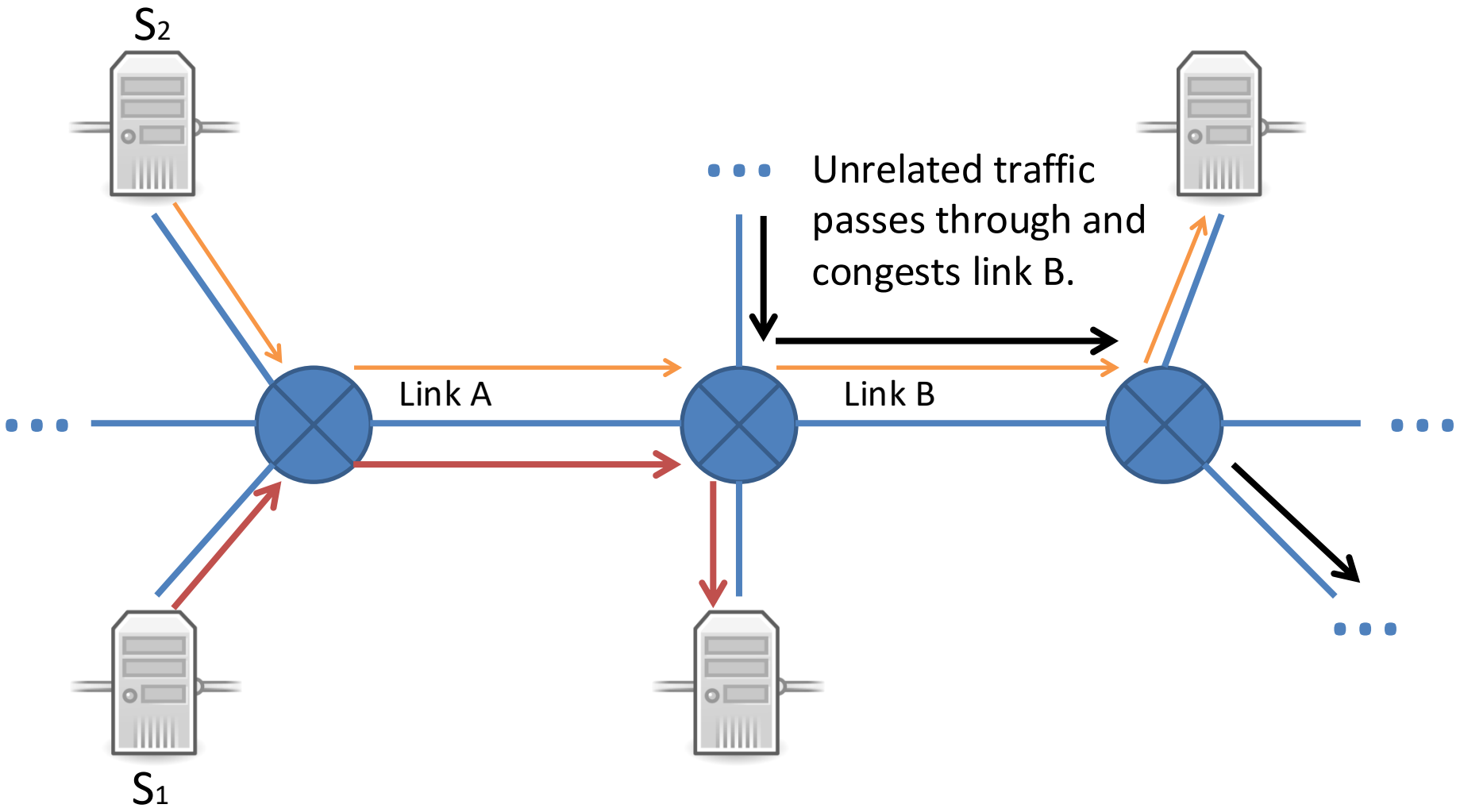
Congestion Collapse



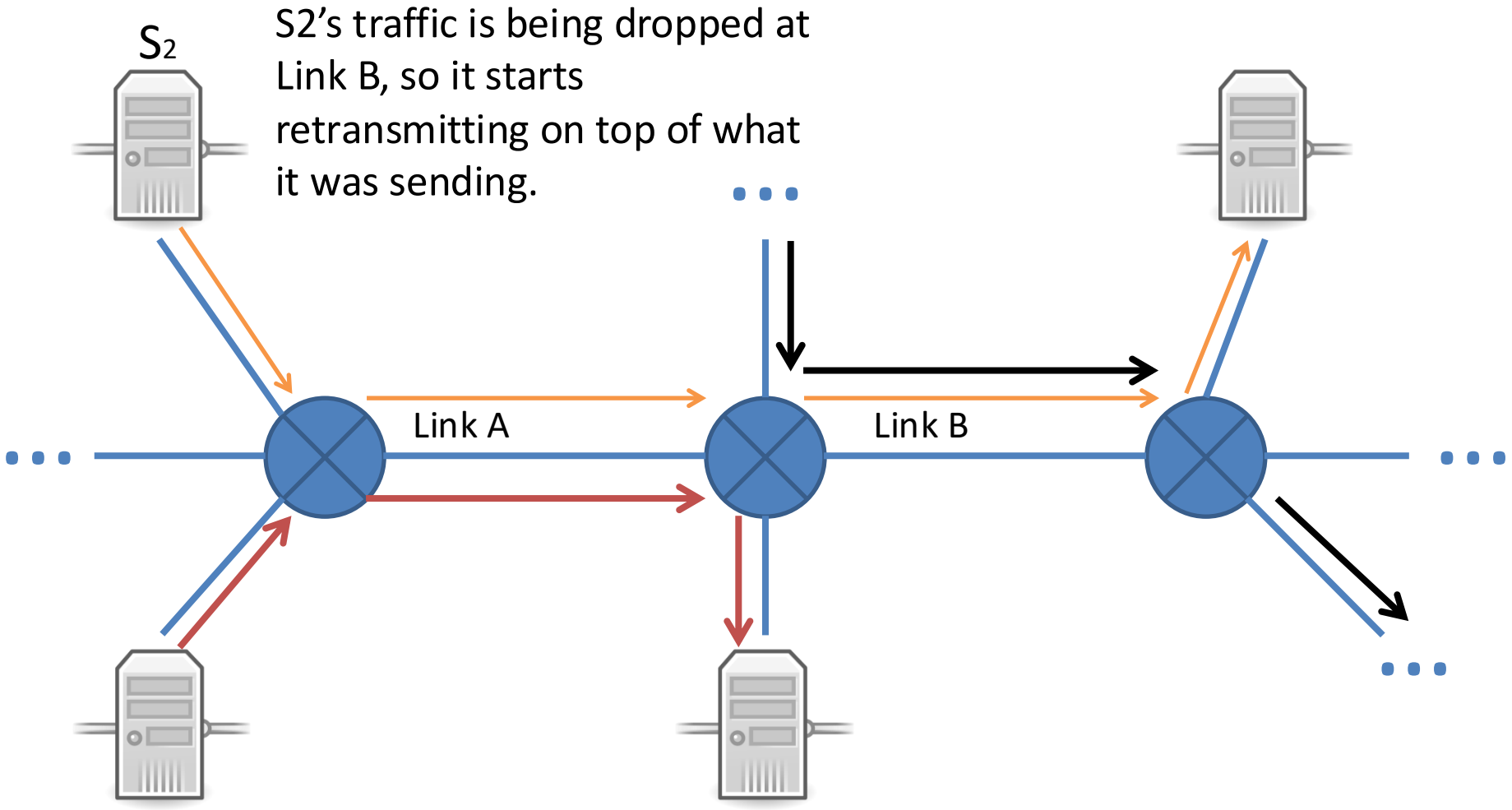
Congestion Collapse



Congestion Collapse

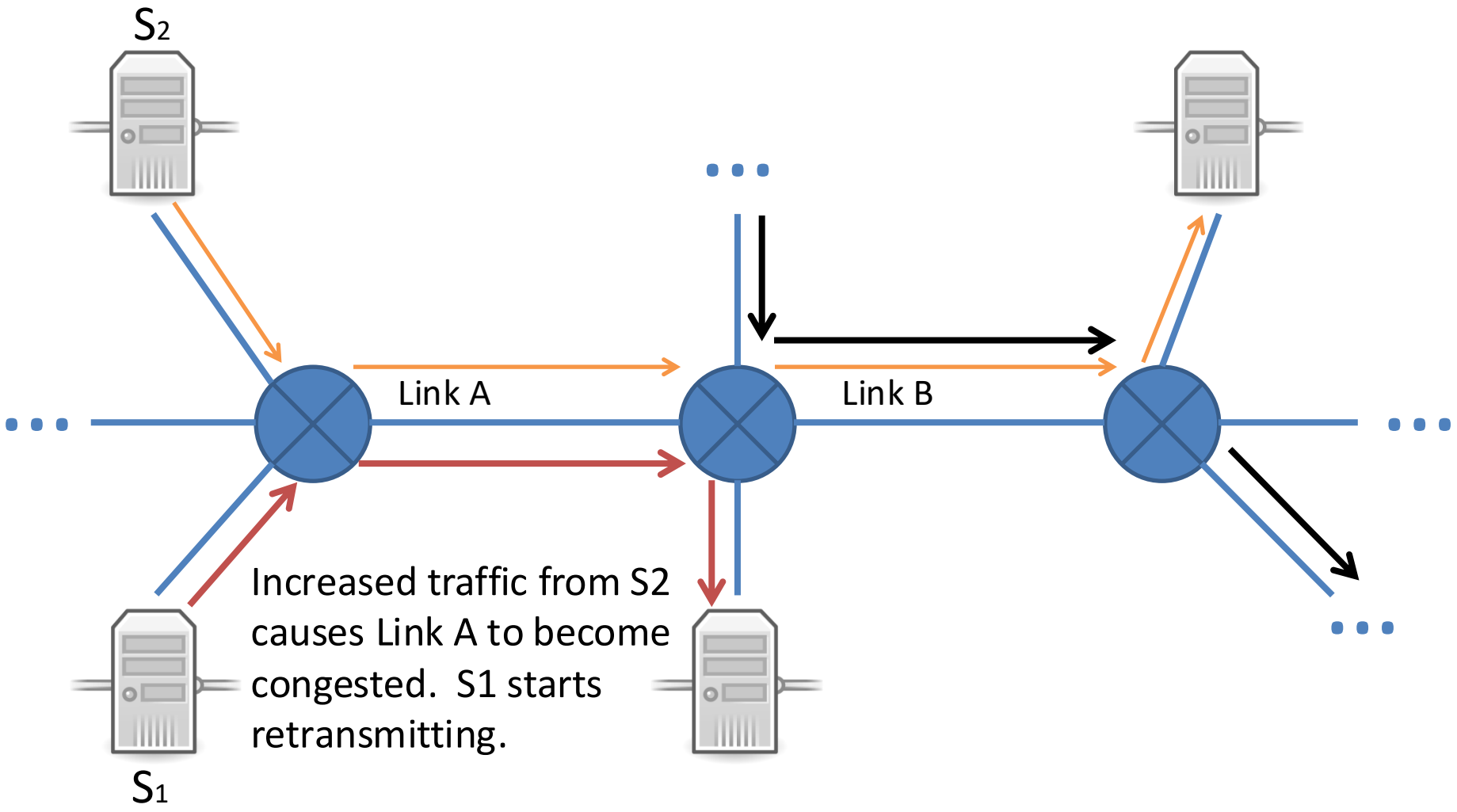


Congestion Collapse

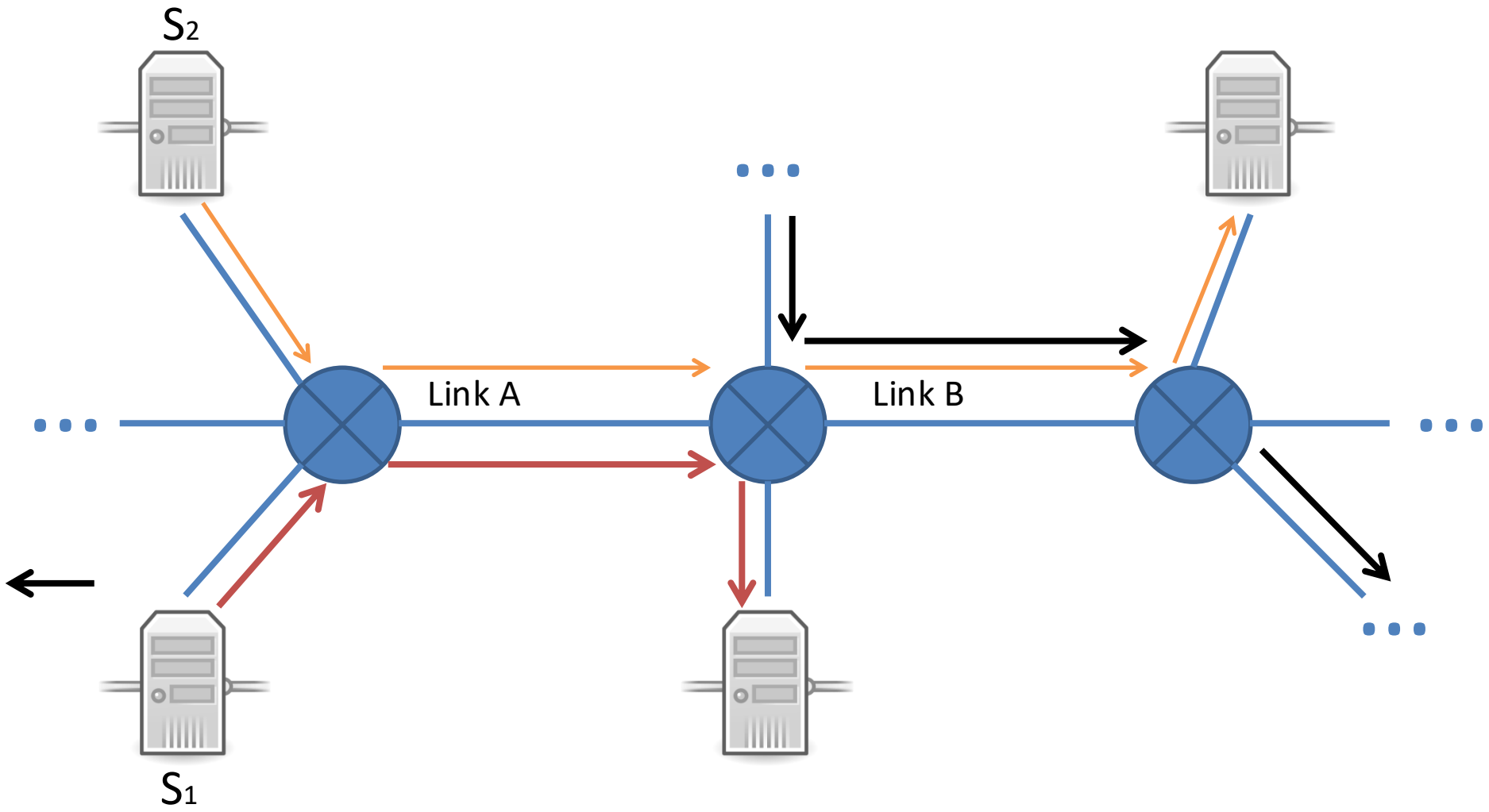


This is very bad. S2 is now sending lots of traffic over link A that has no hope of crossing link B.

Congestion Collapse



Congestion Collapse



What problems do we have without congestion control?

A. Affects Latency
B. Affects loss rate
C. Affects network capacity
D. Affects application layer performance
E. More than one of the above

Discussion: What problems do we have without congestion control?

A. Affects Latency
B. Affects loss rate
C. Affects network capacity
D. Affects application layer performance
E. More than one of the above

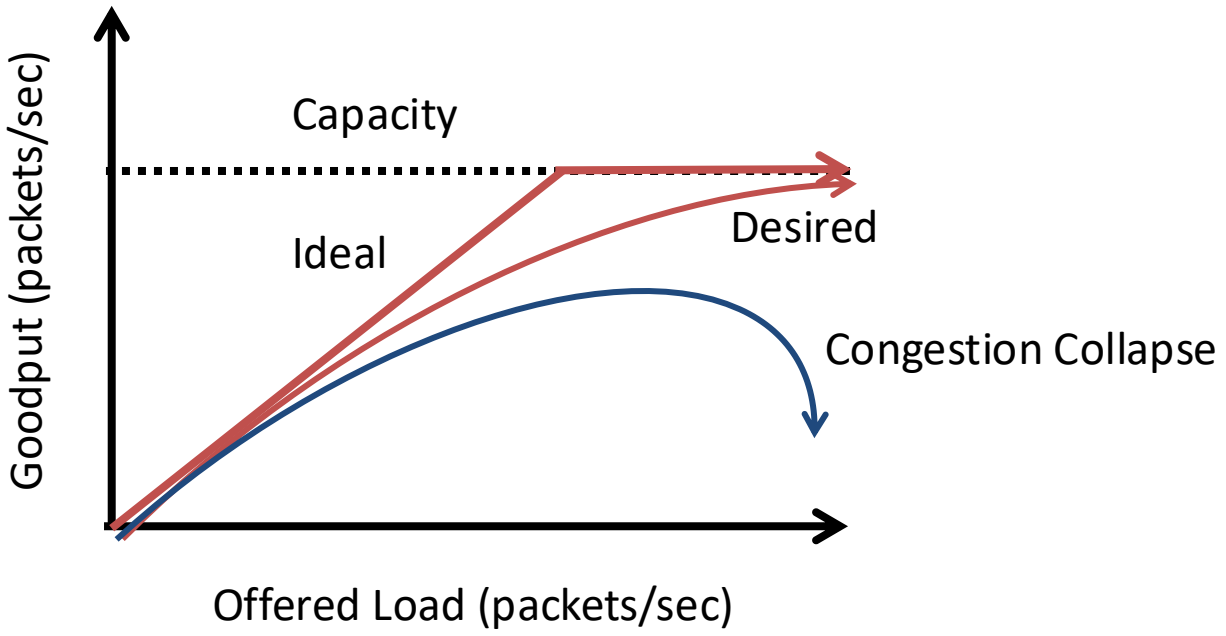


Without Congestion Control

- Congestion…
  - Increases delivery latency
  - Increases loss rate
  - Increases retransmissions, many unnecessary
  - Wastes capacity on traffic that is never delivered
  - Increases congestion, cycle continues…

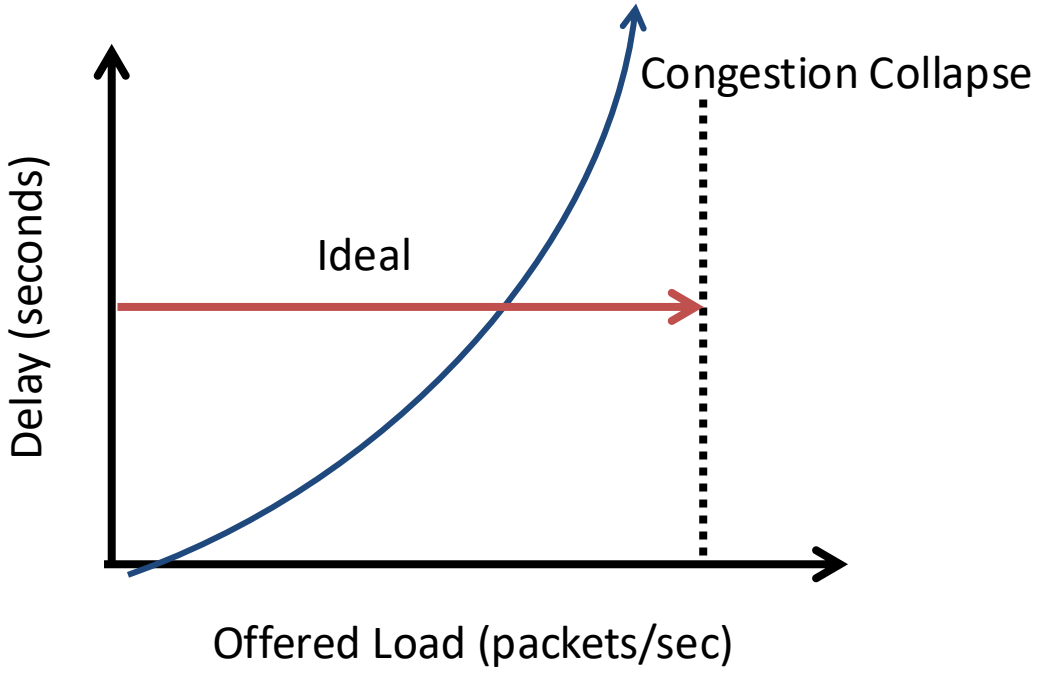
Effects of congestion: what happens to performance when we increase the load?

Goodput: Packets through the network that are not retransmissions
Offered Load: All packets in the network



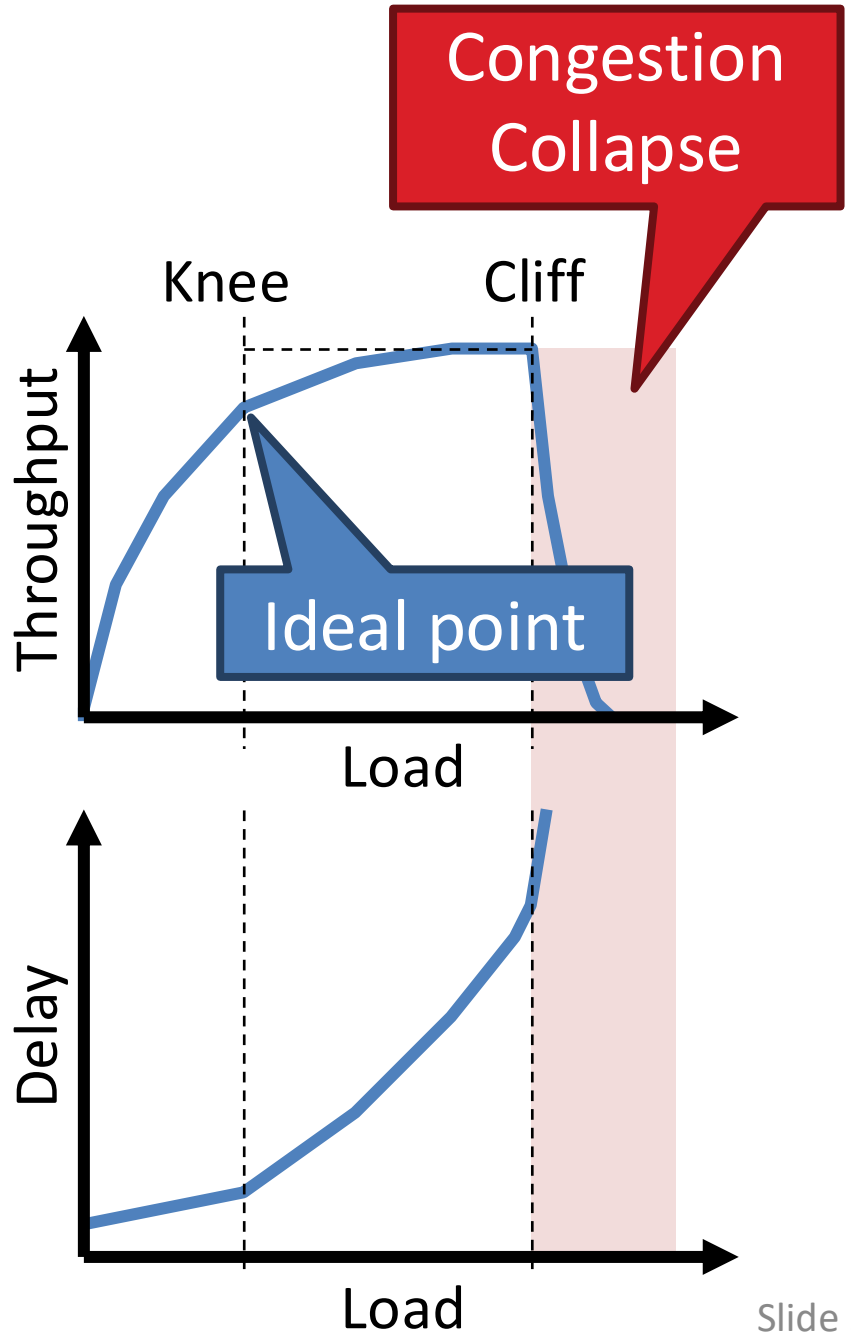
Effects of congestion: what happens to performance when we increase the load?

Goodput: Packets through the network that are not retransmissions
Offered Load: All packets in the network

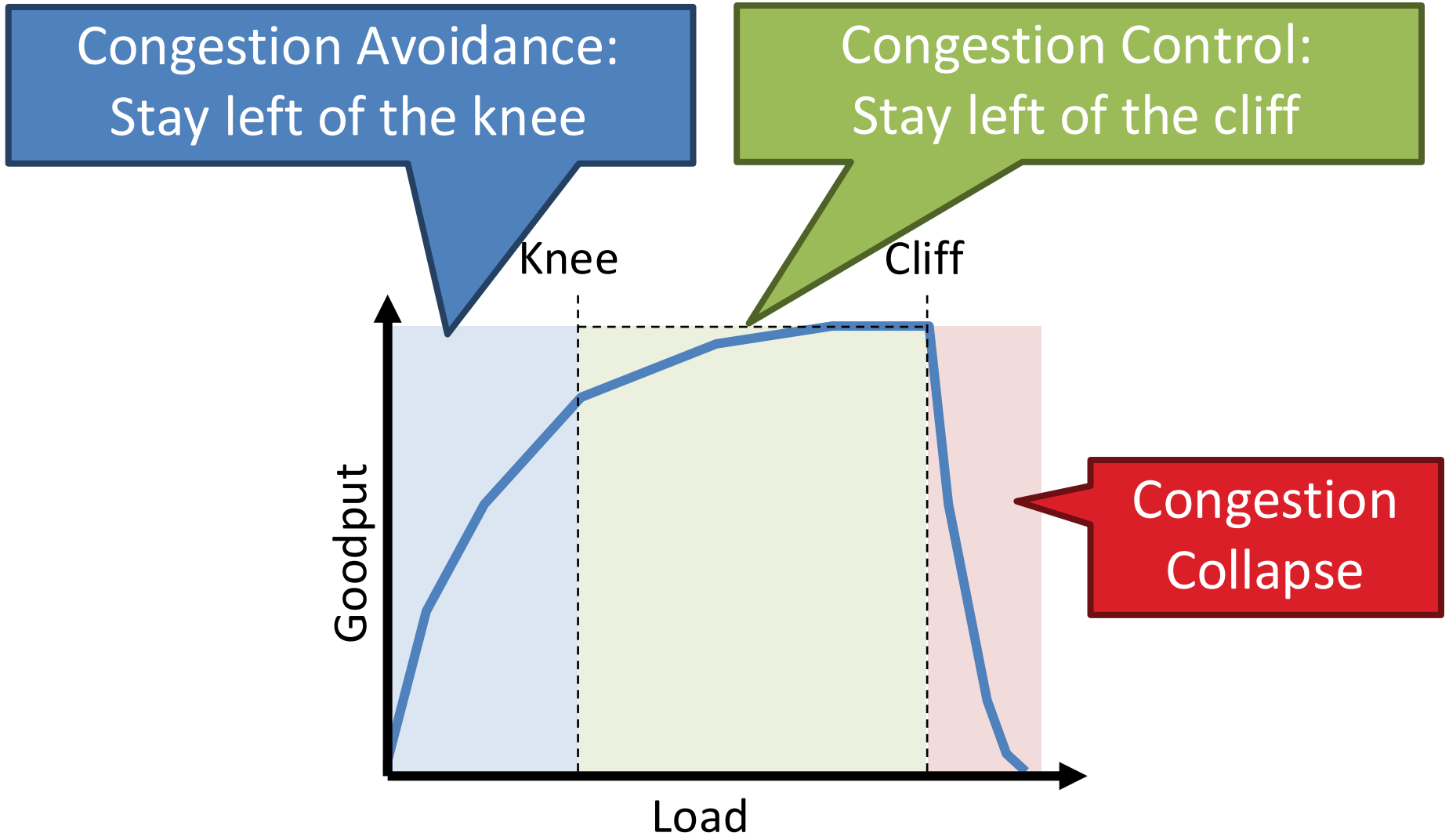


The Danger of Increasing Load

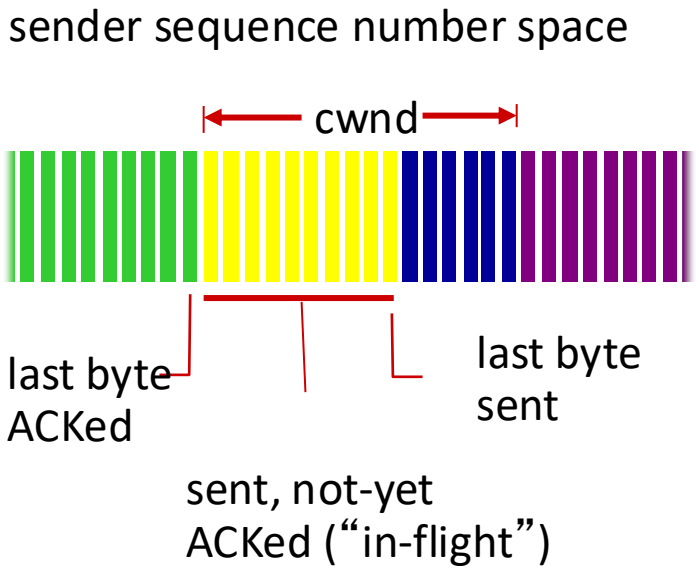
- Knee – point after which
  - Throughput increases very slow
  - Delay increases fast
- Cliff – point after which
  - Throughput → 0
  - Delay → ∞



Cong. Control vs. Cong. Avoidance



TCP Congestion Control: details



TCP sending rate:

- send cwnd bytes, wait RTT for ACKS, then send more bytes

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

- sender limits transmission:

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

- cwnd is dynamic, function of perceived network congestion

What is a “congestion event” from the perspective of a sender in TCP?

A. A segment loss

B. Receiving duplicate acknowledgement(s)

C. A retransmission timeout firing

D. Some subset of the above

E. All of the above

Discussion: What is a “congestion event” from the perspective of a sender in TCP?

A. A segment loss

B. Receiving duplicate acknowledgement(s)

C. A retransmission timeout firing

D. Some subset of the above



E. All of the above

What is a “congestion event” from the perspective of a sender in TCP?

A. A segment loss

B. Receiving duplicate acknowledgement(s)

C. A retransmission timeout firing

D. Some subset of the above

E. All of the above

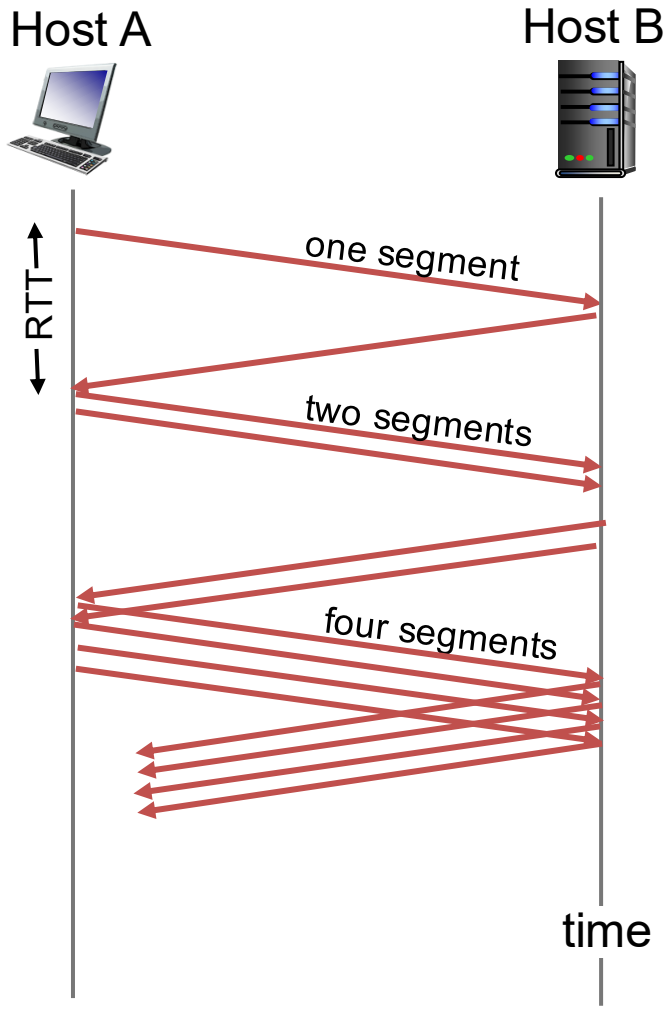
What we care about is segment loss, and both B and C give us a way to know that a segment loss has occurred.

TCP Congestion Control Phases

- Slow start
  - Sender has no idea of network's congestion
  - Start conservatively, increase rate quickly
- Congestion avoidance
  - Increase rate slowly
  - Back off when congestion occurs
    - How much depends on TCP version

TCP Slow Start

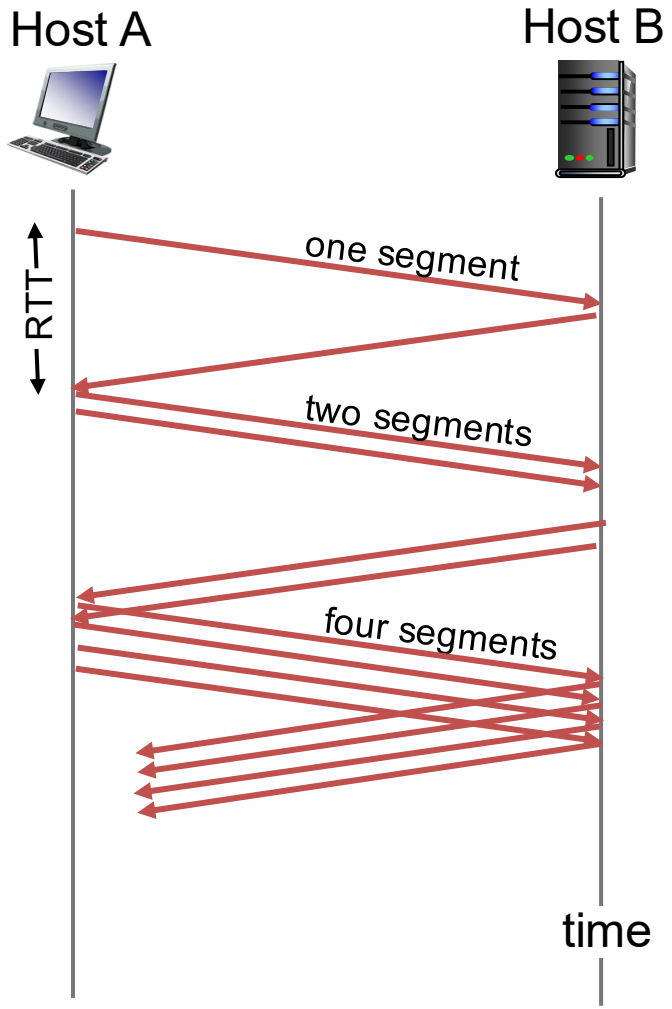
- When connection begins, increase rate exponentially until first loss event:
  - initially **`cwnd`** = 1 MSS
  - double **`cwnd`** every RTT
  - done by incrementing **`cwnd`** for every ACK received
- Summary: initial rate is slow but ramps up exponentially fast



- When do we stop?

TCP Slow Start

- When do we stop?
- Initially
  - On a congestion event
- Later
  - On a congestion event
  - When we cross a previously-determined threshold



Reference slides

Example RTT Estimation

- Suppose EstimateRTT = 64, Dev = 8
- Latest sample: 120

New estimate = 7/8 * 64 + 1/8 * 120 = 56 + 15= 71
New dev = 3/4 * 8 + 1/4 * | 120 - 71 | = 6 + 12 = 18

- Another sample: 400

New estimate = 7/8 * 71 + 1/8 * 400 = 62 + 50 = 112
New dev = 3/4 * 18 + 1/4 * | 400 - 112 | = 13 + 72 = 85

Example RTT Estimation (Smoothing)

