Congestion Control

Lecture 15

http://www.cs.rutgers.edu/~sn624/352-S22

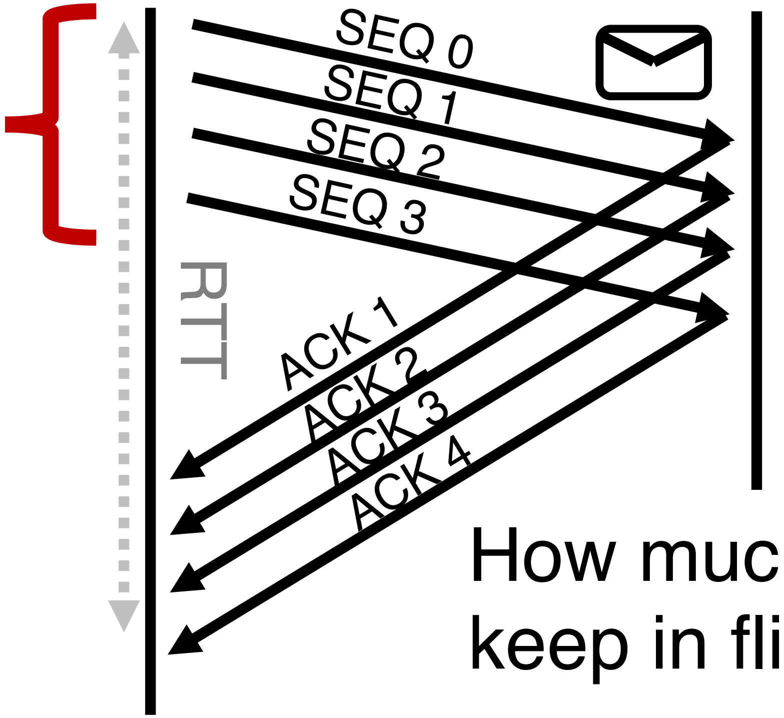
Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

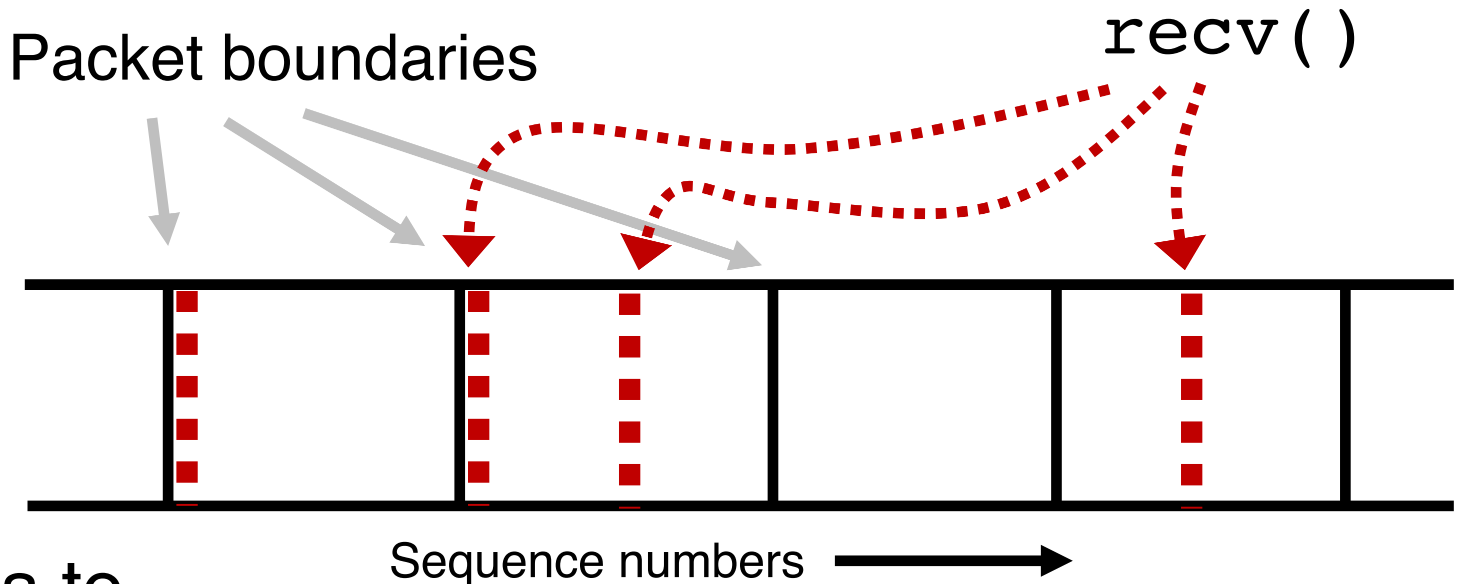
Quick recap of concepts

TCP:
Reliability
Ordering

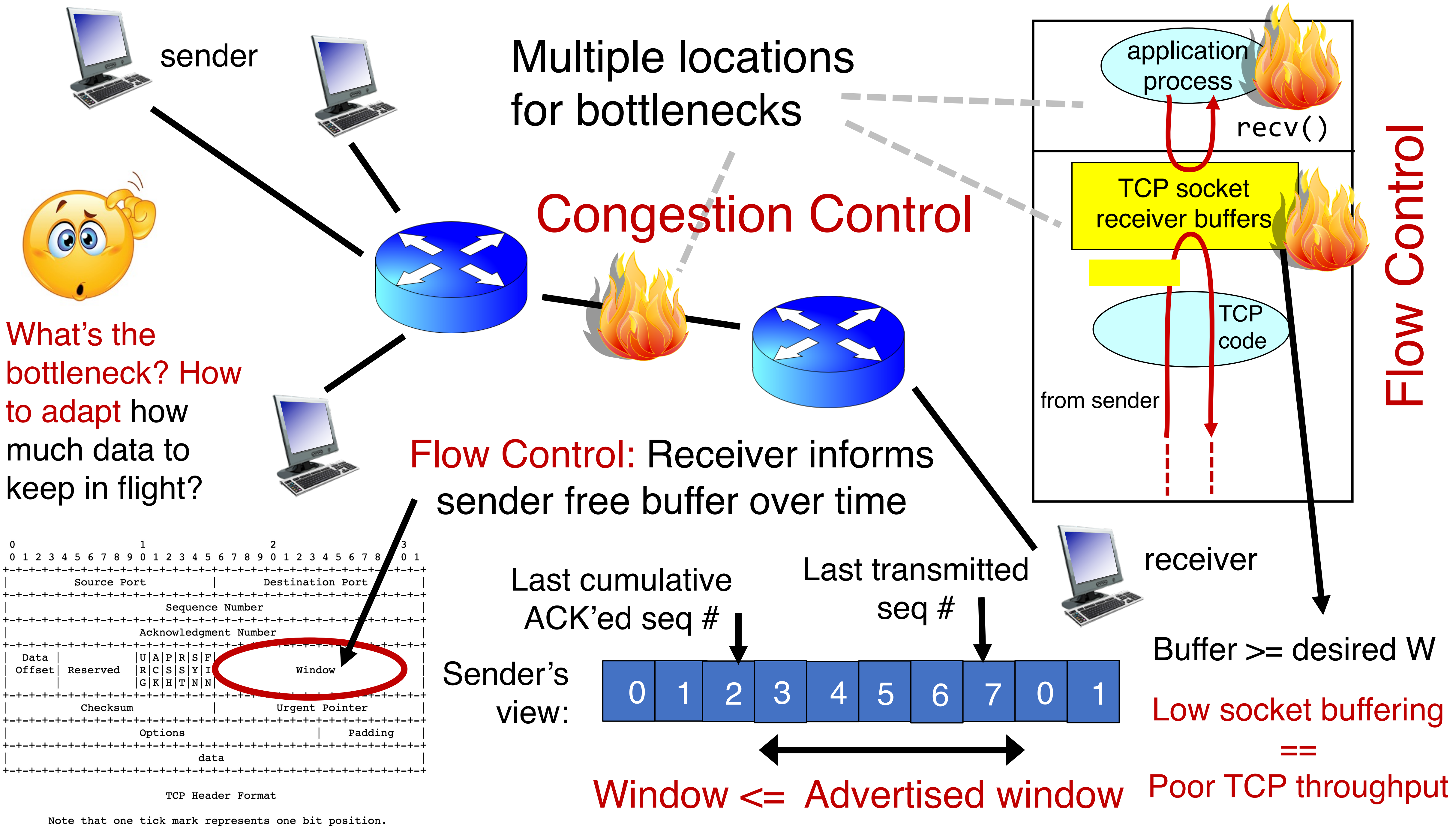
Reordering degrades connection throughput. Apps can't `recv()`.
Packets may even be dropped due to insufficient buffering.



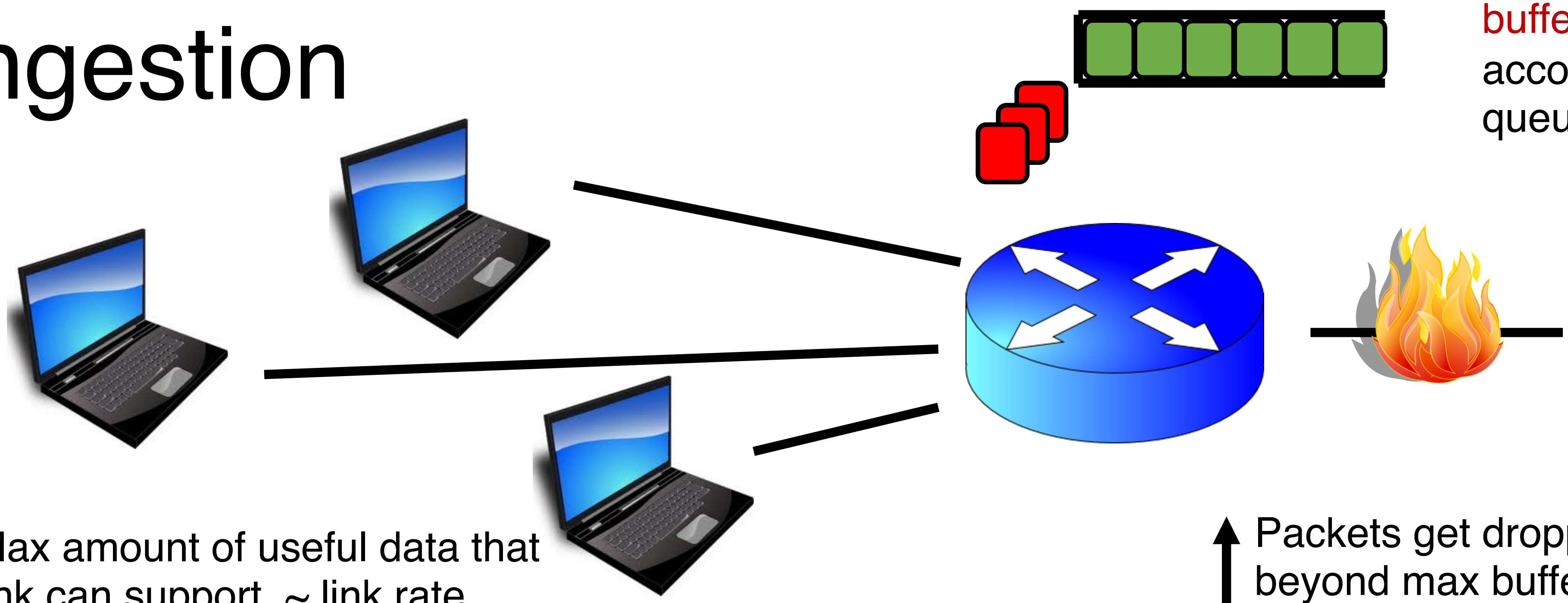
How much data to keep in flight?

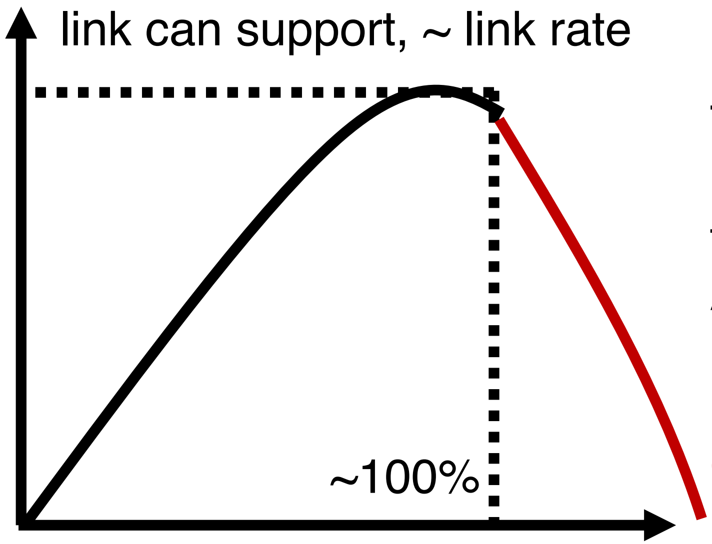


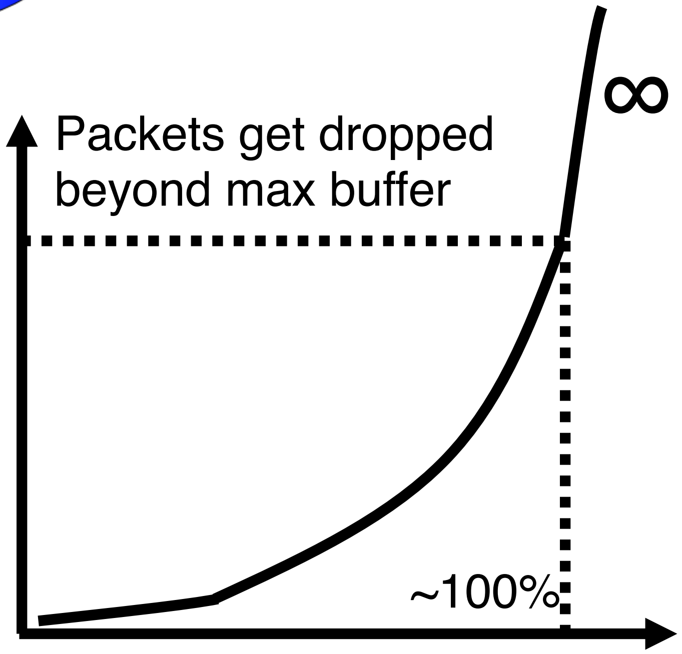
Stream-Oriented Transport


Multiple locations
for bottlenecks
sender
Congestion Control
What's the bottleneck? How to adapt how much data to keep in flight?
Flow Control: Receiver informs sender free buffer over time
application process
recv()
TCP socket receiver buffers
TCP code
from sender
Flow Control
receiver
Buffer >= desired W
Low socket buffering
==
Poor TCP throughput
Last cumulative ACK'ed seq #
Last transmitted seq #
Sender's view:
0 1 2 3 4 5 6 7 0 1
Window <= Advertised window
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Reserved
URG ACK PSH RST SYN FIN
Window
Checksum
Urgent Pointer
Options
Padding
data
TCP Header Format
Note that one tick mark represents one bit position.

Congestion

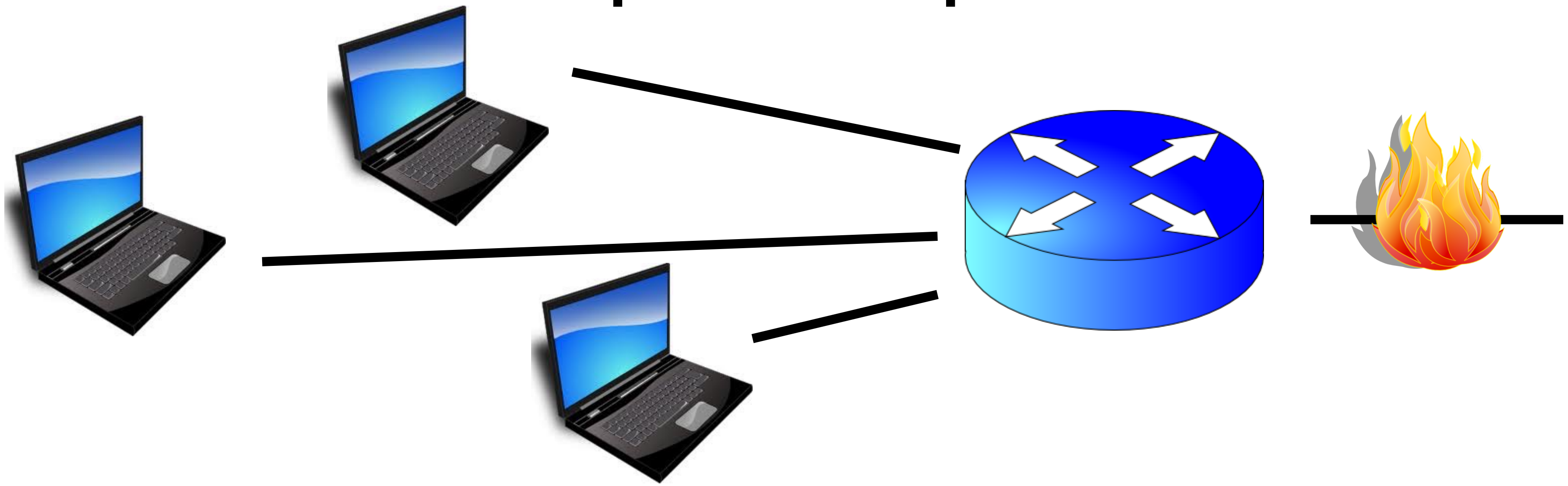






https://en.wikipedia.org/wiki/Network_congestion#Congestive_collapse

How should multiple endpoints share net?



- It is difficult to know where the bottleneck link is
- It is difficult to know how many other endpoints are using that link
- Endpoints may join and leave at any time
- Network paths may change over time, leading to different bottleneck links (with different link rates) over time

The approach that the Internet takes is to use a distributed algorithm to converge to an efficient and fair outcome.

The approach that the Internet takes is to use a distributed algorithm to converge to an efficient and fair outcome.

No one can centrally view or control all the endpoints and bottlenecks in the Internet.

Every endpoint must try to reach a globally good outcome by itself: i.e., in a distributed fashion.

This also puts a lot of trust in endpoints.

The approach that the Internet takes is to use a distributed algorithm to converge to an efficient and fair outcome.

If there is spare capacity in the bottleneck link, the endpoints should use it.

The approach that the Internet takes is to use a distributed algorithm to converge to an efficient and fair outcome.

If there are N endpoints sharing a bottleneck link, they should be able to get equitable shares of the link's capacity.

For example: 1/N'th of the link capacity.

Flow Control vs. Congestion Control

Flow Control

- Avoid overwhelming the receiving application
- Sender is managing the receiver's socket buffer

Congestion Control

- Avoid overwhelming the bottleneck network link
- Sender is managing the bottleneck link capacity and bottleneck router buffers

The approach that the Internet takes is to use a distributed algorithm to converge to an efficient and fair outcome.

How to achieve this?

Approach: sense and react

Example: taking a shower

Use a feedback loop with signals and knobs

Signals and Knobs in Congestion Control

- Signals
  - Packets being ACK'ed
  - Packets being dropped (e.g. RTO fires)
  - Packets being delayed (RTT)
  - Rate of incoming ACKs

Implicit feedback signals measured directly at sender. (There are also explicit signals that the network might provide.)

- Knobs
  - What can you change to "probe" the available bottleneck capacity?
  - Suppose receiver buffer is unbounded:
  - Increase window/sending rate: e.g., add x or multiply by a factor of x
  - Decrease window/sending rate: e.g., subtract x or reduce by a factor of x

Sense and react, sure…but how?

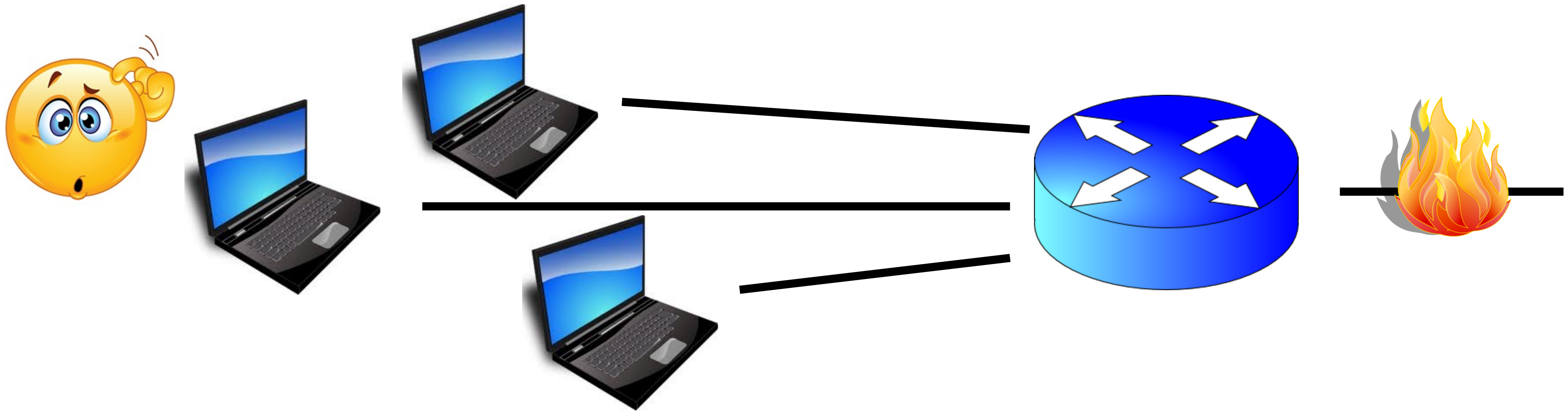
- Where do you want to be?
  - The steady state
- How do you get there?
  - Congestion control algorithms
- Sense accurately
- React proportionately

The Steady State

Efficiency of a single TCP conversation

What does efficiency look like?

- Suppose we want to achieve an efficient outcome for one TCP conversation by observing network signals from the endpoint



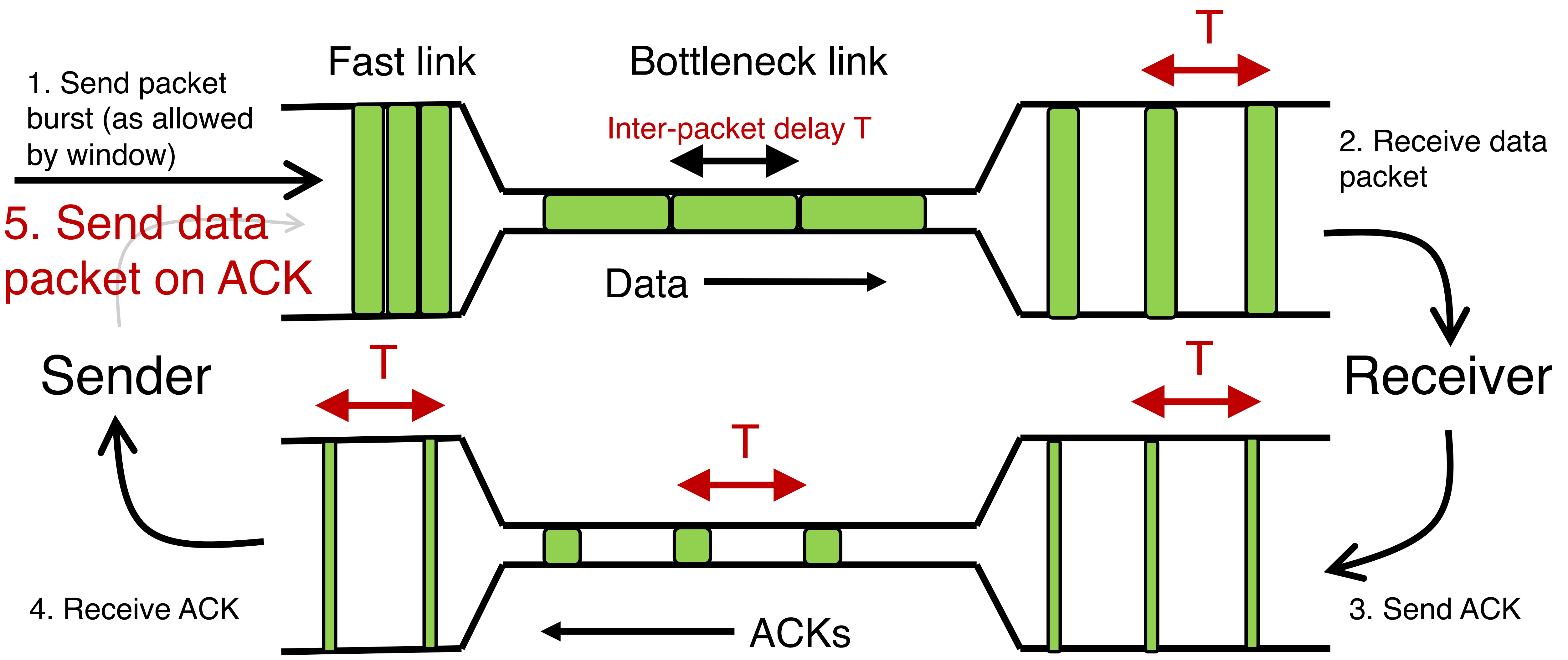
- Q: How should the endpoint behave at steady state?
- Challenge: bottleneck link is remotely located

Steady state: Ideal goal

- High sending rate: Use the full capacity of the bottleneck link
- Low delay: Minimize the overall delay of packets to get to the receiver
  - Overall delay = propagation + queueing + transmission
  - Assume propagation and transmission components fixed
- "Low delay" reduces to low queueing delay
- i.e., don't push so much data into the network that packets have to wait in queues

- Key question: When to send the next packet?

When to send the next packet?



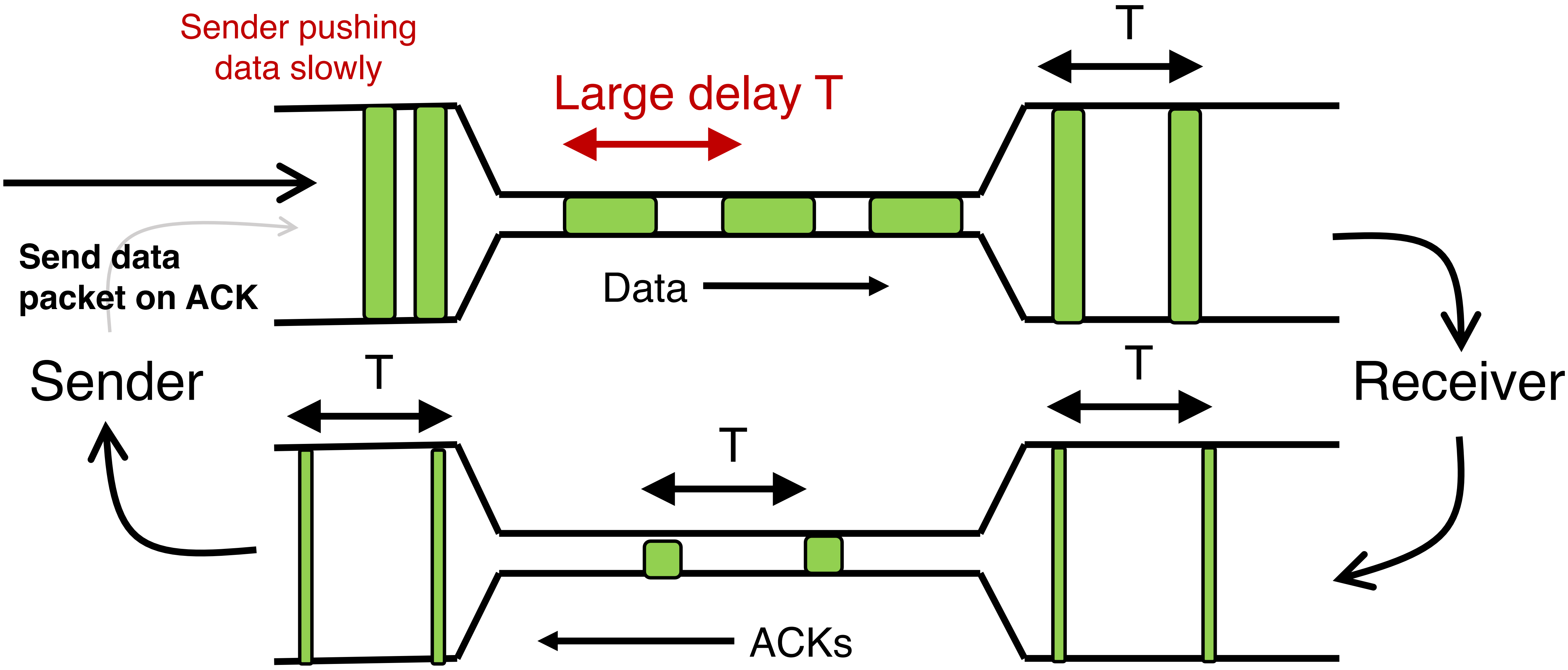
Rationale

- When the sender receives an ACK, that's a signal that the previous packet has left the bottleneck link (and the rest of the network)
- Hence, it must be safe to send another packet without congesting the bottleneck link
- Such transmissions are said to follow packet conservation
- ACK clocking: "Clock" of ACKs governs packet transmissions

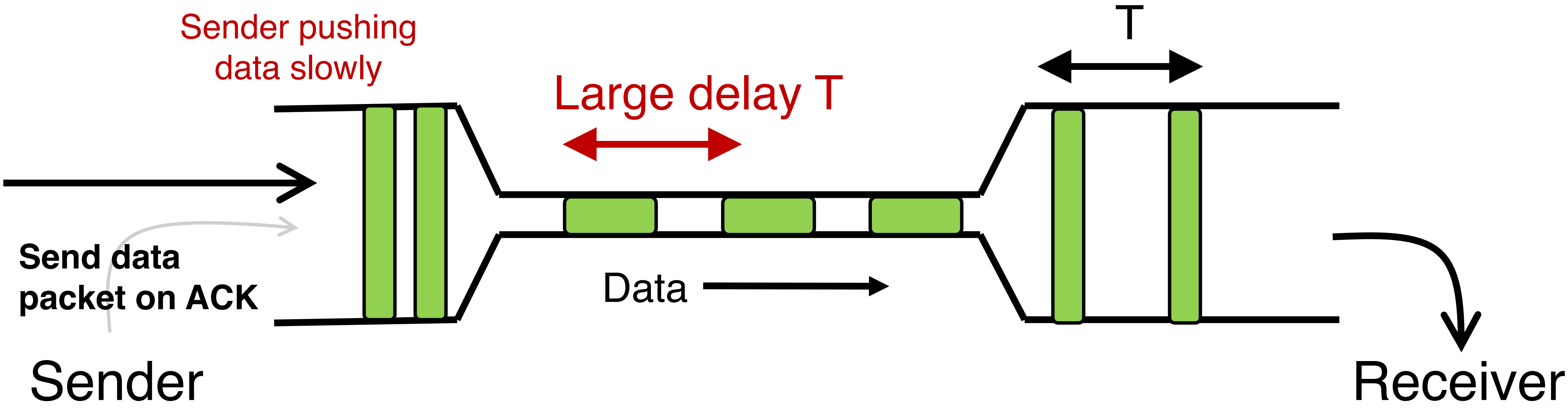
ACK clocking: analogy

- How to avoid crowding a grocery store?
- Strategy: Send the next waiting customer exactly when a customer exits the store
- However, this strategy alone can lead to inefficient use of resources…

ACK clocking alone can be inefficient



ACK clocking alone can be inefficient



The sending rate should be high enough to keep the “pipe” full

Analogy: a grocery store with only 1 customer in entire store

If the store isn’t “full”, you’re using store space inefficiently

Steady State of Congestion Control

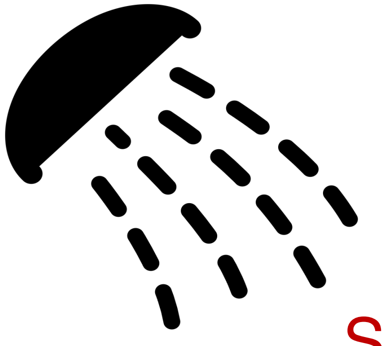
- Send at the highest rate possible (to keep the pipe full)
- while being ACK-clocked (to avoid congesting the pipe)

- So, how to get to steady state?

Getting to Steady State

The Feedback Loop

An example of a feedback loop



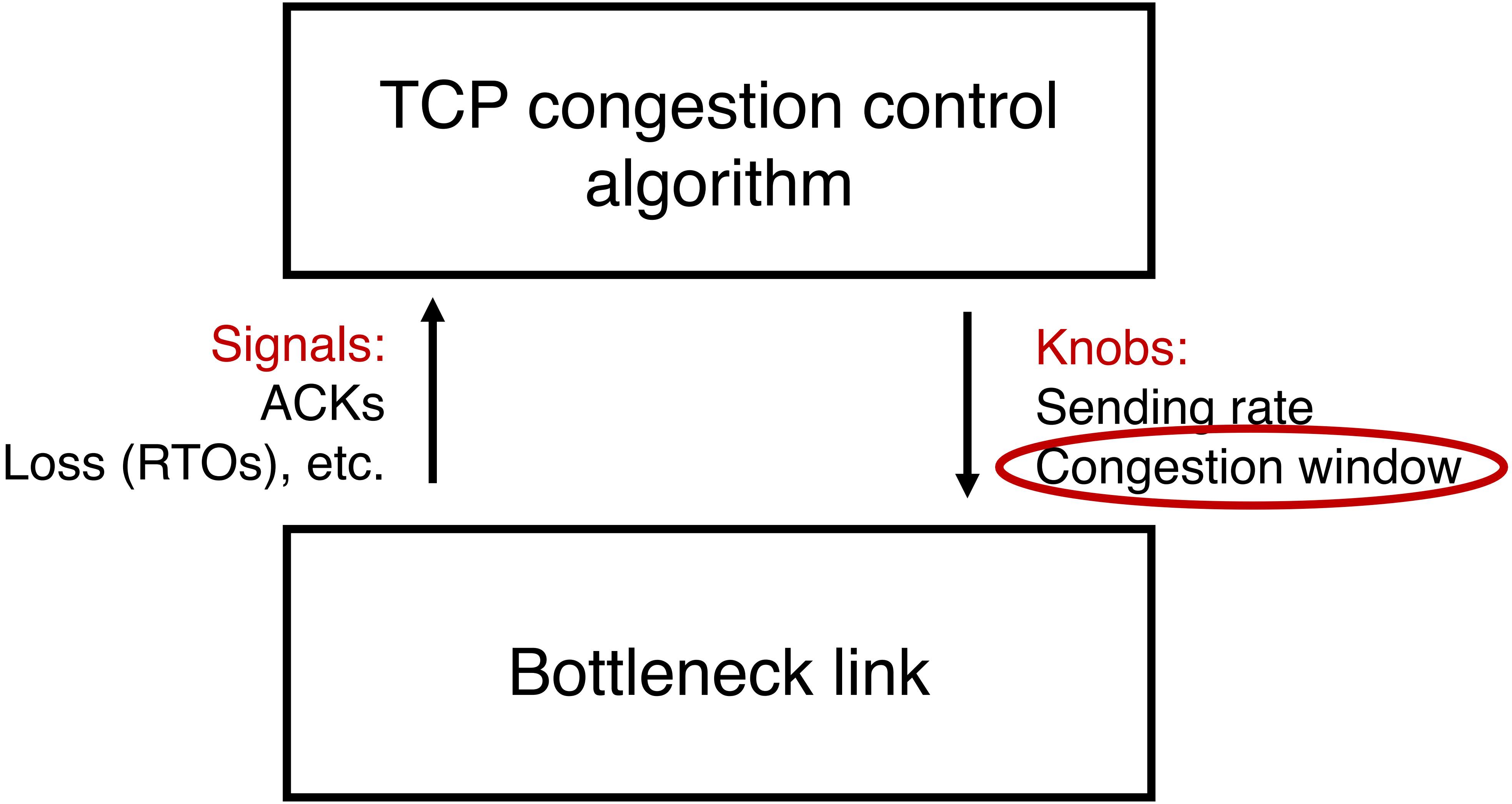
You

Signals:
Water temperature
Water pressure

Knobs:
Turn temperature up/down
Open the tap wider

Your bathroom shower

The congestion control feedback loop



Congestion window

- The sender maintains an estimate of the amount of in-flight data needed to keep the pipe full without congesting it.
- This estimate is called the congestion window (cwnd)
- Recall: There is a relationship between the sending rate (throughput) and the sender's window: sender transmits a window's worth of data over an RTT duration
  - Rate = window / RTT

Interaction b/w flow & congestion control

- Use window = min(congestion window, receiver advertised window)
- Overwhelm neither the receiver nor network links & routers

Window <= Congestion window (congestion control)

Window <= Advertised window (flow control)