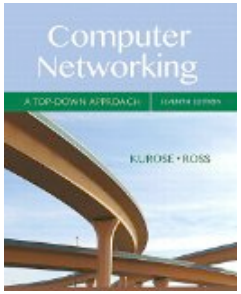
COMP 375: Lecture 23

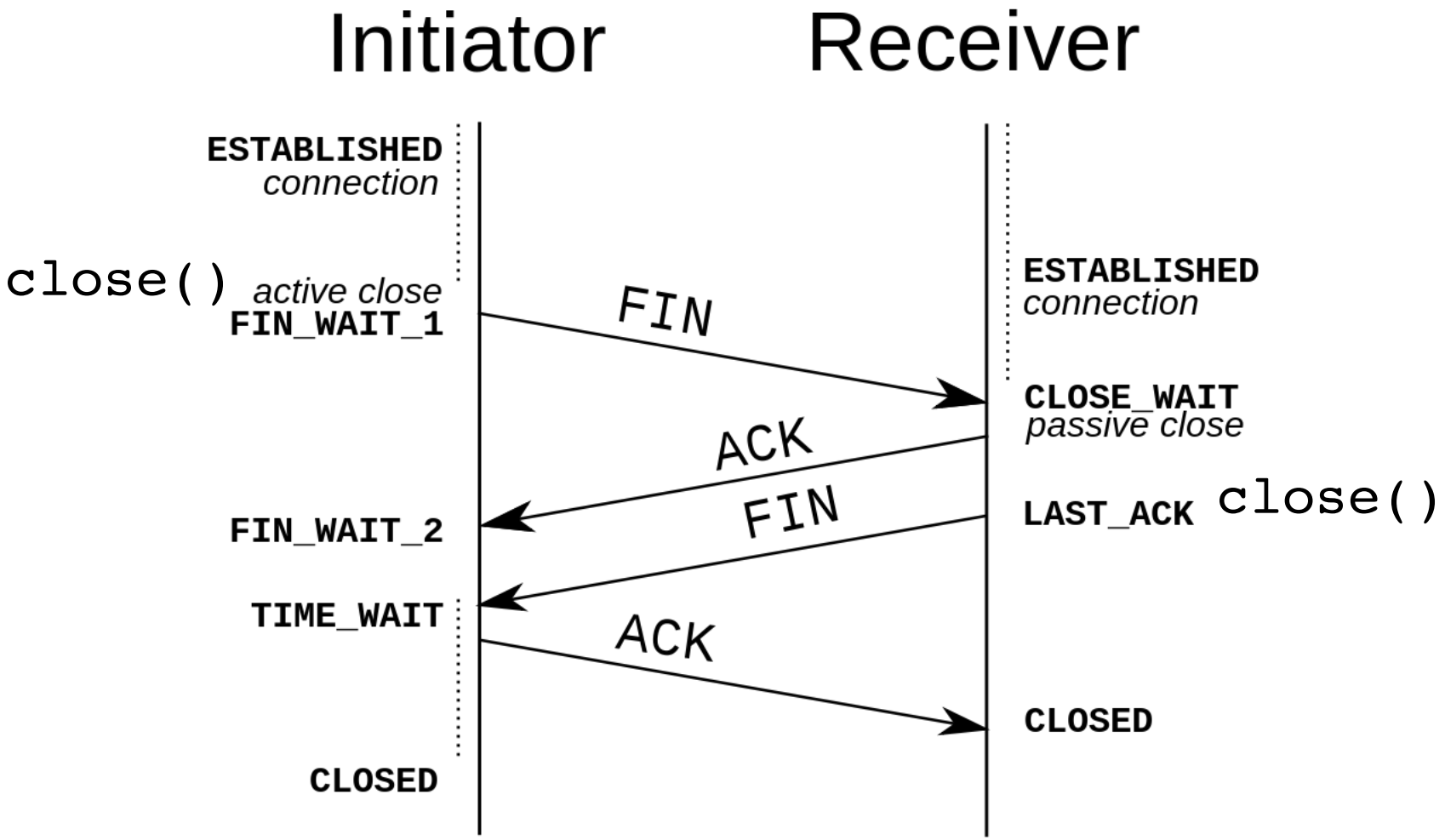


- **News & Notes:**
  - Project #4 due Mon, April 9
- **Reading (Fri, March 23)**
  - Sections 4.{1-2.1}

Over the next two classes, we'll look to answer the following questions about TCP.

- How is pipelining handled?
- How should we choose timeout values?
- **How are connections created and destroyed?**
- How many segments should be in-flight?

TCP **connection teardown** cleans up state, can be initialized by either host.



The TIME_WAIT state is designed for the case where final ACK is lost.

- The initiator waits 2*MSL (≈ 2 minutes) before completing the close.
- *What could go wrong if the final ACK is lost?*

MSL = Maximum segment lifetime (usually 1 minute)

`netstat` can tell you about your TCP connections.

```
sat@cslab1 ~> netstat –at

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:smtp          0.0.0.0:*               LISTEN
tcp        0      0 cslab1.sandiego.edu:ssh 10.41.0.121:55258       ESTABLISHED
tcp        0      0 cslab1.sandiego.e:53128 usd-satellite.sand:5647 ESTABLISHED
tcp        0      0 cslab1.sandiego.edu:906 gamma.sandiego.edu:nfs  ESTABLISHED
tcp        0     52 cslab1.sandiego.edu:ssh 172.17.8.84:56346       ESTABLISHED
tcp        0      0 cslab1.sand:filenet-tms ldap.sandiego.edu:ldaps ESTABLISHED
```

Over the next two classes, we'll look to answer the following questions about TCP.

- How is pipelining handled?
- How should we choose timeout values?
- How are connections created and destroyed?
- **How many segments should be in-flight?**

We want to avoid wasting bandwidth by sending something likely to be dropped.

Discuss: What are some reasons why a segment might be *dropped*?

Flow control is needed when receiver can't handle sender's transfer rate.

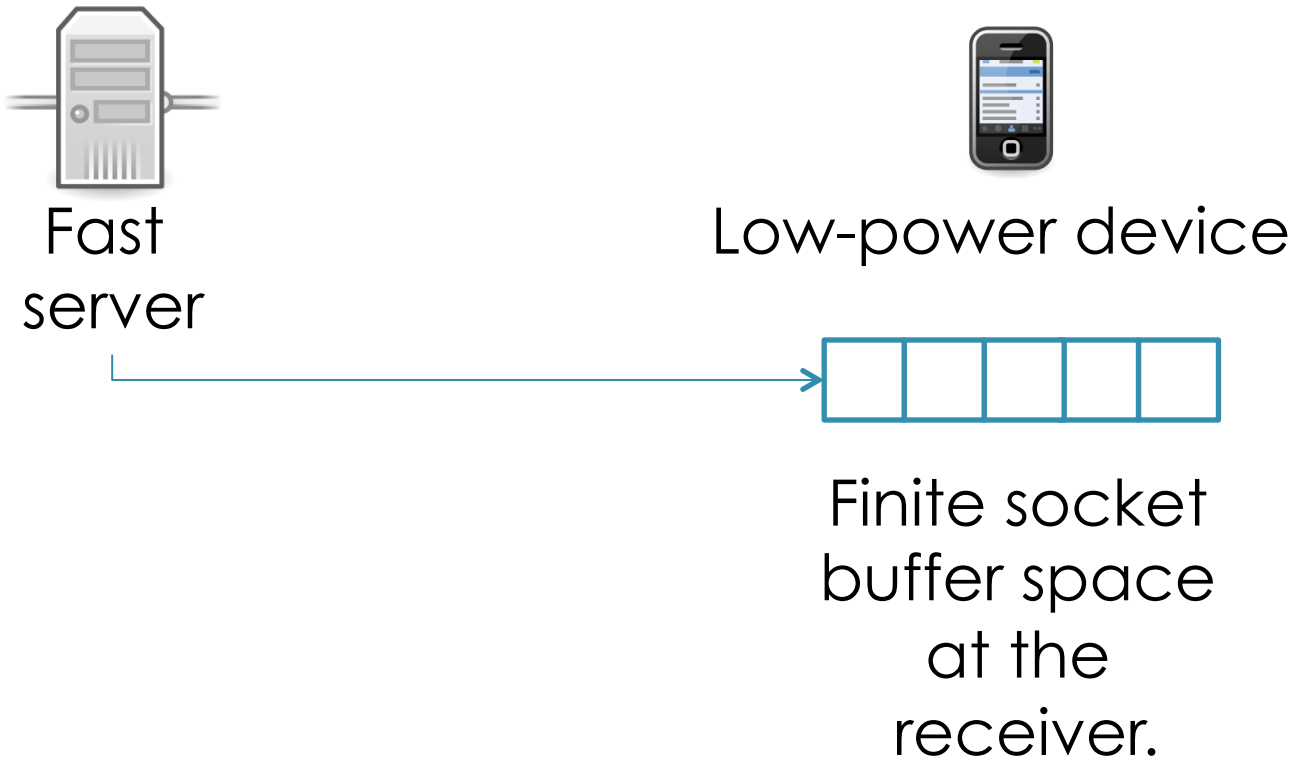
Fast server

Low-power device

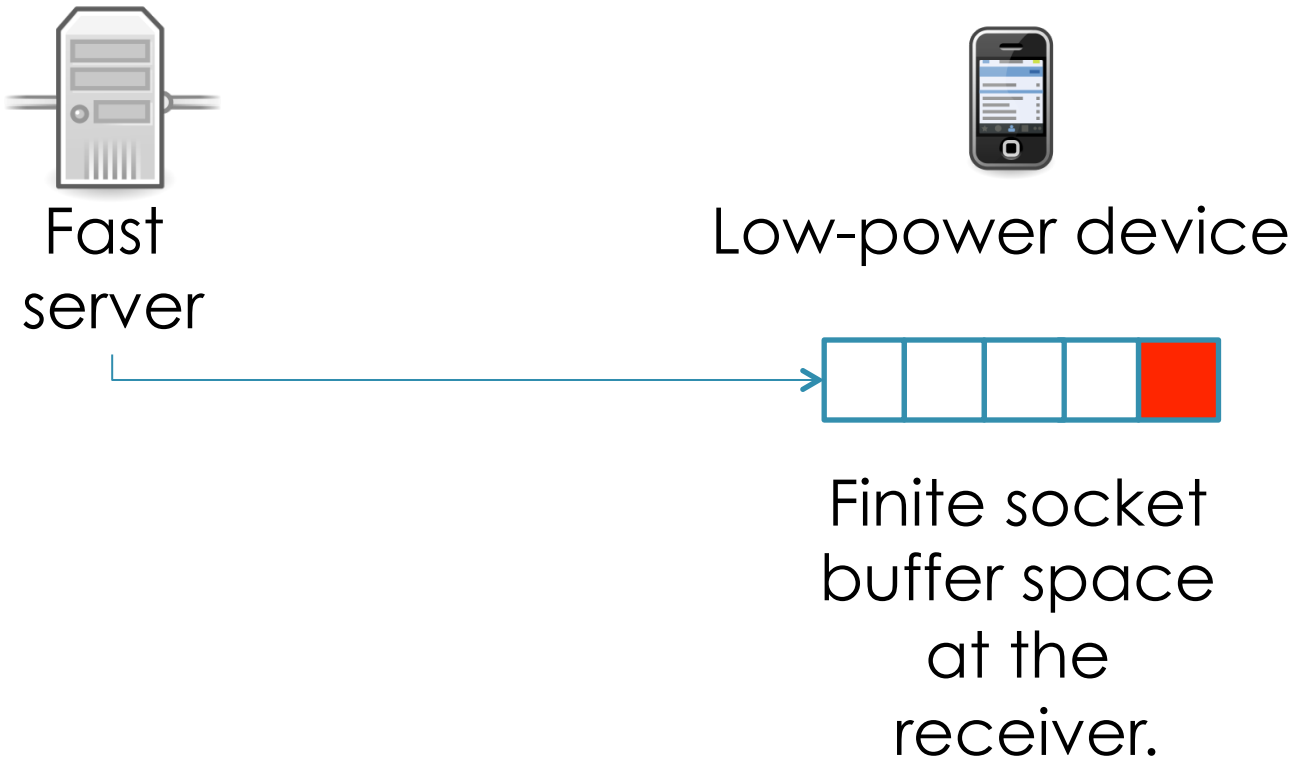
Multiple fast servers

Fast server

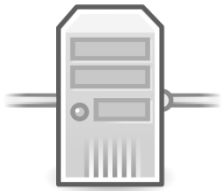
Flow control is needed when receiver can't handle sender's transfer rate.



Flow control is needed when receiver can't handle sender's transfer rate.



Flow control is needed when receiver can't handle sender's transfer rate.



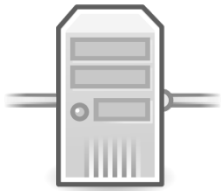
Fast server

App calls `recv()`

Low-power device

Finite socket buffer space at the receiver.

Flow control is needed when receiver can't handle sender's transfer rate.



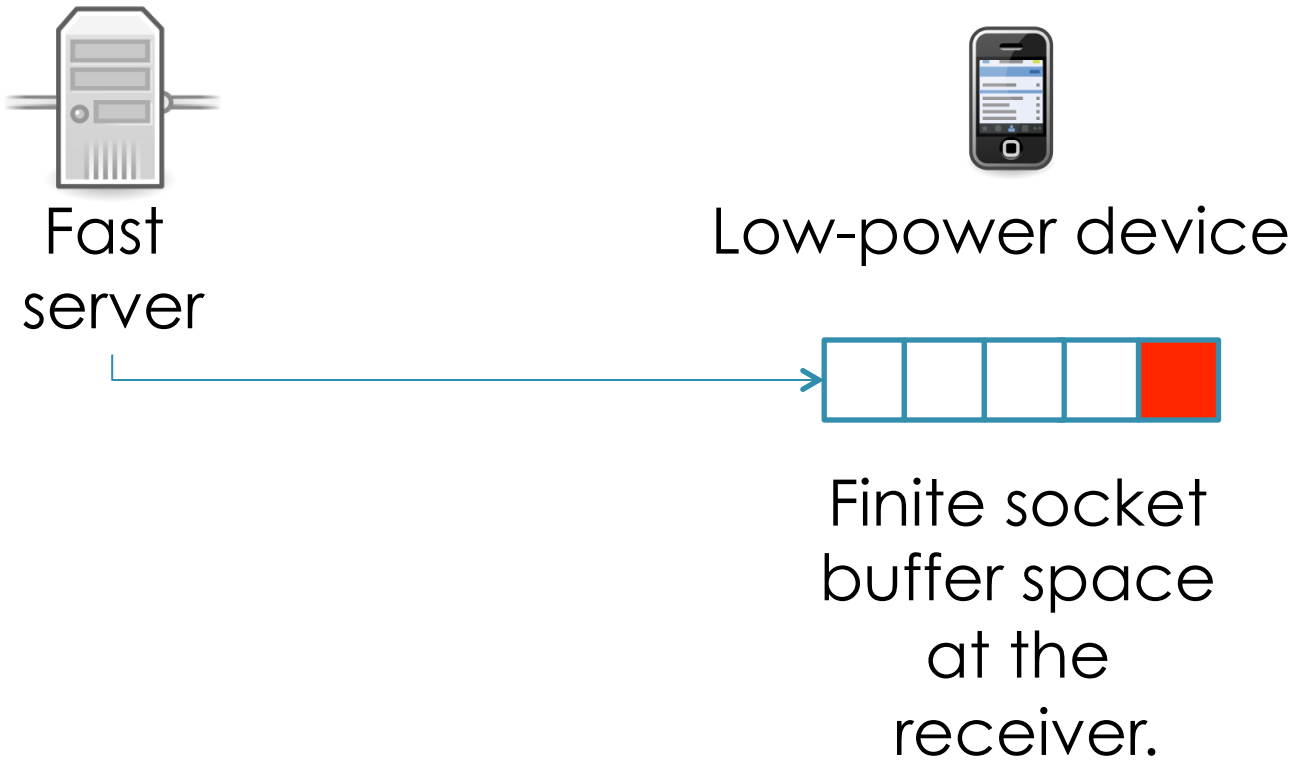
Fast server

App calls `recv()`

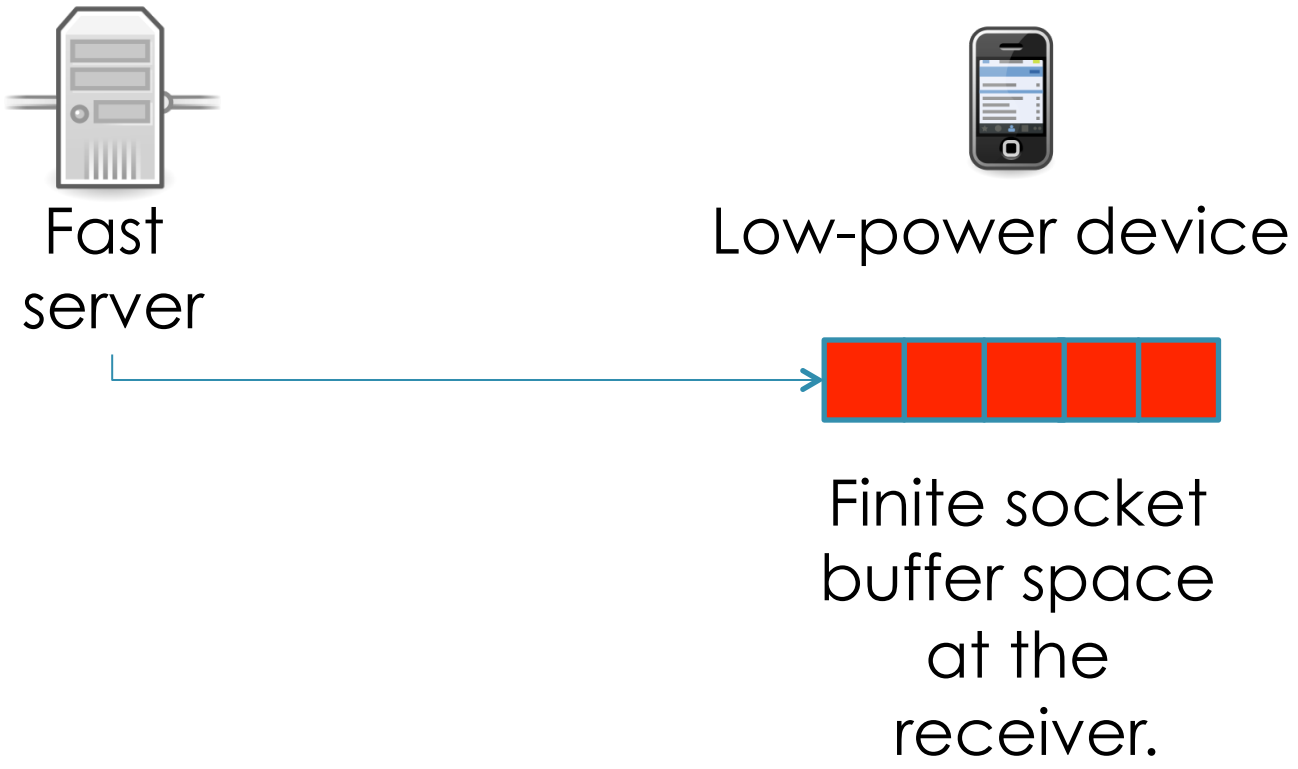
Low-power device

Finite socket buffer space at the receiver.

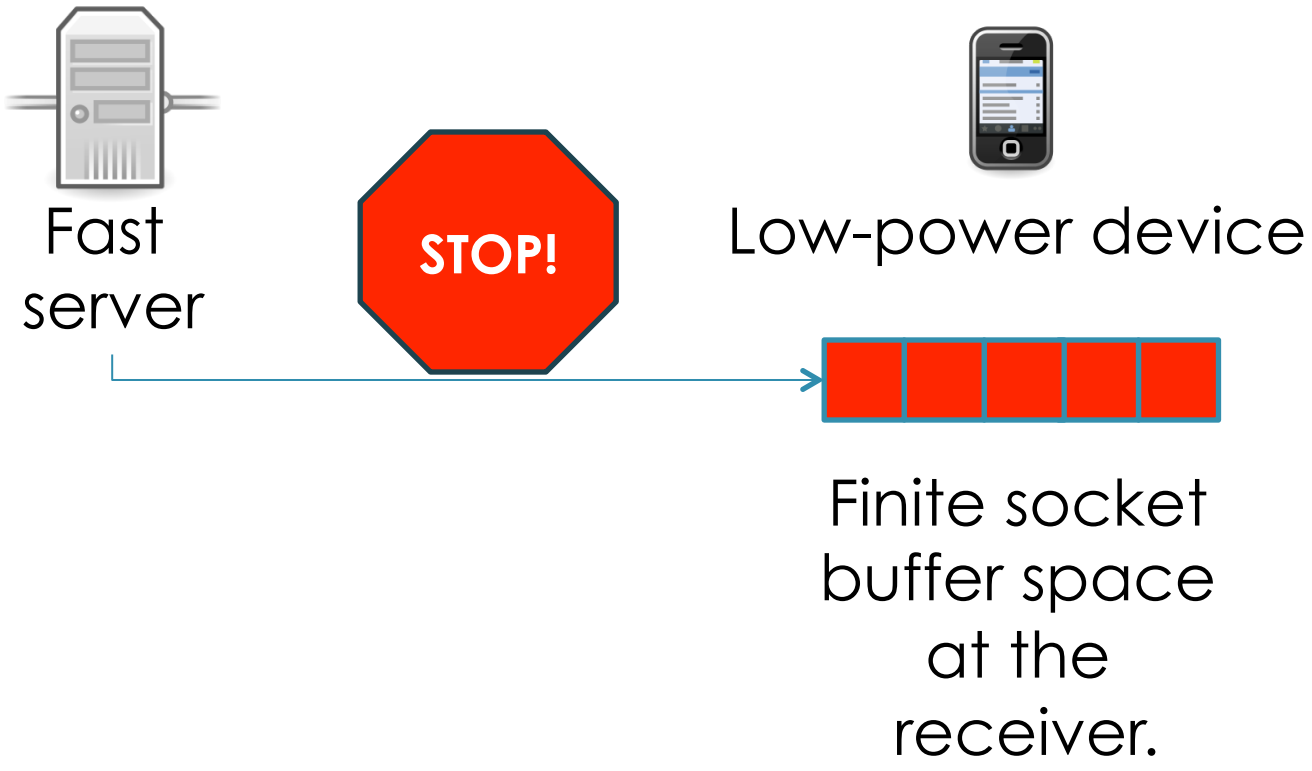
Flow control is needed when receiver can't handle sender's transfer rate.



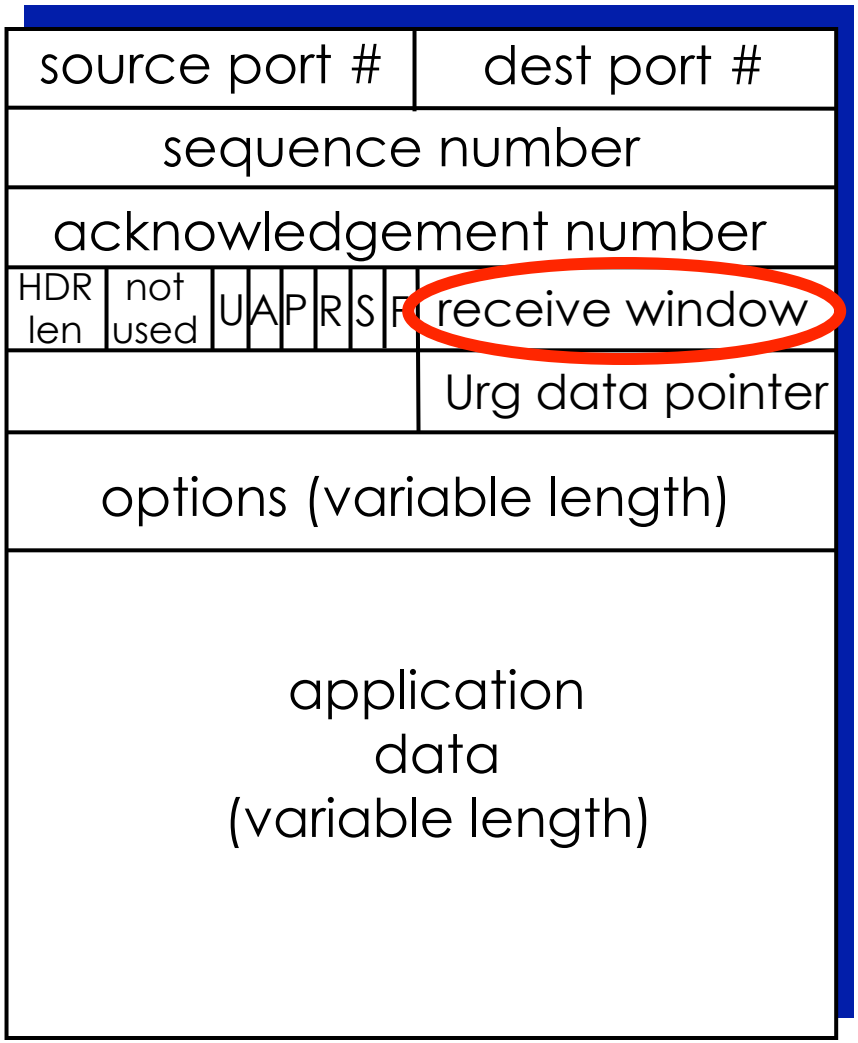
Flow control is needed when receiver can't handle sender's transfer rate.



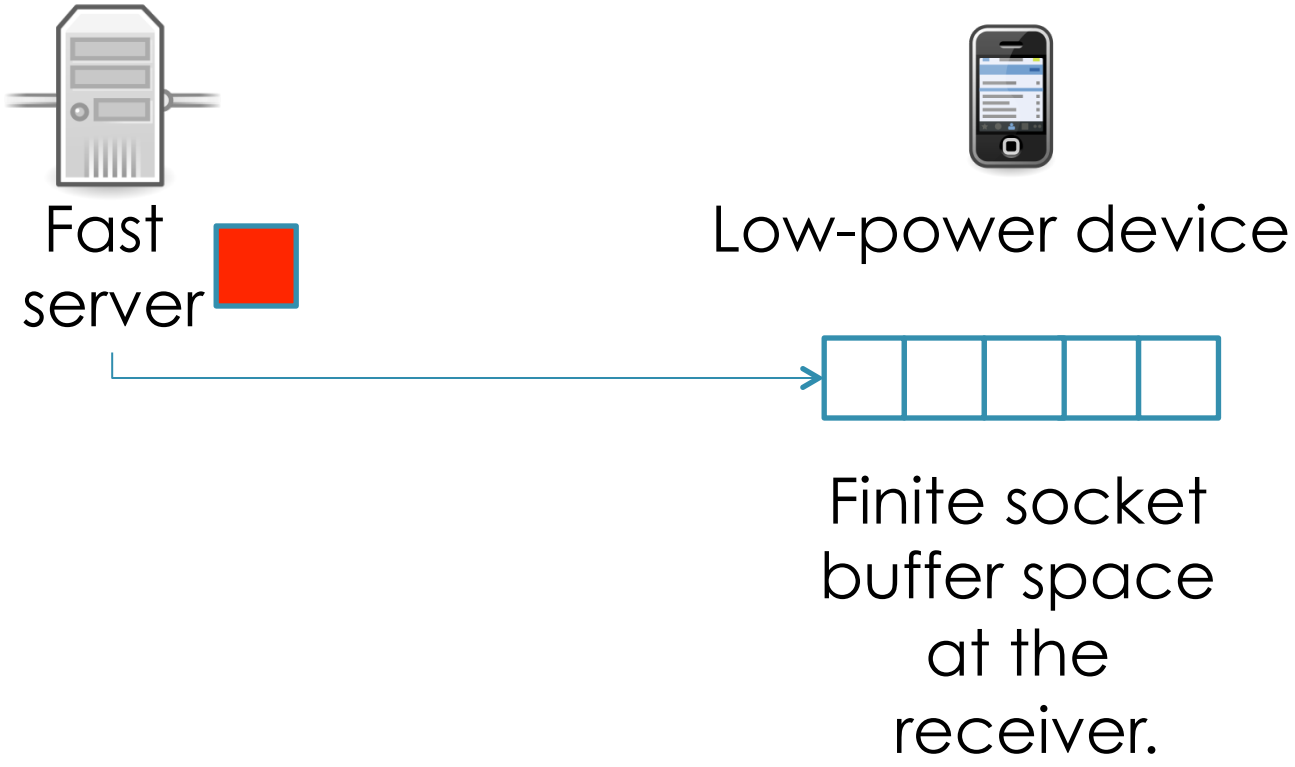
Flow control is needed when receiver can't handle sender's transfer rate.



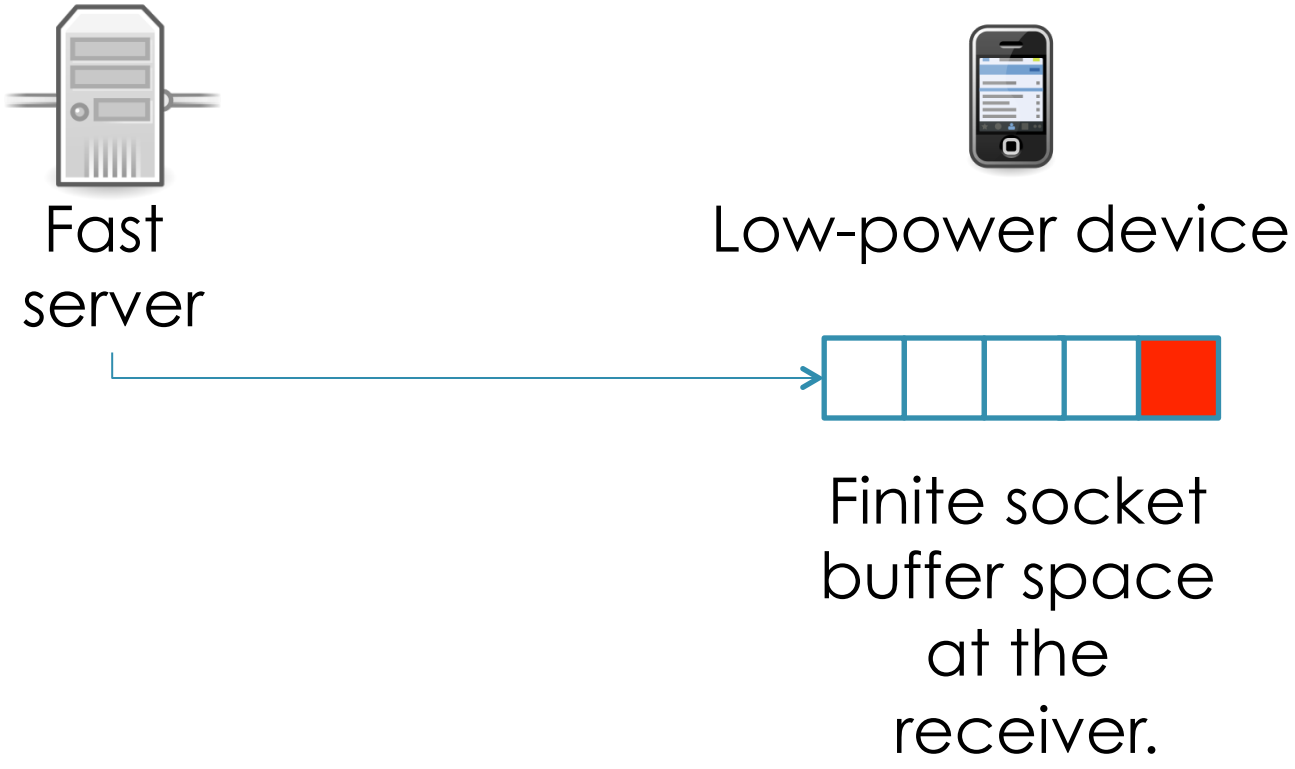
Receiver can inform sender of its window size via normal TCP message.



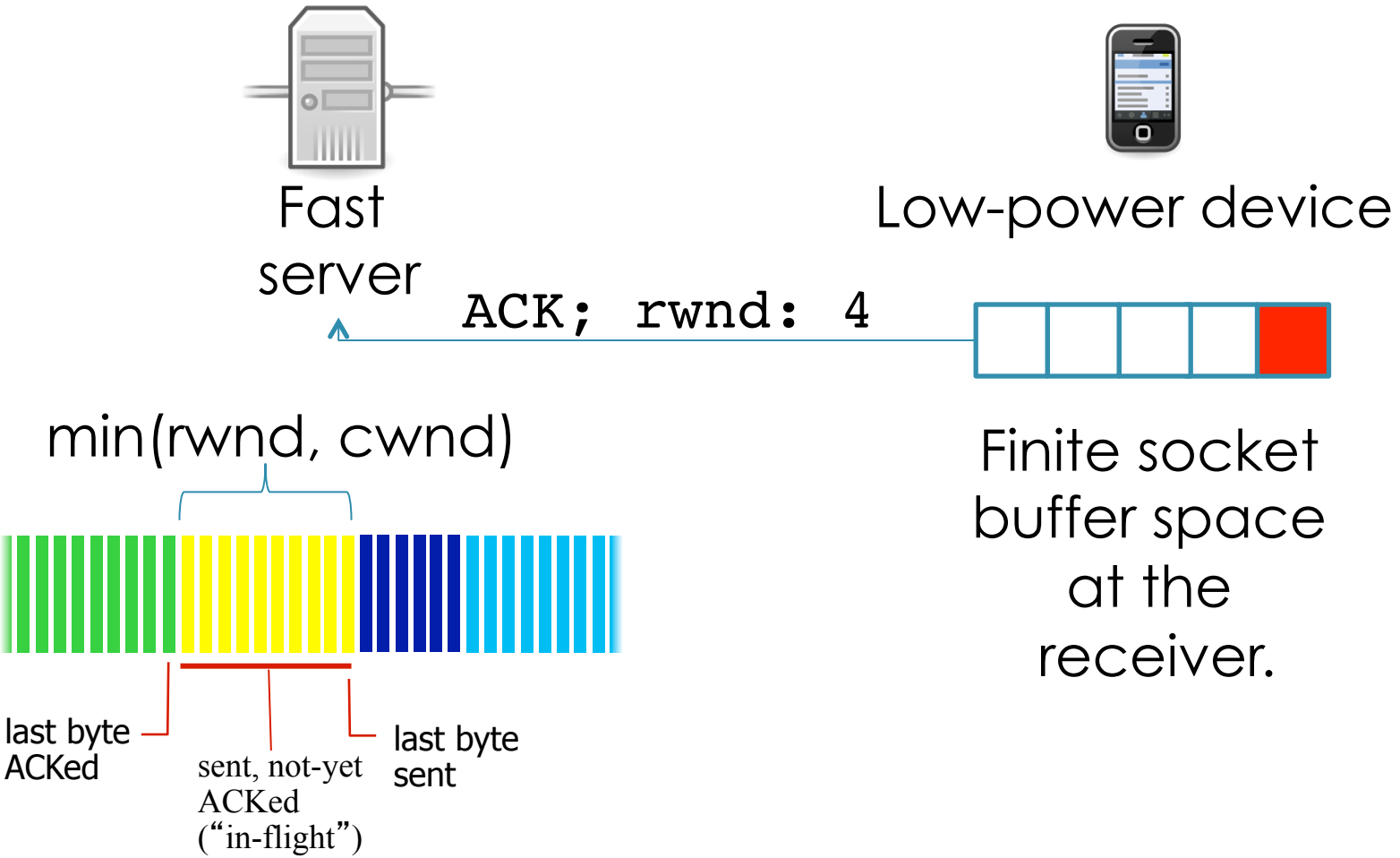
With flow control, sender never sends more than `rwnd`.



With flow control, sender never sends more than `rwnd`.



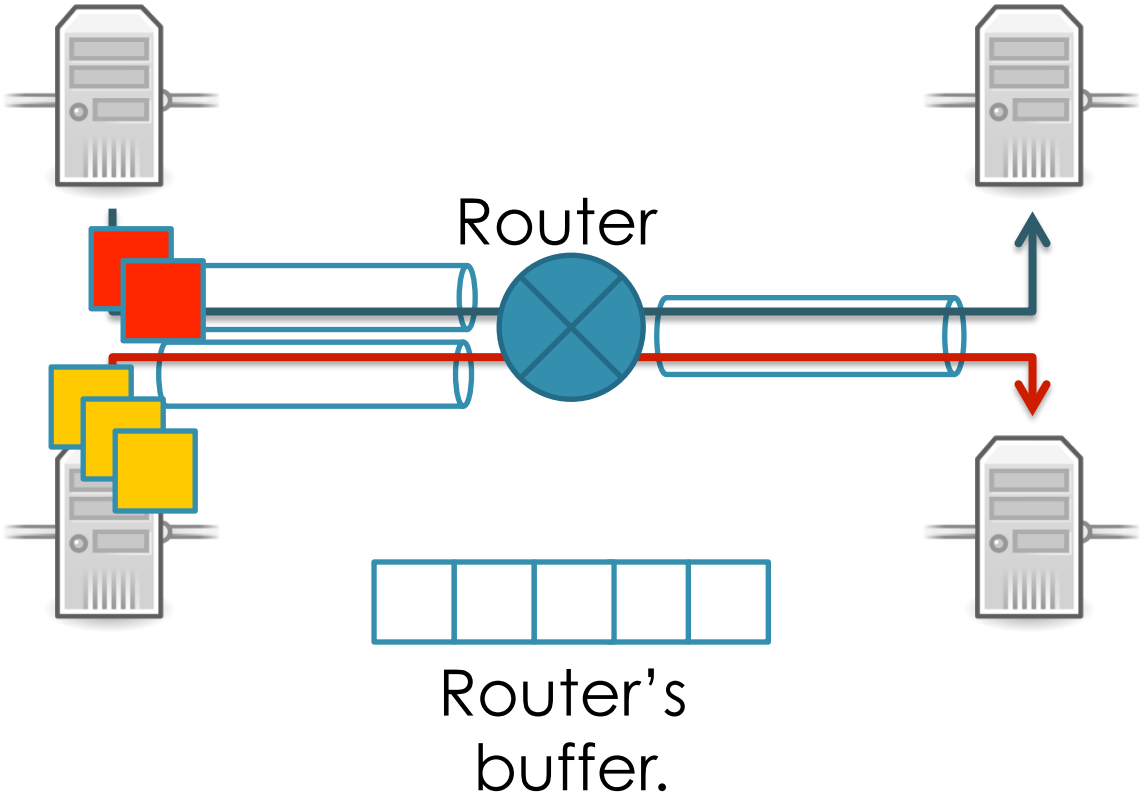
With flow control, sender never sends more than `rwnd`.



Flow control is relatively easy: the receiver tells us what we need to know.

What about the network devices?

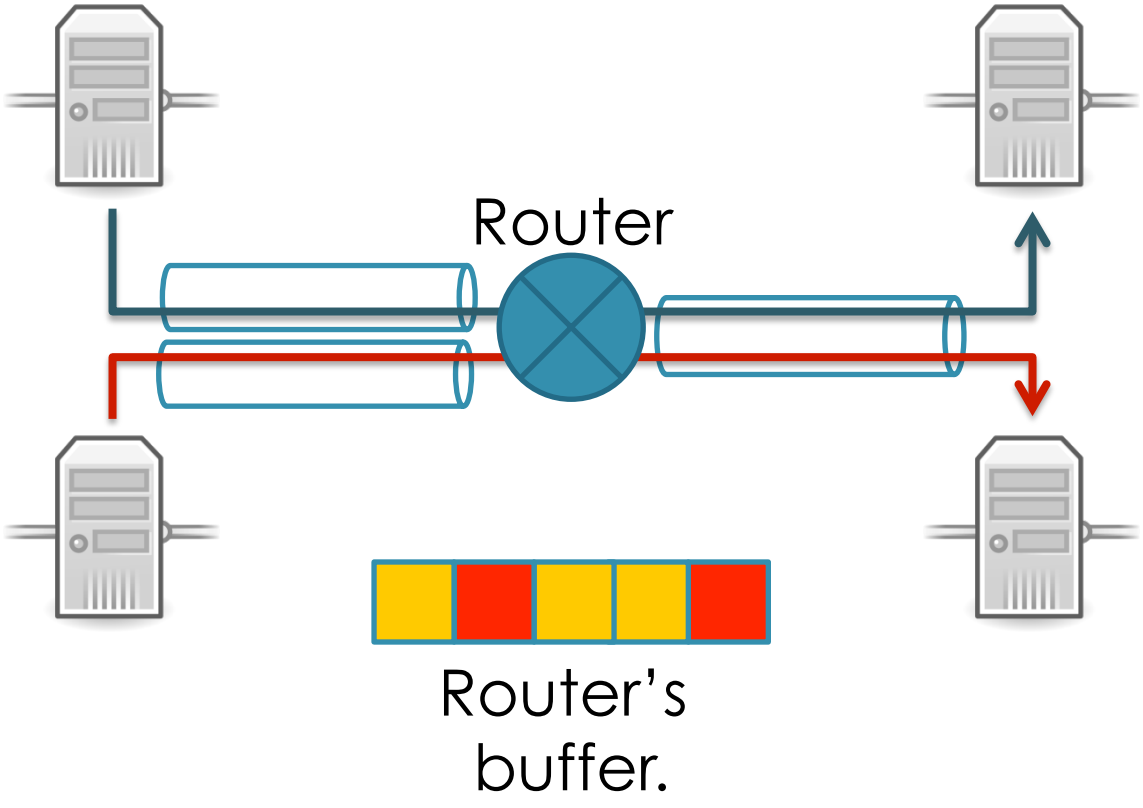
Routers have to buffer packets while they determine the next destination



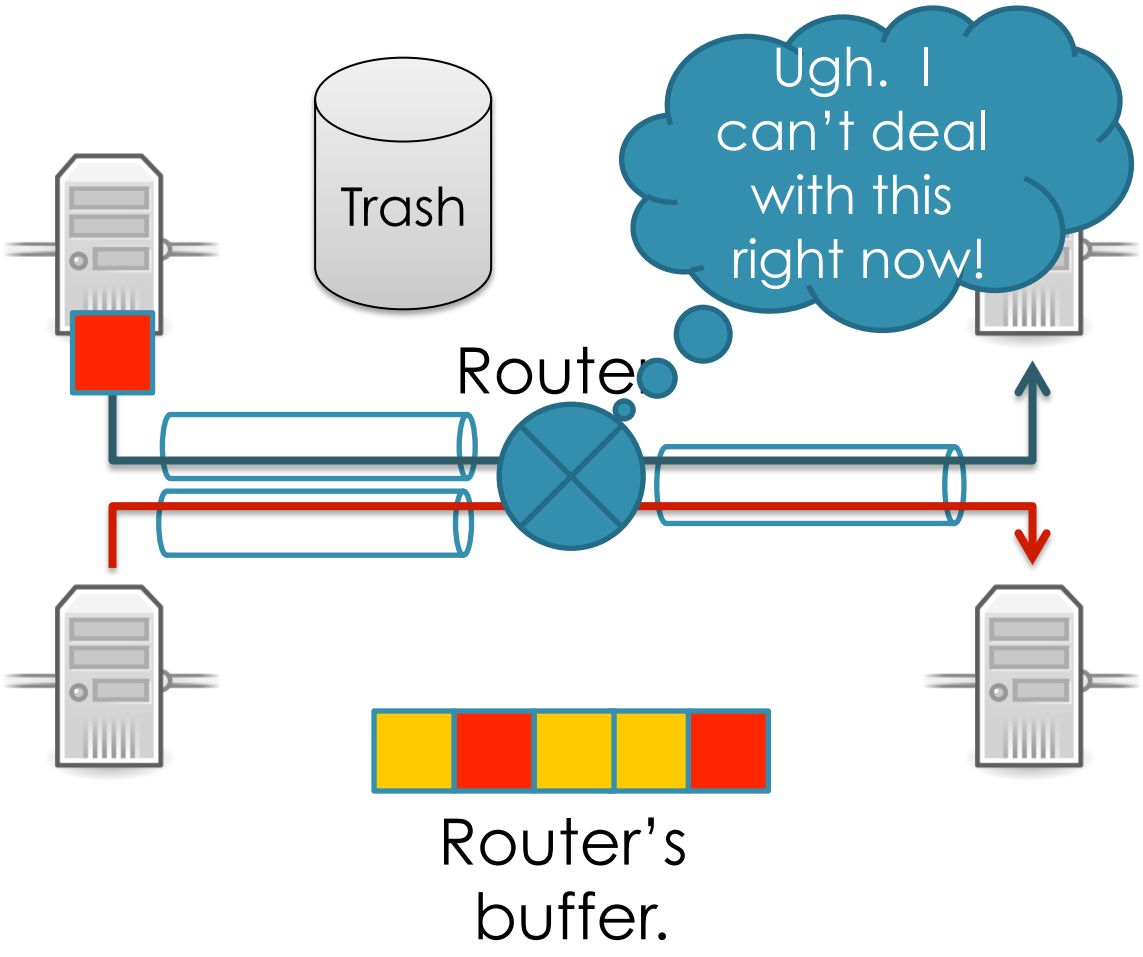
If routers have infinite buffers, we don't have to worry about dropped packets.

- Assume that the max transmission rate of a link is *R* and that our router has an infinite buffer.
- **Discuss:** *If we have 2 connections sending at R/2 each, will there be a problem?*

Routers run into problems when the incoming rate(s) exceed the outgoing rate.



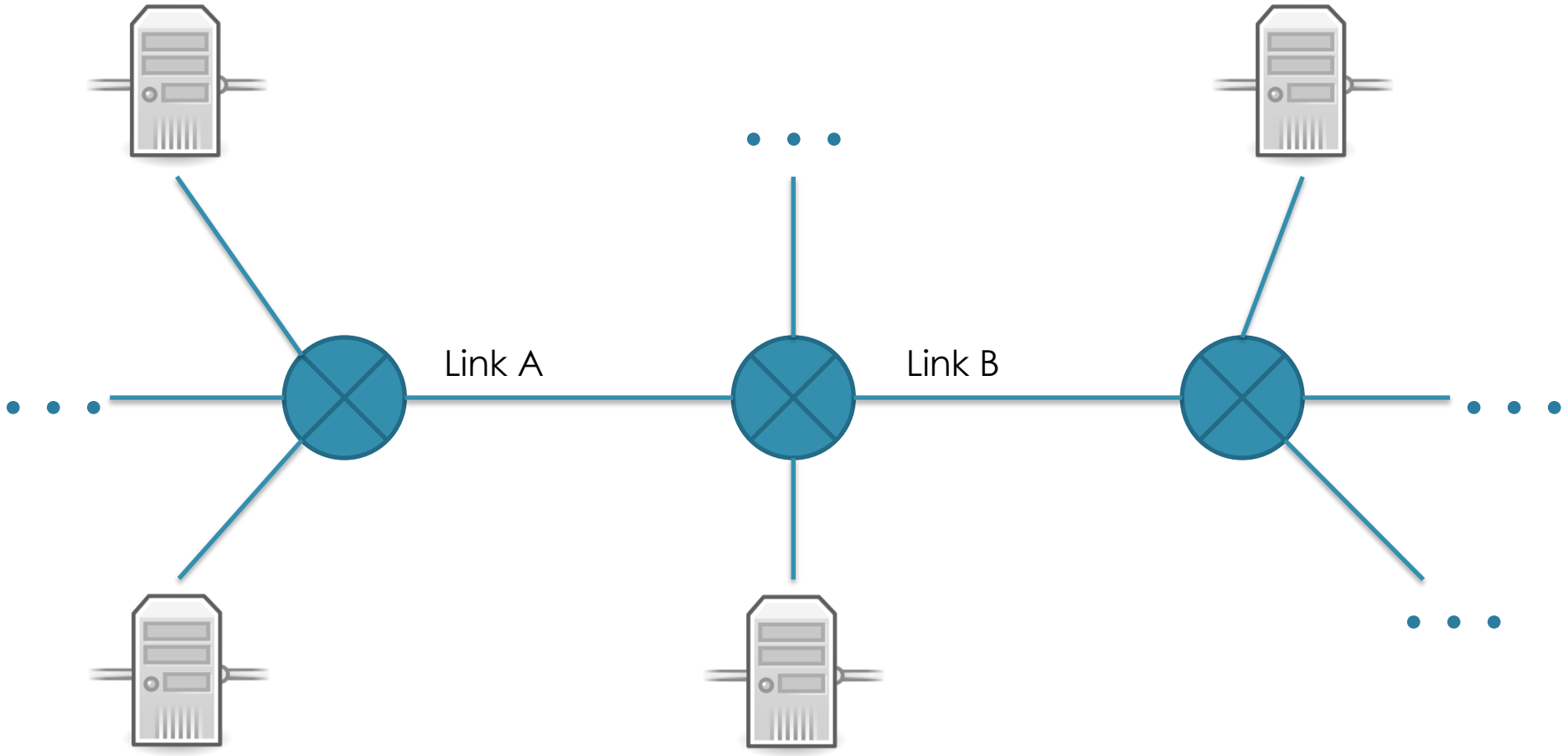
Routers use drop-tail queuing to handle overloaded buffers.



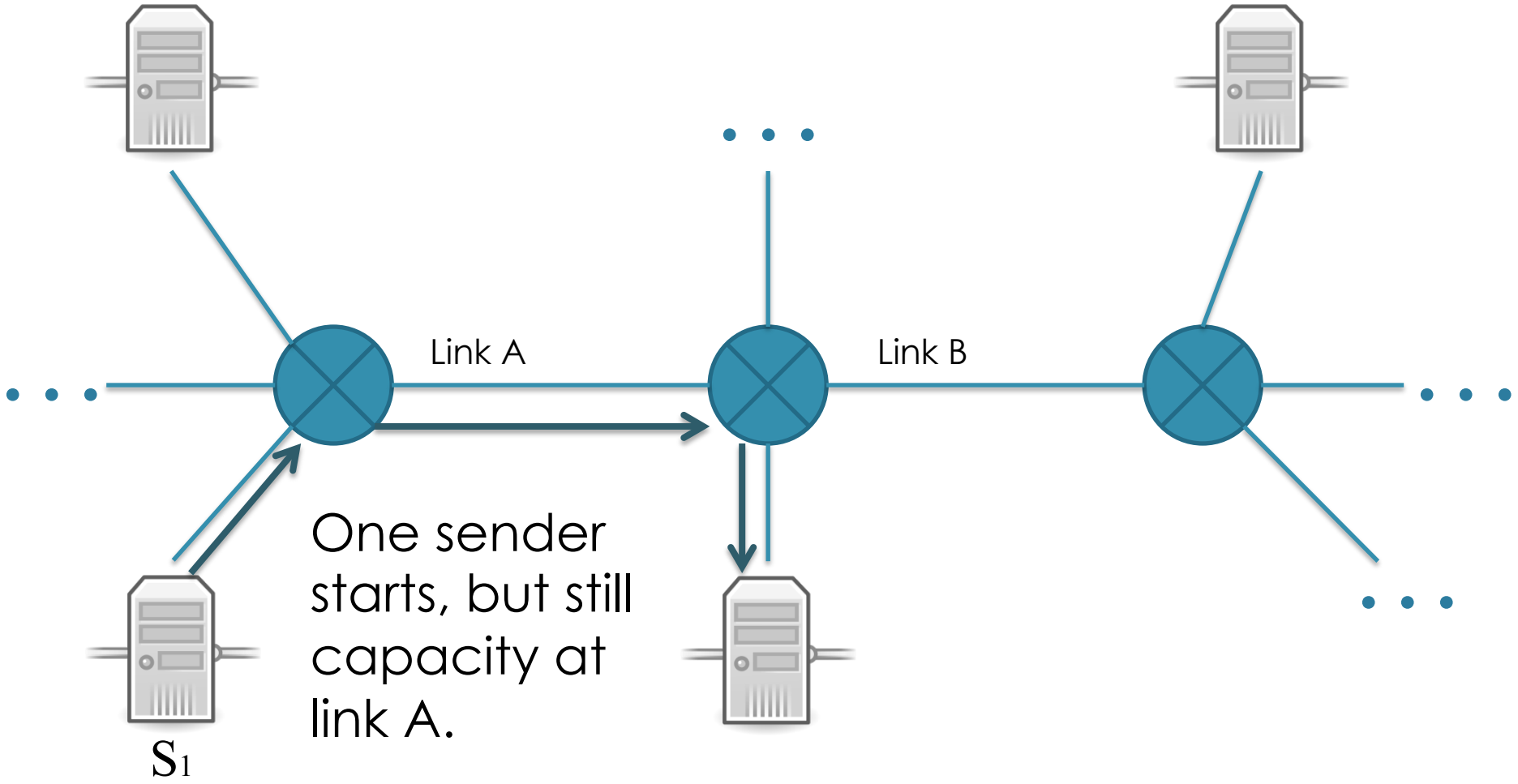
What's the worst that can happen?

A.	This is no problem. Senders just keep transmitting, and it'll all work out.
B.	There will be retransmissions, but the network will still perform without much trouble.
C.	Retransmissions will become very frequent, causing a serious loss of efficiency.
D.	The network will become completely unusable.

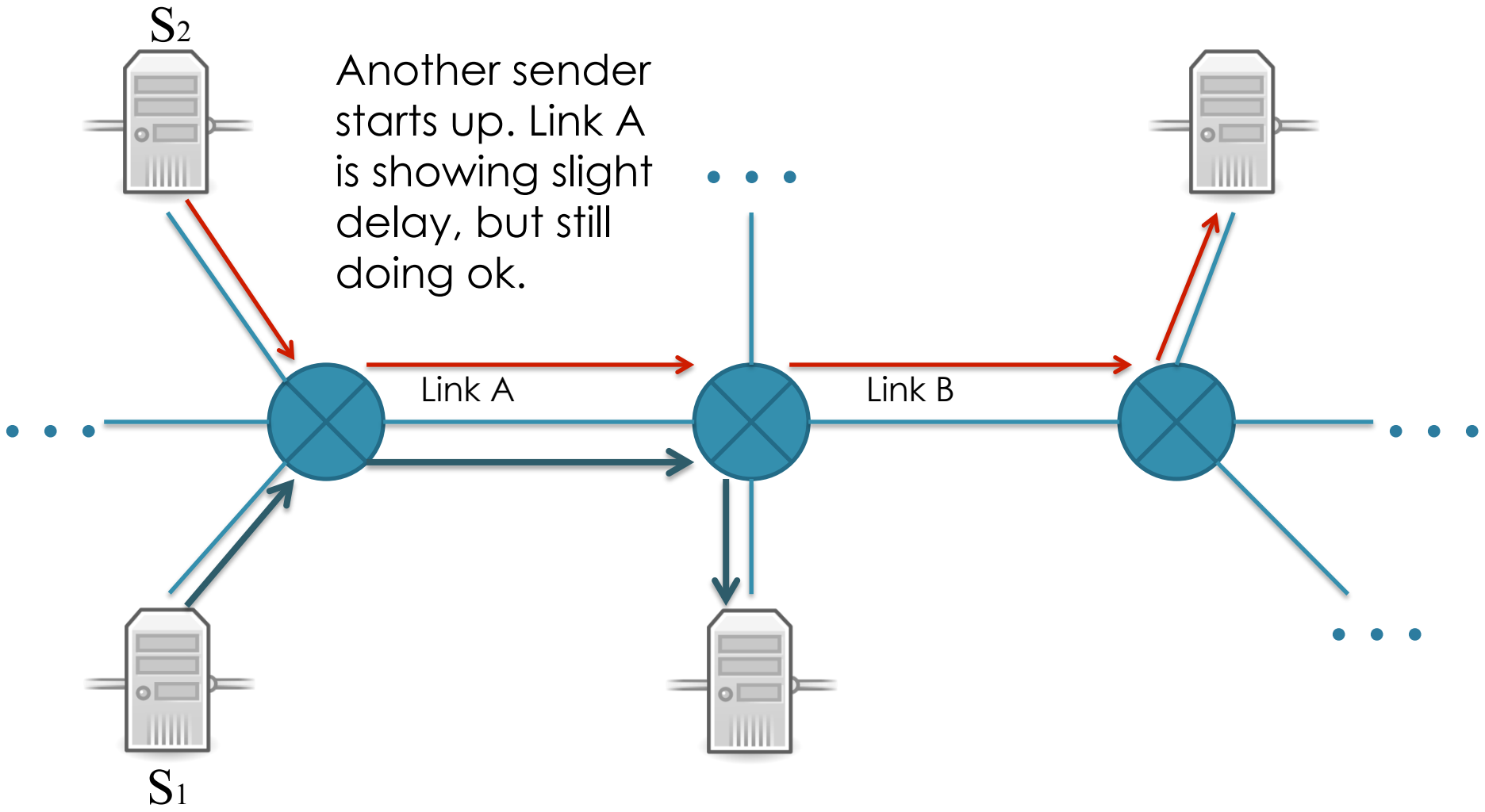
Without control, a network can face congestion collapse.



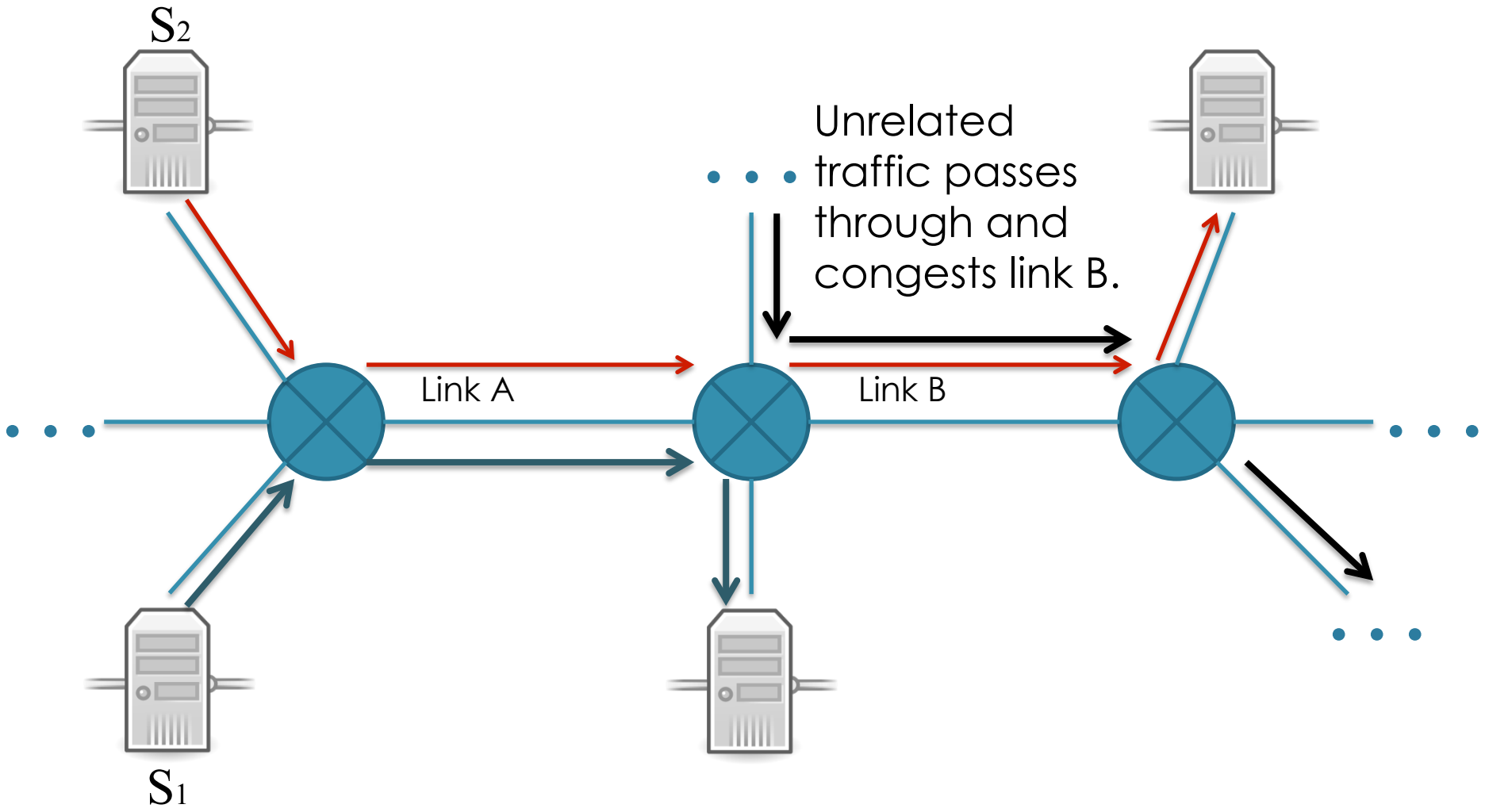
Without control, a network can face congestion collapse.



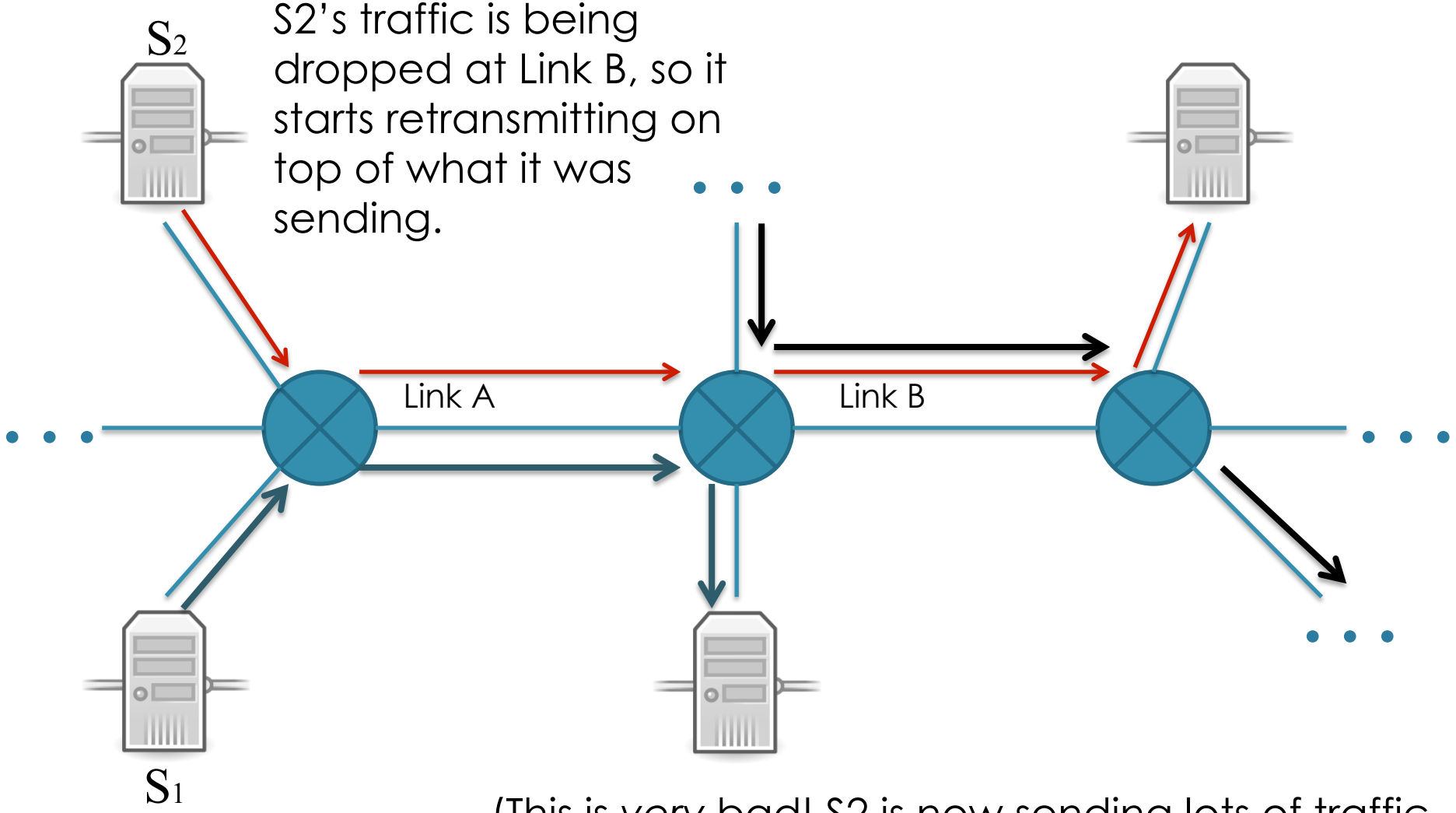
Without control, a network can face congestion collapse.



Without control, a network can face congestion collapse.

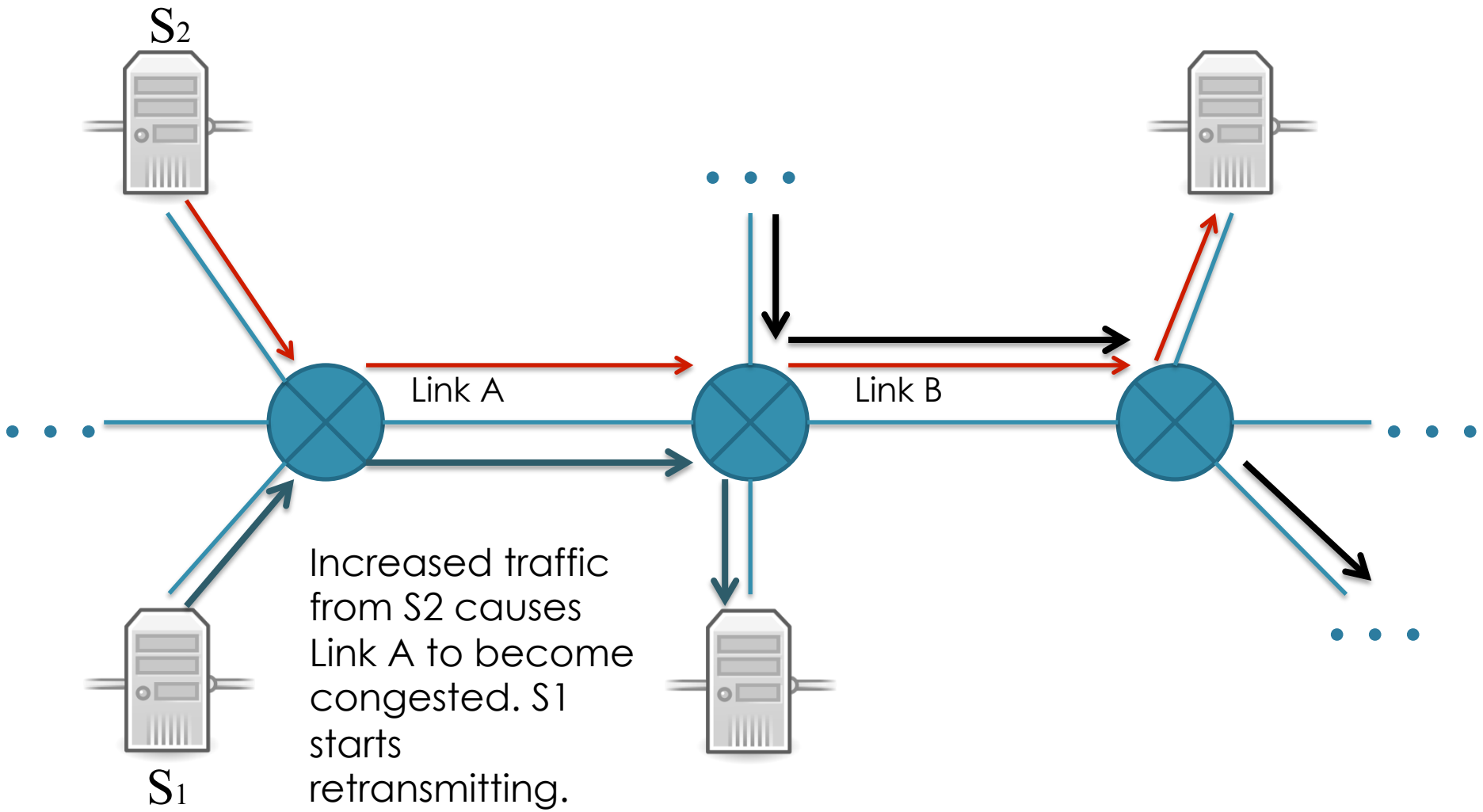


Without control, a network can face congestion collapse.

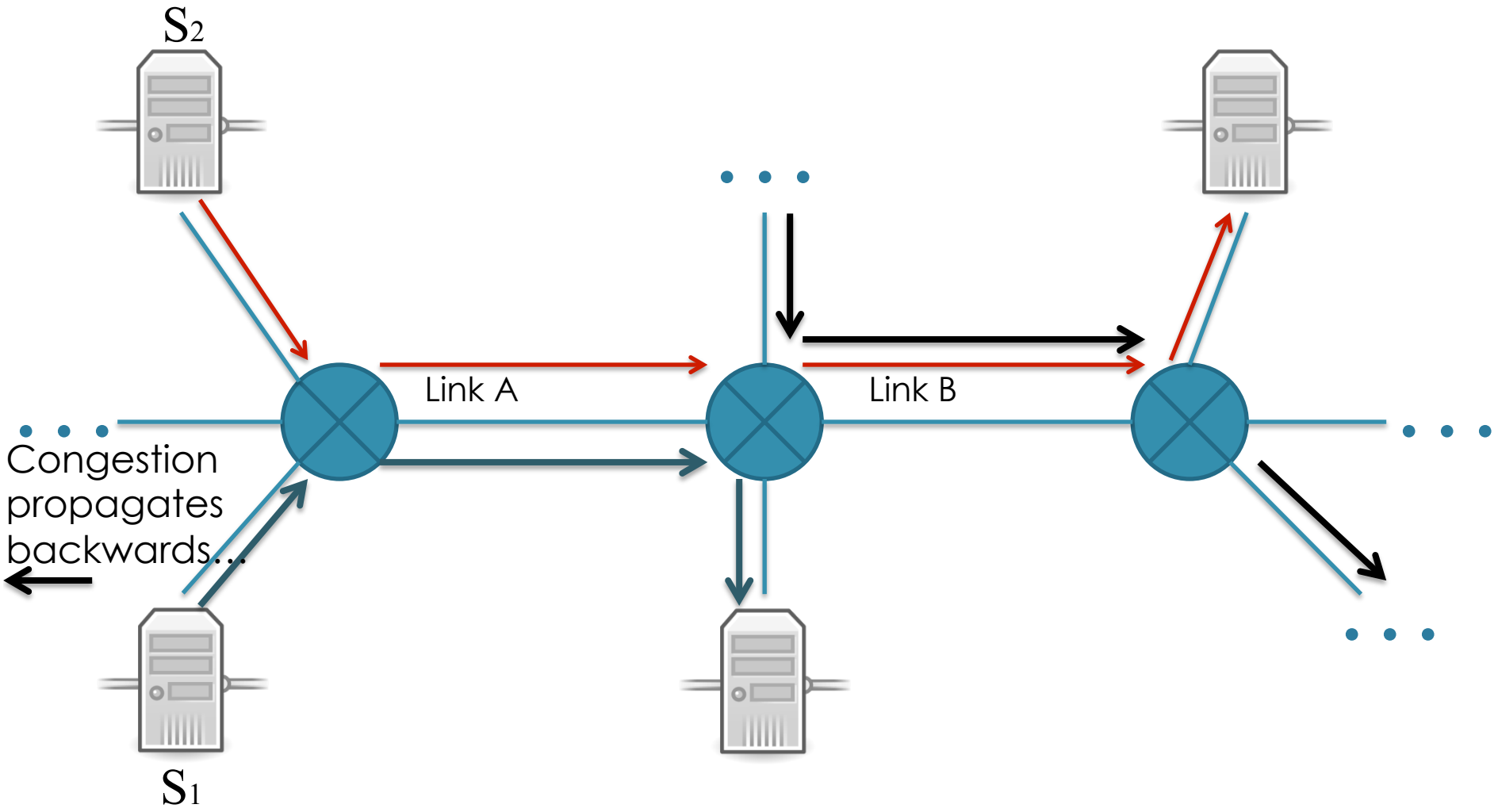


(This is *very* bad! S2 is now sending lots of traffic over link A that has no hope of crossing link B.)

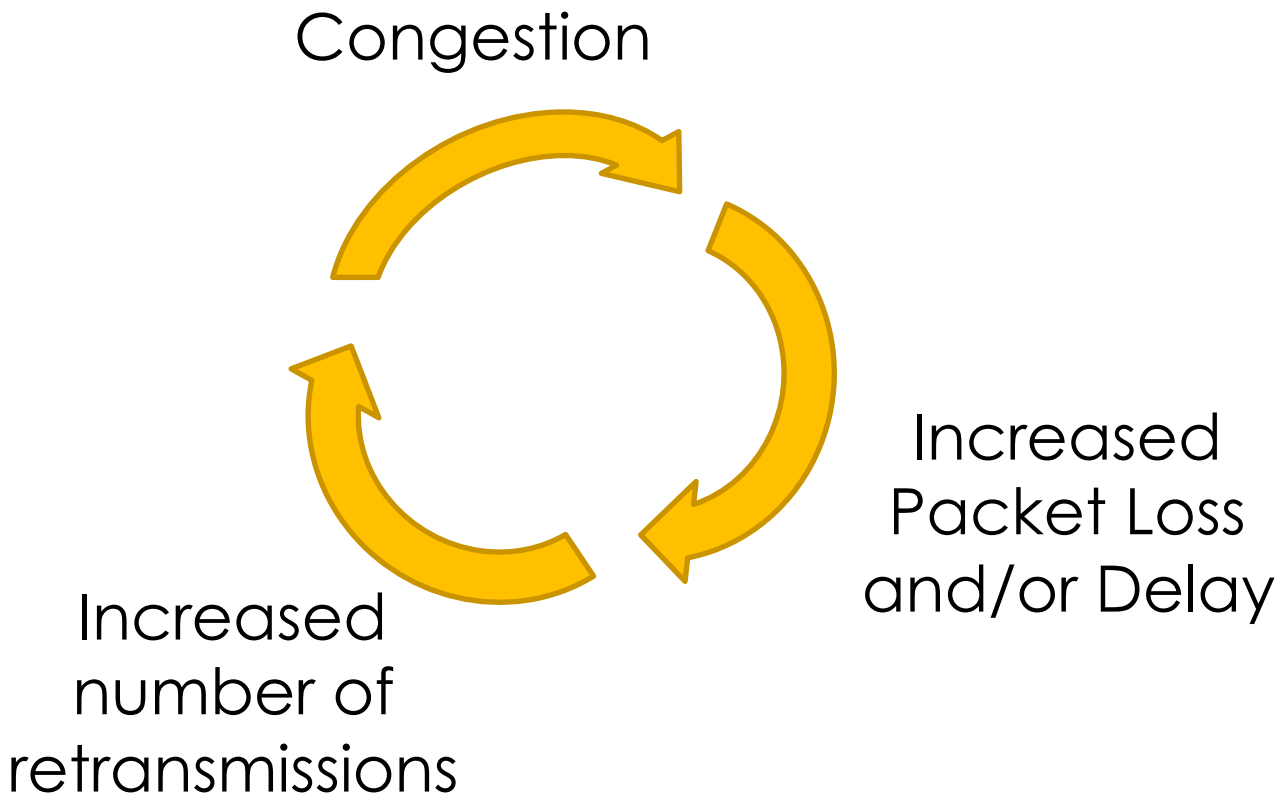
Without control, a network can face congestion collapse.



Without control, a network can face congestion collapse.



Congestion begets more congestion if not controlled.



Congestion Avoidance and Control*

Van Jacobson†
Lawrence Berkeley Laboratory
Michael J. Karels‡
University of California at Berkeley

November, 1988

In October of '86, the Internet had the first of what became a series of 'congestion collapses'. During this period, the data throughput from LBL to UC Berkeley (sites separated by 400 yards and two IMP hops) dropped from 32 Kbps to 40 bps. We were fascinated by this sudden factor-of-thousand drop in bandwidth and embarked on an investigation of why things had gotten so bad. In particular, we wondered if the 4.3BSD (Berkeley UNIX) TCP was mis-behaving or if it could be tuned to work better under abysmal network conditions. The answer to both of these questions was "yes".

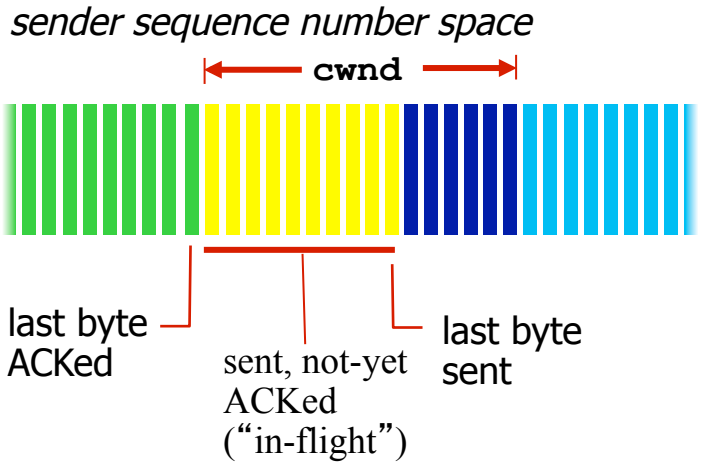
---

* **Note:** This is a very slightly revised version of a paper originally pre-

How do we know if there is congestion in the network?

A.	The network will explicitly tell us.
B.	The sender has to guess based on its observations.
C.	A and B
D.	Some other way.

Sender's rate is a function of `cwnd` and RTT.



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

How should we set `cwnd`?

A.	We should keep raising it until a "congestion event", then back off slightly until we notice no more events.
B.	We should raise it until a "congestion event", then go back to 1 and start raising it again.
C.	We should raise it until a "congestion event", then go back to a median value and start raising it again.
D.	We should send as fast as possible at all times.