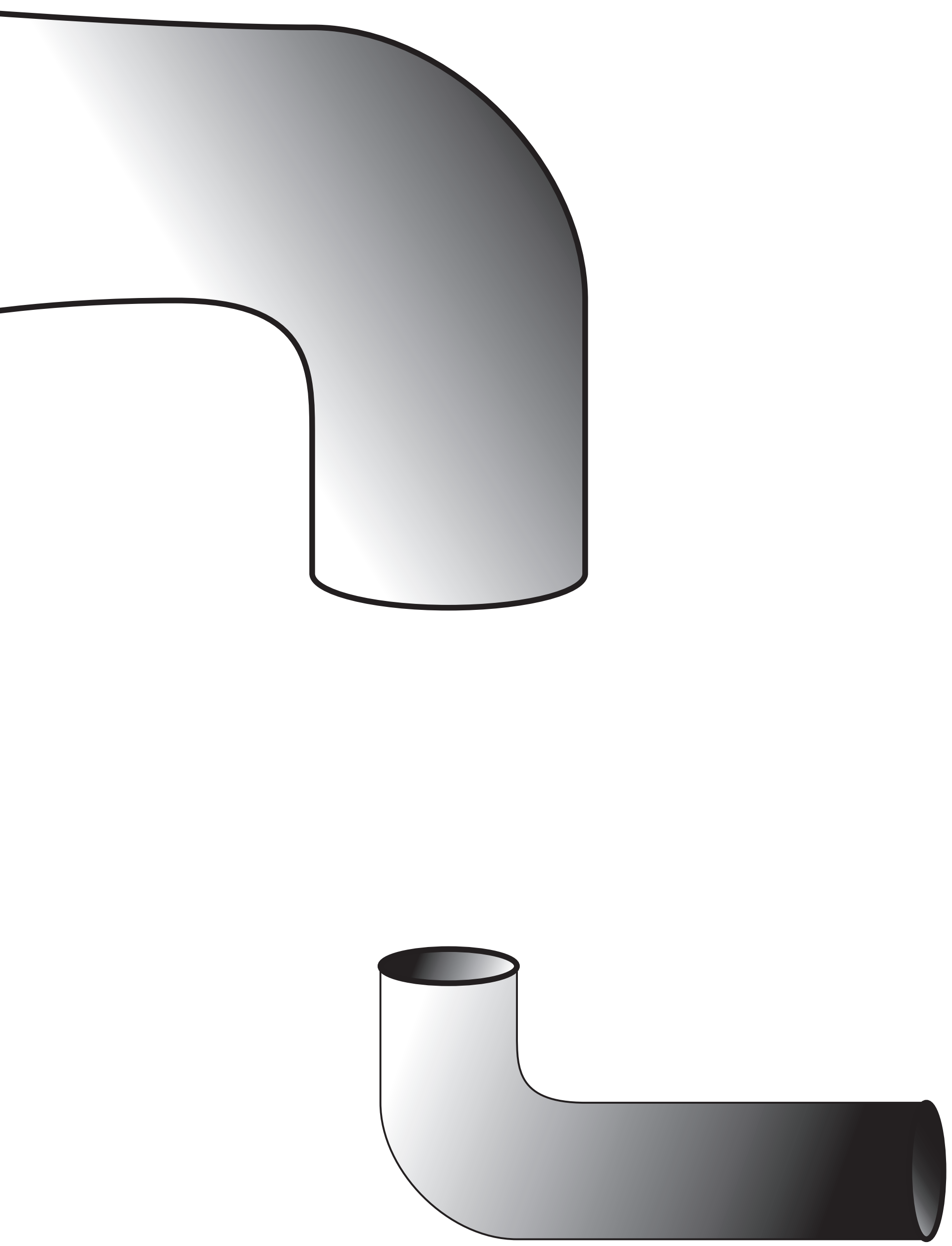
A Rant on Queues

Van Jacobson

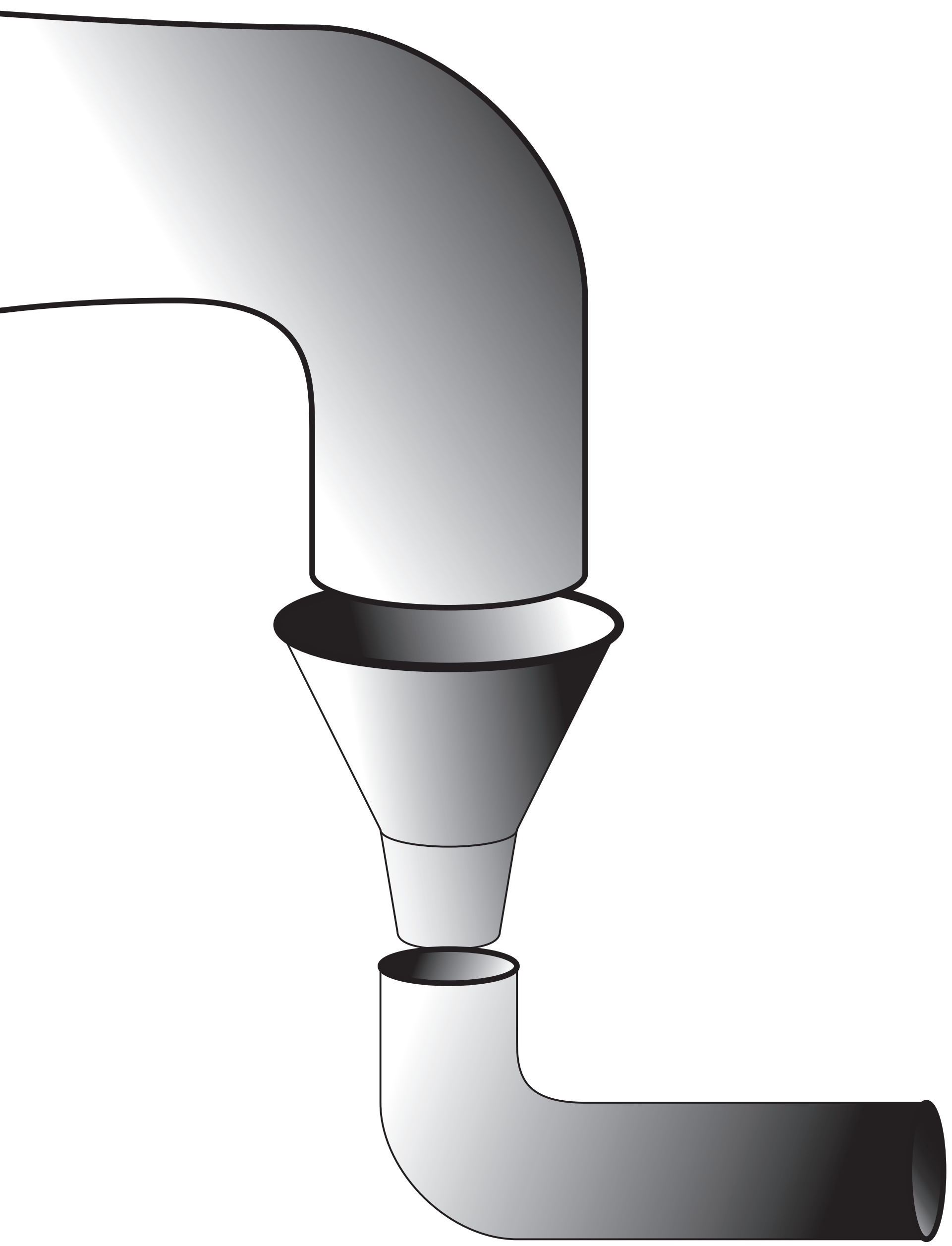
July 26, 2006

MIT Lincoln Labs
Lexington, MA



- Unlike the phone system, the Internet supports communication over paths with diverse, time varying, bandwidth.

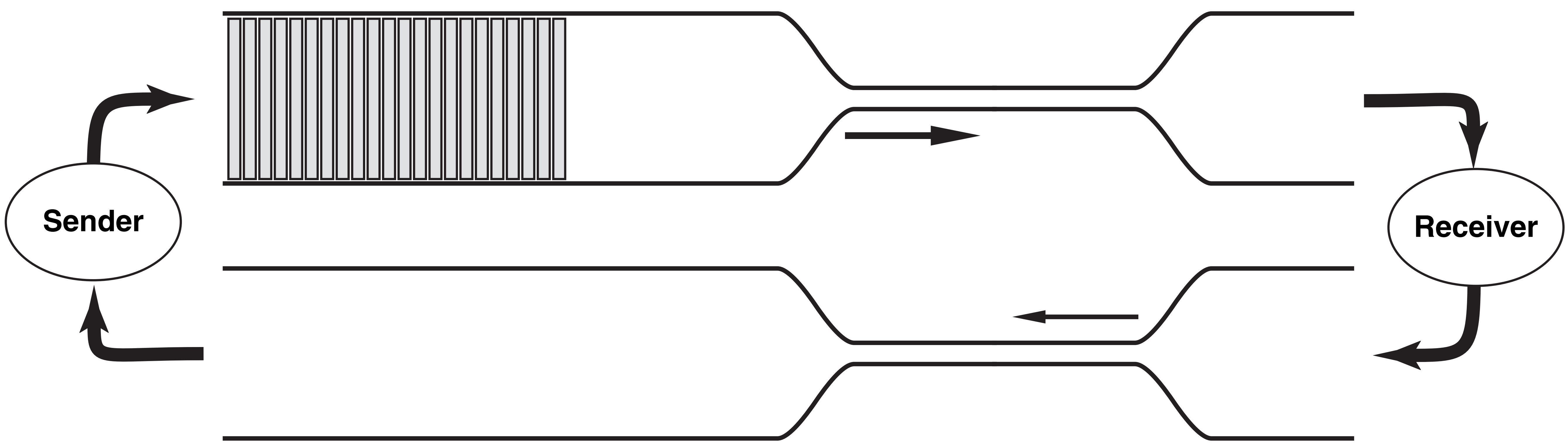
- This means we often have to connect a fire hose to a soda straw.



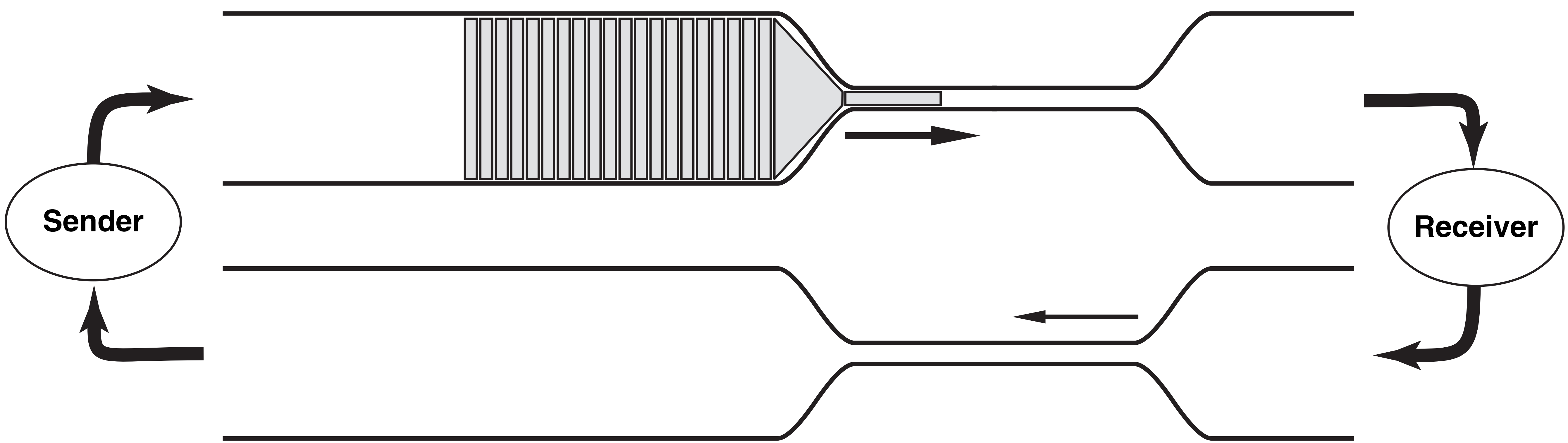
- Unlike the phone system, the Internet supports communication over paths with diverse, time varying, bandwidth.
- This means we often have to connect a fire hose to a soda straw.
- This kind of plumbing needs an adapter. The adapter is called a *queue*.

How a real queue works

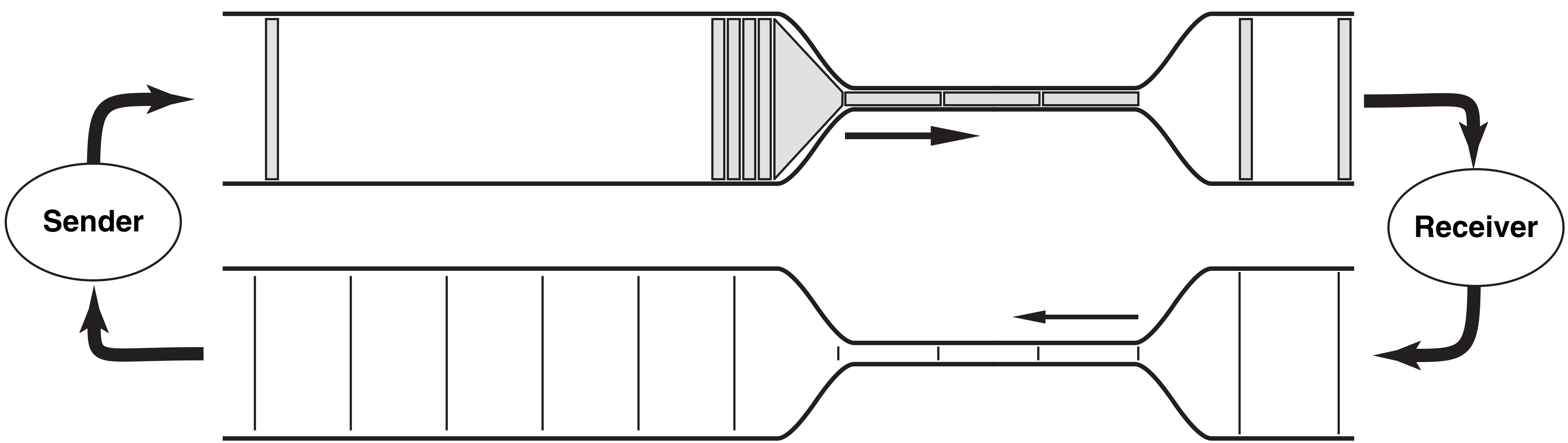
Sender injects a window's worth of packets



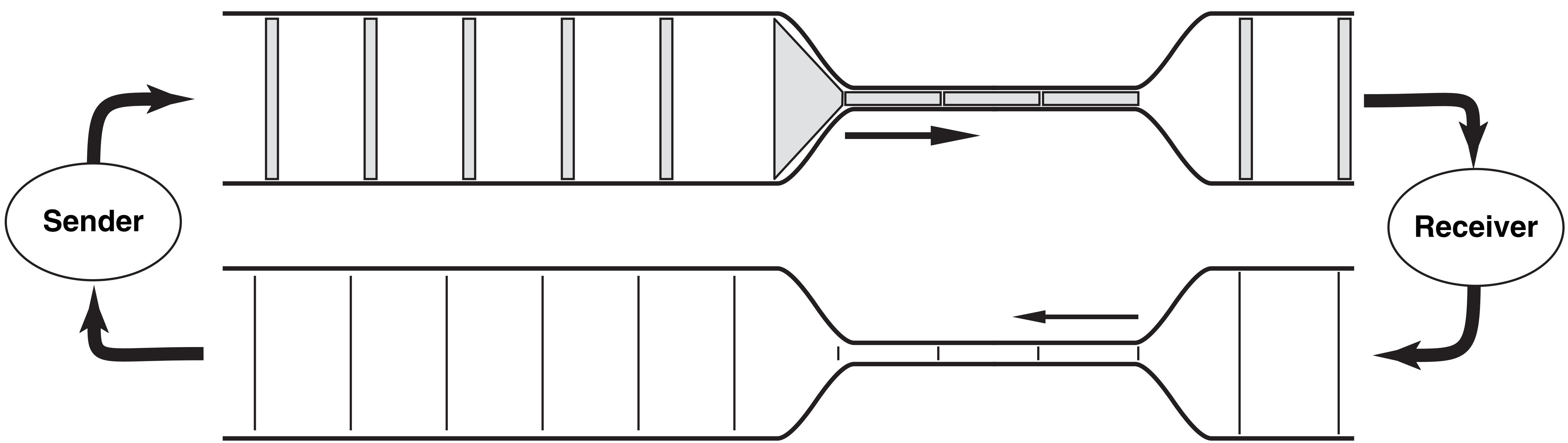
Packets reach high to low bandwidth transition



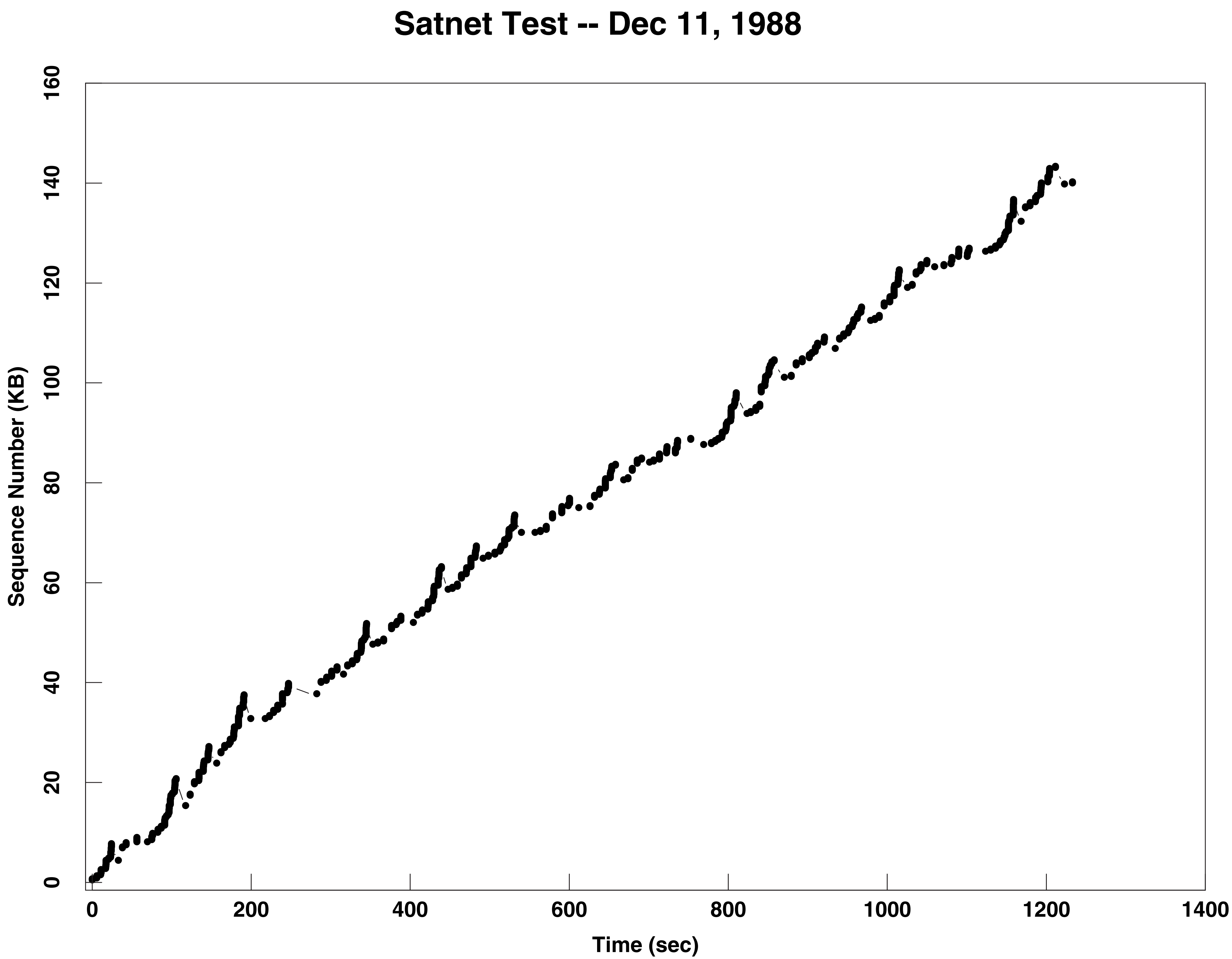
First ack returns and releases next data packet



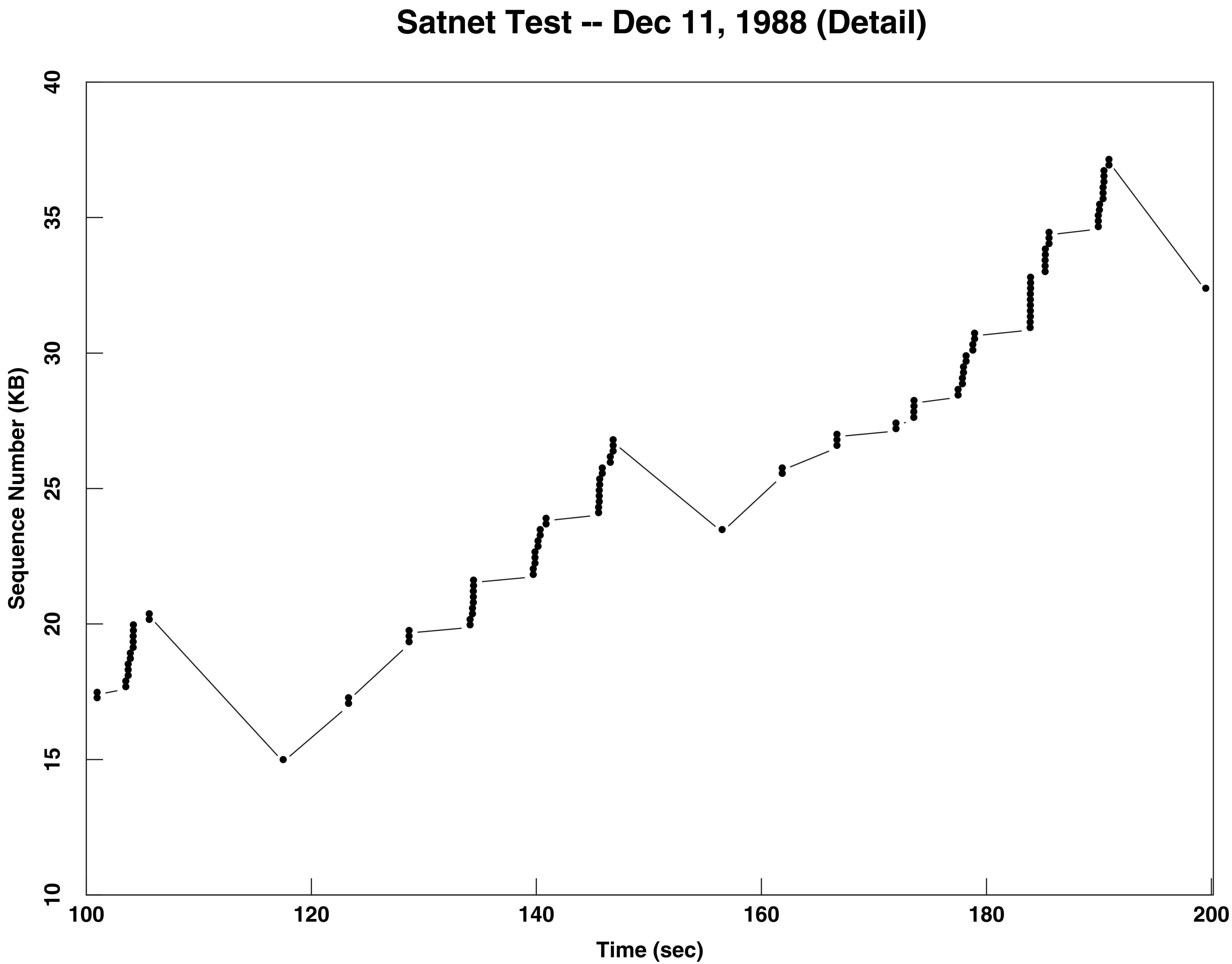
Steady-state reached



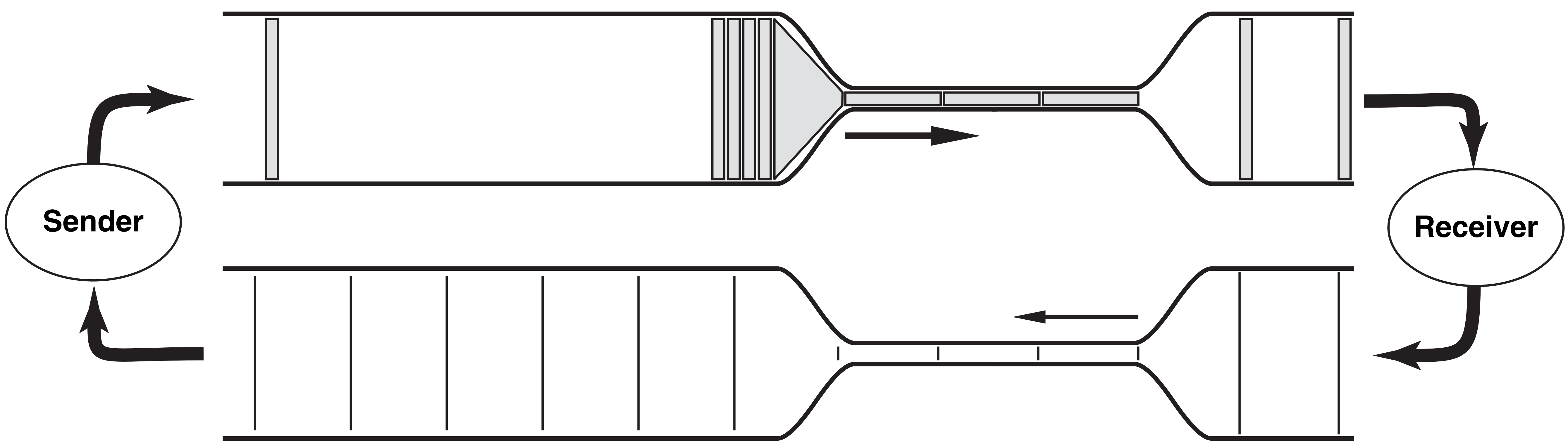
- The amount of data that has to be in transit to run at 100% utilization is the bottleneck bandwidth times the sender-receiver-sender round-trip delay (this is called the *bandwidth*delay product*).
- The bottleneck has to have this much buffer to handle the start up transient.
- What happens if it doesn't?



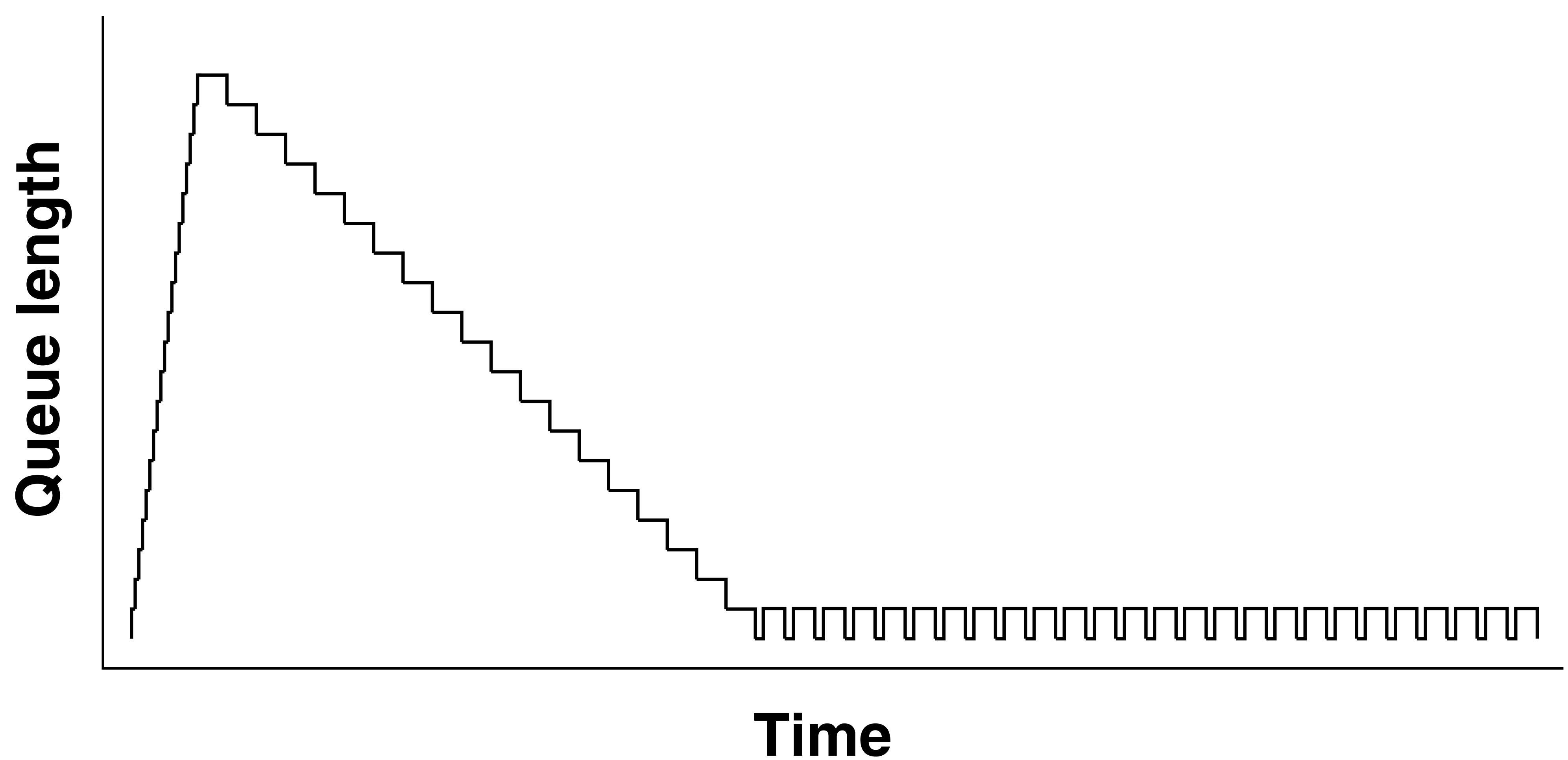
(averaged 100 Bytes/sec on a 8 KByte/sec link)


Satnet Test -- Dec 11, 1988 (Detail)
Sequence Number (KB)
40
35
30
25
20
15
10
100
120
140
160
180
200
Time (sec)

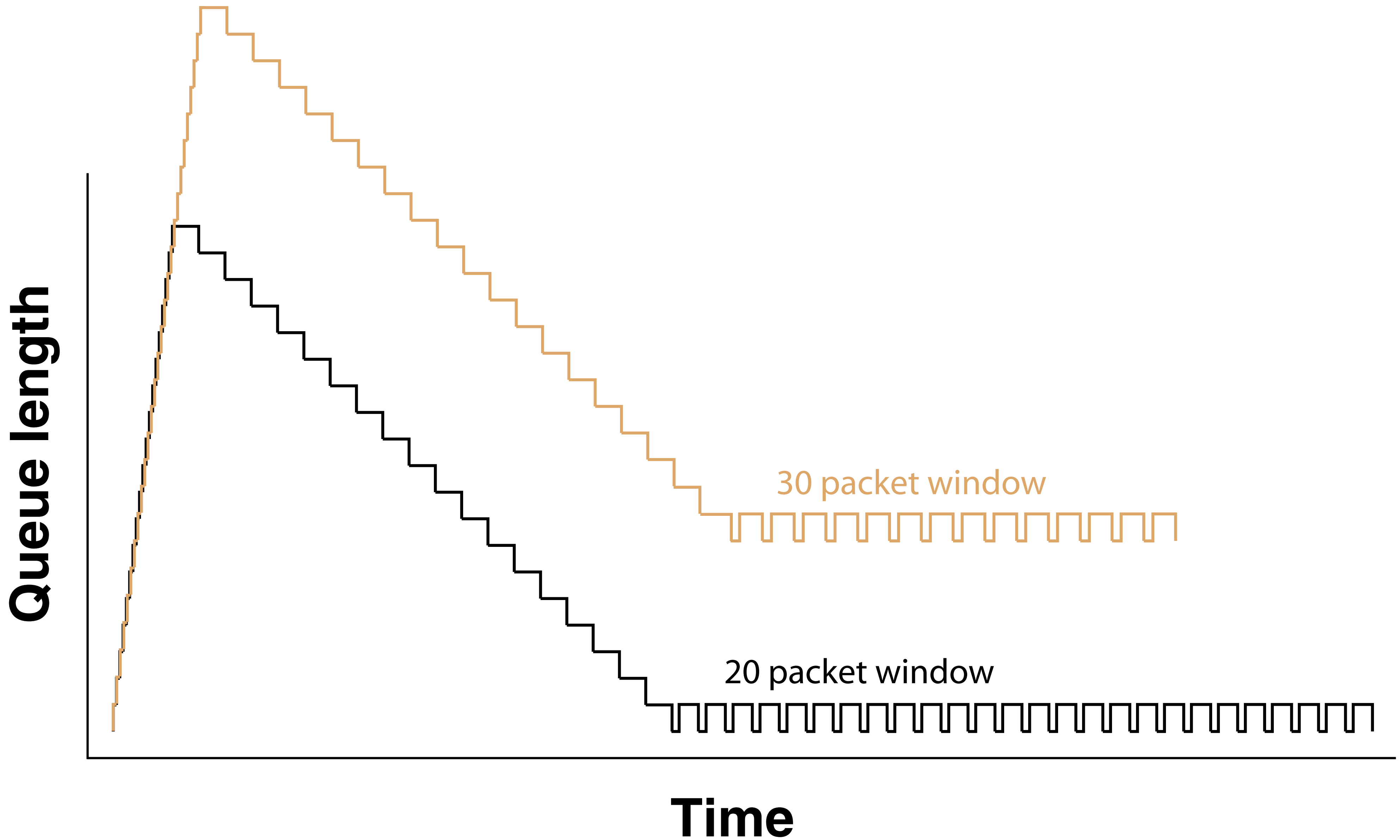
How does the queue behave vs. time?



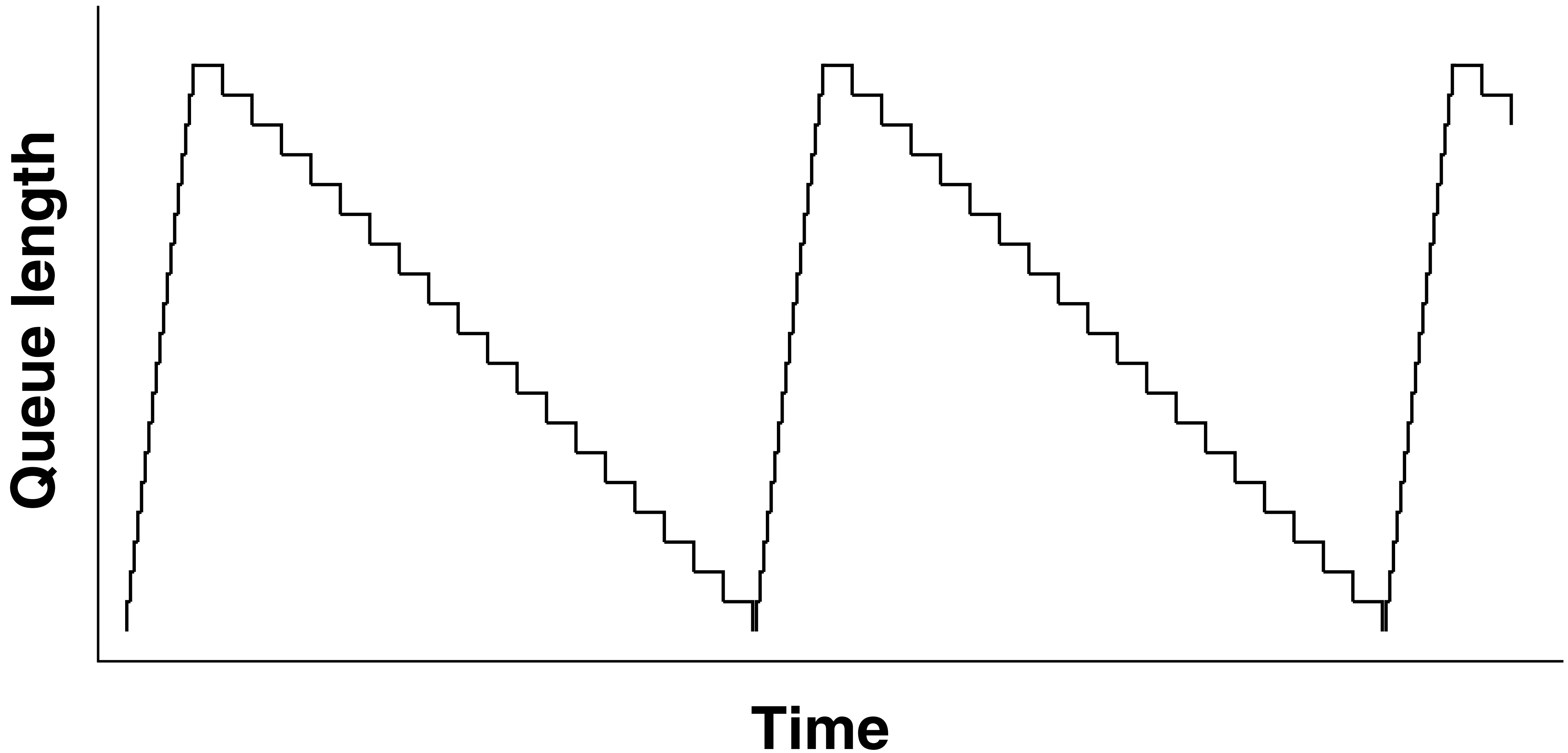
Queue behavior at the fast-to-slow transition



Queue behavior at the fast-to-slow transition



Queue behavior with ack-per-window receiver



Three minor (and completely standard) variations in protocol implementation give three wildly different average queue lengths.

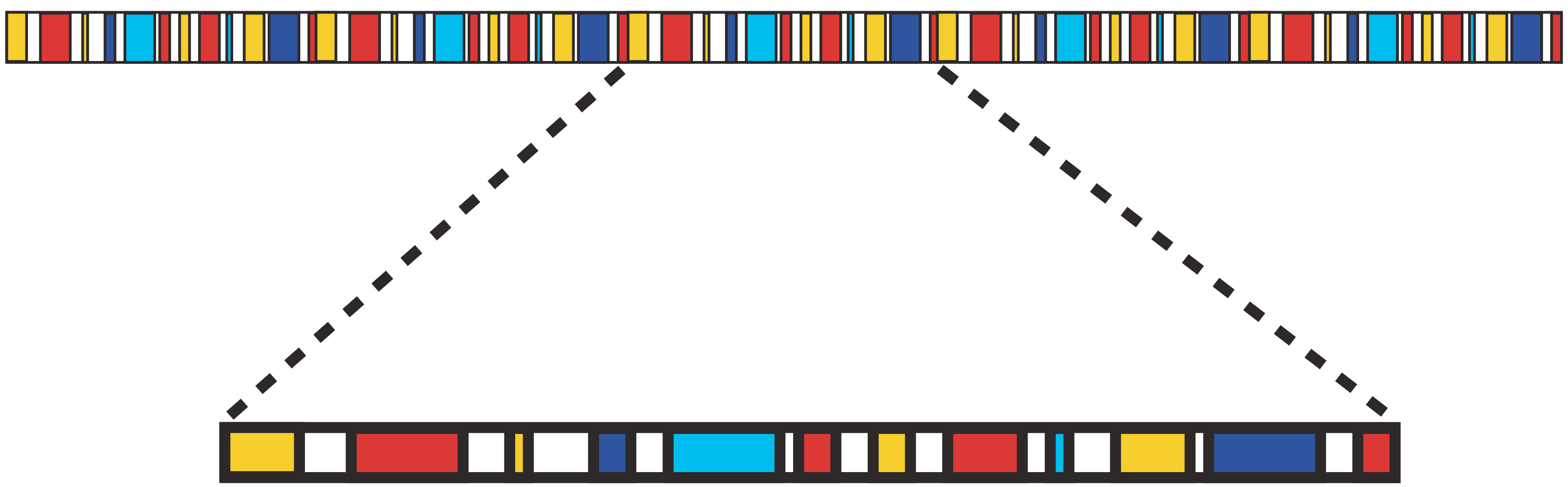
I.e., the average queue length contains *no* information about demand or load.

A mathematical digression ...

$$\int [A(t) - D(t)]$$

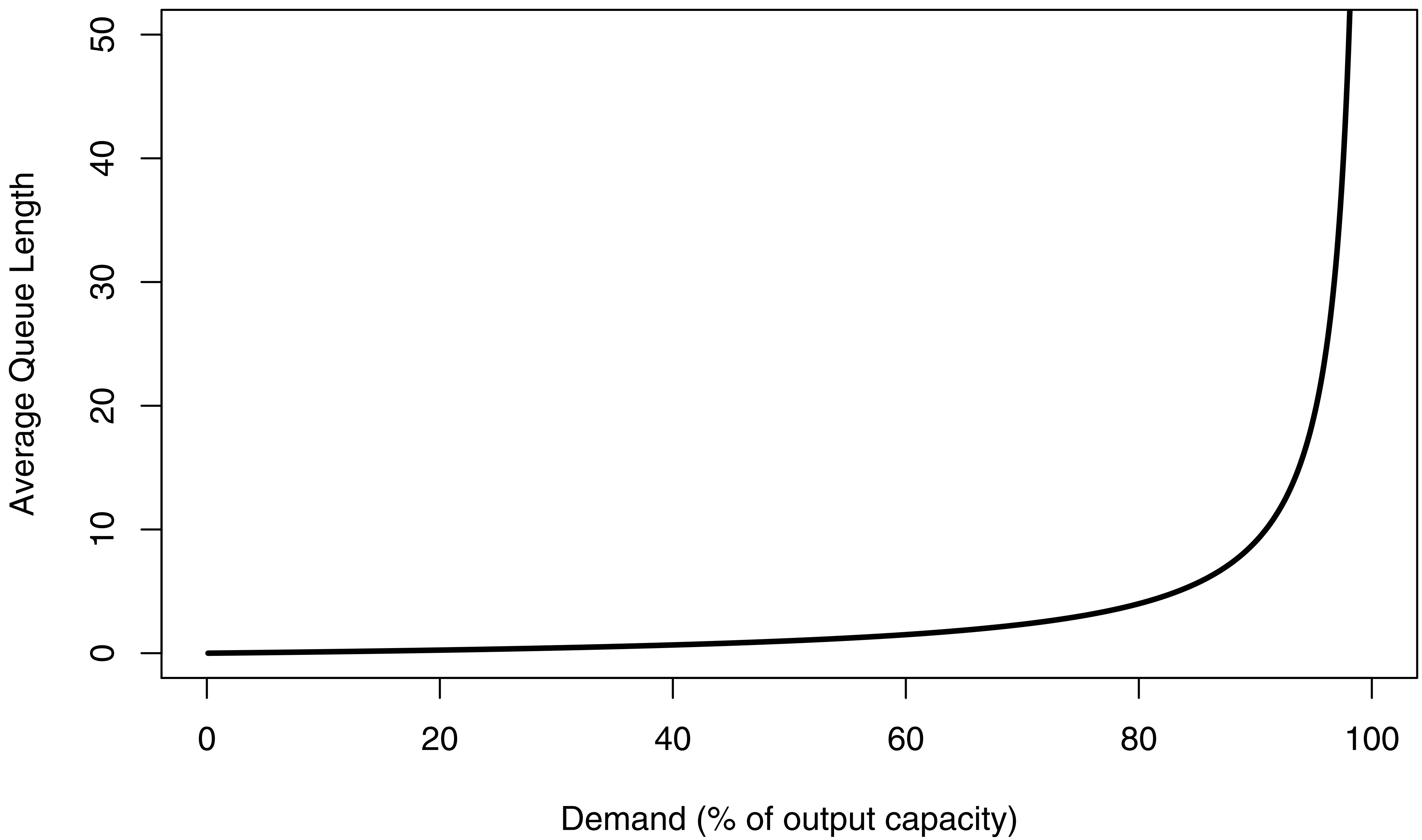
- A queue is the integral of the difference between an arrival process & a departure process.
- For packet network queues, D is usually deterministic and A is some sort of random mixture process.

The Poisson arrival process is beloved by academics the world over

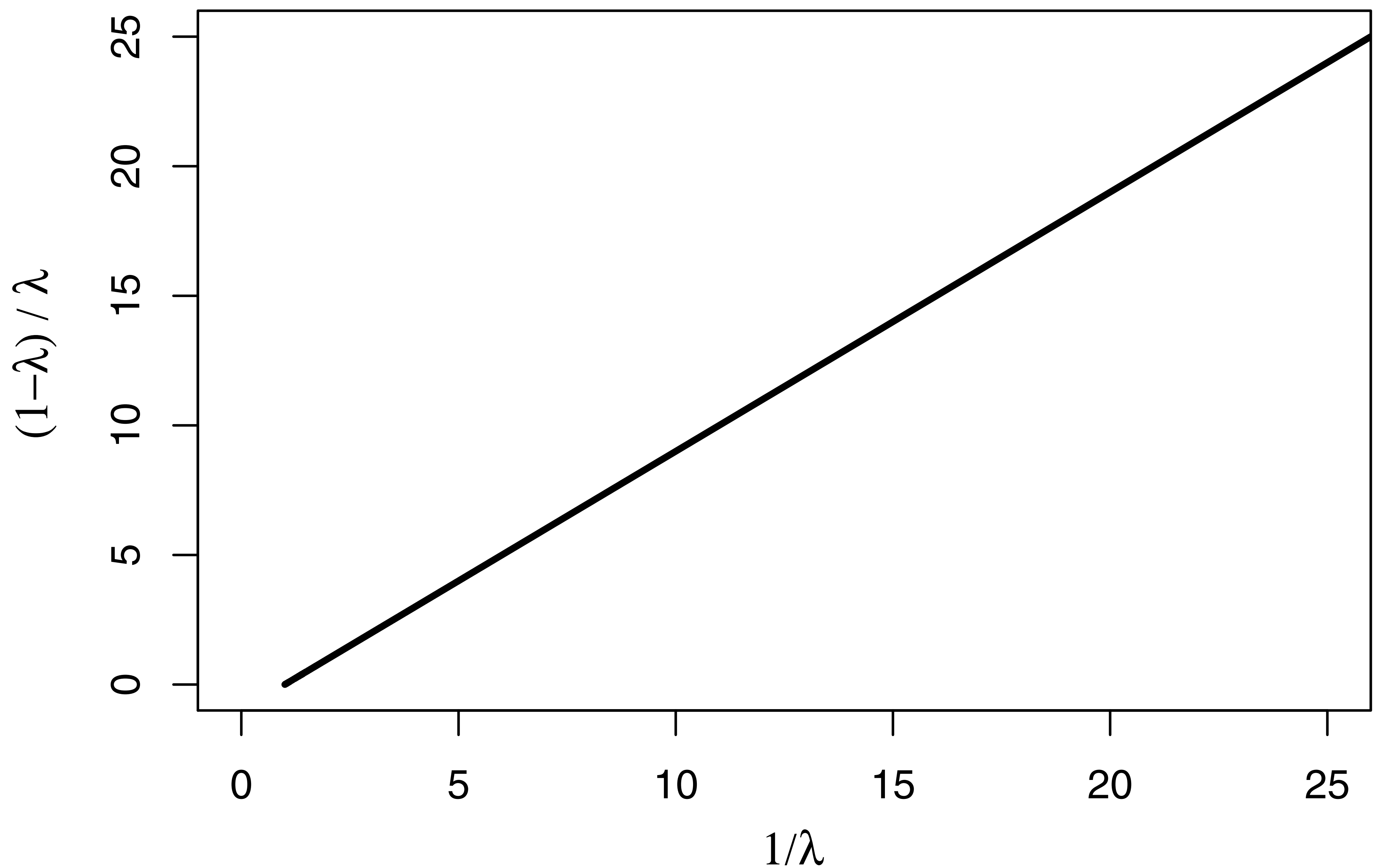


- It describes a low density, IID, uniform random collection of stuff.
- 'Uniform random' means the interarrival time is distributed as $e^{-\lambda t}$.

For a Poisson process, queue length is a function of demand



In fact, inverse queue length is a linear function of 1/λ



For years people have used queue length as a proxy for load in network controls

- In theory this shouldn’t work at all
- In practice it sometimes sort of works because the internet protocols are robust in the face of foolishness.

Theory failures

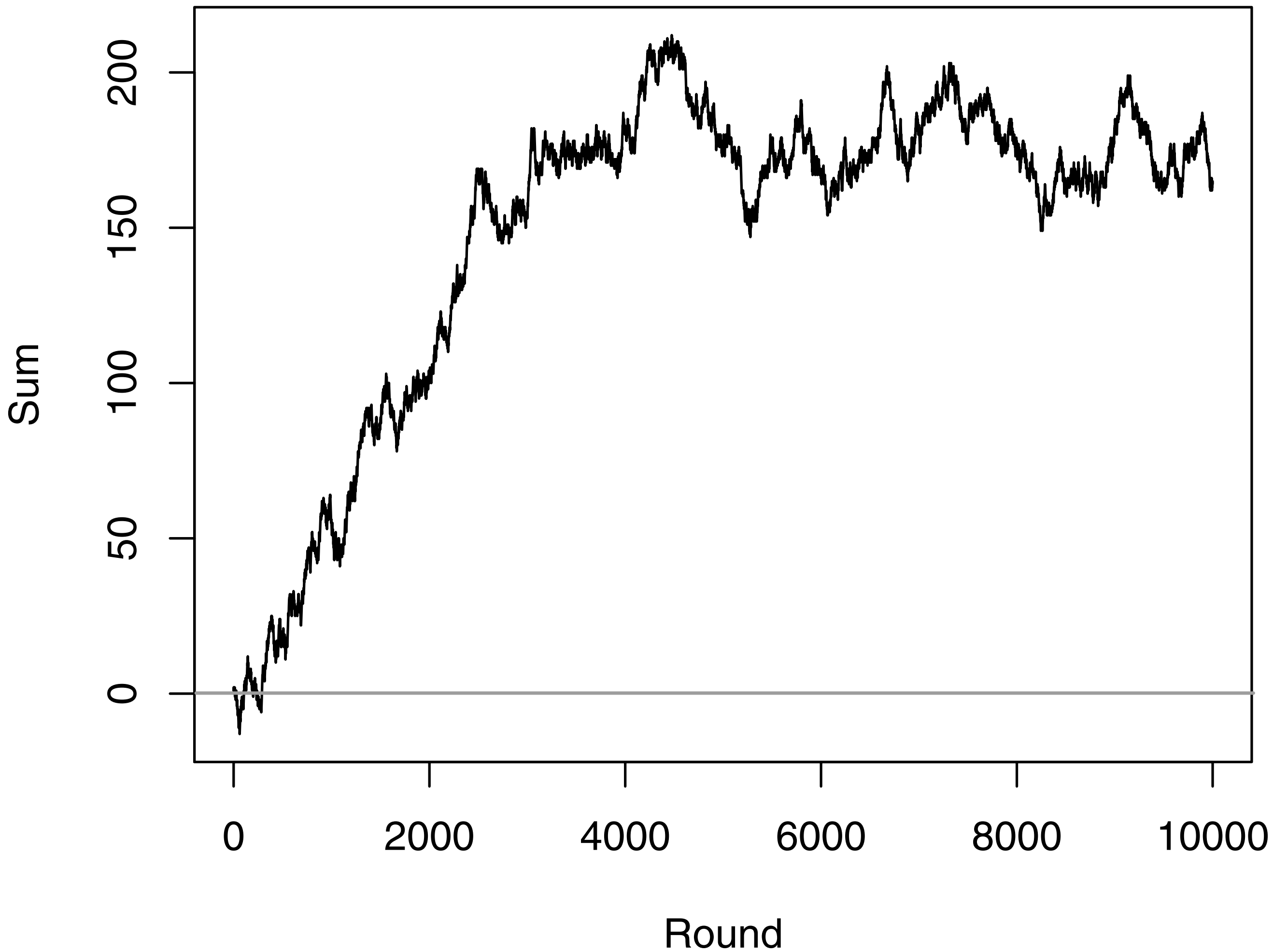
- There can be lots of Poisson traffic in the network but not at a bottleneck.
- Poisson requires *independent*, random uniform traffic but *at a bottleneck* real traffic is highly correlated and not at all uniform.
- The correlations are *intrinsic* since reliability requires a sender-receiver-sender loop and traffic in a loop is *never* poisson.

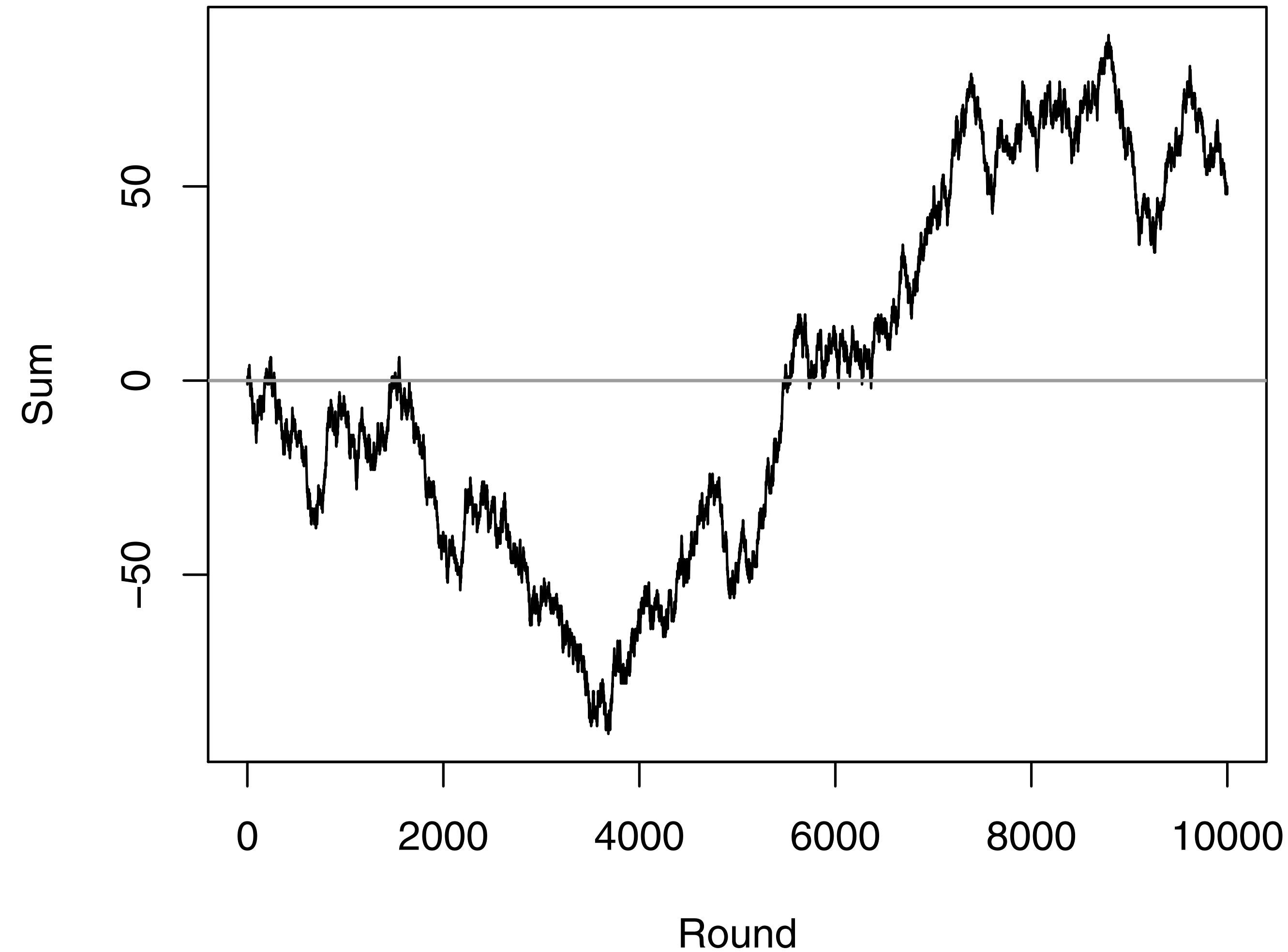
Poisson model isn't even representative

- Queue length is the integral of an arrival rate (with deterministic departures).
- Queue length for a Poisson process is proportional to arrival rate.
- Implies family where $\int F(t) \propto F(t)$
- Exponential (Poisson process) is the *only* member of this family.

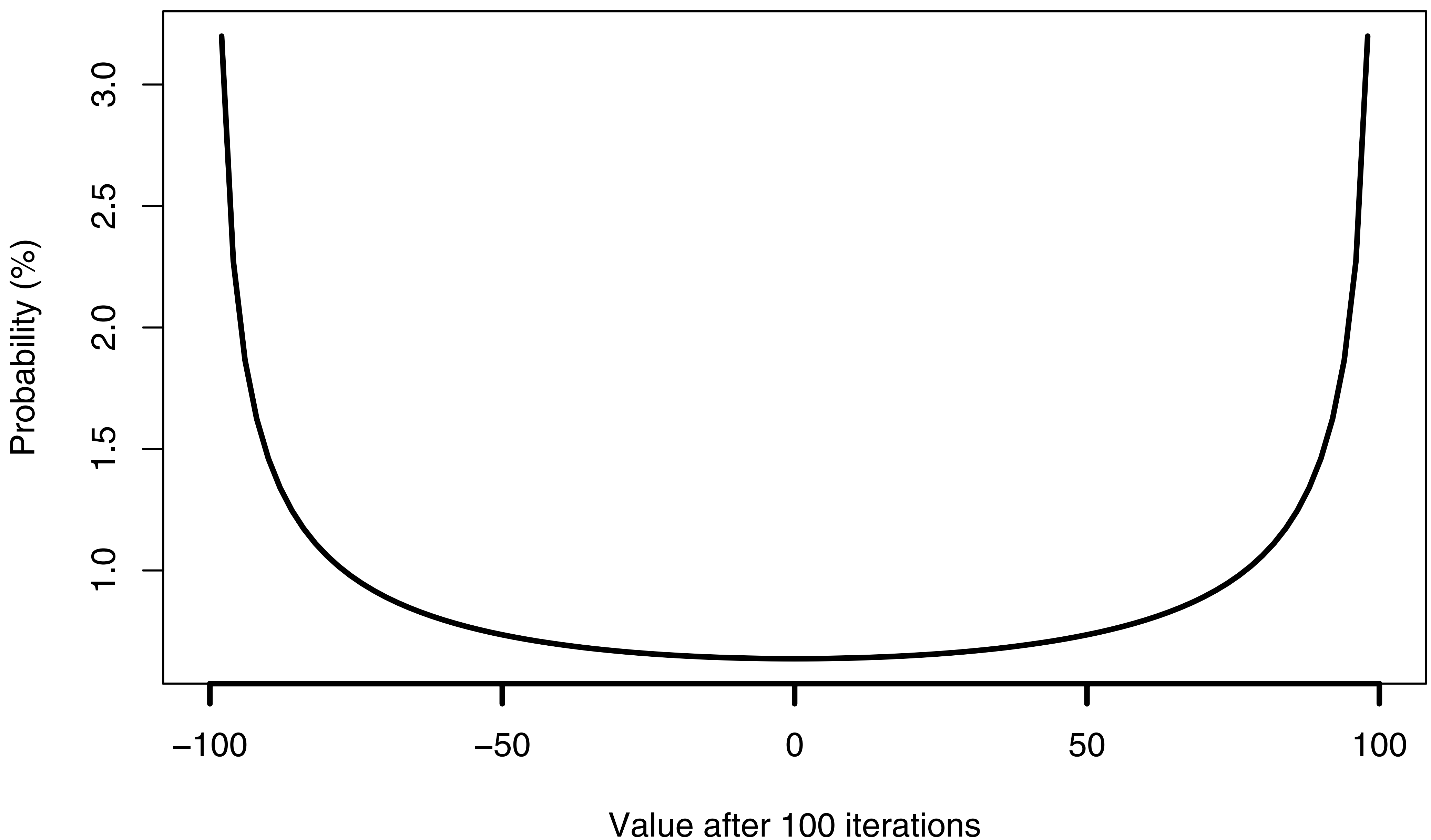
- Poisson models fail because they're memoryless but congested router queues are all about memory.
- A much better mathematical model for real queues is a random process with memory. I.e., a *random walk* or *Brownian motion*.
- Its behavior is the polar opposite of poisson (see Feller, vol.1, chap.3).

Two simulation runs of the simplest random walk (Bernoulli trials)



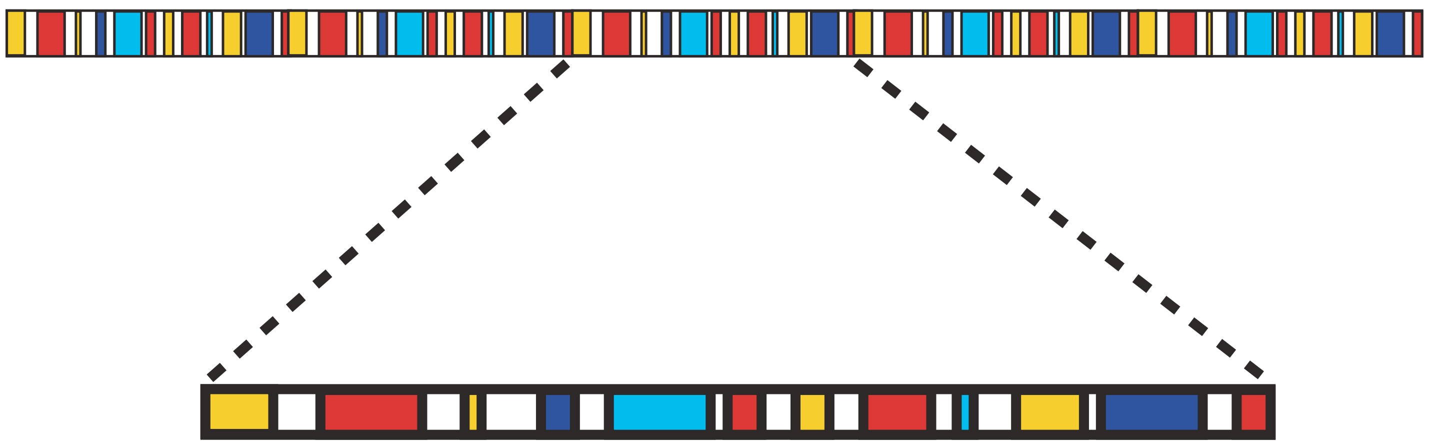


Probability density distribution of the simplest random walk

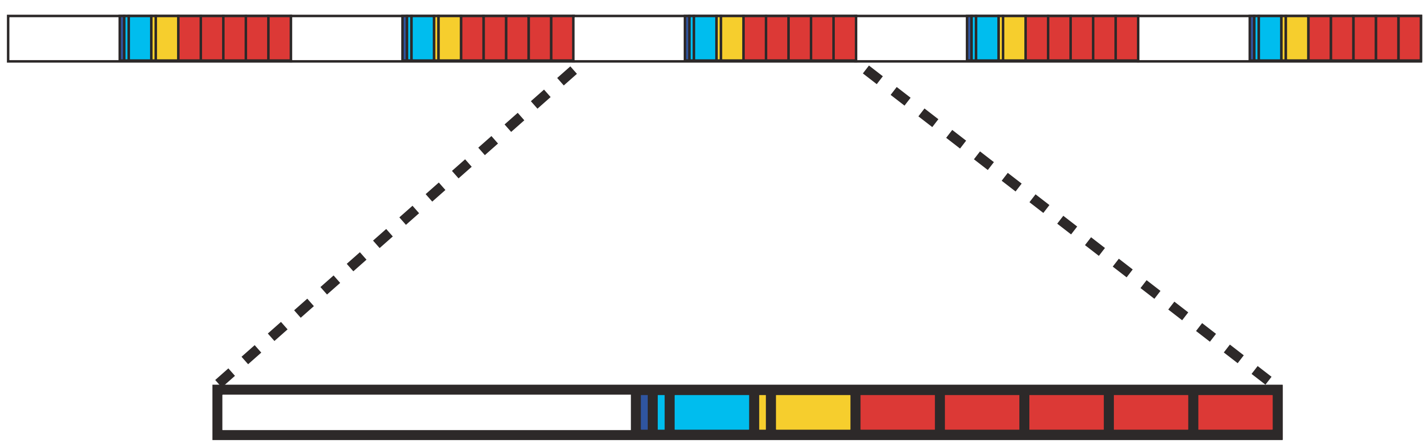


And now back to packets ...

CS profs try to instill intuition based on this picture of traffic:



Measured traffic usually looks more like:

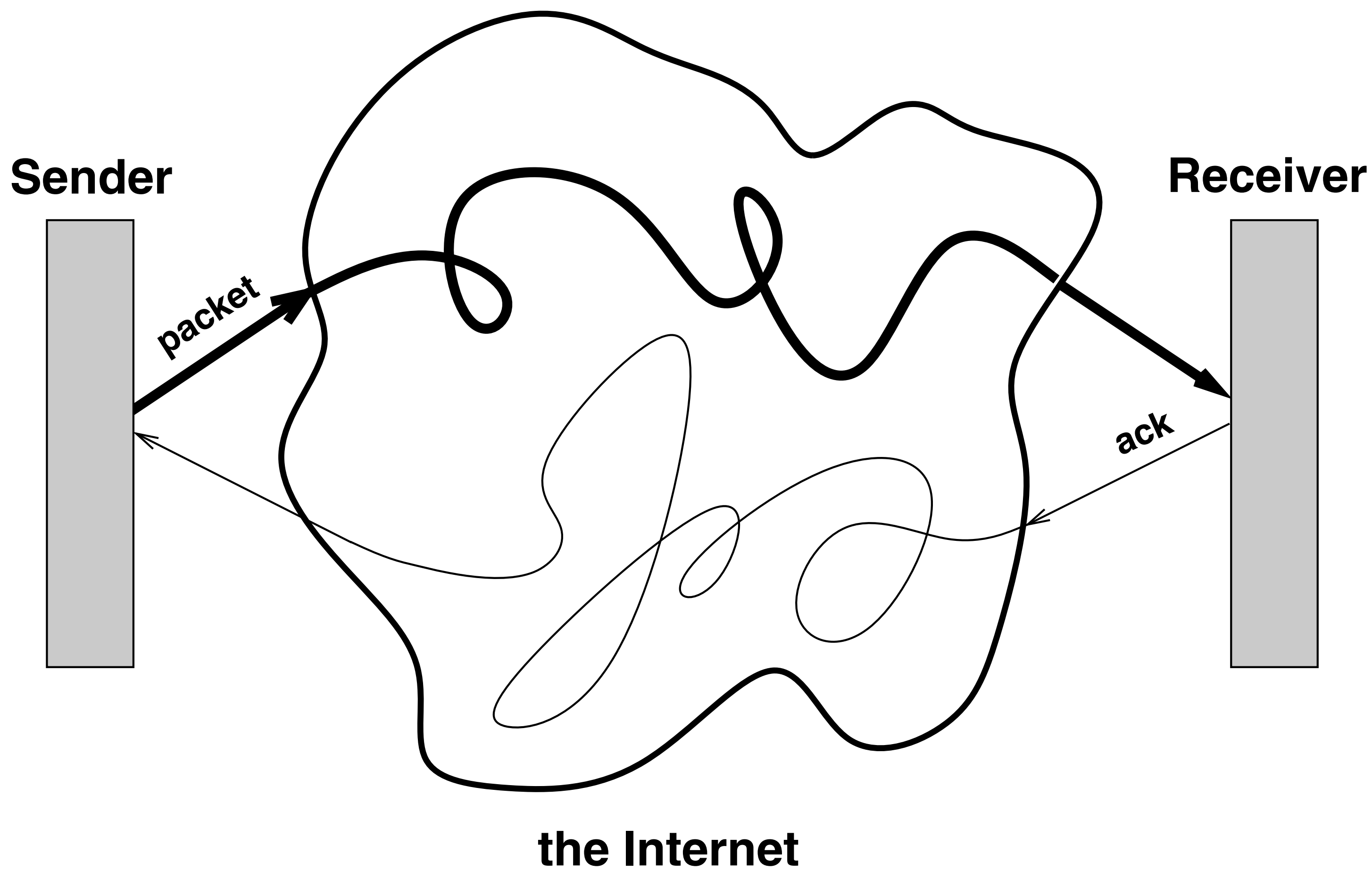


Where does all the structure come from?

- Part of the structure comes from the stability of high rate, bursty arrivals on the upstream side of the queue -- one conversation is unlikely to insert a packet into another's burst.

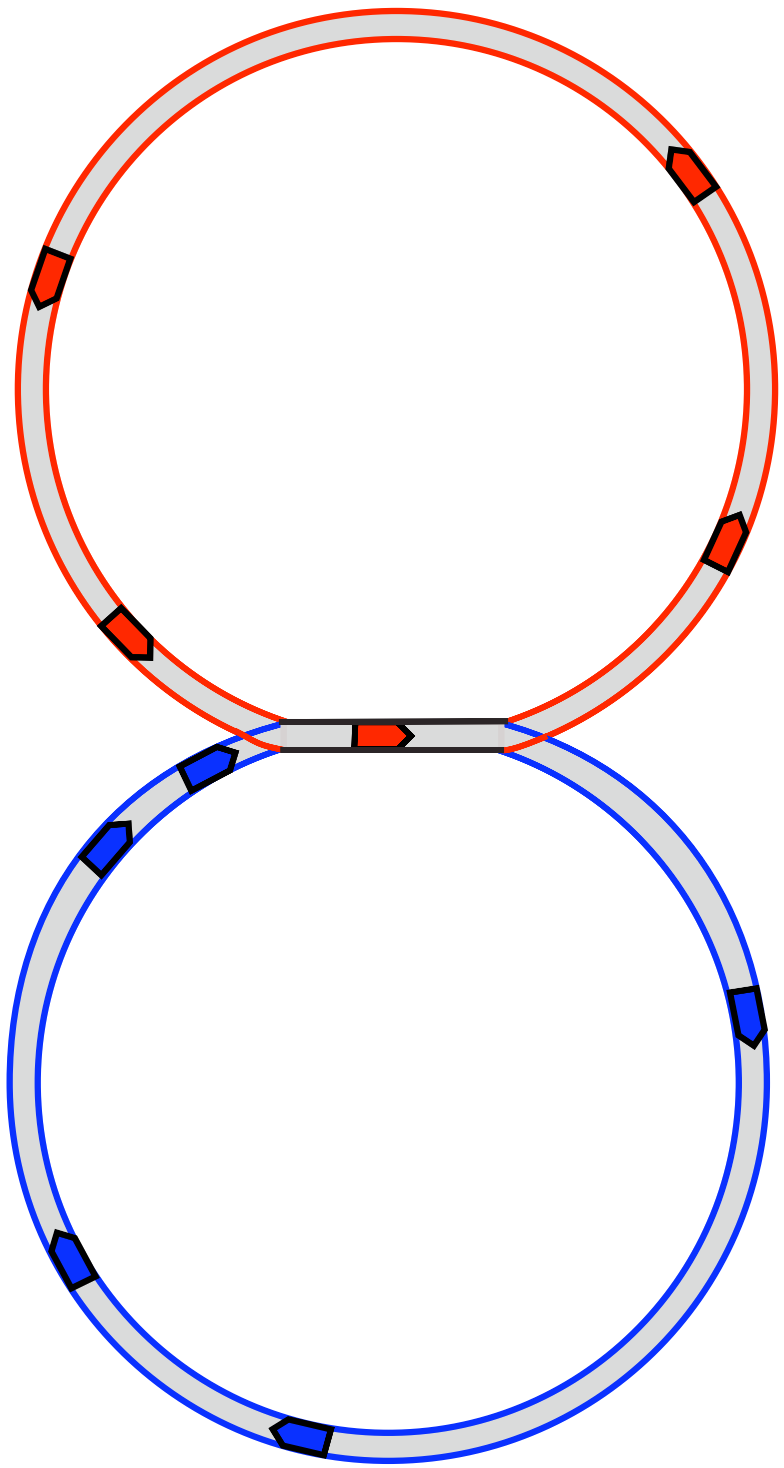
- This tendency to preserve bursts is then driven by some of the non-linear dynamics of the net ...

The Internet at 50,000 feet



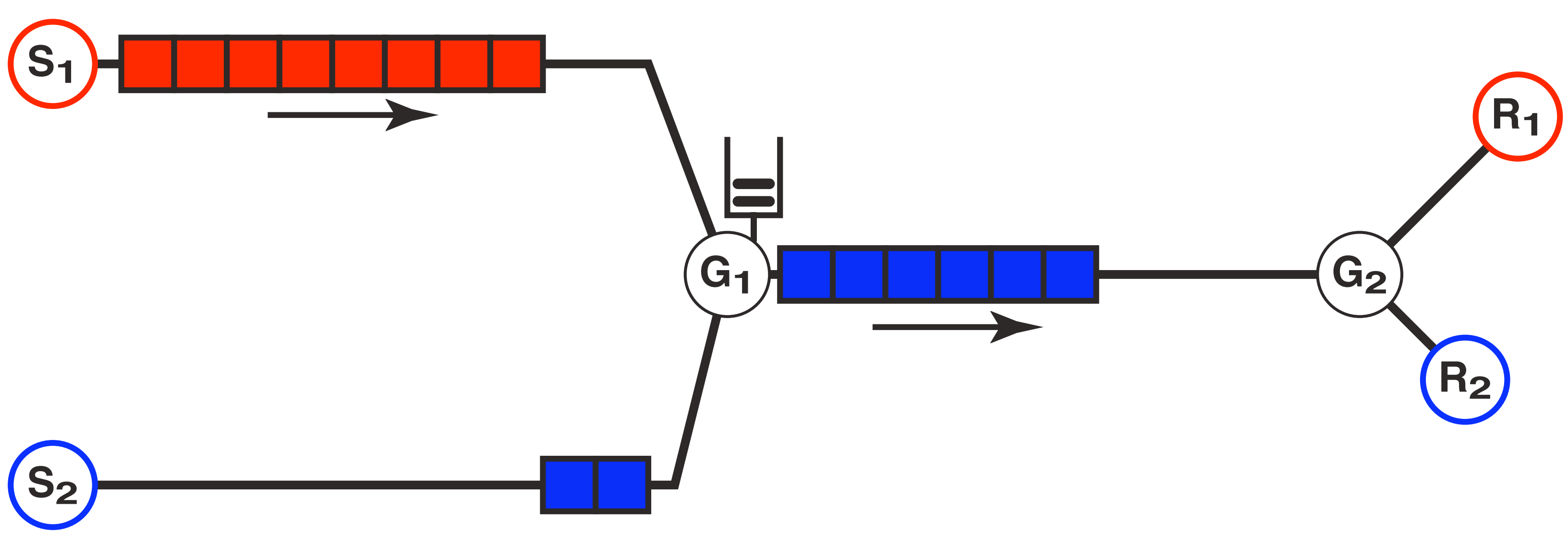
The ends are just mirrors, all the dynamics result from the (usually non-linear) behavior of the net.

How shared links make bursts



- Picture two conversations sharing a congested gateway as two separate train tracks with one common section.
- When a blue train waits for red trains to go through the shared section, the blue trains behind it catch up (get more clumped)
- If the merge rules are efficient (service each color to exhaustion), the system clumps exponentially fast.

- Clumping creates a ‘horizon problem’ that tends to bias traffic managers.
- For satellite systems, the traffic time structure can interact with superframe epoch structure in unfortunate ways (e.g., ACTS hopping beams).
- Increasing delay or epoch length makes things worse (windows increase to compensate).



Suggestions

- Queue length is meaningless (but long term min can be useful).
- Try have at least a bandwidth*delay of buffer.
- Don’t let it stay full.
- Try for a ‘flow-thru’ architecture to minimize packet time-structure disturbance

Suggestions (cont.)

- *Never* introduce additional delay (apps will just try to fill it with packets):
  - Let apps do their own FEC;
    avoid link layer Reed-Solomon and ARQ.
  - Use smooth, simple downlink schedulers
  - Use predictive and anticipatory uplink schedulers