What congestion control?

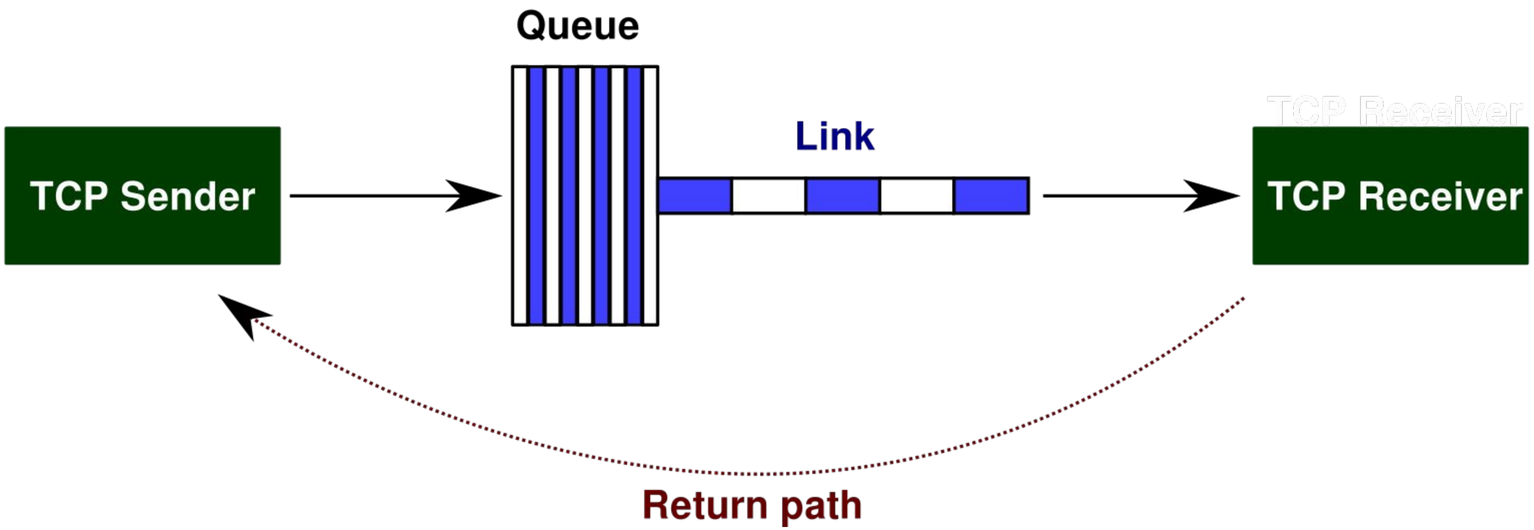
What's the *right* window?

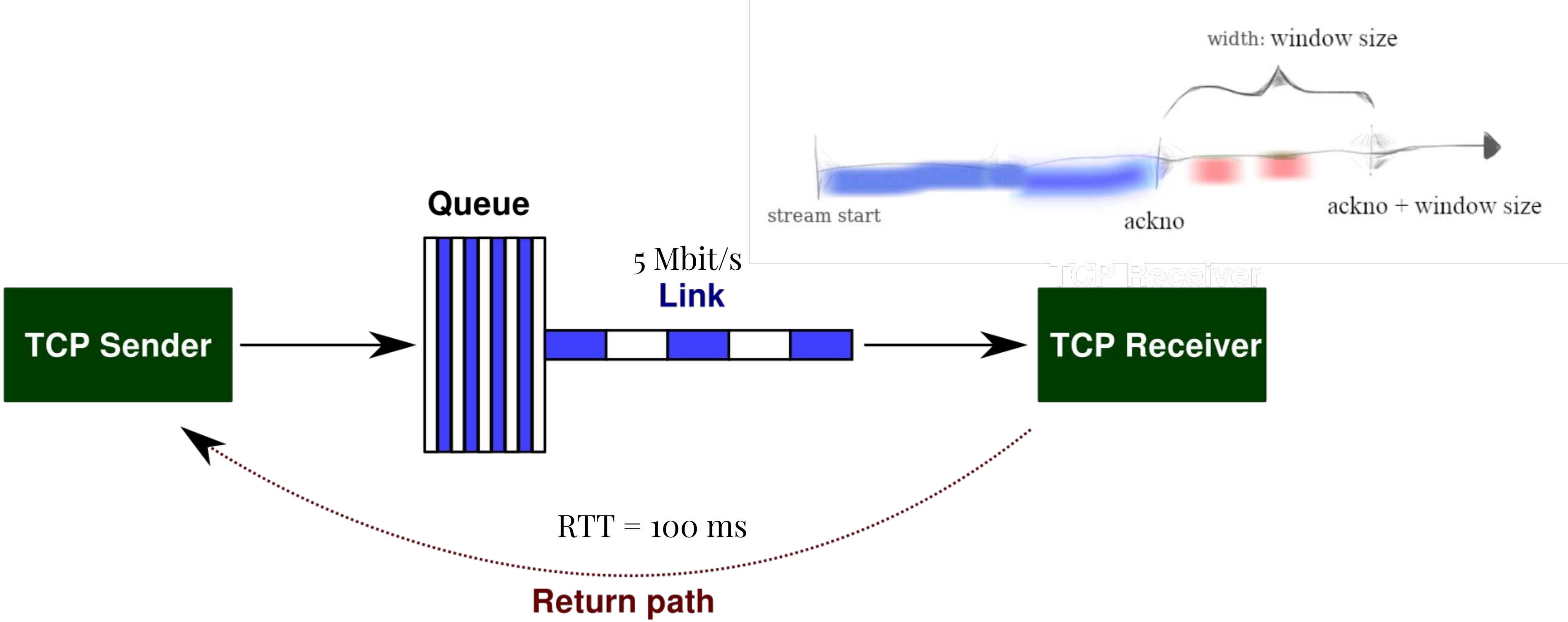
TCP and flow control

- TCP provides a **flow-controlled** bidirectional byte stream
- “**Flow-controlled**”: sender respects **receiver’s** capacity
- But… what about the **network’s** capacity?

From sender’s perspective, three places packets can be

1. In the bottleneck queue
2. In transit on the link
3. At receiver, with acknowledgment in transmit back to sender




width: window size
stream start
ackno
ackno + window size
Queue
5 Mbit/s
Link
TCP Sender
TCP Receiver
RTT = 100 ms
Return path

One way to control congestion: a **second** window

- Sender respects **two** windows. Tighter one controls:
  - receiver's window *(advertised from receiver to sender)*
  - "congestion window" cwnd *(maintained by sender)*
- The congestion window caps # of bytes in flight, same as receiver window.
- When one more byte is acked (or judged lost), one more byte can be sent. This is called "self-clocking."

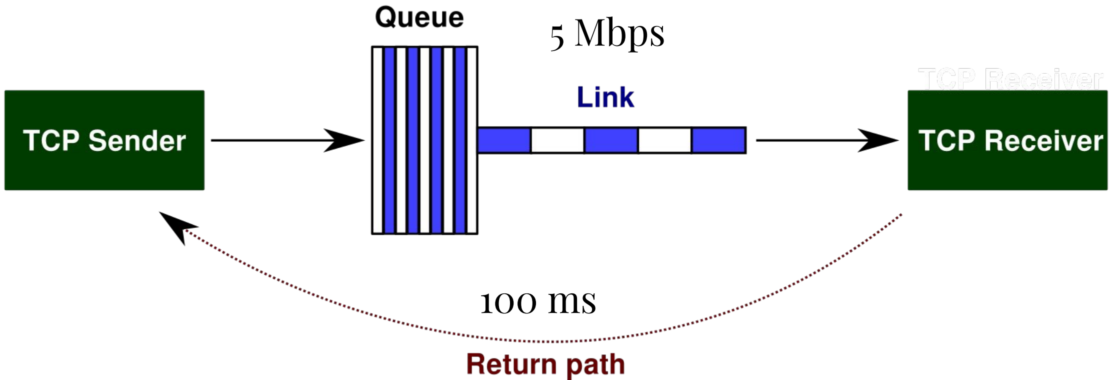
Q: Why not cap "rate" instead of "window"?
A: Self-clocking is powerful! What happens if either is off by 1%?

How much data can be “on the link” at any moment?

1. How fast can link send data?
2. How long until data is acknowledged (without queuing)?

(5 Mbit/s) x (100 ms) = **62.5 kilobytes**

This is called the “bandwidth delay product.”



What is the **right** congestion window?

- Ideal **total** number of bytes outstanding = bandwidth x delay product (**BDP**).
    - Keeps the link always busy, with nothing in the bottleneck queue.
- With one flow, BDP is **ideal window** for that flow.
  (N flows: each flow could use cwnd = BDP/N)
- "No loss" window: anything less than BDP + **max queue size**.

But… values for “ideal” cwnd are unknown at runtime!

TCP sender *doesn’t know*:

- bottleneck link rate
- minimum RTT (without queueing)
- number of other flows contending for the same bottleneck

So… how to approximate the “right” congestion window without omniscience?

Tune in next lecture!