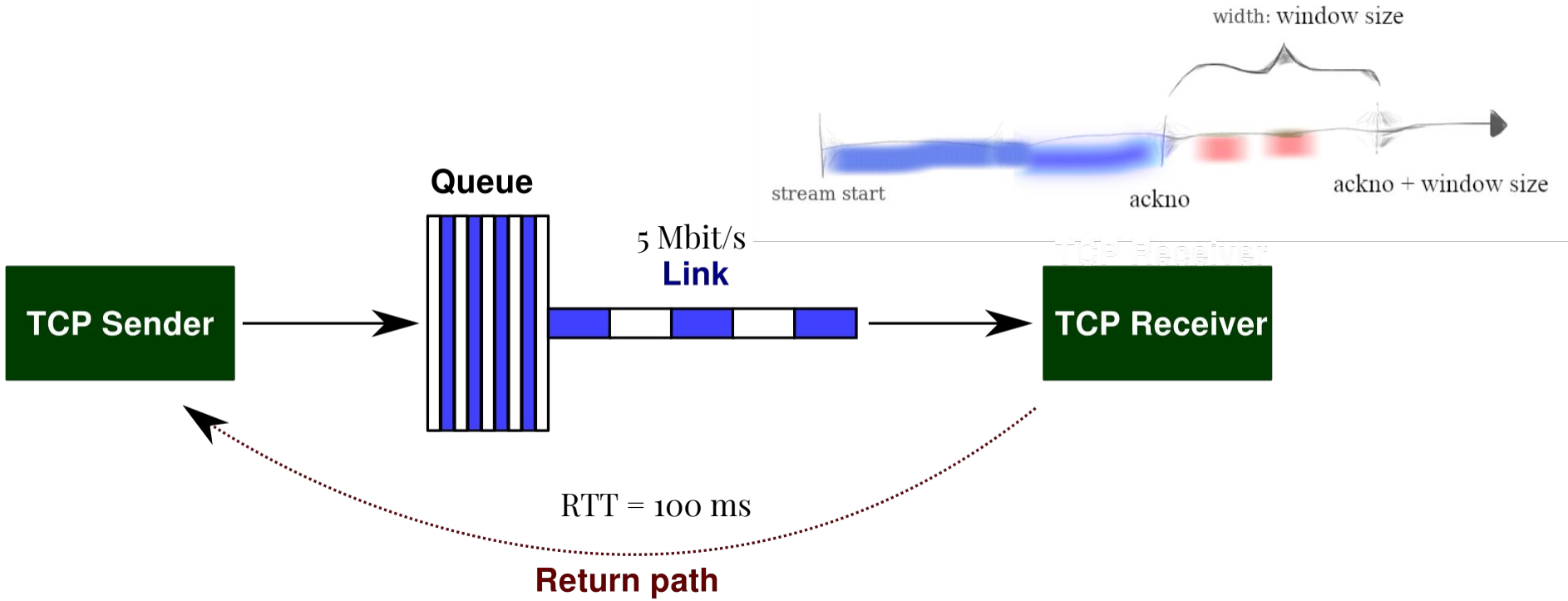
How congestion control?

***How to find* the right window?**


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

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?

One possibility: increase on success, decrease on loss

- Start with cwnd at small value (e.g. 3 segments)
- On **success** (segment fully acknowledged), increase by 1 segment per RTT
  - On each byte acknowledged: cwnd += (segment size)/cwnd
- On **loss**, assume congestion. Cut cwnd in half!
  - Loss inferred when:
    - segment was sent a long time ago, still not acknowledged
    - **or** several later-sent segments have been acknowledged

This is called the "Additive Increase, Multiplicative Decrease" algorithm (AIMD).
AI = +1, MD = /2

Q: Why such a big decrease on loss?

One possibility: increase on success, decrease on loss

- Start with cwnd at small value (e.g. 3 segments)
- On **success** (segment fully acknowledged), increase by 1 segment per RTT
  - On each byte acknowledged: cwnd += (segment size)/cwnd
- On **loss**, assume congestion. Cut cwnd in half!
  - Loss inferred when:
    - segment was sent a long time ago, still not acknowledged
    - **or** several later-sent segments have been acknowledged

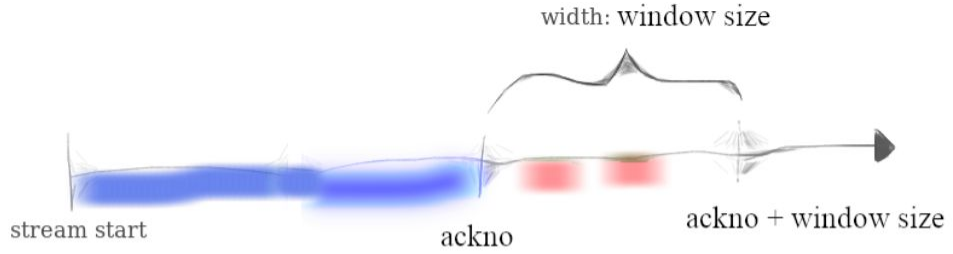
This is called the “Additive Increase, Multiplicative Decrease” algorithm (AIMD).
AI = +1, MD = /2

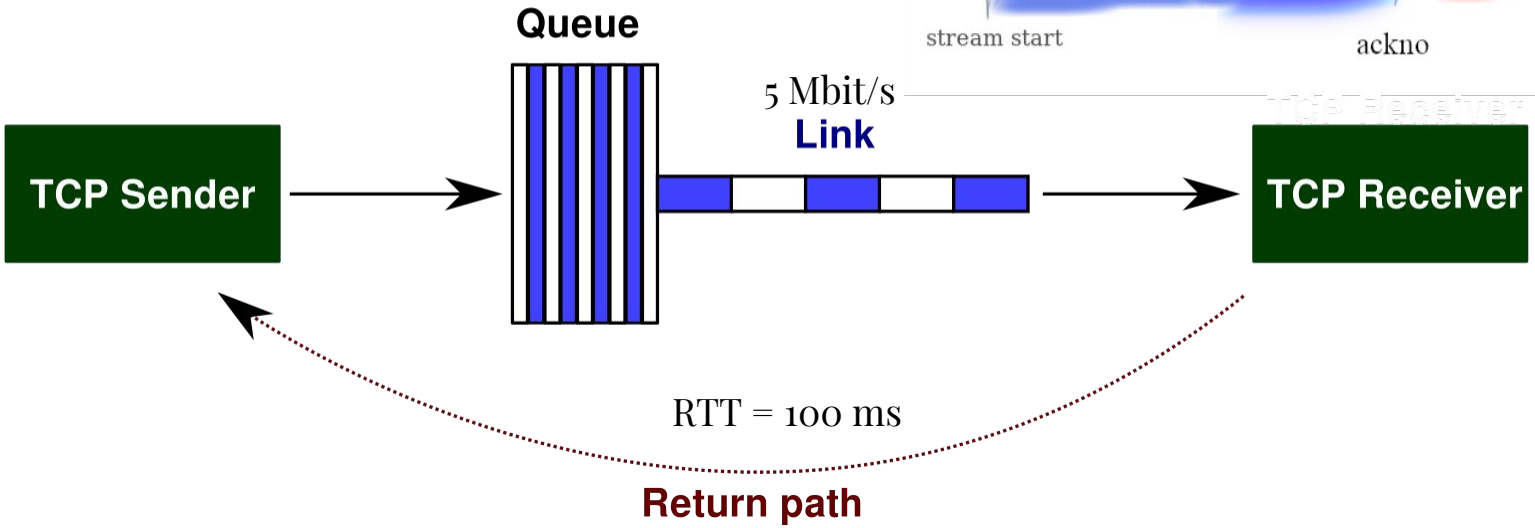
Q: Why such a big decrease on loss?

Slow-start: exponential growth at the beginning!

- It's really slow to start with cwnd=3 segments and grow at 1 segment/RTT
- But it's unacceptable to start with cwnd=receiver window!
- Happy medium is called "slow-start":
  - Start cwnd = 3 segments
  - On **success** (segment fully acknowledged), increase by 1 byte per byte
    - On each byte acknowledged: `cwnd++`
  - On first loss, cut cwnd in half and revert to AIMD.

Q: What is the ideal value for the router's buffer (max queue)?





What about with lots of TCP flows?

Watch online (video 4-4).