Congestion avoidance and control

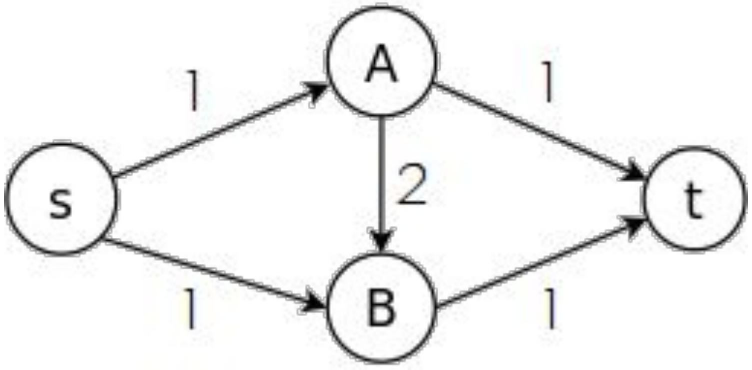
Causes of Congestion

There are only three ways for packet conservation to fail:

1. The connection doesn't get to equilibrium, or

2. A sender injects a new packet before an old packet has exited, or

3. The equilibrium can't be reached because of resource limits along the path.



Desired equilibrium

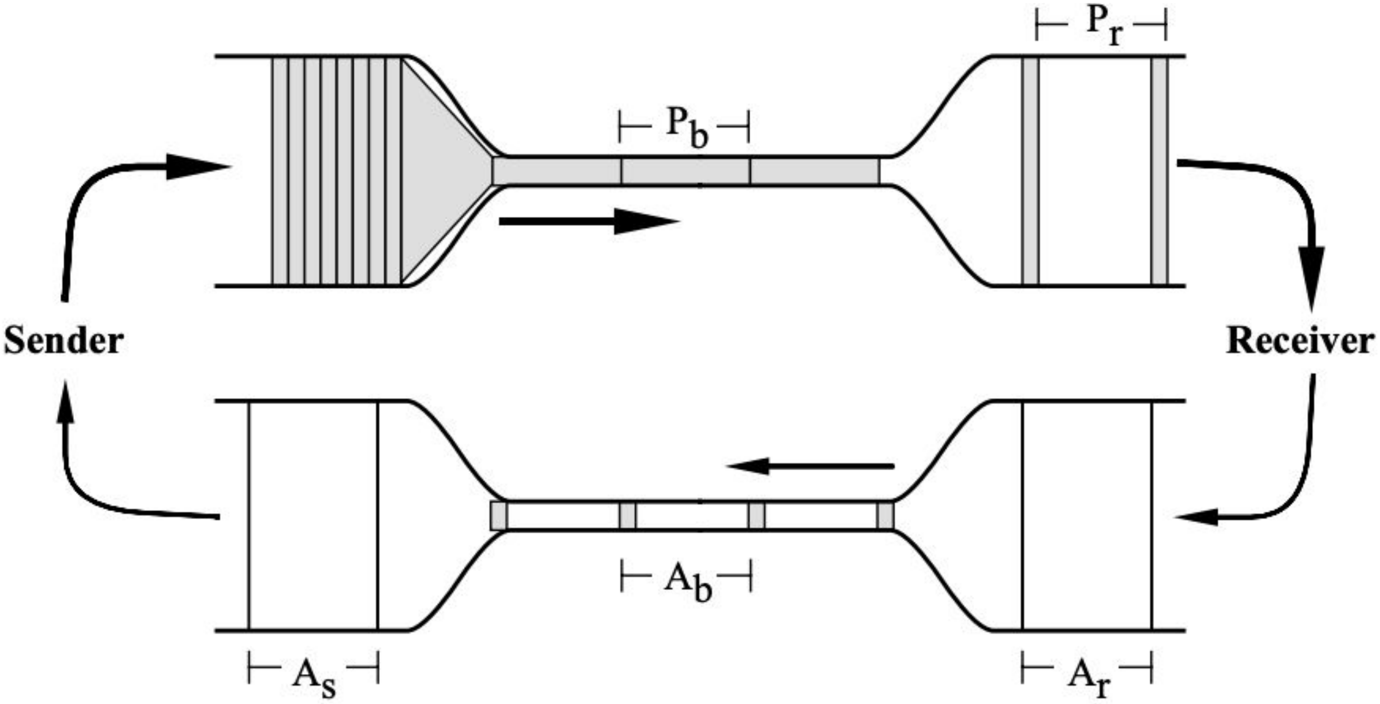
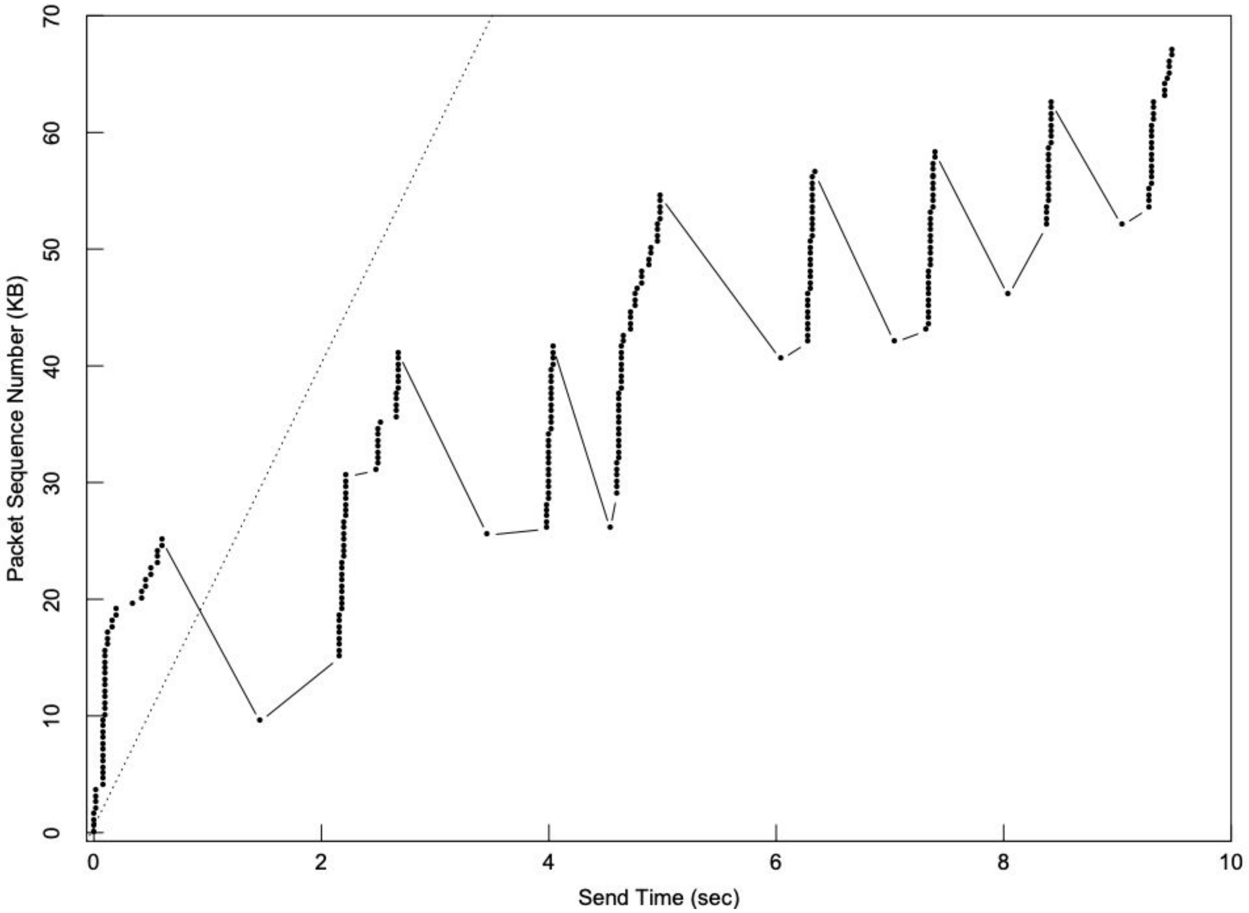


Figure 3: Startup behavior of TCP without Slow-start



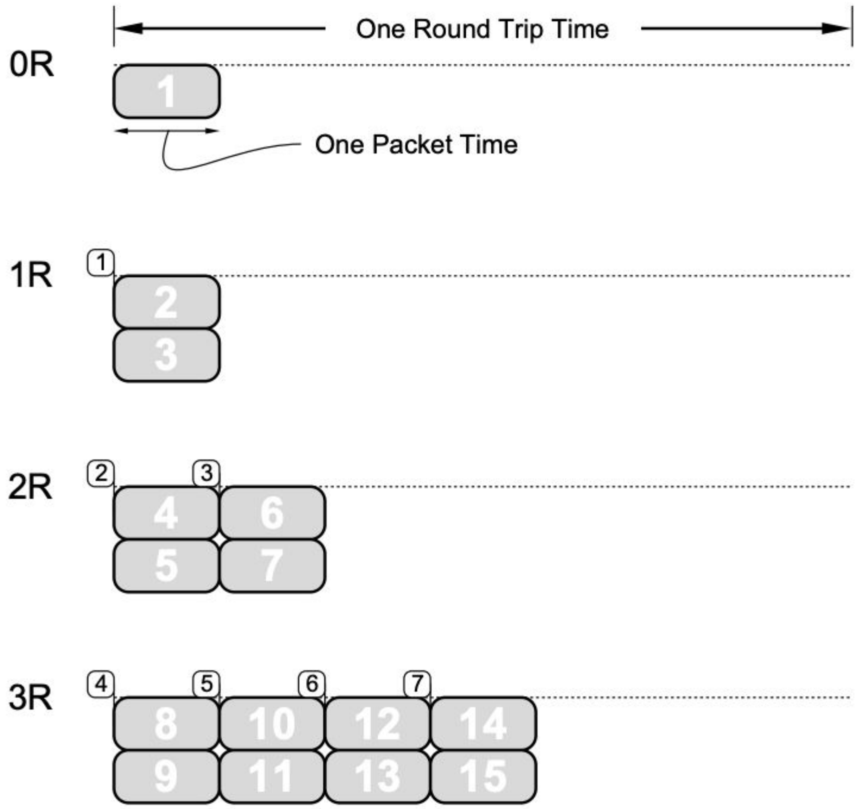
Fix 1: Slow Start

1. cwnd = 1 to start
2. Send cwnd packets
3. Increase cwnd when receiving a packet
4. Stop when you reach limit

Has a doubling effect-

Takes R log(W) time where R is round-trip time and W is window size

Figure 2: The Chronology of a Slow-start



Fix 1: Slow Start

1. cwnd = 1 to start
2. Send cwnd packets
3. Increase cwnd when receiving a packet
4. Stop when you reach limit

Has a doubling effect-

Takes R log(W) time where R is round-trip time and W is window size

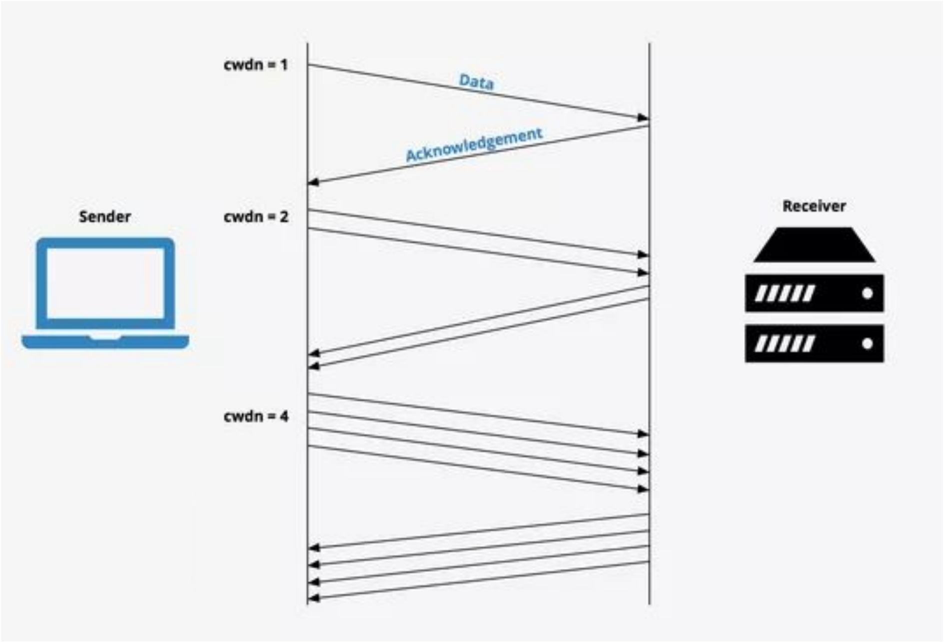
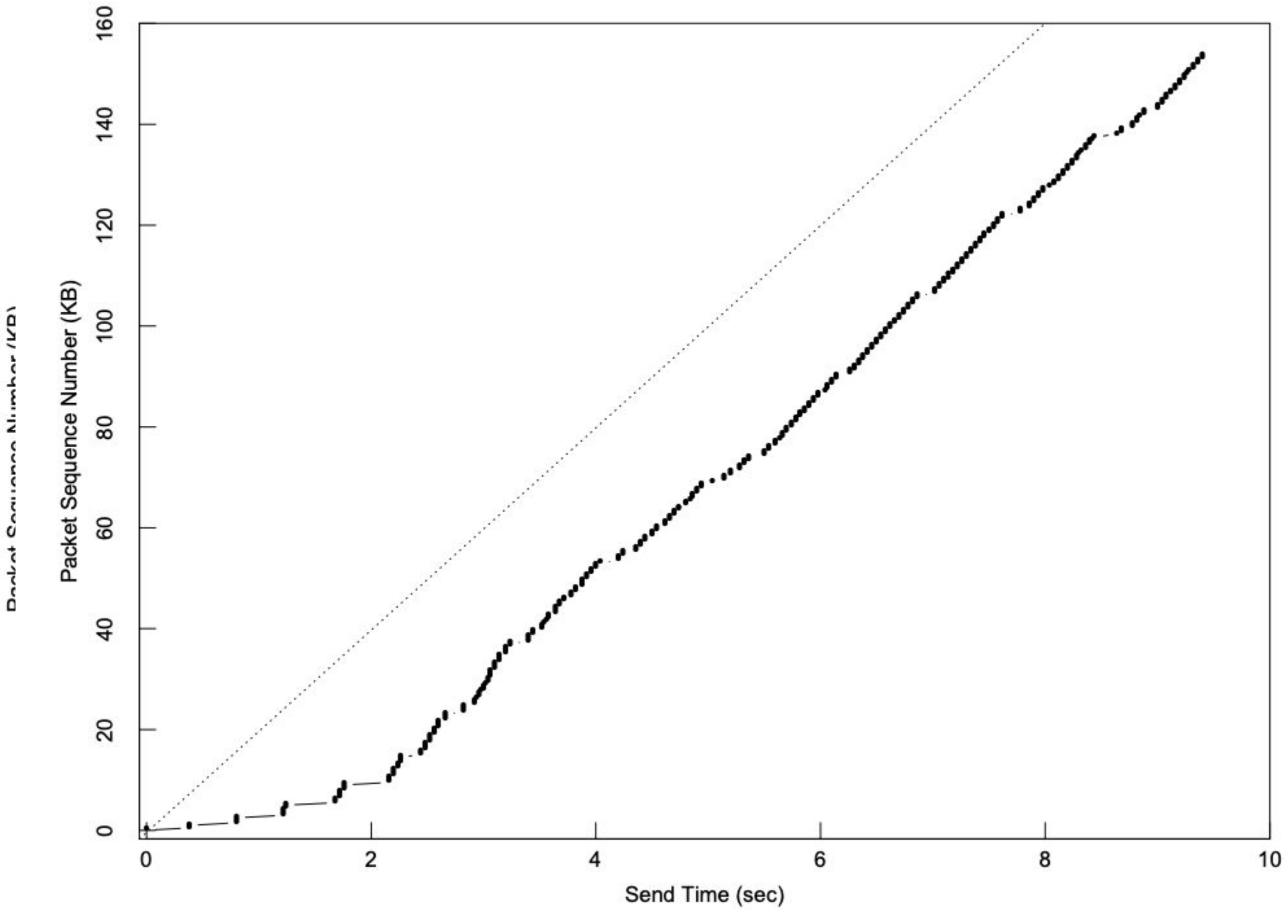


Figure 4: Startup behavior of TCP with Slow-start



Fix 2: Maintaining equilibrium

Goal: keep network running smoothly at equilibrium

Problem: when to retransmit lost packages?

Solution: use a **retransmit timer** for when to retry sending a packet

TCP suggests:

$$R \leftarrow \alpha R + (1 - \alpha) M$$

($\alpha = 0.9$)

Problem: large variance in arrival times

$$R \leftarrow \alpha R + (1-\alpha)M \qquad (\alpha = 0.9)$$

M can vary wildly when loads are high! (From queuing theory)

Fix:

$$Err \equiv m - a$$

$$a \leftarrow a + gErr$$

$$v \leftarrow v + g(|Err| - v)$$

Fix 3: backoff exponentially

Uncongested load:

$$L_i = N$$

Congested load:

$$L_i = N + \gamma L_{i-1}$$

Increases exponentially!

Fix 3: backoff exponentially

Uncongested load:

$$L_i = N$$

Congested load:

$$L_i = N + \gamma L_{i-1}$$

Policy:

$$W_i = W_{i-1} + u \qquad (u \ll W_{max})$$

$$W_i = dW_{i-1} \qquad (d < 1)$$

Congestion-control throwdown

Hamilton vs Burr

Hamilton

- The internet is **too big** to find a global optimum
  - Instead, use “online learning” to try to find good solutions
  - A decision maker tries to make good decisions and observes the outcomes
- Use a **black-box** approach to modeling the internet
  - You probably can’t get a nice offline model of the internet

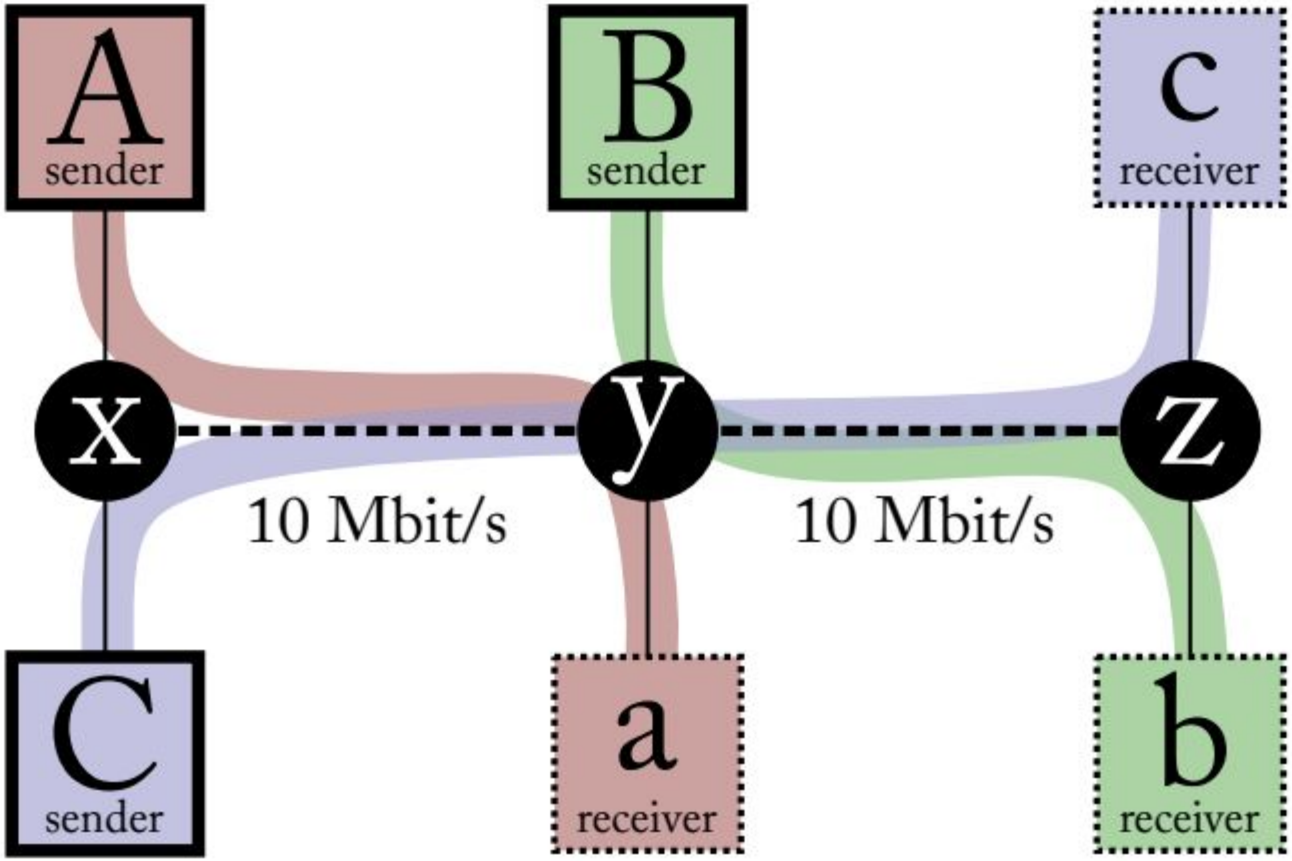
Burr

Hamilton

- The internet is **too big** to find a global optimum
  - Instead, use “online learning” to try to find good solutions
  - A decision maker tries to make good decisions and observes the outcomes
- Use a **black-box** approach to modeling the internet
  - You probably can’t get a nice offline model of the internet

Burr

- Greedy local algorithms **are terrible** in many cases
  - They don’t even work in small examples
  - Hard to scale to arbitrary internet
- Greedy algorithms **already encode** assumptions about the internet
  - Why not use these explicitly to optimize beforehand?
  - “TCP Tahoe-like schemes worked well in the 1990s on shallow-buffered bottlenecks with many flows; they performed poorly in the 2000s…”


A
sender
B
sender
c
receiver
x
y
z
10 Mbit/s
10 Mbit/s
C
sender
a
receiver
b
receiver

Hamilton replies

- But **black-box** can work so well!
  - Machine learning has taken over computer vision, for example
- The real world is much more **messy** than your small examples
  - We can't predict the behavior in a white box!
- We should use different learning algorithms in different contexts

Burr concludes

- Sure, I only care about performance overall
- I'm still skeptical about learning because of the parking-lot example
- Should we even be theorizing?
  - Google just **tries things** like BBR on some fraction of their network to see if it works well
  - Live against all of the real google internet traffic

QUIK

Key point in congestion control

1. How TCP detects the congestion occur
2. How TCP adjust the transmitting rate
3. Which method should be used by TCP to adjust the transmission rate

These 3 questions have been solved previous part.

But what is different in QUIC?

Sensing congestion

Delay_sensing CC

TCP: calculate the delay: sender ->send packet -> receiver -> ack -> sender, then adjust the number of packet.

QUIC: also recode an additional delay: send packet -> receiver -> ack

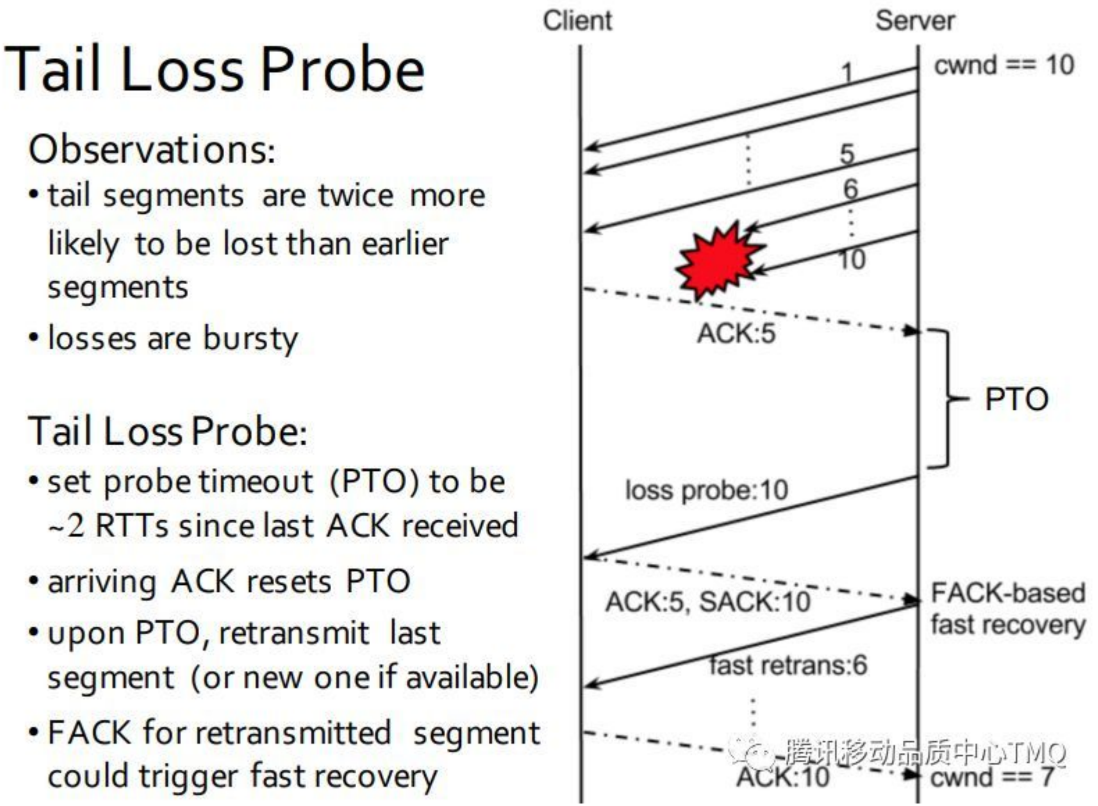
So that we can get the accurate Round-Trip time(RTT).

Use fast retransmit

TCP: retransmit based on timer

QUIC: fast retransmit,

Tail Loss Probes(TLPs)



Flexibility

QUIC implement CC in application level.

Easy to do optimization.