Introduction to Computer Networks

TCP Congestion Control (I)

CS640

https://pages.cs.wisc.edu/~mgliu/CS640/S26/index.html

Ming Liu
mgliu@cs.wisc.edu

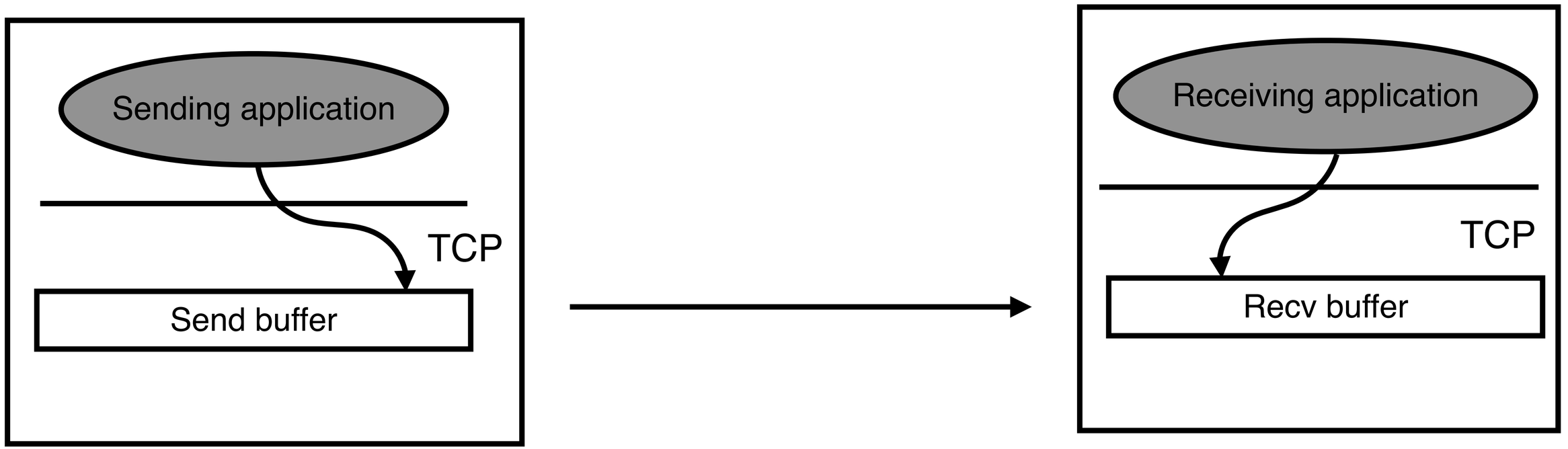
Outline

- Last
  - TCP Reliability Support (II)
- Today
  - TCP Congestion Control (I)
- Announcements
  - Lab 4 due date 04/30/2026

Recap: UDP Issues

- #1: Arbitrary communication
  - Senders and receivers can talk to each other in any ways
- #2: No reliability guarantee
  - Packets can be lost/duplicated/reordered during transmission
  - A checksum is not enough
- **#3: No resource management**
  - **Each channel works as an exclusive network resource owner**
  - **No adaptive support for the physical networks and applications**

Set Up the Context



Set Up the Context

What is the goal of TCP congestion control?

What is the goal of TCP congestion control?

Effectively use the networking resources

What is the goal of TCP congestion control?

Effectively use the networking resources

- Utilization: each networking hardware is fully utilized
- Fairness: each networking hardware is equally shared

Challenges

- #1: Varying resource capacities
  - The underlying networking fabric and hardware are dynamic

Challenges

- #1: Varying resource capacities
  - The underlying networking fabric and hardware are dynamic

- #2: Unpredictable traffic
  - We don’t know how applications use the network and their requirements

The Key Idea

- The smart-sender dumb-receiver philosophy
  - The sender adjusts the sending window based on congestion signals
  - Congestion signal: an event telling that network contention **might** happen

The Key Idea

- The smart-sender dumb-receiver philosophy
  - The sender adjusts the sending window based on congestion signals
  - Congestion signal: an event telling that network contention **might** happen

- Congestion window = Sending window
  - Define the total amount of data the sender can push into the network without overwhelming it
  - AdvertiseWindow: the total amount of data the sender can send to the receiver without overwhelming it

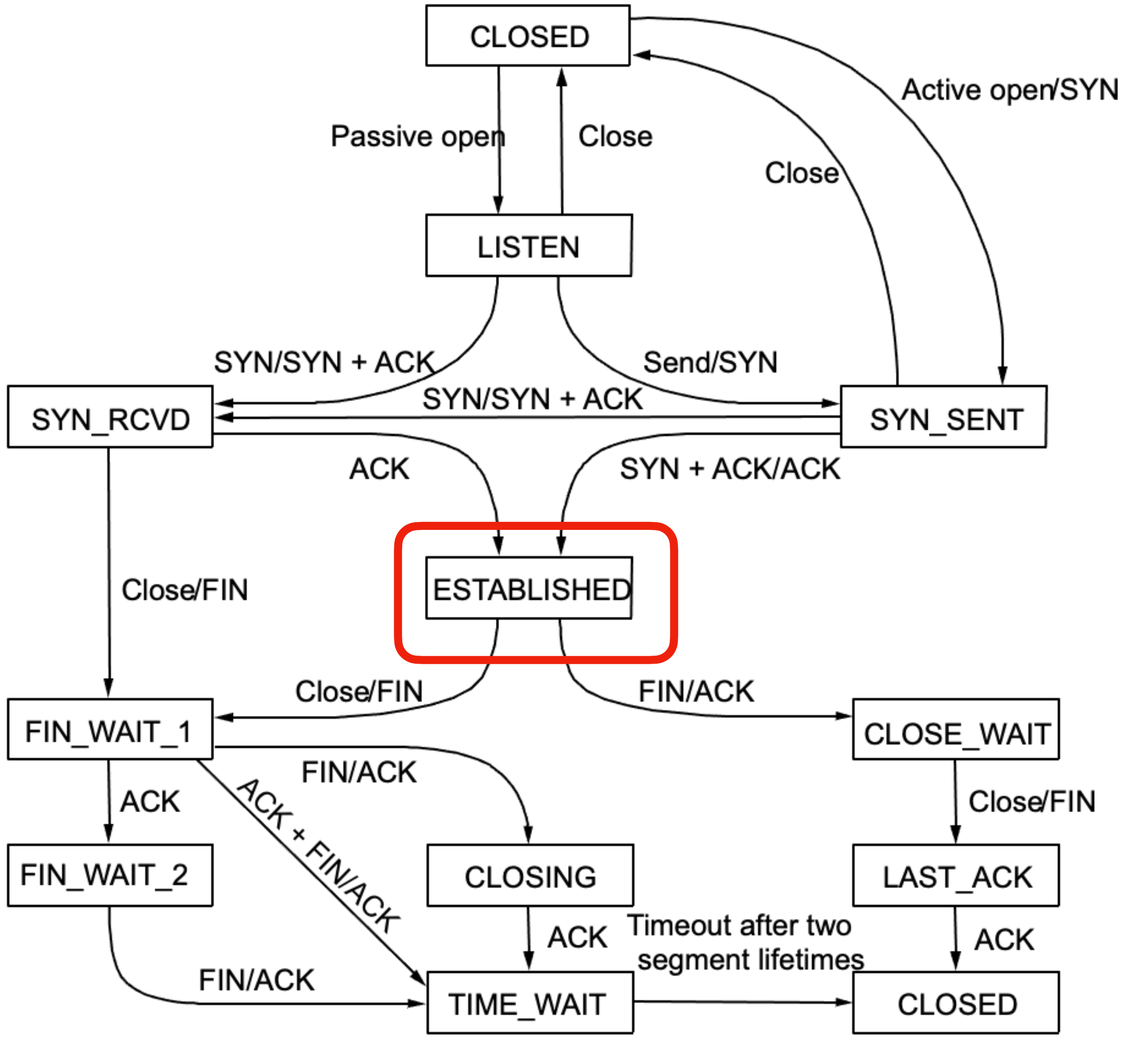
The Key Idea

- The smart-sender dumb-receiver philosophy
  - The sender adjusts the sending window based on congestion signals
  - Congestion signal: an event telling that network contention **might** happen

- Congestion window = Sending window
  - Define the total amount of data the sender can push into the network without overwhelming it
  - AdvertiseWindow: the total amount of data the sender can send to the receiver without overwhelming it
  - **EffectiveWindow=MIN(CongestionWindow, AdvertiseWindow)**

But how?

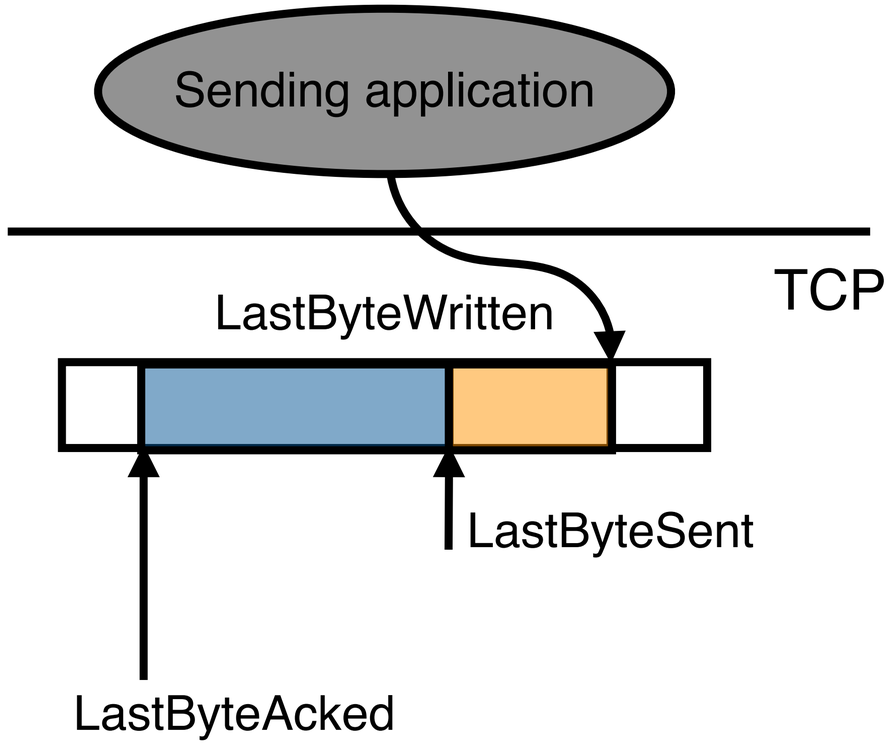
Let’s Start From the Begining



How much data to send at first

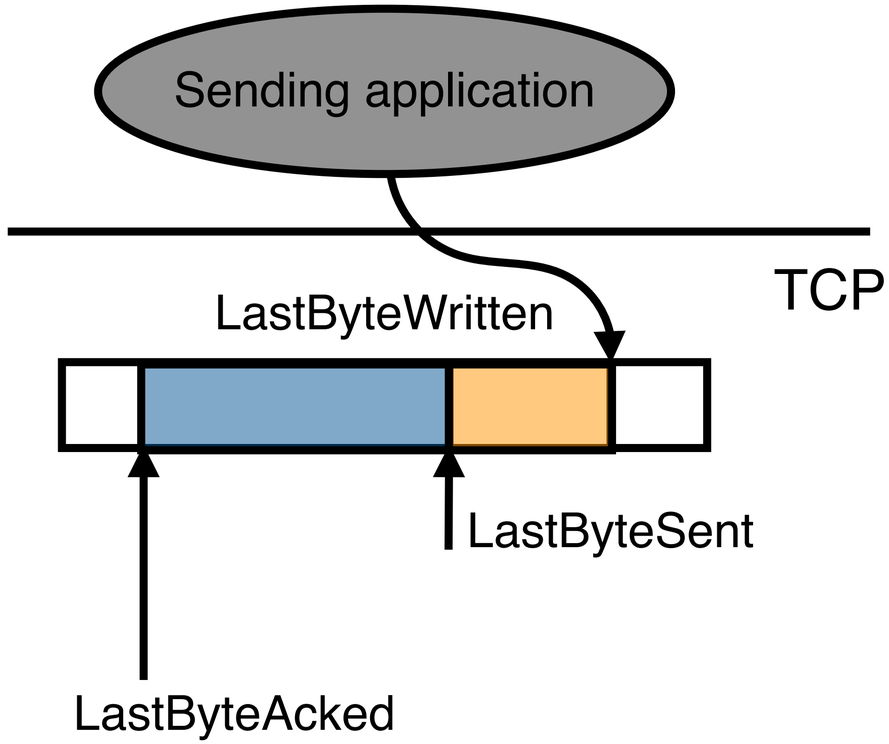
How much data to send at first

- Option #1: send AdvertisedWindow-sized data
  - Not overwhelming the receiver



How much data to send at first

- Option #1: send AdvertisedWindow-sized data
  - Not overwhelming the receiver
  - **But this might overwhelm the network!**



How much data to send at first

- Option #1: send AdvertisedWindow-sized data
  - Not overwhelming the receiver
  - **But this might overwhelm the network!**

- Option #2: send random-sized data, whose size is N
  - N is between 0 and AdvertisedWindow
  - Not overwhelming the receiver

How much data to send at first

- Option #1: send AdvertisedWindow-sized data
  - Not overwhelming the receiver
  - **But this might overwhelm the network!**

- Option #2: send random-sized data, whose size is N
  - N is between 0 and AdvertisedWindow
  - Not overwhelming the receiver
  - **But this might also overwhelm the network!**

How much data to send at first

- Option #1: send AdvertisedWindow-sized data
  - Not overwhelming the receiver
  - **But this might overwhelm the network!**

- Option #2: send random-sized data, whose size is N
  - N is between 0 and AdvertisedWindow
  - Not overwhelming the receiver
  - **But this might also overwhelm the network!**

- Option #3: just send 1 segment
  - A conservative approach but keeps the data pipe moving

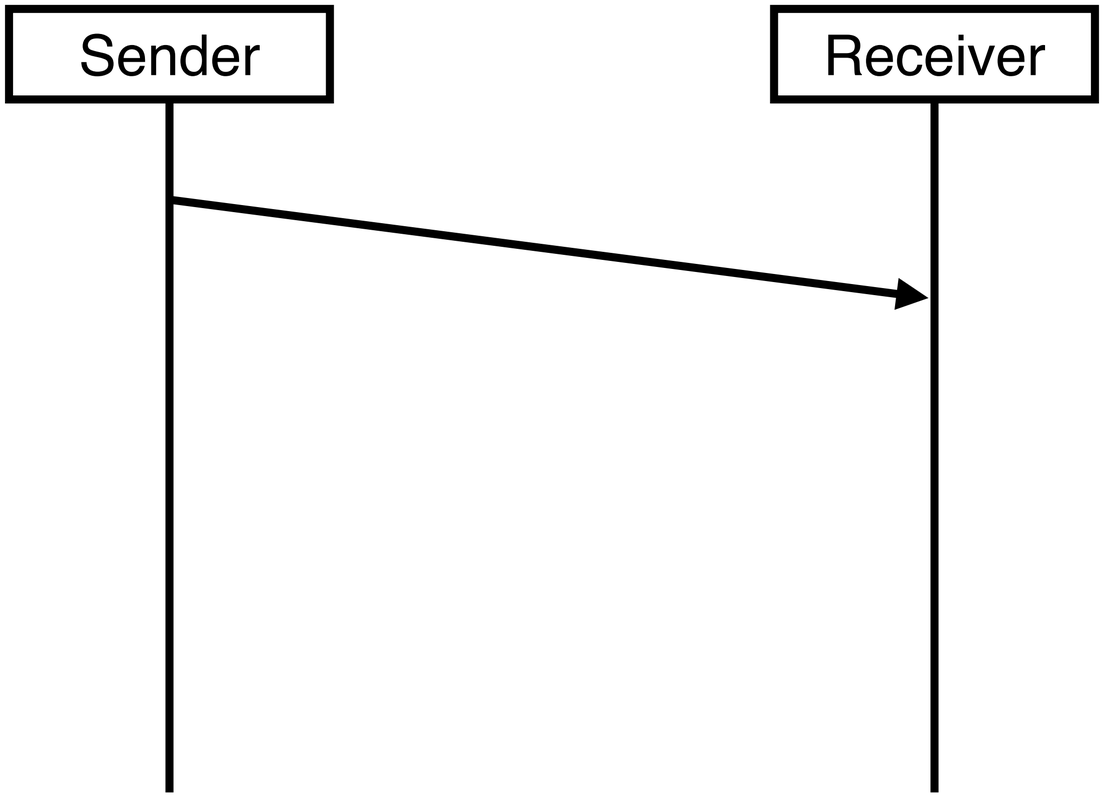
How much data to send at first

- Option #1: send AdvertisedWindow-sized data
  - Not overwhelming the receiver
  - **But this might overwhelm the network!**

- Option #2: send random-sized data, whose size is N
  - N is between 0 and AdvertisedWindow
  - Not overwhelming the receiver
  - **But this might also overwhelm the network!**

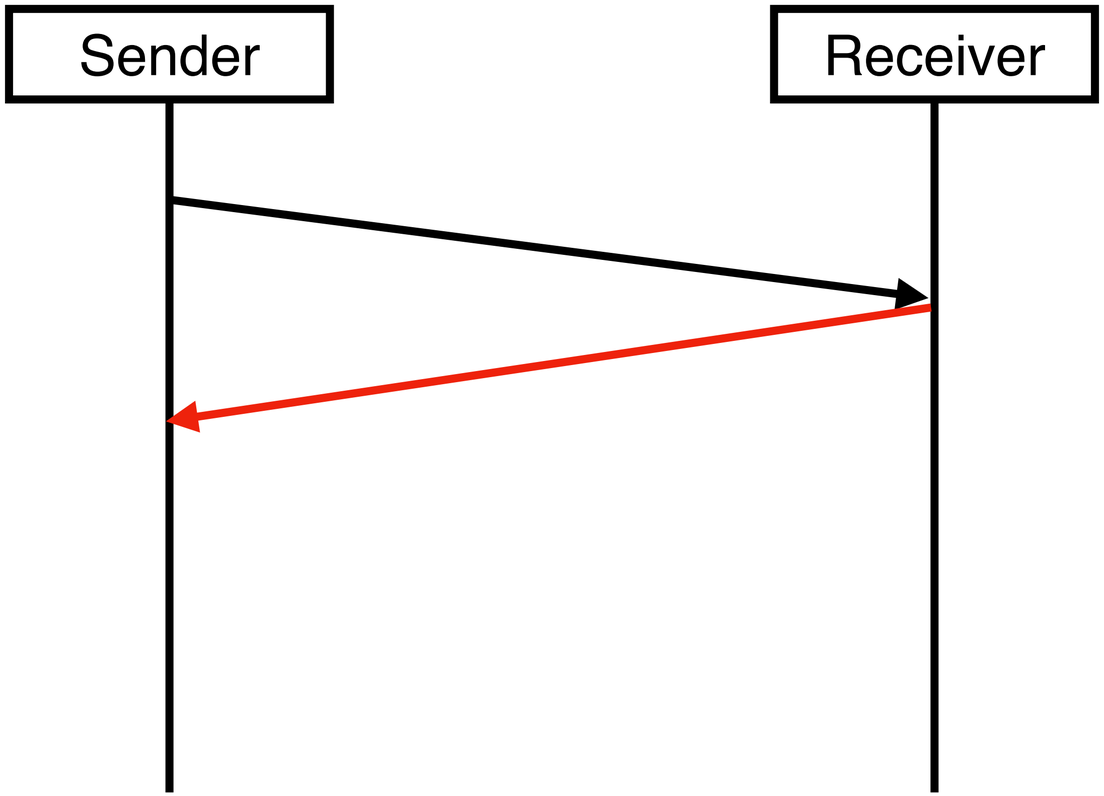
- **Option #3: just send 1 segment => TCP goes with it**
  - **A conservative approach but keeps the data pipe moving**

What happens next



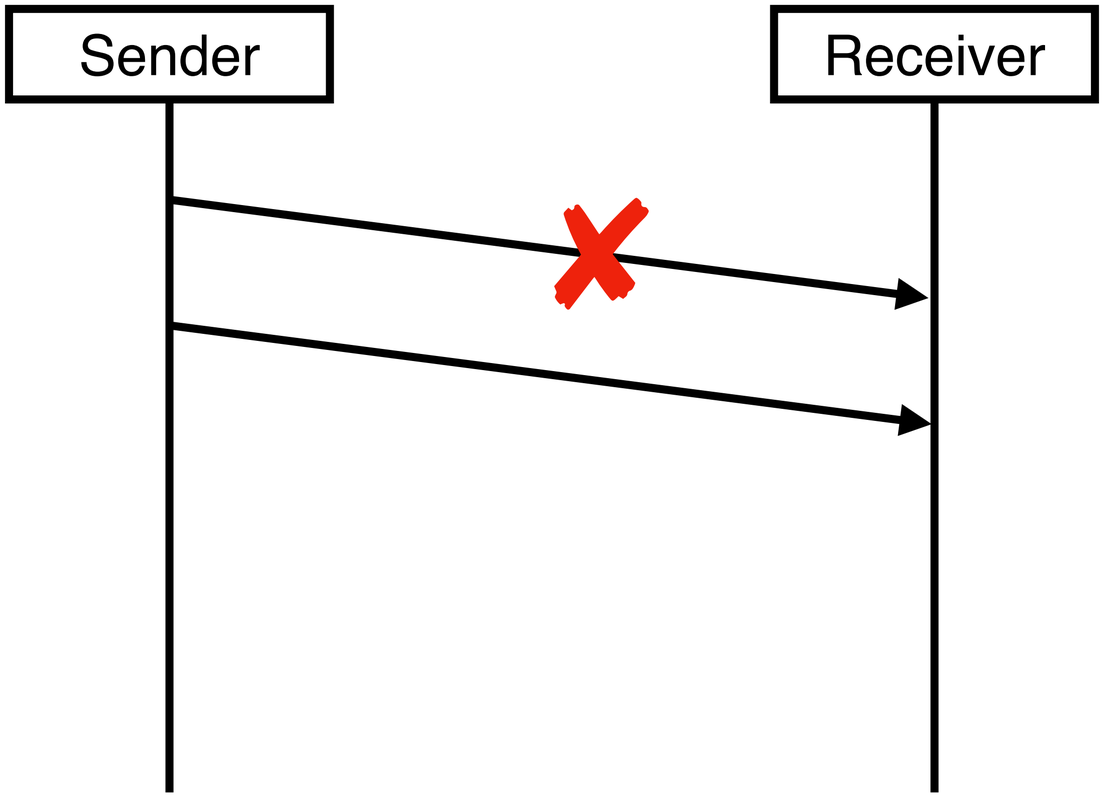
What happens next

- Case #1: the receiver receives the segment and returns an ACK
  - The ACK also carries the AvertisedWindow



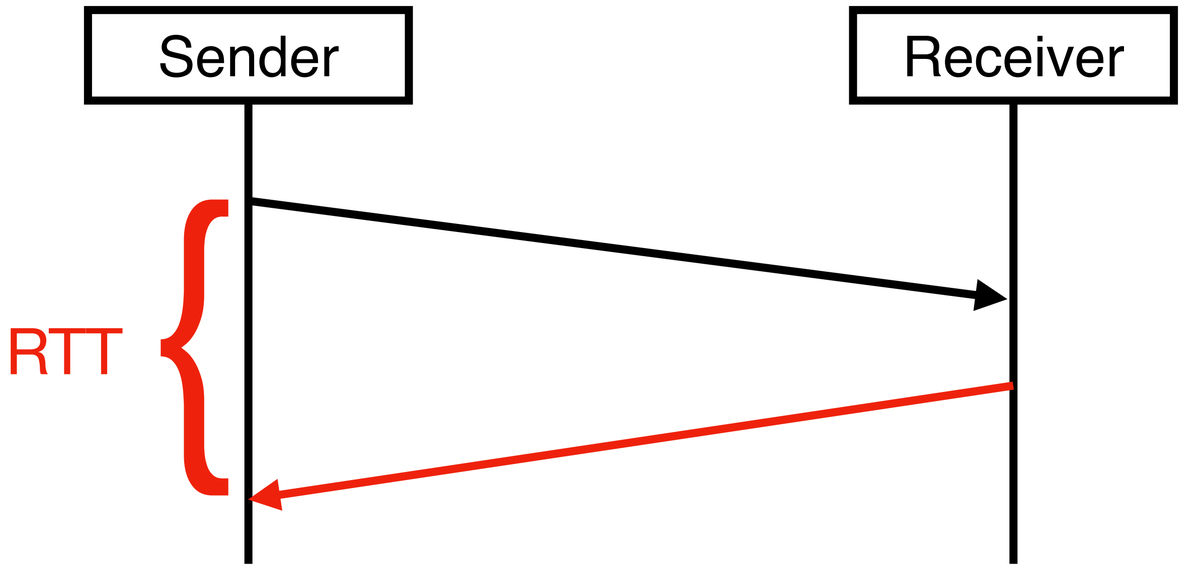
What happens next

- Case #2: the receiver receives nothing
  - The segment or its replied ACK is dropped, but we don’t know which one
  - A local timeout triggers and the sender sends it again



The 1st Round Summary

- Case 1: receive an ACK w/o timeout
  - Effective BW = Segment Size / RTT
  - Congestion Window > 1 segment
- Case 2: receive an ACK w/ timeout
  - Effective BW = Segment Size / Amplified RTT
  - Congestion Window = 1 segment



How much data to send next round

How much data to send next round

- Option #1: send AdvertisedWindow-sized data
  - Not overwhelming the receiver
  - But this might overwhelm the network!

- Option #2: send random-sized data, whose size is N
  - N is between 0 and AdveertisedWindow
  - Not overwhelming the receiver
  - But this might also overwhelm the network!

- Option #3: send 1 segment
  - A conservative approach
  - Slow performance

How much data to send next round

- Option #1: send AdvertisedWindow-sized data
  - Not overwhelming the receiver
  - But this might overwhelm the network!

- Option #2: send random-sized data, whose size is N

The network bandwidth availability is still unknown!

- Option #3: send 1 segment
  - A conservative approach
  - Slow performance

How much data to send next round

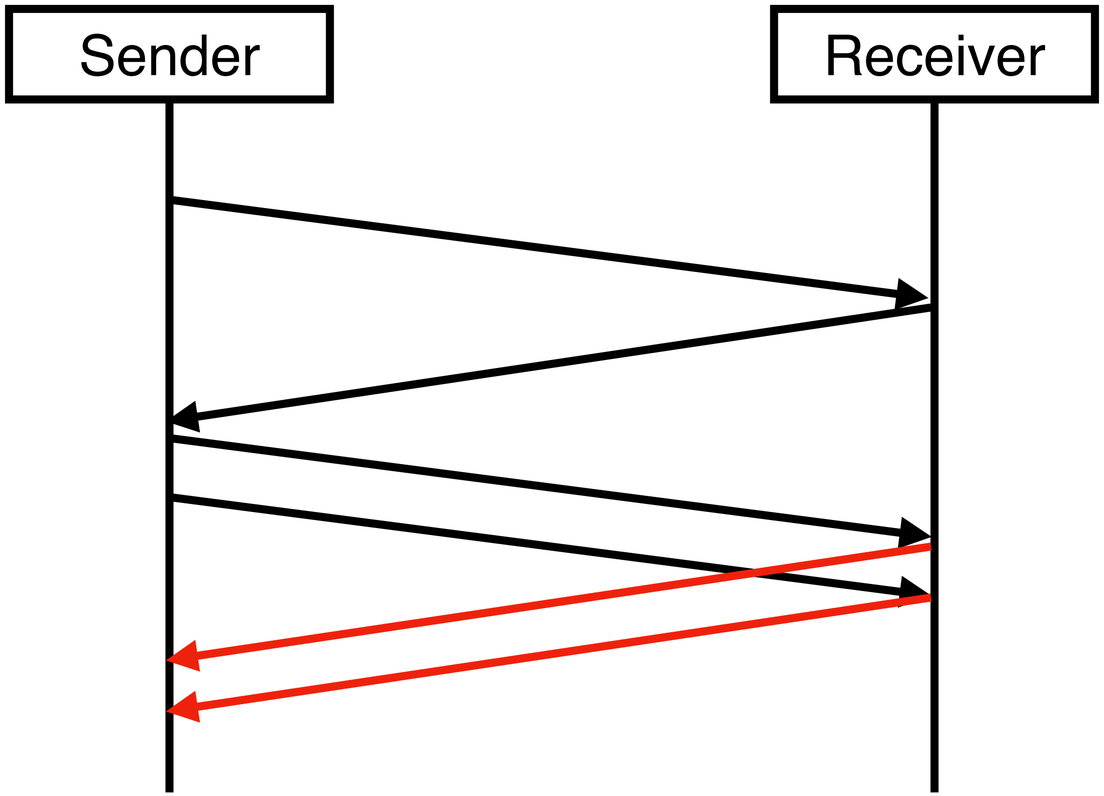
- Option #1: send AdvertisedWindow-sized data
  - Not overwhelming the receiver
  - But this might overwhelm the network!

- Let's introduce some "probing" here
- Option #4: send 2 segments
  - Implication: The 1st round try with 1 segment succeeds. The network might be able to do more!

  - A conservative approach
  - Slow performance

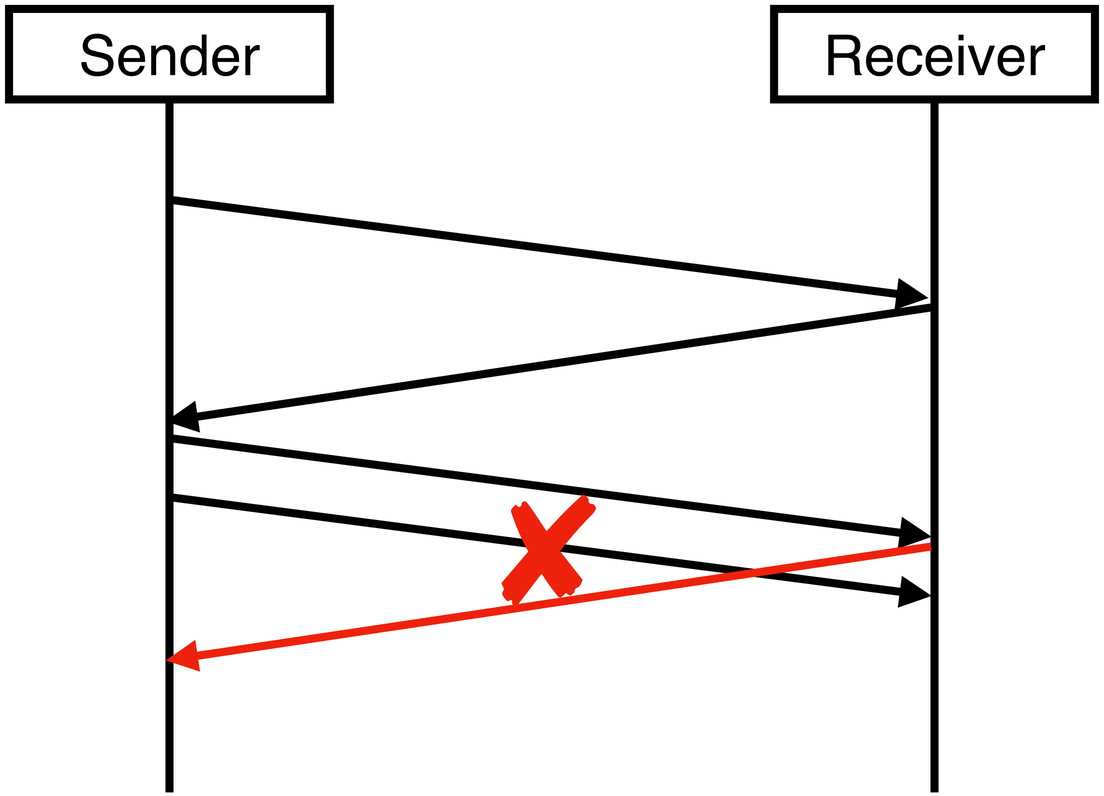
What happens next?

- Case #1: the receiver receives two segments and returns ACKs
  - The probing works!



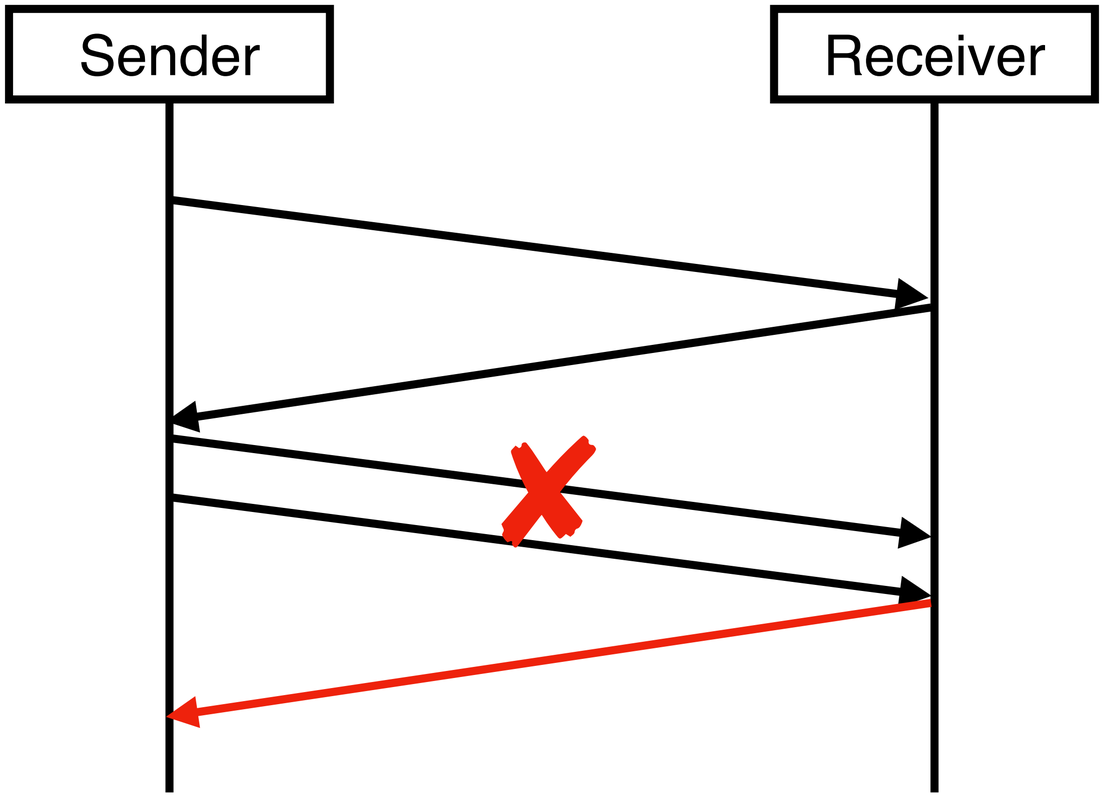
What happens next?

- Case #2: the receiver receives 1st segment and returns an ack
  - The timeout of the 2nd segment => retransmit until receiving an ACK



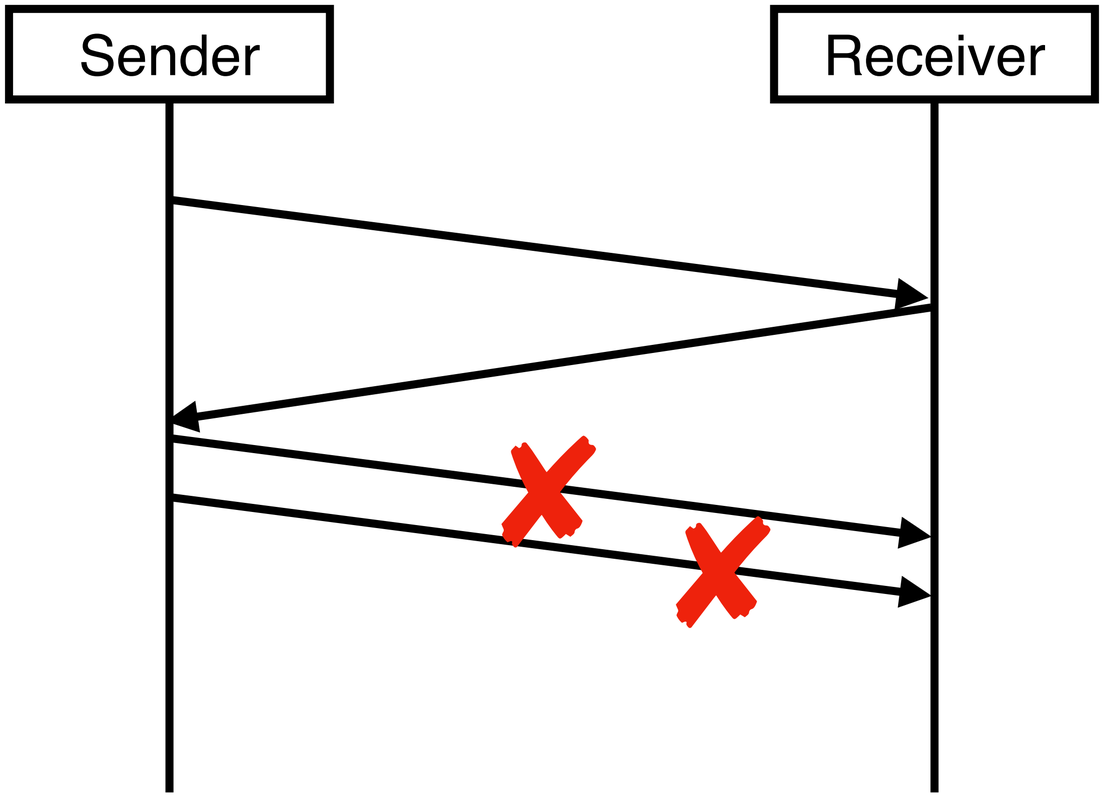
What happens next?

- Case #3: the receiver receives 2st segment and returns an ack
  - An out-of-order ACK => retransmit
  - Or a local timeout => retransmit



What happens next?

- Case #4: the receiver receives nothing
  - Local timeout triggers and retransmit



The 2nd Round Summary

- Case #1: the receiver receives 2 ACKs
  - Average BW = 3 segments / Time-to-send-3-segments
  - 2nd BW = 2 segments / Time-to-send-2-segments
  - Congestion Window > 2 segments

- Case #2: the receiver receives 1ACK (1st seg) w/ 2nd timeout
  - Average BW = 3 segments / Time-to-send-3-segments (amplified)
  - 2nd BW = 2 segments / Time-to-send-2-segments (amplified)

The 2nd Round Summary

- Case #1: the receiver receives 2 ACKs
  - Average BW = 3 segments / Time-to-send-3-segments
  - 2nd BW = 2 segments / Time-to-send-2-segments
  - Congestion Window > 2 segments

- Case #2: the receiver receives 1ACK (1st seg) w/ 2nd timeout
  - Average BW = 3 segments / Time-to-send-3-segments (amplified)
  - 2nd BW = 2 segments / Time-to-send-2-segments (amplified)
  - **Congestion Window = 1 segment**

1 segment is the congestion window from the last round.

The 2nd Round Summary (con't)

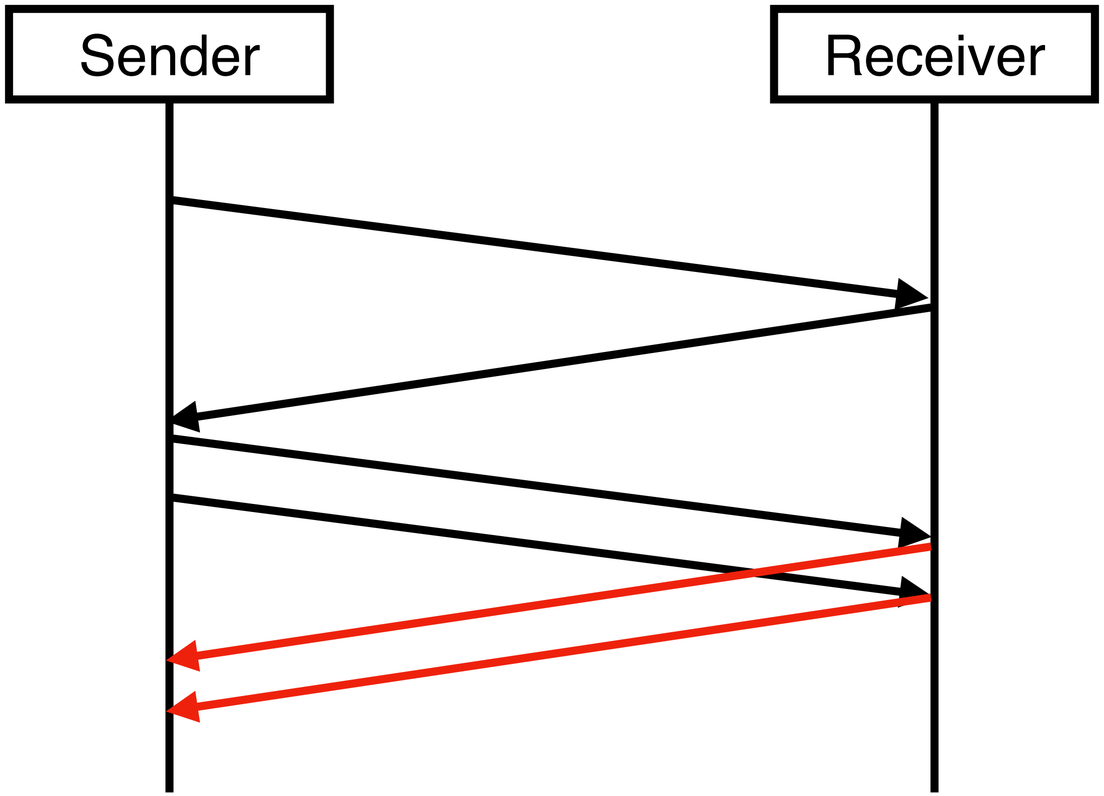
- Case #3: the receiver receives 2 ACK (2nd seg) w/o 2nd timeout
  - An out-of-order ACK happens => an implicit signal on the contention
  - But out-of-order ACK is not as strong as a local timeout
  - Average BW = 3 segments / Time-to-send-3-segments (amplified)
  - 2nd BW = 2 segments / Time-to-send-2-segments (amplified)
  - Congestion Window = 1 segment

- Case #3': the receiver receives 1 ACK (2nd seg) w/ 2nd timeout
  - Average BW = 3 segments / Time-to-send-3-segments (amplified)
  - 2nd BW = 2 segments / Time-to-send-2-segments (amplified)
  - Congestion Window = 1 segment

The 2nd Round Summary (con’t)

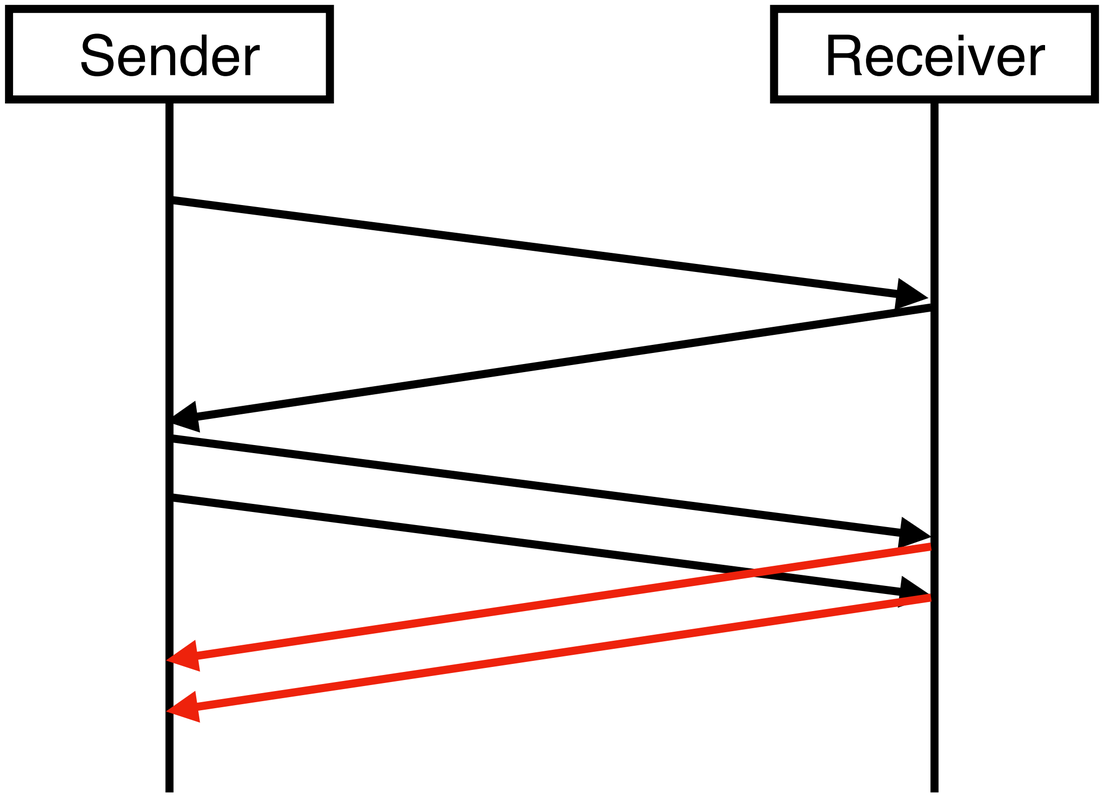
- Case #4: the receiver receives nothing
  - Average BW = 3 segments / Time-to-send-3-segments (amplified)
  - 2nd BW = 2 segments / Time-to-send-2-segments (amplified)
  - Congestion Window = 1 segment

How much data to send next round?



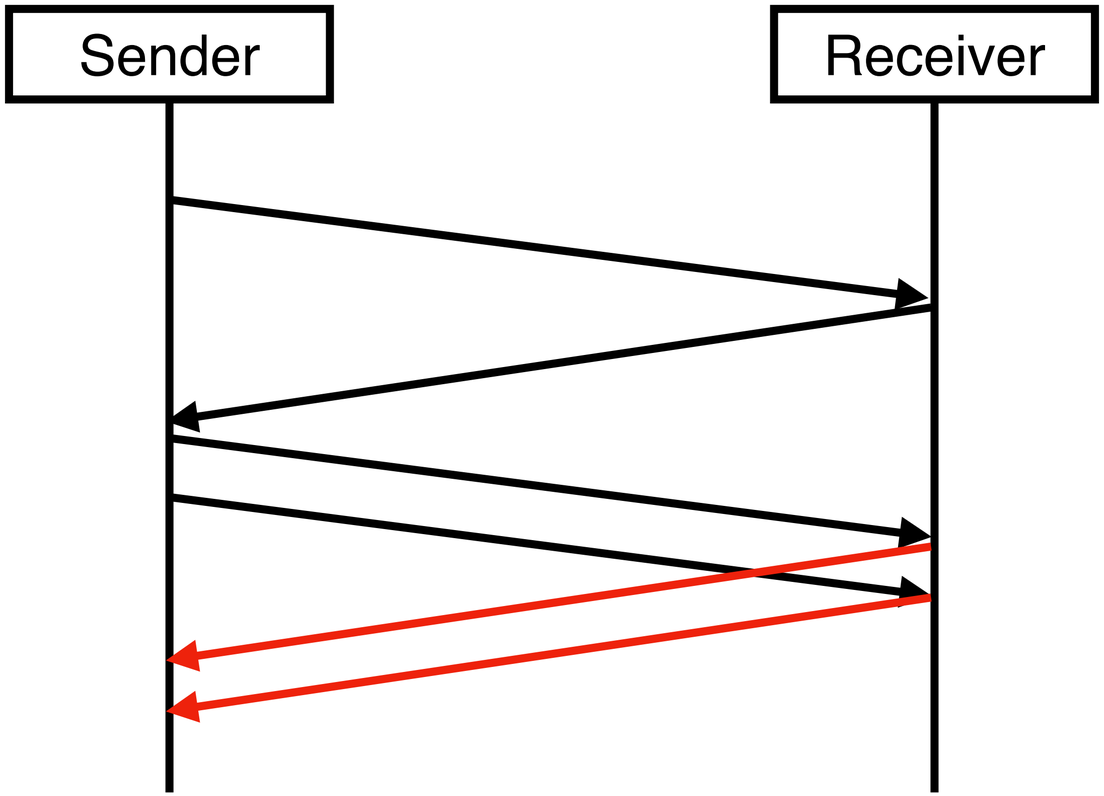
How much data to send next round?

- Keep probing if the last round succeeds
- Otherwise, just send 1 segment



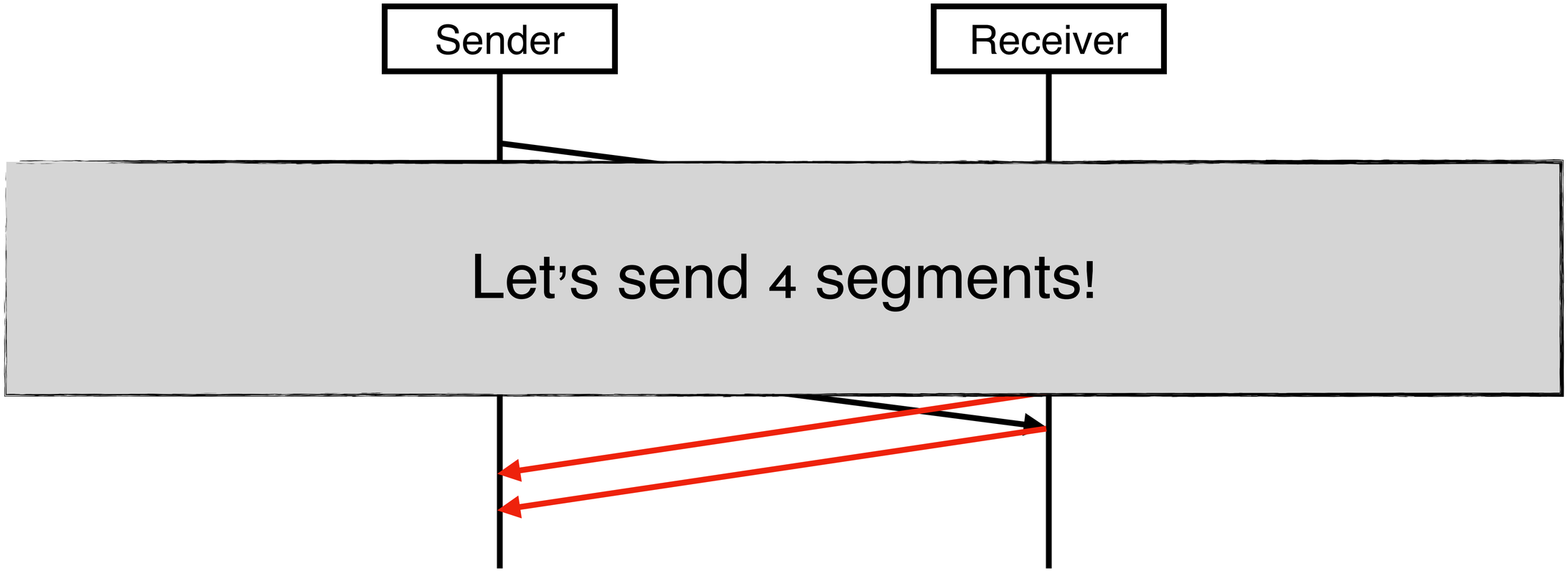
How much data to send next round?

- Suppose we do a probing
  - Problem: how can we quickly find the maximum available capacity
  - Let’s do an exponential increase



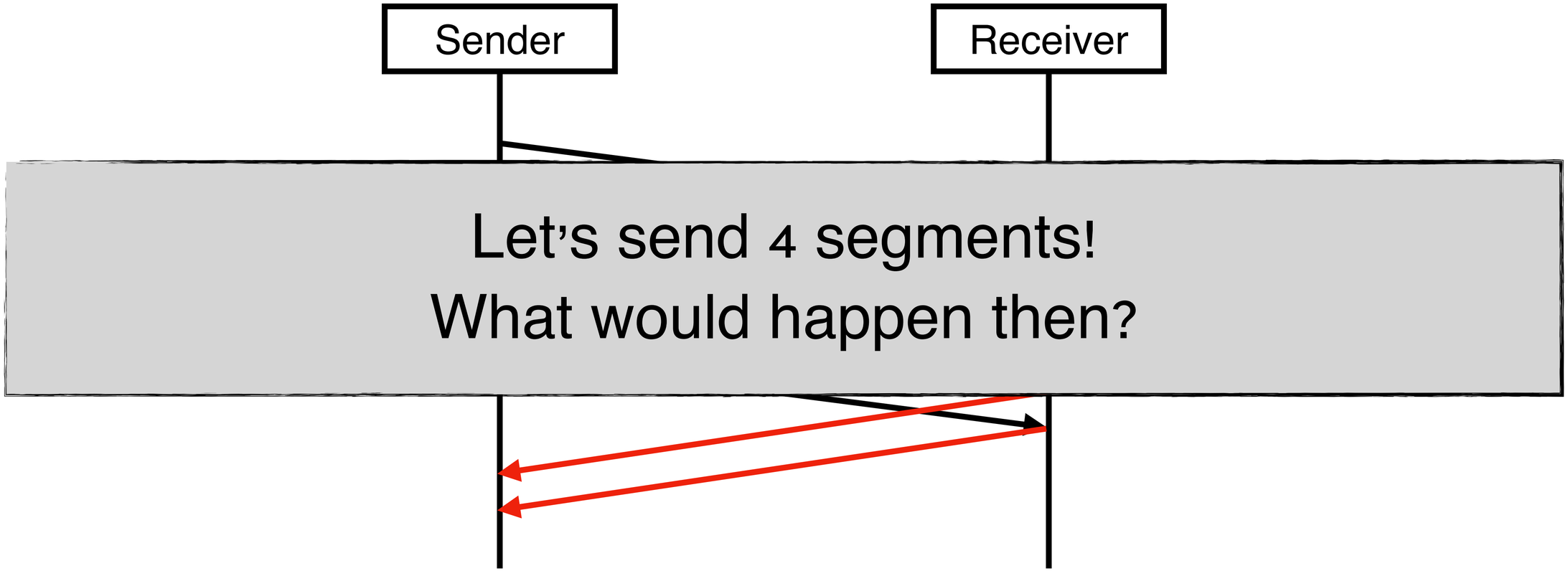
How much data to send next round?

- Suppose we do a probing
  - Problem: how can we quickly find the maximum available capacity
  - Let's do an exponential increase



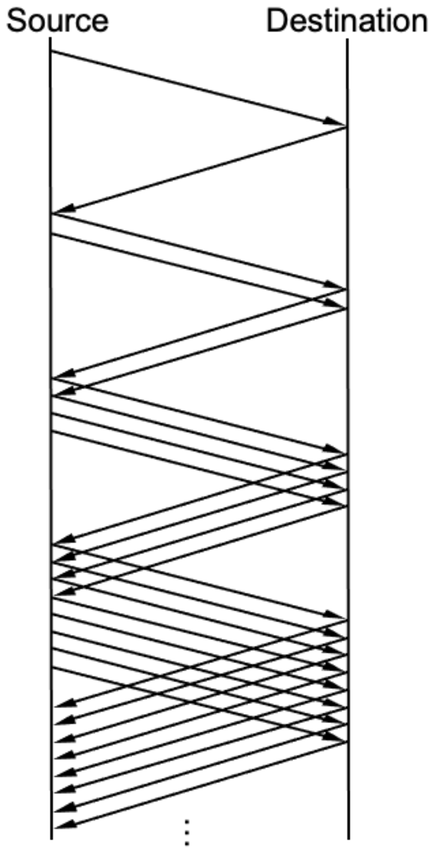
How much data to send next round?

- Suppose we do a probing
  - Problem: how can we quickly find the maximum available capacity
  - Let's do an exponential increase



TCP Slow Start

- Determine the available networking capacity exponentially
  - At round i, probe the congestion window with 2^(i-1) segments



TCP Slow Start

- Determine the available networking capacity exponentially
  - At round i, probe the congestion window with 2^(i-1) segments
- Congestion signal: out-of-order ACK
  - An indirect indicator of a congested network
  - Probing should stop

TCP Slow Start

- Determine the available networking capacity exponentially
  - At round i, probe the congestion window with 2^(i-1) segments
- Congestion signal: out-of-order ACK
  - An indirect indicator of a congested network
  - Probing should stop
  - Congestion window = congestion threshold
  - Congestion threshold = congestion window/2

The congestion threshold is continuously updated every round to capture the bandwidth availability.

TCP Slow Start

- Determine the available networking capacity exponentially
  - At round i, probe the congestion window with 2^(i-1) segments
- Congestion signal: out-of-order ACK
  - An indirect indicator of a congested network
  - Probing should stop
  - Congestion window = congestion threshold
  - Congestion threshold = congestion window/2
- Congestion signal: local time out
  - A strong indicator of a congested network
  - Probing should stop
  - Congestion window = 1 segment

TCP Slow Start

- Determine the available networking capacity exponentially
  - At round i, probe the congestion window with 2^(i-1) segments
- Congestion signal: out-of-order ACK
  - An indirect indicator of a congested network
  - Probing should stop
  - A strong indicator of a congested network
  - Probing should stop
  - Congestion window = 1 segment

How does the transmission look like so far?

TCP Slow Start

- Determine the available networking capacity exponentially
  - At round i, probe the congestion window with 2^(i-1) segments
- Congestion signal: out-of-order ACK
  - An indirect indicator of a congested network
  - Probing should stop
  - A strong indicator of a congested network
  - Probing should stop
  - Congestion window = 1 segment

Is this efficient?

The goal of TCP congestion control:

Effectively use the networking resources

- Utilization: each networking hardware is fully utilized
- Fairness: each networking hardware is equally shared

Running Phase Bandwidth Adjustment: AIMD

- Adjust the window for fairness and high utilization

- Additive Increase/Multiplicative Decrease
  - Additive increase CongestionWindow when the congestion goes down
  - Multiplicate decrease CongestionWindow when the congestion goes up

Congestion goes up

- Signals:
  - Packet loss
  - Out-of-order ACK

Congestion goes up

- Signals:
  - Packet loss
  - Out-of-order ACK

- Multiplicative Decrease and Why

Congestion goes up

- Signals:
  - Packet loss
  - Out-of-order ACK

- Multiplicative Decrease and Why
  - Quickly shrink the congestion window to avoid congestion collapse
  - Reduce to 1 segment under heavy contention => packet loss
  - Reduce to >1 segment under light contention => out-of-order ACK
  - Congestion Window = Congestion Window X Parameter

Congestion goes down

- Signals:
  - Packets from the last congestion window are delivered successfully

Congestion goes down

- Signals:
  - Packets from the last congestion window are delivered successfully

- Additive Increase and Why

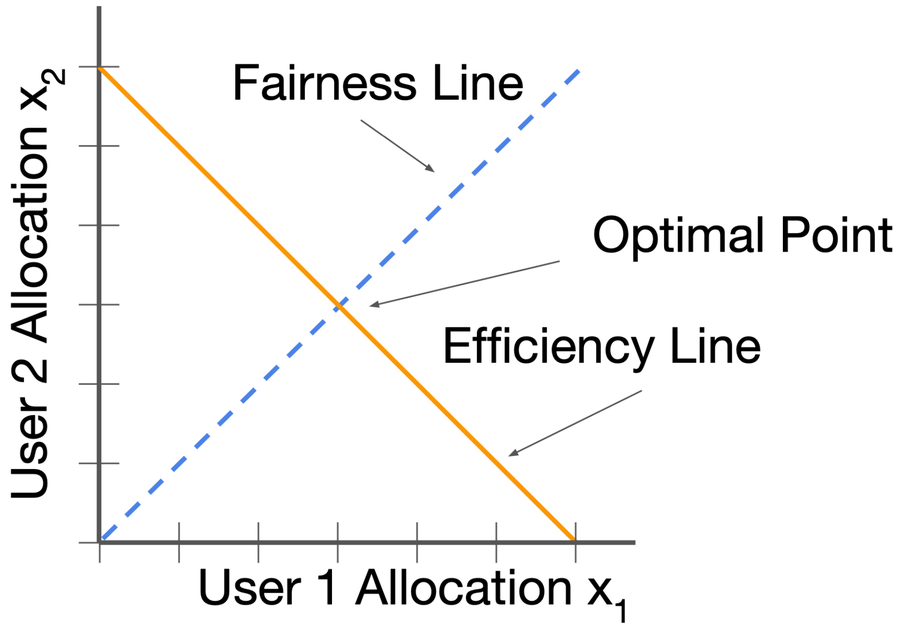
Congestion goes down

- Signals:
  - Packets from the last congestion window are delivered successfully

- Additive Increase and Why
  - Gradually approach the equal bandwidth share offered by the network
  - The addition enables fairness guarantees implicitly
  - No exponential increase since the slow start has found the max in
  - Congestion Window = Congestion Window + Parameter

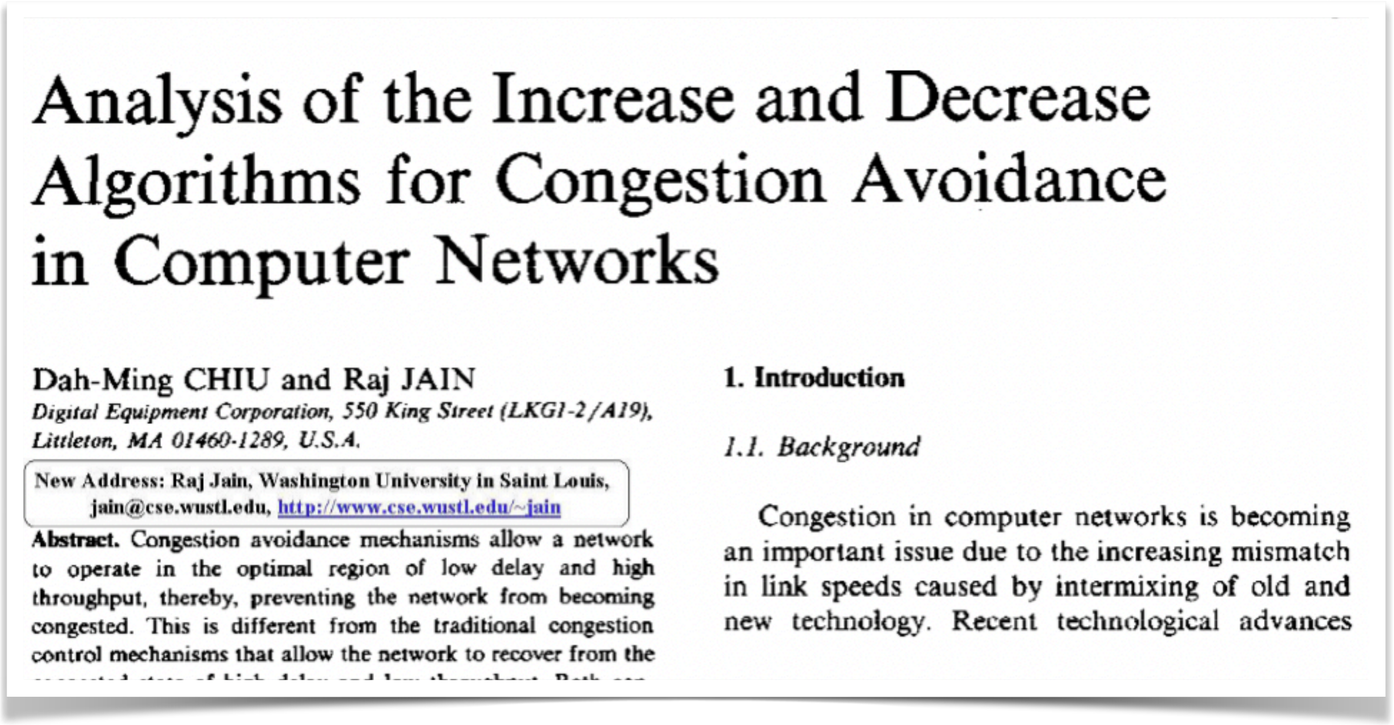
Discussion

- Parameter selection is hard
  - People sometimes use the fluid model to find out the optimal ones
- The congestion control should converge to the optimal point.
  - AIMD can, but takes several rounds.



Some References

- An ideal AIMD algorithm
  - Efficiency, fairness, convergence, and distributeness



Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks

Dah-Ming CHIU and Raj JAIN
Digital Equipment Corporation, 550 King Street (LKG1-2/A19), Littleton, MA 01460-1289, U.S.A.

New Address: Raj Jain, Washington University in Saint Louis, jain@cse.wustl.edu, http://www.cse.wustl.edu/~jain

Abstract. Congestion avoidance mechanisms allow a network to operate in the optimal region of low delay and high throughput, thereby, preventing the network from becoming congested. This is different from the traditional congestion control mechanisms that allow the network to recover from the

1. Introduction

1.1. Background

Congestion in computer networks is becoming an important issue due to the increasing mismatch in link speeds caused by intermixing of old and new technology. Recent technological advances

Summary

- Today
  - TCP congestion control (I)

- Next lecture
  - TCP congestion control (II)