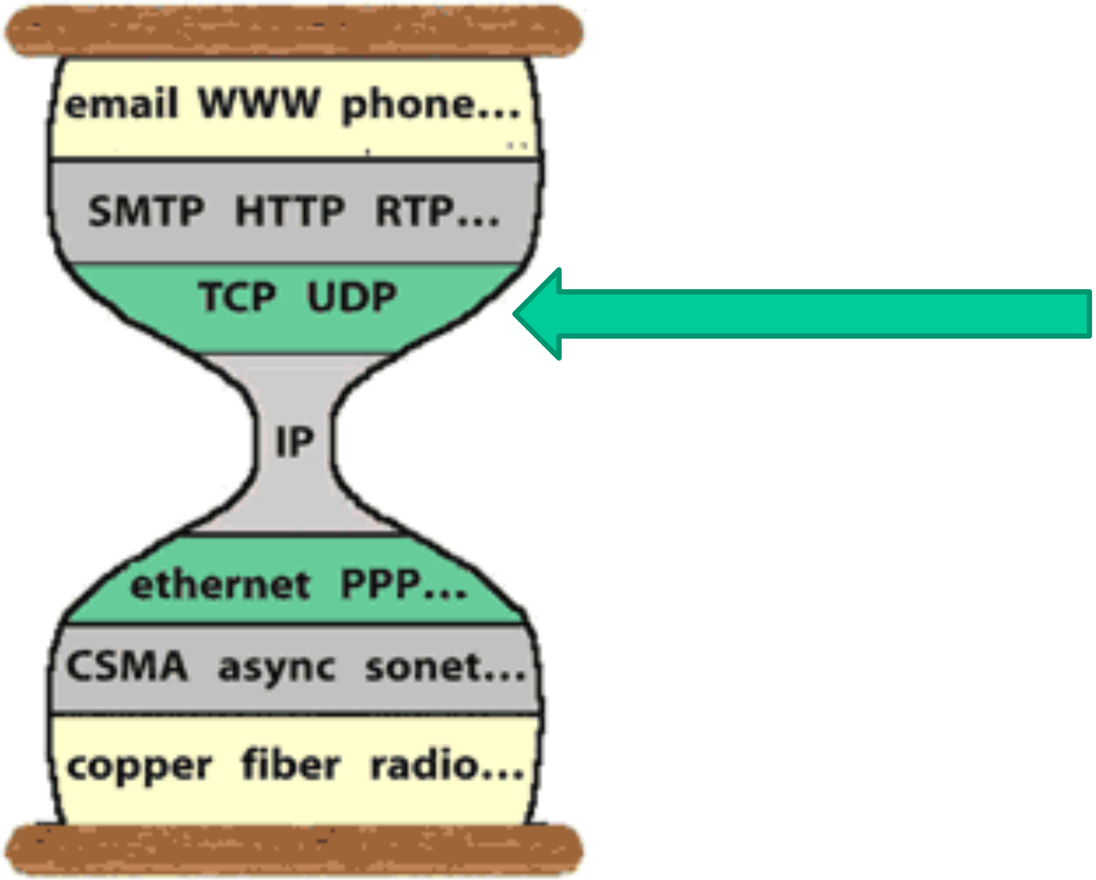
Wrapping Up Congestion Control

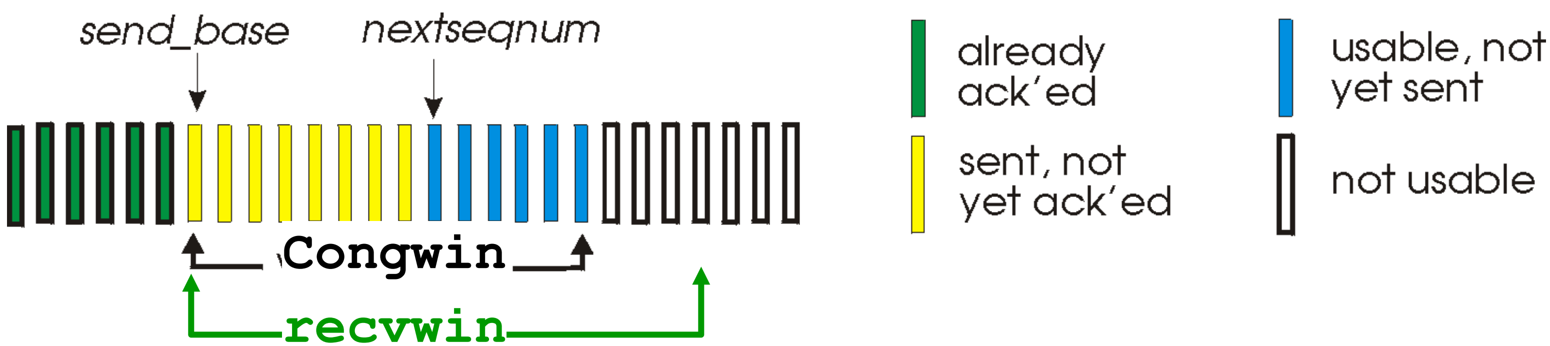


Chapter 3

3.5 TCP Congestion Control

TCP Congestion Control

- Add a "congestion control window" **`cwnd`** on top of flow-control window
- Sender limits: **`LastByteSent-LastByteAcked`** ≤ **`cwnd`**



- **`cwnd`** initialized to 1 packet, increases until congestion
  - How sender infers congestion: packet loss (timeout, or 3 dup. ACKs)
- Upon loss: *decrease* **`cwnd`**, then begin increasing again
  - Two phases: (1) slow start, (2) congestion avoidance
  - a threshold **`(sshthresh)`** defines the boundary between the two

MSS: maximum segment size

"Slow" Start

- **Objective: quickly gauge the pipeline size**

1. Start with cwnd = 1 MSS
2. Send cwnd packets
3. If all packets got acked
   - cwnd = 2 * cwnd
   - goto 2
4. Else

   have gone too far

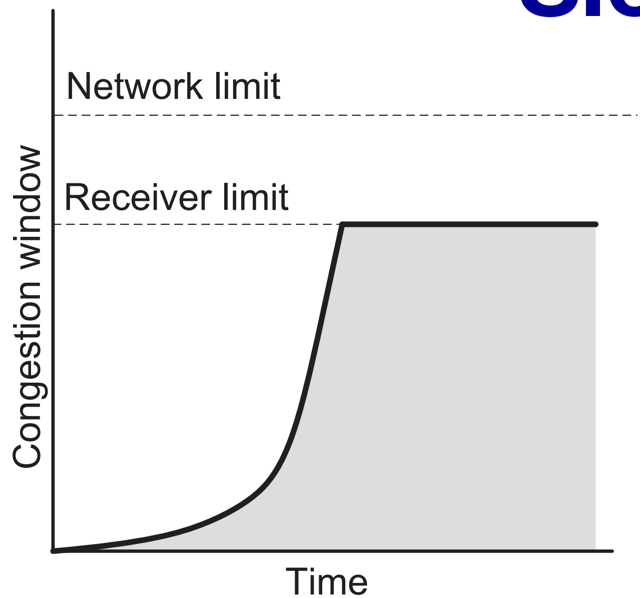
   - ss-thresh = cwnd / 2
   - cwnd = 1 MSS
   - goto 2

Same, but using "self-clocking" of TCP

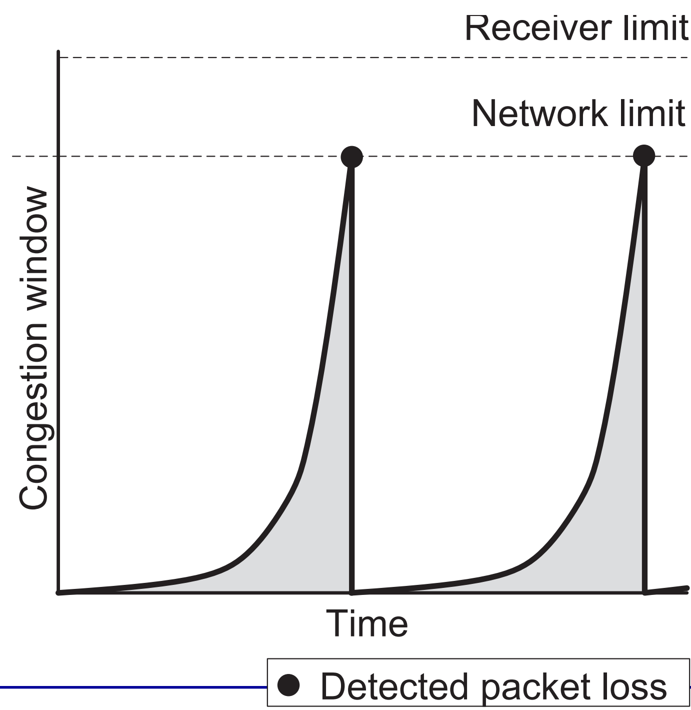
1. Start with cwnd = 1 MSS
2. Send cwnd packets
3. If ack
   - cwnd = cwnd + 1 MSS
   - send packets
4. If timeout
   - sshthresh = cwnd / 2
   - cwnd = 1 MSS
   - goto 2

Multiplicative Increase, Reset to 1 Decrease

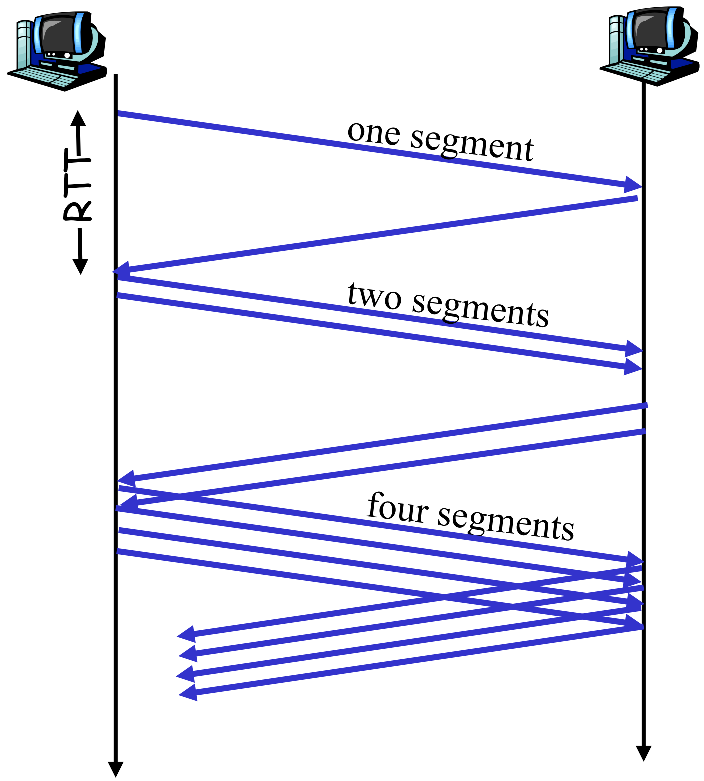
Slow Start



Network limit
Receiver limit
Congestion window
Time



Receiver limit
Network limit
Congestion window
Time
Detected packet loss



RTT
one segment
two segments
four segments

Congestion Avoidance

- Objective: maintain steady state, probe for unused resources, avoid conflicts

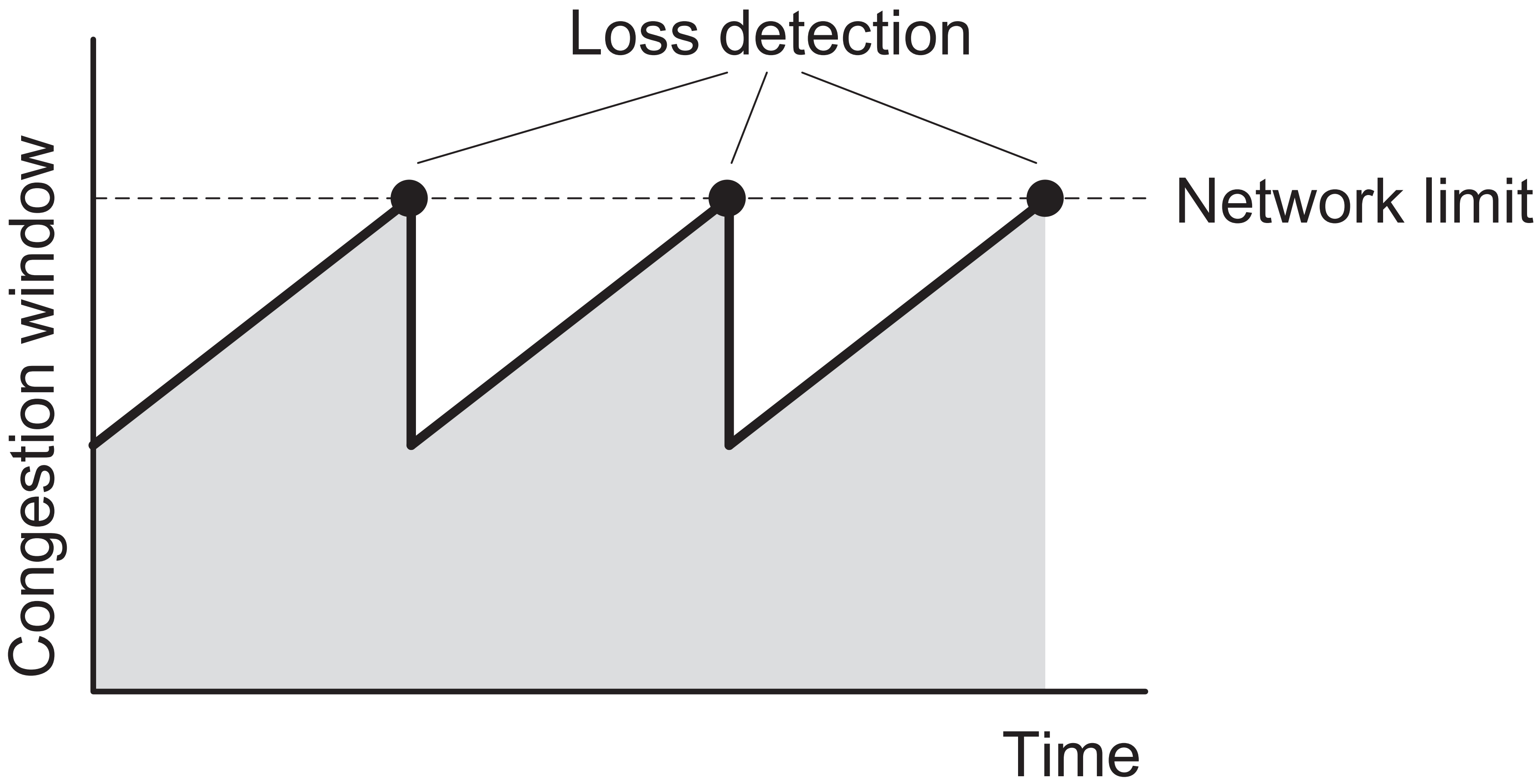
- Send cwnd packets
- If all sent packets got ack
  - cwnd = cwnd + 1 MSS
- Else
  - cwnd = cwnd / 2

Send cwnd packets
If ack
- cwnd = cwnd + $(1\ \text{MSS})^2$/cwnd

If timeout
- cwnd = cwnd / 2

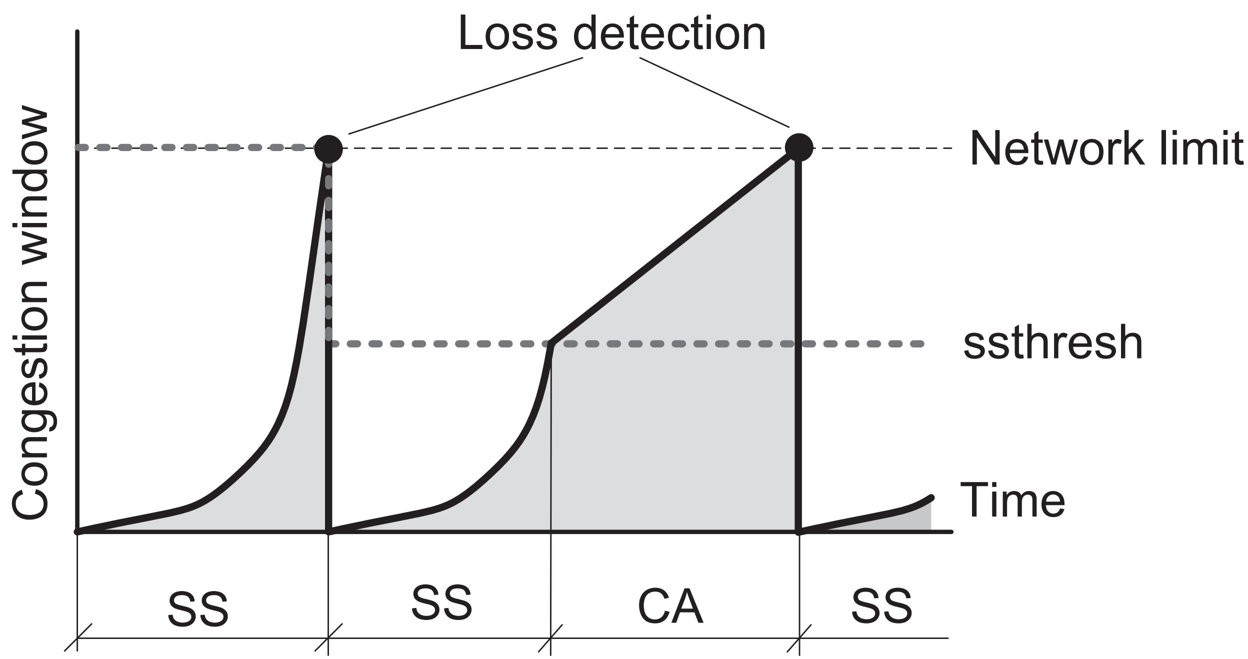
- Additive Increase, Multiplicative Decrease (AIMD)

Congestion Avoidance



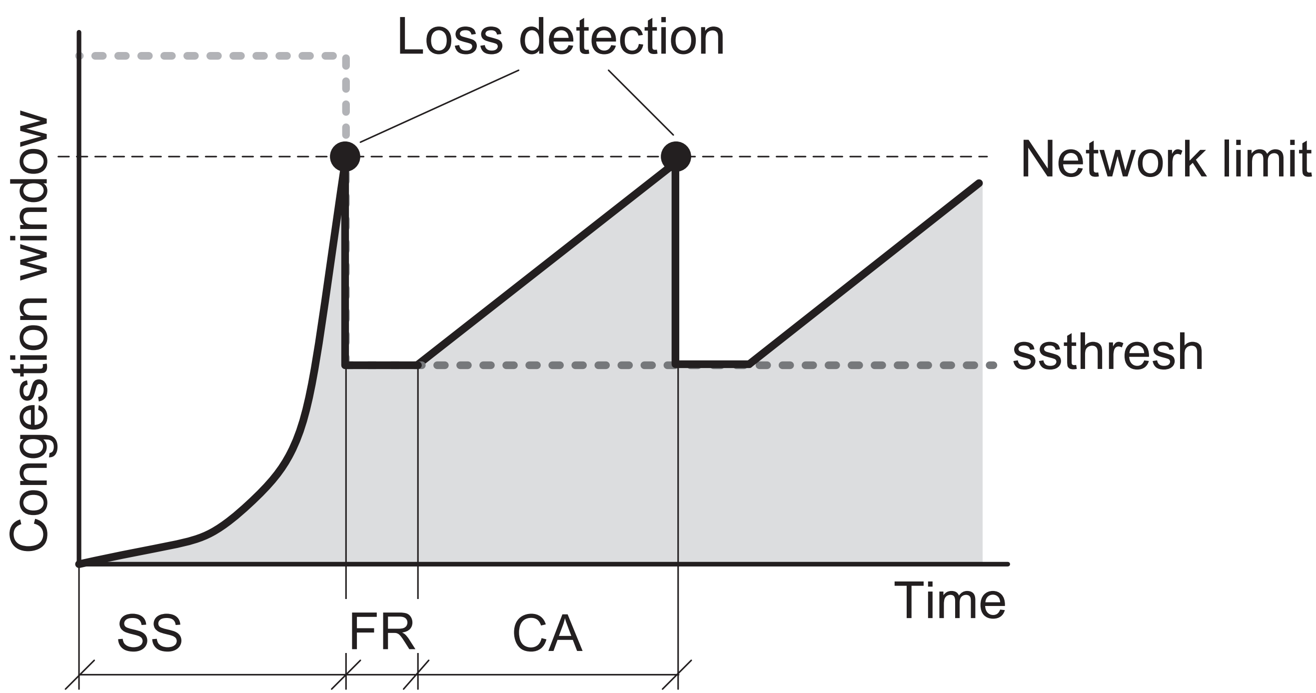
Combining Slow Start with Congestion Avoidance (Tahoe)

- Set initial `ss-thresh`
- When `cwnd < ss-thresh`, use Slow Start
  - sshthresh will get updated
- when `cwnd` ≥ `ss-thresh`, use Congestion Avoidance

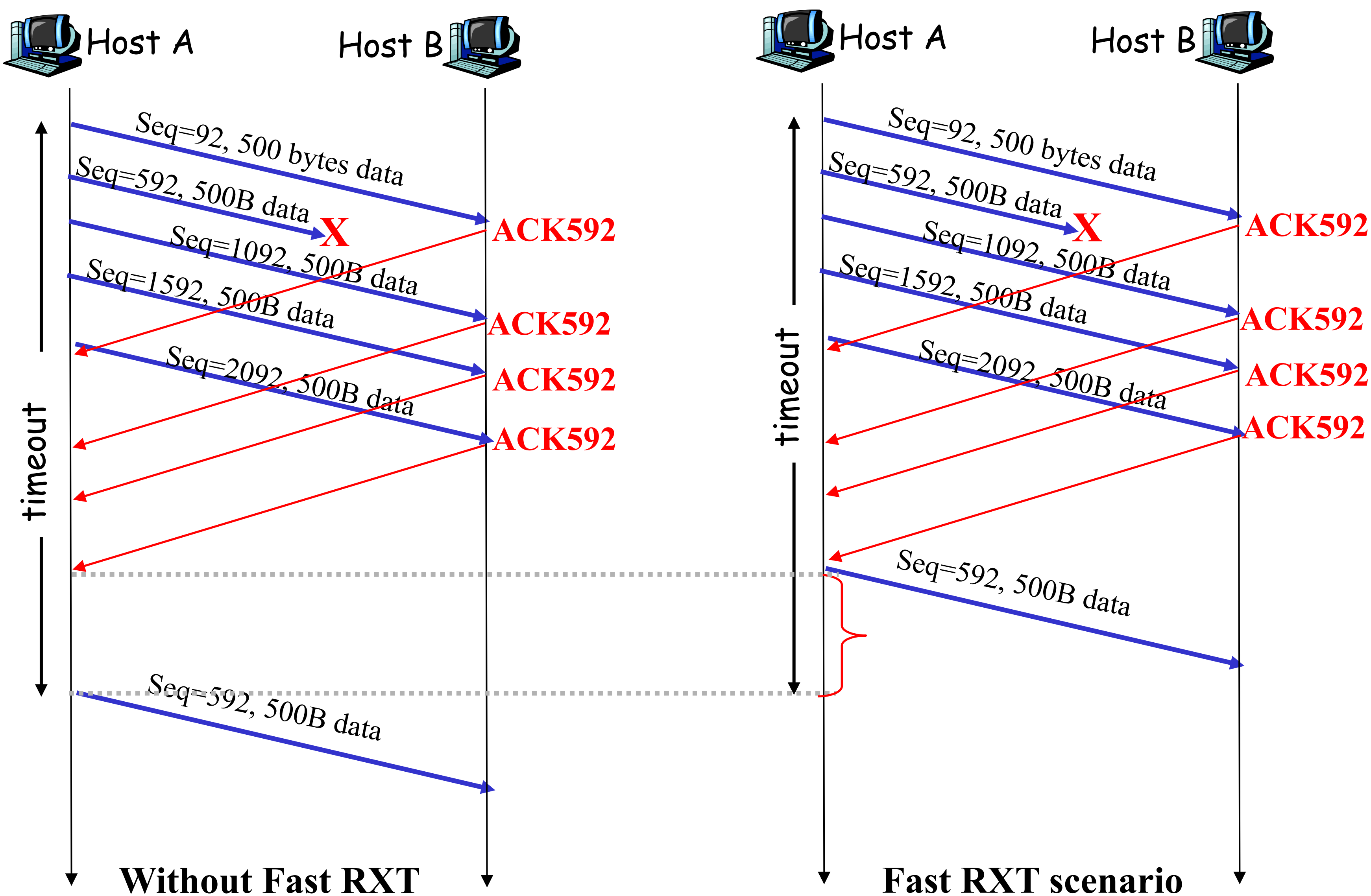


Slow Start and Congestion Avoidance (Reno)

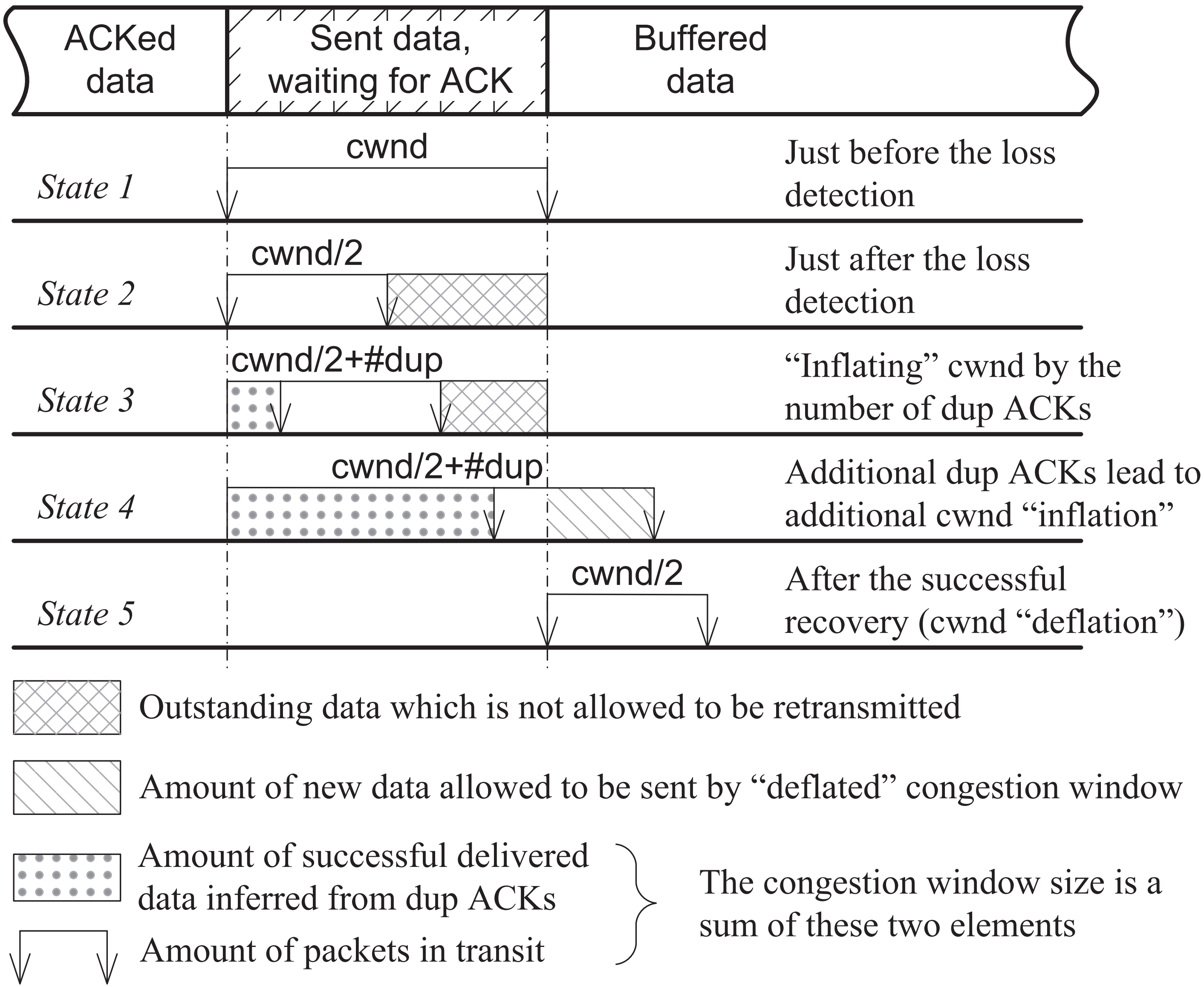
- If just one packet is lost, dropping cwnd to 1 is an overreaction
- What if loss detected through 3 dup acks, we just reduce cwnd by half, and quickly recover from the loss (Fast Recovery)



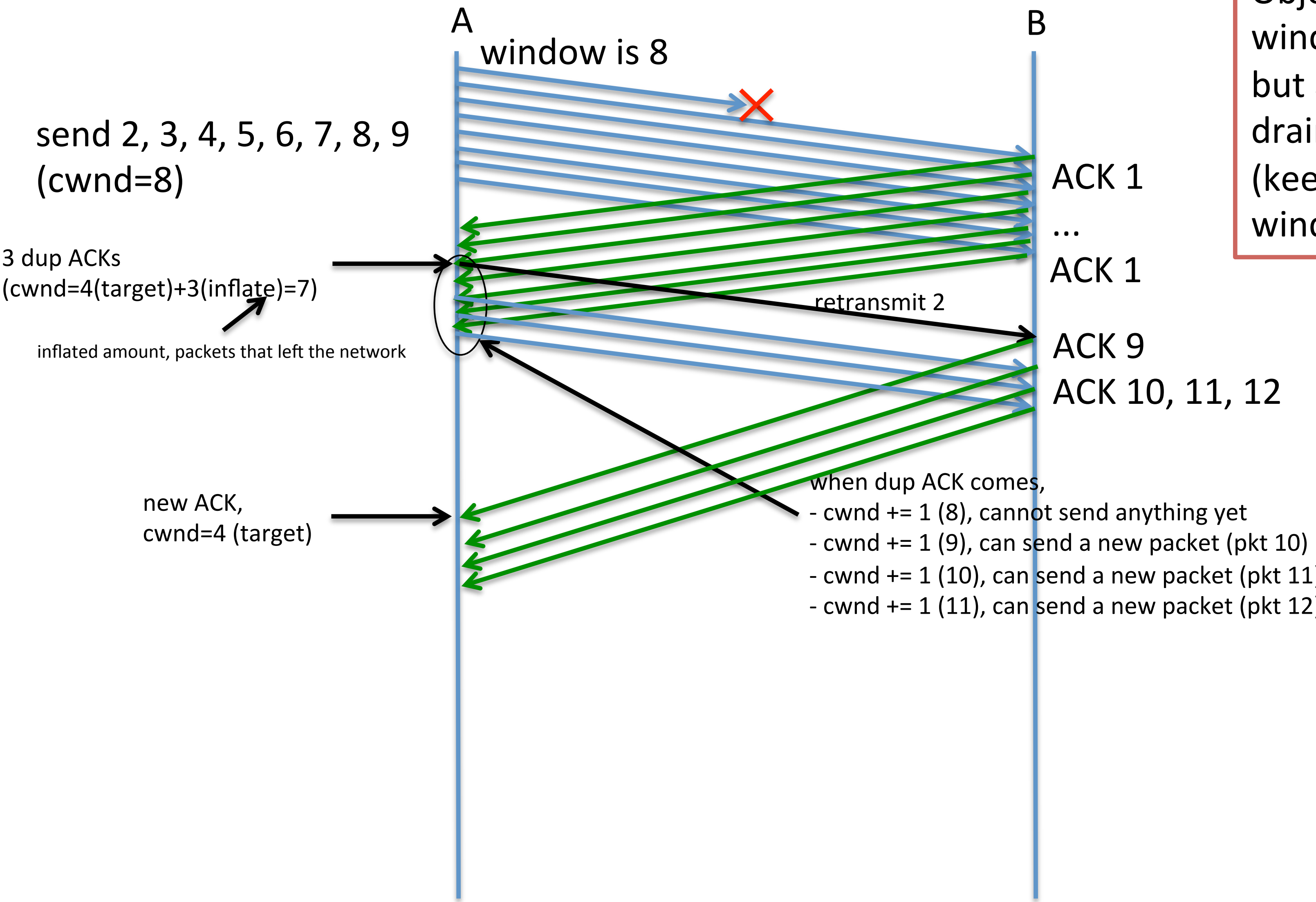
TCP fast retransmit example



Fast Recovery/Retransmit (Reno)



Fast Retransmit / Fast Recovery



Objective: cut window by half, but avoiding draining the pipe (keep ~half of window in flight)

Is TCP fair?

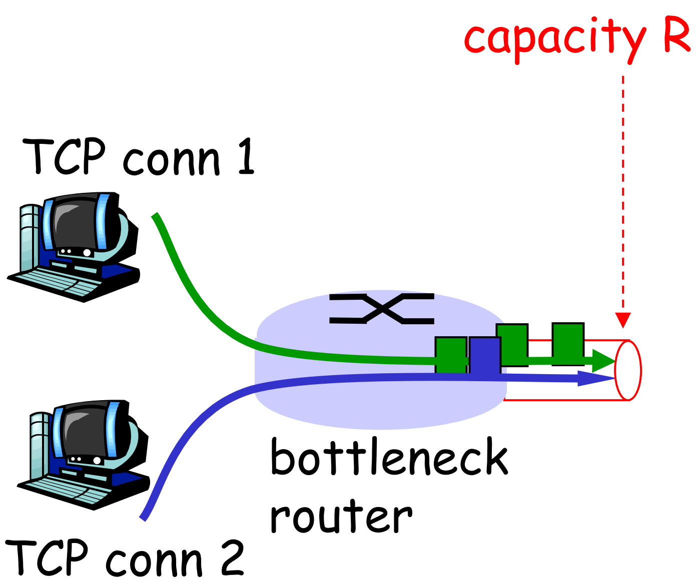
Fairness: if N TCP sessions share same bottleneck link, each should get 1/N of link capacity

Jain's fairness index

n is the number of users sharing the path

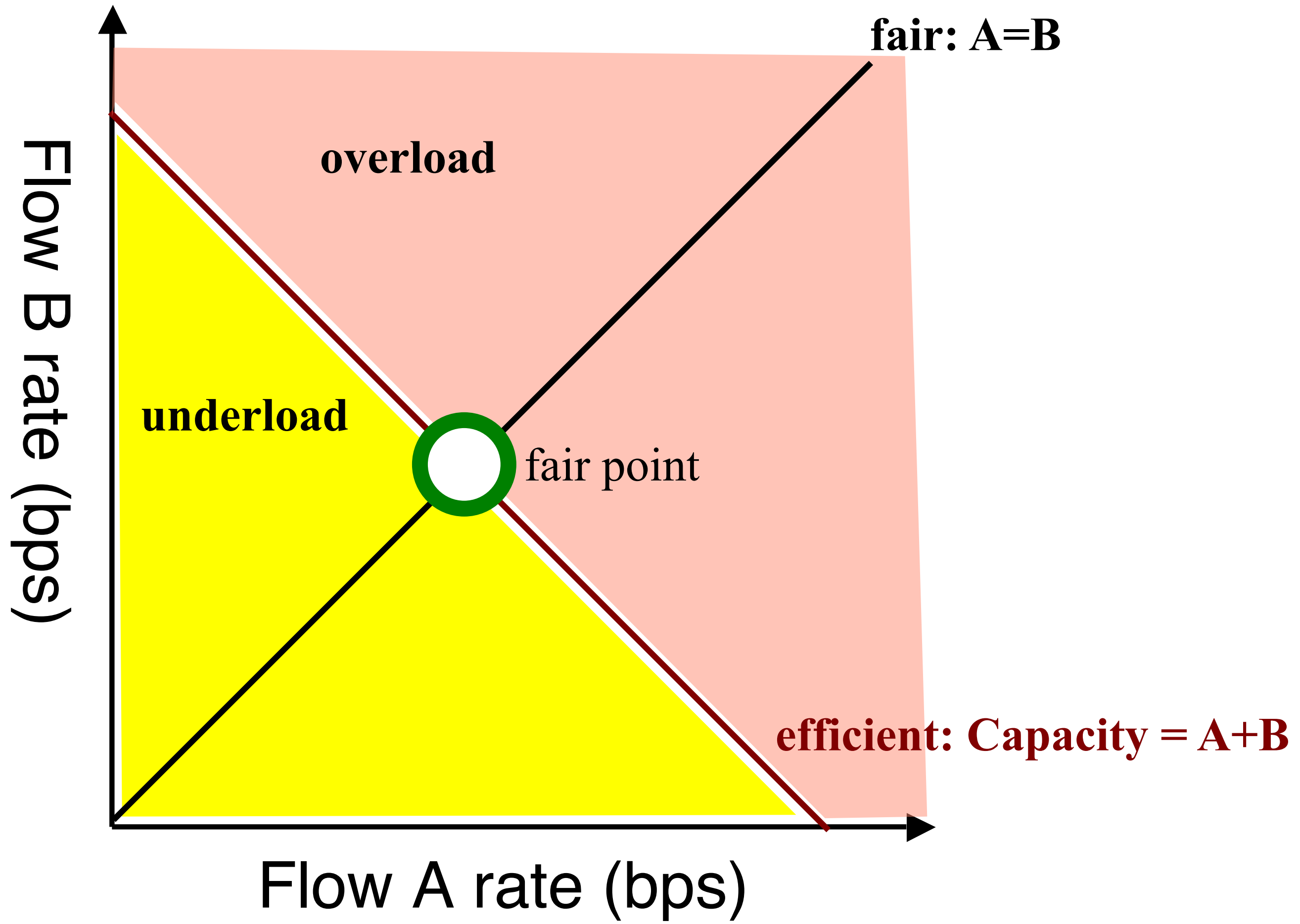
f_i is the network share of i^{th} user

$$F = \frac{(\sum_{i}^{n} f_i)^2}{n \cdot \sum_{i}^{n} f_i^2}$$

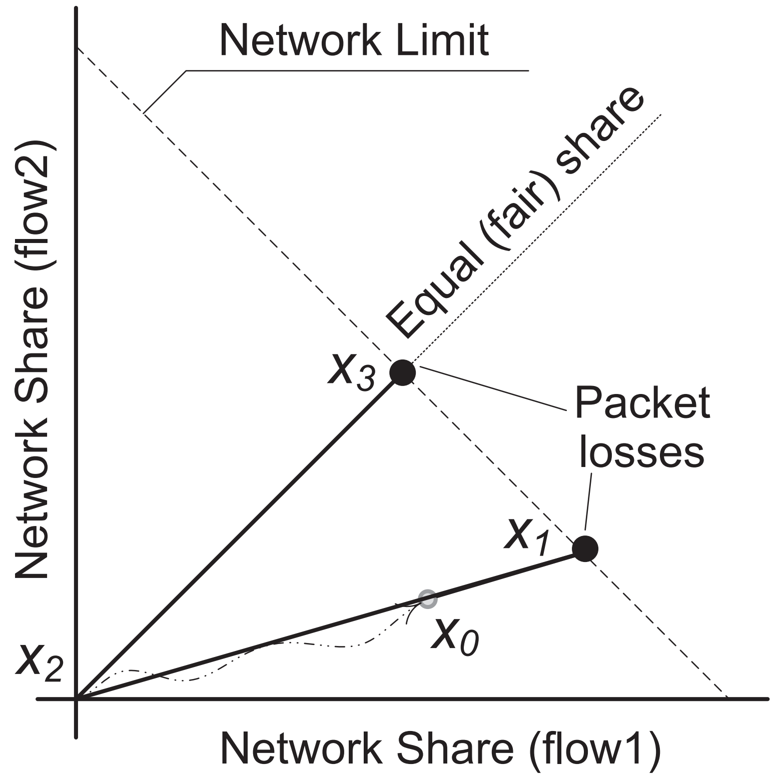


bigger cwnd, larger share TCP flow can "take"

Chiu Jain Phase Plots

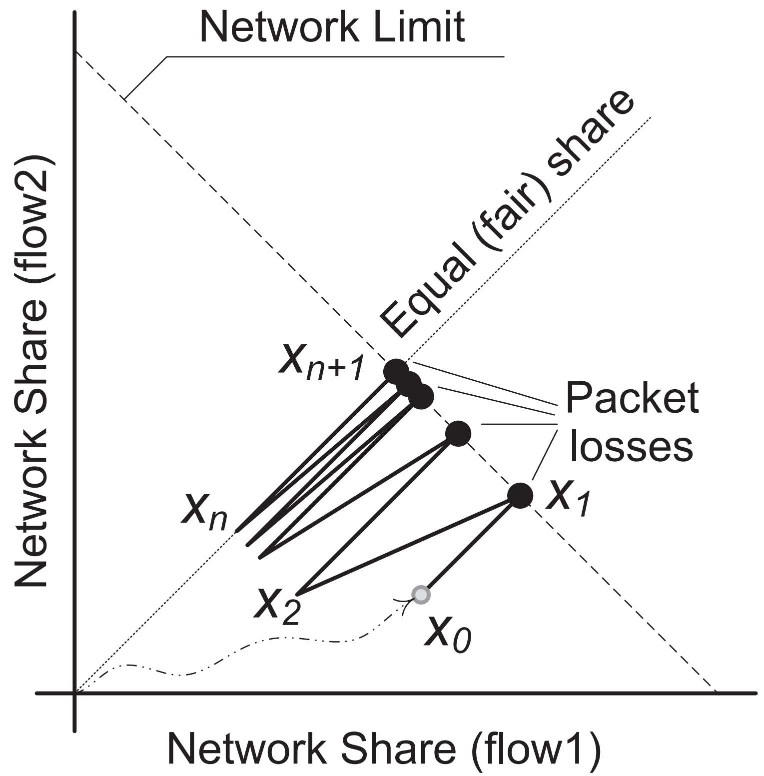


Is Slow Start of TCP Fair?



$x_0 - x_1, \ldots, x_n - x_{n+1}$ multiplicative increase (both flows have the same increase rate of their congestion windows)
$x_1 - x_2$ equalization of the congestion window sizes

Is Congestion Avoidance Is Fair?



$x_0 - x_1, \ldots, x_n - x_{n+1}$ additive increase (both flows have the same increase rate of their congestion windows)
$x_1 - x_2, \ldots, x_{n-1} - x_n$ multiplicative decrease (a flow with the larger congestion window decreases more than a flow with the smaller)

Summary: TCP sender congestion control

State	Event	TCP Sender Action	Commentary
Slow Start (SS)	Received ACK for previously unacked data	CongWin = CongWin + MSS If (CongWin > Threshold) set state to "Congestion Avoidance"	Resulting in a doubling of CongWin every RTT
Congestion Avoidance (CA)	Received ACK for previously unacked data	CongWin = CongWin+MSS * (MSS/CongWin)	Additive increase, resulting in increase of CongWin by 1 MSS every RTT
SS or CA	Loss event detected by 3 duplicate ACK	Threshold = CongWin/2, CongWin = Threshold, Set state to "Congestion Avoidance"	Fast recovery, implementing multiplicative decrease. CongWin will not drop below 1 MSS.
SS or CA	Timeout	Threshold = CongWin/2, CongWin = 1 MSS, Set state to "Slow Start"	Enter slow start
SS or CA	Duplicate ACK	Increment duplicate ACK count for segment being acked	CongWin and Threshold not changed

TCP Throughput

- What's TCP throughout as a function of window size and RTT?
- Ignore slow start: let W = window-size when loss occurs
  - When window is W: throughput = W / RTT
  - Just after loss

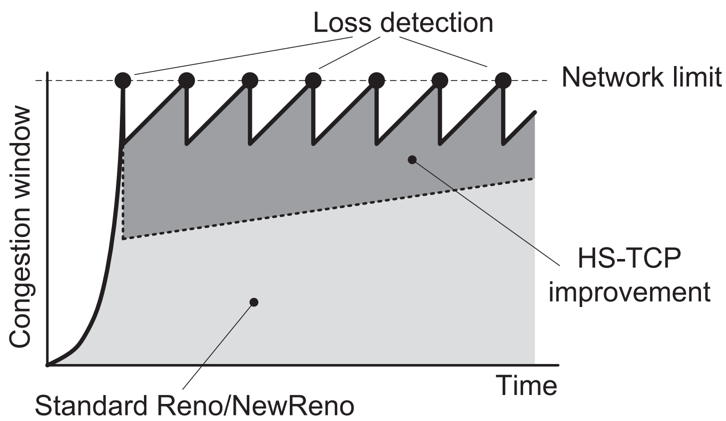
    window → W/2, throughput → W/2RTT
  - Average throughout: 0.75 W/RTT

Why Do We Need Other TCP Variants?

TCP Throughput (Mbps)	RTTs Between Losses	W	P
1	5.5	8.3	0.02
10	55.5	83.3	0.0002
100	555.5	833.3	0.000002
1000	5555.5	8333.3	0.00000002
10000	55555.5	83333.3	0.0000000002

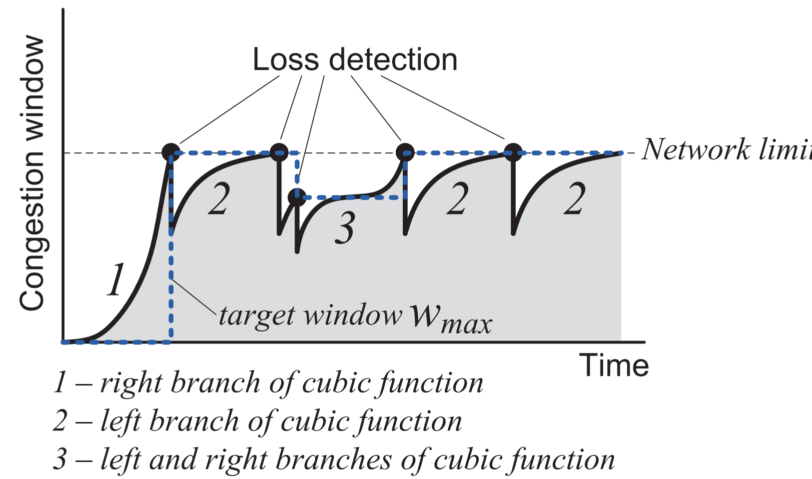
- HS TCP (High-Speed TCP)
  - OS X, available in Linux
- C-TCP (Compound TCP)
  - default on Windows, available in Linux
- CUBIC TCP
  - default in Linux

HS-TCP and CUBIC TCP



HS-TCP

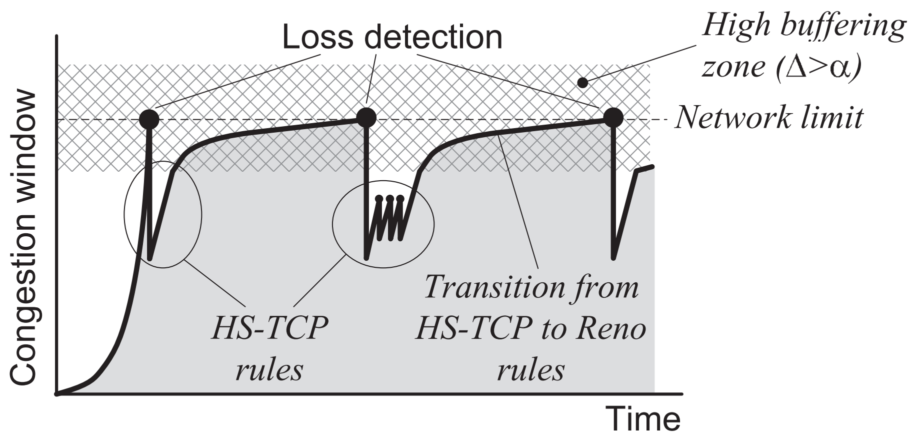
AIMD, but with increased additive increase and decreased multiplicative decrease



CUBIC-TCP

AIMD, but with increased additive increase as a cubic function (fast at first, slow later) and decreased multiplicative decrease

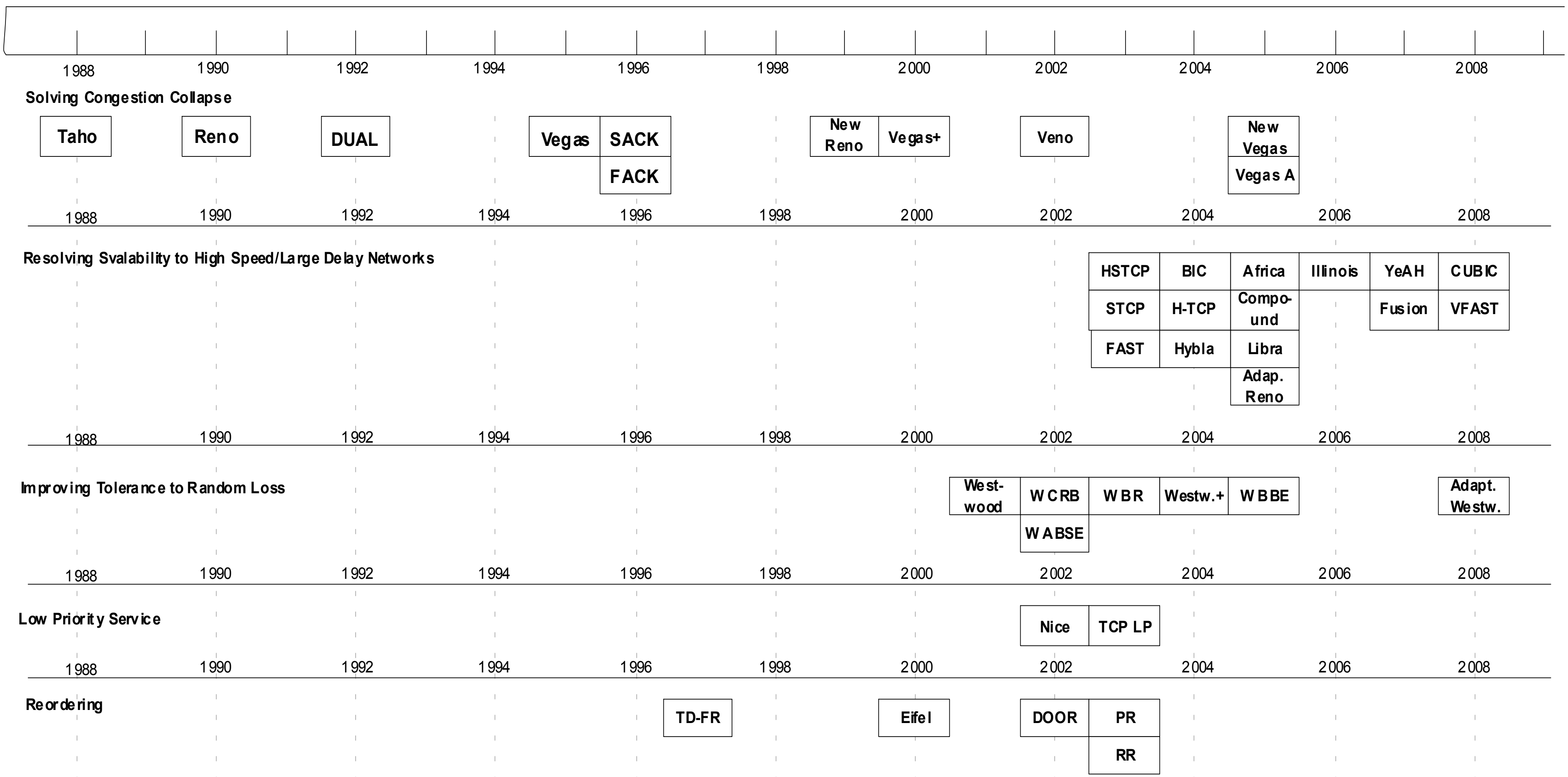
Compound TCP



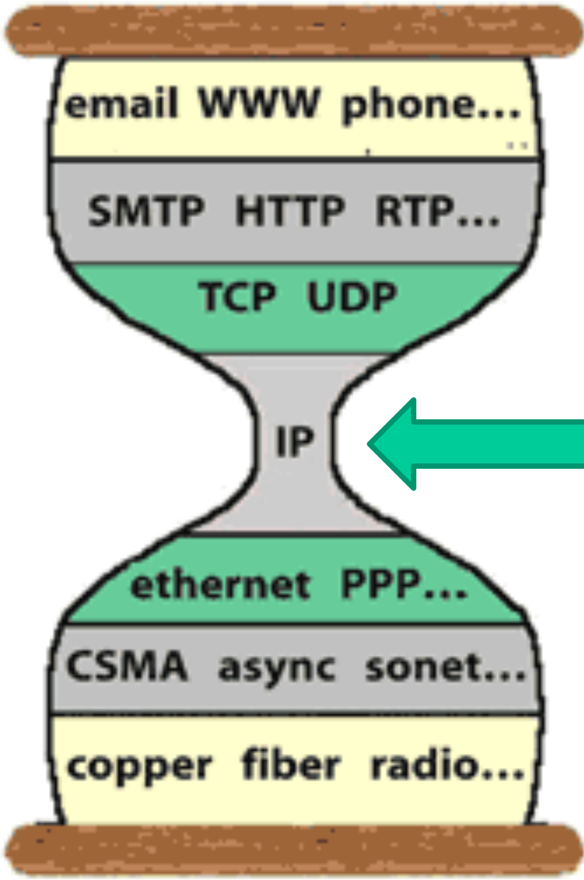
C-TCP

AIMD with multiple zones of different coefficients, depending on estimated buffering in the network

TCP Variants Timeline



Chapter 4: Network Layer



4. 1 Introduction

4.2 Virtual circuit and datagram networks

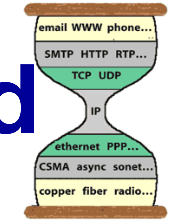
4.3 What's inside a router

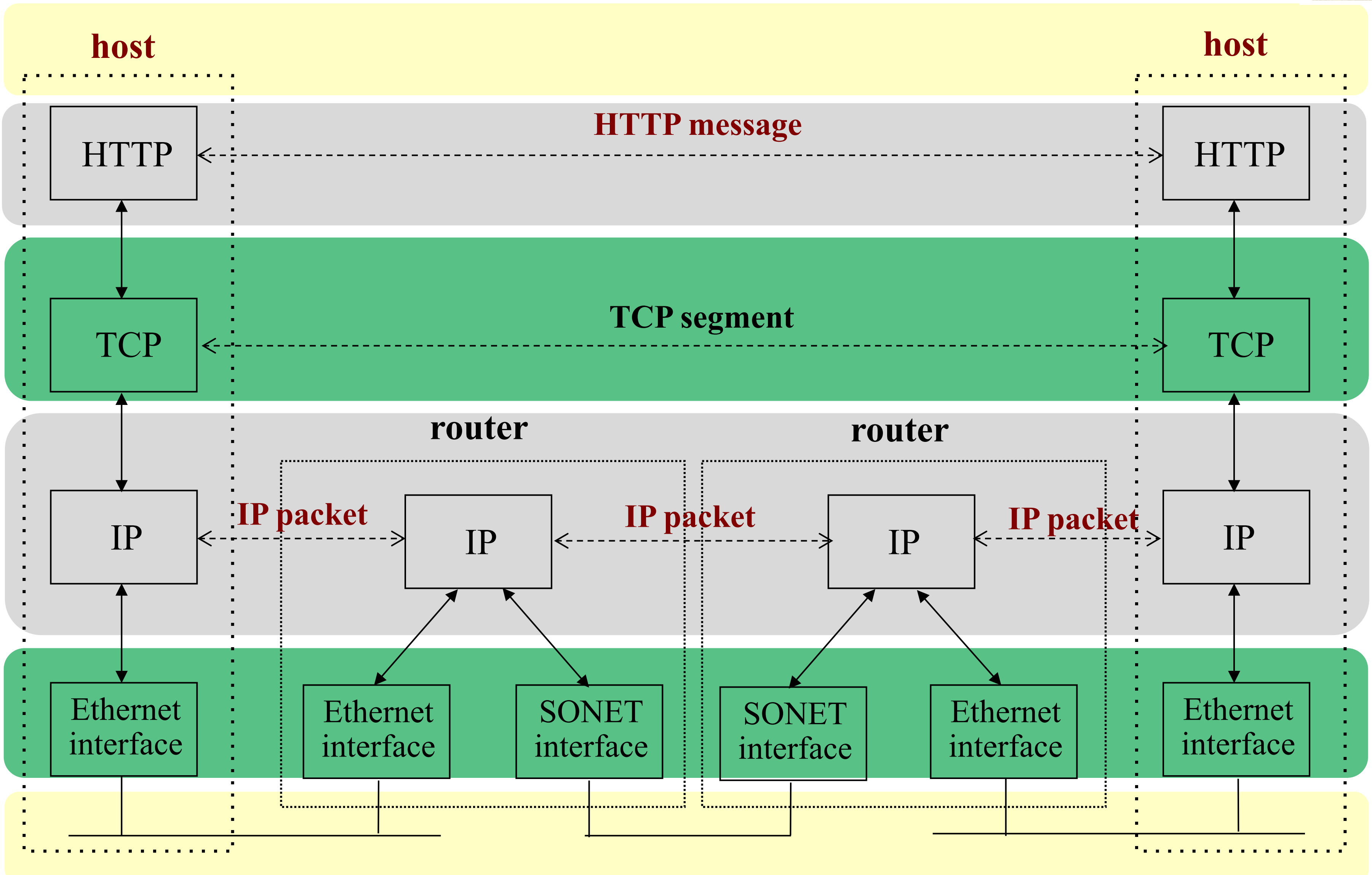
4.4 IP: Internet Protocol

- IP datagram format
- IPv4 addressing
- ICMP
- IPv6

This and next week

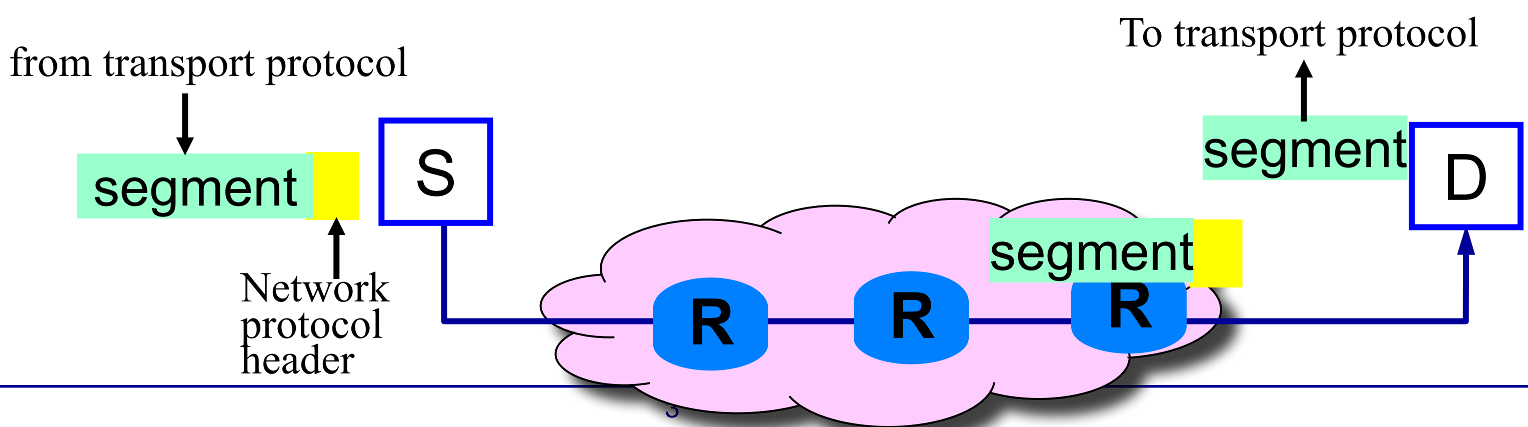
Always keep the big picture in mind





Network layer

- Network layer protocols: in *every* host and router
- Moving data *segments* from sending to receiving host
  - Routing: calculate the best path to each destination
  - Forwarding: use FIB to move packets towards destinations
- Source host: encapsulates segments into *packets*
- Destination host: delivers segments to transport layer
- Each router along the path: examines the header field of a packet to determine where to forward it



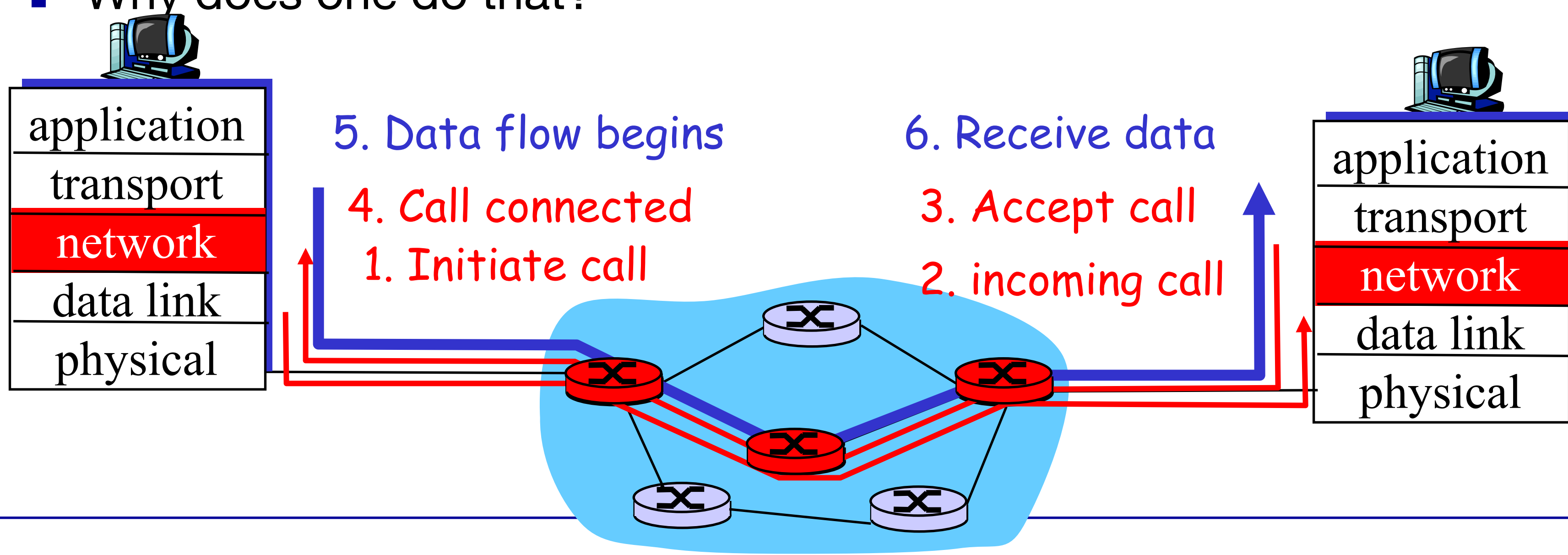
Network layer: Connection vs. connection-less delivery

- *Virtual Circuit:* network sets up a connection, then delivers packets over the connection
  - works in a way very much like telephone circuit
- *Datagram*: Each packet finds its way to destination independently
- analogous to TCP vs. UDP at transport-layer, *except* that a given network provides one or the other, but not both (as in transport layer)

Virtual circuit (VC) Network

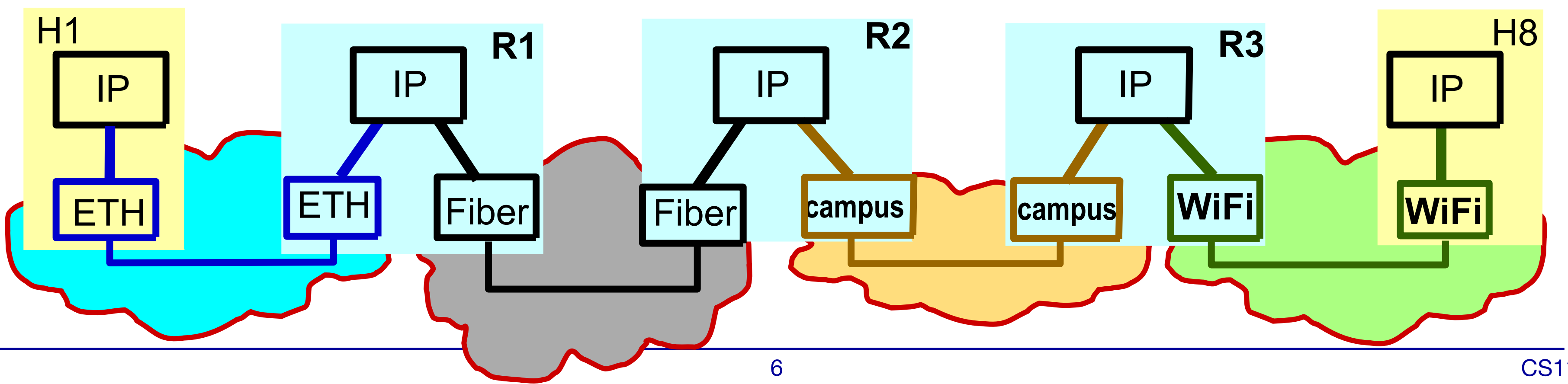
FYI

- Use a signaling protocol to setup connection first
- every router on the path maintains "state" for each passing virtual connection
  - link, router resources (bandwidth, buffers) allocated to the VC
- each packet carries VC identifier (*not destination host address*)
- VC number changes on each link
  - Why does one do that?

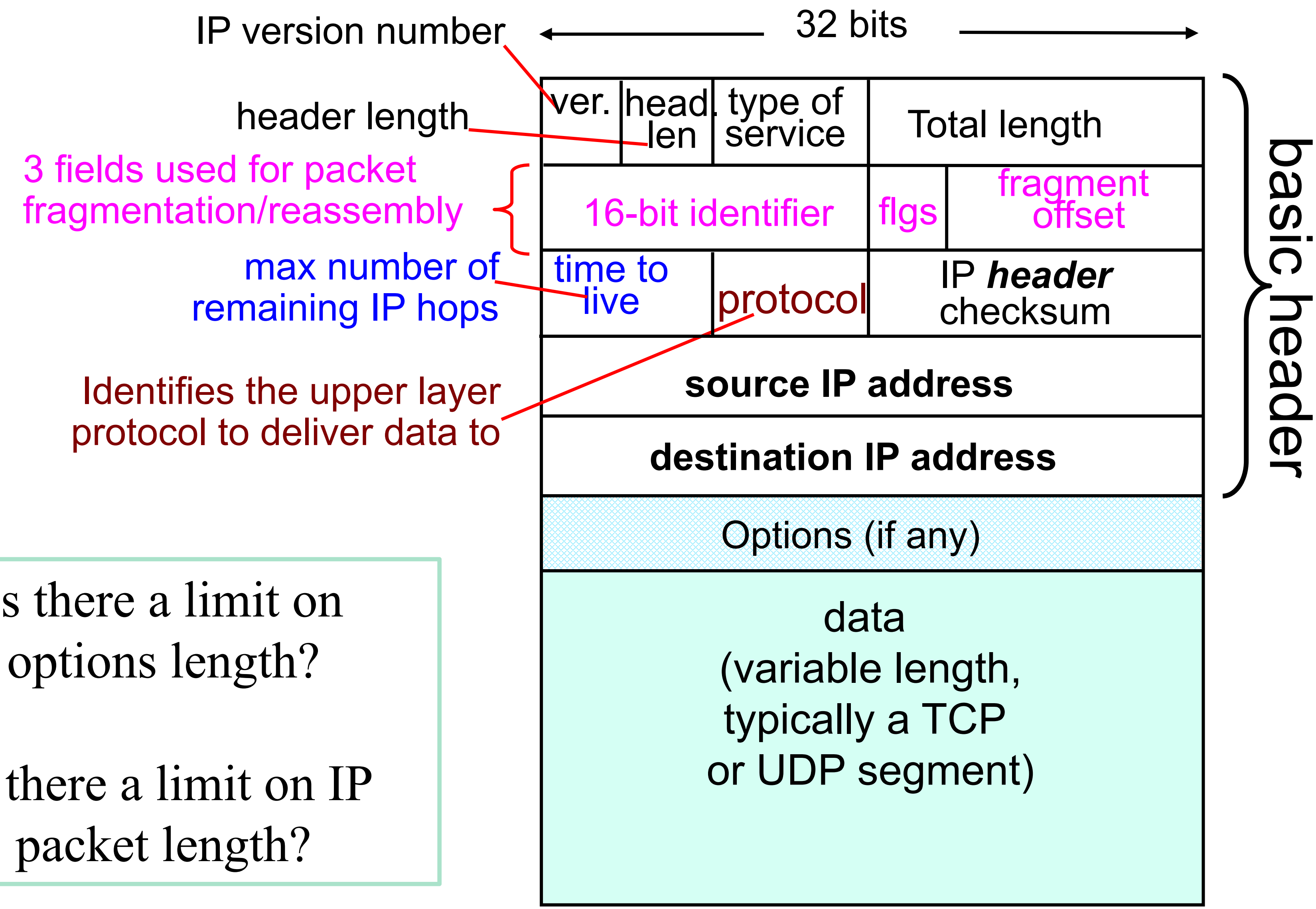


Internet: A Datagram Network

- hosts are connected to subnets
- subnets are interconnected by routers
- All hosts and routers speak IP
- IP provides two basic functions
  - Assigning globally unique addresses to all connected points
  - Datagram delivery from source to destination hosts



IP datagram format

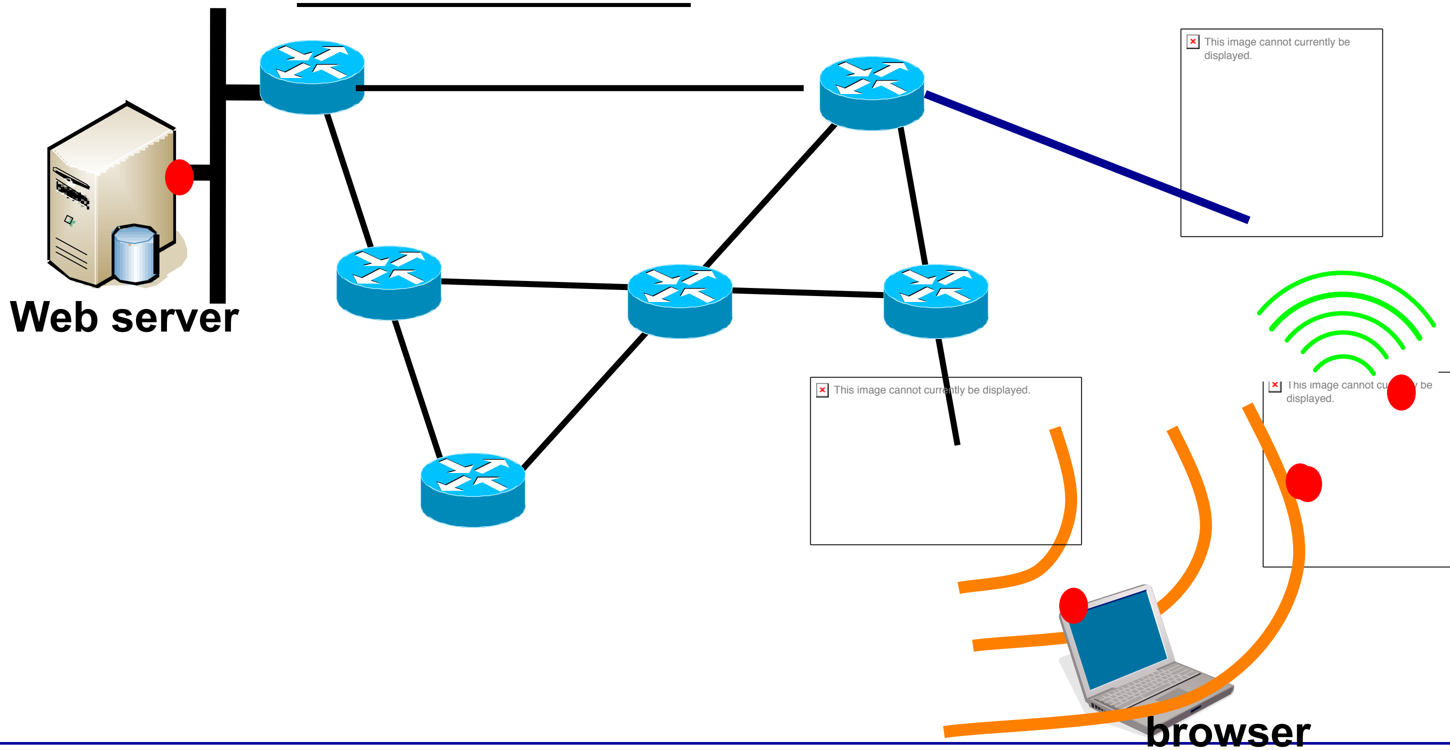


Is there a limit on options length?

Is there a limit on IP packet length?

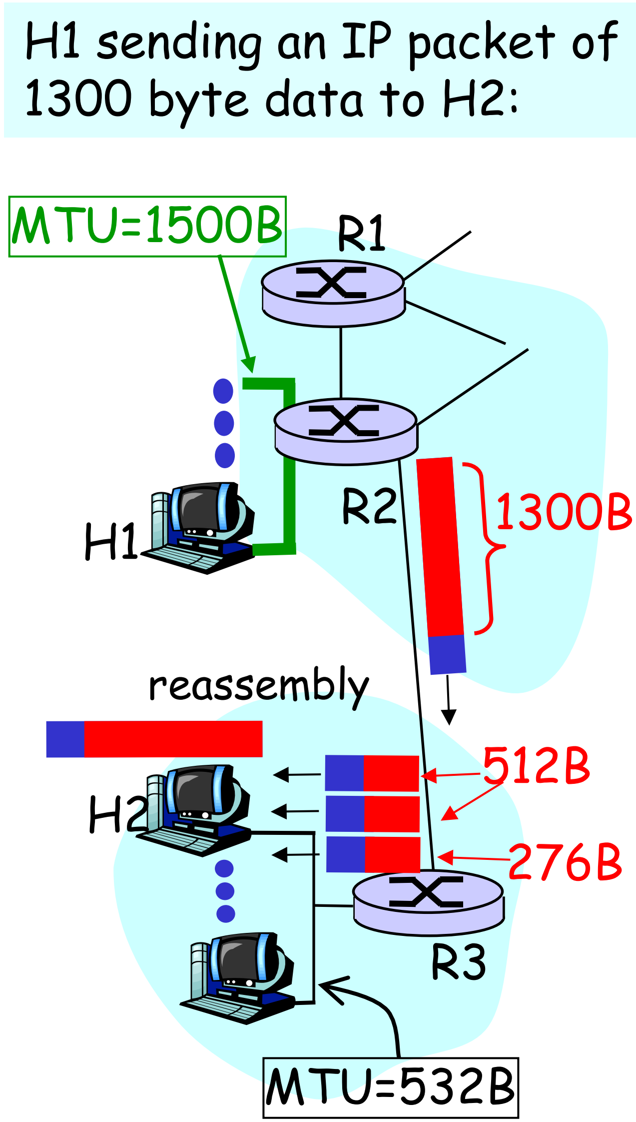
Two further clarifications

- IP assigns a globally unique IP address to each attachment point
- When delivering a packet that is too big for next hop: IP routers fragment the packet and then reassemble it at destination host



IP Fragmentation & Reassembly

- Different subnets may have different MTUs (Maximum Transmission Unit)
- Sending host uses its local MTU size
- Routers *fragment* IP packets if the next link has a smaller MTU
  - chop packets to the MTU size of next link
  - further fragmentation down the path possible
- packet fragments are reassembled at destination host



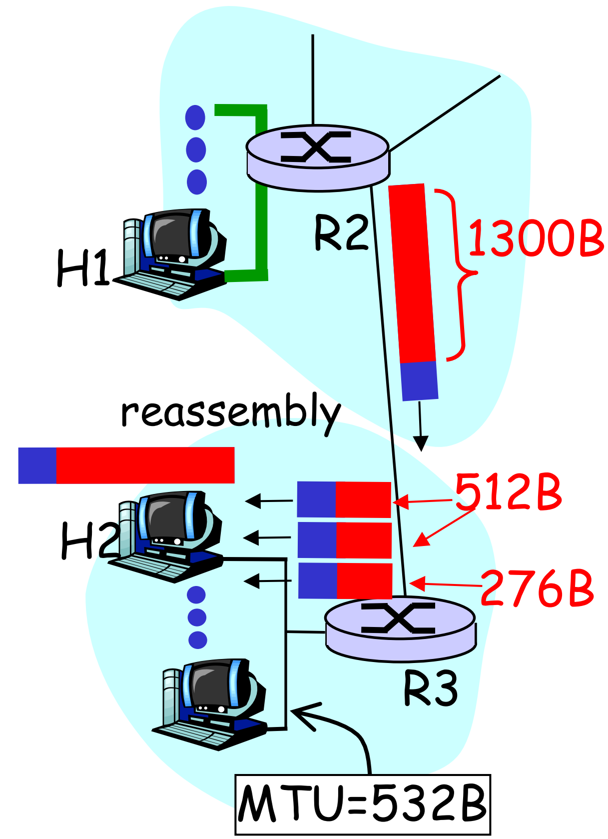
IP Fragmentation: An example

4	5	TOS	1320
7394		0 0 0	0
rest of the IP header			
data (1300 bytes)			

4	5	TOS	532
7394		0 0 1	0
rest of the IP header			
data (512 bytes)			

4	5	TOS	532
7394		0 0 1	64
rest of the IP header			
data (512 bytes)			

4	5	TOS	296
7394		0 0 0	128
rest of the IP header			
data (276 bytes)			



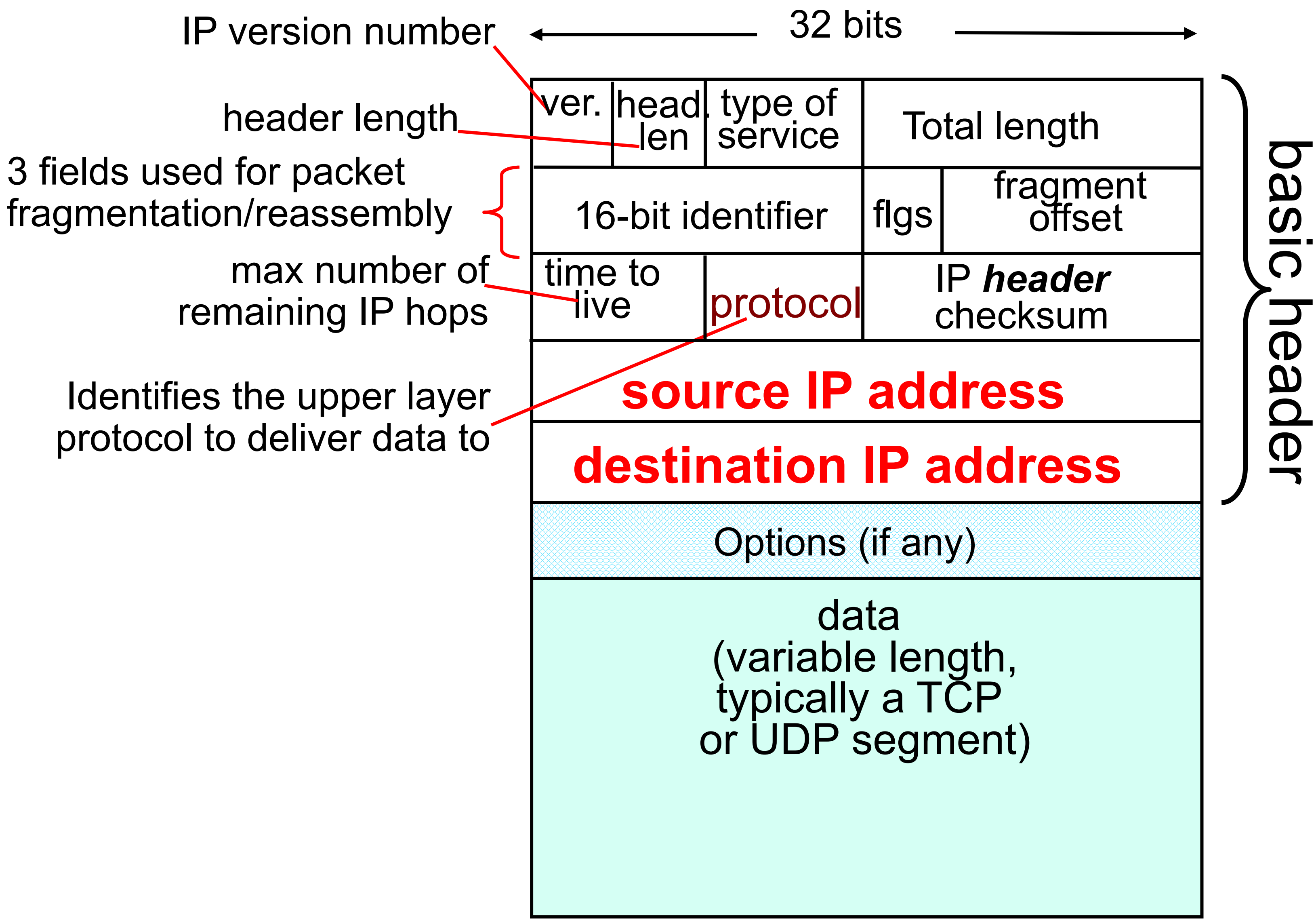
Destination host: examming IP header fields:

identifier: tell all pieces in the same packet

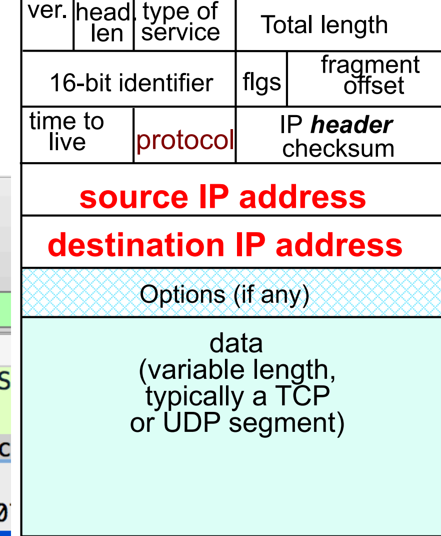
flag bit: *if* MF=0, the last fragment

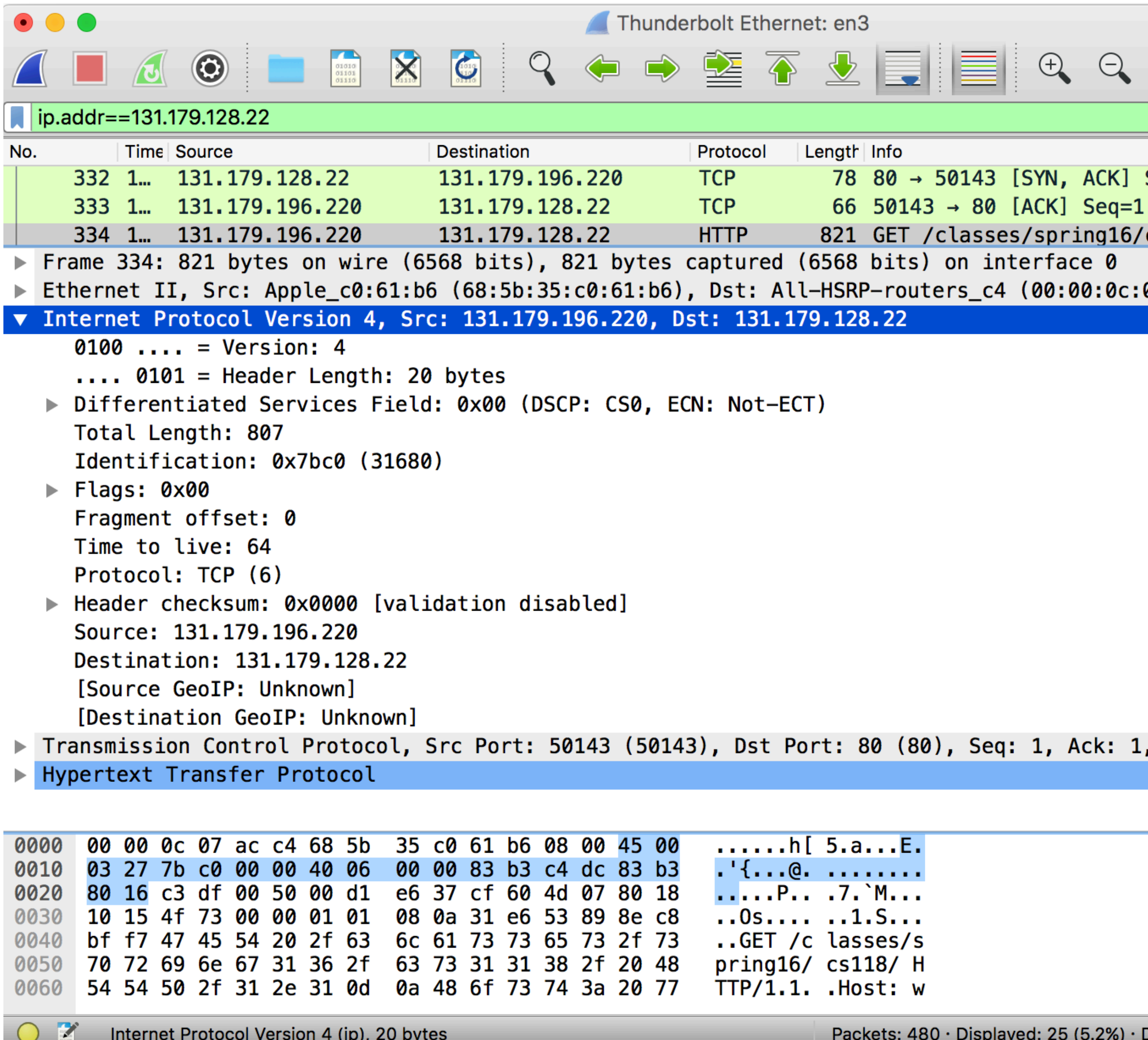
offset: tell whether there are holes missing in the middle

IP datagram format



IP Packets in Wireshark





IPv4 Address structure

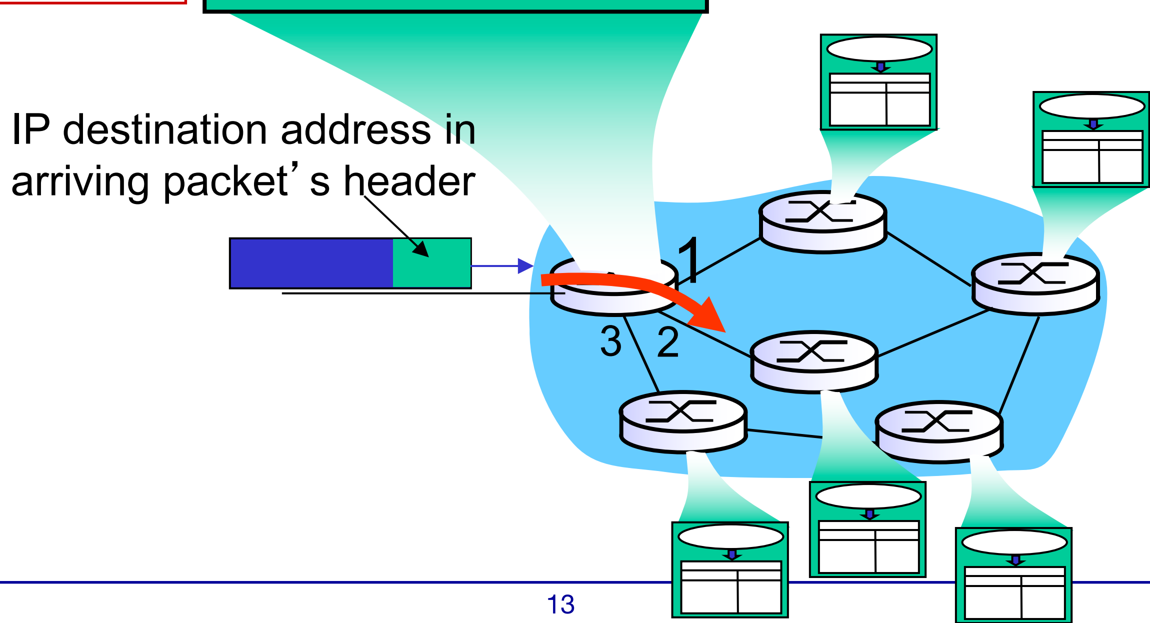
32-bits, uniquely identifies a host or router *interface*

– *interface:* connecting point between host/router and physical link

$$173.1.1.10 = \underbrace{10101101}_{173}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00001010}_{10}$$

4 billion IP addresses, so rather than list individual destination address, list ***range*** of addresses (aggregate table entries)

local forwarding table	
dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1



How to define IP ranges in routing table?

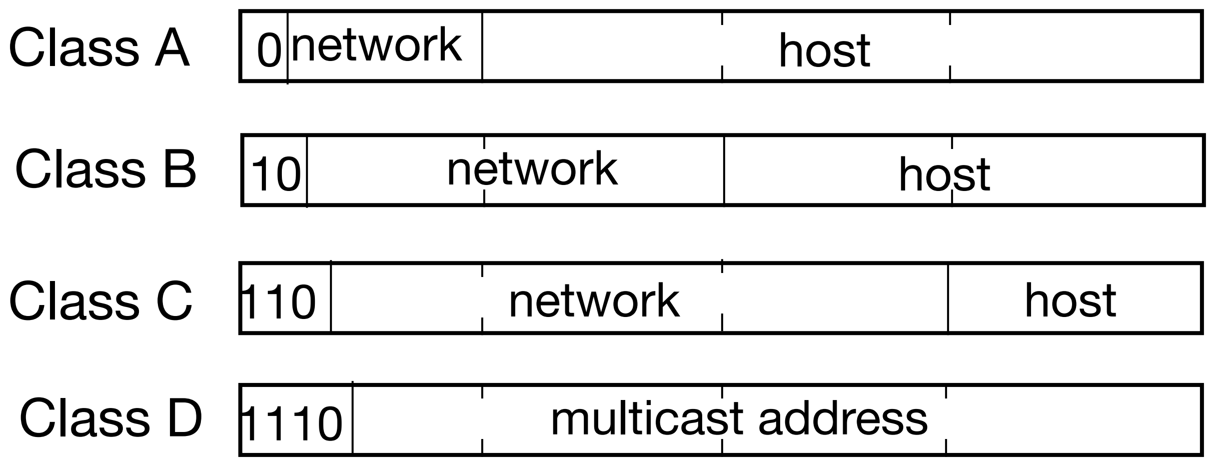
- Should be compact
- Should be fast and easy to process

Designate a number of bits for the "network".

Routing table will contain only "networks"

How many bits for network ID

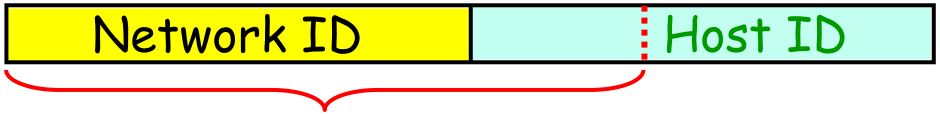
- Original IP design (RFC791): class-based address



Calculate the address ranges

Today's IPv4 Address Structure

- Two changes added later:
  - CIDR: Classless InterDomain Routing (today)
    - network portion can take any arbitrary number of bits



  - Subnetting: An organization gets one address block, then split the host part into two parts: subnet and host part

Classless **I**nter**D**omain **R**outing

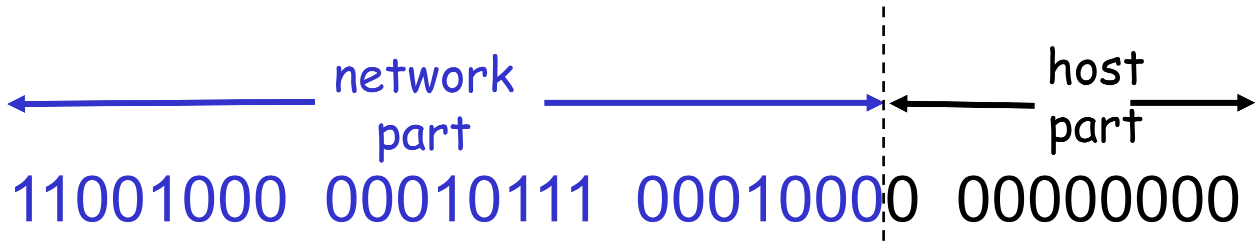
network part | host part

11001000 00010111 00010000 | 0 00000000

- Internet Service Providers (ISPs), and some large user sites, get blocks of IP addresses from the Regional Internet Registries (RIRs)
- Internet customers get a sub-block from their ISP’s address block

ISP's block	11001000 00010111 00010000 00000000
Organization 0	11001000 00010111 0001**000**0 00000000
Organization 1	11001000 00010111 0001**001**0 00000000
Organization 2	11001000 00010111 0001**010**0 00000000
...	....
Organization 7	11001000 00010111 0001**111**0 00000000

CIDR Address Format



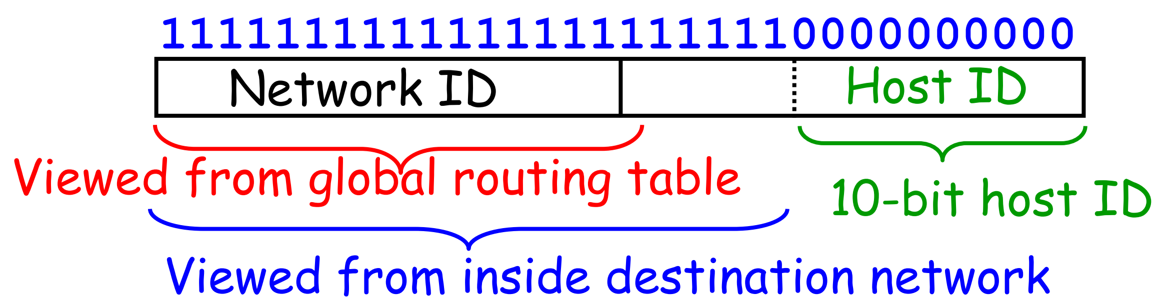
- a.b.c.d/x, x = #bits in network ID portion of the address
- address: a.b.c.d, network mask: $2^{32} - 2^{(32-x)}$

200.23.16.0/23 **address 200.23.16.0, netmask 255.255.254.0**

important

IP Subnet

- subnet mask: indicates the portion of the address that is considered as "network ID" *by the local site*
  - Does not need to align with byte boundary



- subnets are invisible outside the local site
  - <u>backbone routers only know how to forward packets to the networkID</u>
  - Within the organization:
    - routers store: [subnet, mask, next hop]
    - Each host is configured with an IP address and a subnet mask

Special Addresses

- 0.0.0.0/8
  - “this network”
- 255.255.255.255/32
  - broadcast address of “this network”
- first address of the network (e.g., 192.168.1.0 for 192.168.1.0/24)
  - network address (cannot be used for end-hosts)
- last address of the network (e.g., 192.168.1.255 for 192.168.1.0/24)
  - broadcast address for the network