

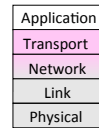
Operating Systems and Networks

Network Lecture 10: Congestion Control

Adrian Perrig
 Network Security Group
 ETH Zürich

Where we are in the Course

- More fun in the Transport Layer!
 - The mystery of congestion control
 - Depends on the Network layer too

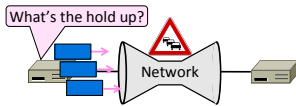


Computer Networks

2

Topic

- Understanding congestion, a “traffic jam” in the network
 - Later we will learn how to control it

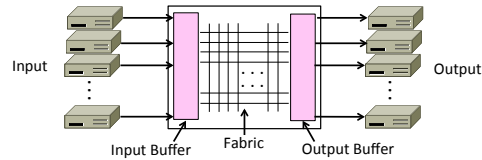


Computer Networks

3

Nature of Congestion

- Routers/switches have internal buffering for contention

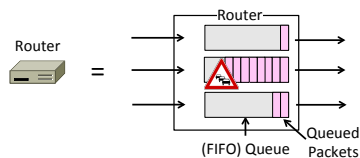


Computer Networks

4

Nature of Congestion (2)

- Simplified view of per port output queues
 - Typically FIFO (First In First Out), discard when full



Computer Networks

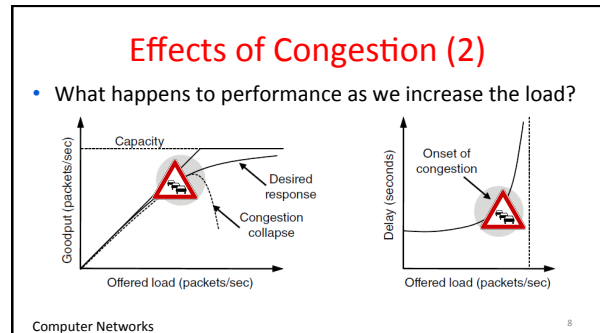
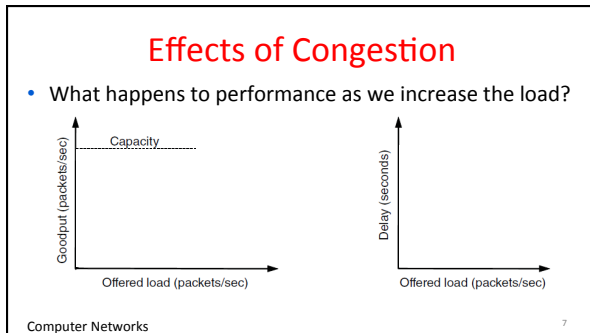
5

Nature of Congestion (3)

- Queues help by absorbing bursts when input > output rate
- But if input > output rate persistently, queue will overflow
 - This is congestion
- Congestion is a function of the traffic patterns – can occur even if every link have the same capacity

Computer Networks

6



Effects of Congestion (3)

- As offered load rises, congestion occurs as queues begin to fill:
 - Delay and loss rise sharply with more load
 - Throughput falls below load (due to loss)
 - Goodput may fall below throughput (due to spurious retransmissions)
- None of the above is good!
 - Want to operate network just before the onset of congestion

Computer Networks 9

Bandwidth Allocation

- Important task for network is to allocate its capacity to senders
 - Good allocation is efficient and fair
- **Efficient** means most capacity is used but there is no congestion
- **Fair** means every sender gets a reasonable share of the network

Computer Networks 10

Bandwidth Allocation (2)

- Key observation:
 - In an effective solution, Transport and Network layers must work together
- Network layer witnesses congestion
 - Only it can provide direct feedback
- Transport layer causes congestion
 - Only it can reduce offered load

Computer Networks 11

Bandwidth Allocation (3)

- Why is it hard? (Just split equally!)
 - Number of senders and their offered load is constantly changing
 - Senders may lack capacity in different parts of the network
 - Network is distributed; no single party has an overall picture of its state

Computer Networks 12

Bandwidth Allocation (4)

- Solution context:
 - Senders adapt concurrently based on their own view of the network
 - Design this adaption so the network usage as a whole is efficient and fair
 - Adaption is continuous since offered loads continue to change over time

Computer Networks

13

Topics

- Nature of congestion
 - Fair allocations
 - AIMD control law
 - TCP Congestion Control history
 - ACK clocking
 - TCP Slow-start
 - TCP Fast Retransmit/Recovery
 - Congestion Avoidance (ECN)
- } This time

Computer Networks

14

Fairness of Bandwidth Allocation (§6.3.1)

- What's a "fair" bandwidth allocation?
 - The max-min fair allocation



Computer Networks

15

Recall

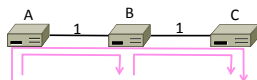
- We want a good bandwidth allocation to be fair and efficient
 - Now we learn what fair means
- Caveat: in practice, efficiency is more important than fairness

Computer Networks

16

Efficiency vs. Fairness

- Cannot always have both!
 - Example network with traffic $A \rightarrow B$, $B \rightarrow C$ and $A \rightarrow C$
 - How much traffic can we carry?

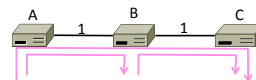


Computer Networks

17

Efficiency vs. Fairness (2)

- If we care about fairness:
 - Give equal bandwidth to each flow
 - $A \rightarrow B$: $\frac{1}{2}$ unit, $B \rightarrow C$: $\frac{1}{2}$, and $A \rightarrow C$: $\frac{1}{2}$
 - Total traffic carried is $1 \frac{1}{2}$ units

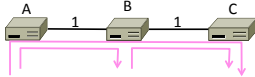


Computer Networks

18

Efficiency vs. Fairness (3)

- If we care about efficiency:
 - Maximize total traffic in network
 - $A \rightarrow B$: 1 unit, $B \rightarrow C$: 1, and $A \rightarrow C$, 0
 - Total traffic rises to 2 units!



Computer Networks

19

The Slippery Notion of Fairness

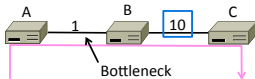
- Why is “equal per flow” fair anyway?
 - $A \rightarrow C$ uses more network resources (two links) than $A \rightarrow B$ or $B \rightarrow C$
 - Host A sends two flows, B sends one
- Not productive to seek exact fairness
 - More important to avoid starvation
 - “Equal per flow” is good enough

Computer Networks

20

Generalizing “Equal per Flow”

- Bottleneck for a flow of traffic is the link that limits its bandwidth
 - Where congestion occurs for the flow
 - For $A \rightarrow C$, link $A-B$ is the bottleneck

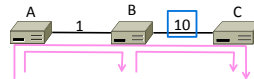


Computer Networks

21

Generalizing “Equal per Flow” (2)

- Flows may have different bottlenecks
 - For $A \rightarrow C$, link $A-B$ is the bottleneck
 - For $B \rightarrow C$, link $B-C$ is the bottleneck
 - Can no longer divide links equally ...



Computer Networks

22

Max-Min Fairness

- Intuitively, flows bottlenecked on a link get an equal share of that link
- Max-min fair allocation is one that:
 - Increasing the rate of one flow will decrease the rate of a smaller flow
 - This “maximizes the minimum” flow

Computer Networks

23

Max-Min Fairness (2)

- To find it given a network, imagine “pouring water into the network”
 1. Start with all flows at rate 0
 2. Increase the flows until there is a new bottleneck in the network
 3. Hold fixed the rate of the flows that are bottlenecked
 4. Go to step 2 for any remaining flows

Computer Networks

24

Max-Min Example

- Example: network with 4 flows, links equal bandwidth
 - What is the max-min fair allocation?

Computer Networks 25

Max-Min Example (2)

- When rate=1/3, flows B, C, and D bottleneck R4—R5
 - Fix B, C, and D, continue to increase A

Computer Networks 26

Max-Min Example (3)

- When rate=2/3, flow A bottlenecks R2—R3. Done.

Computer Networks 27

Max-Min Example (4)

- End with A=2/3, B, C, D=1/3, and R2—R3, R4—R5 full
 - Other links have extra capacity that can't be used

Computer Networks 28

Adapting over Time

- Allocation changes as flows start and stop

Computer Networks 29

Adapting over Time (2)

Computer Networks 30

Additive Increase Multiplicative Decrease (AIMD) (§6.3.2)

- Bandwidth allocation models
 - Additive Increase Multiplicative Decrease (AIMD) control law



Computer Networks

31

Recall

- Want to allocate capacity to senders
 - Network layer provides feedback
 - Transport layer adjusts offered load
 - A good allocation is efficient and fair
- How should we perform the allocation?
 - Several different possibilities ...

Computer Networks

32

Bandwidth Allocation Models

- Open loop versus closed loop
 - Open: reserve bandwidth before use
 - Closed: use feedback to adjust rates
- Host versus Network support
 - Who sets/enforces allocations?
- Window versus Rate based
 - How is allocation expressed?

TCP is a closed loop, host-driven, and window-based

Computer Networks

33

Bandwidth Allocation Models (2)

- We'll look at closed-loop, host-driven, and window-based too
- Network layer returns feedback on current allocation to senders
 - At least tells if there is congestion
- Transport layer adjusts sender's behavior via window in response
 - How senders adapt is a control law

Computer Networks

34

Additive Increase Multiplicative Decrease

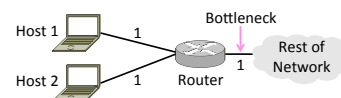
- AIMD is a control law hosts can use to reach a good allocation
 - Hosts additively increase rate while network is not congested
 - Hosts multiplicatively decrease rate when congestion occurs
 - Used by TCP ☺
- Let's explore the AIMD game ...

Computer Networks

35

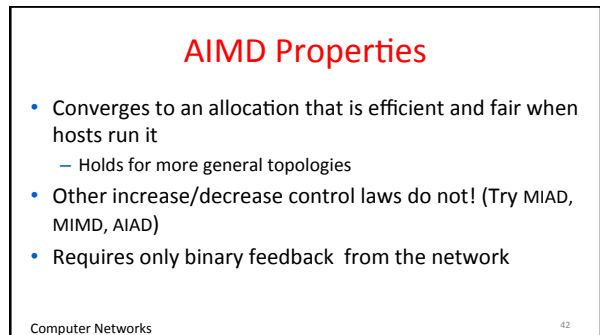
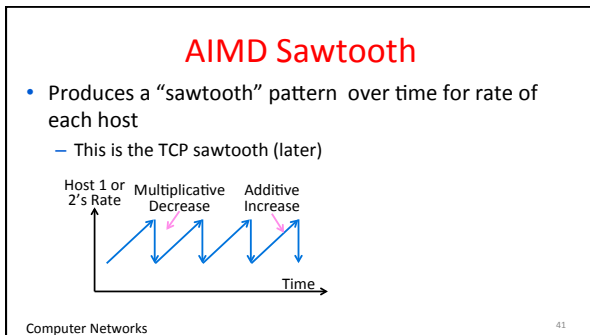
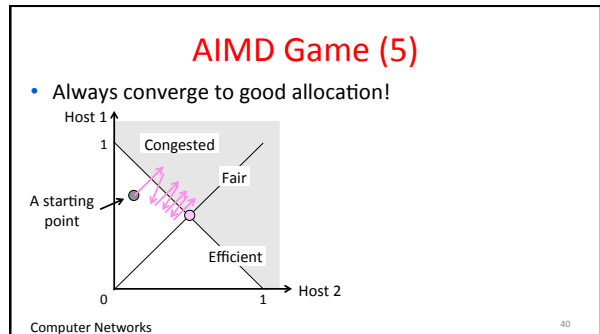
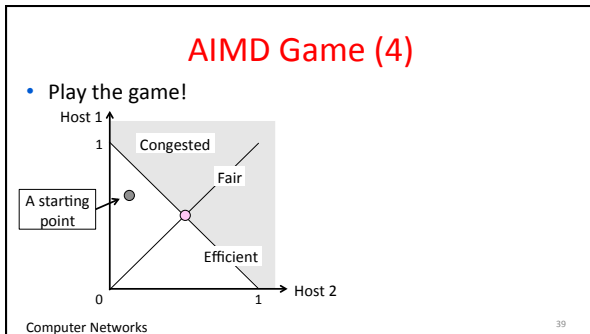
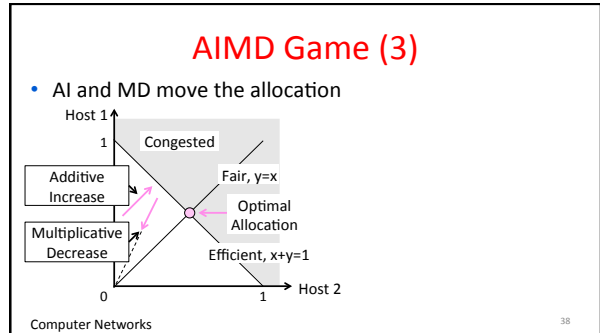
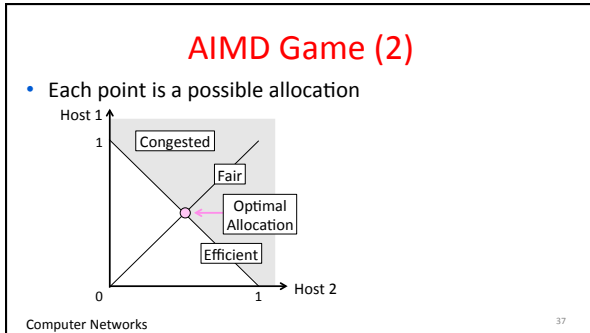
AIMD Game

- Hosts 1 and 2 share a bottleneck
 - But do not talk to each other directly
- Router provides binary feedback
 - Tells hosts if network is congested



Computer Networks

36



Feedback Signals

- Several possible signals, with different pros/cons
 - We'll look at classic TCP that uses packet loss as a signal

Signal	Example Protocol	Pros / Cons
Packet loss	TCP NewReno Cubic TCP (Linux)	Hard to get wrong Hear about congestion late
Packet delay	Compound TCP (Windows)	Hear about congestion early Need to infer congestion
Router indication	TCPs with Explicit Congestion Notification	Hear about congestion early Require router support

Computer Networks 43

History of TCP Congestion Control (§6.5.10)

- The story of TCP congestion control
 - Collapse, control, and diversification

Computer Networks 44

Congestion Collapse in the 1980s

- Early TCP used a fixed size sliding window (e.g., 8 packets)
 - Initially fine for reliability
- But something strange happened as the ARPANET grew
 - Links stayed busy but transfer rates fell by orders of magnitude!

Computer Networks 45

Congestion Collapse (2)

- Queues became full, retransmissions clogged the network, and goodput fell

Computer Networks 46

Van Jacobson (1950—)

- Widely credited with saving the Internet from congestion collapse in the late 80s
 - Introduced congestion control principles
 - Practical solutions (TCP Tahoe/Reno)
- Much other pioneering work:
 - Tools like traceroute, tcpdump, pathchar
 - IP header compression, multicast tools

Source: Wikipedia (public domain)

Computer Networks 47

TCP Tahoe/Reno

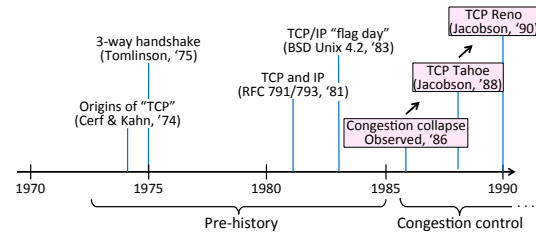
- Avoid congestion collapse without changing routers (or even receivers)
- Idea is to fix timeouts and introduce a congestion window (cwnd) over the sliding window to limit queues/loss
- TCP Tahoe/Reno implements AIMD by adapting cwnd using packet loss as the network feedback signal

Computer Networks 48

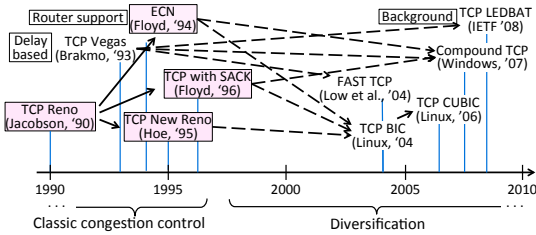
TCP Tahoe/Reno (2)

- TCP behaviors we will study:
 - ACK clocking
 - Adaptive timeout (mean and variance)
 - Slow-start
 - Fast Retransmission
 - Fast Recovery
- Together, they implement AIMD

TCP Timeline

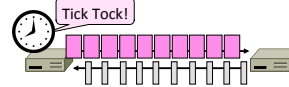


TCP Timeline (2)



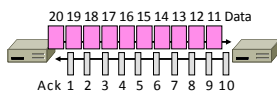
TCP Ack Clocking (§6.5.10)

- The self-clocking behavior of sliding windows, and how it is used by TCP
 - The "ACK clock"



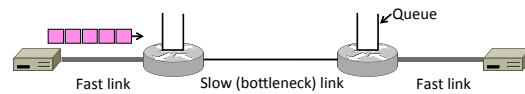
Sliding Window ACK Clock

- Each in-order ACK advances the sliding window and lets a new segment enter the network
 - ACKs "clock" data segments



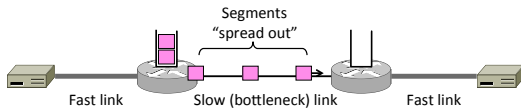
Benefit of ACK Clocking

- Consider what happens when sender injects a burst of segments into the network



Benefit of ACK Clocking (2)

- Segments are buffered and spread out on slow link

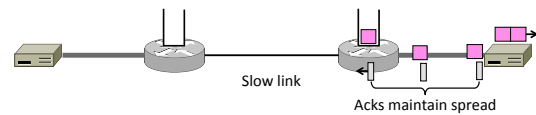


Computer Networks

55

Benefit of ACK Clocking (3)

- ACKs maintain the spread back to the original sender

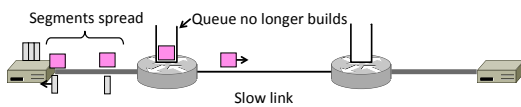


Computer Networks

56

Benefit of ACK Clocking (4)

- Sender clocks new segments with the spread
 - Now sending at the bottleneck link without queuing!



Computer Networks

57

Benefit of ACK Clocking (4)

- Helps the network run with low levels of loss and delay!
- The network has smoothed out the burst of data segments
- ACK clock transfers this smooth timing back to the sender
- Subsequent data segments are not sent in bursts so do not queue up in the network

Computer Networks

58

TCP Uses ACK Clocking

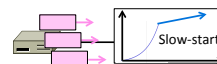
- TCP uses a sliding window because of the value of ACK clocking
- Sliding window controls how many segments are inside the network
 - Called the congestion window, or cwnd
 - Rate is roughly $cwnd/RTT$
- TCP only sends small bursts of segments to let the network keep the traffic smooth

Computer Networks

59

TCP Slow Start (§6.5.10)

- How TCP implements AIMD, part 1
 - "Slow start" is a component of the AI portion of AIMD



Computer Networks

60

Recall

- We want TCP to follow an AIMD control law for a good allocation
- Sender uses a congestion window or cwnd to set its rate ($\approx cwnd/RTT$)
- Sender uses packet loss as the network congestion signal
- Need TCP to work across a very large range of rates and RTTs

Computer Networks

61

TCP Startup Problem

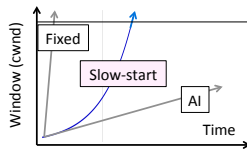
- We want to quickly near the right rate, $cwnd_{IDEAL}$, but it varies greatly
 - Fixed sliding window doesn't adapt and is rough on the network (loss!)
 - AI with small bursts adapts cwnd gently to the network, but might take a long time to become efficient

Computer Networks

62

Slow-Start Solution

- Start by doubling cwnd every RTT
 - Exponential growth (1, 2, 4, 8, 16, ...)
 - Start slow, quickly reach large values



Computer Networks

63

Slow-Start Solution (2)

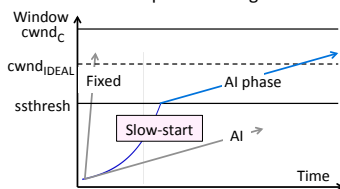
- Eventually packet loss will occur when the network is congested
 - Loss timeout tells us cwnd is too large
 - Next time, switch to AI beforehand
 - Slowly adapt cwnd near right value
- In terms of cwnd:
 - Expect loss for $cwnd_C \approx 2BD + queue$
 - Use $ssthresh = cwnd_C/2$ to switch to AI

Computer Networks

64

Slow-Start Solution (3)

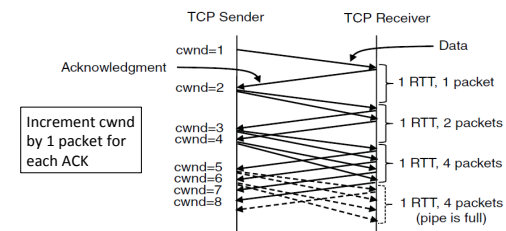
- Combined behavior, after first time
 - Most time spend near right value



Computer Networks

65

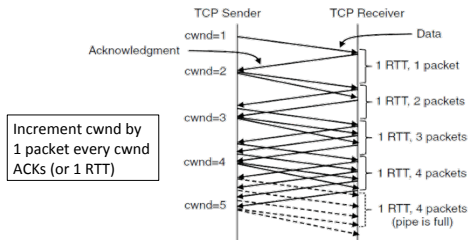
Slow-Start (Doubling) Timeline



Computer Networks

66

Additive Increase Timeline



Computer Networks

67

TCP Tahoe (Implementation)

- Initial slow-start (doubling) phase
 - Start with $cwnd = 1$ (or small value)
 - $cwnd += 1$ packet per ACK
- Later Additive Increase phase
 - $cwnd += 1/cwnd$ packets per ACK
 - Roughly adds 1 packet per RTT
- Switching threshold (initially infinity)
 - Switch to AI when $cwnd > ssthresh$
 - Set $ssthresh = cwnd/2$ after loss
 - Begin with slow-start after timeout

Computer Networks

68

Timeout Misfortunes

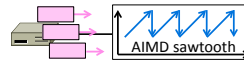
- Why do a slow-start after timeout?
 - Instead of MD cwnd (for AIMD)
- Timeouts are sufficiently long that the ACK clock will have run down
 - Slow-start ramps up the ACK clock
- We need to detect loss before a timeout to get to full AIMD
 - Done in TCP Reno (next time)

Computer Networks

69

TCP Fast Retransmit / Fast Recovery (§6.5.10)

- How TCP implements AIMD, part 2
 - “Fast retransmit” and “fast recovery” are the MD portion of AIMD



Computer Networks

70

Recall

- We want TCP to follow an AIMD control law for a good allocation
- Sender uses a congestion window or cwnd to set its rate ($\approx cwnd/RTT$)
- Sender uses slow-start to ramp up the ACK clock, followed by Additive Increase
- But after a timeout, sender slow-starts again with $cwnd=1$ (as it no ACK clock)

Computer Networks

71

Inferring Loss from ACKs

- TCP uses a cumulative ACK
 - Carries highest in-order seq. number
 - Normally a steady advance
- Duplicate ACKs give us hints about what data hasn't arrived
 - Tell us some new data did arrive, but it was not next segment
 - Thus the next segment may be lost

Computer Networks

72

Fast Retransmit

- Treat three duplicate ACKs as a loss
 - Retransmit next expected segment
 - Some repetition allows for reordering, but still detects loss quickly

Ack 1 2 3 4 5 5 5 5

Computer Networks 73

Fast Retransmit (2)

Computer Networks 74

Fast Retransmit (3)

- It can repair single segment loss quickly, typically before a timeout
- However, we have quiet time at the sender/receiver while waiting for the ACK to jump
- And we still need to MD cwnd ...

Computer Networks 75

Inferring Non-Loss from ACKs

- Duplicate ACKs also give us hints about what data has arrived
 - Each new duplicate ACK means that some new segment has arrived
 - It will be the segments after the loss
 - Thus advancing the sliding window will not increase the number of segments stored in the network

Computer Networks 76

Fast Recovery

- First fast retransmit, and MD cwnd
- Then pretend further duplicate ACKs are the expected ACKs
 - Lets new segments be sent for ACKs
 - Reconcile views when the ACK jumps

Ack 1 2 3 4 5 5 5 5

Computer Networks 77

Fast Recovery (2)

Computer Networks 78

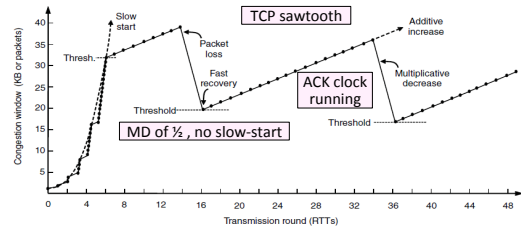
Fast Recovery (3)

- With fast retransmit, it repairs a single segment loss quickly and keeps the ACK clock running
- This allows us to realize AIMD
 - No timeouts or slow-start after loss, just continue with a smaller cwnd
- TCP Reno combines slow-start, fast retransmit and fast recovery
 - Multiplicative Decrease is $\frac{1}{2}$

Computer Networks

79

TCP Reno



Computer Networks

80

TCP Reno, NewReno, and SACK

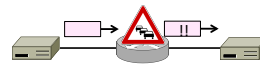
- Reno can repair one loss per RTT
 - Multiple losses cause a timeout
- NewReno further refines ACK heuristics
 - Repairs multiple losses without timeout
- SACK is a better idea
 - Receiver sends ACK ranges so sender can retransmit without guesswork

Computer Networks

81

Explicit Congestion Notification (§5.3.4, §6.5.10)

- How routers can help hosts to avoid congestion
 - Explicit Congestion Notification



Computer Networks

82

Congestion Avoidance vs. Control

- Classic TCP drives the network into congestion and then recovers
 - Needs to see loss to slow down
- Would be better to use the network but avoid congestion altogether!
 - Reduces loss and delay
- But how can we do this?

Computer Networks

83

Feedback Signals

- Delay and router signals can let us avoid congestion

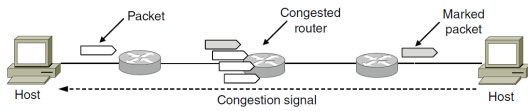
Signal	Example Protocol	Pros / Cons
Packet loss	Classic TCP Cubic TCP (Linux)	Hard to get wrong Hear about congestion late
Packet delay	Compound TCP (Windows)	Hear about congestion early Need to infer congestion
Router indication	TCPs with Explicit Congestion Notification	Hear about congestion early Require router support

Computer Networks

84

ECN (Explicit Congestion Notification)

- Router detects the onset of congestion via its queue
 - When congested, it marks affected packets (IP header)

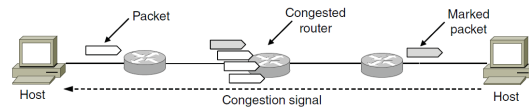


Computer Networks

85

ECN (2)

- Marked packets arrive at receiver; treated as loss
 - TCP receiver reliably informs TCP sender of the congestion



Computer Networks

86

ECN (3)

- Advantages:
 - Routers deliver clear signal to hosts
 - Congestion is detected early, no loss
 - No extra packets need to be sent
- Disadvantages:
 - Routers and hosts must be upgraded

Computer Networks

87