Congestion Control

Context

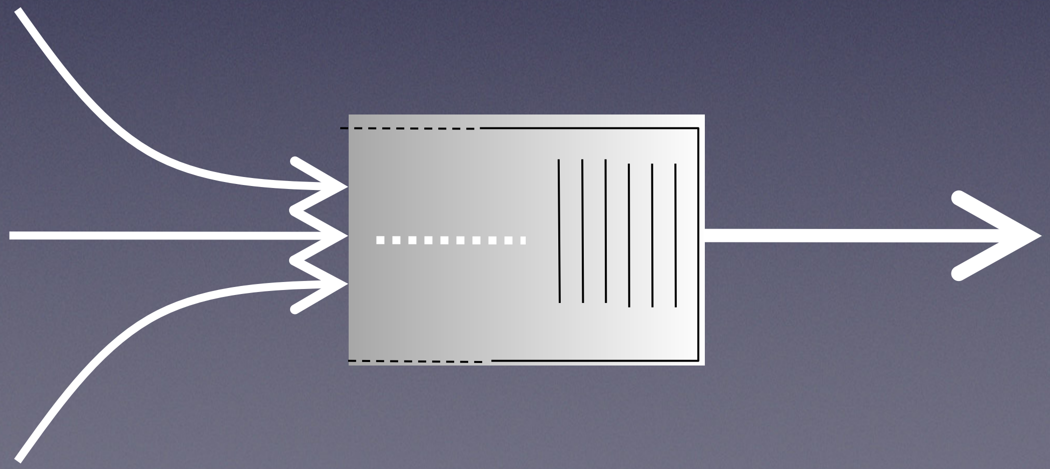
- TCP is the dominant transport protocol in today's Internet
    - Web page loads, BitTorrent transfers, some video streaming, some of Skype
- Embodies some of Internet design principles
    - packet switching (i.e., no state on switches)
    - smart host, dumb network

TCP Mechanisms

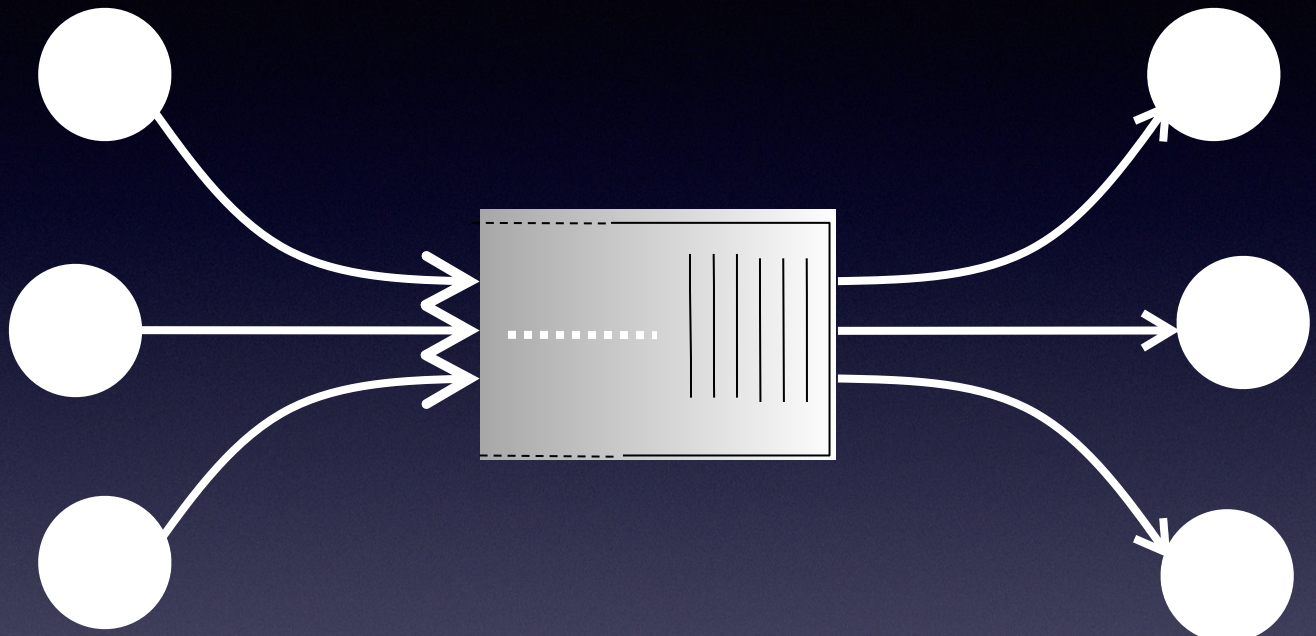
- Flow control: prevent sender from overwhelming the receiver
- Reliable delivery: lost packets are retransmitted
- Congestion control: react to network congestion

Congestion Control

- "Best effort" delivery on the Internet
  - Let everybody send, try to deliver what you can, and drop the rest
  - If many packets arrive in a short period of time, router buffers fill up and packets are lost
  - Loss indicates congestion; so does increased delay



Congestion Control



- What should be the goals of congestion control? (Or what is an ideal congestion control protocol?)

Observations

- Congestion is inevitable, and arguably desirable
- If packets are dropped, then retransmissions can make congestion even worse

Original TCP Design

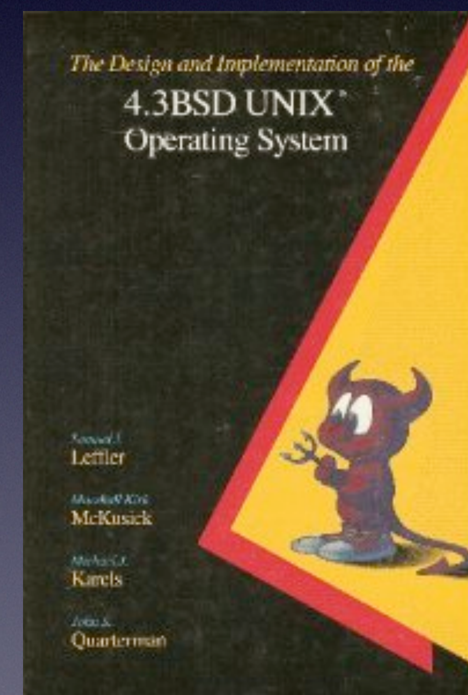
Van Jacobson

- Formerly at LBL
- Internet pioneer
- Inventor tcpdump, traceroute

Michael J. Karels

- Very involved in BSD development
- Replaced Bill Joy as developer

Cited more than 7,000 times.



Context

TCP didn't work well under congestion (circa 1988)

Congestion collapse:

- Breakdowns in performance noted in 1986 on NSFNet.
- 40Kb/s links operating as slow as 32b/s.
- NSFNet was a forerunner of today's Internet backbone (from 1986 to 1995).

RTT Variation Estimate

Q: Why is it important to estimate RTT well?

Q: How can you improve RTT estimation?

Q: Can we do better than cumulative ACKs?

Main contributions

Seven new algorithms:

1. RTT Variance estimation
2. Karn's algorithm (accurate RTT)
3. Slow-start
4. Exponential retransmit timer backoff
5. Dynamic window sizing on congestion
6. Receiver ack policy (delayed ACK or not)
7. Fast retransmit

Packet Conservation

‘Conservation of packets’ principle:

For a connection ‘in equilibrium’, i.e., running stably with a full window of data in transit…

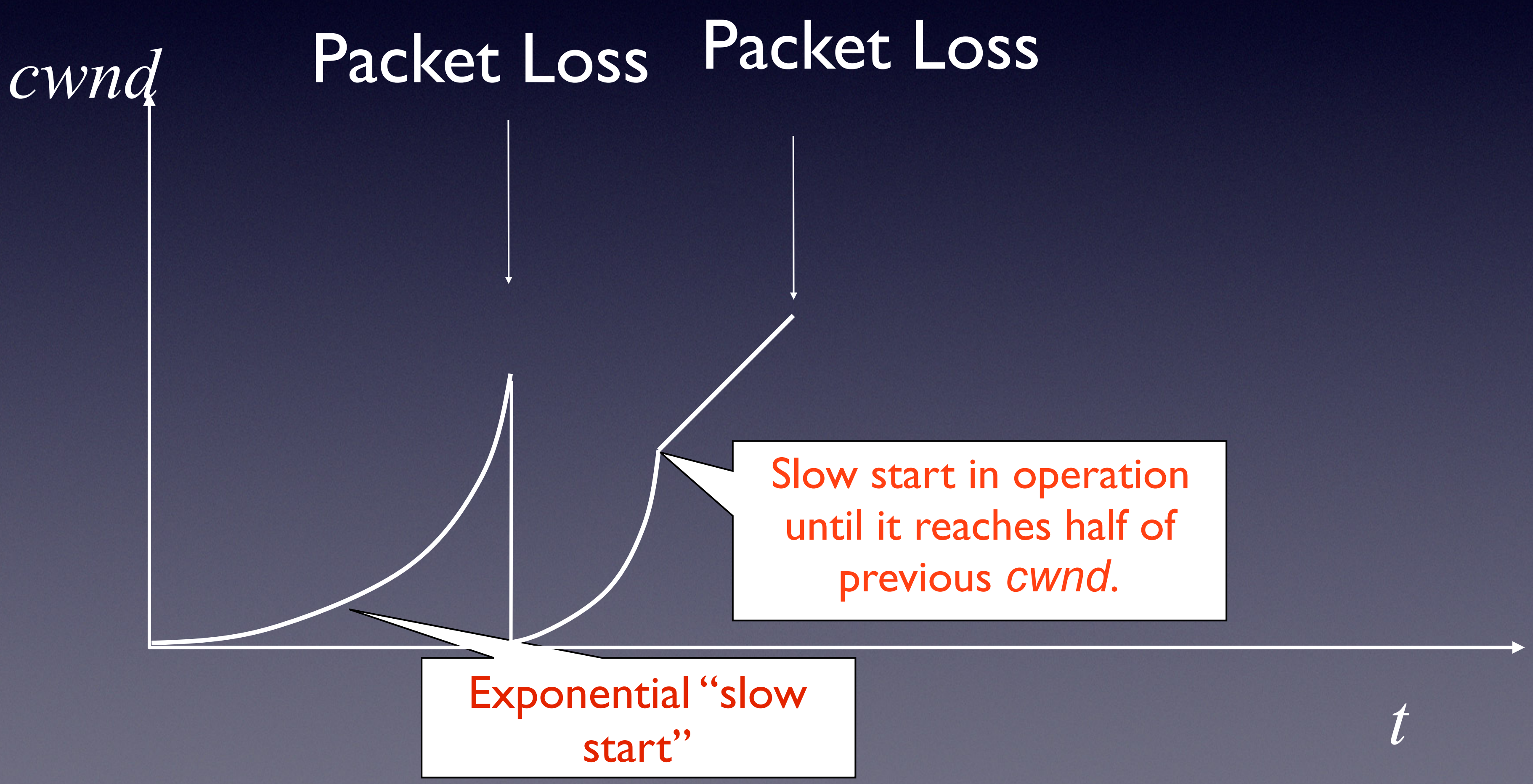
A new packet shouldn’t be put into the network until an old packet leaves.

Additional Packets

- In “slow start”, insert an additional packet for every ACK
- In “congestion avoidance”, insert an additional packet every round trip

Slow-start + AIMD (Tahoe)

Window size = min(advertised window, *cwnd*)



Getting to equilibrium
Slow-start

Q: What is slow-start trying to accomplish?

Q: How long does it take slow-start to reach equilibrium?

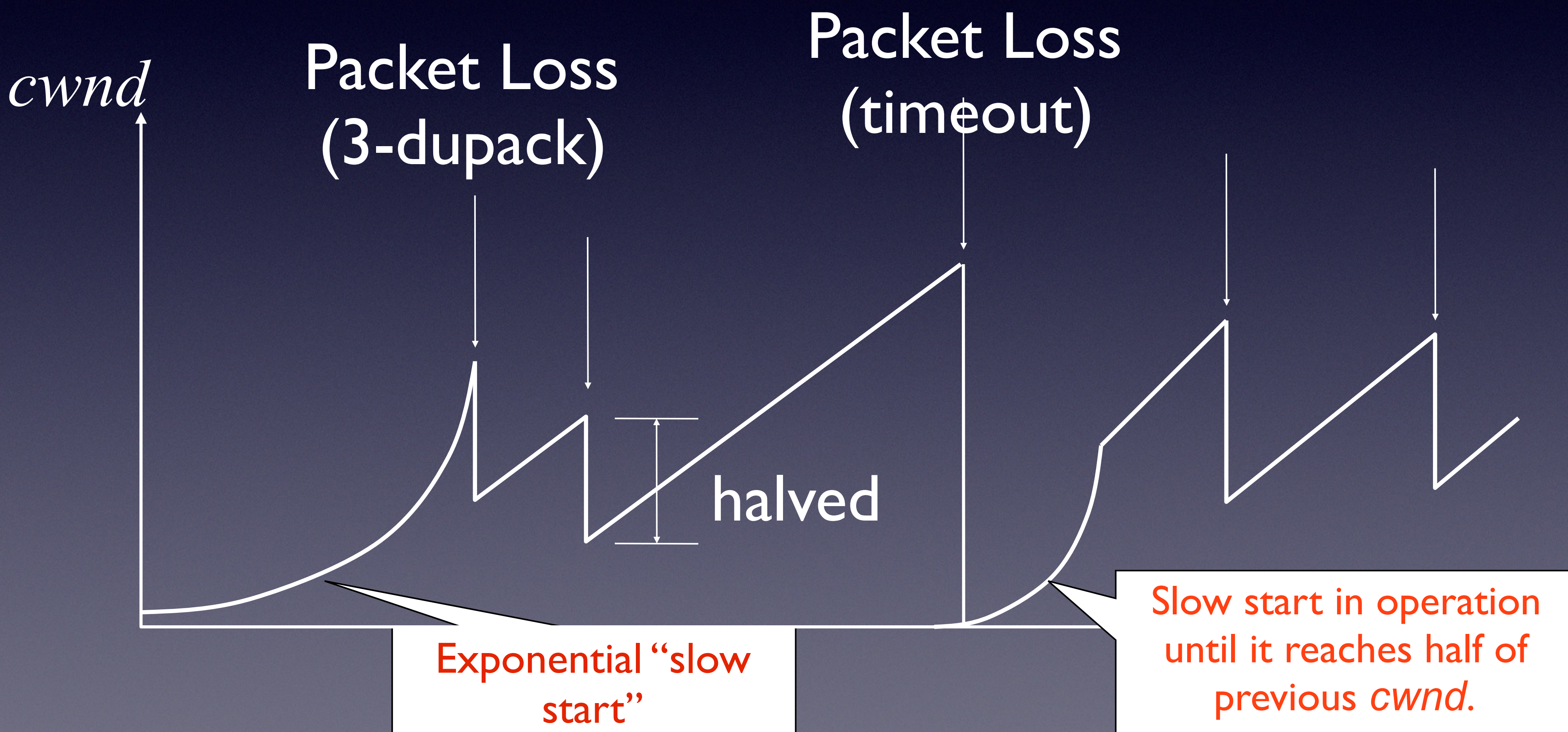
Q: How does AIMD compare to AIAD, MIMD, and MIAD?

TCP Reno

- Enhancements include:
    - early detection of packet loss using 3-dupack (where acks are cumulative ACKs)
    - fast retransmit of such packets
    - fast recovery of congestion window

Slow-start + AIMD (Reno)

Window size = min(advertized window, *cwnd*)



TCP NewReno

- During fast recovery:
  - keep track of last unacked pkt when entering fast recovery
  - on every dupack, inflate congestion window by 1 pkt
    - start sending out new packets while fast retransmit is in flight
  - when last packet is acked, return to congestion avoidance -- set cwnd back to value set when starting fast recovery

TCP Challenges

- TCP early designs were in 80s and 90s
- What are the new challenges for TCP in today’s world?

TCP Congestion Control

- Allocate resources without requiring network support
- “Try and Backoff” strategy:
  - Start with low transfer rate, ramp up rate
  - FIFO routers drop packets when queues fill up
  - Congestion inferred from packet loss
  - Endpoint responds to packet loss by throttling rate

Limits of Try-and-Backoff

- In theory, the link capacity is fully utilized for long flows, but
  - Initial ramp-up takes up most of the response time
  - Channel capacity is left unused
    - If "n" is capacity, takes log(n) steps for the initial ramp-up
    - Wasted capacity during that period: O(n log(n))
  - At the tail of the ramp-up, the rate overshoots the channel capacity
- Could start with higher transfer rates, but could result in higher packet loss/congestion

Network-assisted Congestion Control

1) Routers provide feedback to end-systems
2) Routers explicitly allocate bandwidth to flows

Problem: makes routers complicated and hinders adoption

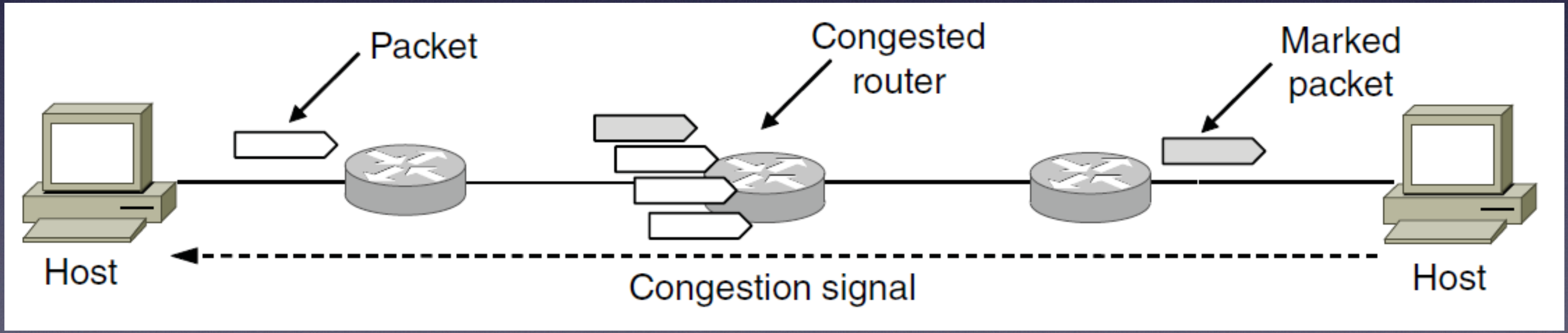
Feedback Signals

- 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

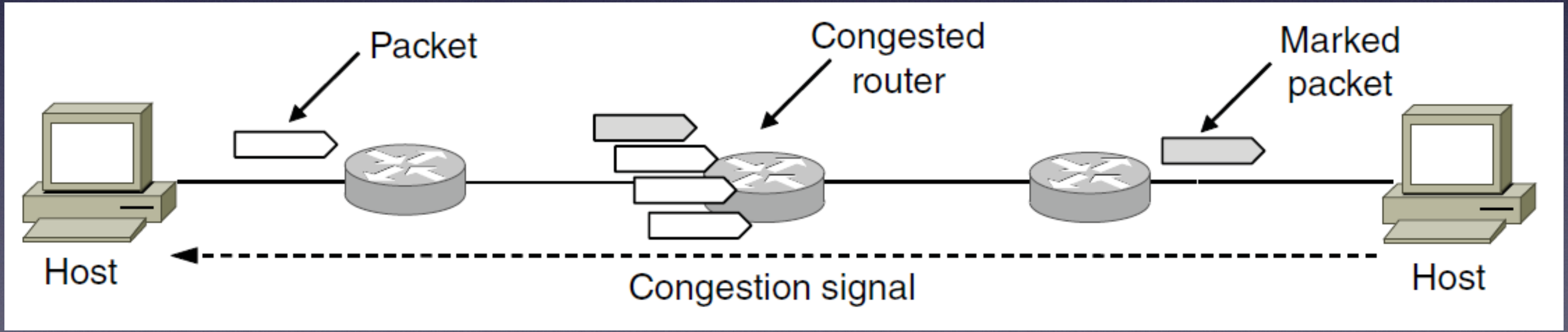
ECN (Explicit Congestion Notification)

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



ECN (2)

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



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