Networked Systems: TCP

Yu-Ju Huang

Dec. 3, 2019

Some slides from CS6410 on 2009 and 2013, and CS144 Stadford University.

Outline

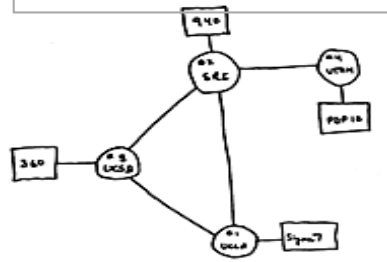
- Network history
- Network basics
  - Layering
  - End-to-end principle
- Congestion Avoidance and Control
- TCP Congestion Control with a Misbehaving Receiver

Parallel beginnings

Brief History of the Internet paper:
"It happened that the work at MIT (1961-1967), at RAND (1962-1965), and at NPL (1964-1967) had all proceeded in parallel without any of the researchers knowing about the other work."

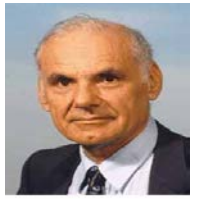
J.C.R. Licklider describes an Intergalactic Network connecting everyone on the globe. (1962)

Four nodes interconnected (UCLA, SRI, UCSB, Utah)



MIT (Leonard Kleinrock) First paper on packet switching theory. (1964)

RAND (Paul Baran) Packet switching for survivable networks.



DARPA (Larry Roberts) plans for "ARPANET".

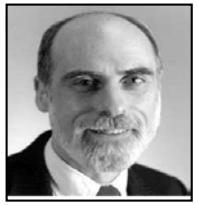
NPL, UK (Donald Davies)
Packet network.

WAN connects two time-sharing computers - btw Mass. and Cal. (circuit switching)

1960 1965 1966 1968 1969

New networks appear: ALOHAnet, Cyclades, IBM SNA.

First public demonstration of this network technology. Also, electronic mail was introduced. (1972)



"Internetting" and TCP born (DARPA), led by Vint Cerf and Bob Kahn. (1974)

TCP/IP deployed (1983)

World Wide Web, by Tim Berners-Lee, became publicly available (1991)

1st Web browser, Mosaic, by Marc Andreessen (1993)

From CS144, Stanford University
with modification, source from Wikipedia

Useful References

1. The Early History of Data Networks
   G. J. Holzmann, B. Pehrson, IEEE Press 1994.

1. The Design Philosophy of the DARPA Internet Protocols.
   D. Clark, ACM Sigcomm 1988

2. Brief History of the Internet
   B. M. Leiner, V. Cerf, D. D. Clark et al.
   http://www.internetsociety.org/internet/internet-51/history-internet/brief-history-internet

Design Philosophy of DARPA Internet Protocols

- Top level goal: effective technique for multiplexed utilization of existing interconnected networks

- The Internet must… (sorted based on importance)
  - continue despite loss of networks or gateways
  - support multiple types of communications service
  - accommodate a variety of networks
  - permit distributed management of its resources
  - be cost effective
  - permit host attachment with a low level of effort
  - resources used in the internet architecture must be accountable

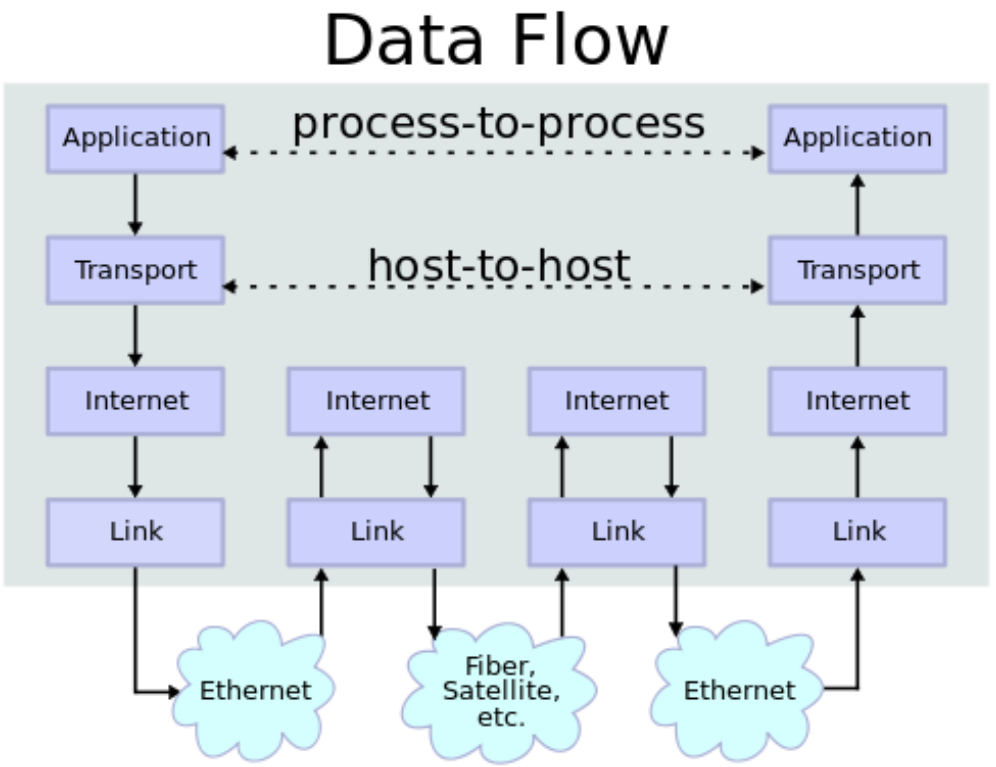
Design Philosophy of DARPA Internet Protocols

- Top level goal: effective technique for multiplexed utilization of existing interconnected networks

- The Internet must… (sorted based on importance)
  - continue despite loss of networks or gateways
    - state information which describes the on-going conversation must be protected
  - support multiple types of communications service
    - TCP, UDP
  - accommodate a variety of networks
    - including military and commercial facilities

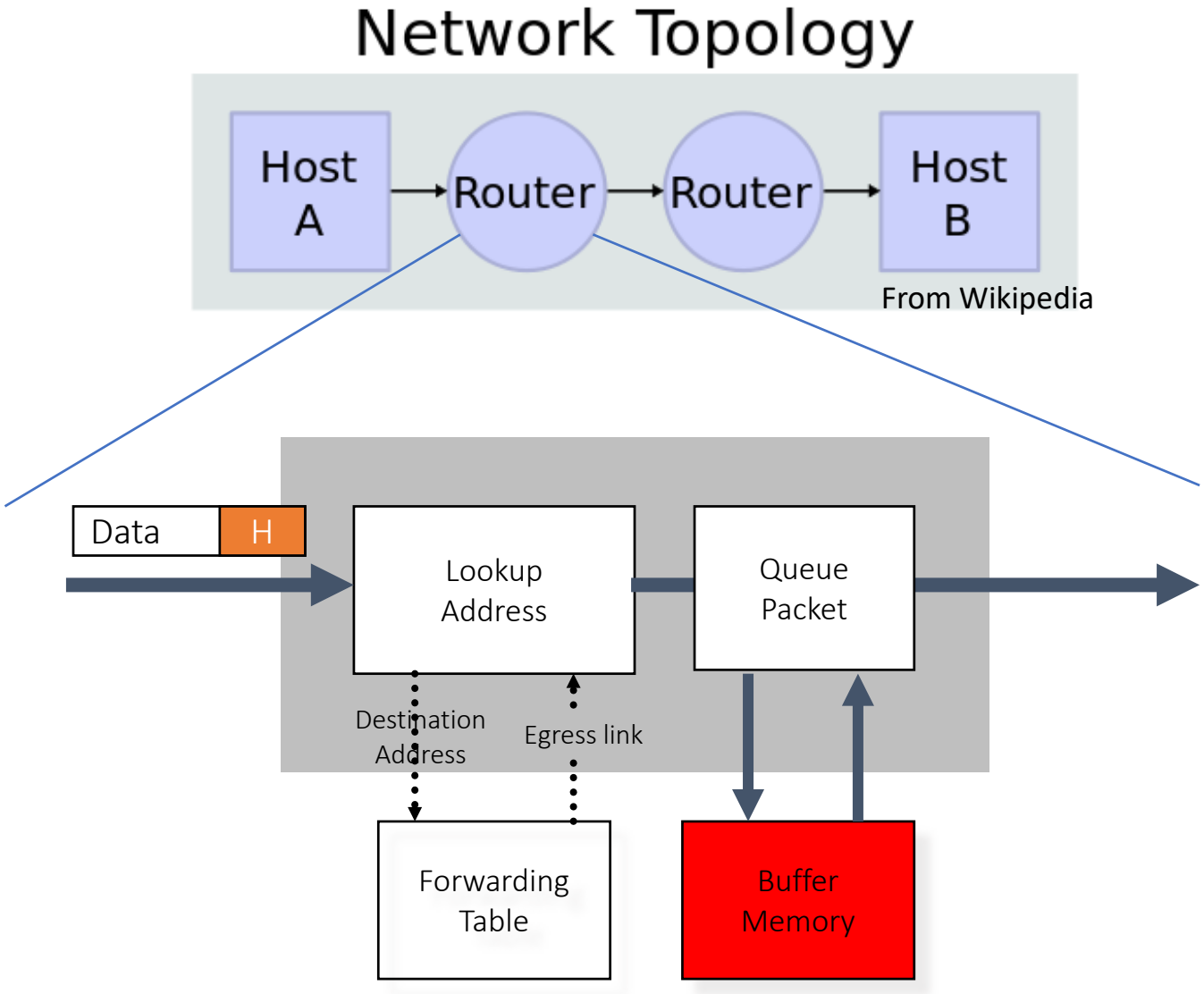
Network Layers

- Layering principle
- End-to-end principle
  - IP layer: best-effort delivery
- TCP
  - Guaranteed in-order delivery



From Wikipedia

Router - Lookup and Forward



From CS144, Stanford University

Congestion Avoidance and Control (SIGCOMM'88)

Van Jacobson

- Adjunct professor at UCLA
- One of the primary contributors to the TCP/IP protocol stack

Problems

- A series of congestion collapses in Oct. 1986
  - Data throughput from LBL to UC Berkeley dropped from 32 Kbps to 40 bps

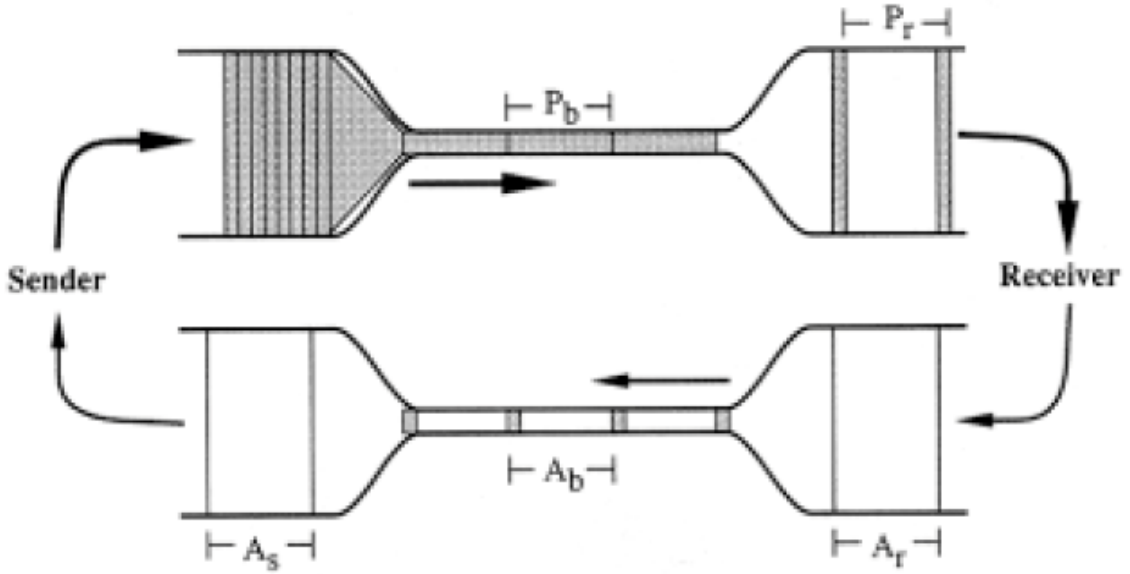
Analysis

- Conservation principle break
  - A new packet isn’t put into the network until an old packet leaves

- Possible failure reasons
  1. The connection doesn’t get to equilibrium
     - Equilibrium: running stably with a full window of data in transit
  2. A sender injects a new packet before an old packet has exited
  3. The equilibrium can’t be reached because of resource limits along the path

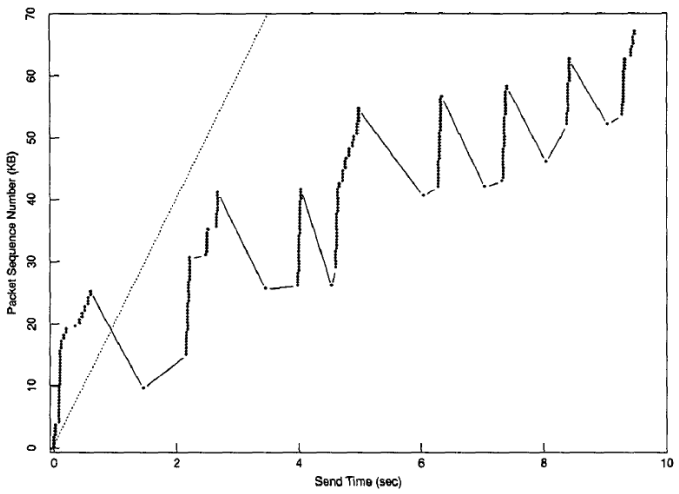
Getting to Equilibrium: Slow-start

- Self-clocking
  - Use ACK as the clock
- So, how to start?

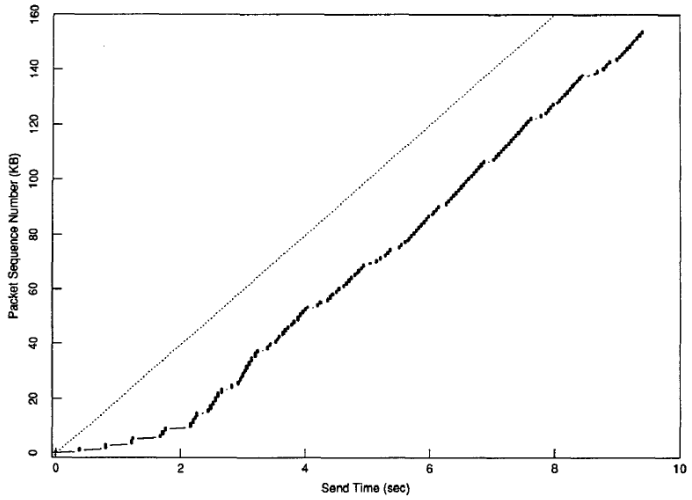


Getting to Equilibrium: Slow-start (2)

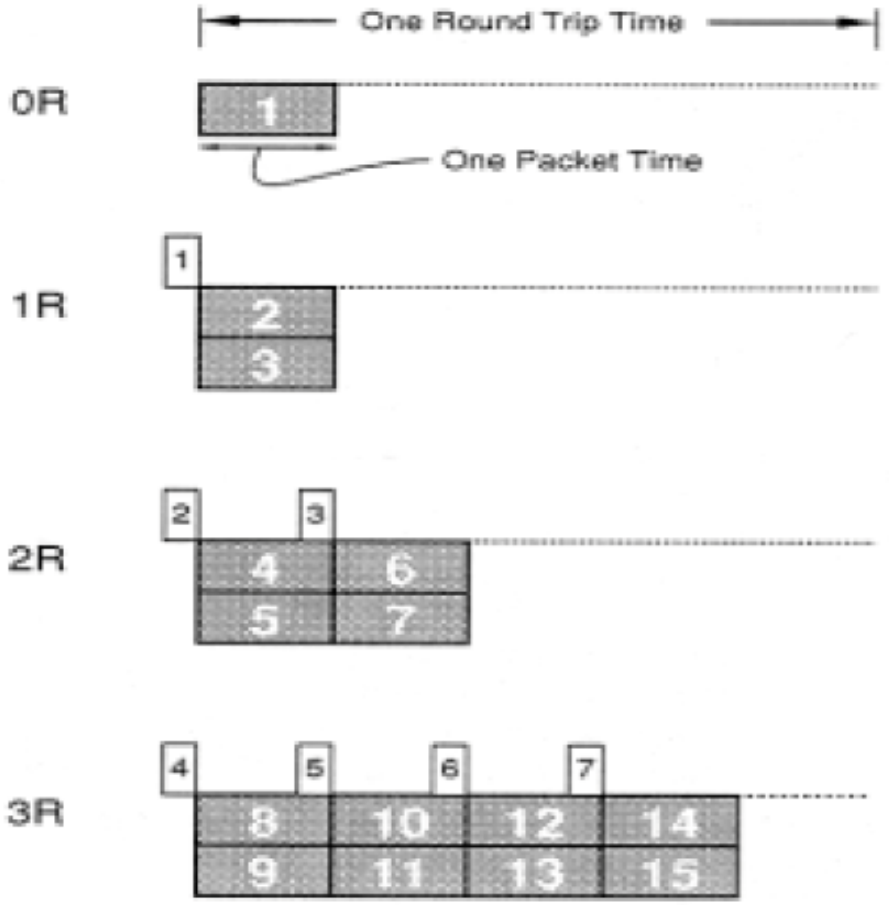
- Slow start
  - Start from cwnd=1
  - Increase cwnd by 1 for each ACK
  - Slow start but grow fast!



Before

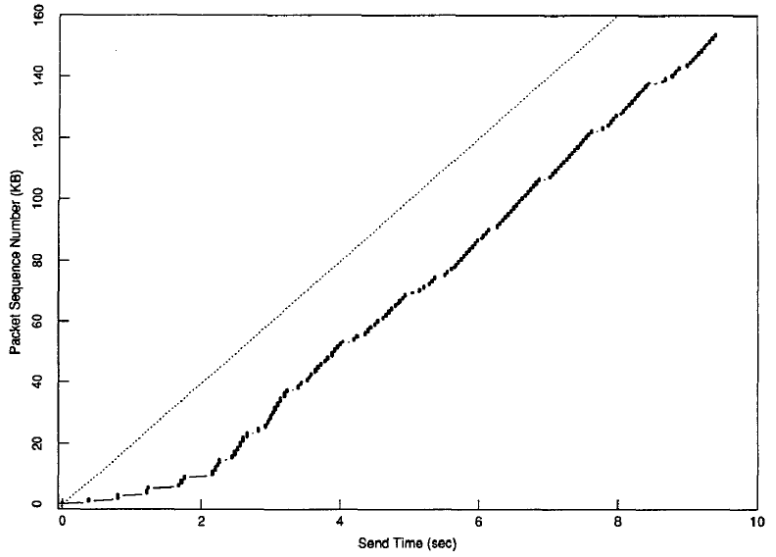


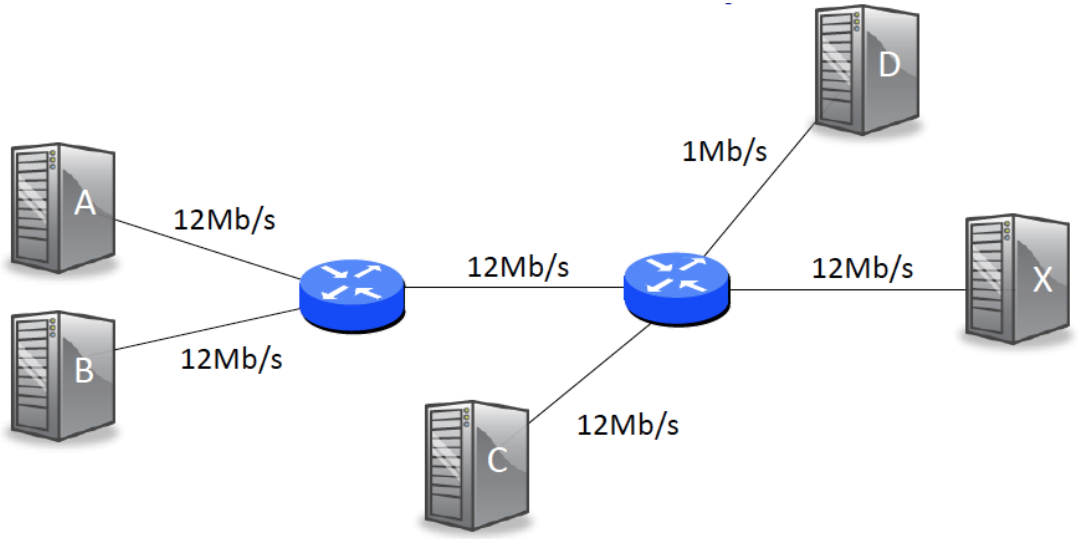
After



After Slow-start

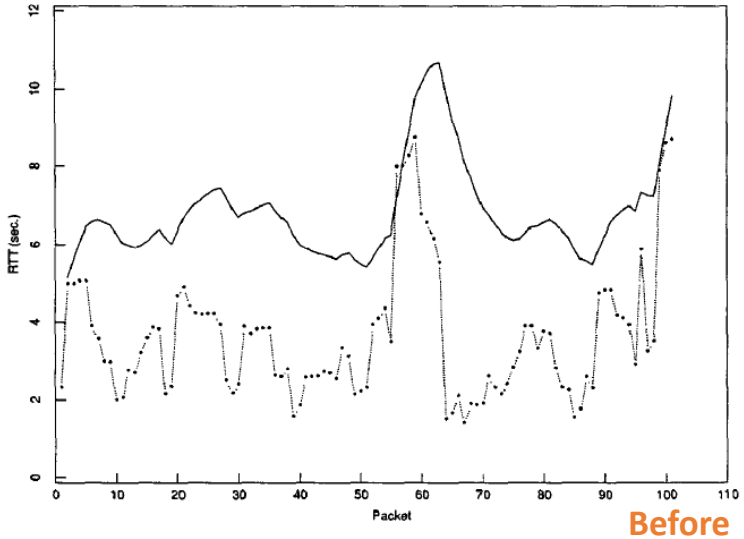
- How to converge to equlibrium?
- Key insight: when congestion happens, packets drop
  - Packet drop reason: insufficient buffer
- Question
  - How to know when packets drop?
  - How to adjust cwnd gracefully?



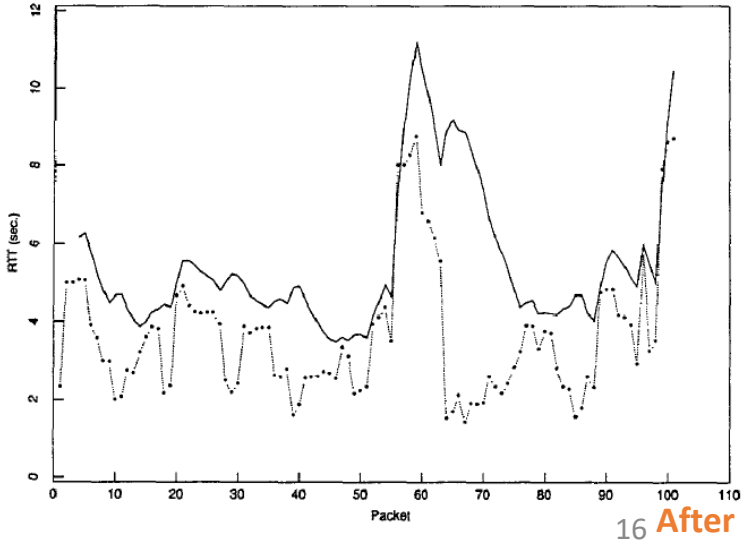


How to know when packets drop?

- Use timeout!
  - Timeout causes retransmission
  - If timeout is not well estimated, a sender will injects a new packet before an old packet has exited
- Timeout value is related to round-trip time (RTT)
  - RTT changes dynamically
  - EstimatedRTT = α * EstimatedRTT + (1−α) * MeasuredRTT
  - Timeout value = β * EstimatedRTT
- Mistake: not considering RTT variation
  - Propose a cheap method for estimating variation



Before



After

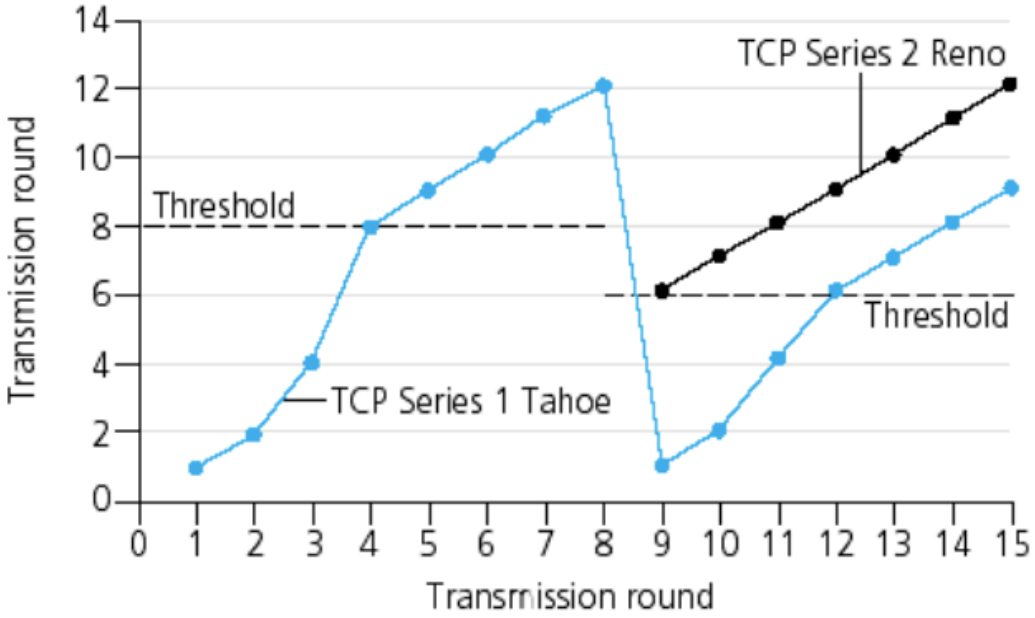
How to Adjust cwnd Gracefully?

- Congestion Avoidance
  - Cannot grow like slow-start, it's too fast
  - Need a way to backoff

- Additive increase / Multiplicative decrease (AIMD)
  - On no congestion
    - cwnd = cwnd + u (u > 0)
  - On congestion
    - cwnd = d * cwnd (d < 1)

Put It All Together

- Start with cwnd = 1
- Slow start: Increase cwnd by 1 for each ack
- On a timeout
  - ssthresh = cwnd / 2
  - cwnd= 1
  - cwnd < ssthresh: cwnd += 1 for each ack (slow start)
  - cwnd > ssthresh: cwnd += 1 / cwnd for each ack (additive increase)



AIMD Analysis

- "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Network", Dah-Ming Chiu and Raj Jain (1989)

- Criteria
  - Quick convergence
  - Efficiency: high utilization
  - Fairness: each end-host gets fair-share

$$\text{Fairness:} \quad F(x) = \frac{(\Sigma x_i)^2}{n(\Sigma x_i^2)}.$$

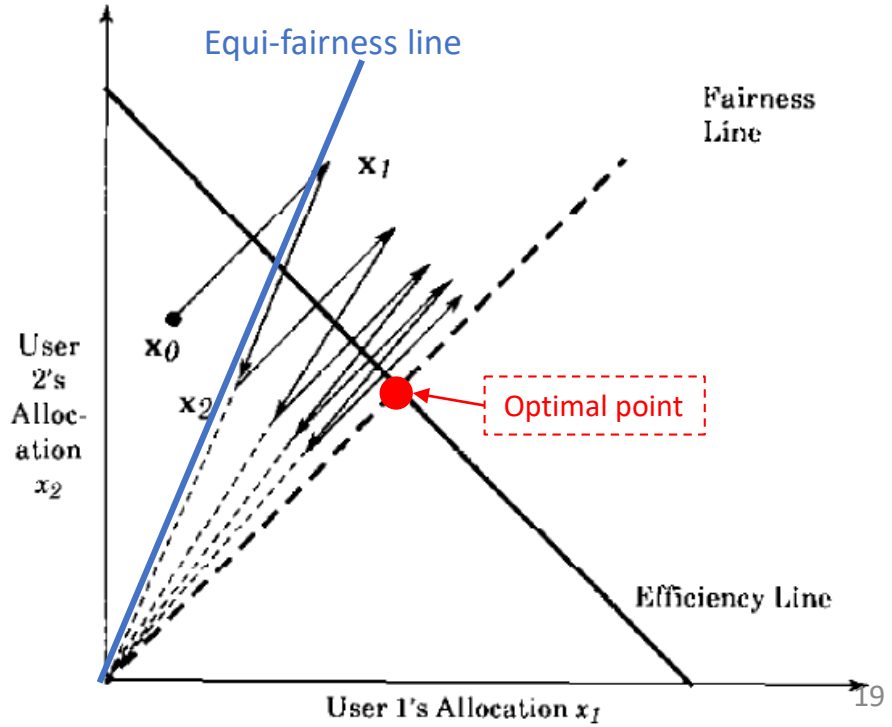


Fig. 5. Additive Increase/Multiplicative Decrease converges to the optimal point.

Other congestion control algorithm

Technique	Target	Feature
TCP Tahoe	General	Slow start, congestion window, fast retransmit
TCP Reno	General	Fast recovery
TCP Hybla	High-latency connection	Analytical evaluation of the congestion window dynamics, remove RTT
TCP BIC	Linux 2.6.8-18	Tries to find the maximum window size to keep for a long period of time, by using a binary search
TCP CUBIC	Linux 2.6.19	The window is a cubic function of time since the last congestion event, independent of RTT
Compound TCP	Microsoft	Use estimate of queuing delay to measure congestion
Fast TCP	High latency	Improved speed of convergence and stability
Data center TCP	General	Allows end-to-end notification of network congestion without dropping packets

https://en.wikipedia.org/wiki/TCP_congestion_control#Algorithms

From CS6410 on 2013.

Other Congestion Control Mechanism

- Timeout or duplicate ACK are actually implicit notification

- Explicit congestion notification (ECN)
  - Rate Control Protocol (RCP)
    - Router divides outgoing link bandwidth equally among all the flows
    - Encode the rate in packet header
  - XCP
    - Router encode hints in packet and let sender know how to adjust cwnd
  - Datacenter TCP (DCTCP)

DCTCP algorithm

Sender side

1. Maintain the fraction of ECN marked seg. for each RTT

$$F = \frac{\#\ \text{of marked segs}}{\#\ \text{of segs}}$$

and update average fraction of marked seg. (α)

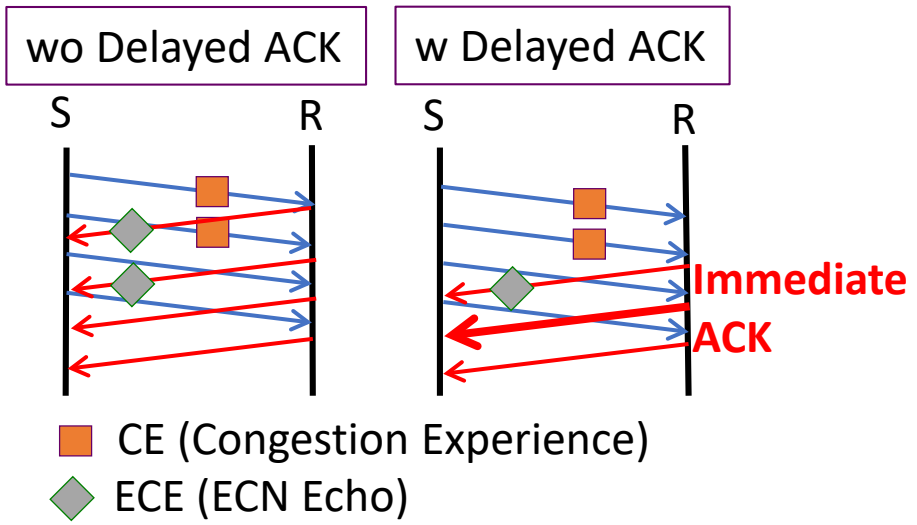
$$\alpha \leftarrow (1 - g)\ \alpha + gF$$

2. Adopt alpha to cwnd decrease

$$\text{cwnd} \leftarrow (1 - \frac{\alpha}{2})\ \text{cwnd}$$

Receiver side

- Mark ECE only when CE packet is received
- send immediate ACK when CE state is changed (regardless of delayed ACK)



Slide from https://slideplayer.com/slide/4764537/

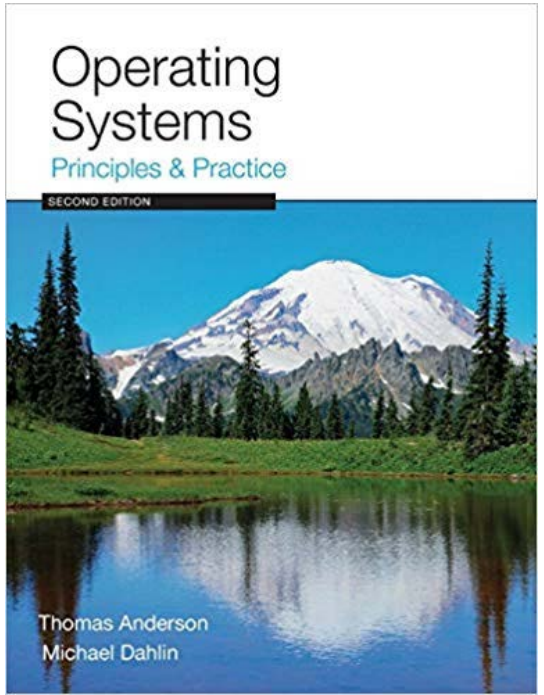
TCP Congestion Control with a Misbehaving Receiver (SIGCOMM'99)

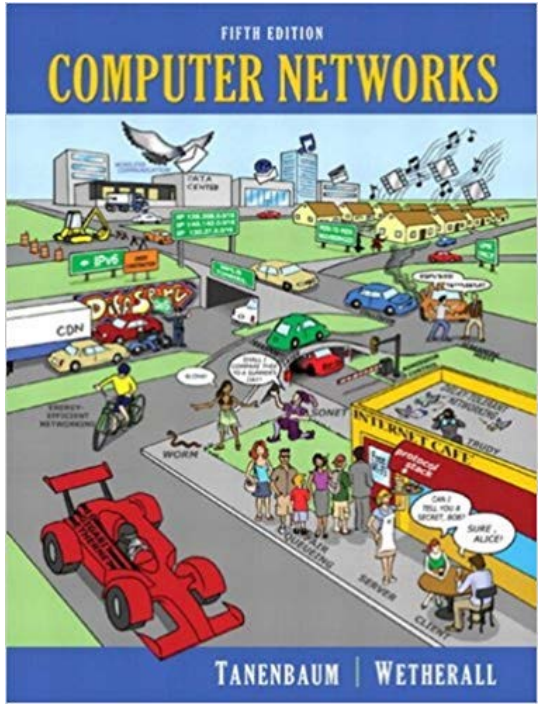
Stefan Savage, PhD at UW, now Professor at UCSD

Neal Cardwell, MS at UW, now at Google

David Wetherall, Professor at UW, now at Google AI

Tom Anderson, Professor at UW





Images from Amazon

Misbehavior on TCP’s congestion control

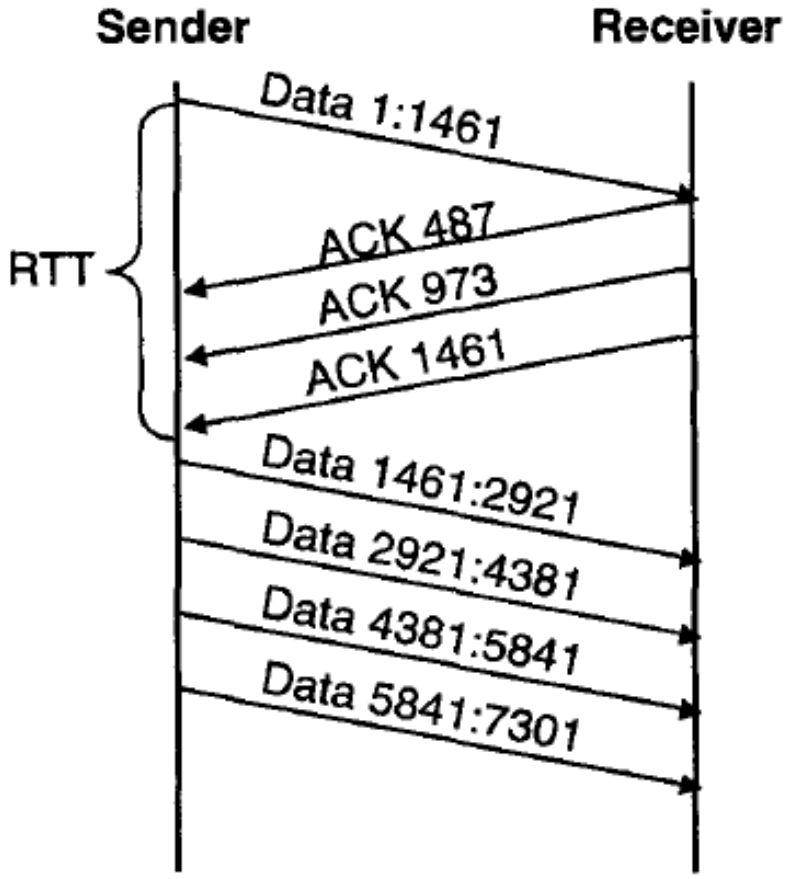
- TCP mechanisms implicitly rely on both endpoints to cooperate in determining the proper rate at which to send data

- TCP's vulnerabilities arise from
  - Unstated assumptions
  - Casual specification
  - Congestion control that are backward compatible with previous TCP

- Proposal: designing robust protocols
  - Principle 1: Every message should say what it means
  - Principle 2: The conditions for a message to be acted upon should be clearly set out
  - Principle 3. If the identity of a principal is essential to the meaning of a message, it is prudent to mention the principal's name explicitly in the message.

ACK division

- TCP spec
  - During slow start, TCP increments cwnd by at most SMSS bytes for each ACK received.
  - During congestion avoidance, cwnd is incremented by 1 full-sized segment per round-trip time (RTT).
- Attack
  - Upon receiving a data segment containing N bytes, the receiver divides the resulting ACK into M separate acknowledgments



Misbehavior: cwnd=4 instead of 2!

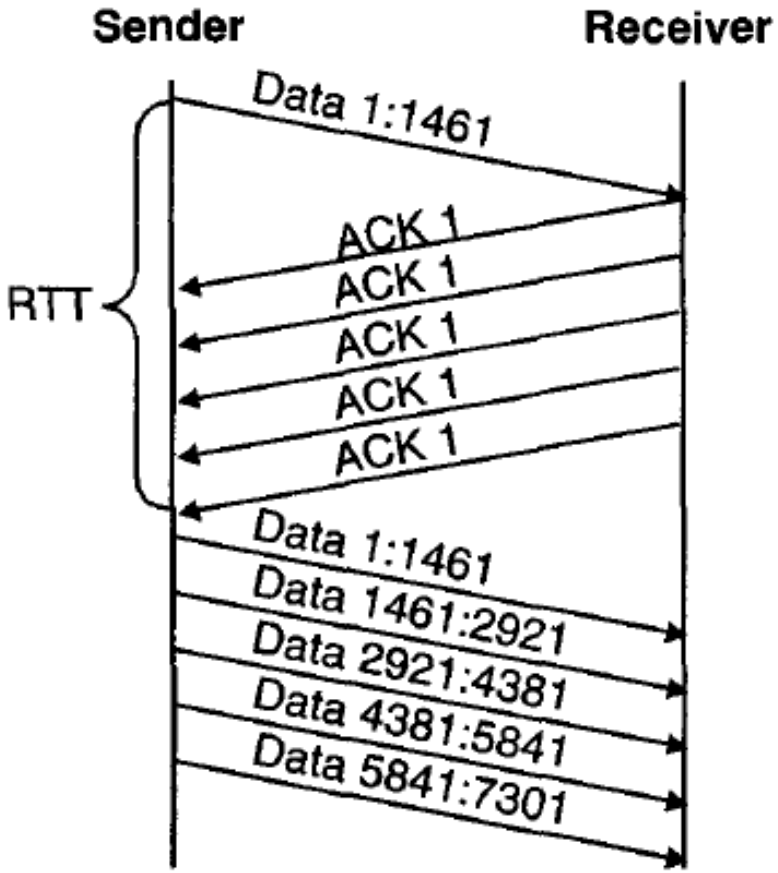
ACK division - Solution

- This vulnerability arises from an ambiguity about how ACKs should be interpreted

- Two solutions
  - modify the congestion control mechanisms to operate at byte granularity
    - virtually identical to the "byte counting" modifications to TCP discussed in [Al198, A1199]
  - guarantee that segment-level granularity is always respected
    - only increment cwnd by one SMSS when a valid ACK arrives that covers the entire data segment sent
    - In Linux 2.2.x

DupACK spoofing

- TCP fast recovery
  - Set cwnd to ssthresh plus 3*SMSS
  - For each additional duplicate ACK received, increment cwnd by SMSS
- Attack
  - Upon receiving a data segment, the receiver sends a long stream of acknowledgments for the last sequence number received



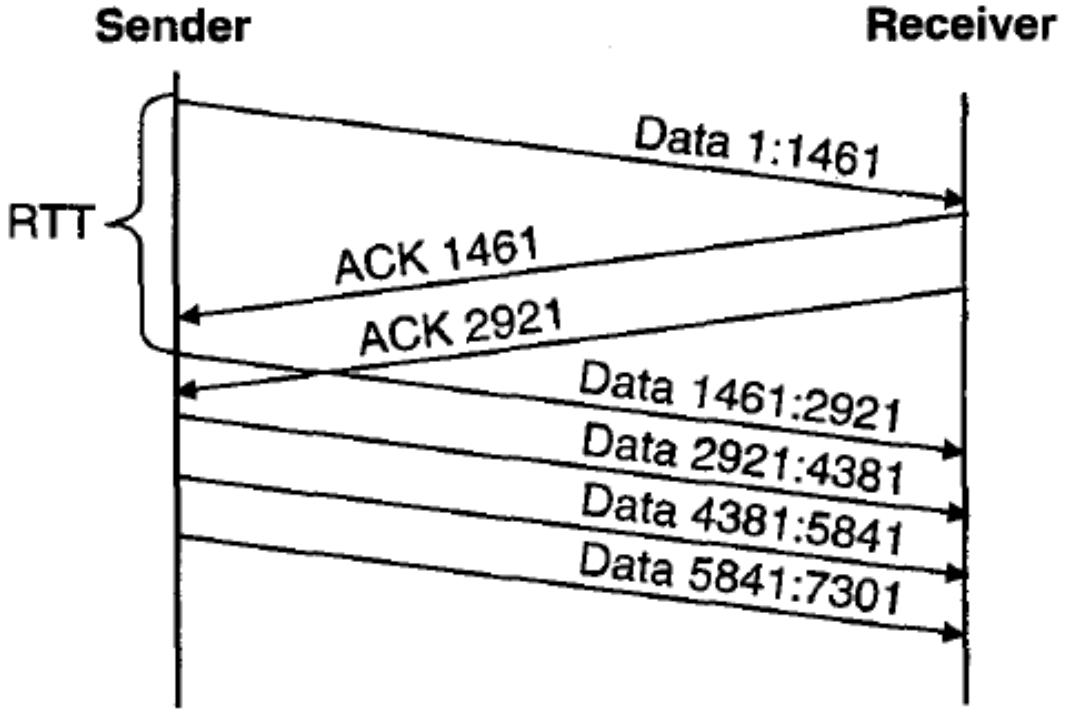
DupACK spoofing - Solution

- This vulnerability arises from the meaning of a duplicate ACK is implicit, dependent on previous context, and consequently difficult to verify.

- Solution
  - Two new fields into the TCP packet format: Nonce and Nonce reply
  - Sender: fills the Nonce field with a unique random number
  - Receiver: echoes the nonce value by writing it into the Nonce Reply

Optimistic ACKing

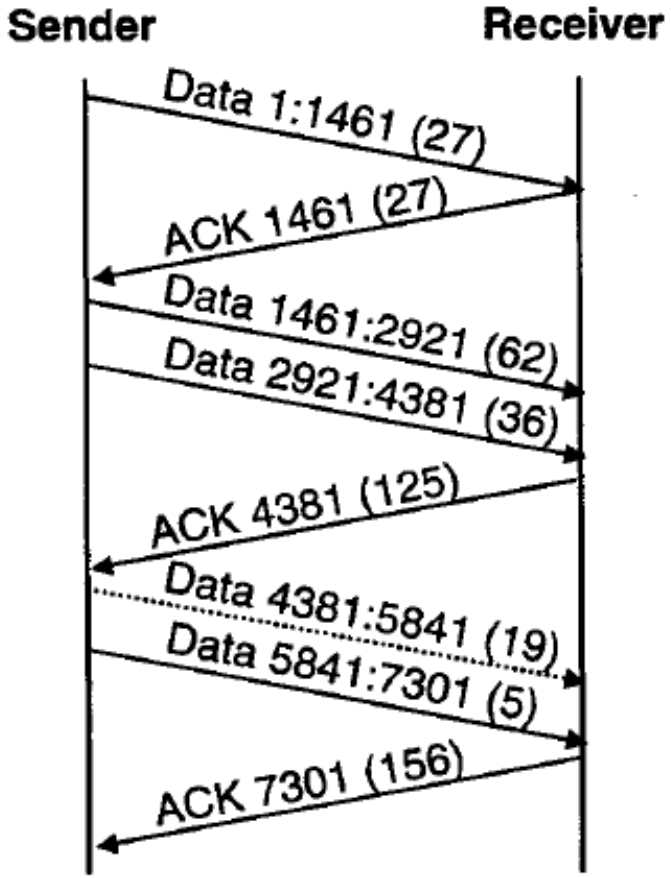
- TCP spec
  - assumption that the time between a data segment being sent and an ACK is at least one round-trip time. Since TCP's congestion
  - However, there is no mechanism to enforce this assumption
- Attack
  - Upon receiving a data segment, the receiver sends a stream of ACKs anticipating data that will be sent by the sender



Optimistic ACKing - Solution

- The optimistic ACK attack is possible because ACKs do not contain any proof regarding the identity of the data segment(s) that caused them to be sent

- Solution
  - Cumulative Nonce



Last ACK's cumulative nonce value is incorrect (156 instead of the expected value of 149)

Different Implementation

	ACK Division	DupACK Spoofing	Optimistic Acks
Solaris 2.6	Y	Y	Y
Linux 2.0	Y	Y (N)	Y
Linux 2.2	N	Y	Y
Windows NT4/95	Y	N	Y
FreeBSD 3.0	Y	Y	Y
DIGITAL Unix 4.0	Y	Y	Y
IRIX 6.x	Y	Y	Y
HP-UX 10.20	Y	Y	Y
AIX 4.2	Y	Y	Y

Recap

- Congestion control
  - Slow-start
  - RTT estimation using variation
  - AIMD
- TCP attack
  - ACK division
  - DupACK spoofing
  - Optimistic ACKing

Perspective

- There are **4.1 billion Internet users** in the world as of December 2018
  - compared to 3.9 billion Internet users in mid-2018 and about 3.7 billion Internet users in late 2017
- Mobile traffic is responsible for 52.2 percent of Internet traffic in 2018
  - compared to 50.3 percent from 2017

Numbers from https://hostingfacts.com/internet-facts-stats/

Thanks!