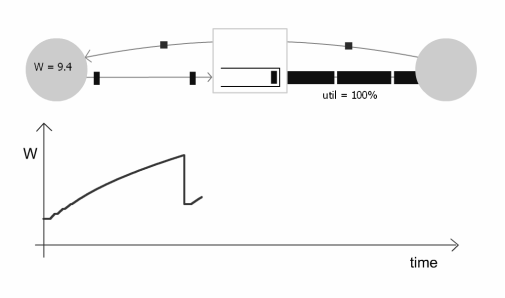
Internet Programming & Protocols Lecture 27

Review

Info on take-home final

TCP additive increase multiplicative decrease (AIMD)

cwnd ← cwnd + 1/cwnd for each ACK
Loss → cwnd ← cwnd/2



Plan of attack

- Network overview
- BSD sockets and UDP
- TCP
  - Socket programming
  - Reliable streams
  - Header and states
  - Flow control and bandwidth-delay
  - Measuring performance
  - Historical evolution (Tahoe …SACK)
  - Congestion control
- Network simulation (ns)
- TCP accelerants
- TCP implementations
- TCP over wireless, satellite, …

The lectures

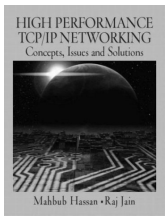
LECTURES
1 overview, class mechanics, networks 101
2 Ethernet, IP, ARP
3 IP routing, tcpdump/ethereal ICMP ping/traceroute
4 UDP, BSD sockets, client/servers
5 UDP, DNS
6 TCP socket programming
7 reliable streams, TCP header
8 TCP states, flow control, bandwidth-delay
9 performance tools
10 nagle, delayed ACKs, timers, RTT estimation, TCP slow-start
11 TCP congestion control, TCP Tahoe
12 TCP Reno, NewReno, SACK, FACK
13 other network programming paradigms, review

LECTURES	
14	Models and measurement
15	emulation and simulation
16	ns
17	S-TCP, HSTCP BI-TCP
18	Bandwidth estimation, auto-tuning
19	Vegas, fast, westwood
20	AQM, RED, ECN. XCP
21	Parallel streams, rate based, UDP
22	slow links, asymmetric channels
23	satellites
24	Wireless, mobile, ad hoc
25	Kernel implementation
26	instrumentation, zero copy
27	review

The text

1. Intro and history
2. TCP/IP fundamentals
3. Measuring performance (tools)
4. Network simulation
5. TCP modeling
6. Wireless nets
7. Mobile nets
8. Optical nets
9. Satellite nets
10. Asymmetric nets
11. TCP flavors and ns
12. AQM
13. TCP implementation

Appendices: M/M/1 Queues, FreeBSD, Auto-tuning



The readings

- **Hobbes' Internet history**
- **CACM '86 Ethernet: Distributed Packet Switching for Local Computer Networks**
- **RFC 791 IP**
- **RFC 768 UDP**
- **Design philosophy of the DARPA Internet Protocols '88**
- **RFC 793 TCP**
- **RFC 1323 window scale, timestamps, PAWS**
- **Jacobson Congestion Avoidance and Control 1988**
- **Floyd Simulation-based comparisons of Tahoe, Reno, NewReno, and SACK TCP**
- **RFC 2581 TCP congestion control**
- **On the Effective Evaluation of TCP**
- **bi-tcp**
- **TCP Vegas: New techniques for congestion detection and avoidance**
- **effects of parallel TCP sockets**
- **Faster TCP**
- **delay-tolerant networking**
- **An Analysis of TCP Processing Overhead '89**
- **The Transmission Control Protocol**

The assignments

1. Hello, Internet
2. Tcpdump/ethereal
3. UDP
4. Reliable UDP
5. TCP client/server
6. Brain food
7. Hello ns
8. ns chapter 11
9. ns RTT and RED
10. NewReno with UDP

The layers

- Physical/data link
  - Ethernet, checksums, encapsulation, CSMA/CD
  - Wireless, bit error rates, TOE
  - Transmission and propagation delay (satellite)
- Network layer
  - IP, datagrams, routing, RTT, addressing, ICMP, TTL, fragmentation
  - AQM, ECN
- Transport layer
  - UDP
  - TCP
  - Flow control, congestion and loss
- Application layer
  - BSD sockets
  - Ports and services
  - Network tools

Concept Collection

- **ACK/NAK cumulative ACK**
- **ACK clocking**
- **AIMD**
- **AQM/RED**
- **Auto-tuning**
- **Bandwidth-delay product**
- **Best effort**
- **Bit error rate**
- **Byte counting (ACK)**
- **Checksums**
- **Client/server/concurrent/iterative**
- **Compressed ACK**
- **Header compression**
- **Congestion control/avoid**
- **Conservation of packets**
- **CIDR**
- **CSMA/CD**
- **cwnd/sstrhesh**
- **Datagram vs reliable stream**
- **Delay-based congestion mgt**
- **Discrete event simulation**
- **Dup threshold**
- **ECN**
- **Exponential backoff**
- **Flow control**
- **Forward ACK**
- **Fragmentation**
- **Inverse sqrt p**
- **Layers/encapsulation**
- **Maximum segment lifetime(MSL)**
- **MTU MSS/MTU discovery**
- **Network mask**
- **Packet switching vs circuit-based**
- **Partial ACK**
- **promiscuous**
- **Routing**
- **RTT and RTT estimation**
- **Selective ACK (SACK)**
- **Self-clocking**
- **Sliding window**
- **Slow-start**
- **Snoop/split/proxy TCP**
- **Stretch ACK**
- **Subnets/supernets**
- **Switch vs hub**
- **TTL**

Our tool set

- ping/traceroute
- ifconfig/netstat
- strace
- lsof
- dig
- ethereal tcpdump/tcptrace/xplot
- ttcp/iperf/netperf
- ns

Programming TCP

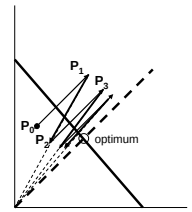
- Reliable stream of bytes (readn())
- socket(), bind(), connect(), listen(), accept(), read(), write()
- Error returns
  - Can get various timeouts (connect failure, retransmit)
  - Connection reset
- Socket options
  - SO_KEEPALIVE – kernel sends probes on idle socket, early notification of broken connection
  - SO_REUSEADDR – TCP close() can linger awhile, this allows you to restart your server with same port
  - TCP_NODELAY disable Nagle
  - TCP_DEBUG (BSD) and TCP_INFO (linux)
  - SO_SNDBUF SO_RCVBUF
    - Send and receive buffer sizes
    - Size to bandwidth-delay product

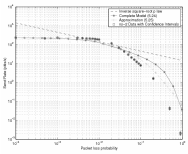
Things that go bump in the net

- TCP connect, and no server process
- TCP connect, server host down
- active TCP session, ctrl-c server
- inactive TCP session, ctrl-c server
- active TCP session, server computer crashes
- inactive TCP session, server computer crashes
- Inactive TCP session, several routers on the path crash and reboot
- inactive TCP session with KEEPALIVE, server computer crashes
- inactive TCP session, server computer crashes and reboots
- start 2nd copy of server
- server tries to bind to port < 1024
- A sends faster than B can receive

Modeling TCP

- Congestion control (AIMD)
- Slow-start
  - Double cwnd each RTT cwnd ← cwnd+1
  - To reach window size of N segments, takes $\log_2(N)$ RTT's
- Congestion avoidance
  - Increase cwnd by one each RTT
  - Each ACK cwnd ← cwnd + 1/cwnd
- Inverse sqrt(p) $\frac{MSS\sqrt{3/2}}{RTT\sqrt{p}}$
- Sensitive to MSS and RTT (speed of light)
- Scalable, fair, friendly, stable





Theory, experiment, simulation

- **Live internet tests**
  - See results in ultimate environment
  - Real TCP stacks/OS, traffic
  - Vary time and host/paths
  - Worry about impact?
- **Test beds**
  - Controlled traffic, but real OS
  - Usually LAN based, no queuing
  - Repeatable
  - Not very good for cross-traffic
- **Emulators**
  - Same as testbed
  - Plus control delay, loss, data rates, dup's, out-of-order
  - Easy to reconfigure
- **Need tools to probe and measure**
- **Simulations**
  - Easily reconfigured
    - Complex topology
    - Vary TCP flavor
  - Repeatable
  - Detailed feedback/instrumentation
  - Add delay, loss, cross-traffic, queues
  - Randomness for confidence
  - Investigate "new" networks/protocols
  - cheap
  - Can be slow
  - Not real TCP

Experimental/simulation measurements

Things to consider for both test beds and simulations

- Learn about good experimental design
  - Adequate tests and confidence intervals
  - Random start times, re-order experiments
  - Anecdotal (illustrate a point) vs prove a point
  - Steady-state, test duration
- Selecting and configuring your flavor of TCP
  - Tahoe, Reno, Newreno, SACK, FACK …
  - Window sizes, RTT, timer tick resolution, delayed ACK, Nagle
  - Knowing what your OS is doing: timestamps, window-scaling, Linux
  - Router queue sizes and management (droptail, RED, WFQ, ECN)
- Selecting competing traffic
  - Bottleneck links
  - Realistic traffic? (bursty, Pareto)
  - Traffic on the reverse path

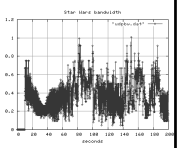
Things that slow us down …

- **Physical layer**
  - Loose connectors
  - RF interference
  - Collisions
  - Slow media or media errors (BER)
  - Speed of light
  - Backhoe
- **Link layer**
  - Half/full duplex mismatch
  - CRC errors
  - ARQ (retry)
  - Exponential backoff
  - Packet reordering
  - NIC queues (txquelen)
  - Device (NIC) Driver software
    - interrupts
- **Network layer**
  - Fragmentation
  - Long routes
  - Slow links
  - Congestion
  - queue overflows (drops) AQM
  - Synchronous routing updates?
  - Packet reordering (route/Juniper)
  - Software implementations/bugs
  - Firewalls/encryption
    - Block ports, ICMP
    - Examine/modify packets

Encapsulation overhead: just handling all the layering extra bits in headers

Things that slow us down … UDP

- Transport layer (UDP)
  - Some UDP applications (streaming) do not backoff under heavy network load, hurting the other transport protocol (TCP) – not "TCP-friendly"
    - RealPlayer audio: 10 pkts/sec (rate-based) 70 kbs
      - 100 users, 7 mbs → 70% of 10mbs ethernet
    - Star Wars mpeg streaming video 400 kbs
  - DNS lookups can slow a network application
  - Hackers use UDP to flood the network (denial of service)
- Sending a packet to a remote host
  1. ARP for local DNS server (IP address in /etc/resolv.conf)
  2. Send DNS query to local DNS (this could take a while)
  3. ARP for subnet router
  4. Send one or more packets to remote via subnet router and then out into the Internet …



Things that slow us down … TCP

- SNDBUF limits
- RCVBUF limits
- NIC speed or bottleneck link speed
- Slow-start, delayed ACK, Nagle
- Packet loss and congestion
  - TCP recovery variants (Tahoe to Westwood)
  - Queue management
- Packet reordering
- Slow ACK path (asymmetric net)
- TCP implementation
- Application "protocol"

- Recovery rate sensitive to RTT (speed of light) and MSS

Accelerating TCP

- Tuning configuration parameters
  - SNDBUF/RCVBUF – bandwidth-delay product
  - Txquelen
  - RFC1323 (window scaling, timestamps)
  - Nagle, delayed ACK
  - Initial slow-start
- Speeding recovery after packet loss
  - Fast retransmit, fast recovery
  - SACK/FACK
  - AIMD, STCP, HSTCP, BI-TCP, TCPW
- Avoiding packet loss
  - Dup threshold (out of order resiliance)
  - Slow-start and congestion avoidance (reduces losses)
  - Vegas/FAST

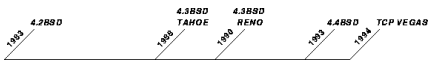
Parallel TCP

TCP evolution

- Window blast, go-back-N, no receiver out of order buffering
- Tahoe (slow start, expo. timeout, RTT estimation, fast retransmit, AIMD)
- Reno (fast recovery)
- New Reno (partial ACKs)
- SACK/FACK (fill holes in one RTT)
- Delay-based: Vegas, FAST
- STCP
- HSTCP
- TCPW
- BI-TCP

TCP Tahoe summary

- Van Jacobson's tweaks to TCP in 4.3 BSD ('88)
- Exponential backoff on timeouts
- Improved RTT estimator
- Slow-start (startup, packet loss, idle)
- Congestion management (AIMD) cwnd/ssthresh
  - Sender can't send more than min(cwnd, his SNDBUF, receiver's adv. window)
    - Can't have more than cwnd un-ACK'd packets (e.g., packets in flight)
  - If packet loss, cut sending rate in half, then slowly increase
- Fast retransmit (3 dup ACKs), avoid timeouts



Tahoe Reno NewReno

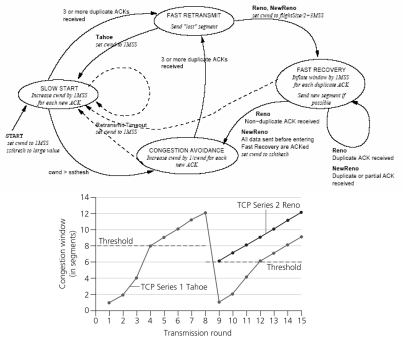
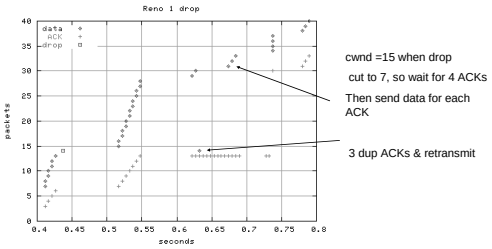


Figure 3.51 ♦ Evolution of TCP's congestion window (Tahoe and Reno)

Reno recovery

- The graph illustrates Reno sending new packets for dup ACKs after half the dup ACKs were accounted for. This helps performance some, the real advantage of Reno over Tahoe is starting cwnd at cwnd/2 and not cwnd=1.



cwnd =15 when drop

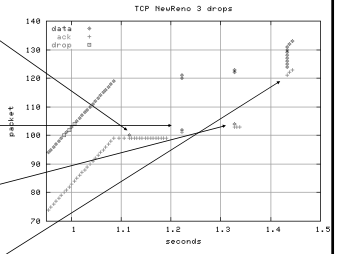
cut to 7, so wait for 4 ACKs

Then send data for each ACK

3 dup ACKs & retransmit

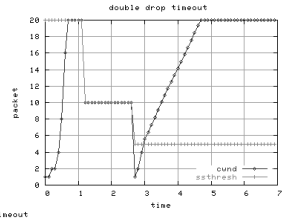
NewReno fix

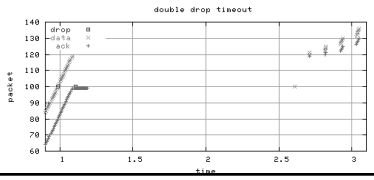
- Same loss scenario {100,102,104} lost
- 3 dup ACK, retransmit 100
- Cut cwnd in half (20 → 10)
- After 1 RTT, ACK for 100-101, partial ACK, retransmit 102 and 120,121
- One more RTT, ACK for 102-103, retransmit 104 and 122,123
- One more RTT, cumulative ACK for rest of window, exit recovery, cwnd = 10 (note packet burst from cumulative ACK)



TCP timeouts

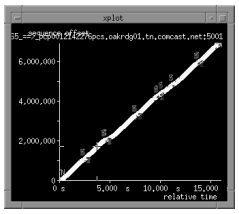
- NewReno with double drop (timeout)
- 3-dup retransmit (and lost)
  - ssthresh ← cwnd/2
  - cwnd ← cwnd/2
- Timeout
  - ssthresh ← cwnd/2
  - cwnd ← 1

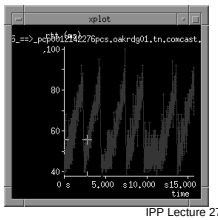




Delay-based congestion avoidance

- Standard TCP detects congestion by packet loss
  - Then we must go thru all sorts of gyrations to speed recovery
    - Fast retransmit, fast recovery, SACK, FACK, HSTCP, BI-TCP
- TCP Vegas tries to avoid packet loss by slowing down (reducing cwnd) when RTT starts to increase
  - **Assumption**: congestive loss is preceded by buildup in router queue which can be sensed by the increasing RTT





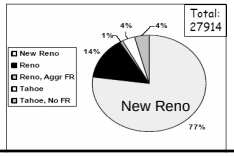
TCP choices

- Scalable, stable, fair, friendly, available

Flavor	control method	trigger	response
TCP	AIMD(1, ½)	loss	W = W – ½ W
		ACK	W = W + 1/W
HSTCP	AIMD(a(W),b(W))	loss	W = W – a(w)W
		ACK	W = W + b(W)/W
STCP	MIMD(1%,1/8)	loss	W = W – W/8
		ACK	W = W + 0.01
N TCPs	AIMD(N,1/(2N))	loss	W = W – W/(2N)
		ACK	W = W + N/W
BI-TCP	AIMD(b,1/8)	loss	W = W – W/8
		ACK	W = W + binary incr.
TCPW	AI?(1,FSE)	loss	ssthresh = RTTmin*FSE
		ACK	W = W +1/W
FAST/Vegas	RTT delta	RTT	W = W*minRTT/RTT + α

- Survey says

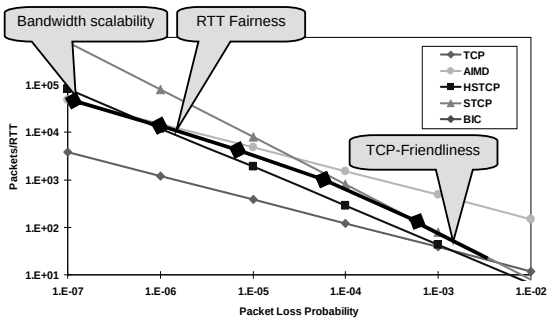
Feb 2004 TBIT results



ECN < 1%
SACK ~ 80%
timestamp ~ 13%
window 64k ~ 73%
window scale ~ 15%

Response Functions AIMD(a,b)

$$bw = \frac{MSS\sqrt{2-b}\sqrt{a}}{RTT\sqrt{2b}\sqrt{p}}$$



TCP 'n it

- Read about it
- Program it
- Model it
- Simulate it
- Measure it
- Diagnose it
- Implement it
- Accelerate it
- Cuss it
- Understand it?

Objectives

- Writing internet software (TCP and UDP)
- Understanding Internet protocols
- Measuring, diagnosing, understanding network performance
- Simulating network performance
- Optimizing TCP performance
- Becoming a network wizard

finale

- Final in ~dunigan/ipp05/final.pdf
  - Take home
  - Open book/notes
  - Due Sunday dec 11 6 pm

- Powerpoint lecture slides in ~dunigan/ipp05/lecsppt.zip

Amen! Hallelujah!