Internet Programming & Protocols
Lecture 14

TCP models

TCP measurement

midterm   assignment 6

www.cs.utk.edu/~dunigan/ipp/

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, …

LECTURES	
14	Models and measurement
15	emulation and simulation
16	ns
17	S-TCP, HSTCP BI-TCP
18	Bandwidth estimation
19	Vegas, fast, westwood
20	AQM, RED, ECN. XCP
21	Satellite and asymmetric channels
22	Wireless
23	Parallel streams, rate based
24	Kernel implementation
25	Web100 and offload engines
26	Cluster TCP, zero copy
27	review

Evaluating the performance of TCP

- Experimental
  - Standalone testbeds
  - Emulator testbeds
  - Live tests on the Internet
  - Active tools (iperf, ping, traceroute) / passive tools (tcpdump/netflows)
  - Collect flow packet trace, full traffic traces
  - Instrumented kernels (Web100)
- Theoretical
  - Analytical models to characterize a TCP flow
  - Stochastic/statistical models to characterize flow interactions (background)
  - Queuing models to characterize router behavior
  - Linear feedback (control) systems to characterize optimal solutions
- Simulation
  - Repeatable, flexible, instrumented

Performance metrics

- NOC issues: capacity management
  - Utilization (peak, daily, hourly) trends
  - Traffic mix (services), UDP vs TCP
  - Link errors
- Bulk traffic characteristics
  - Statistical distribution (smooth, bursty)
  - Correlations, patterns
  - Interpacket arrival times, packet size distribution
  - RTT jitter/distributions
- Single flow characteristics
  - Jitter
  - Data rate
  - Fair
  - "friendly"
  - duration

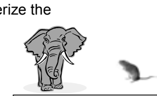
Motivation for TCP modeling

- TCP operating scale is very large
  - Models are required to gain deeper understanding of TCP dynamics
- Uncertainties can be modeled as stochastic processes
- Drive the design of TCP-friendly algorithms for multimedia applications
- Optimize TCP performance

Some network models:
- aggregate traffic models
- queuing models
- control system models
- single-flow models

Modeling Internet traffic

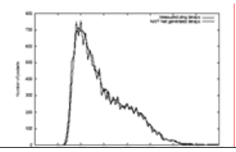
- To design congestion control protocols and provide "realistic" data for network simulations, it is desirable to be able to characterize the "background traffic" of the internet
- Collect traffic traces at the core routers
  - Huge data sets
  - Several sites with data sets available
  - Traffic characteristics evolve over the years
    - telnet → email → http → streaming video → gaming
    - More hosts, faster links
- Long flows? Short flows?
- Are packet arrival rates Poisson? Service rates uniform ?
- No, Internet traffic is bursty, heavy-tailed distributions, self-similar



Elephants & mice
- •A small percent of flows carry bulk of traffic
- •Lots of tiny flow
  - 78% < 10 pkts
  - 95% < 50 pkts

Heavy tailed distribution

- Not nice bell-shaped curves
- Heavy tails
- Correlated
- In your testing/simulation, your background traffic needs to mimic this behavior

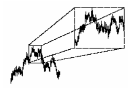


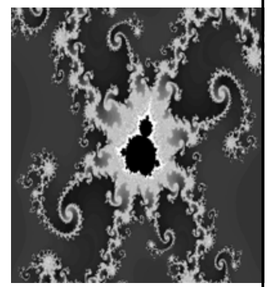
Packet delay distribution

Distribution of file sizes on a computer system is also heavy-tailed.

Self-similarity

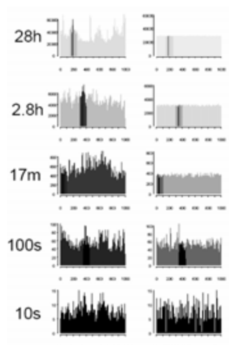
- Real word: visually similar over range of spatial scales
- Fractals: geometrically similar over all spatial scales
- Time series: statistically similar over range of time scale





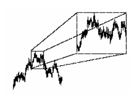
Network traffic self-similar

- Ethernet and Internet traffic appear self-similar [Willinger '95]
- Visual self similarity over 5 orders of magnitude
- X axis time, Y axis is packets/unit-time
- Other network metrics are scale invariant
  - TCP flow durations
  - Bytes/unit time
  - Interarrival times



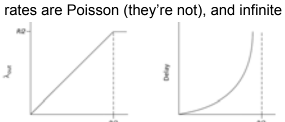
Self-similar traffic

- Traffic is bursty (on-off) Pareto distribution
- Why?
  - Complexity at many levels
    - Geographic scale (wide area internet) multiple routes
    - Concentrated traffic points (universities)
    - Different media speeds (modem, Ethernet, OC192)
    - Different media (wireless, cable)
    - Different services (http, streaming)
    - Temporal complexity
  - Synchronization from router overflows?
- Implications:
  - Markovian models (for modeling or simulation) are not adequate since they allow traffic to be "smoothed" out through finite buffering
  - Using packet loss as a notice of congestion may prevent transport protocols from utilizing available bandwidth



Queuing models

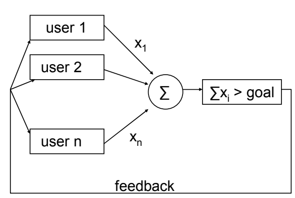
- Network congestion can be viewed as classic queuing problem
- Packets enter router at some arrival rate λ (packets/sec), router tries to forward them on at some (server) rate μ. Queue can build even if λ = μ
  - Server rate == transmission delay, e.g 200kbs link, 40 ms to put 1 KB pkt on wire
  - 10 pkts in queue ahead of you, your RTT increases by 10*40 == 400 ms
- Analytical queuing models allow us to predict queuing times, mean number of packets in the queue, loss rates as function of μ and λ
- For M/M/1 queues, assumptions are service times are exponential, arrival rates are Poisson (they're not), and infinite queues! ☹



But the basic principles apply, throughput increases with the arrival rate, but delay increases as the queues build.

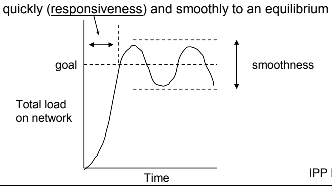
Control system model of a network

- N users sharing resource
- Each user presents a load (x_i) e.g, packets/sec
- Network provides some sort of feedback so users can adjust (increase or decrease) their offered load over time to achieve operating goal

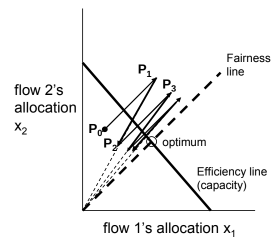


Selecting a rate adjustment algorithm

- linear vs non-linear
  - $x_i(t+1) = a\,x_i(t)+b$
  - $x_i(t+1) = x_i(t) + a\,x_i(t)^k$
- Critera
  - Efficient – operating just under capacity line
  - Fair
    - Roughly, N users should each get 1/N of the capacity
    - $(\sum x_i)^2/(n(\sum x_i^2)) = 1$ fair
  - Converges quickly (responsiveness) and smoothly to an equilibrium



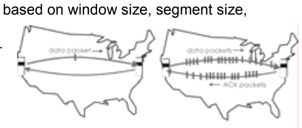
Additive increase, multiplicative decrease (AIMD)



- If over-utilized, decrease rapidly (conservative)
- If under-utilized, increase gradually
- Converges to optimum!
- Jain ('87) (DECnet) suggested
  - Multiplicative decrease 7/8
  - Additive increase 1
- If net congested, decrease rate multiplicatively, otherwise increase rate additively

Models of a TCP flow

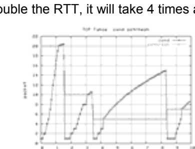
- We have already developed several models to characterize a single network flow in a steady-state based on window size, segment size, and RTT
  - Bandwidth = windowsize/RTT
  - BW(t) = W(t)/RTT



- Latency = transmission delay + propagation delay
- Bandwidth-delay product
  - Given path RTT and capacity C (bits/second), for sliding-window flow control, user must provide RTT*C bits of buffer space at both ends in order to run at "full speed"

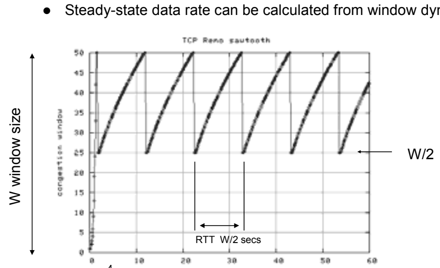
Models of TCP congestion control

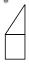
- TCP slow-start
  - Data rate (cwnd) doubles for each arriving ACK
  - To reach window size of N segments, takes $\log_2(N)$ RTT's
  - After k RTT's, instantaneous data rate is $(2^{k+1} - 1)$MSS/RTT
- TCP AIMD
  - Cut sending rate in half if congestion is detected (packet loss)
  - Cwnd increased by 1 each RTT
    - In one second we will add (1/RTT) segments
    - So at end of that second we will have sped up by **MSS/RTT²** bits/sec
    - If you double the RTT, it will take 4 times as long to reach data rate



Reno sawtooth

- TCP Reno sender with send buffer bigger than router queue size
- Steady-state data rate can be calculated from window dynamics





Data rate: (area of box + area of triangle)/time

$(W/2)^2 + ½\,(W/2)^2 =$ **3/8 W² per cycle = ¾ W /sec**

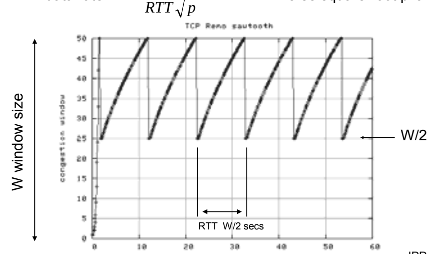
TCP with periodic loss

- We get the same saw-tooth if we have a path with constant packet loss probablity of 1/p -- there is a packet loss after every p packets

Packets per cycle = $(W/2)^2 + ½\,(W/2)^2 = 3/8\,W^2 = 1/p$ or **W = sqrt(8/3p)**

data rate = (# pkts * MSS)/cycle-time = (MSS/p)/(RTT W/2)

data rate = $\dfrac{MSS\sqrt{3/2}}{RTT\sqrt{p}}$ inverse square-root p law

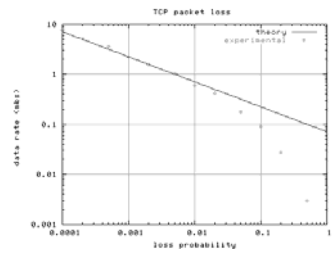


Inverse square root p law $\frac{MSS\sqrt{3/2}}{RTT\sqrt{p}}$

- If path has RTT of 200 ms and a loss probability of 0.05, then average data rate with MSS of 1500 bytes is
  - 1500/0.2 sqrt(1.5/0.05) = 41 Kbytes/sec
- For a satellite path, RTT 590 ms and BER 10^{-5}, max data rate is 8 mbs
- Looking at it the other way: if you want to sustain a data rate of 10 gbs over a 100 ms RTT path, your loss rate must be less than 10^{-14}
  - Can even fiber do this?
- As always, longer RTT aggravates the problem, and a bigger MSS (MTU) can improve performance

Theory vs real TCP Newreno $\frac{MSS\sqrt{3/2}}{RTT\sqrt{p}}$

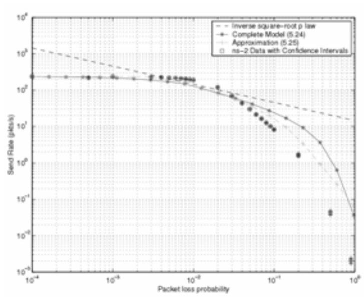
- Model is good for low to moderate loss
- Not so good for high loss – TCP timeouts, backoff not accounted for
- (ns simulation with simple loss model with TCP Newreno)



Refining the model

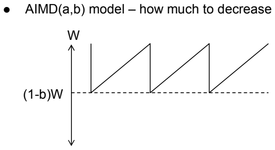
- Model can be refined by factoring in
  - Dup Acks
  - Timeouts
  - Exponential backoff
  - Receiver window limits

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)}$$



Variations on AIMD

- AIMD(a,b) model – how much to decrease by, how much to increase by



Window cut to (1-b)W
Duration: (b/a)W + 1
Loss rate: 1/p ➔ p packets in a cycle

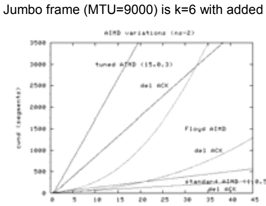
$$W = \sqrt{\frac{2a}{b(2-b)p}}$$

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

- TCP uses AIMD of (1, ½) – inverse square root p law says $bw = \frac{\sqrt{3/2}}{RTT\sqrt{p}}$
- Equating two, we get a = 3b/(2-b)
- So for your congestion control algorithm you could choose any (a,b) but if you want to be TCP-friendly, they should be related as above
  - Example instead of cutting by ½ cut by 1/8 (b=1/8) then *a* should be 1/5
    - AIMD (1/5, 1/8) takes 5 RTT to increase cwnd by 1

Long Fat Networks (LFN)

- TCP linear recovery on paths with high bandwidth and long RTT
  - Takes cwnd/2 RTT's and slope of line is MSS/RTT^2 /sec bits/sec
  - 10 Gig, 100 ms RTT needs window of 83,333 segments
    - Recovering from cwnd/2 takes 4,166 seconds – over an hour!
- Some not so TCP-friendly proposals to speed recovery for LFNs
  - Floyd's HS TCP (a,b) a function of current cwnd (table lookup)
  - Scalable TCP (1%,1/8), increase cwnd by 1% each ACK
  - Virtual MSS (k, ½), increase cwnd by k/cwnd for each ACK
    - Jumbo frame (MTU=9000) is k=6 with added benefits



Easy to experiment with AIMD in ns
$tcp set decrease_num_ 0.875
$tcp set increase_num_ 32

Experimental 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

Real tests or simulations

- **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

Live Internet tests

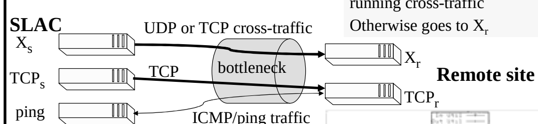
SLAC/s TCP stack tests 2003

- Series of tests of different TCP stacks (NewReno, HSTCP, STCP, Westwood, FAST, BI-TCP) over the wide area
- Test nodes across US, Europe
  - Different pairs participating at different times
  - GigE interfaces, 0C12 bottleneck (622 mbs)
- Used iperf, ping, UDP (competing traffic), some parallel streams too
  - Vary background traffic, window size (RCVBUF/SNDBUF)
  - Measure datarate and RTT (from concurrent ping)
  - Measure stability in face of UDP background
  - Measure fairness with competing TCP stacks

Measurements

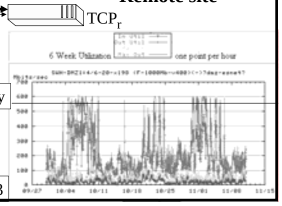
- 20 minute tests, long enough to see stable patterns
- Iperf reports incremental and cumulative throughputs at 5 second intervals
- Ping interval about 100ms

Ping traffic goes to TCP_r when also running cross-traffic
Otherwise goes to X_r



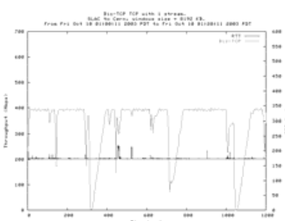
Over a thousand 20 minute measurements or 300 hours

600 Mbps capacity

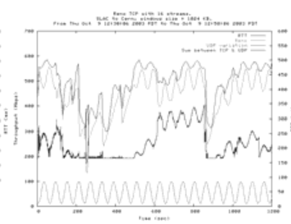


Utilization of SLAC ESnet link Sep-Nov '03

Bandwidth and RTT (SLAC to CERN)

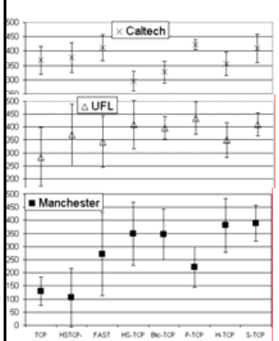


BI-TCP 8 MB window



NewReno (16 streams) with UDP

Throughput



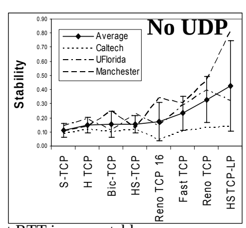
Avg throughput for optimal & large window sizes from SLAC to CalTech, UFl & Manchester

Stack more important for long RTTs

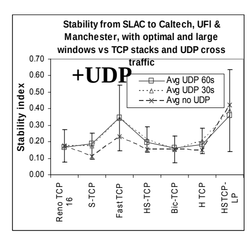
Single stream Reno & HSTCP-LP poorer on large RTTs

Stability

Stability from SLAC to Caltech, U Florida & Manchester



Short RTT is more stable



Little difference between periodicity of UDP (30 & 60 secs)
HSTCP-LP & FAST have larger stability indices (less stability)

Fairness (F)

Avg Fairness from SLAC to UFl. Cross-traffic=> Source	Reno TCP 16	S-TCP	Fast TCP	HS-TCP	Bic-TCP	H TCP	HSTCP-LP	Avg
P-TCP	1.00	0.92	0.89	0.90	0.95	0.94	0.69	0.90
S-TCP	0.92	1.00	0.87	0.90	0.91	0.92	0.78	0.90
Fast TCP	0.89	0.87	1.00	0.92	0.93	0.99	0.78	0.91
HS-TCP	0.90	0.90	0.92	0.97	0.95	0.94	0.95	0.93
Bic-TCP	0.95	0.91	0.93	0.95	1.00	0.99	0.93	0.95
H-TCP	0.94	0.92	0.99	0.94	0.99	1.00	0.95	0.96
HSTCP-LP	0.69	0.78	0.78	0.95	0.93	0.95	1.00	0.87
Average	0.90	0.90	0.91	0.93	0.95	0.96	0.87	0.92

- Most have good intra-protocol fairness (diagonal elements), except HS-TCP
- Worse for larger RTT (Caltech F~0.999+-0.004, U Florida F~0.995+-0.14, Manchester F~0.95+-0.05)
- Inter protocol Bic & H appear more fair against others
- Worst fairness are HSTCP-LP, P-TCP, S-TCP, Fast, HSTCP-LP
- But cannot tell who is aggressive and who is timid

Preliminary Conclusions (SLAC)

- Advanced stacks behave like TCP-Reno single stream on short distances for up to Gbits/s paths, especially if window size limited
- TCP Reno single stream has low performance and is unstable on long distances
- P-TCP is very aggressive and impacts the RTT badly
- HSTCP-LP is too gentle, **this can be important for providing scavenger service without router modifications**. By design it backs off quickly, otherwise performs well
- Fast TCP is very handicapped by reverse traffic
- S-TCP is very aggressive on long distances
- HS-TCP is very gentle, like H-TCP has lower throughput than other protocols
- BI-TCP performs very well in almost all cases

Next time …

- Network emulation
- Network simulation (ns)