Measurements of End-to-End Internet Dynamics

Vern Paxson

Network Research Group
Lawrence Berkeley National Laboratory
EECS Division
University of California, Berkeley

vern@ee.lbl.gov

DIMACS Workshop on
Performance of Realtime Applications on the Internet

November 7, 1996

Overview:

Why characterizing end-to-end Internet dynamics is hard.

A measurement framework for tackling this.

Analysis headaches.

TCP pathologies.

Network pathologies.

Results [*] on characterizing traffic dynamics.

Future work.

Characterizing end-to-end Internet dynamics:

Packet loss & delay, congestion levels, available bandwidth *crucial* for understanding how the network behaves, how to adapt existing protocols to it better, how to design new protocols.

Little is known. [Mogul92, Sanghi93, Bolot93] all studied end-to-end packet behavior, but for just a few traces or sites. How representative?

The Internet is tremendously diverse — no single site or path is “typical” or “representative”.

A framework for conducting end-to-end measurements:

Key idea: run a specially designed measurement process at a number of Internet sites.

The N^2 effect: if there are N measurement sites, adding just one more adds $2N$ more Internet paths that can be measured end-to-end. Total number of measurable paths: $O(N^2)$.

With enough sites and repeated measurements, can capture a reasonably representative cross-section of Internet behavior.

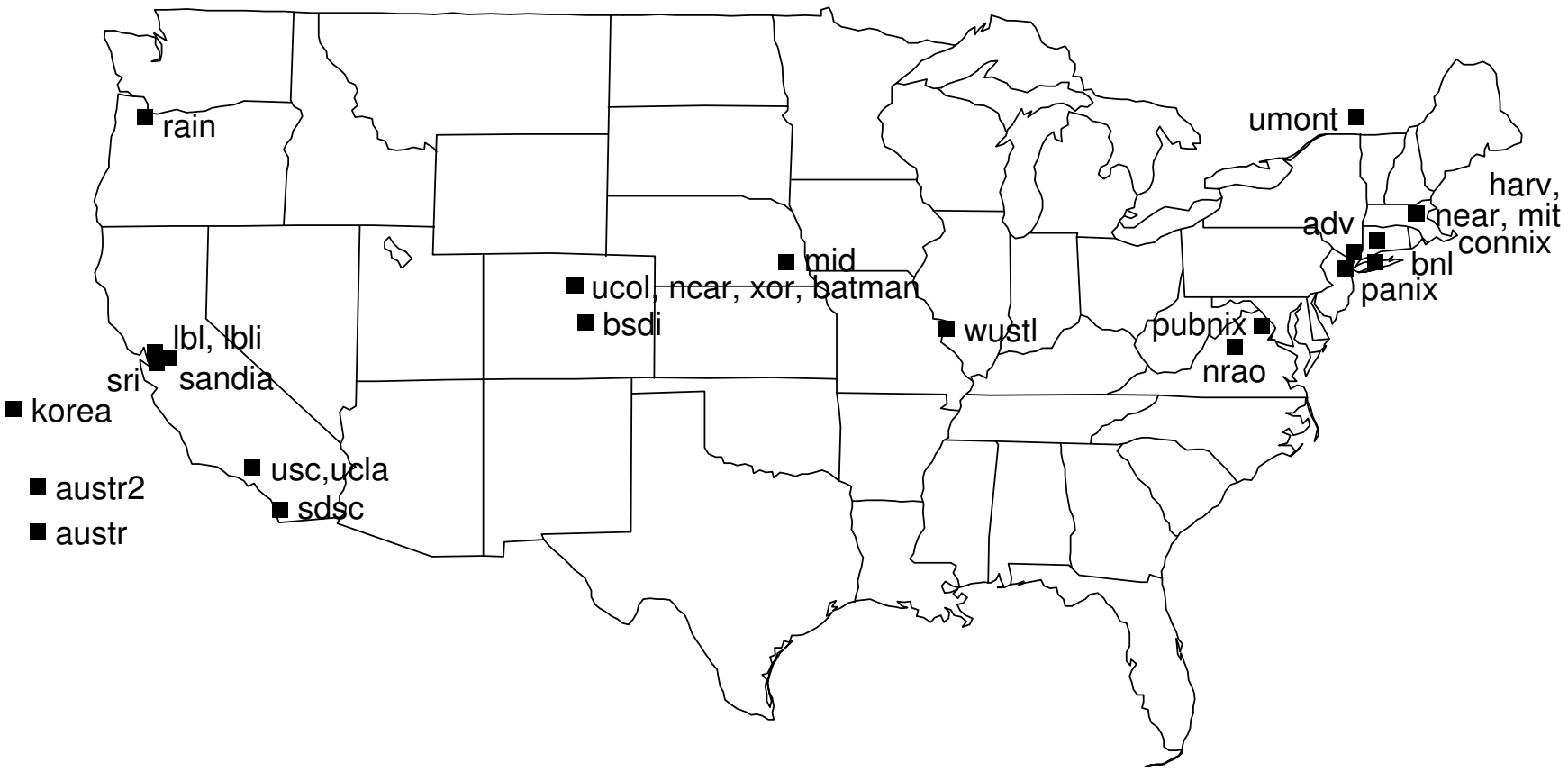
The Network Probe Daemon (NPD):

NPD runs at different sites around the Internet.

It accepts (authenticated) requests to:

- `traceroute` to a remote host
- source/sink a TCP probe
- run `tcpdump/rtcpdump` to monitor a probe
- send/reset log
- self-test

Participating sites — North America



Links traversed — North America

End-to-end routing behavior:

SIGCOMM '96 paper analyzes 40,000 repeated `traceroute`'s between 37 Internet sites. Looks at pathologies, routing stability, routing symmetry.

Pathologies: significantly increased during 1995.
Stability: most paths heavily dominated by a single route.
Persistence: 2/3's of the routes unchanged over periods of days to weeks.
Symmetry: at the end of 1995, half the routes visited at least one different city in the two directions.

How to probe:

Why TCP?

- Real-world traffic
- Fine-grained time scales . . .
- . . . but with congestion control

Why not TCP?

- Endpoint TCP behavior makes analysis a major headache
- Self-interference makes analysis a major headache
- Note: acks don't self-interfere! [*]

The raw data:

$\mathcal{T}_1$: 2,822 100 KB TCP transfers between 25 sites, traced at both ends, Dec 1994. 730,000 packets.

$\mathcal{T}_2$: 18,490 100 KB TCP transfers between 31 sites, traced at both ends, Nov–Dec 1995. 4,850,000 packets. Big windows.

23 sites in common.

($\mathcal{T}_3$: 979 1 MB TCP transfers between 28 sites, Jan 1996.)

Analysis headaches — experimental design:

10-minute max. measurement lifetime introduces bias towards *underestimation*.

Centralized architecture biases towards *underestimation*.

These could be fixed by having NPD “batch” requests.

Analysis headaches — packet filter prevarication:

Drops: present in 15% of $\mathcal{T}_1$ measurements, 10% of $\mathcal{T}_2$.

Truncations: frequently with Solaris 2.3.

Extras: frequently with IRIX machines.

Reorderings: frequently with Solaris.

Causality: e.g., see Ack_{512}, Ack_{1024}, Pkt_{1024}.

Analysis headaches — matching sender and receiver:

Because of retransmissions, it's not obvious which packets sent at one end match those seen at the other.

Idea: use IP ID number in addition to sequence numbers.

Problem: Some TCP's (Linux) reuse IP ID on rexmits, too.
Problem: Some forms of SLIP errors change the ID's!

But with some timing heuristics can generally pair up the measurements.

Analysis headaches — clocks:

We don't require synchronized clocks. (Worse disagreement: 2.4 days!)

Time travel: 0.14% of traces in $\mathcal{T}_1$ (single site), 2.9% in $\mathcal{T}_2$ (two sites).

Forward adjustments: trickier to detect.

Resolution: 50 μsec to 10 msec. Some OS's are misleading.

Clock headaches, con't:

Skew: tricky to assess, due to network dynamics.

Method: suppose S at local time s_1 sends a packet to R, which arrives as r_1. Later R sends at r_2 to S, arriving at s_2.

Should always have:

$$s_2 - s_1 > r_2 - r_1,$$

$$s_2 - s_1 \geq r_2 - r_1 + \text{RTT}_{\text{min}}$$

Skew $>$ 1%: 3% of traces in $\mathcal{T}_1$, 2% in $\mathcal{T}_2$.

How bad can it get? 448%!

Analysis headaches — TCP behavior:

Crucial to understand packet cause & effect: to assess RTT's, measurement drops, unusual network behavior.

Analysis program `tcpanaly` knows about 16 different implementations:

SunOS, Solaris 2.3/2.4, Digital OSF/1, IRIX 4.0/5.1/5.2/6.2, BSDI 1.1/2.0/2.1, NetBSD 1.0, HP/UX 9.05/10.00, VJ 1/2, Linux

All but Solaris, and Linux are Tahoe/Reno variants.

Can generally infer checksum errors, ICMP Source Quench.

TCP pathologies:

Broken retransmit timers: Solaris!

Entire window retransmitted upon loss: Linux!

(If the whole world was Solaris 2.3/2.4 and/or Linux, *the Internet would collapse*.)

Slow start skipped: some versions of Reno! But only under fairly rare conditions.

Network pathologies — out-of-order data delivery:

30% of traces in $\mathcal{T}_1$, 8% of traces in $\mathcal{T}_2$.

2% of packets in $\mathcal{T}_1$, 0.3% in $\mathcal{T}_2$.

Median reordering time: 10 msec [*] in $\mathcal{T}_1$, 4 msec in $\mathcal{T}_2$.

Mean reordering time: 31 msec in $\mathcal{T}_1$, 11 msec in $\mathcal{T}_2$.

(Minimum is bounded.)

Out-of-order delivery — how bad it can get:

Worst connections: 36% of packets in $\mathcal{T}_1$, 28% in $\mathcal{T}_2$.

Worst site: 24% of all packets sent!

Worst gap: 37 packets! [*]

Worst delay: 9 seconds! [*]

Network pathologies — rare & unusual:

Packet replication:

In $\mathcal{T}_1$, happened once: 2 packets in, 16 out.

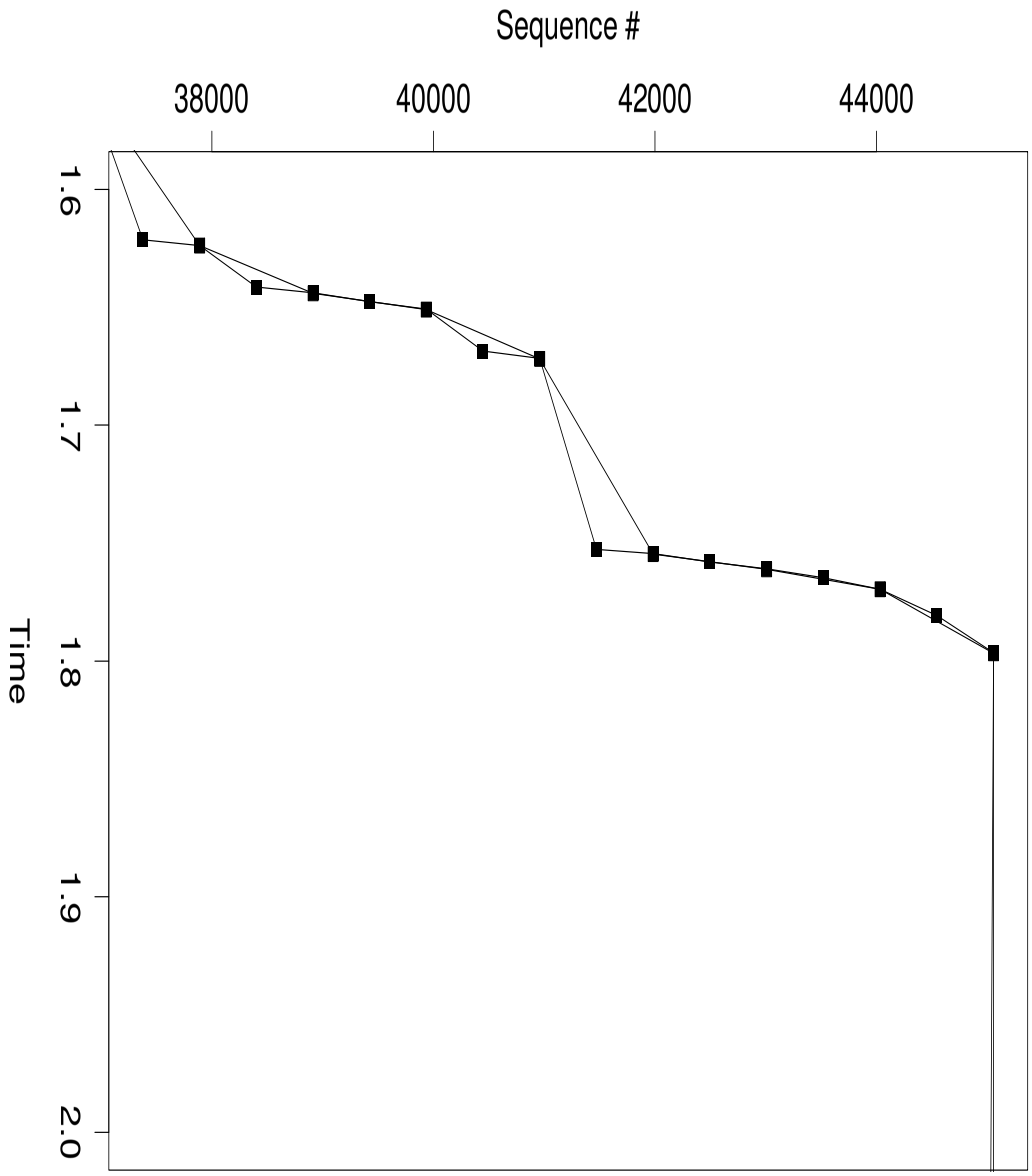
In $\mathcal{T}_2$, 65 times. Most striking: 1 packet in, 22 out.

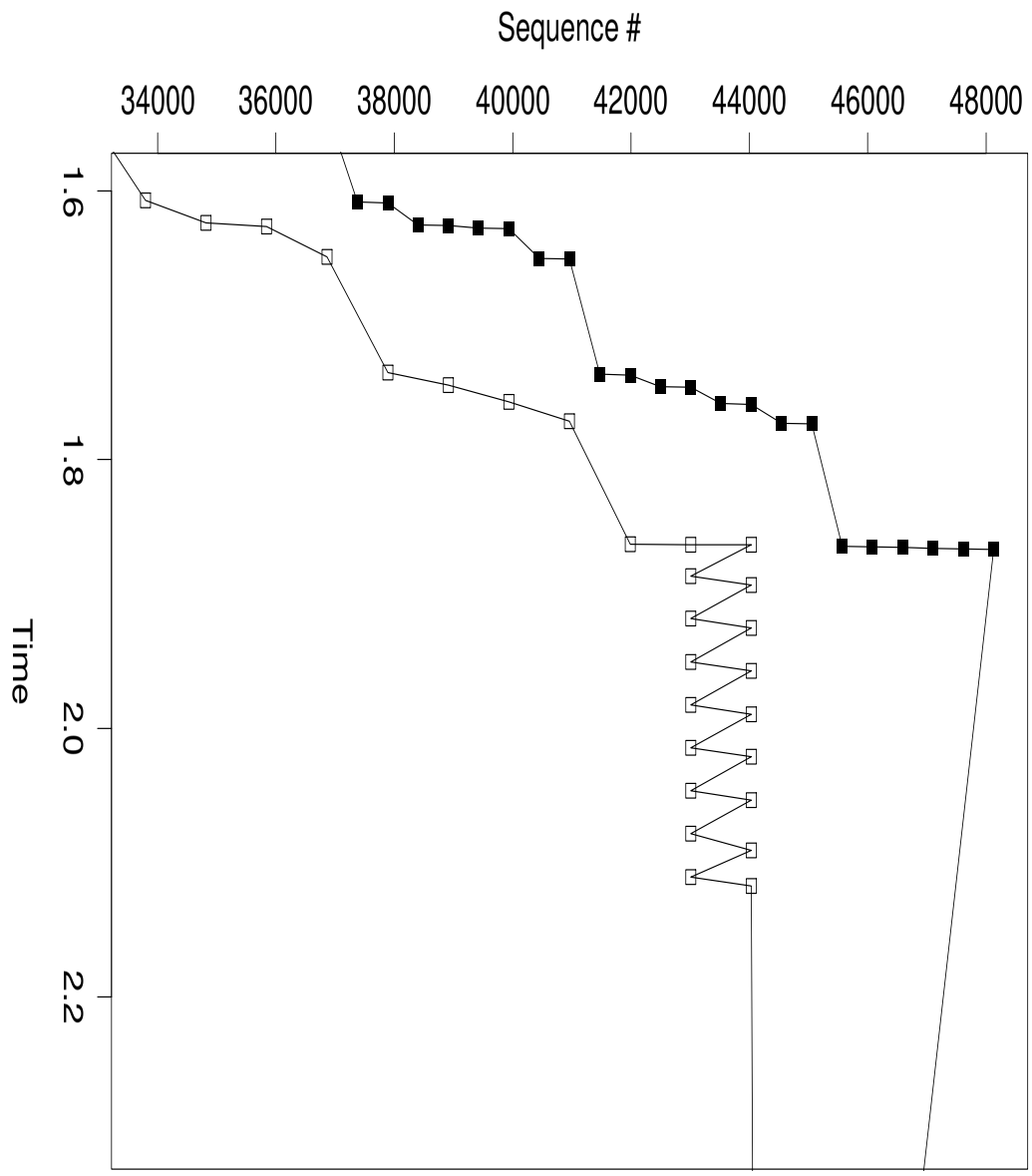
TCP data checksum errors: 3% of conn. in $\mathcal{T}_1$, 4.5% in $\mathcal{T}_2$.

0.02% of packets in $\mathcal{T}_1$, 0.06% in $\mathcal{T}_2$.

For conn. with at least one: 0.7% of pkts in $\mathcal{T}_1$, 1.5% in $\mathcal{T}_2$.

Packet gaining data in flight: twice in $\mathcal{T}_2$!


Sequence #
38000
40000
42000
44000
Time
1.6
1.7
1.8
1.9
2.0


Sequence #
34000
36000
38000
40000
42000
44000
46000
48000
Time
1.6
1.8
2.0
2.2

Packet loss:

In $\mathcal{T}_1$, overall packet loss rate = 3.2%.

In $\mathcal{T}_2$, it rose to 4.8%.

For sites in common, 2.8% $\Rightarrow$ 5.5%!

Loss-free (data pkt) connections: 46% $\Rightarrow$ 26%.

Median overall loss rate: 0.5% $\Rightarrow$ 1.3%.

Median, if any loss: 2.9% $\Rightarrow$ 3.8%.

But: bigger $\mathcal{T}_2$ windows may be (partly?) responsible.

Packet loss, con't:

In $\mathcal{T}_1$, data loss rate vs. ack loss rate: 3.1% vs 3.3%.

In $\mathcal{T}_2$: 4.8% vs 4.7%.

In $\mathcal{T}_1$, correlation between data and ack loss rates: 0.2.

In $\mathcal{T}_2$, correlation: -0.03.

How bad can it get? In $\mathcal{T}_1$, 40%; in $\mathcal{T}_2$, 50/68%.

Biggest outage: 9 minutes, 64 packets.

Biggest retransmit interval: 90 sec.

Packet loss patterns:

Probability data pkt lost if previous not: $\mathcal{T}_1$, 2.1%; $\mathcal{T}_2$, 3.1%.

Probability data pkt lost if previous lost: $\mathcal{T}_1$, 34%; $\mathcal{T}_2$, 40%.

Probability ack lost if previous not: $\mathcal{T}_1$, 2.6%; $\mathcal{T}_2$, 3.4%.

Probability ack lost if previous lost: $\mathcal{T}_1$, 25%; $\mathcal{T}_2$, 32%.

$\Rightarrow$ Packet and ack loss not well described as independent.

Retransmissions:

In $\mathcal{T}_1$, 5.3% of packets retransmitted (only 3.1% lost).

In $\mathcal{T}_2$, 8.3% (4.8%).

Unnecessary retransmissions: 44%, 38%.

Of these, unavoidable (all acks lost): 18%, 22%.

Of avoidable, due broken timer: 70%, 37%.

Due to timer failing to adapt: 3%, 9%.

Spurious: 1%, 3%.

Fixed by SACK: 22%, 51%.

Round trip time (RTT):

Largest seen: 15.1 seconds.

Largest minimum seen: 4.2 seconds.

Mean ratio of maximum to minimum: 3.7 in $\mathcal{T}_1$, 2.7 in $\mathcal{T}_2$.

For $\mathcal{T}_2$, ratio has Pareto tail, $\alpha = 1.8$.

RTT variation:

Ratio of mean MSS RTT to minimum MSS RTT:

For $\mathcal{T}_1$, Pareto with $\alpha = 2.1$.

For $\mathcal{T}_2$, Pareto tail (12%) with $\alpha = 2.3$.

Holds for ratio of median RTT to minimum, too.

$\Rightarrow$ Suggests queues have heavy-tailed distributions — consistent with self-similar traffic?

Could be due to TCP dynamics.

One-way trip time (OTT):

Extremely useful to characterize (1) clocks, (2) queueing, (3) ack & data compression and expansion.

Ratio of median MSS OTT to minimum MSS OTT:

For $\mathcal{T}_1$, Pareto tail (12%), $\alpha = 2.8$

For $\mathcal{T}_2$, Pareto tail (24%), $\alpha = 1.5$!

Ratio of median ack OTT to minimum ack OTT:

For $\mathcal{T}_1$, Pareto tail (11%), $\alpha = 2.7$

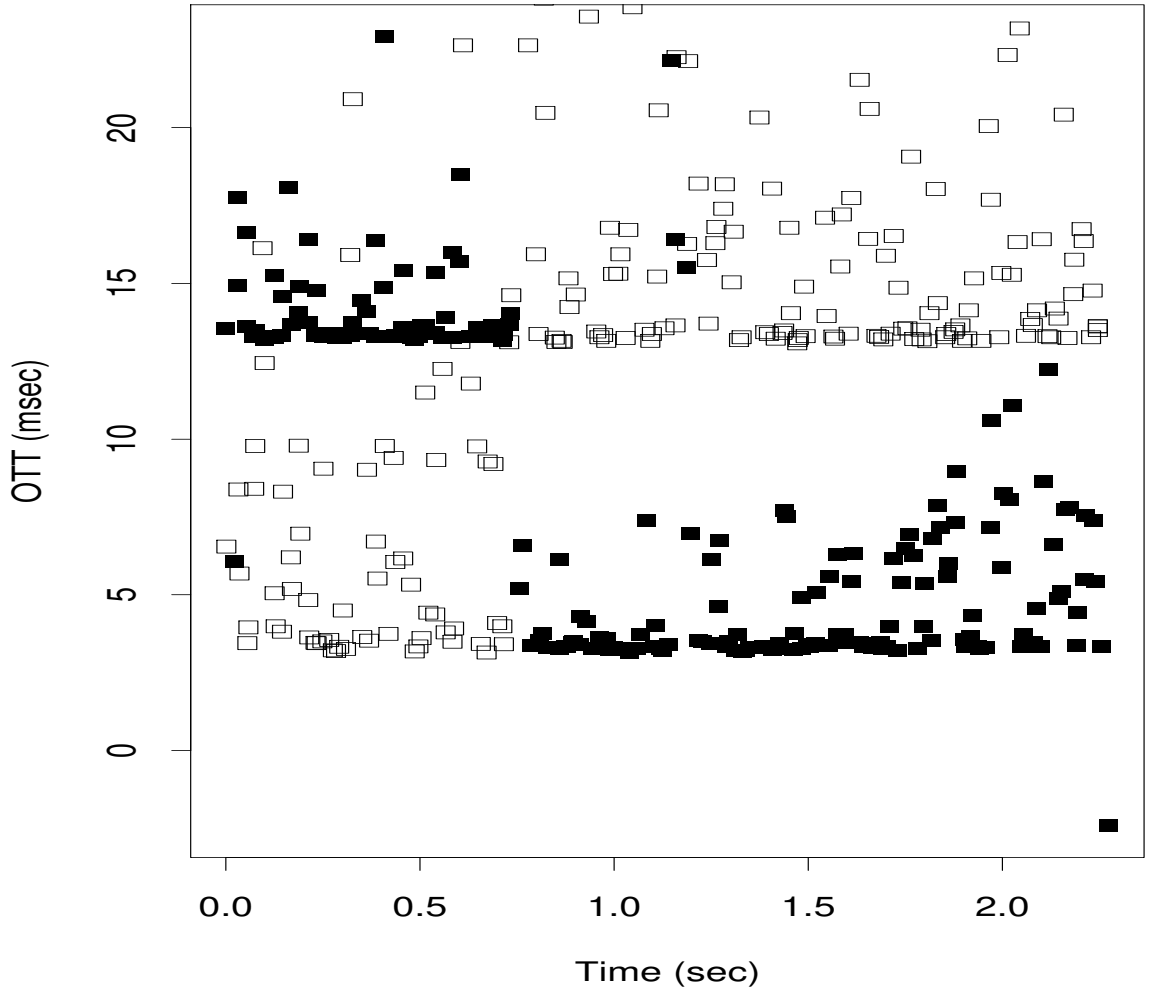
For $\mathcal{T}_2$, Pareto tail (8%), $\alpha = 2.3$

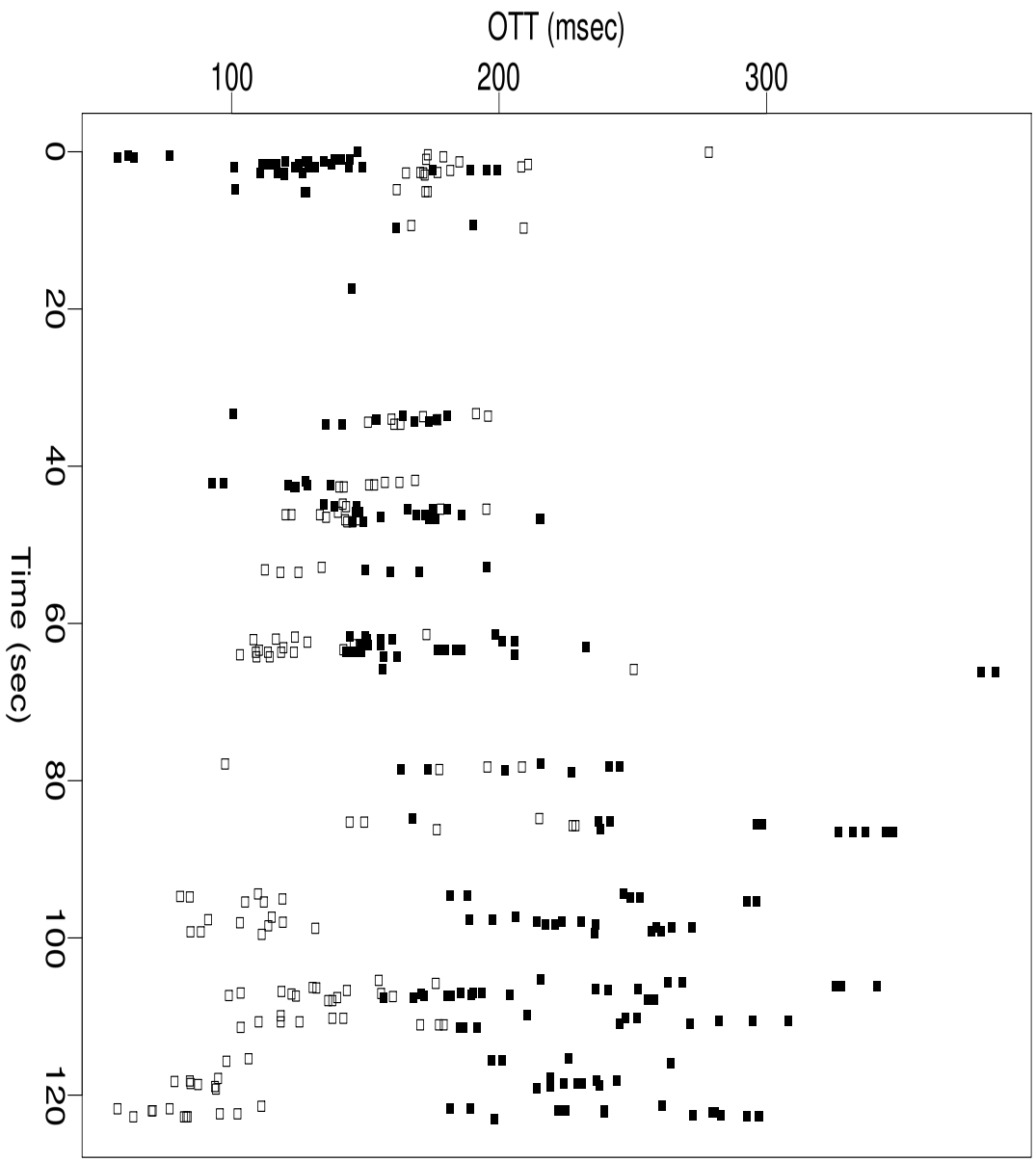
OTT, con't:

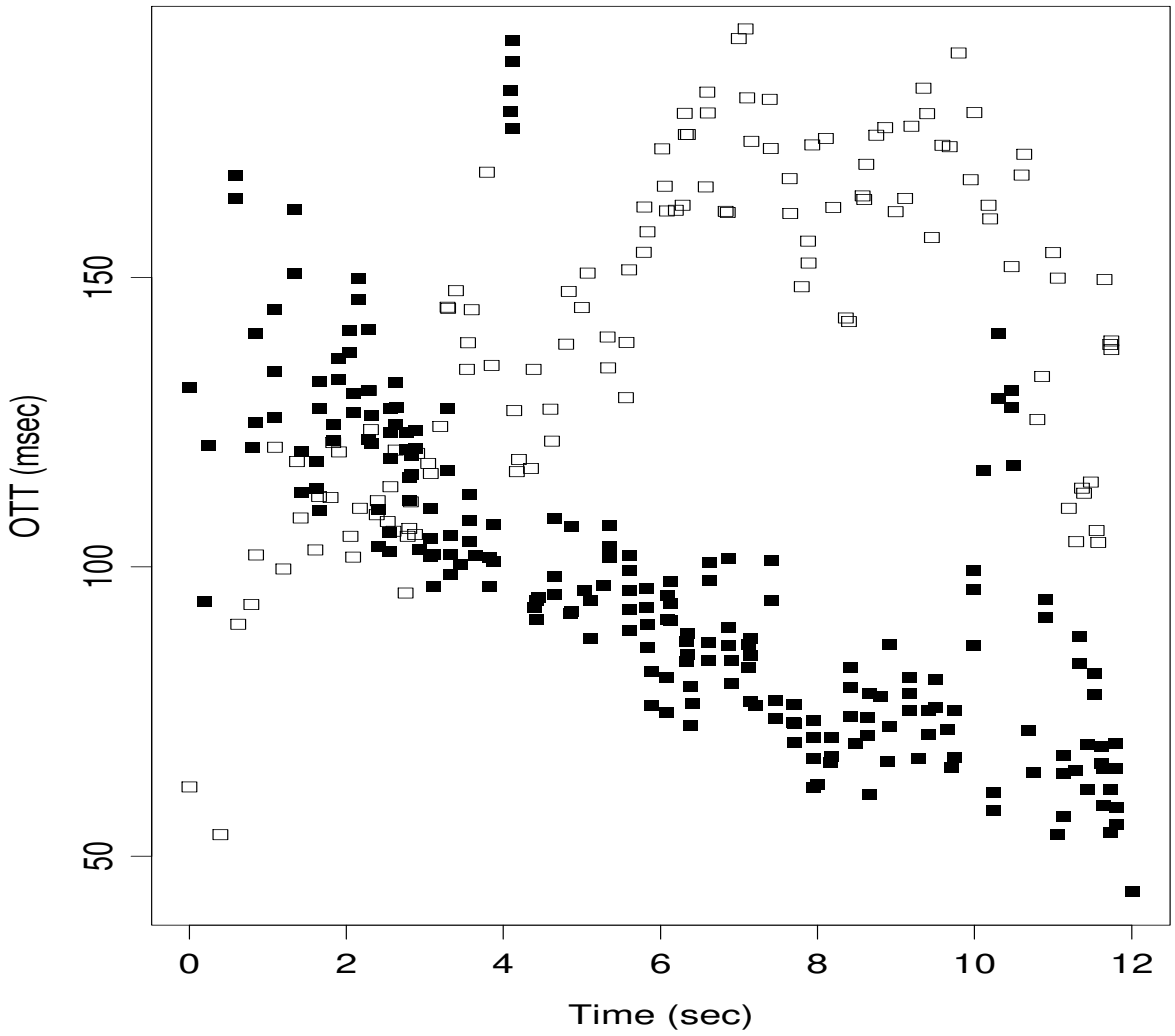
Note: Pareto *tail*, not full distribution.

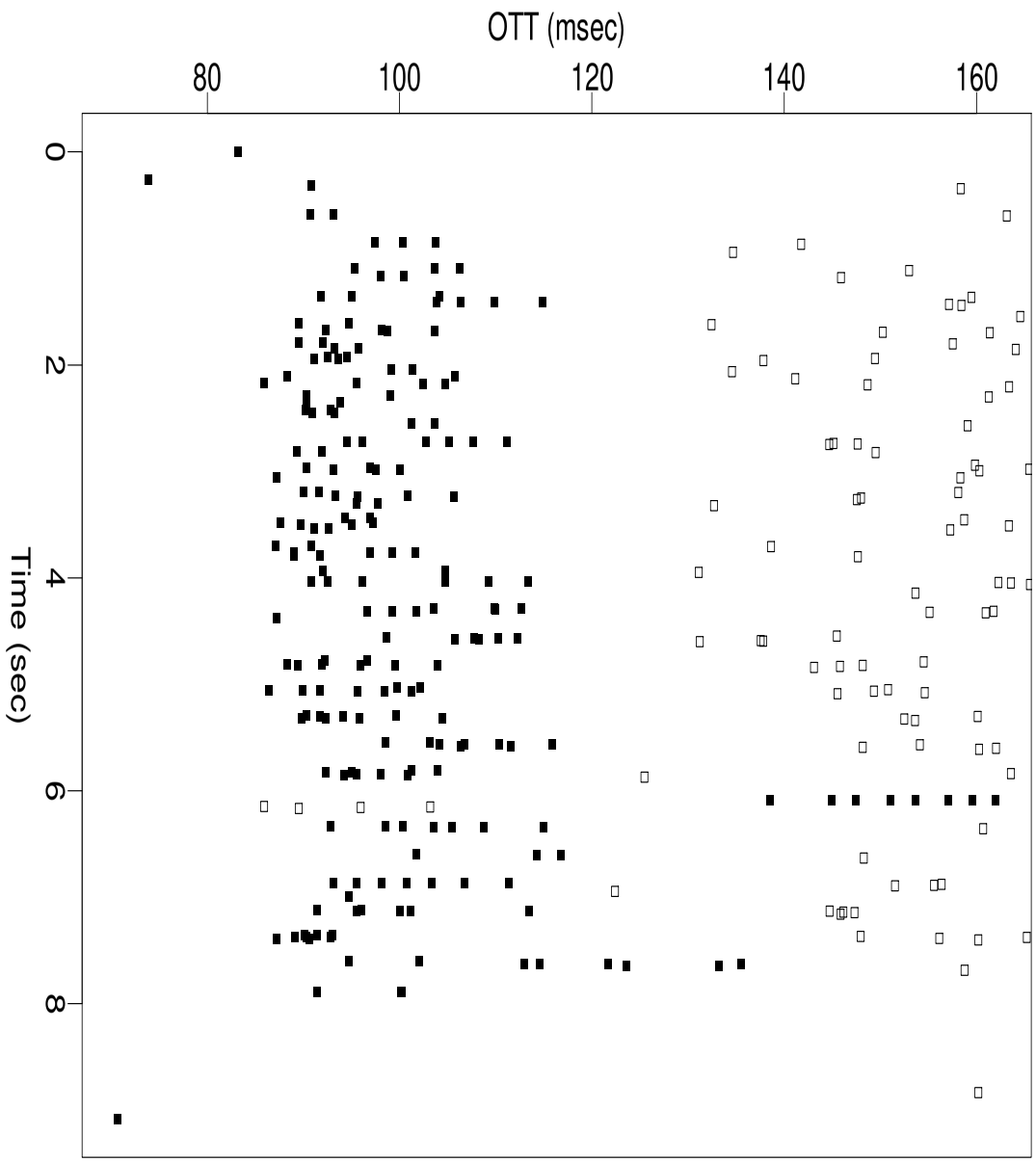
Note: Broken clocks can completely change these!

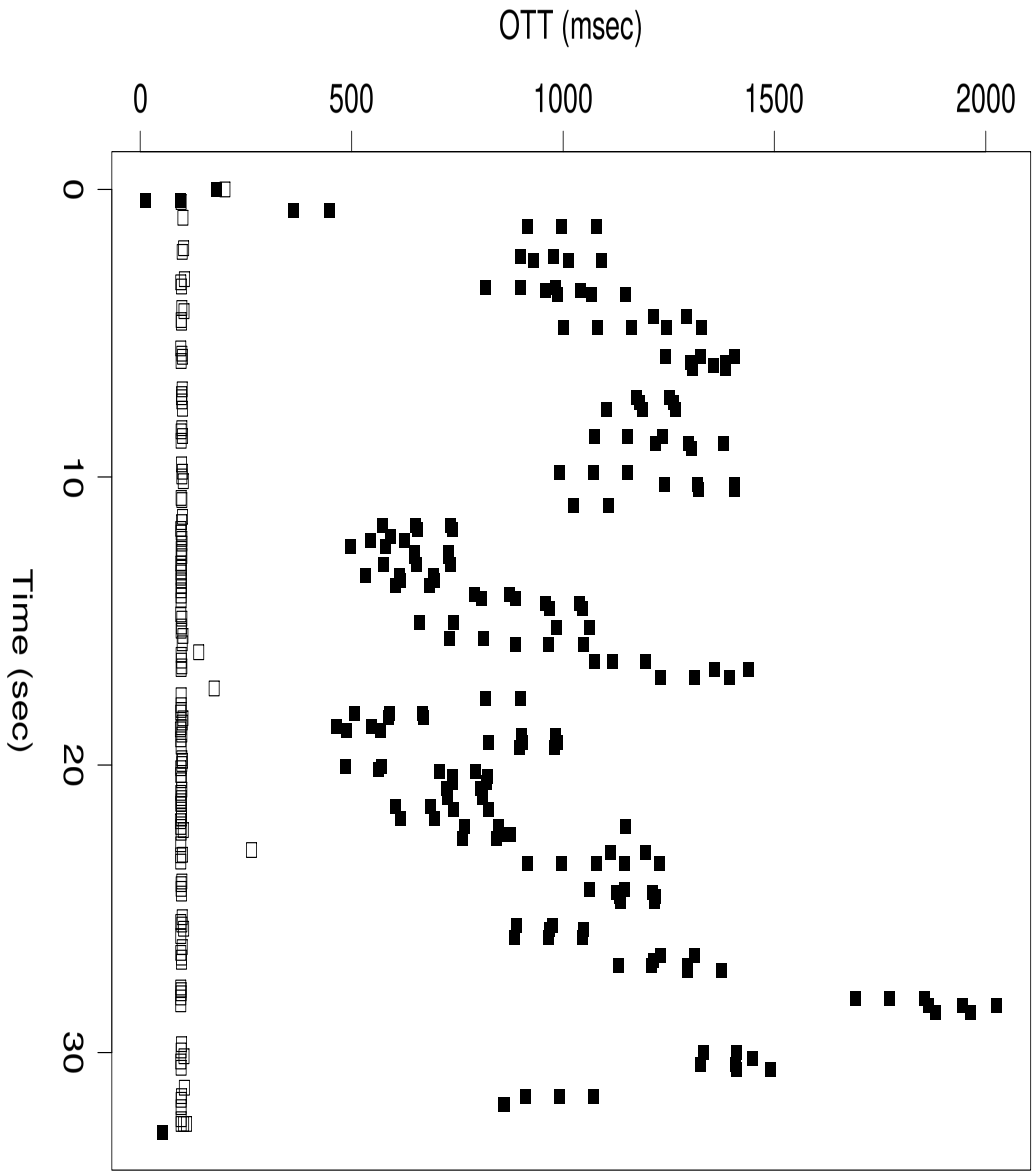
Data/ack ratios correlated (0.14 – 0.19) — suggests bidirectional congestion or *self-interference*.


OTT (msec)
0
5
10
15
20
0.0
0.5
1.0
1.5
2.0
Time (sec)


OTT (msec)
100
200
300
Time (sec)
0
20
40
60
80
100
120


OTT (msec)
50
100
150
0
2
4
6
8
10
12
Time (sec)


OTT (msec)
80
100
120
140
160
Time (sec)
0
2
4
6
8


OTT (msec)
0
500
1000
1500
2000
Time (sec)
0
10
20
30

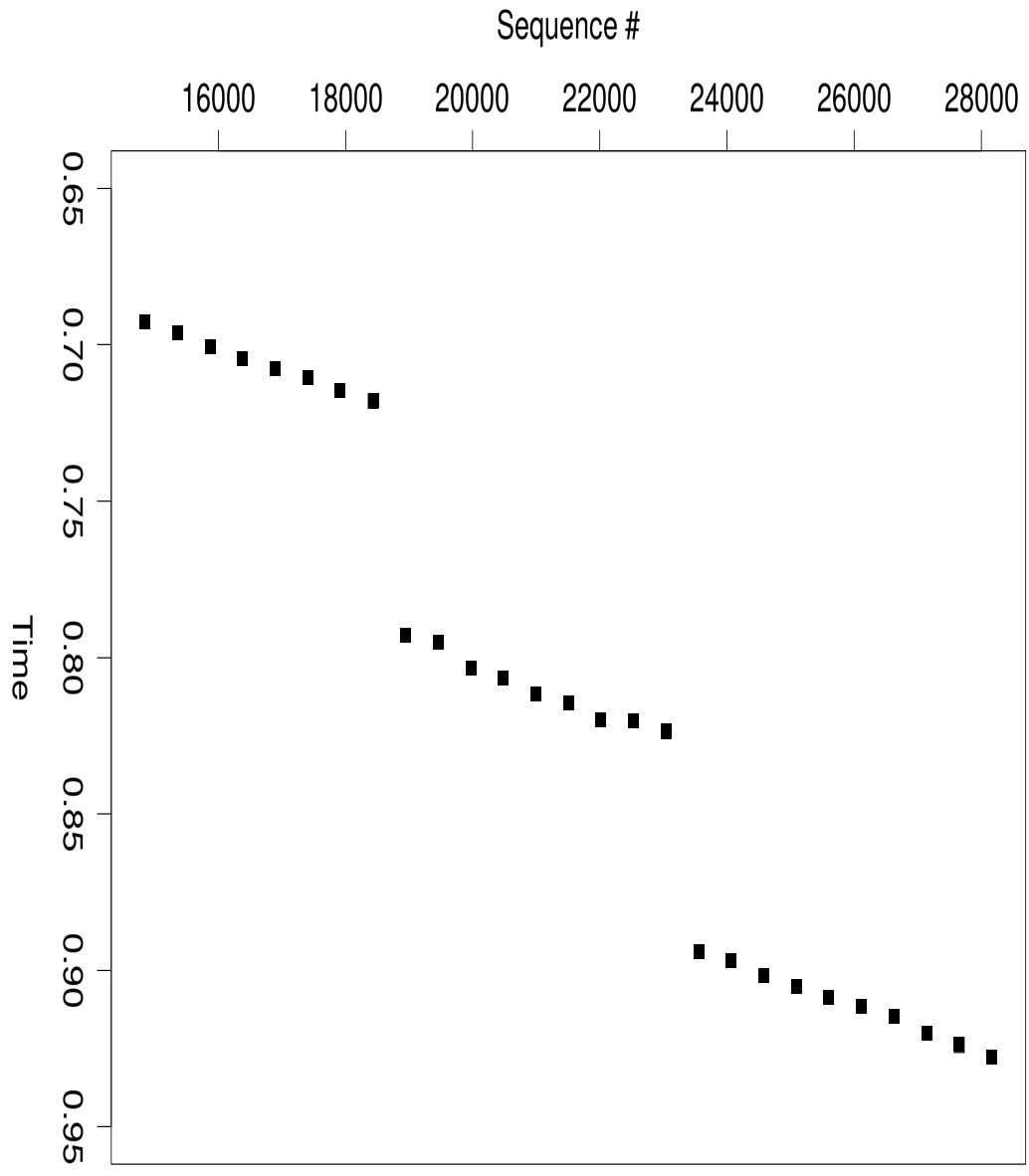
Bottleneck bandwidth:

In general, there's a limiting rate on a connection: the bottleneck bandwidth. Different from available bandwidth.

If you know it $\Rightarrow$ bandwidth $\cdot$ delay product, self-interference time constant. (Also, gives clock consistency check.)

Can estimate *conservatively* as bytes received at receiver during an interval of at least RTT (if you know that!).

Can estimate *optimistically* as bytes received during one "flight."


Sequence #
16000
18000
20000
22000
24000
26000
28000
Time
0.65
0.70
0.75
0.80
0.85
0.90
0.95

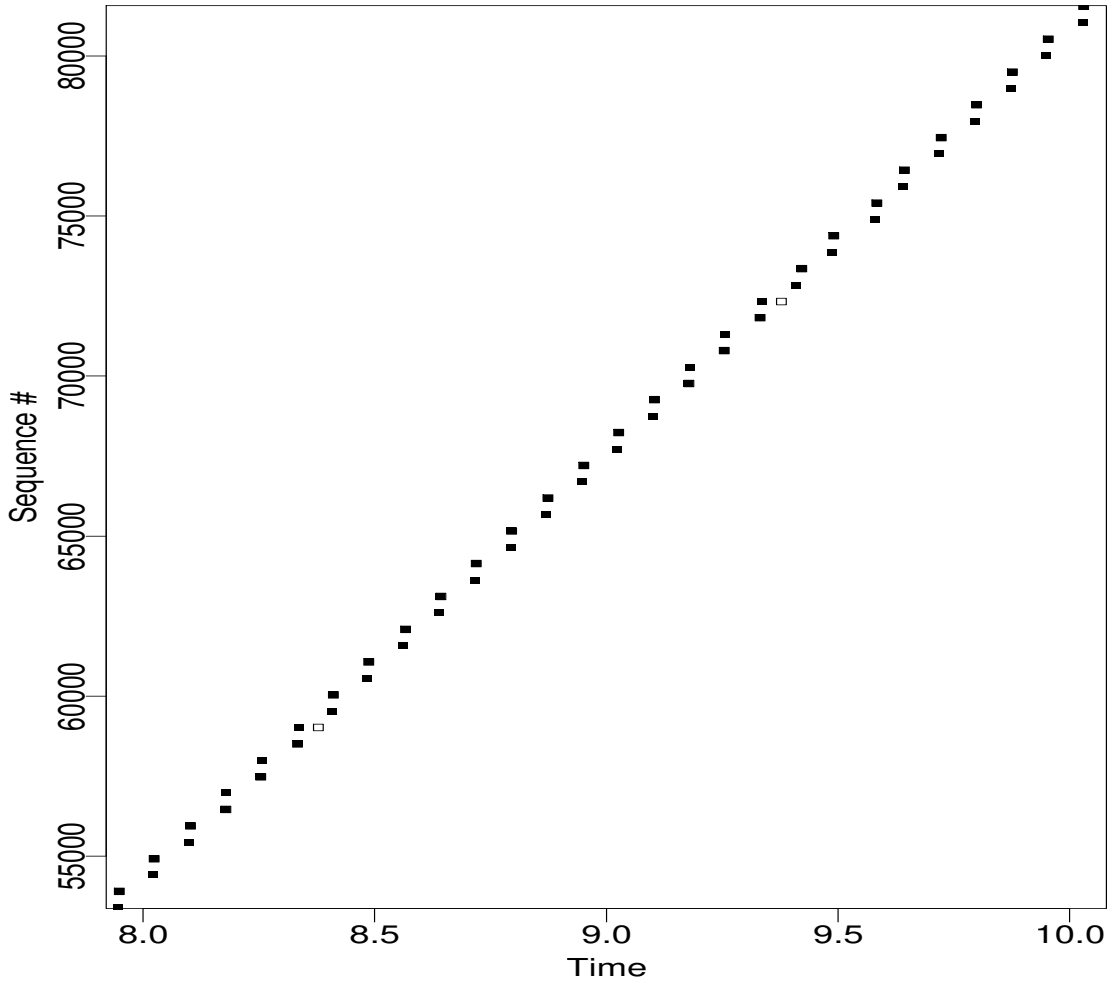
Alternative approach:

Packet pair or packet bunch (Jacobson's self-clocking, Keshav's fair queueing analysis).

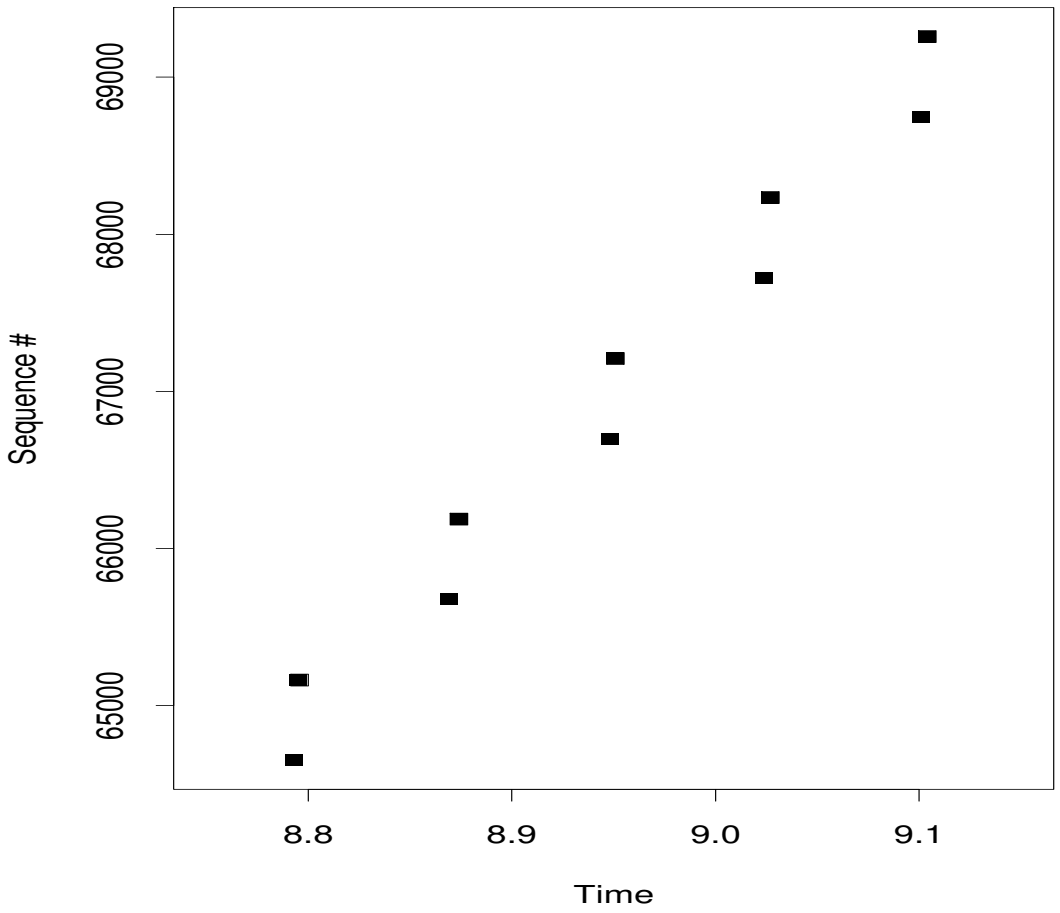
For every n packets that arrive with a spacing at least ρ greater than they were sent, use bytes/ΔT as bandwidth estimate.

Very noisy. Search for *mode(s)* to form robust estimates.
In general, works well.
Should work great when no congestion, but . . . ...

Packet-pair fails:



Packet-pair fails, con't:



Still to do:

More robust clock evaluation. Perhaps remove skew.

Congestion time scales, relationship to LRD models.

Available bandwidth vs. bottleneck bandwidth.

Degree of quiescence.

Effectiveness of sender-only measurement (ack compression).

Quick estimation algorithms.

Mid-stream changes.

Evaluating/rethinking TCP's retransmit timer.

“Bandwidth discovery” — testing Vegas, Janey Hoe's startup.

$\Rightarrow$ How characteristics change over (relatively short) times.

Acknowledgements:

Guy Almes and Bob Camm (`adv`); Jos Alsters (`unij`); Jean-Chrysostome Bolot (`inria`); Hans-Werner Braun, Kim Claffy, and Bilal Chinoy (`sdsc`); Randy Bush (`rain`); Jon Crowcroft and Atanu Ghosh (`ucl`); Peter Danzig and Katia Obraczka (`usc`); Mark Eliot (`sri`); Robert Elz (`austr`); Teus Hagen (`oce`); Steinar Haug and Håvard Eidnes (`sintef1`, `sintef2`); John Hawkinson (`near` and `panix`); TR Hein (`xor`); Tobias Helbig and Werner Sinze (`ustutt`); Paul Hyder (`ncar`); Alden Jackson (`sandia`); Kate Lance (`austr2`); Craig Leres (`lbl`); Kurt Lidl (`pubnix`); Peter Linington, Alan Ibbetson, Peter Collinson,

Acknowledgements, con't:

and Ian Penny (`ukc`); Steve McCanne (`lbli`); John Milburn (`korea`); Walter Mueller (`umann`); Evi Nemeth, Mike Schwartz, Dirk Grunwald, Lynda McGinley (`ucol`, `batman`); Francóis Pinard (`umont`); Jeff Polk and Keith Bostic (`bsdi`); Todd Satogata (`bnl`); Doug Schmidt and Miranda Flory (`wustl`); Sorell Slaymaker and Alan Hannan (`mid`); Don Wells and Dave Brown (`nrao`); Gary Wright (`connix`); John Wroclawski (`mit`); Cliff Young and Brad Karp (`harv`); and Lixia Zhang, Mario Gerla, and Simon Walton (`ucla`).