An Introduction to
Internet Measurement and Modeling

Vern Paxson

Network Research Group
Lawrence Berkeley National Laboratory
vern@ee.lbl.gov

SIGCOMM '98 Tutorial
August 31, 1998

Roadmap:

1. Why understanding the Internet is painfully hard.

2. Measurement tools & techniques.

3. The state of the art.

4. The fractal revolution.

5. Case study: measuring throughput.

Major themes:

1. The Internet is tremendously diverse, complex, and dynamic. Nothing is "typical"!

2. Because it's so diverse, we are desperate for invariants and parsimonious models.

3. Sound measurement requires discipline and a willingness to consider that you might be wrong.

4. Similarly, assessing someone else's work requires skepticism.

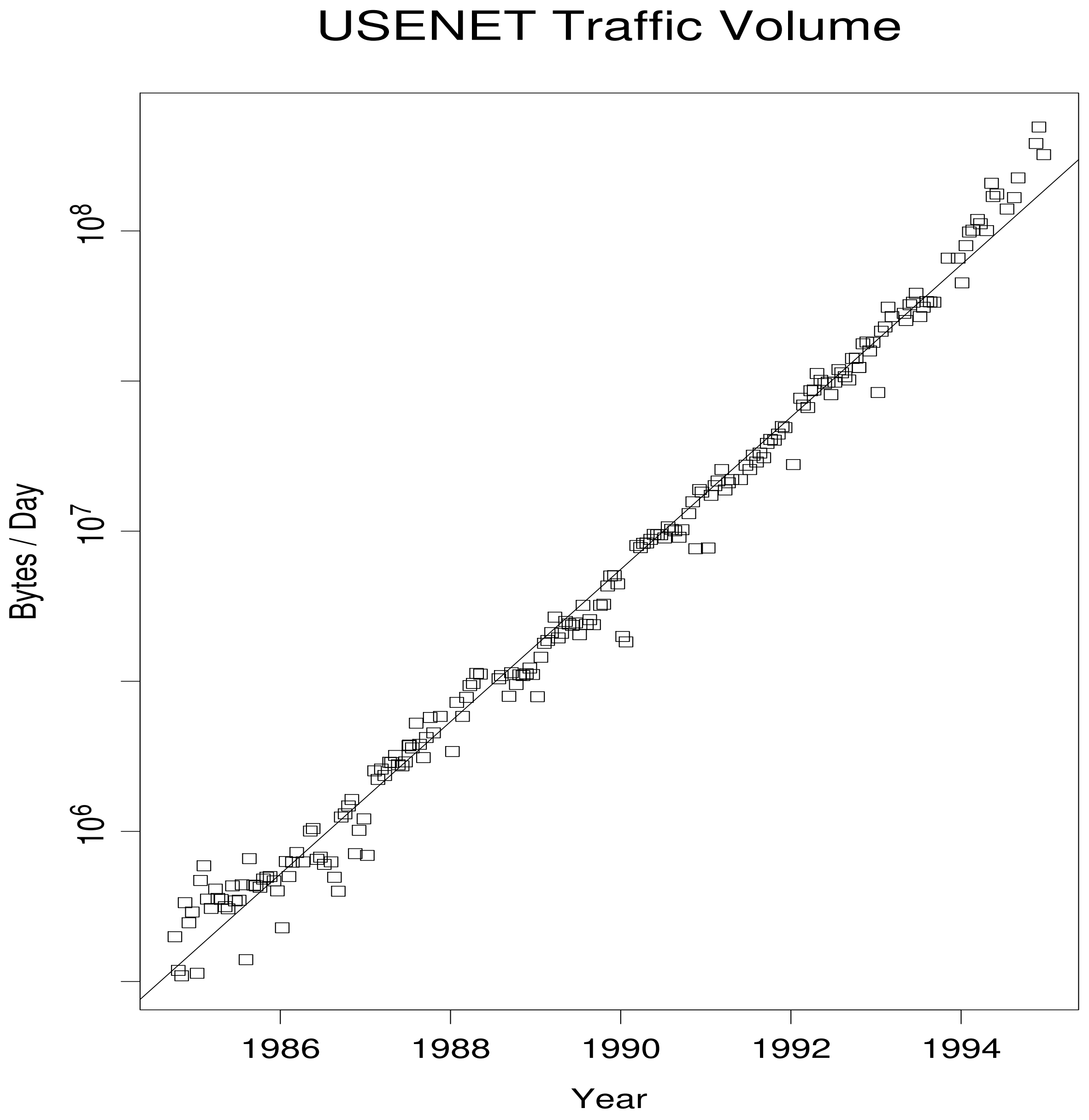
Major themes, con't:

5. Whenever possible, calibrate what you measure.

6. To do so, testing for self-consistency is extremely powerful.

7. Network traffic is highly structured: it pays to know as much about the structure as possible.

8. You don't know anything about what's going on unless you measure!


USENET Traffic Volume
Bytes / Day
10^8
10^7
10^6
1986
1988
1990
1992
1994
Year

The Internet — huge & diverse:

In January, 1990, the Internet had about 200,000 hosts.

Five years later, it had about 4,900,000.

$\Rightarrow$ If you blink, you're out of date.

What about since then?

9.5 million in 1996. 16.1 million in 1997.

(*http://www.nw.com/zone/WWW/top.html*).

Q: How do you measure this?

A: By traversing the entire Domain Name System (DNS) tree, counting hostname $\Rightarrow$ IP address mappings.

The Internet's size in Jan '98:

Q: So what about Jan '98?

A: 29.7M hosts.

So it's basically doubling every year, right?

<u>No.</u>

New methodology: count IP address $\Rightarrow$ hostname mappings (PTR records).

Best guess is previous estimates $\approx$ a quarter too low.

July 1998: 36.7M hosts.

Estimating the size of the Internet, con't:

Different methodology (*http://www.netsizer.com*):

Pick k random IP addresses ($k \approx 10^5$). See how many have associated hostnames (PTR records). Call this l. Then estimate size of the Internet as

$$\text{Size} = \frac{l}{k} \cdot 2^{32}.$$

Estimate for August 12, 1998: 36.6M hosts.

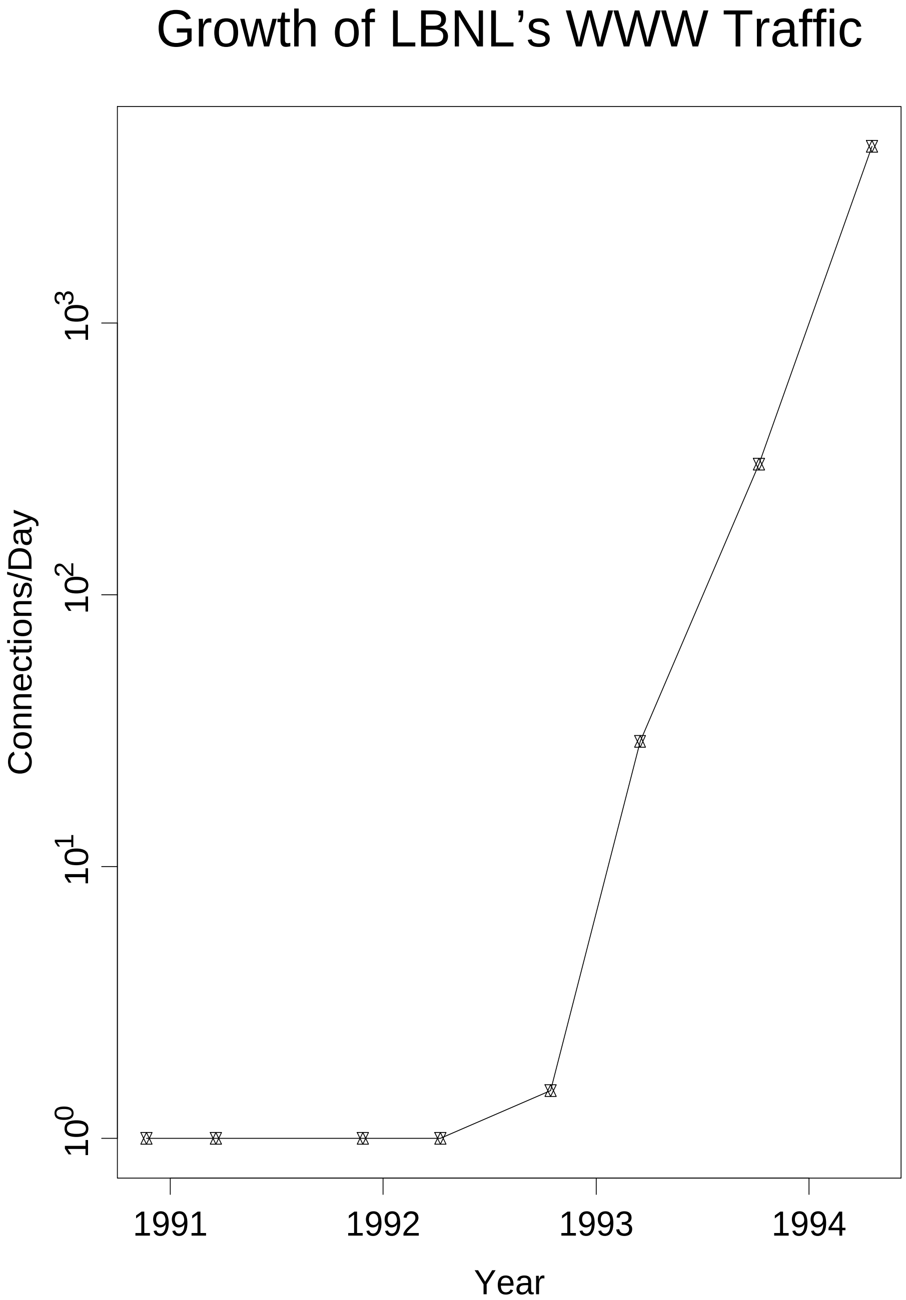
The Internet — huge & diverse, con't:

In January, 1991, about 15% of the data bytes going into LBNL were Network News. At USC, it was 60%.
$\Rightarrow$ There's no such thing as a typical site.

In October, 1992, the median FTP data transfer at LBNL was 4,500 bytes. Five months later, it was 2,100 bytes.
$\Rightarrow$ There's no such thing as a typical time period.

(On an MCI backbone link in Sep. 1997, 70% of the packets were TCP port 80. For UCB's Internet link in Dec. 1997, the same figure was 37%.)

(In Mar. 1998, median FTP data transfer at LBNL was 10,900 bytes.)


Growth of LBNL's WWW Traffic
Connections/Day
10^3
10^2
10^1
10^0
1991
1992
1993
1994
Year

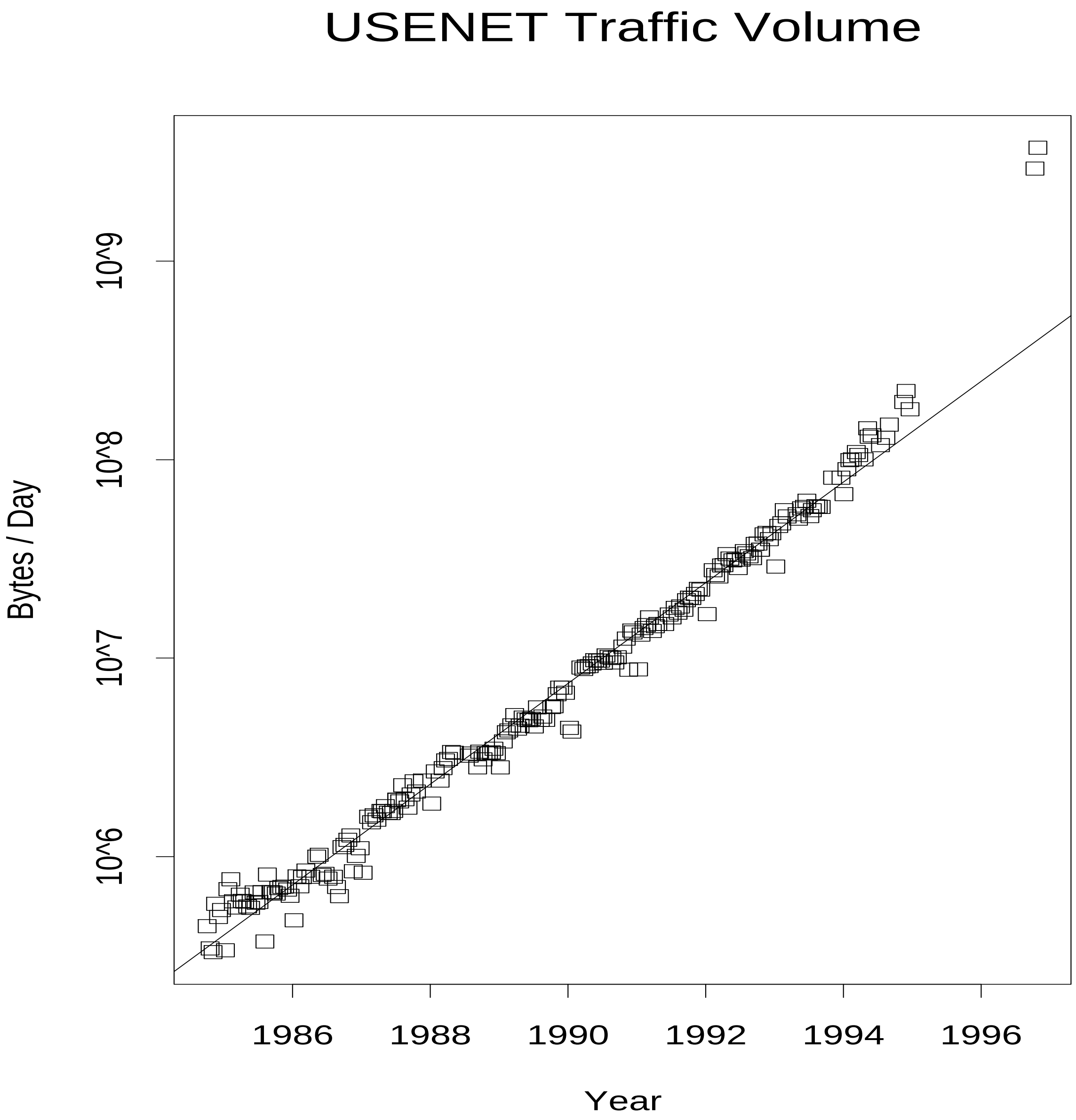
New applications radically alter the landscape:

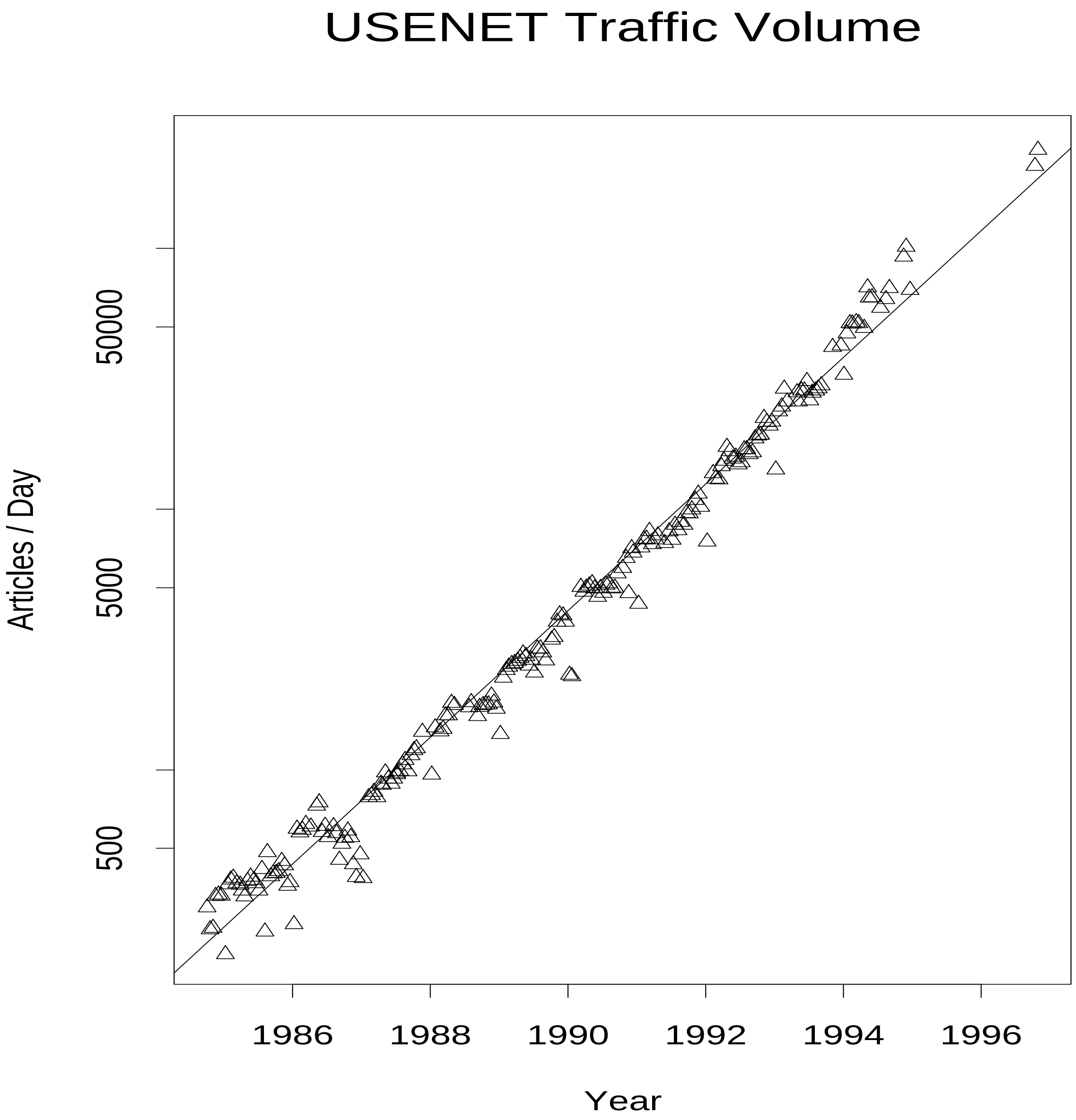
In the *month* of October, 1992, LBNL made *45* World-Wide Web connections. 6 years later, it makes about *600,000* each *day*.

In January, 1992, the “MBone” multicast backbone didn't exist. In January, 1995, it accounted for 20% of the traffic at DEC-WRL, 40% at LBNL, and more than 50% at Bellcore.

(In December, 1997, it was about 12% at LBNL.)

What else is coming? Pricing. *Differentiated services.*


USENET Traffic Volume
Bytes / Day
10^9
10^8
10^7
10^6
1986
1988
1990
1992
1994
1996
Year


USENET Traffic Volume
Articles / Day
500
5000
50000
1986
1988
1990
1992
1994
1996
Year

Pawlita's survey [ITC12, 1989]:

In the two decades from 1966 to 1987, several thousand papers were published on queueing.

Number published on traffic measurements: 44.
Fraction of these that say something interesting: maybe 50%.

$\Rightarrow$ Measurement is hard.
$\Rightarrow$ No one knows what's going on.

(It's gotten better.)

How to measure?

<u>Active</u> measurement: write a program to use the network, analyze its performance.

<u>Passive</u> measurement: watch what existing applications do.

by instrumenting them

by monitoring the network

$\Rightarrow$ These can be combined.

Packet filters:

Idea: watch a network link. Extract those packets matching a particular pattern.

Tool of choice: `tcpdump` [JLM89] (also `libpcap` [MLJ94]).

Key concerns: not dropping packets of interest; getting accurate timestamps.

Avoiding packet drops:

1. Use a *powerful* kernel packet filter, if at all possible.
   - ideal is BPF (Berkeley Packet Filter) [MJ93]
   - 4.4BSD, DEC OSF/1 uses this
2. Capture only packet headers, if at all possible.
3. Run on a lightly-loaded machine. (Problematic for point-to-point links.)
4. Use as large kernel buffers as possible.

Avoiding packet drops, con't:

5. Use the packet filter creatively:

    `tcpdump 'tcp[13] & 0x7 != 0'`

    captures all SYN/FIN/RST packets.

    From these you can determine: connection start time, duration, participating hosts, application protocol, bytes transferred in each direction (cf. `tcp-reduce` [Pa95]).

Avoiding packet drops, con't:

6. Sampling.

   Record 1 in n packets. Or, record each packet with probability p (geometric sampling). [CPB93b]

   BPF unfortunately doesn't support this (yet).

   Only certain measurement problems fit with sampling.

Avoiding common measurement mistakes:

Quantify losses:

How many packets were dropped by the interface? By the kernel? How do you know?

ask `libpcap` (sometimes)

generate calibration traffic

check for internal consistency

(In the face of small loss, think about repeating until you get <u>none</u>.)

Avoiding common measurement mistakes, con't:

Investigate spikes and outliers:

Computers are great at doing the same thing over and over, or to excess.

Whenever something occurs in large quantities, figure out why.

Avoiding common measurement mistakes, con't:

E.g. LBL `finger` traffic went from 270/day $\Rightarrow$ 4,865/day in 5 months.

E.g. `ftp` traffic with a strong periodic component (weather-maps).

E.g. 1,988 `telnet` connections with 180.436 ± 0.002 sec. interarrivals.

E.g. Rapidly growing "other" traffic on port 80.

E.g. Web connection of size 4,294,967,294 bytes.

Avoiding common measurement mistakes, con't:

Split out different modes:

Network traffic is a mish-mash of different stuff. When you see different types of behavior, figure out why. First place to look: application protocol (e.g., Telnet vs. WWW).

Make reality checks:

Build a mental model of what you expect to find. When you find something else, figure out why. It means either your measurements are broken; your data reduction is broken; or you've just learned something interesting.

Avoiding common measurement mistakes, con't:

Calibrate:

If you have another way to measure something, compare what you get.

My favorite example: PSC, the Alphas, the 150 Mbps FDDI, and the stopwatch.

Avoiding common measurement mistakes, con't:

Make self-consistency checks:

If there's something that "must" happen, <u>check to see if it did</u>.

E.g., TCP connections require a three-way packet exchange before they send data. If you don't see it, you probably dropped some packets!* ("Self calibration")

(* How could it be anything other than dropping packets?)

Avoiding common measurement mistakes, con't:

Cross-check:

If you've constructed a description of some network traffic, create synthesized traffic from it and see if it matches.

Make predictions:

When you've figured out what's going on, take new data and see if you're right. Alternatively: use <u>subsampling.</u>

What we know about the Internet — the USC/UCB/Bellcore study [DJCME92]:

What they measured: all packet headers on the DMZ networks connecting USC, UCB, and Bellcore to the Internet.

When: 1991 / 1989 / 1989.

Duration: 1 / 1 / 3 days.

Drops: 0.6% / 0 / 0.

Volume: 5.2 Mpkt / 5.9 Mpkt / 1.7 Mpkt.

The USC/UCB/Bellcore study - what they found:

Great variation in traffic mix between the sites.

Up to 90% of bulk transfers send $<$ 10 KB, request/response.

50% of interactive connections last $<$ 90 s. The responder can generate 10 times more data than the originator.

Telnet packet interarrivals have a fixed distribution.

Data used to construct *empirical* models (Tcplib).

Empirically-derived analytic models [Pa94a]:

Measured: seven months of LBNL wide-area SYN/FIN data, 1990-1993 (2.5 million connections).

Plus donated by others: a single day from UCB, USC, coNCert, UK–US; 3 days from DEC-WRL; 13 days from Bellcore (350 thousand connections). 1991, 1989.

How to generalize 1 GB of trace data into useful models?

Empirical vs. analytic models:

	Empirical	Analytic
Basic model	raw, measured distribution	analytic function fit to distribution.
Form	$O(1000)$ numbers	$F(x;\theta)$
Example	Tcplib [DJ91]	Log-normal
Pros	Exactly represents an aspect of the Real World. Preserves quirks (if present).	Mathematically tractable. Concise. Easy to communicate. Easy to compare.
Cons	No good if atypical.	No good if oversimplifies.

Summary of models:

Proto.	Variable	Model	Invariant?	Corr.
telnet	orig. bytes	$\log$-extreme	N	$+$
	resp. bytes	lgnorm, 80-100%	Y	$+$
	duration secs.	lgnorm	Y	$+$
nntp	orig. bytes	lgnorm	N	$++$
smtp	orig. bytes	$2\times$ lgnorm + 300B	N, N	$++$
ftp	conn. bytes	lgnorm	N	$++$
	session bytes	lgnorm	Y	no
	burst bytes	Pareto, 95-100%	N	$++, \pm$

If "invariant" then fixed parameters for the distribution are adequate.

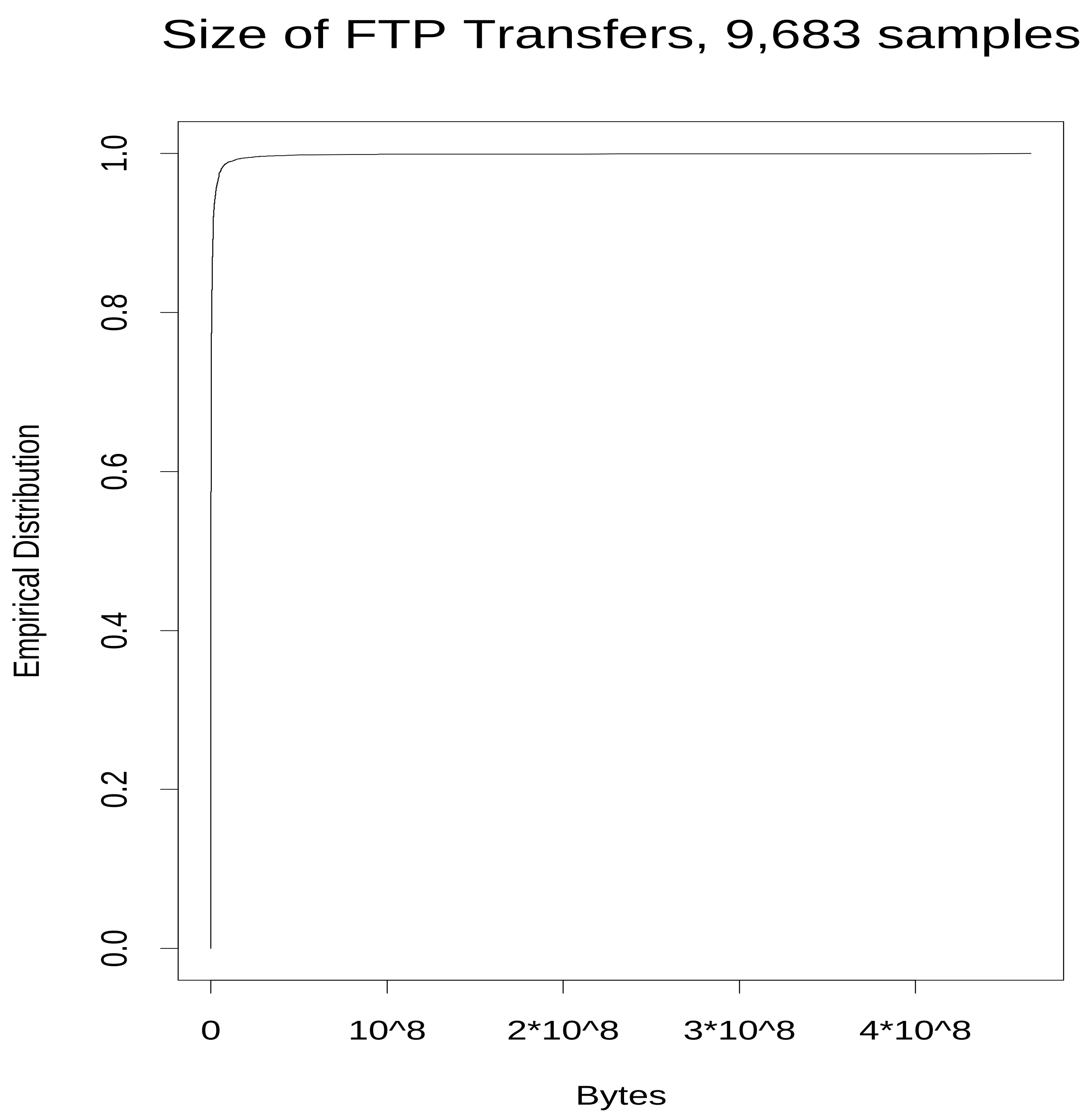
$+$ = weak positive correlation (mean lag-1 ACF ≤ 0.1).
$++$ = strong positive correlation (mean lag-1 ACF > 0.1).
$\pm$ = weak undirected correlation (sometimes > 0, sometimes < 0).

Aside — Transformations:

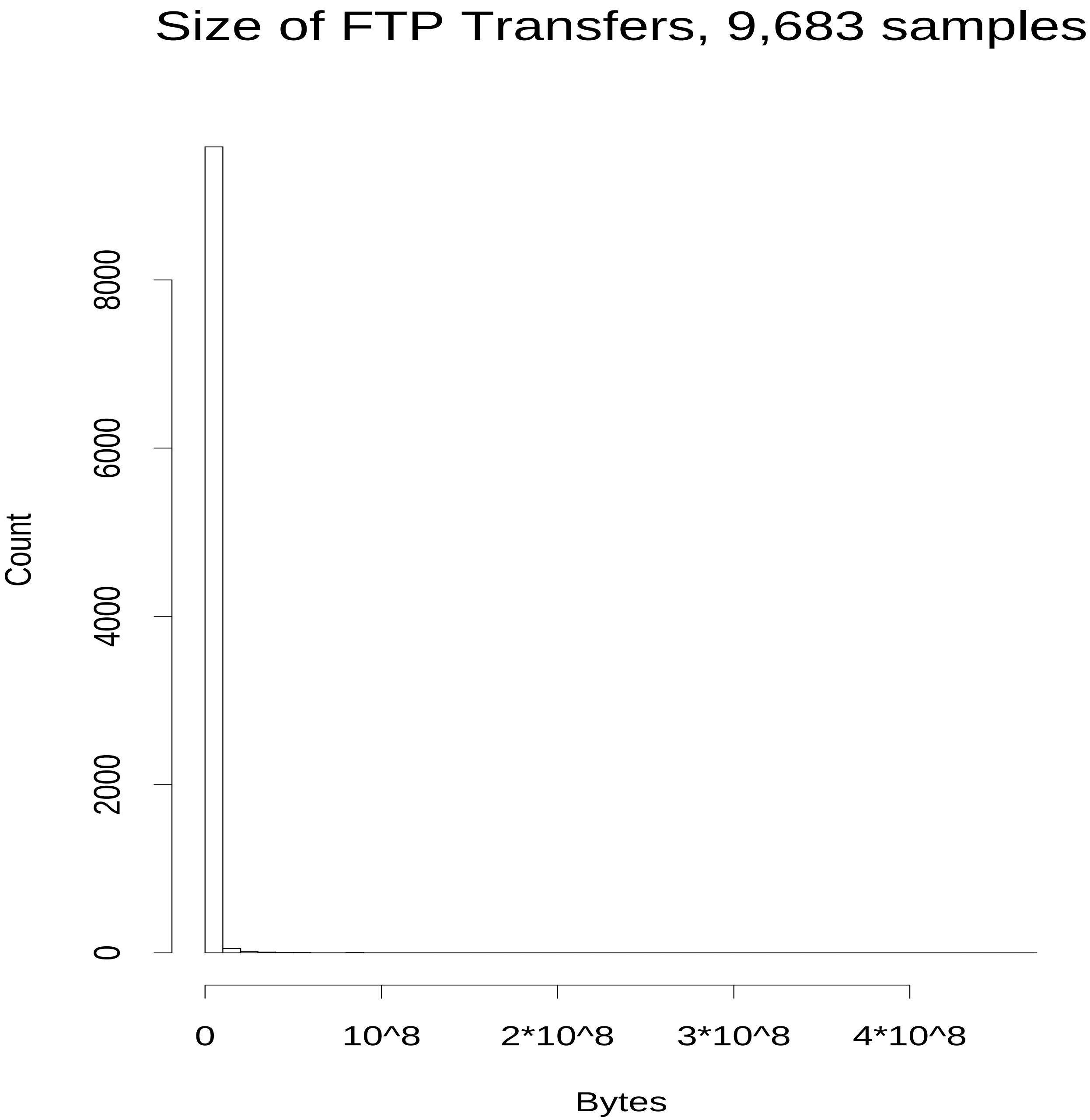
In general, for a random variable X you can define a new random variable Y as a function of X: $Y = G(X)$. The distribution of Y can then be computed in terms of the distribution of X, or (sometimes) vice versa.

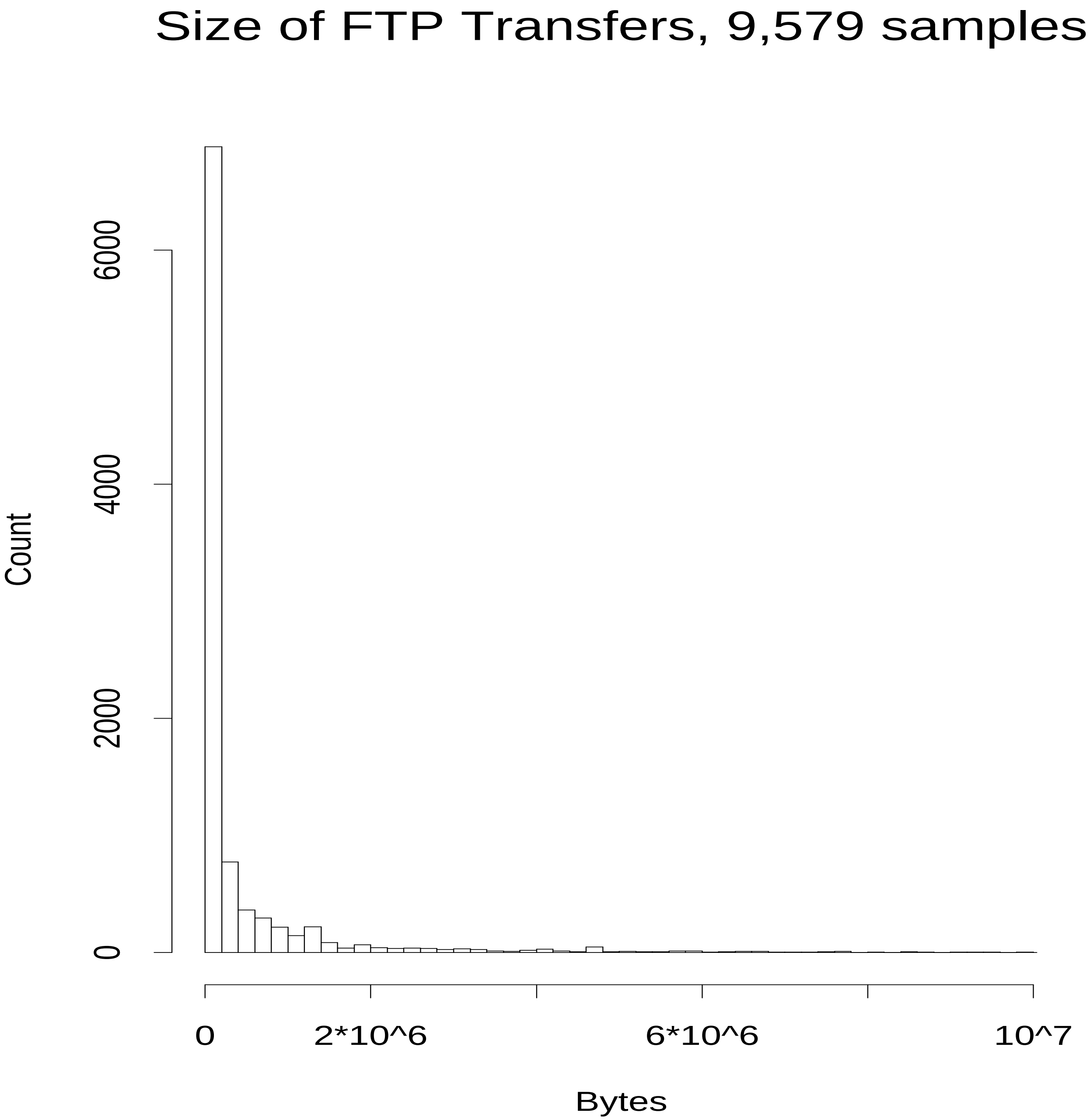
For networking, the transformation to always consider is the logarithmic one: $Y = \log X$ (base doesn't matter).

Why? Because network behavior occurs on many different scales.

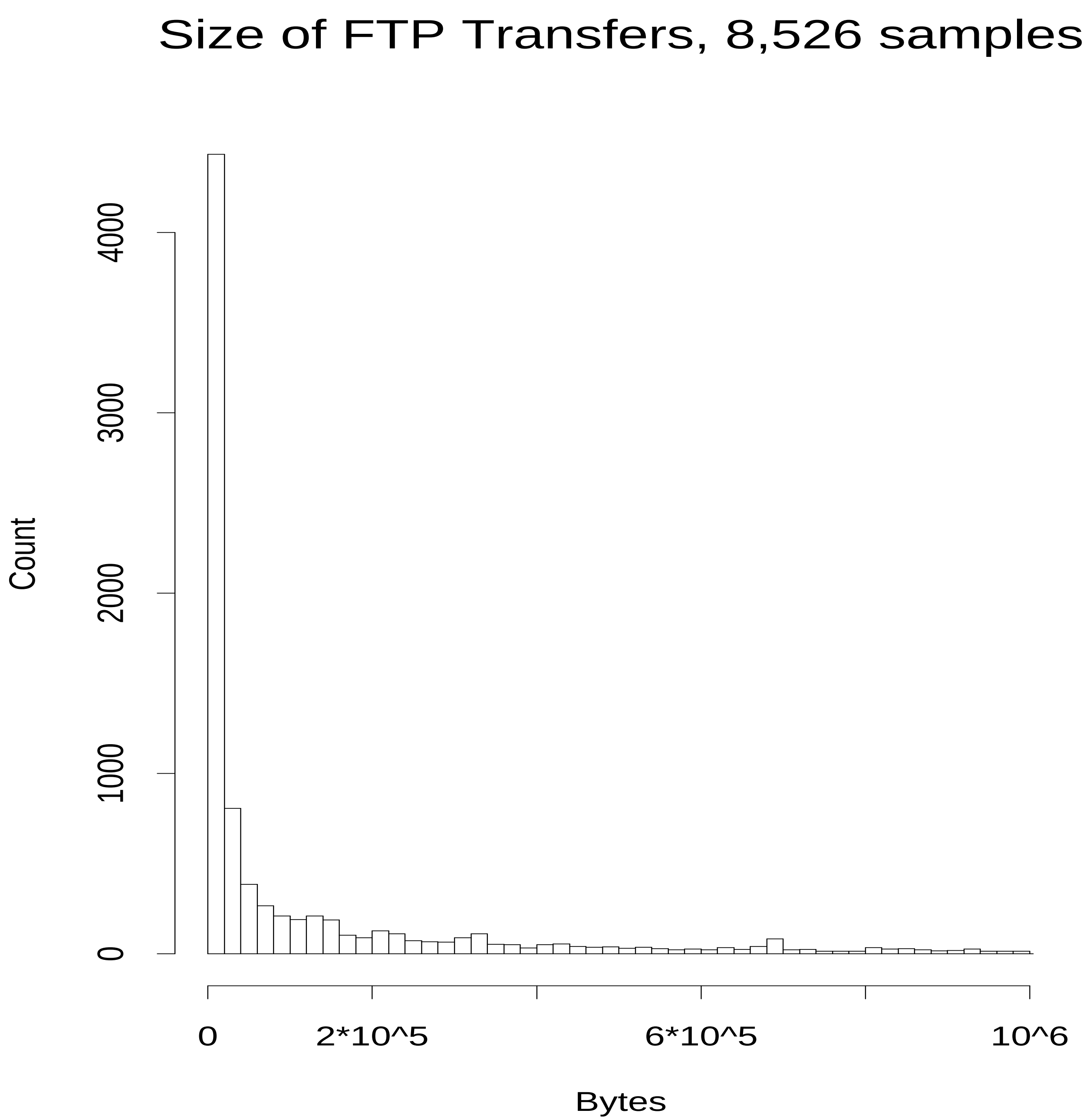


Size of FTP Transfers, 9,683 samples
Empirical Distribution
0.0
0.2
0.4
0.6
0.8
1.0
0
10^8
2*10^8
3*10^8
4*10^8
Bytes

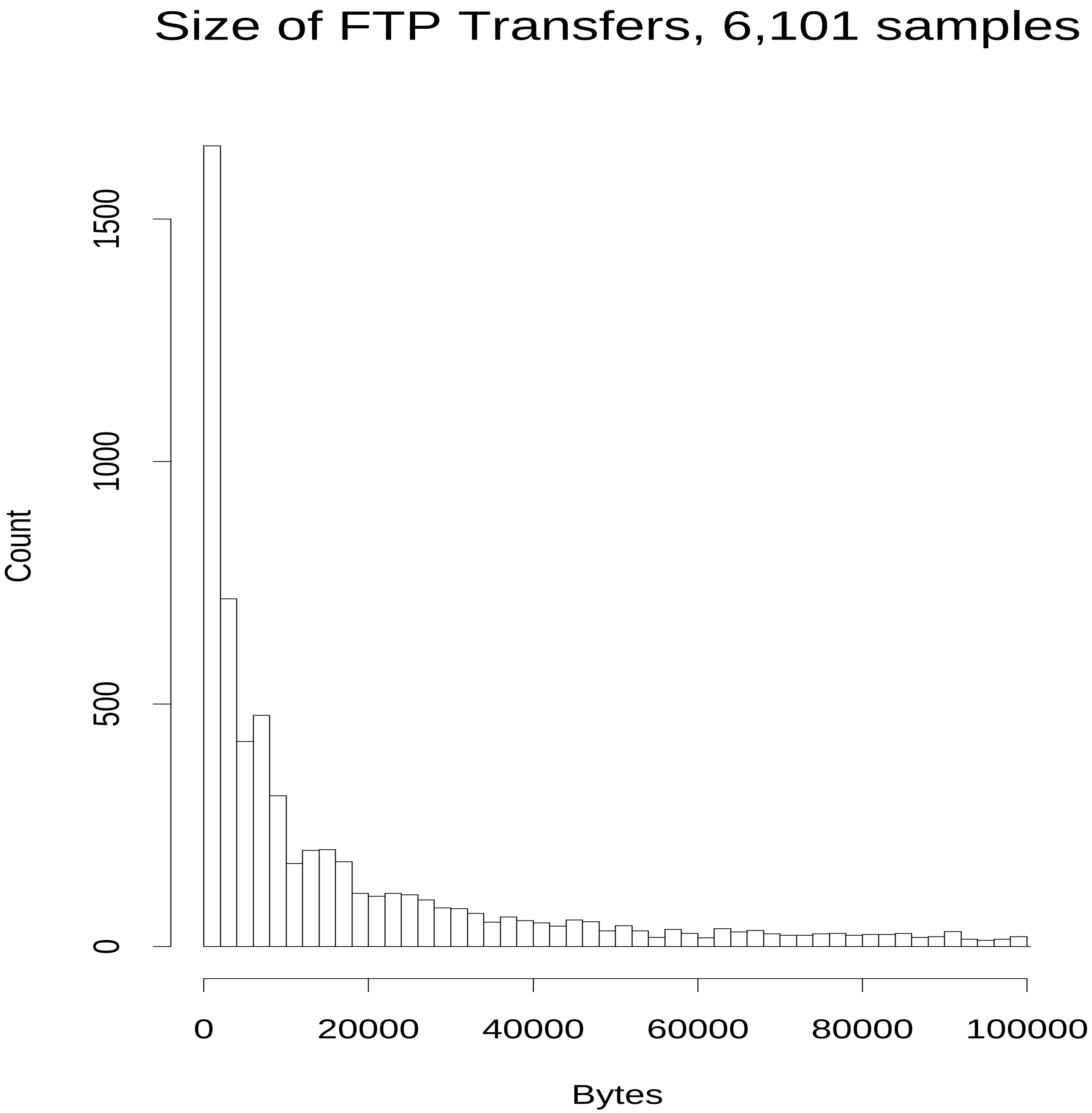

Size of FTP Transfers, 9,683 samples
Count
0
2000
4000
6000
8000
0
10^8
2*10^8
3*10^8
4*10^8
Bytes

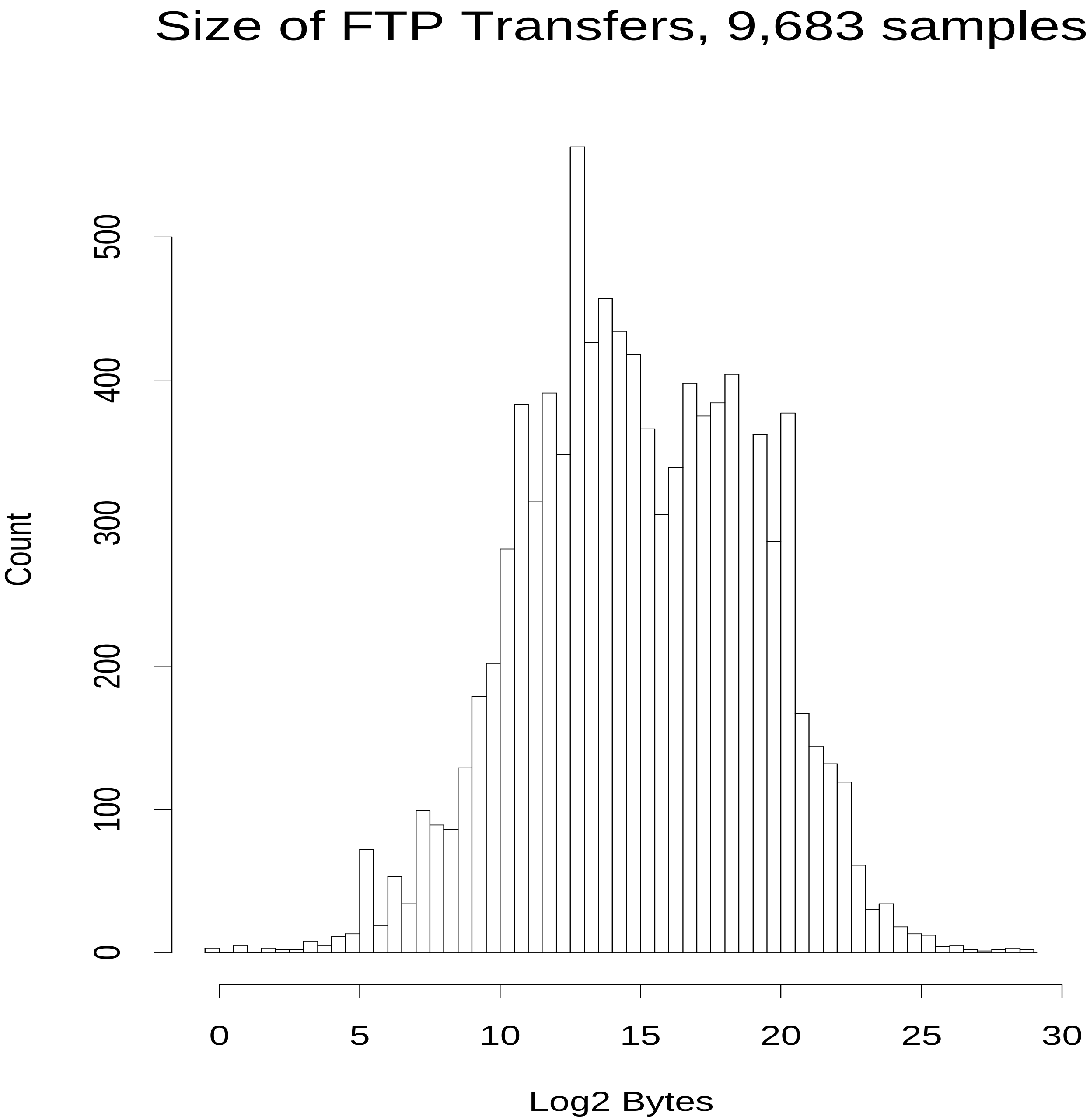


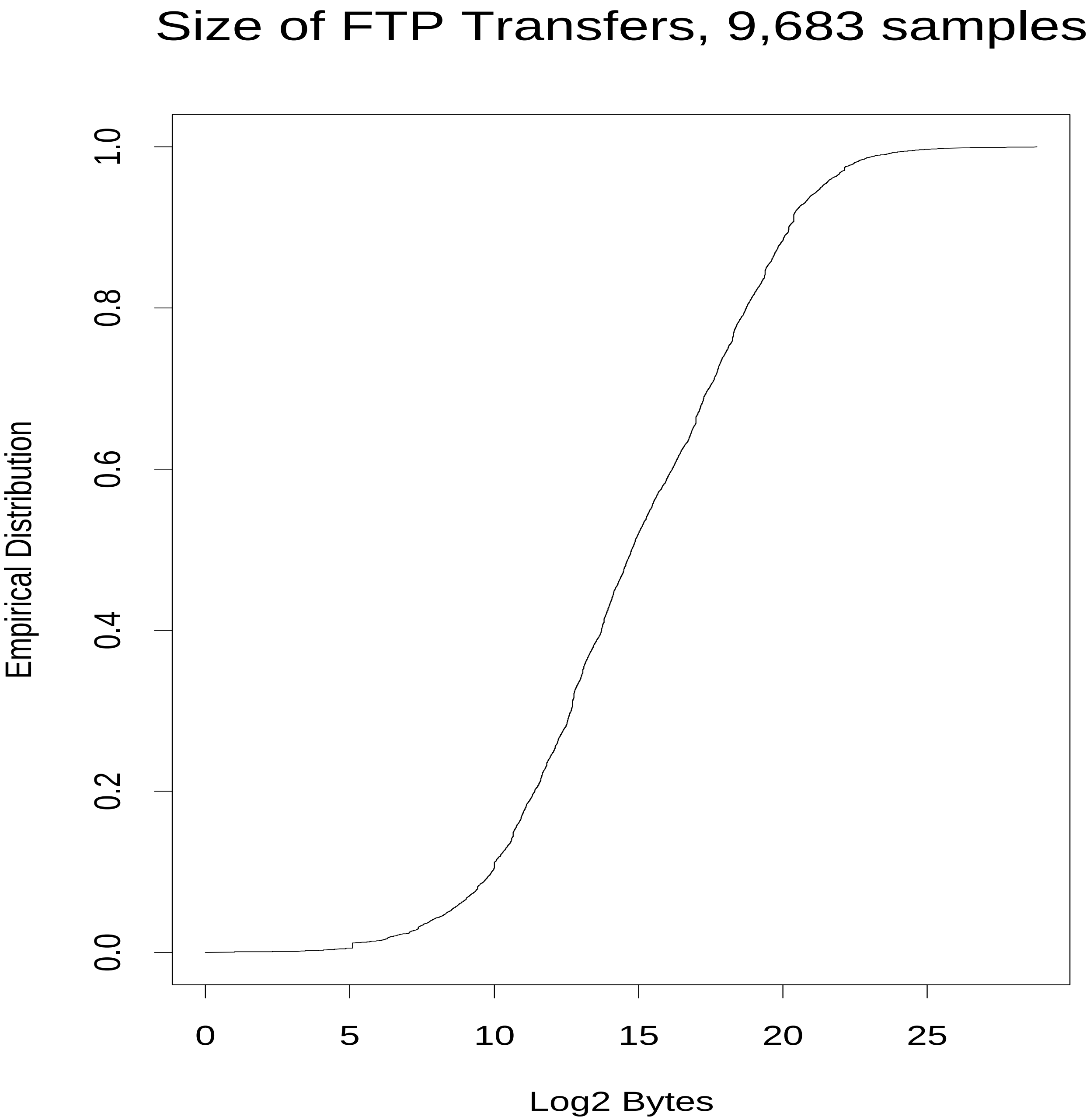
Size of FTP Transfers, 9,579 samples
Count
0
2000
4000
6000
0
2*10^6
6*10^6
10^7
Bytes



Size of FTP Transfers, 8,526 samples
Count
0
1000
2000
3000
4000
0
2*10^5
6*10^5
10^6
Bytes


Size of FTP Transfers, 6,101 samples
Count
0
500
1000
1500
0
20000
40000
60000
80000
100000
Bytes


Size of FTP Transfers, 9,683 samples
Count
0
100
200
300
400
500
0
5
10
15
20
25
30
Log2 Bytes



Size of FTP Transfers, 9,683 samples
Empirical Distribution
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
Log2 Bytes

Transformations, con't:

If a logarithmic transformation of X is useful, then summarizing the mean of X is not:

mean size of all FTP transfers: 913 KB.

mean of FTP transfers $< 10^7$ bytes: 402 KB.

mean of FTP transfers $< 10^6$ bytes: 118 KB.

mean of FTP transfers $< 10^5$ bytes: 17 KB.

The mean is not robust.

In general, summarizing the median and the tails is often much more meaningful than the mean and variance.

Transformations, con't:

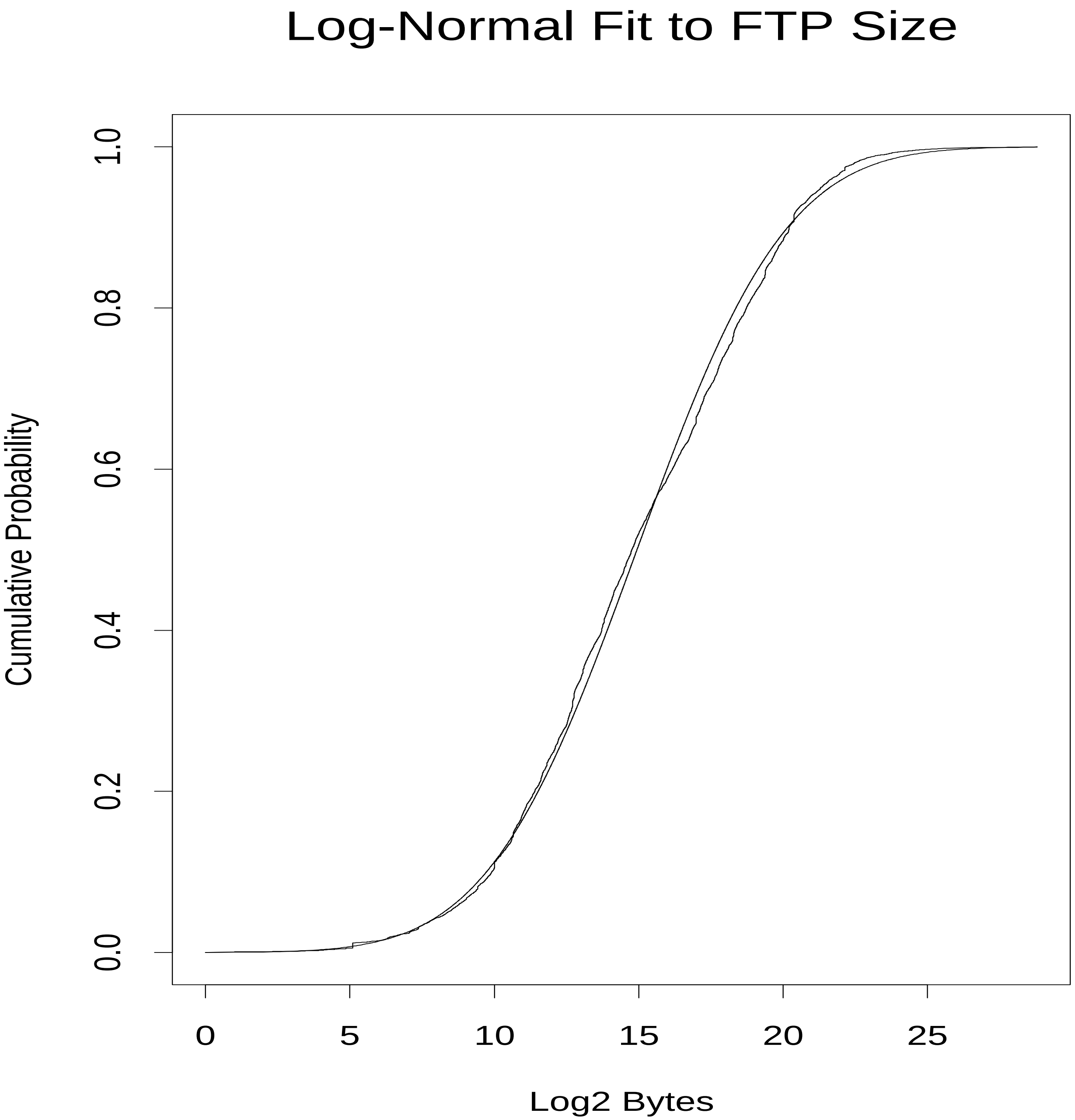
Summarizing the mean and variance of $\log X$ can make sense. First gives the geometric mean. Second doesn't have a name, but relates to a "standard deviation" factor.

For dataset of FTP transfers:

$$2^{\overline{\log_2 X}} = 31 \text{ KB.}$$

$$2^{\sqrt{\text{Var}[\log_2 X]}} = \times 16.8.$$

Median transfer size: 28 KB.



Log-Normal Fit to FTP Size
Cumulative Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
Log2 Bytes

Logarithmic transformations, con't:

Pareto distribution: $F(x) = 1 - (a/x)^{\beta}, \quad a, \beta \geq 0, \;\; x \geq a$.

If $\beta \leq 2$, distribution has infinite variance (a heavy tail).

If $\beta \leq 1$, distribution also has infinite mean.

$\Rightarrow$ If X has a Pareto distribution, then $\log(X/a)$ has an exponential distribution with $\lambda = \beta$.

When thinking about the upper tail of a distribution, think Pareto.

Findings of analytic models study:

1. Analytic models describe traffic as well as empirical ones.

2. Both types of models poorly describe the upper 1% tails.

3. Traffic varies significantly, both over time and site-to-site.

4. Use log-normal distributions for size of bulk-transfer traffic.

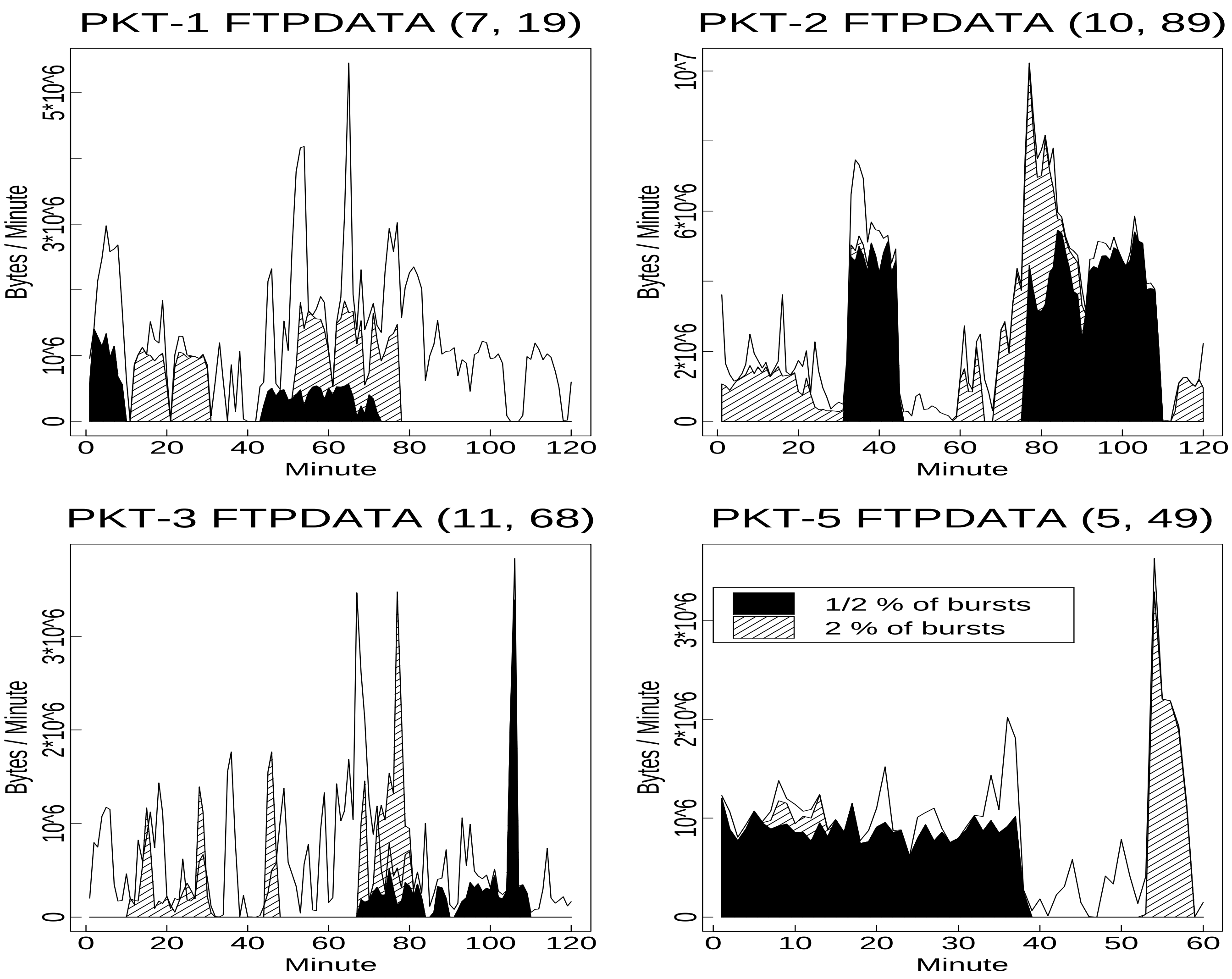
Findings, con't:

5. The computer side of a TELNET connection sends 20 times as many bytes as the user side.

6. Half of all* FTP *sessions* transfer more than 32 KB; 1/6 transfer more than 500 KB.

7. The upper tail of FTP "bursts" is so large that 2% of the bursts account for 50-80% of <u>all</u> the FTP data bytes.

Dominance of largest FTP bursts:



Web servers:

Arlitt et al examined logs from 6 Web servers [AW96], spanning 1 week to 1 year.

About 80–90% of HTML requests retrieve something. Remainder either “not modified” (4–13%), redirected (0.3–4.2%), or failed (0.6–3.9%).

Less than 3% of the requests were for distinct items $\Rightarrow$ server caching a big win.

Web servers, con't:

About $\frac{1}{3}$ of the files and bytes are accessed only once (regardless of the amount of time spent observing) $\Rightarrow$ server caching a bit tricky.

References to individual items are well-modeled as Poisson.

Internet FTP traffic:

Danzig et al traced 8 days of FTP traffic between Boulder and NSFNET [DHS93]. $4.79 \cdot 10^8$ IP packets ($\approx$ 6% of NSFNET bytes), 35% FTP. Drops: 0.32%.

134,000 transfers.
43% of conn. did nothing!(?) 8% just list directory.

30% of bytes uncompressed.
Possible savings: 6% of NSFNET bytes.
Developed cache design to save another 21% of bytes.

Domain Name System:

Danzig et al: two 24-hour traces from a DNS root name server [DOK92], captured in 1991.

Identified seven types of DNS errors. Worst: looping, two servers asking each other the same thing over and over.

Adding negative caching: a good "safety valve" (500 CMU machines queried `THISHOST`!), but fixing bugs more important.

8% of NSFNET packets were DNS. Deploying already-existing DNS algorithms would reduce this by a factor of <u>twenty</u>.

BGP Routing:

Labovitz and colleagues analyzed BGP routing messages seen at public exchange points [LMJ97]:

- 99% of routing information pathological
- impact believed small; now believed fixed
- routing traffic still has strong periodicities [FJ94]
- most routes quite stable
- diverse routing instability sources
  - instability correlated with load

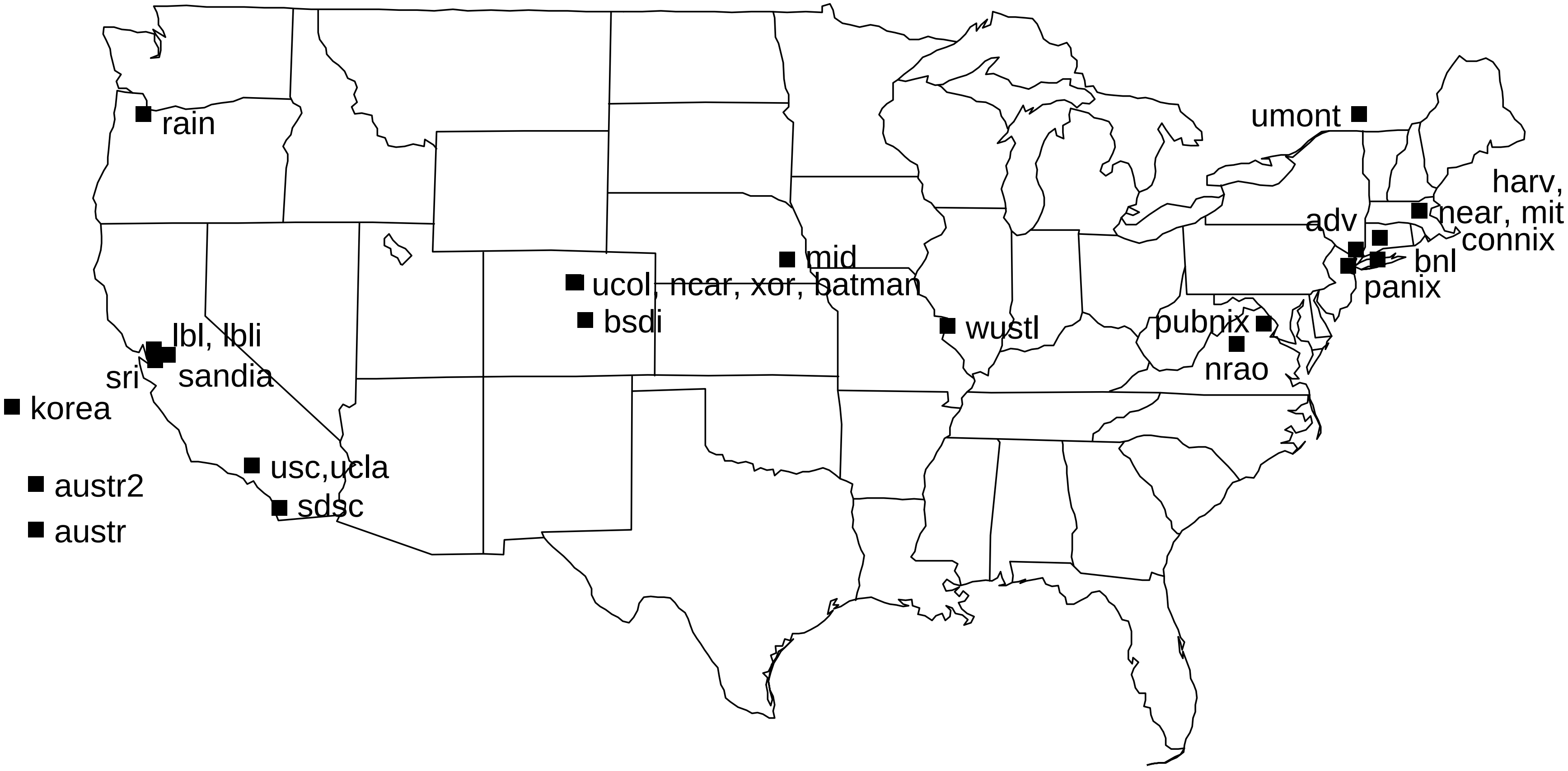
A framework for conducting end-to-end measurements:

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.

Participating sites — North America



Links traversed — North America

End-to-end Routing:

Paxson [Pa97a] analyzed 40,000 repeated “traceroute”s between 37 Internet sites. Looked at pathologies, routing stability, routing symmetry. (Data available from the Internet Traffic Archive [ITA].)

Pathologies: the likelihood of encountering a significant problem rose from 1.5% at the end of 1994 to 3.3% at the end of 1995. (We'll look more at these in a minute.)

Routing, con't:

Stability: most “virtual paths” through the Internet are heavily dominated by a single route. 2/3's of the route remain unchanged over periods of days to weeks. The other 1/3 change on time scales from seconds to hours.

Symmetry: The likelihood at the end of 1995 that a route from A to B visited at least one different city than that from B to A was about 50%.

End-to-end packet dynamics:

Paxson [Pa97b] analyzed 20,000 TCP bulk transfers, 100 KB each, between 36 Internet sites.

Each connection traced at <u>both ends</u> (sender and receiver).

Measurements made at <u>Poisson</u> intervals $\Rightarrow$ unbiased.*

Two datasets, $\mathcal{N}_1$ (Dec 1994) and $\mathcal{N}_2$ (Nov–Dec 1995).

End-to-end packet dynamics, con't:

Analyzed: network pathologies, bottleneck bandwidth, TCP retransmission, patterns of packet loss, timescales of queueing, available bandwidth.

Analysis done with `tcpanaly`, a tool for analyzing TCP and network behavior [Pa97c].

How to probe:

Why TCP?

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

Why not TCP?

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

Network pathologies — out-of-order data delivery:

36% of traces in $\mathcal{N}_1$, 12% of traces in $\mathcal{N}_2$.

2% of packets in $\mathcal{N}_1$, 0.6% in $\mathcal{N}_2$.

But: strongly site-specific.

Worse site: 24% of all packets sent!

Biggest delay: 9 seconds!

Biggest gap: 54 packets!

Impact on performance: generally, <u>minimal</u>.

Network pathologies — rare & unusual:

Packet replication:

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

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

Checksum errors: 1 pkt in 5,000 (after removing outlier).

$\Rightarrow$ bad packets accepted: 1 in 300,000,000.

But: hard to believe this figure.

Need independent confirmation.

(Packet gaining data in flight: twice!)

Packet loss:

Overall packet loss rate: 2.7% $\Rightarrow$ 5.2%.

Loss-free (“quiescent”) connections: 52% vs. 49%.

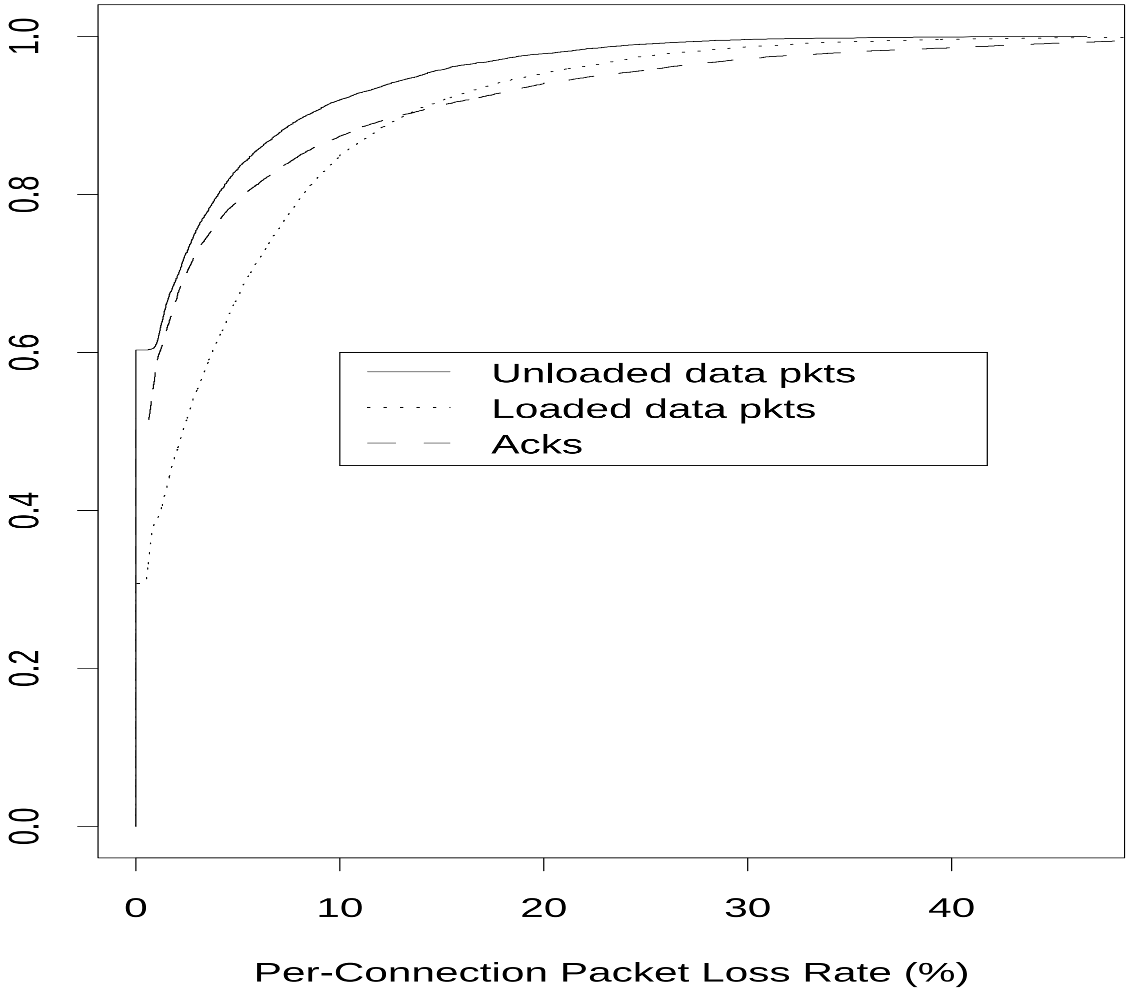
Ack loss-rate if not “quiescent” (“busy”): 5.7% vs. 9.2%.

But: strong regional & time-of-day variations.

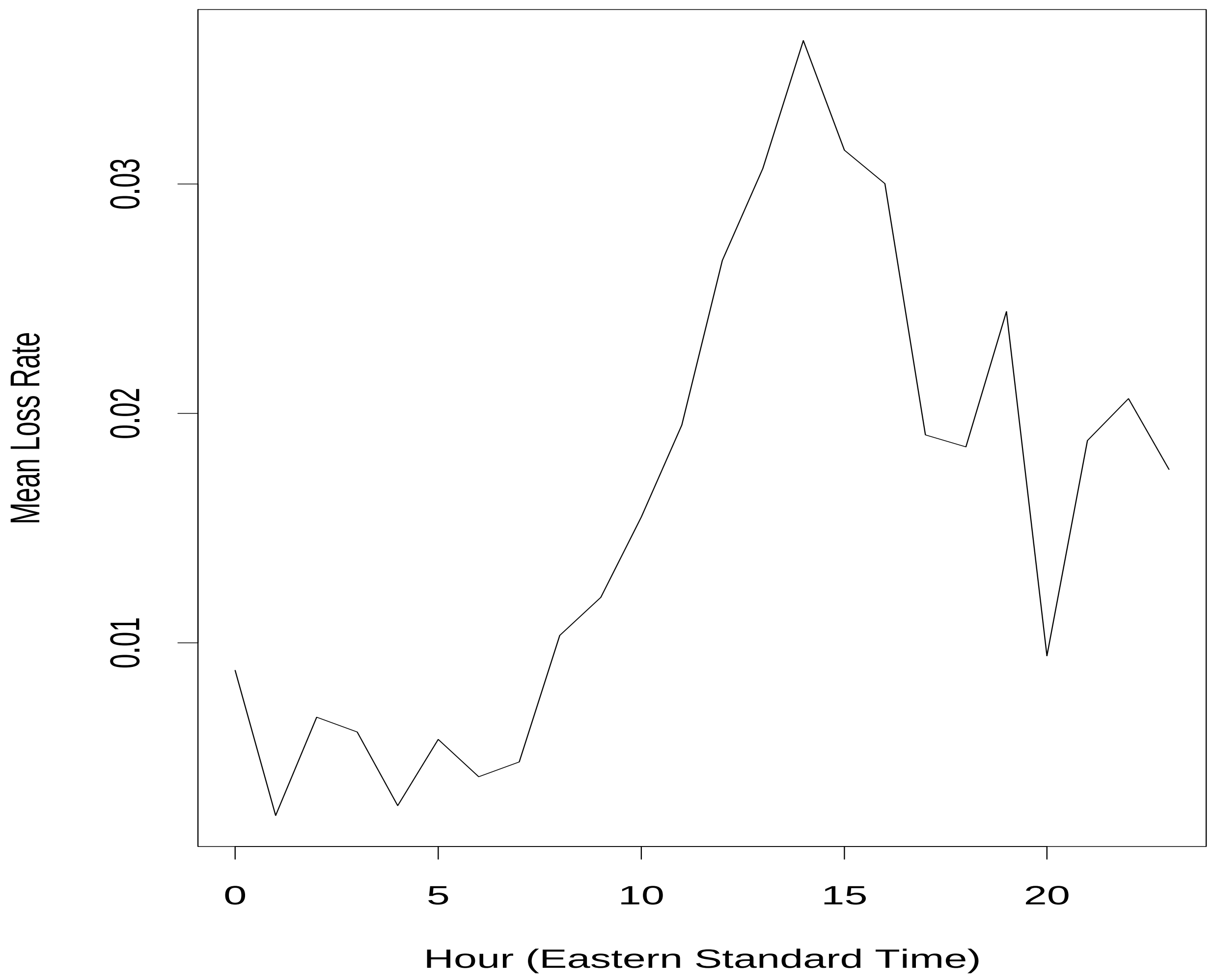
Loss not well described as independent.

Loss outages have infinite variance.

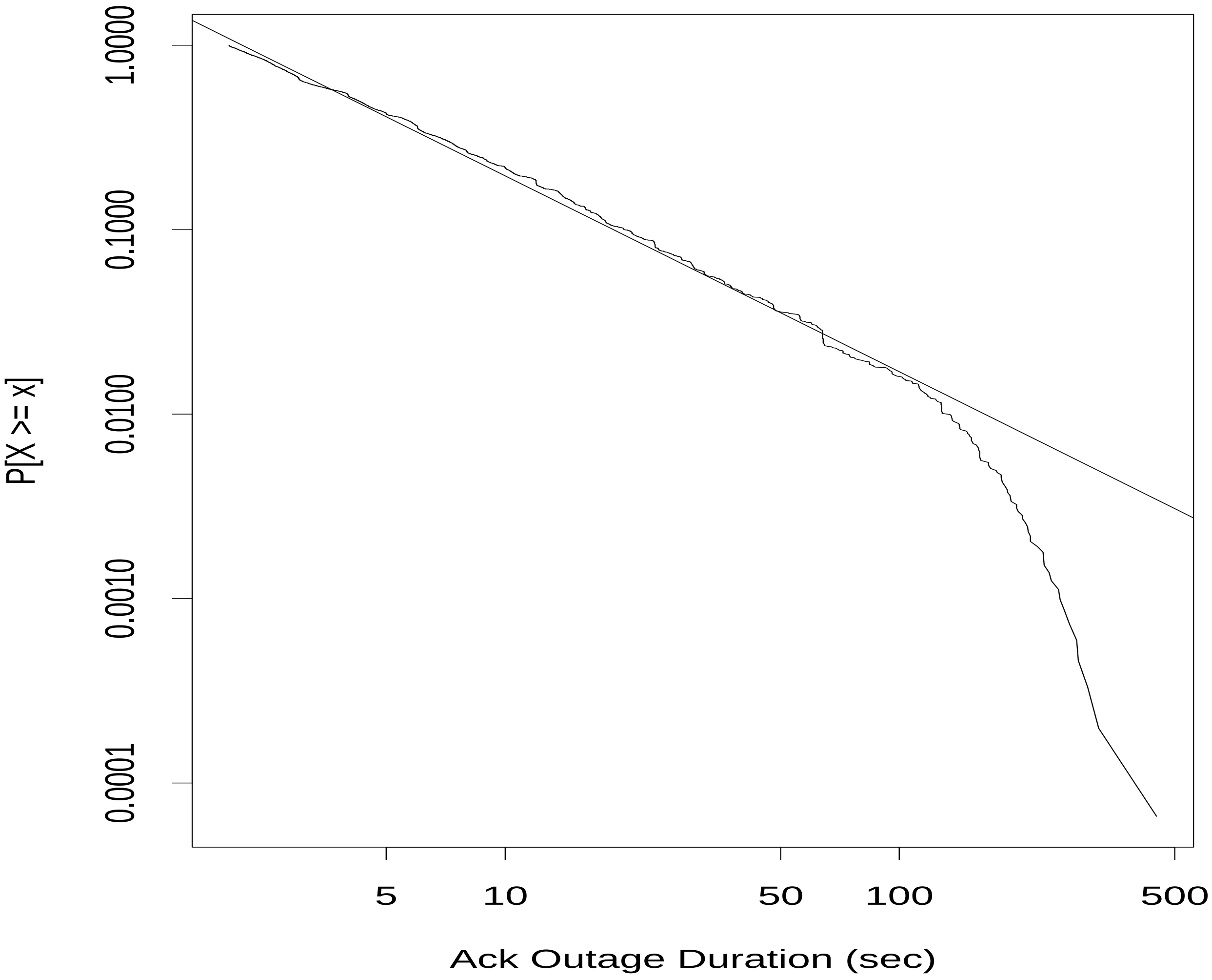
TCP retransmission efficient, except for a few broken TCP implementations.



Loss rates for data packets and acks: wide variation.



Packet loss by time of day, North American connections.



Log-log CDF plot of ack loss outages ($\alpha = 1.06$).

Round trip time (RTT):

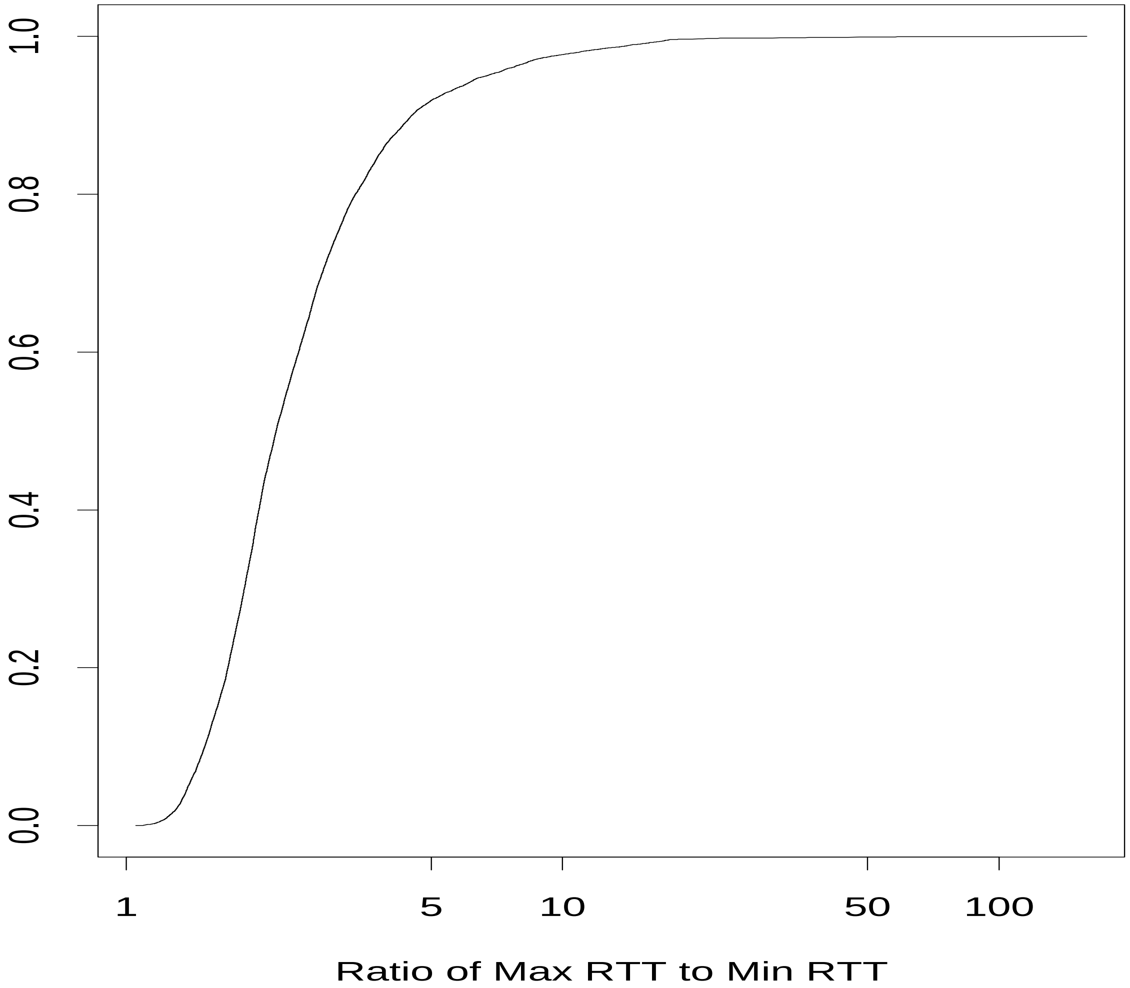
Largest seen: 15.1 seconds.

Largest minimum seen: 4.2 seconds.

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

Sometimes immense peak-to-peak RTT variation.

Further analysis: bottleneck bandwidth, queueing timescales, available bandwidth.



Clearly, sometimes immense peak-to-peak RTT variation.

IPPM:

IPPM: IP Performance Metrics effort within the IETF. Goal: develop standardized Internet measurements, foster market forces for improving performance, and to aid in trouble-shooting [IPPM].

Framework document discusses how to define metrics & methodologies as well as a wide-range of measurement concerns [PAMM98].

Additional draft RFCs define metrics for connectivity [MP98], one-way loss & delay [AKZ98a, AKZ98b].

Traditional network modeling — Poisson traffic:

Idea: everything is independent of everything else. There's just a single **rate** of calls/hour.

$\Rightarrow$ **memoryless** property.

Times between calls ("interarrivals") distributed exponentially: $F(x) = 1 - e^{-\lambda x}$, and *independently*. Exponential is the *memoryless* distribution.

Number of calls ("arrivals") has Poisson distribution:

$$P\{N(t+s) - N(t) = n\} = e^{-\lambda s} \frac{(\lambda s)^n}{n!}.$$

Memoryless property makes analysis easy:

Nothing depends on what happened previously! Or, throw in a queue, and you introduce "short range dependence" — things influence the near-term. $\Rightarrow$ *Analysis is easy.*

There are many variants of Poisson modeling, but all share same basic properties.

Poisson modeling a huge success: telephone system works because of it. $\gg 1,000$ papers published in last 30 years.

If assumptions correct, can analyze intricate systems.

Poisson modeling tells us:

Correlations are fleeting.

Modest buffer increases give large gains in capacity.

Traffic multiplexes extremely well:

$$\mathsf{Var}[X^{(m)}] \sim m^{-1}.$$

It's easy to define a "burst."

Traditional/Poisson assumptions:

Independence among arrivals.

(for sessions, connections, packets)

Exponentially distributed sizes, durations.

Bulk transfers are infinitely large, and unidirectional.

Interactive connections transfer equal amounts of traffic in each direction.

Virtually all of these are wrong.

It took 25 years to figure this out—a testament to the power of analytic principles in shaping our thinking.

Testing for Poisson arrivals:

Packet interarrivals *not* exponentially distributed:

[JR86], [Gu90], [FL91], [DJCME92]

Connection interarrivals *not* exponentially distributed [PF94]:

- except for user *sessions*, when viewed over 1-hour intervals.
- FTP, NNTP, SMTP also show significant correlations.

The failure of Poisson modeling:

- packet arrivals are not Poisson
- most connection arrivals are not Poisson
- connection sizes are log-normal, not exponential
- packet arrivals are correlated
- connection arrivals are correlated

Poisson modeling fails on all fronts. With what do we replace it?

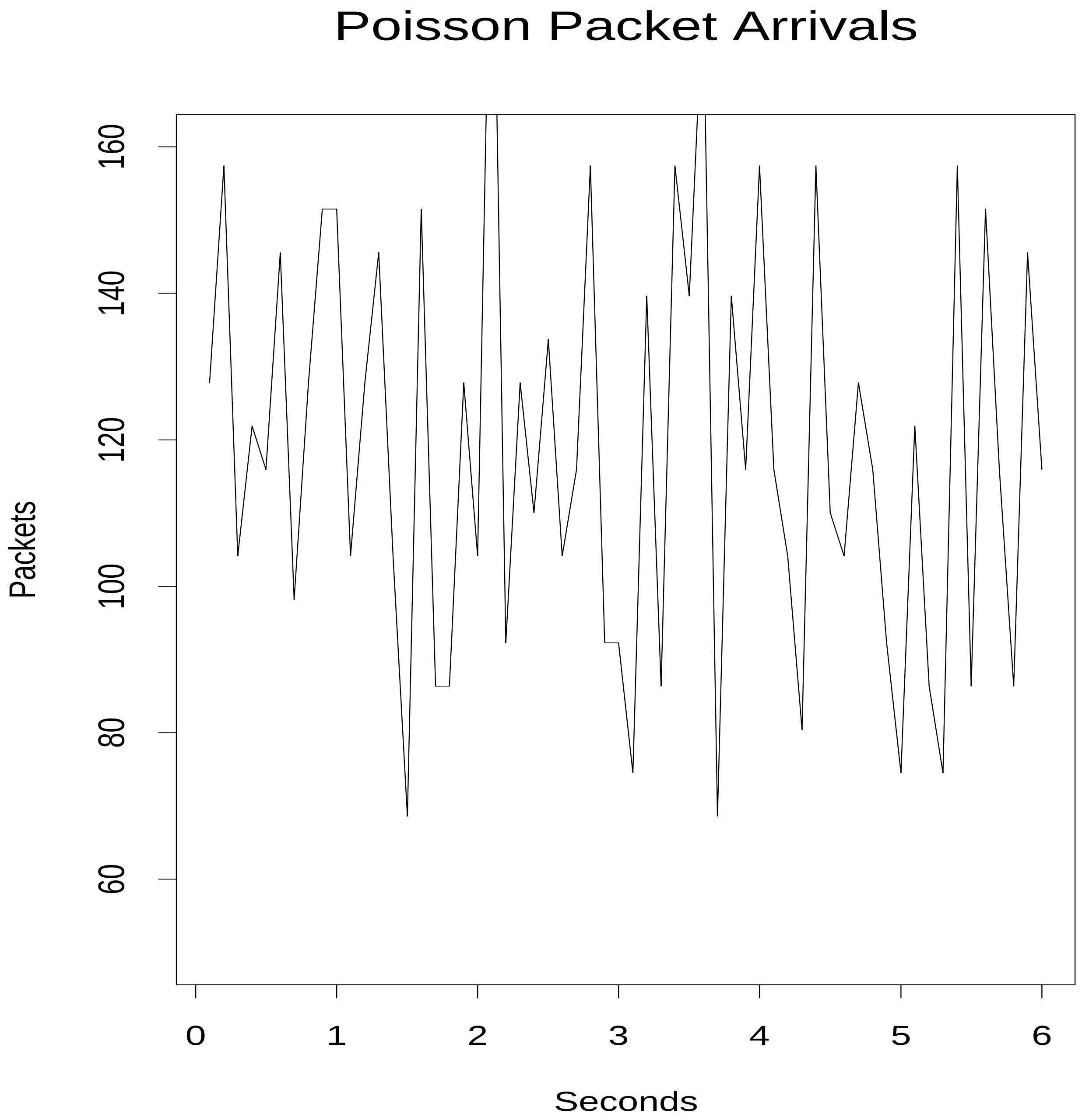
The fractal revolution:

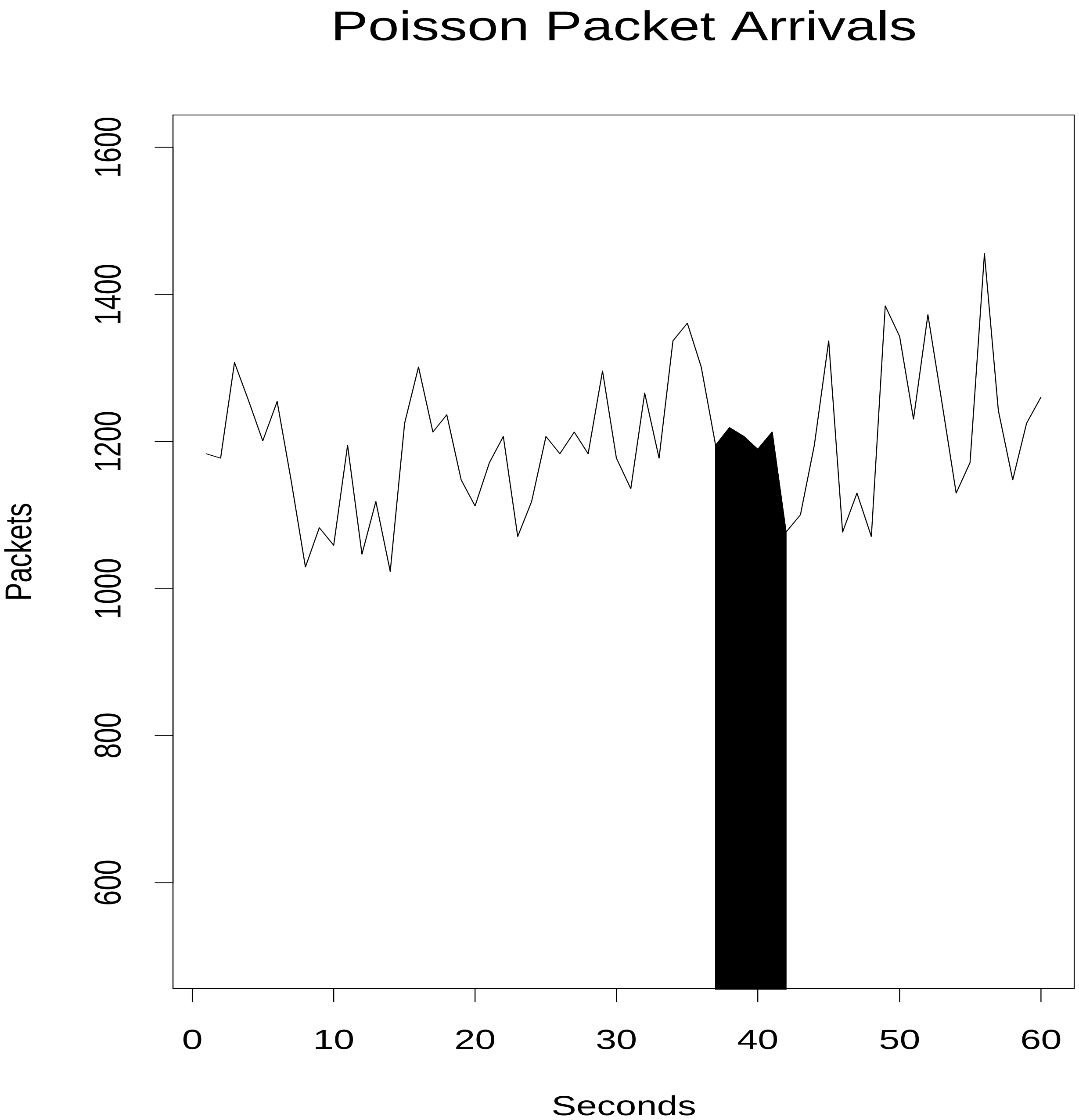
Poisson models have a *fundamental* problem: they grievously underestimate burstiness.

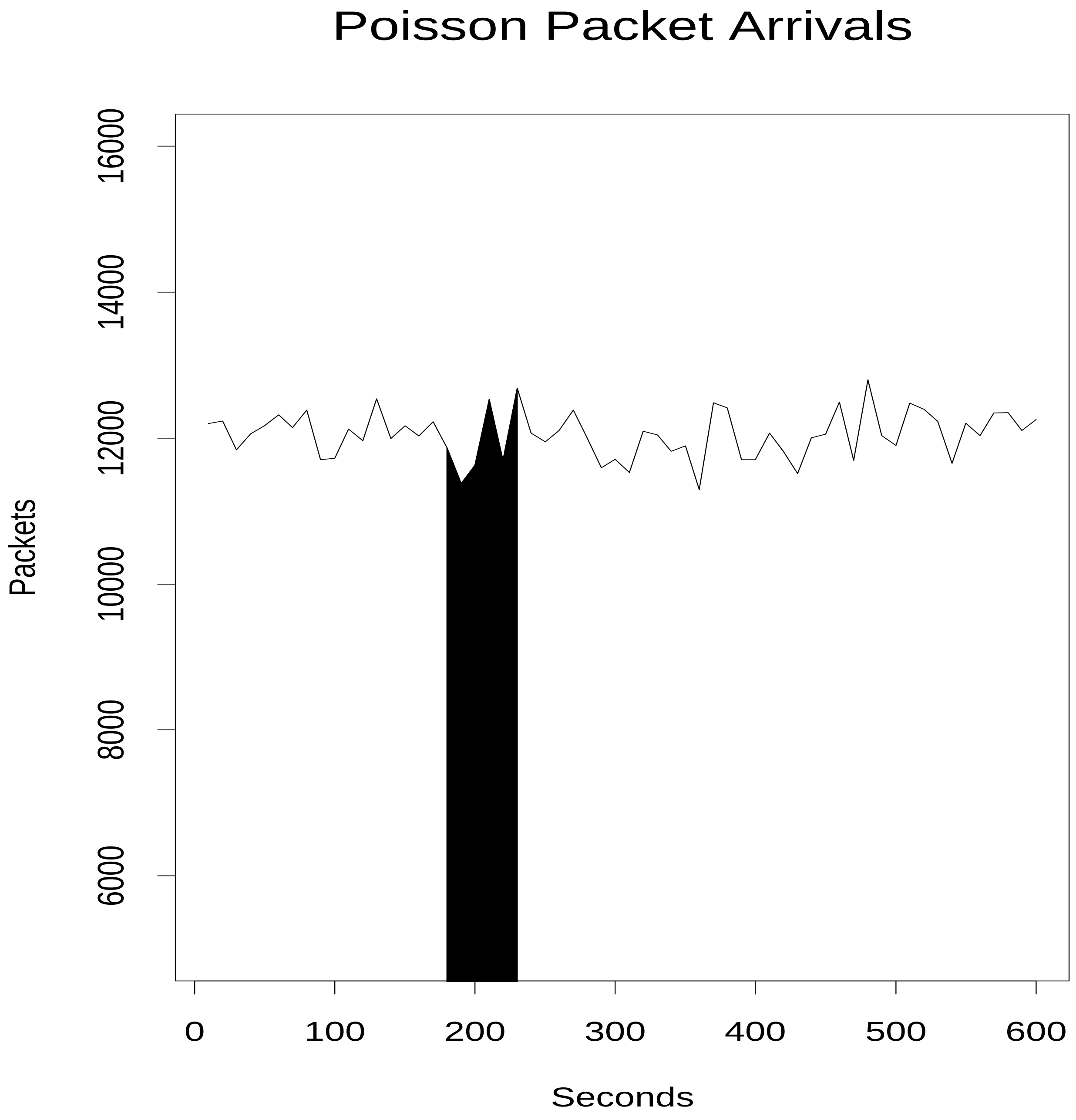
Consider the *arrival process*: X_k gives the number of packets arriving during interval k (for some fixed interval length T).

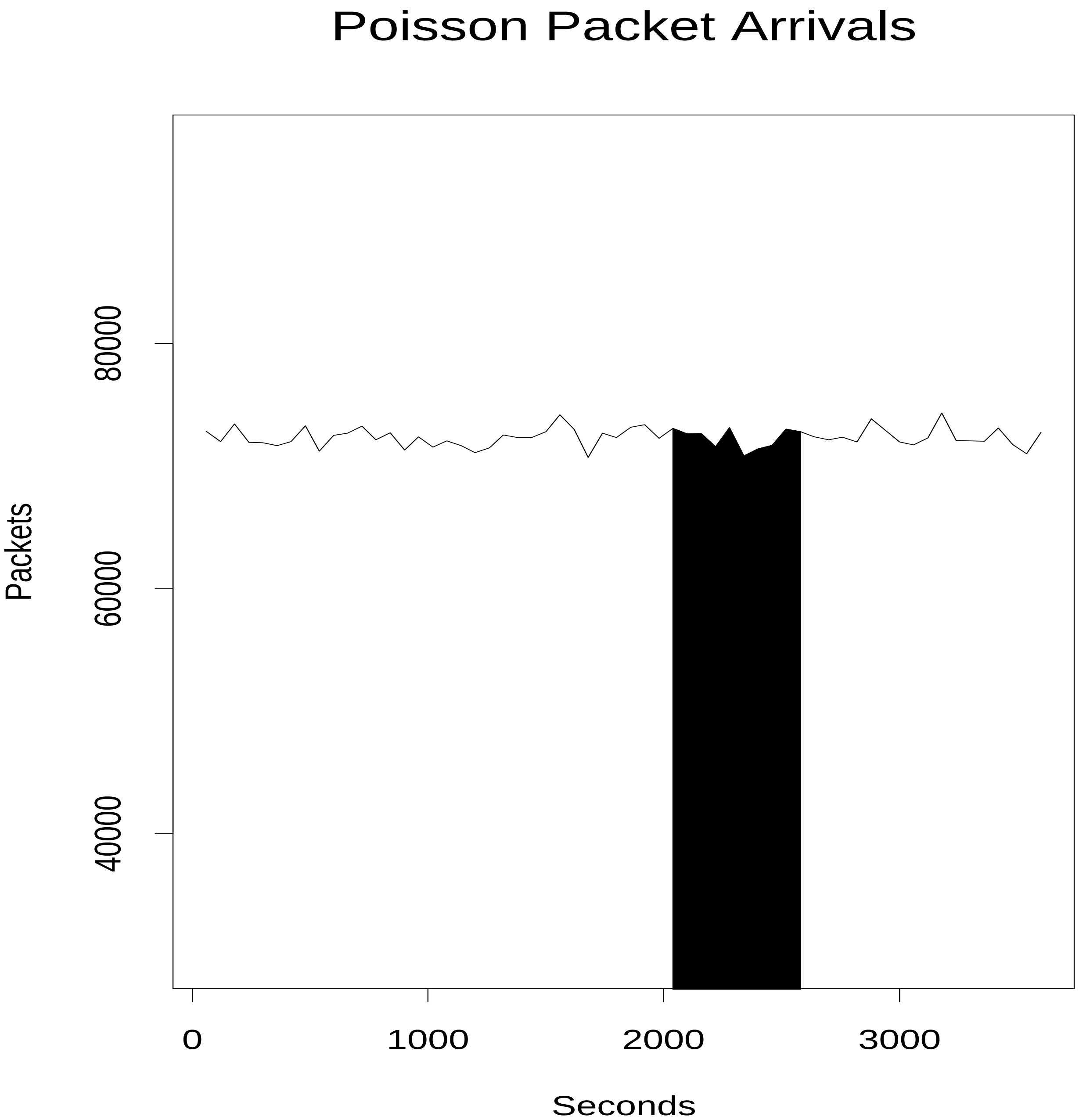
Example:

- take 1 hour of traffic from DEC's Internet link (4.3 Mpkt)
- generate Poisson arrivals . . .
- . . . with same mean and variance [*]

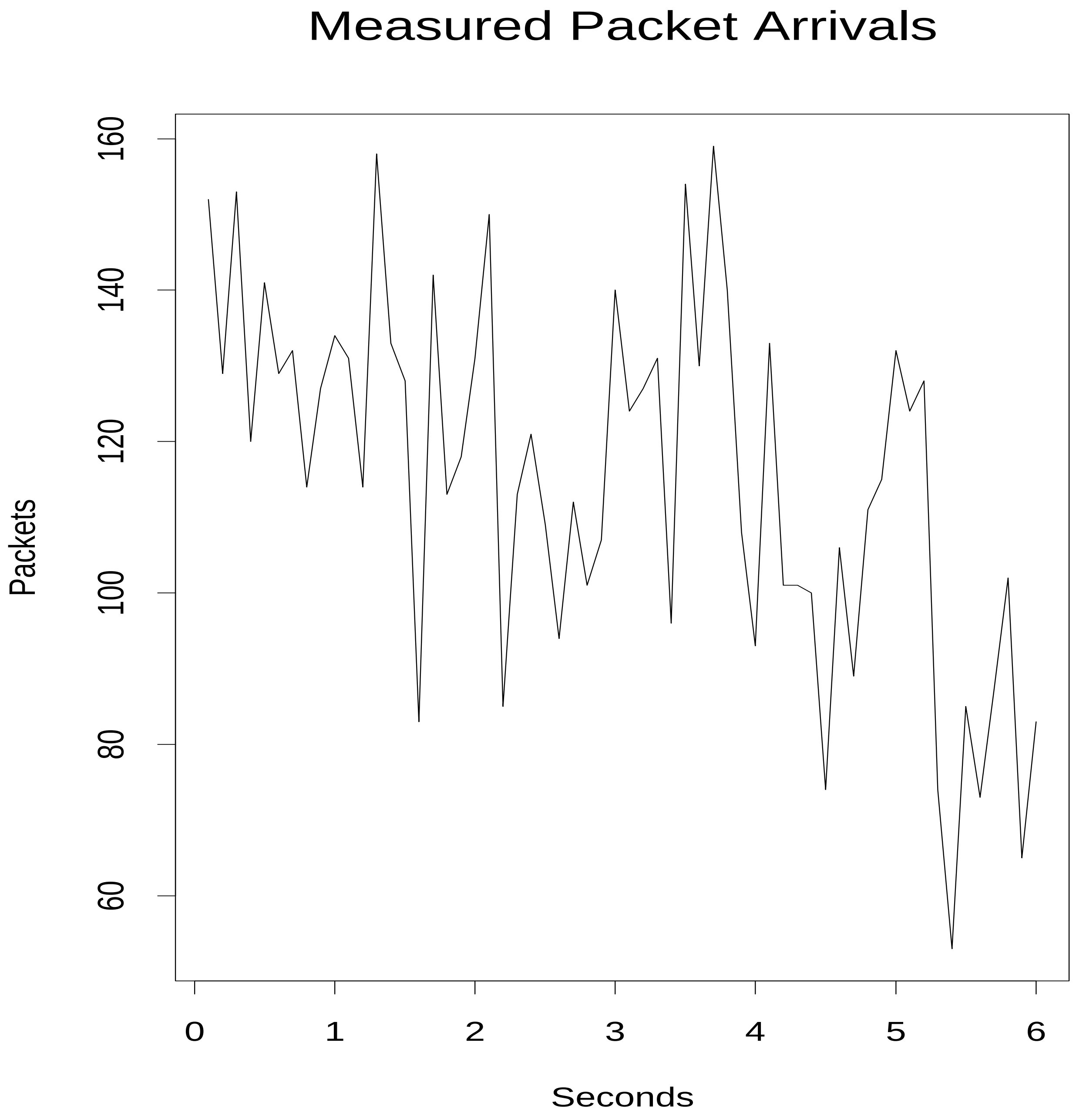

Poisson Packet Arrivals
Packets
60
80
100
120
140
160
0
1
2
3
4
5
6
Seconds

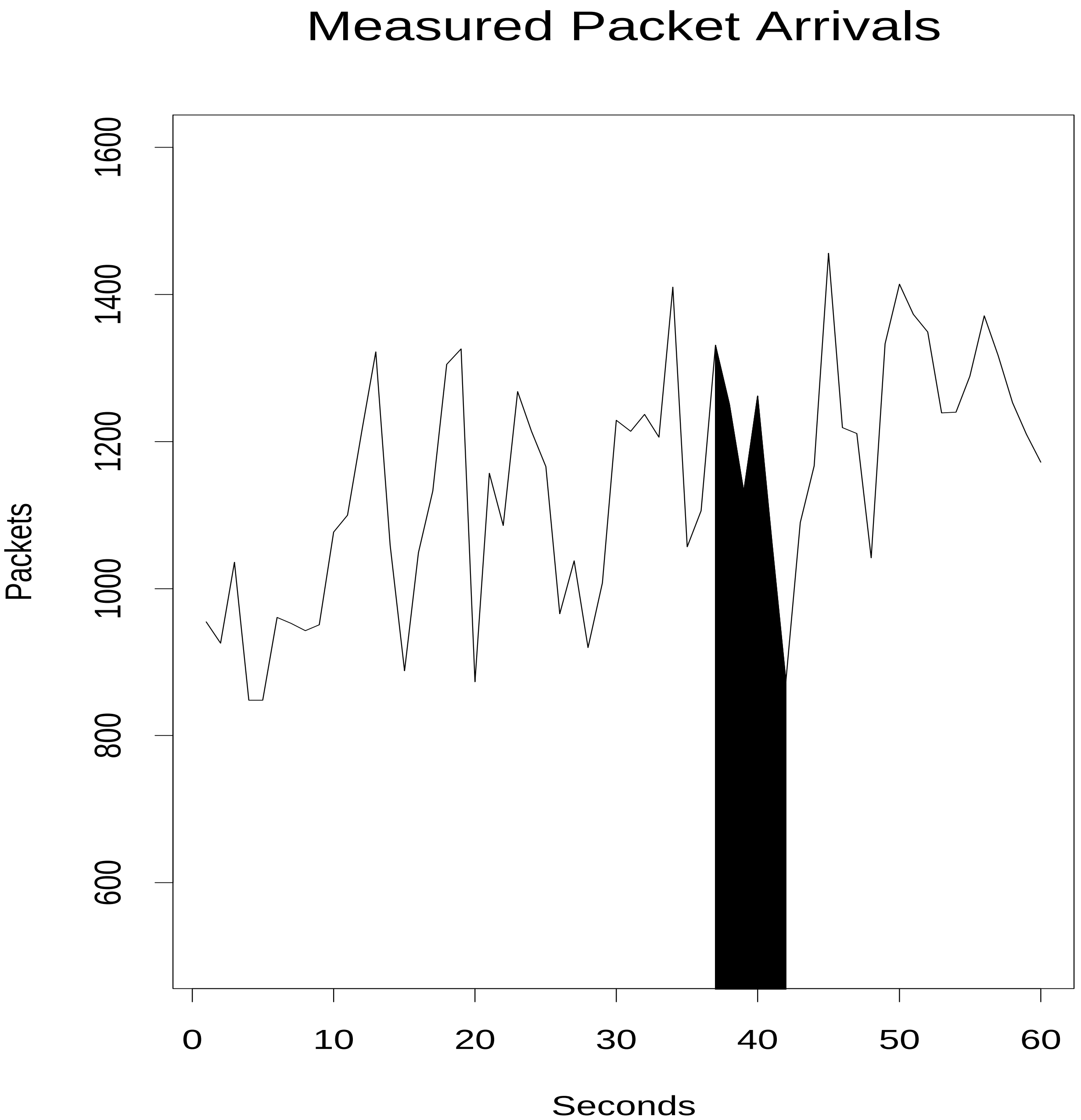

Poisson Packet Arrivals
Packets
600
800
1000
1200
1400
1600
0
10
20
30
40
50
60
Seconds

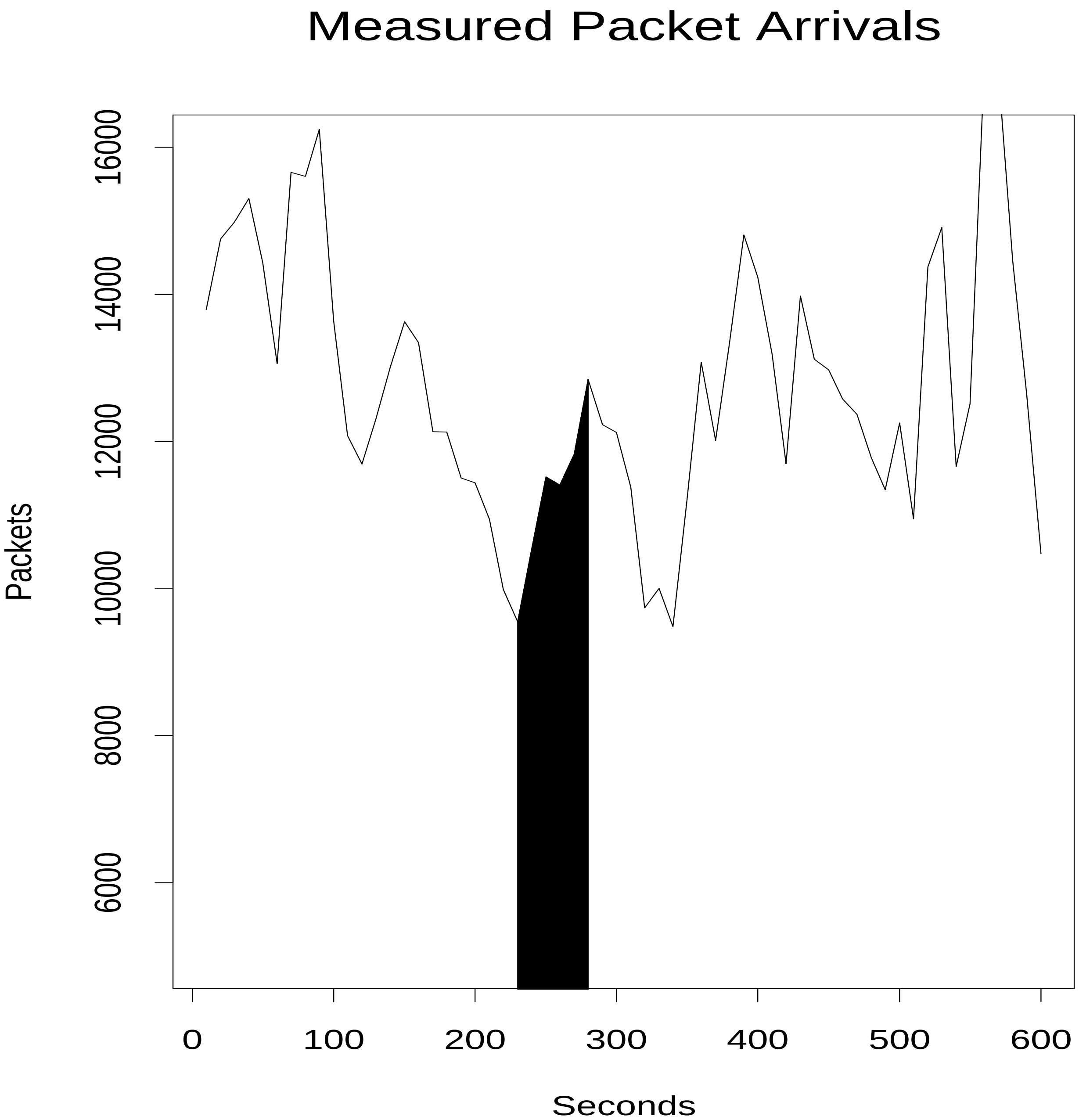

Poisson Packet Arrivals
Packets
16000
14000
12000
10000
8000
6000
0
100
200
300
400
500
600
Seconds

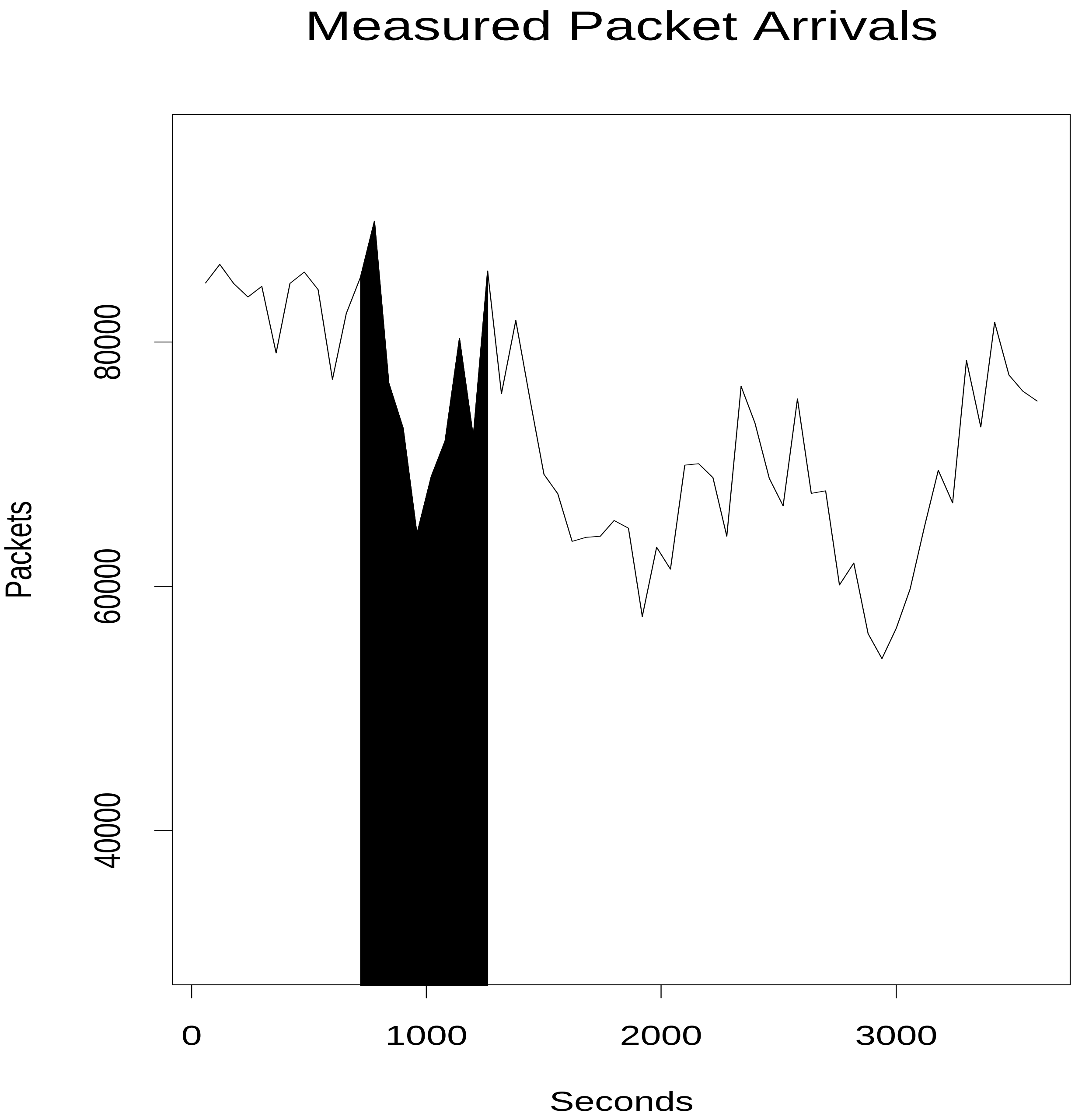


Poisson Packet Arrivals
Packets
40000
60000
80000
0
1000
2000
3000
Seconds

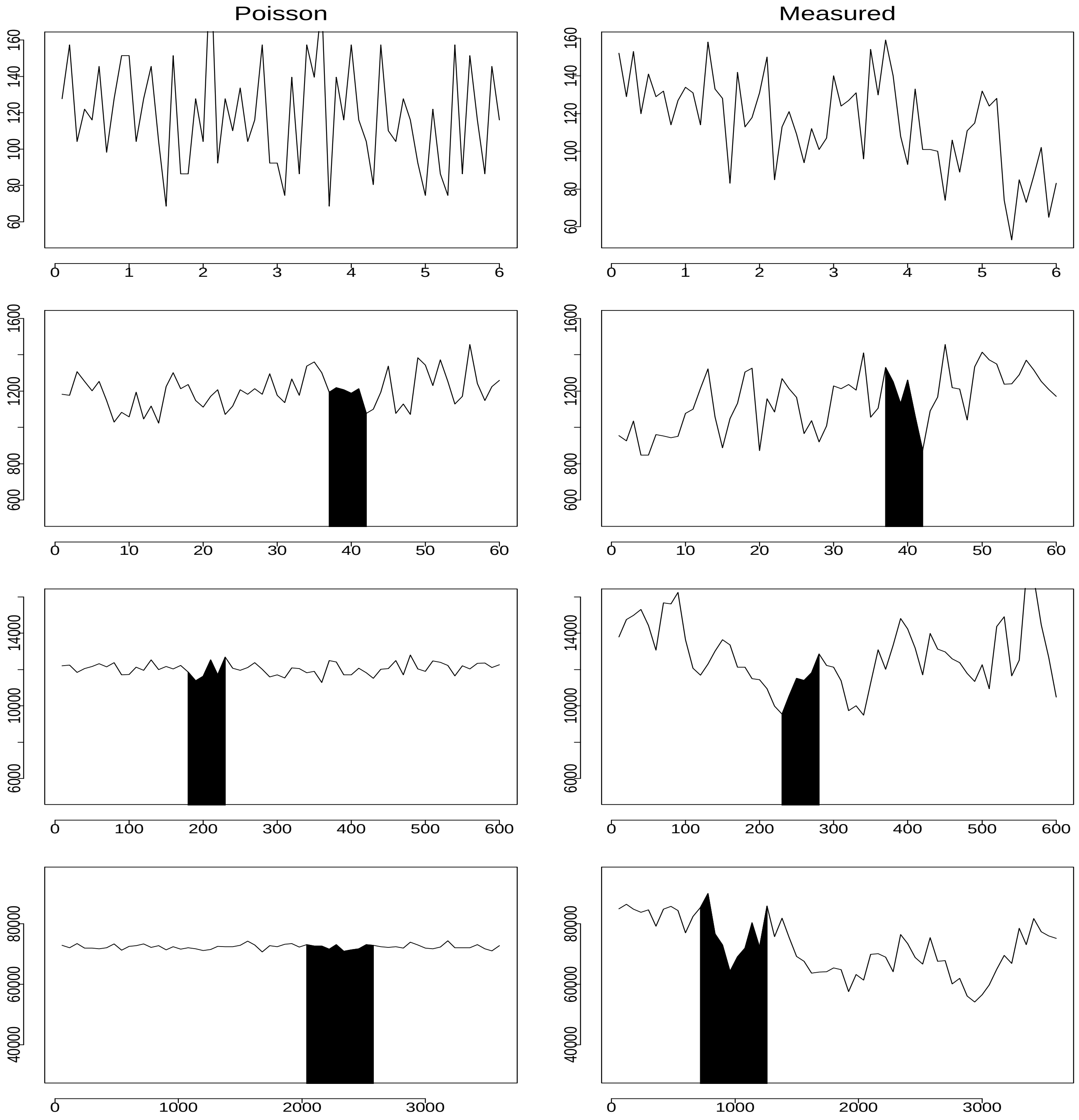

Measured Packet Arrivals
Packets
60
80
100
120
140
160
0
1
2
3
4
5
6
Seconds


Measured Packet Arrivals
Packets
600
800
1000
1200
1400
1600
0
10
20
30
40
50
60
Seconds


Measured Packet Arrivals
Packets
16000
14000
12000
10000
8000
6000
0
100
200
300
400
500
600
Seconds



Measured Packet Arrivals
Packets
40000
60000
80000
0
1000
2000
3000
Seconds



Poisson
Measured
60
80
100
120
140
160
0
1
2
3
4
5
6
600
800
1200
1600
0
10
20
30
40
50
60
6000
10000
14000
0
100
200
300
400
500
600
40000
60000
80000
0
1000
2000
3000

Burstiness over many time scales:

"Visual self-similarity," per previous plots.

Power spectrum: flat for Poisson process, diverges to ∞ for $\omega \to 0$ in measurements.

$\Rightarrow$ To build Poisson-based models for this type of traffic takes major fiddling (many parameters).

$\Rightarrow$ But because the Internet is such a moving target, we are <u>desperate</u> for <u>parsimonious</u> models.

Variance-time plots:

Suppose we have X_k measured for $T = 100$ msec. Considered a “smoothed” version ${X_k}^{(10)}$, corresponding to viewing X_k on a time interval $T' = 10 \times T = 1$ sec:

$$ {X_k}^{(10)} = \frac{1}{10}(X_{10k-10+1} + \cdots + X_{10k}). $$

E.g., ${X_3}^{(10)} = \frac{1}{10}(X_{21} + X_{22} + \cdots + X_{30})$.

More generally, for any degree of smoothing m:

$$ {X_k}^{(m)} = \frac{1}{m}(X_{mk-m+1} + \cdots + X_{mk}). $$

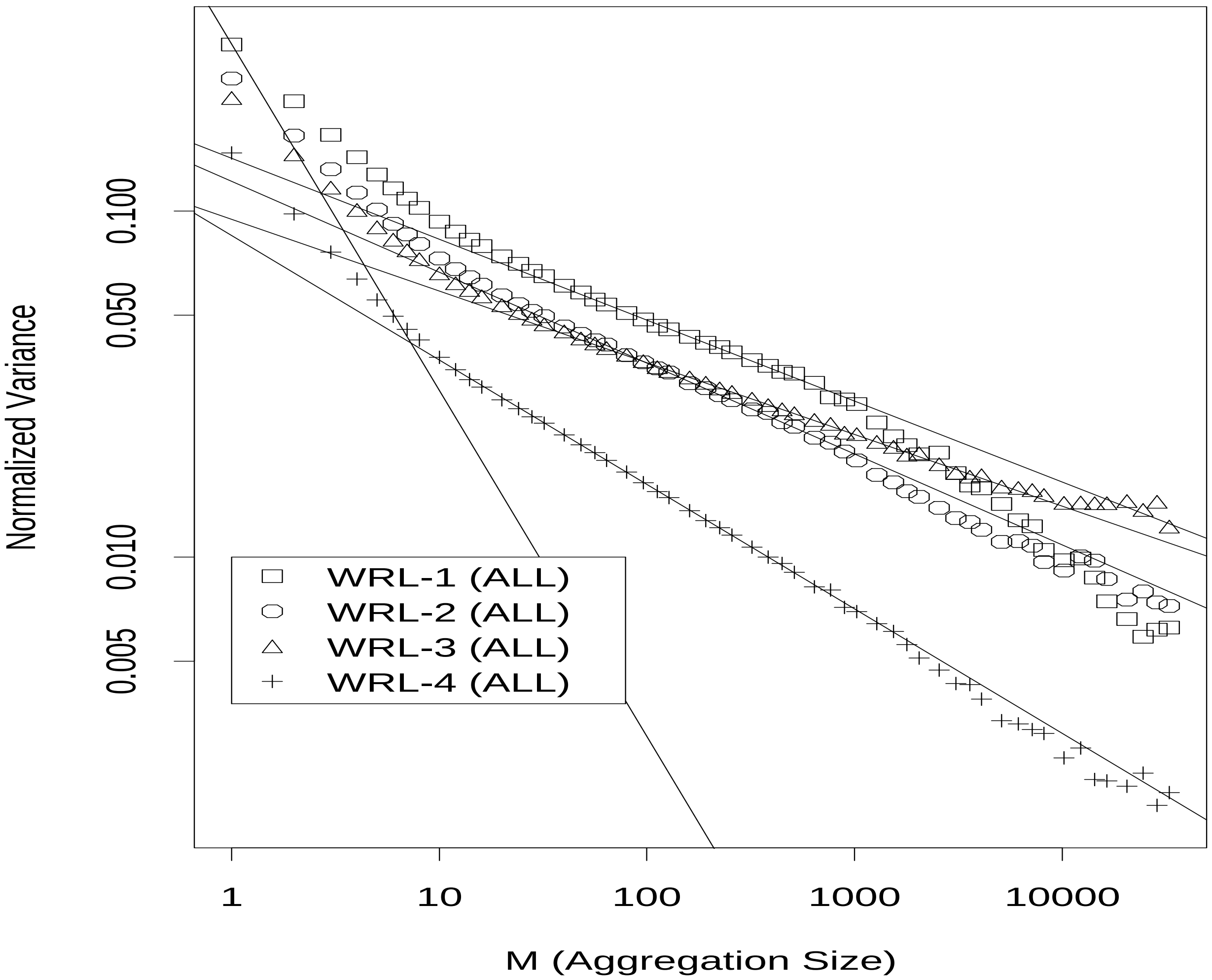
Variance-time plots, con't:

A natural measure of burstiness is *variance*:

$$\mathsf{Var}[Y] = \mathsf{E}[(Y - \mathsf{E}[Y])^2].$$

How does Var$[X_k{}^{(m)}]$ vary with m? For a Poisson process:

$$\mathsf{Var}[X_k{}^{(m)}] \sim m^{-1}.$$


Normalized Variance
0.100
0.050
0.010
0.005
1
10
100
1000
10000
M (Aggregation Size)
WRL-1 (ALL)
WRL-2 (ALL)
WRL-3 (ALL)
WRL-4 (ALL)

A look at fractals:

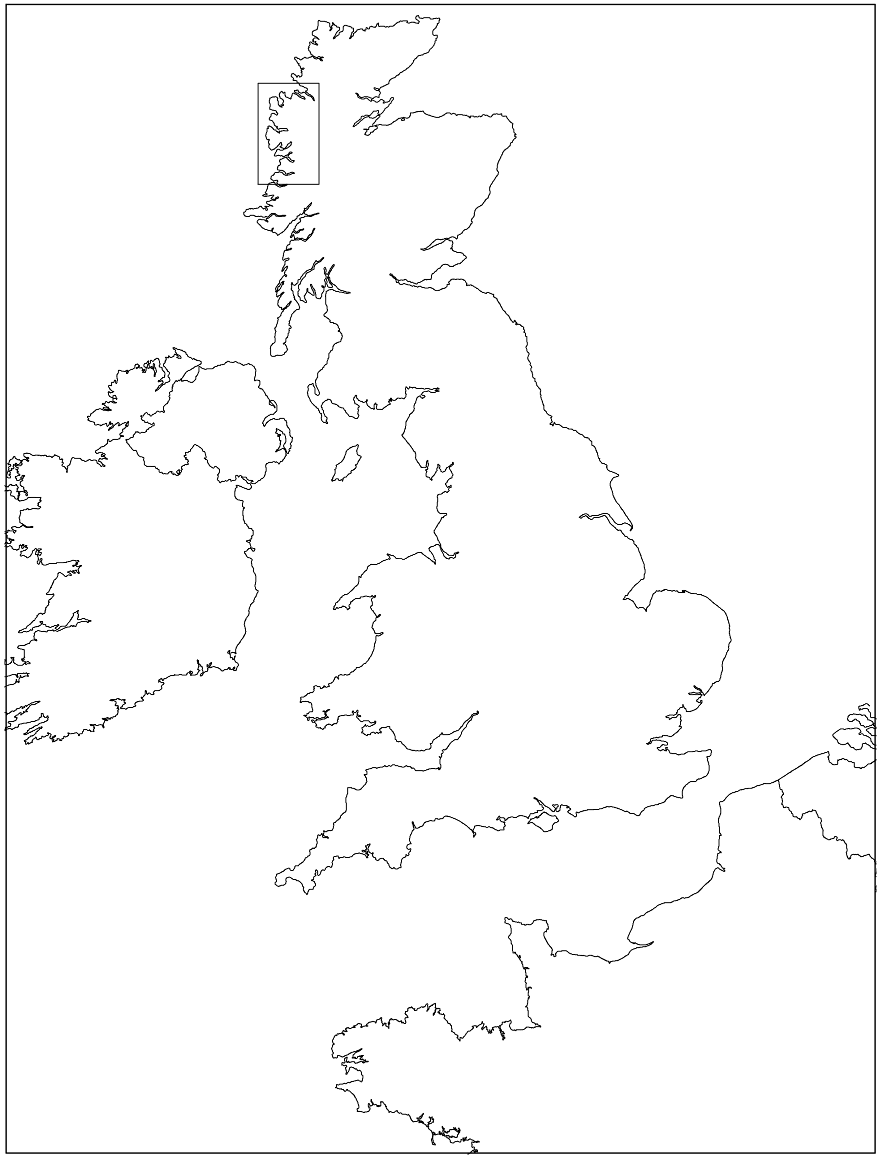
Informally: a (large) class of objects that have surprising scaling properties.

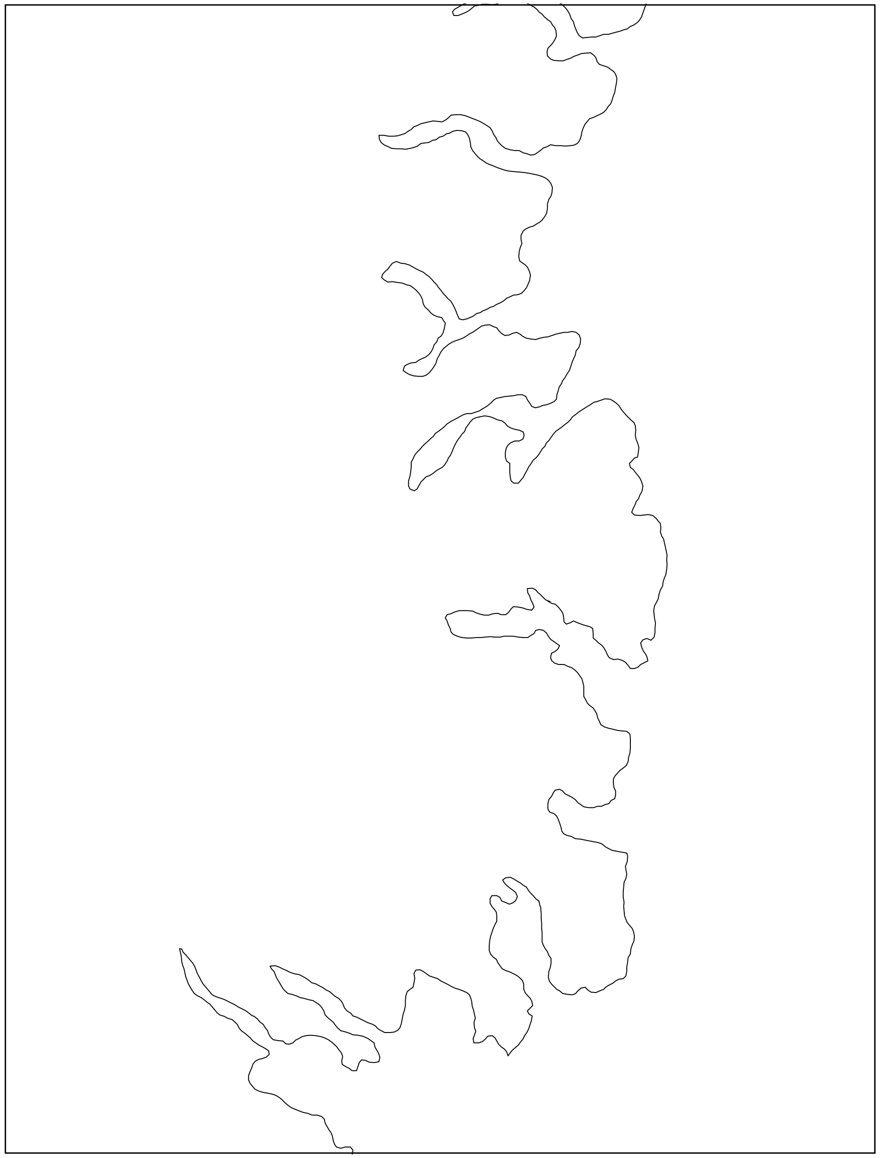
Classic example: How long is the British coastline? Answer: it depends on the length of your ruler.

Length of Coastline: 6,012 km

Length of Coastline: 7,033 km

Length of Coastline: 7,764 km





The fractal scaling property:

There is no natural coastline "length." If you look in different reference books, you get *different answers.*

Just as there is no natural network burst "length."

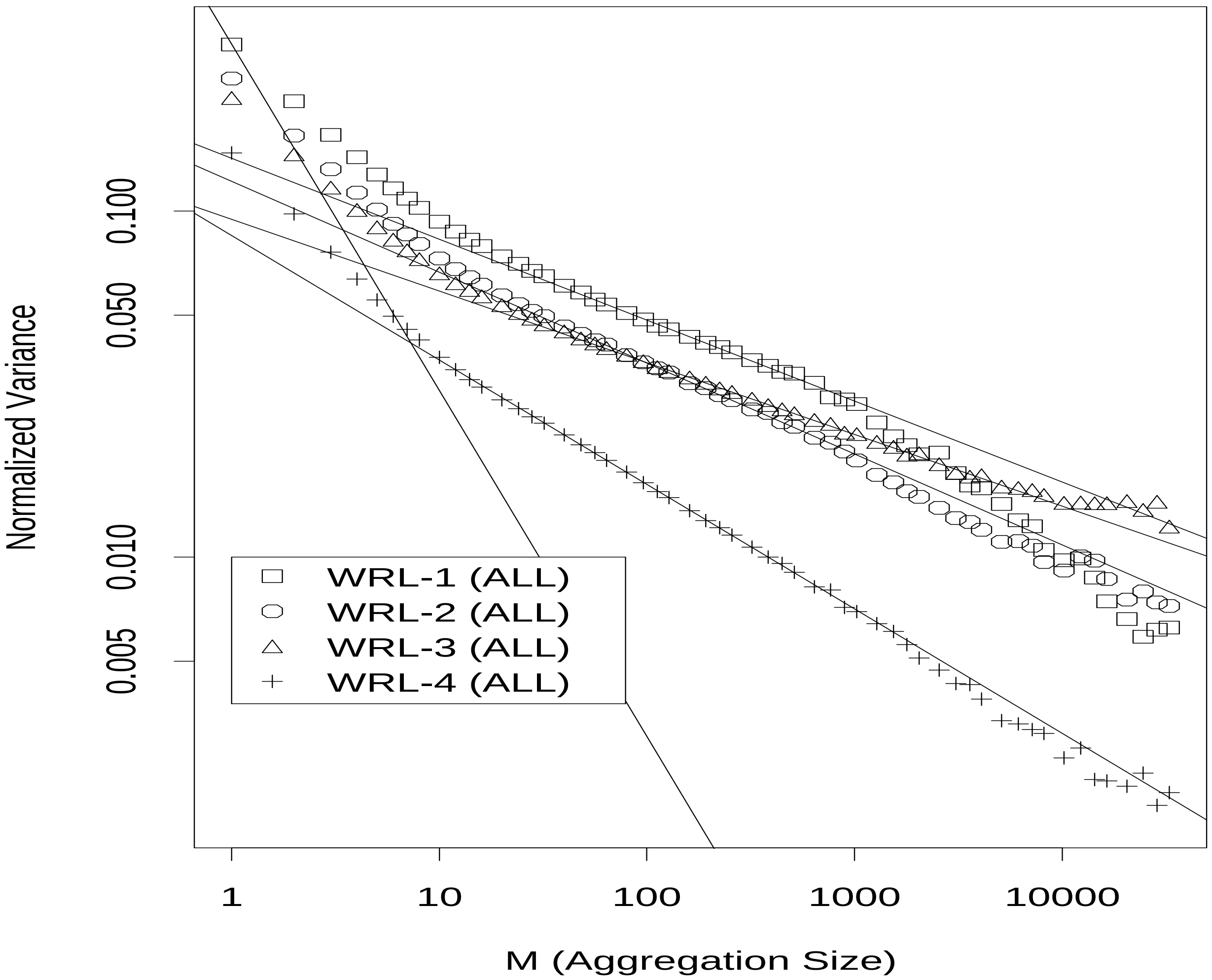
Self-similar processes:

A special type of fractal is a *self-similar* process. Informally: something that "looks the same" regardless of scale. More formally: a stochastic process for which the autocorrelation of ${X_k}^{(m)}$ is the same as that for X_k.

Properties: "long-range dependence," or "long memory." <u>Activity at all time scales.</u>

Signature: variance decays slowly as scale increases.

$$\mathsf{Var}[{X_k}^{(m)}] \sim m^{-\beta}, \quad 0 < \beta < 1.$$

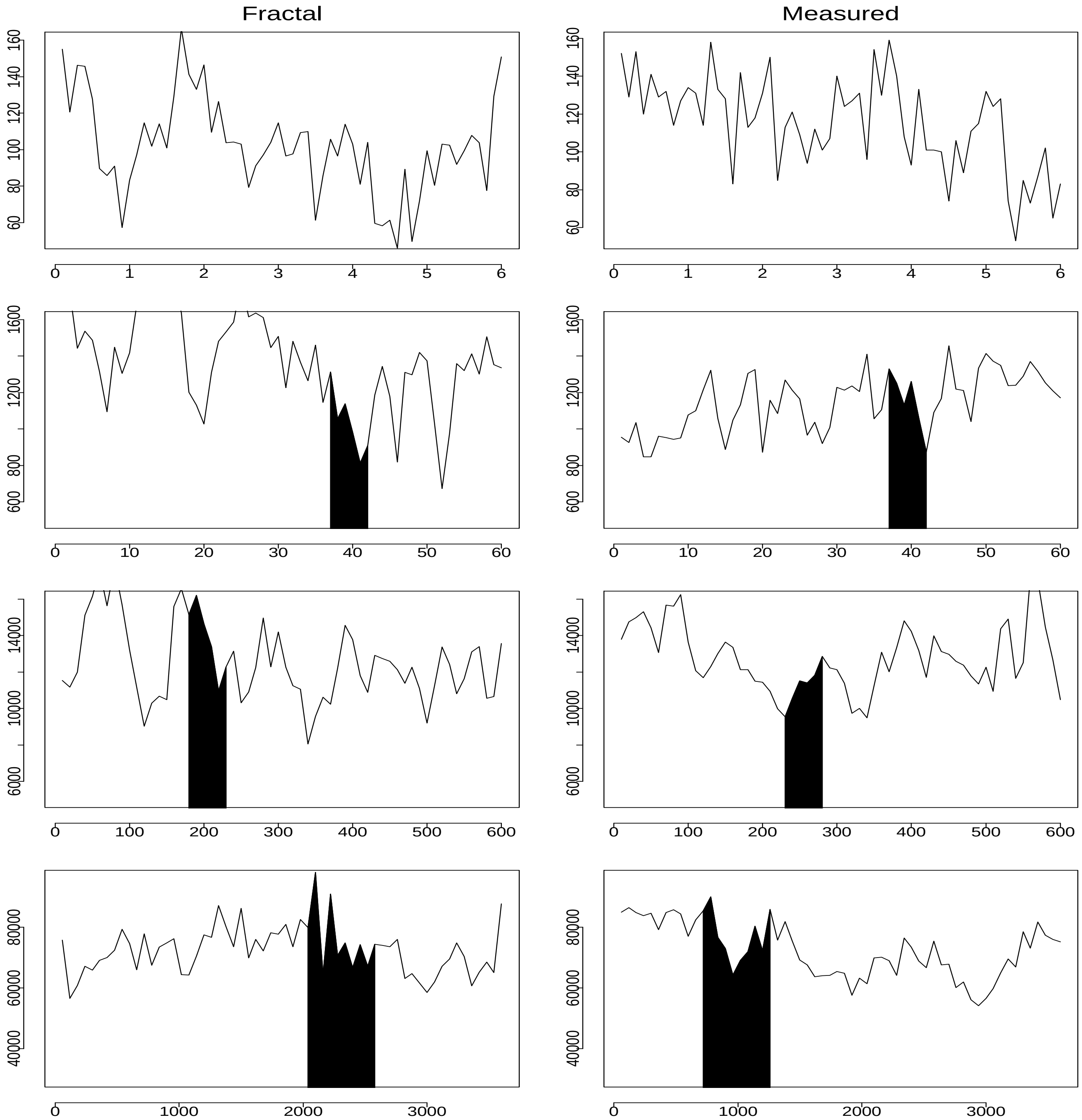

Normalized Variance
0.100
0.050
0.010
0.005
1
10
100
1000
10000
M (Aggregation Size)
WRL-1 (ALL)
WRL-2 (ALL)
WRL-3 (ALL)
WRL-4 (ALL)

How does self-similarity relate to reality?

In the 1940-50's, Harold Edwin Hurst (“Abu Nil”) studies the 800-year record of flooding along the Nile river. Finds long-range dependence.

Is the Nile special? No: the Saint Lawrence, Colorado, Mississippi, Loire rivers all have this property. (The Rhine doesn't!)

Implications: it's hard to predict floods (long memory).


Fractal
Measured
60 80 100 120 140 160
0 1 2 3 4 5 6
600 800 1200 1600
0 10 20 30 40 50 60
6000 10000 14000
0 100 200 300 400 500 600
40000 60000 80000
0 1000 2000 3000

Searching for a different model:

1991 Fowler & Leland, looking at LAN traffic, observe that, in reality: *Traffic `spikes' (which cause actual losses) ride on longer-term `ripples', that in turn ride on still longer-term `swells'* [FL91].

1993 Leland, Taqqu, Willinger & Wilson analyze additional LAN traffic. Propose self-similar models [LTWW93]. Bombshell!

1993 Garrett studies coded Star Wars video: finds long-range dependence in corresponding bit-rate [Ga93].

Searching for a different model, con't:

1993 Beran, Sherman, Taqqu & Willinger: video is replete with long-range dependence [BSTW95]

1994 Paxson & Floyd study WAN traffic. Find much closer to self-similar than any Poisson models [PF94]. From *skeptic* $\Rightarrow$ *proponent.*

Others: LAN, NSFnet backbone, office automation network traffic all much closer matched by self-similar models.

How can something as bizarre as self-similarity come about?

Let's talk money.

Question: How many double-billionaires are there?*

[*] Fortune magazine, September 11 1989: no Bill Gates!

Talking money, con't:

Some possibly helpful information: in 1989, the median net worth in the U.S. was \$47,700. 80% of the people had a net worth $\leq$ \$627,000.

Fitting a normal distribution to these values predicts that, out of a population of 5 billion people, the single wealthiest person would be worth around \$4.35 million.

Answer: 65.

Way more than any “reasonable” distribution of wealth allows.

$\Rightarrow$ A handful of people have all the money.

How many people are worth more than US$ 5,000,000,000? Is anyone worth more than US$ 10,000,000,000?


Who Has All The Money?
Billions
25
20
15
10
5
0
Ranking

Pareto's law:

$\Rightarrow$ A handful of billionaires have all the billions.

Income distribution has a **heavy tail**: $F(x) = 1 - (x/a)^{-\beta}$.

In economics, this is known as Pareto's law (circa 1900). True for many different societies.

If $\beta < 2$, then the distribution has infinite variance.

Pareto distributions crop up *everywhere*.

Heavy tails:

A handful of cities have all the people.

A handful of asteroids have all the mass.

A handful of words have all the use.

A handful of islands have all the area.

A handful of earthquakes do all the damage.

A handful of extinctions wiped out all the animals.

A handful of non-coding DNA strands have all the length.

Implications of Pareto distributions:

The Pareto distribution is what you get if the *logarithm* of what you are measuring has an exponential distribution. That is, the *logarithm* is "memoryless" $\Rightarrow$ the thing you are measuring is independent of scale!

$\Rightarrow$ There are theorems directly relating processes having Pareto durations to self-similar behavior.

Relationship between heavy tails and self-similarity:

Suppose you have an ON/OFF process, for which <u>either</u> the ON or the OFF times have a heavy-tailed distribution (infinite variance). If you add such processes together (such as by sharing a network link), the aggregation is asymptotically self-similar [WTSW95].

[LTWW93] told us: <u>Go look for heavy tails.</u>

Heavy tails in computing:

CPU time consumed by Unix processes [LO86,H-BD96].

Size of Unix files [Ir93].

Size of compressed video frames [Ga93].

Size of FTP bursts [PF94].

Telnet packet interarrivals [DJCME92,PF94].

Size of Web items [CB96].

Ethernet bursts [WTSW95].

Complementary distribution plots:

For an exponential distribution:

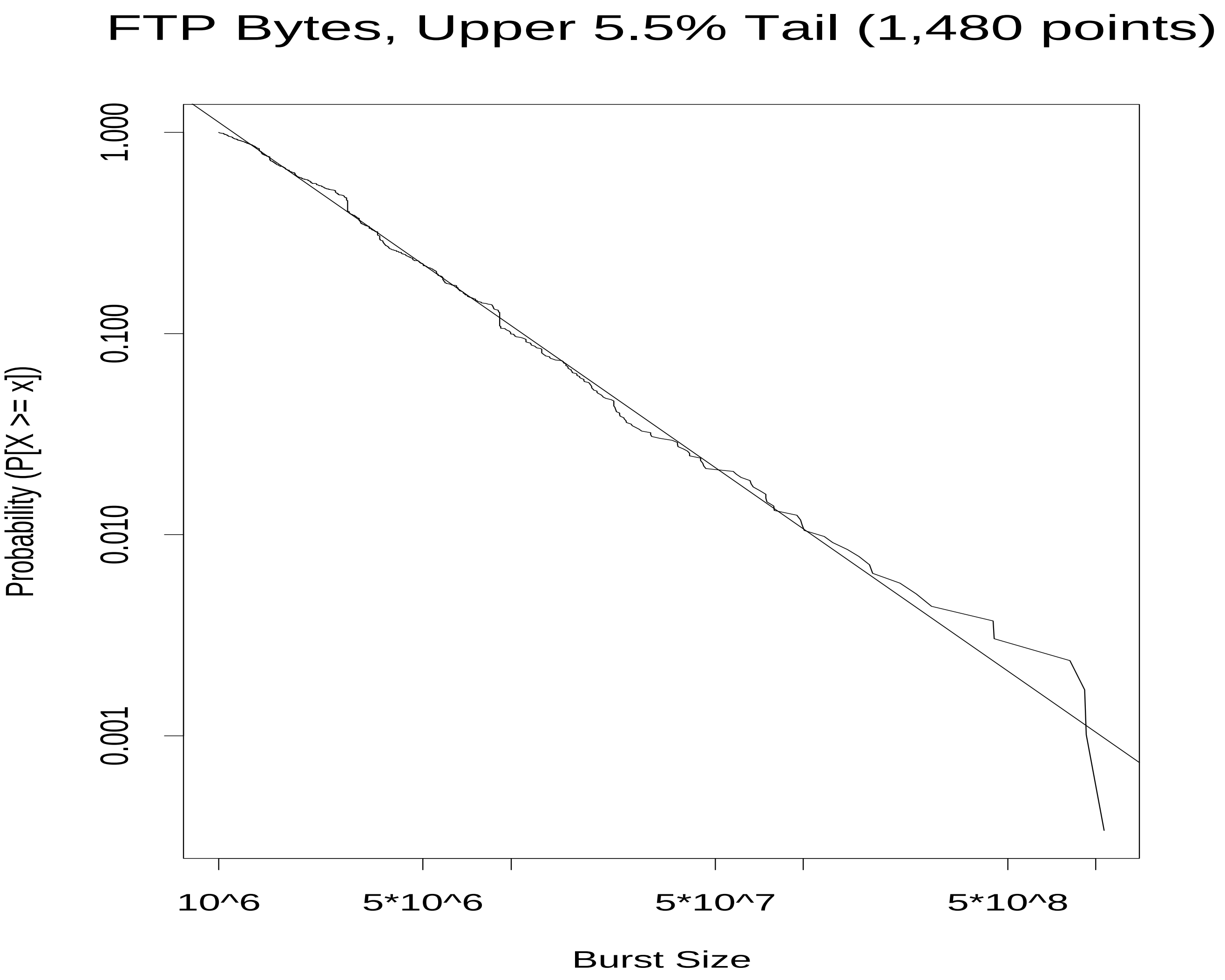
$$F(x) = 1 - e^{-\lambda x}$$

$$\overline{F}(x) = 1 - F(x) = e^{-\lambda x}$$

$$\log \overline{F}(x) = -\lambda x.$$

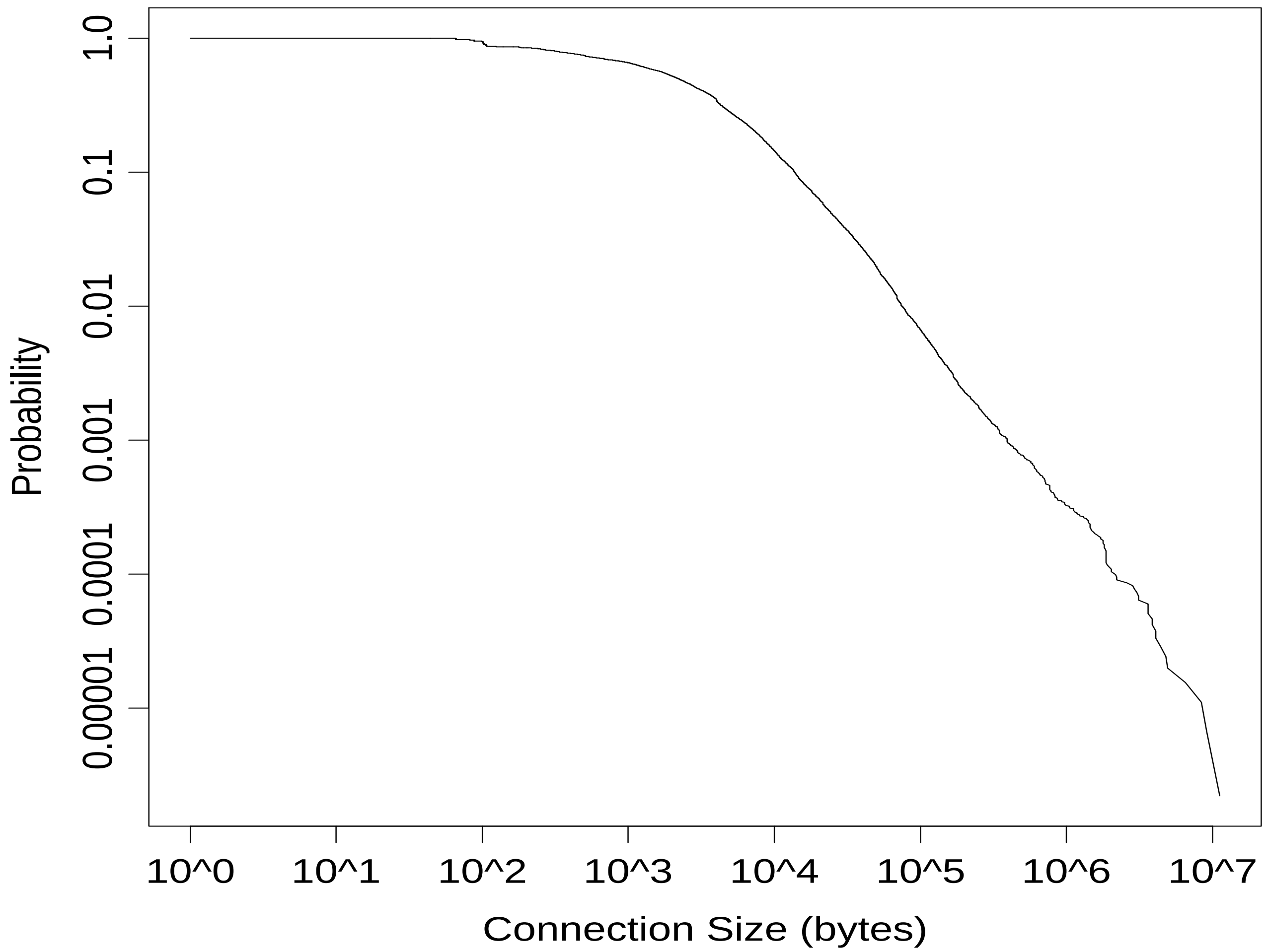
So you can check for an exponential distribution by plotting $\log \overline{\widehat{F}}_n(x)$ vs. x. If it's a straight line, the fit is good, and the slope gives $-\lambda$.

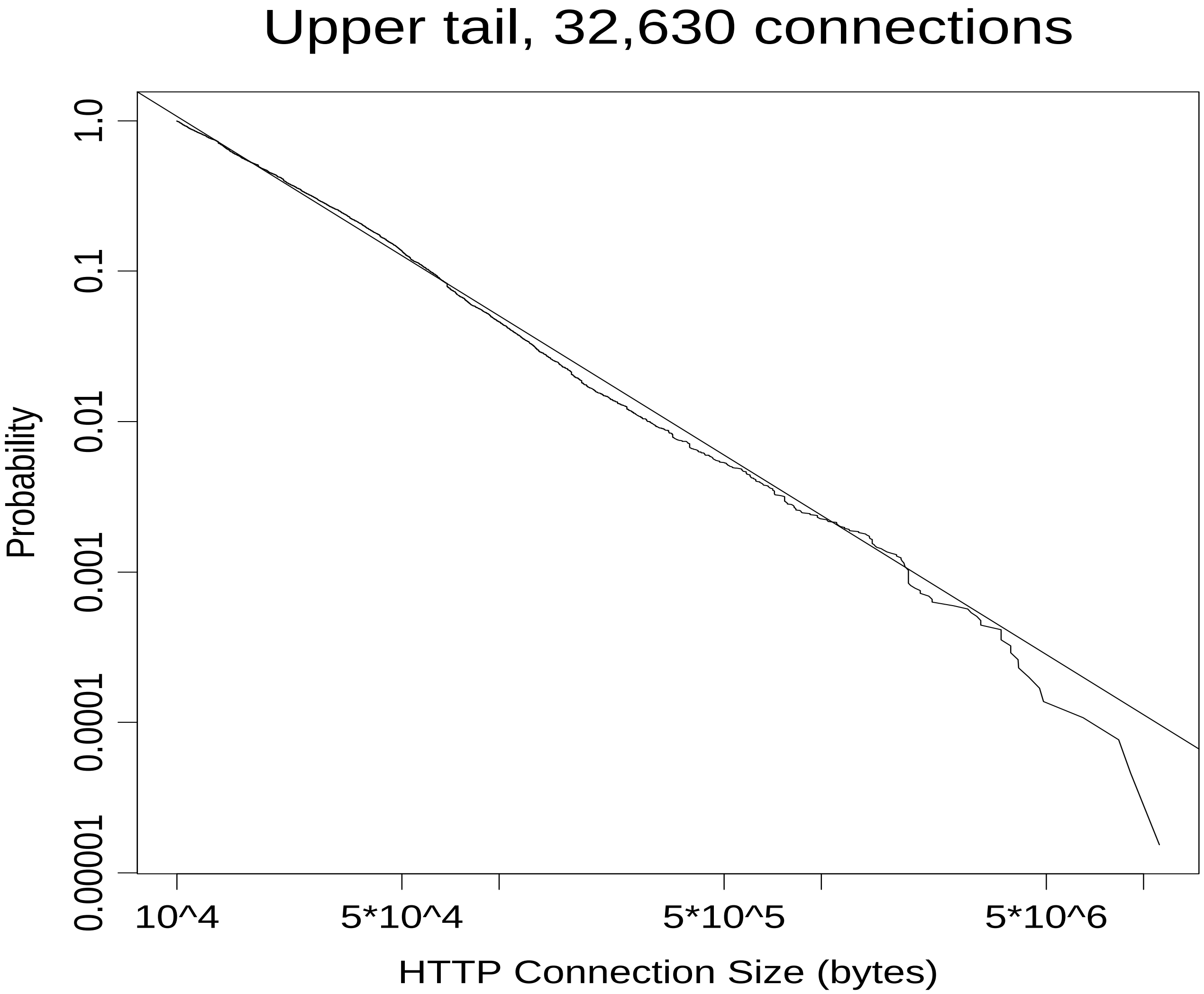
For Pareto, just plot $\log \overline{\widehat{F}}_n(x)$ vs. $\log x$.



FTP Bytes, Upper 5.5% Tail (1,480 points)
Probability (P[X >= x])
1.000
0.100
0.010
0.001
10^6
5*10^6
5*10^7
5*10^8
Burst Size

226,386 HTTP Connections




Upper tail, 32,630 connections
Probability
1.0
0.1
0.01
0.001
0.0001
0.00001
10^4
5*10^4
5*10^5
5*10^6
HTTP Connection Size (bytes)

What does this mean for network traffic?

Queueing delays are *much higher* in the presence of long-range dependence than for Poisson traffic [ENW96].

To avoid dropping packets, buffers have to be <u>huge</u>.

You have to be very careful predicting future traffic based past measurements.

You can't look at a little bit of video and decide how much buffer it's going to require.

What does this mean for network traffic, con't?

But note:

- You can still have a multiplexing gain, it's just less.
- You can have strong correlations that are not large.
- There's still a lot of haggling over when it matters and when it doesn't [RE96,GB96].

Assessing self-similarity:

H can be estimated using:

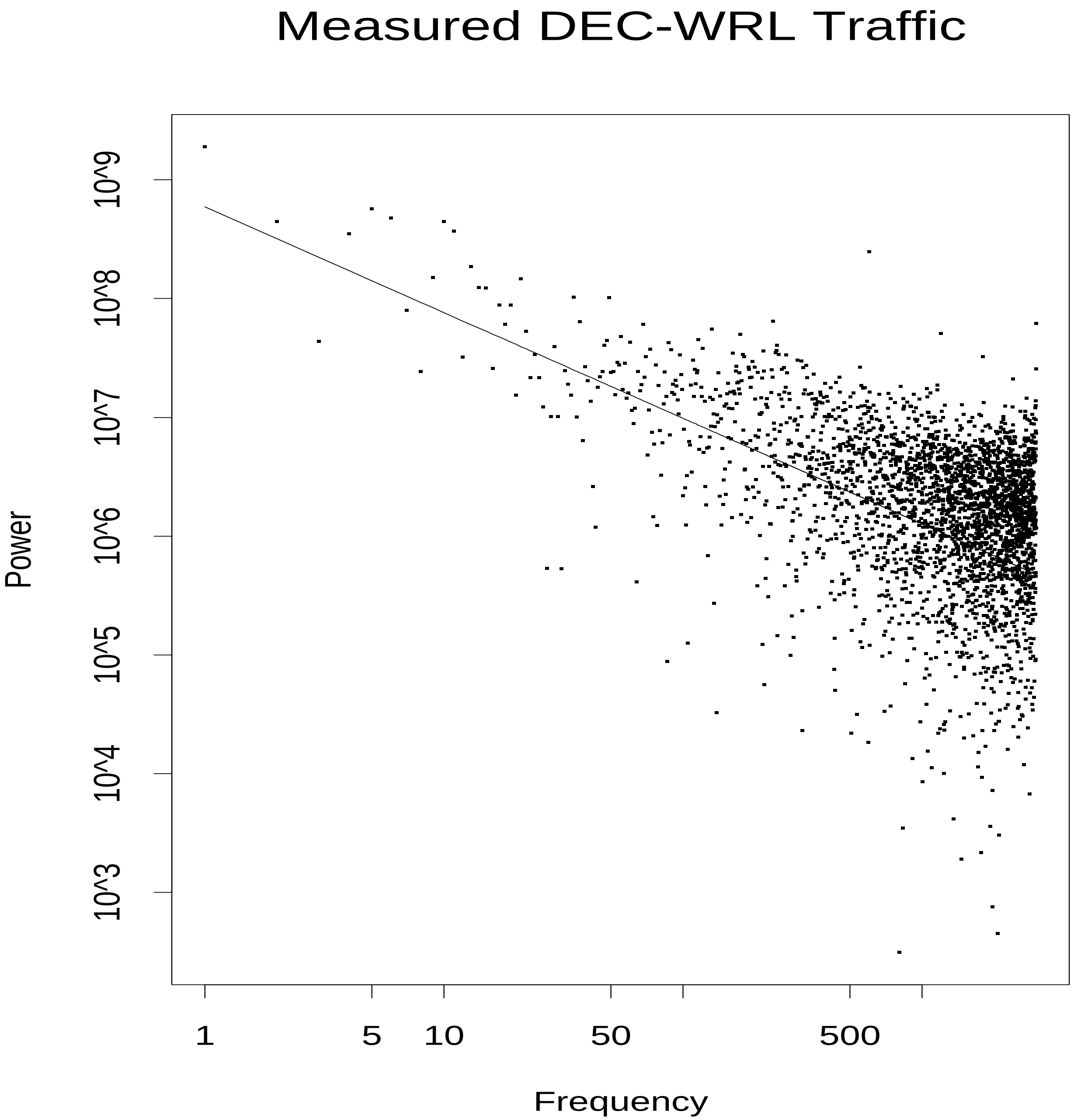
- variance-time plot — slope estimates $-\beta$
- R/S plot [LTWW94] — hard to interpret
- log power / log frequency plot — power diverges as $\omega \to 0$

These are all (valuable) heuristics.

They all have problems with bias [TTW95].

Better: Whittle's estimator, if Gaussian process [Be94].

Best: wavelet techniques [AV98].


Measured DEC-WRL Traffic
Power
10^9
10^8
10^7
10^6
10^5
10^4
10^3
1
5
10
50
500
Frequency

Pitfalls in assessing self-similarity:

1. Looking at non-stationary traffic. (E.g.: 6 hours' worth of traffic.)
2. Using a “visual self-similarity” plot with autoscaling on the Y-axis.
3. Eyeballing the “straight-line” variance-time plot fit instead of actually fitting to it.
4. Glossing over discrepancies between different Hurst estimators.
5. Not worrying about Hurst estimates very near 1.0.
6. Failing to try wavelet techniques (or Whittle).

Major research questions for fractal modeling:

How do we incorporate short-range dependence? What is the right model for short-range dependence, anyway?

How do we model <u>interarrivals</u> given that we know <u>arrival counts</u>?

<u>Lots</u> of remaining work in understanding causes, implications, when it matters & when it doesn't, good measurement techniques, ways to ameliorate, etc.

Lessons of the fractal revolution:

“[Hurst put us] in one of those situations, so salutary for theoreticians, in which empirical discoveries stubbornly refuse to accord with theory.” - E.H. Lloyd

$\Rightarrow$ Measure, measure, measure . . .

. . . and *analyze*.

Expect heavy tails — think about how to accommodate them in your designs.

And as soon as you find heavy tails, think *fractal*.

Attempting sound measurement – assessing throughput:

Common measurement question: what is the throughput from Internet host A to Internet host B?

You see this question and answers to it all the time. Most of those answers are wrong, for different definitions of “wrong.”

Let's rule out blasting UDP packets from A to B and counting the survivors. (*Why?*)

Can we just run `ttcp` from A to B, write down its number, and call it a measurement?

Assessing throughput, con't:

Goal #1: Convince you that to understand network behavior, you have to look at the wire.

Goal #2: Convince you that, even doing this, you have to think.

Goal #3: Warn you that your tools can lie to you.

Sequence plots:

Recall that each data byte has a sequence number giving its offset from the beginning of the connection.

The difference between two sequence numbers is a byte count.

The same byte might be (re)transmitted at different physical locations in different packets, but it always has the same sequence number.

Sequence plots, con't:

A TCP packet's header identifies the sequence number of the first byte in the payload. The packet's length (given in IP header) then implicitly gives sequence number of last byte in payload.

Each TCP acknowledgement gives the sequence number just beyond what's been received "in sequence."

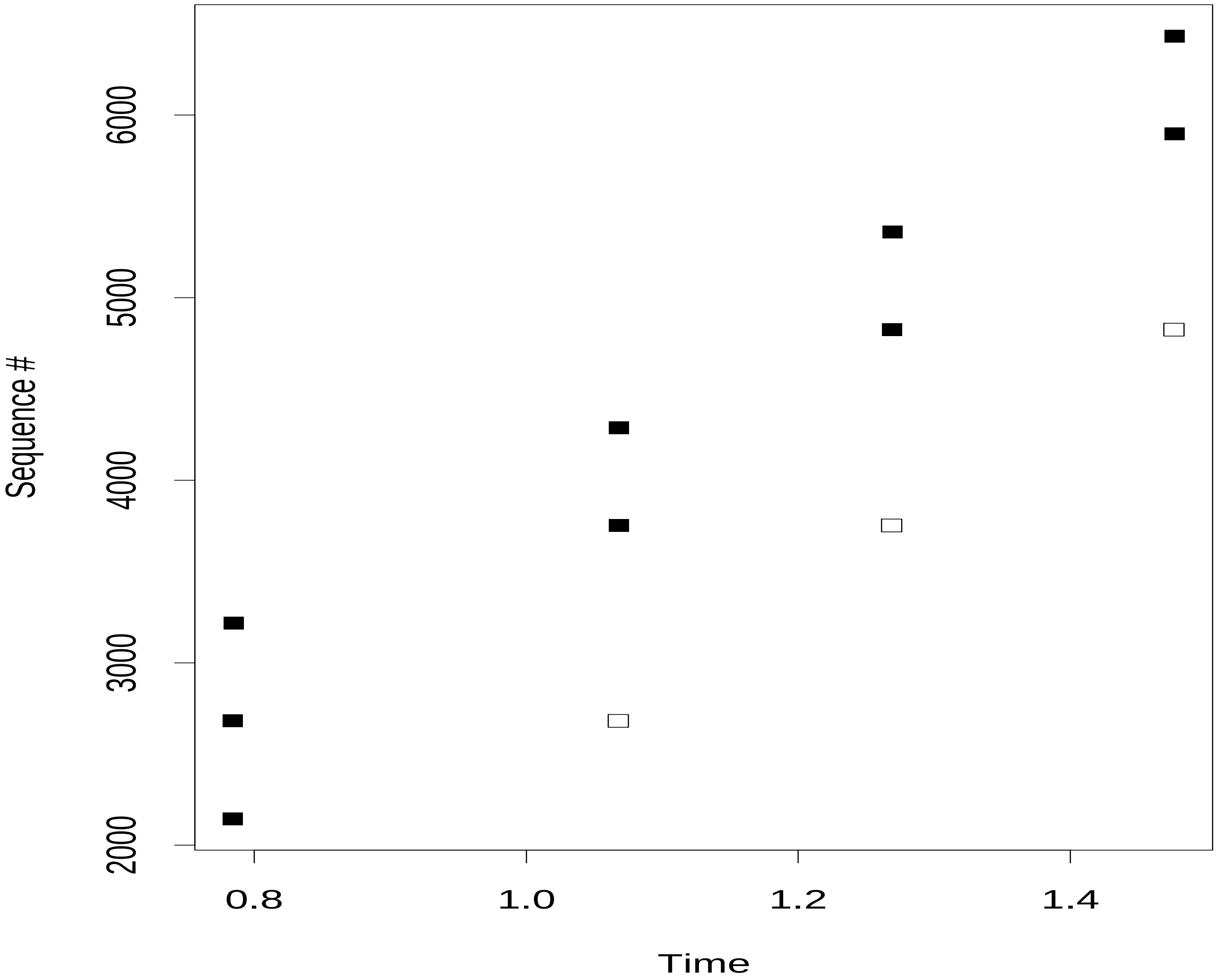
So if a packet with b bytes of data and starting sequence S arrives in sequence, the receiving TCP will ack $S + b$.

Sequence plots, con't:

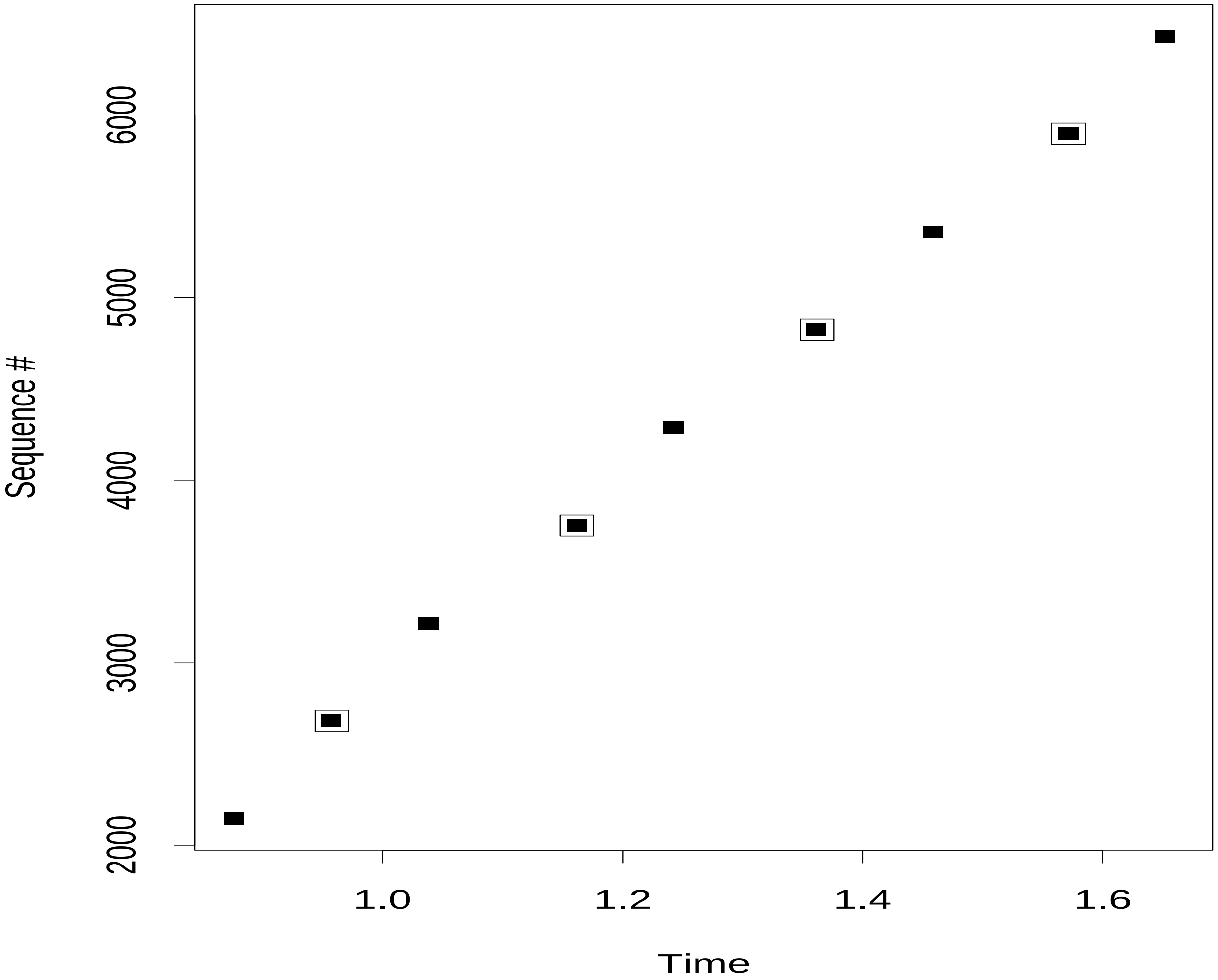
A fundamental tool for analyzing TCP traffic is `tcpdump` plus a *sequence-time* plot. This is just a plot of time vs. TCP (relative) sequence number. Greg Minshall's `tracelook` tool in the Internet Traffic Archive does this.

http://www.acm.org/sigcomm/ITA/

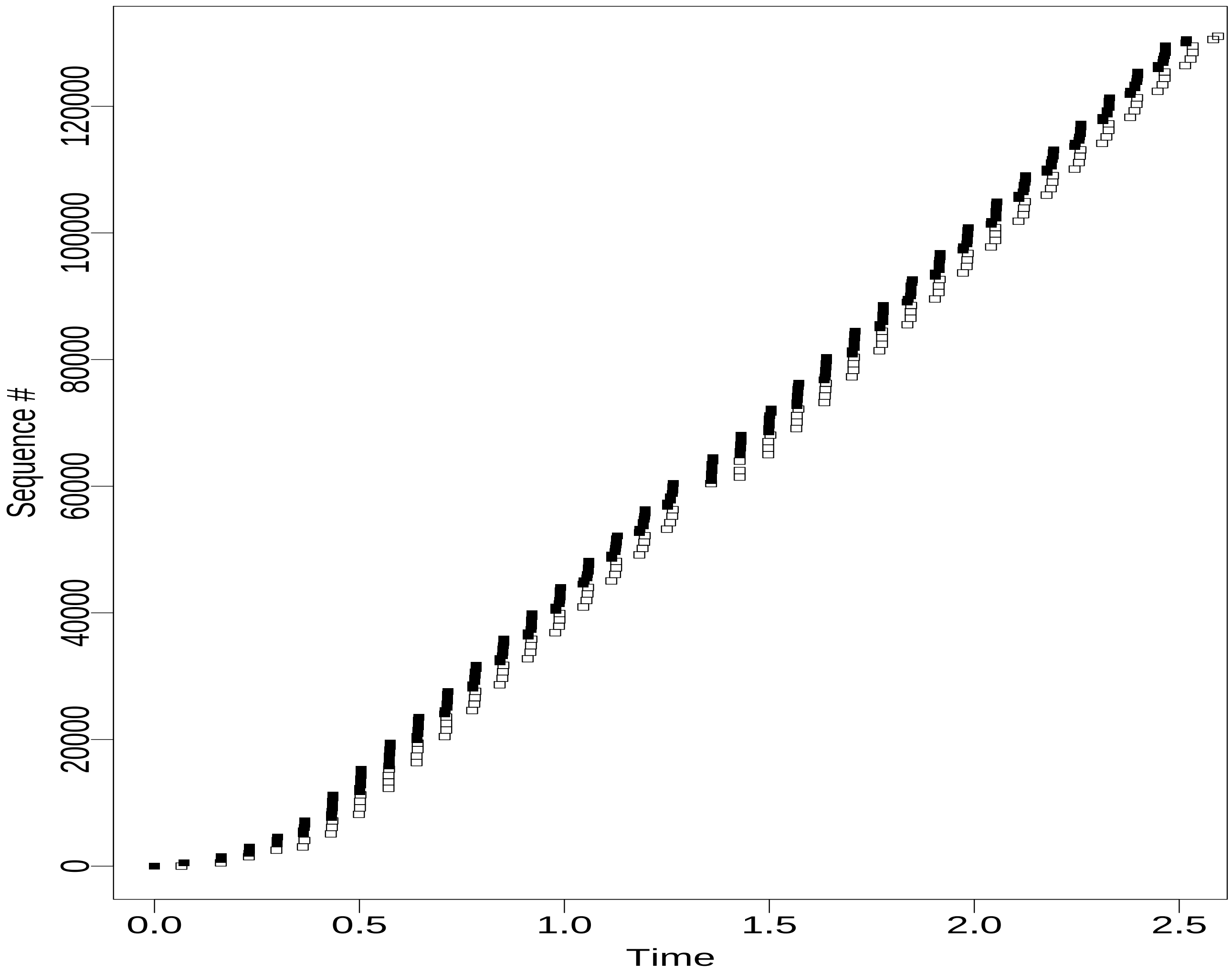
By convention, the plot shows packets according to their highest sequence number (either $S + b$ or what they ack).



Plot gives us: MSS, window size, RTT, data rates.



Same packets plotted <u>at receiver</u>.



Total throughput: 50.1 KB/sec.

Is it so low because TCP is saddled with too much overhead?

Fundamentals of TCP performance:

A TCP sender has a <u>window</u> limiting how much data it can have in flight at one time.

The window fixes the (maximum) sending rate.

If window W bytes and round-trip time is RTT sec, then:

$$\text{Max Throughput} = \frac{W}{\text{RTT}} \text{ bytes/sec.}$$

because it takes time RTT to get an acknowledgement for a given packet and "advance" the window.

TCP performance fundamentals, con't:

In the previous example, $W = 4096$ bytes and $T = 0.066$ sec.

So, maximum throughput is $4096/0.066 = 62,060$ bytes/sec.

Q: If the connection achieves 62 KB/sec, what have we learned about the network path?

A: Very little! Just that it can support at least 62 KB/sec.

It could be <u>much</u> faster.

TCP performance fundamentals, con't:

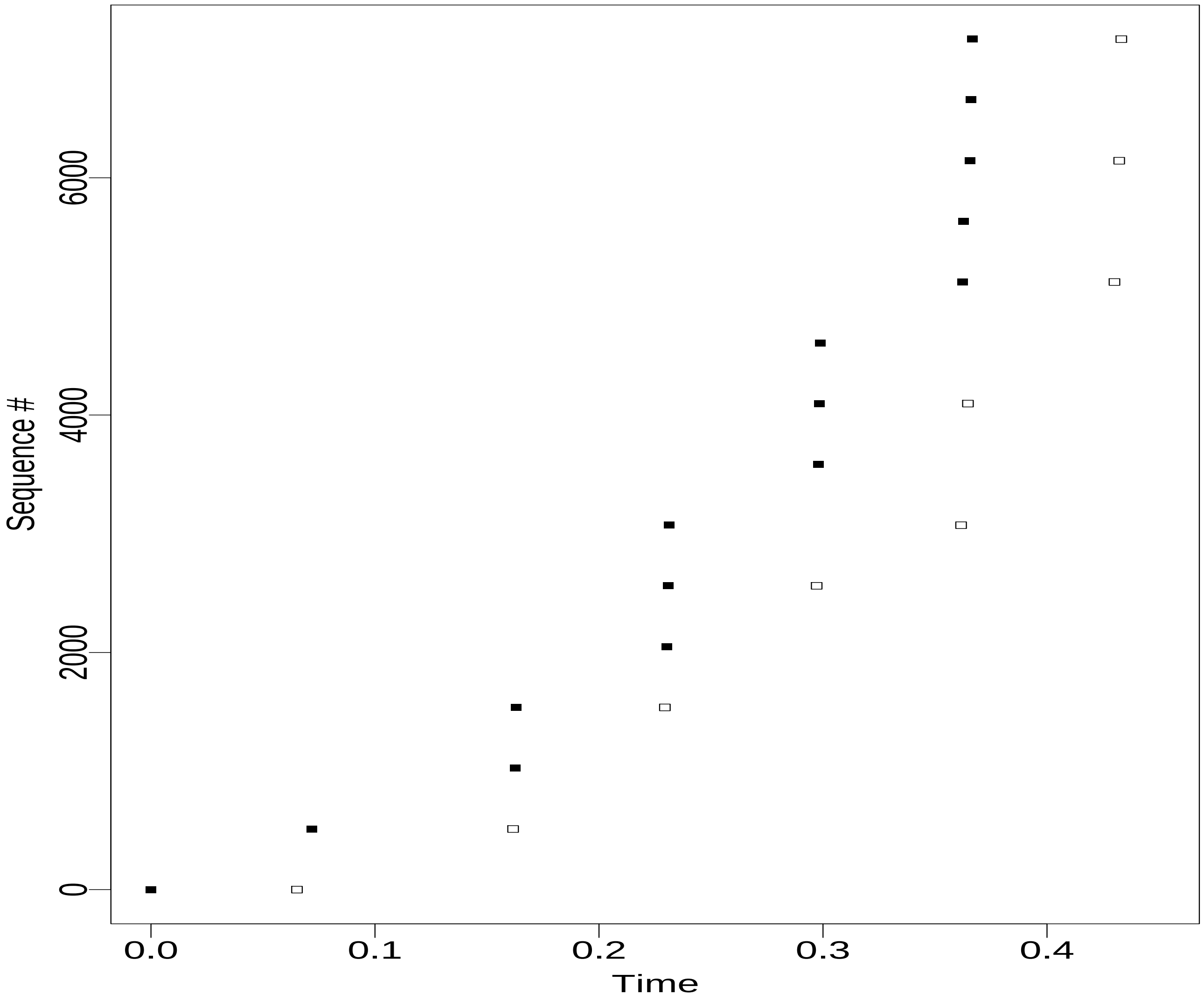
Q: If a network path has a "bottleneck" rate of ρ bytes/sec, what window value do we need to fully utilize the path?

A: "Bottleneck" means "can't go faster than.". So:

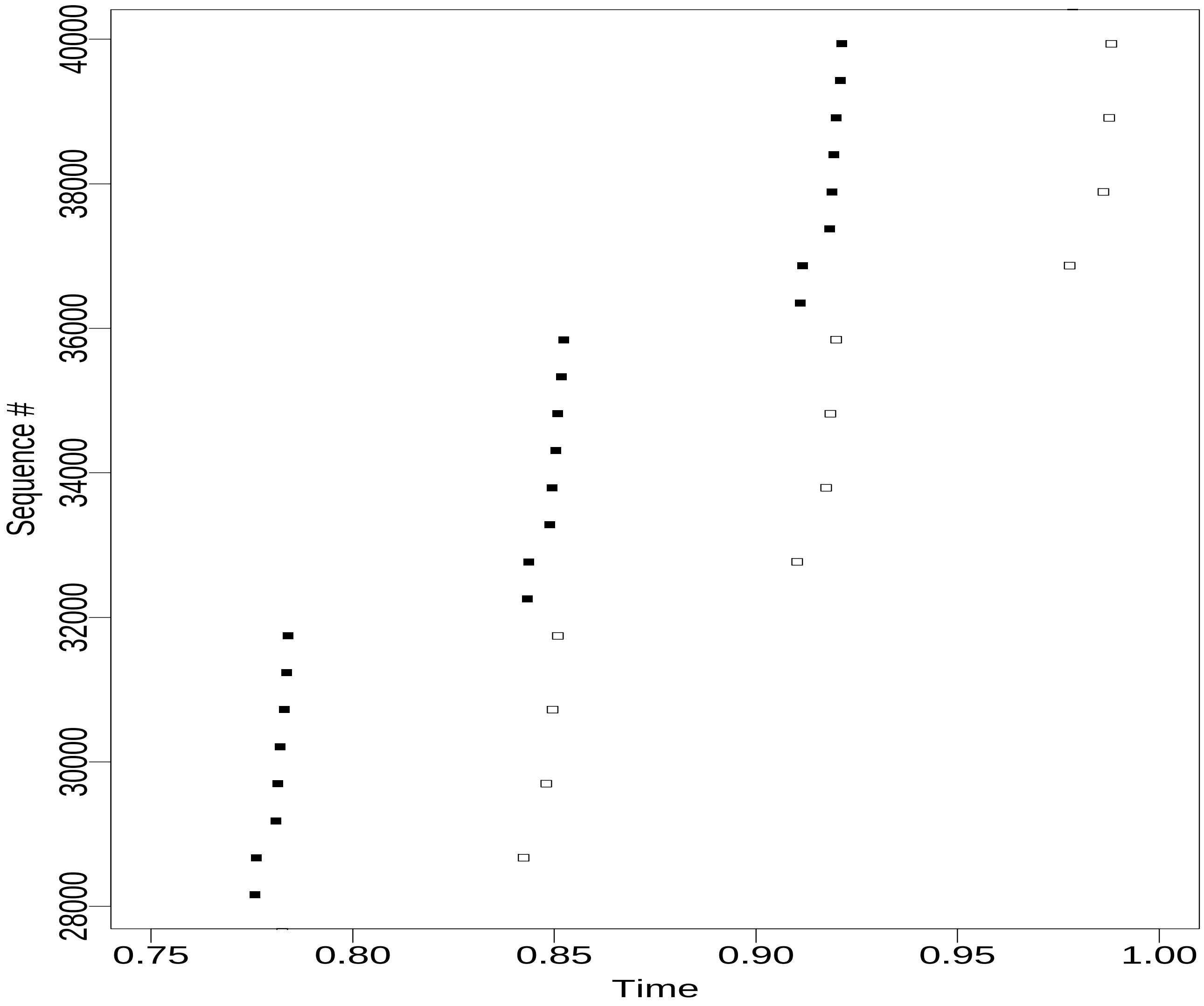
$$\begin{aligned} \text{Max Throughput} &= \frac{W}{\text{RTT}} \\ \rho &= \frac{W}{\text{RTT}} \\ W &= \rho \cdot \text{RTT}. \end{aligned}$$

$\rho \cdot$ RTT gives the <u>bandwidth-delay product</u>.

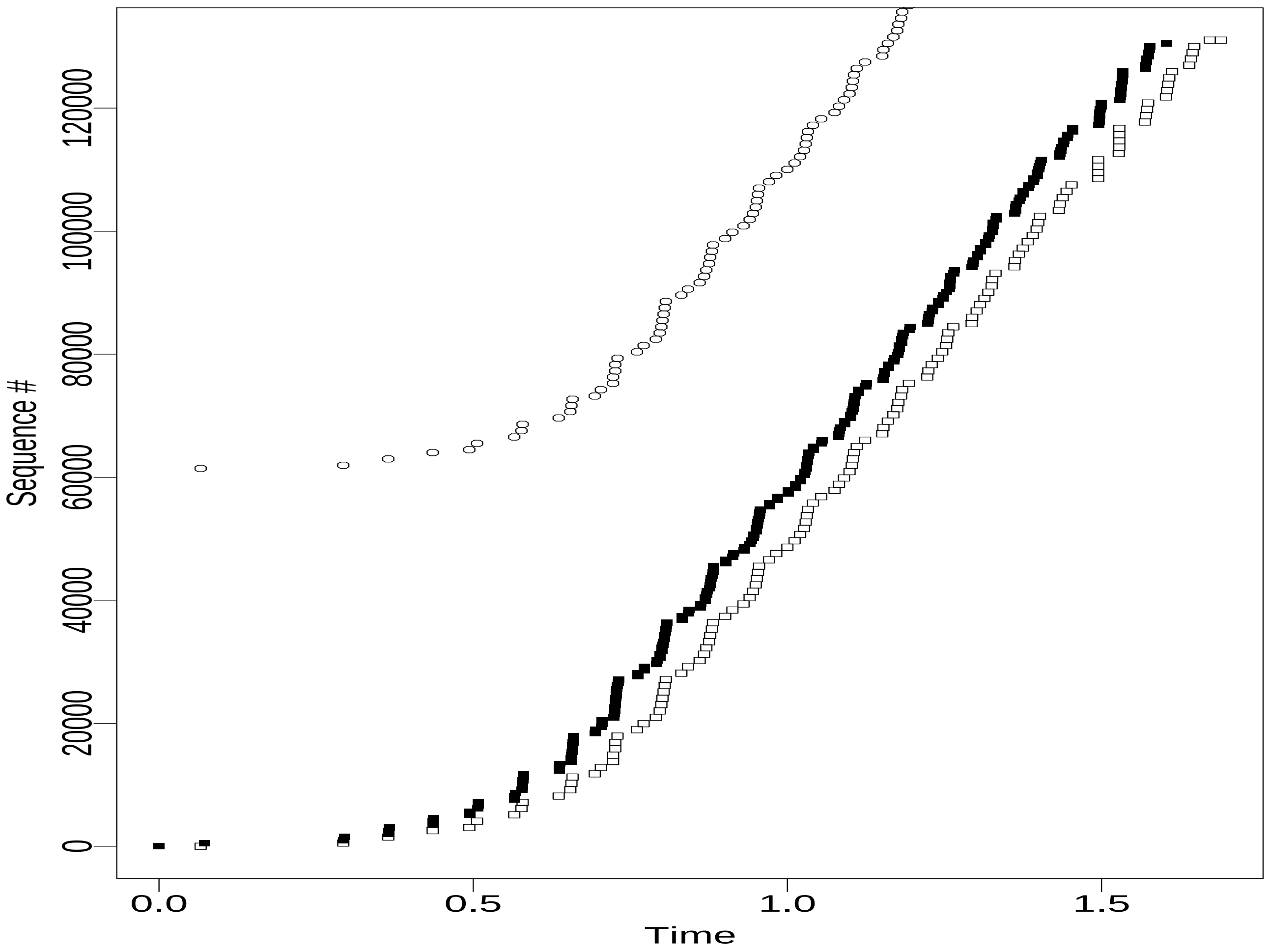
There's no point in having $W > \rho \cdot$ RTT.



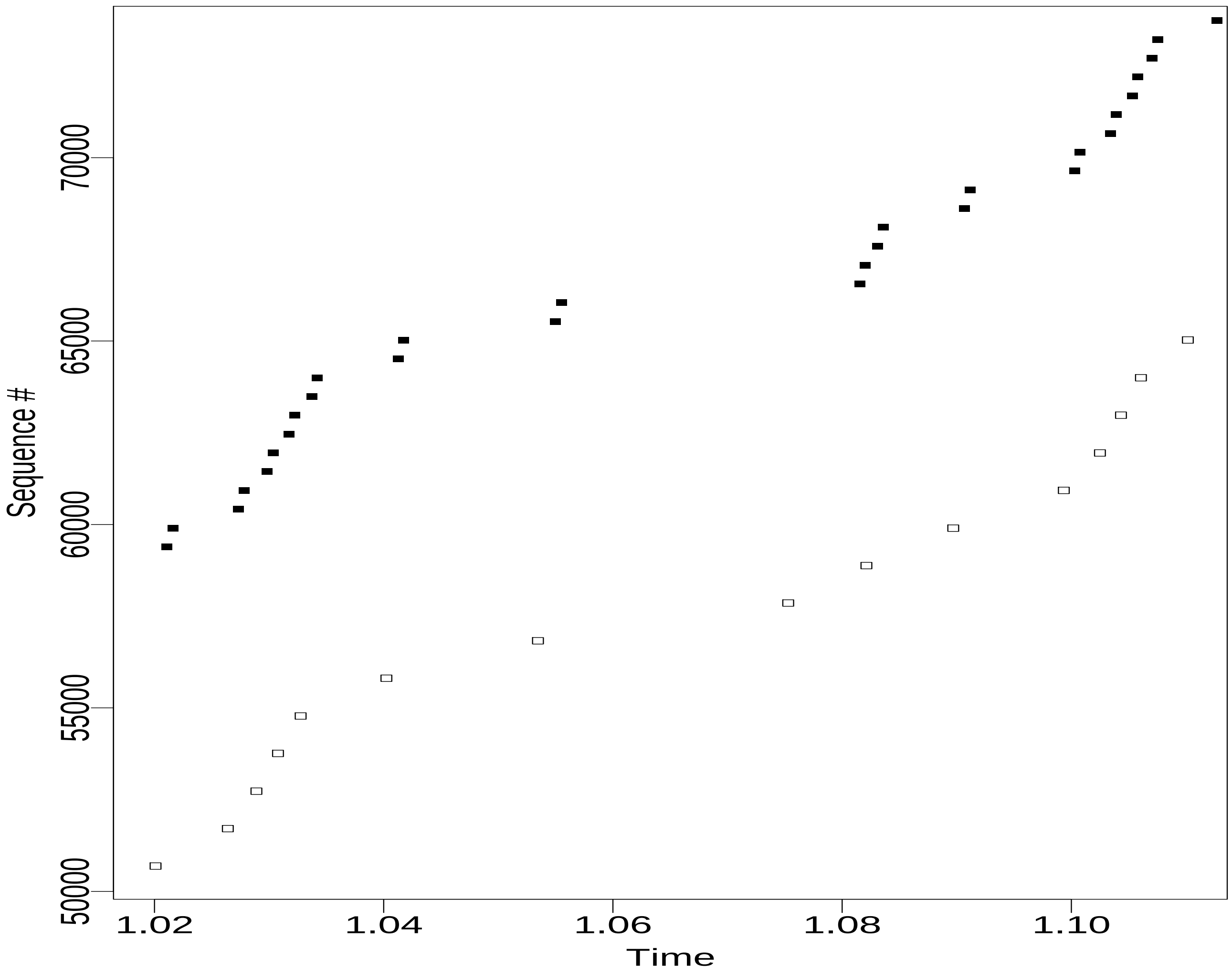
Slow start. 512 byte packets. RTT, congestion window readily apparent.



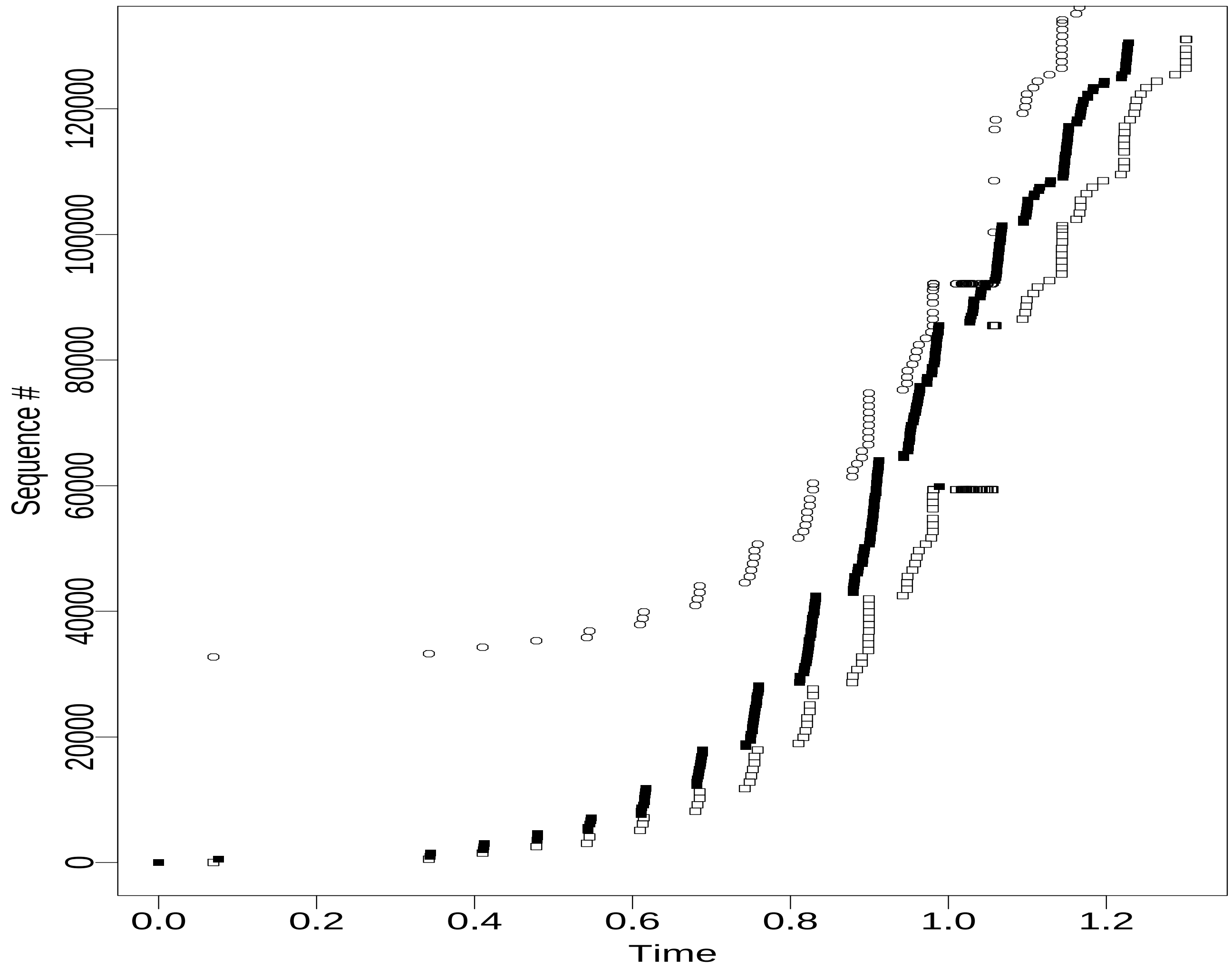
“Self clocking.”



No receiver window limitation. Throughput = 77.6 KB/sec.
Once the window opens, throughput ≈ 125 KB/sec.
Now what's the problem?

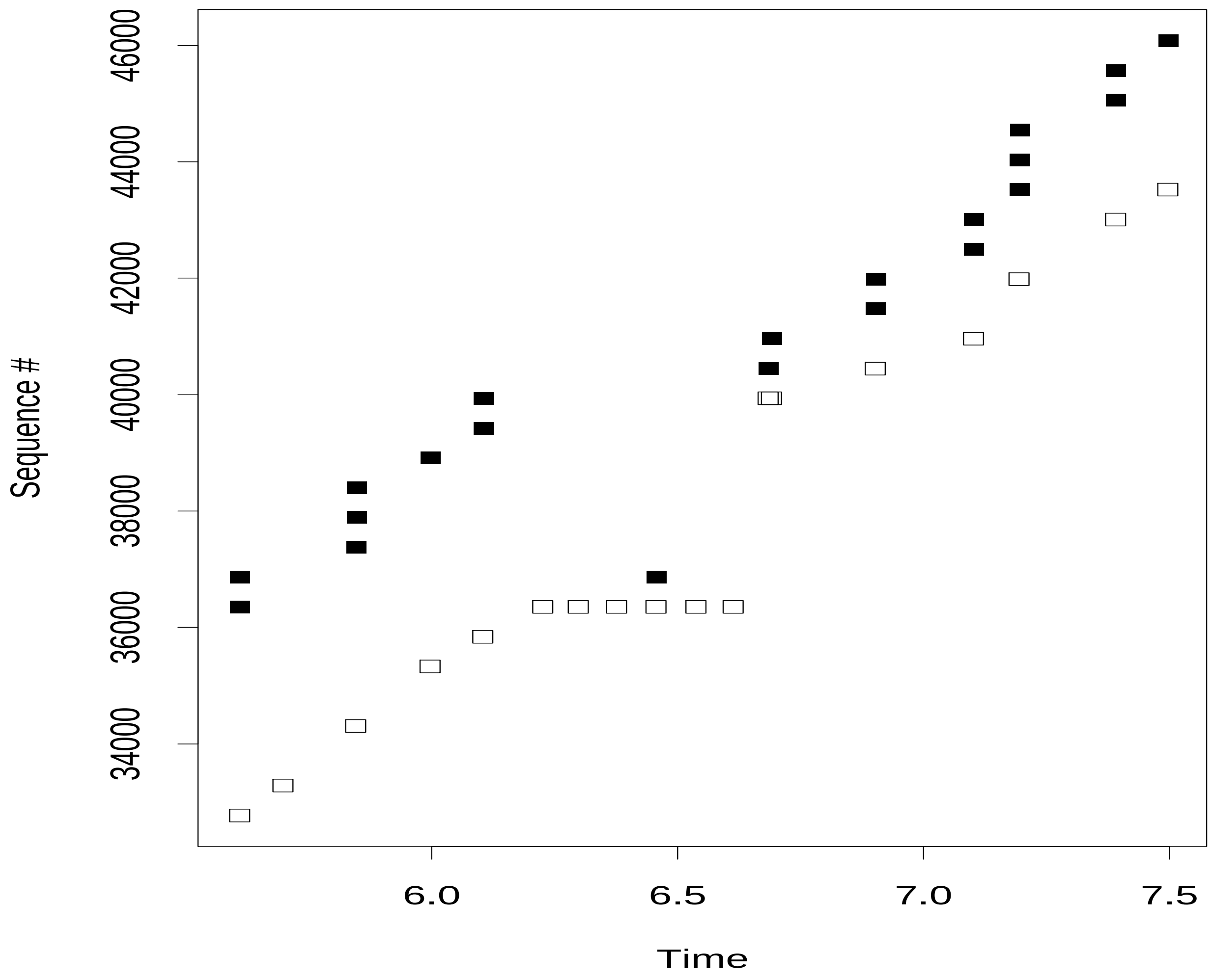


Sender window is 9.2 KB. Limits throughput to:
$9216/0.066 = 140$ KB/sec.

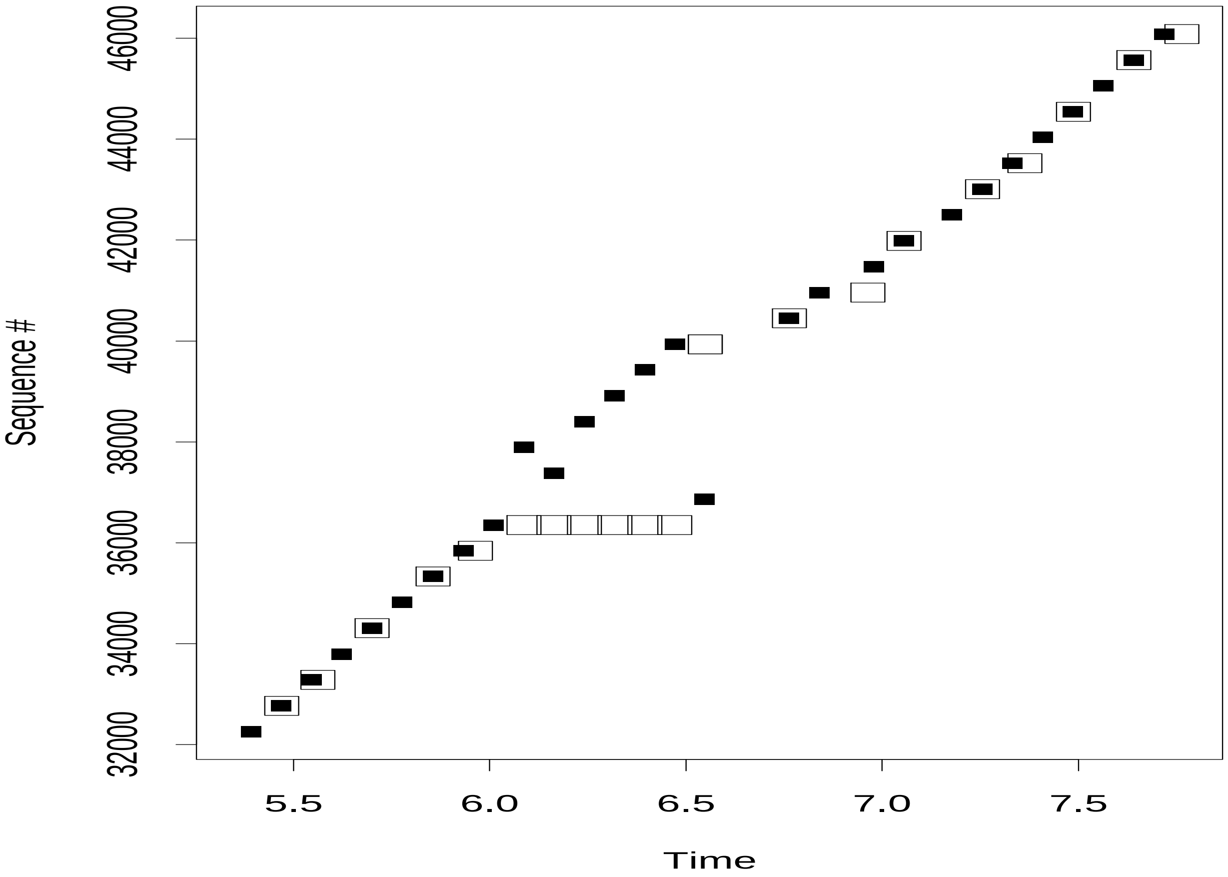


Total throughput: 100 KB/sec.

But things look complicated . . .



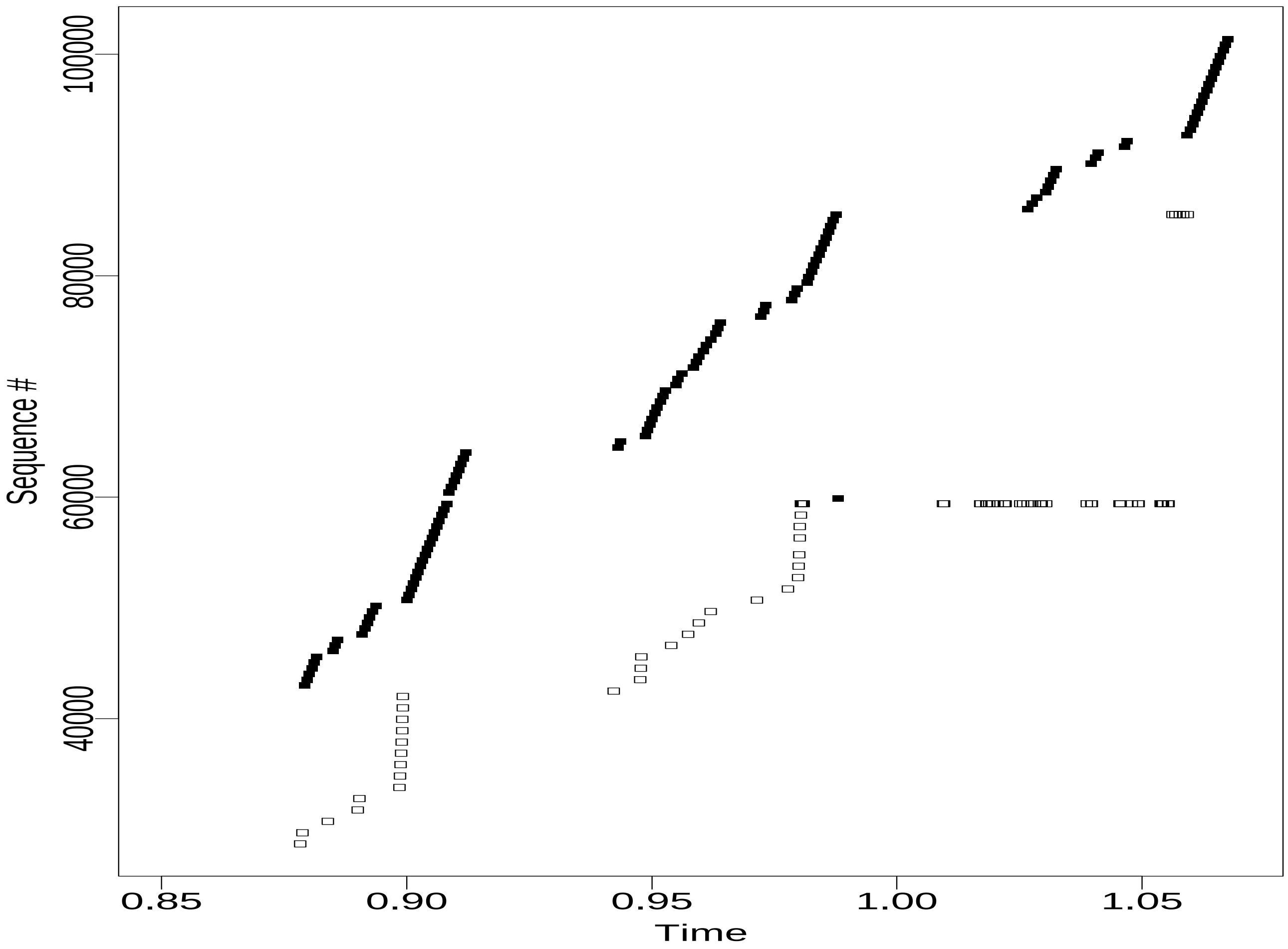
Fast retransmit after third duplicate ack for 36,865.



Same plot, receiver perspective.

Data arrives at <u>bottleneck</u> rate of 6.6 KB/sec.

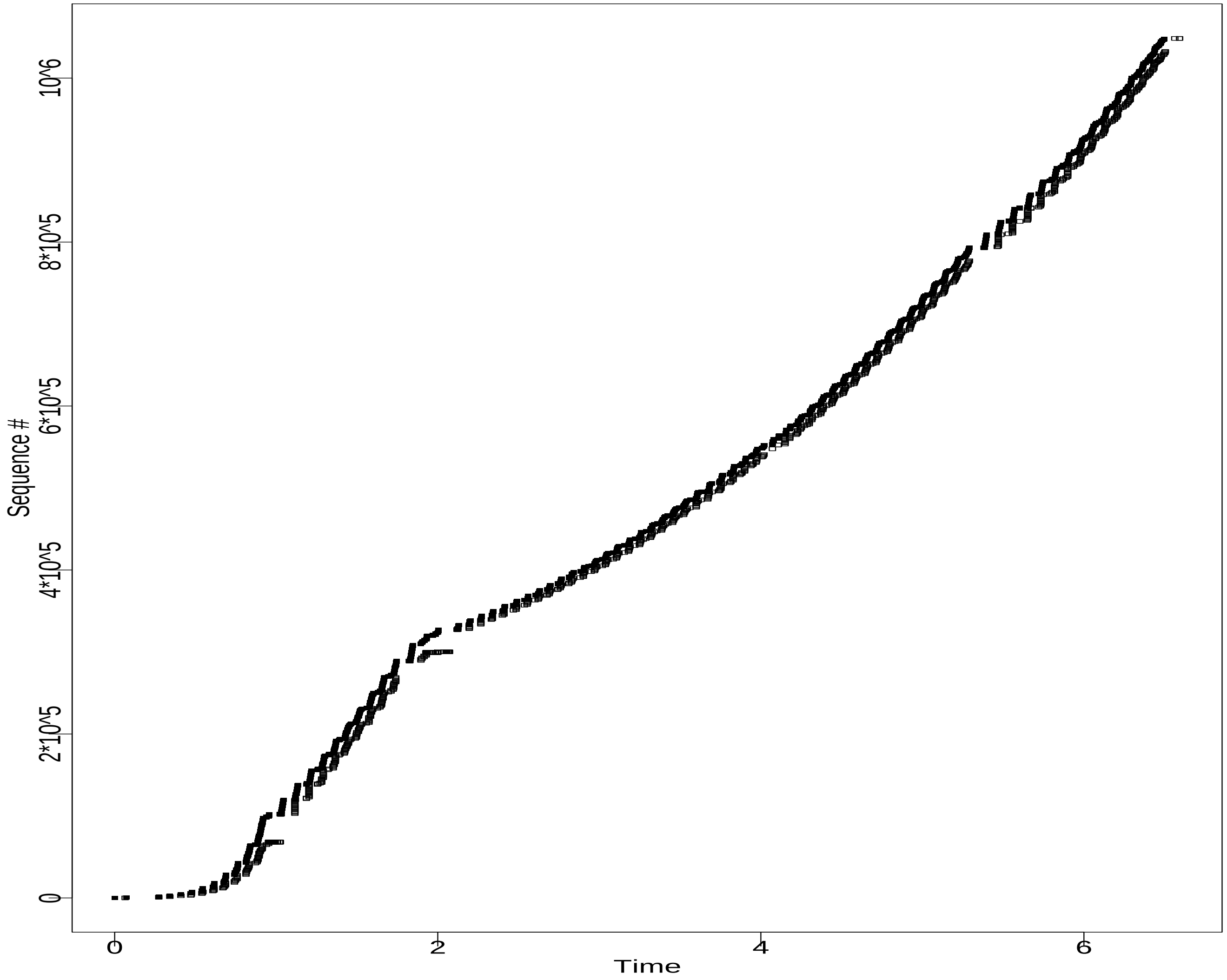
What happened just after $T = 6.0$?



Window throughput: $\approx$ 200–300 KB/sec.

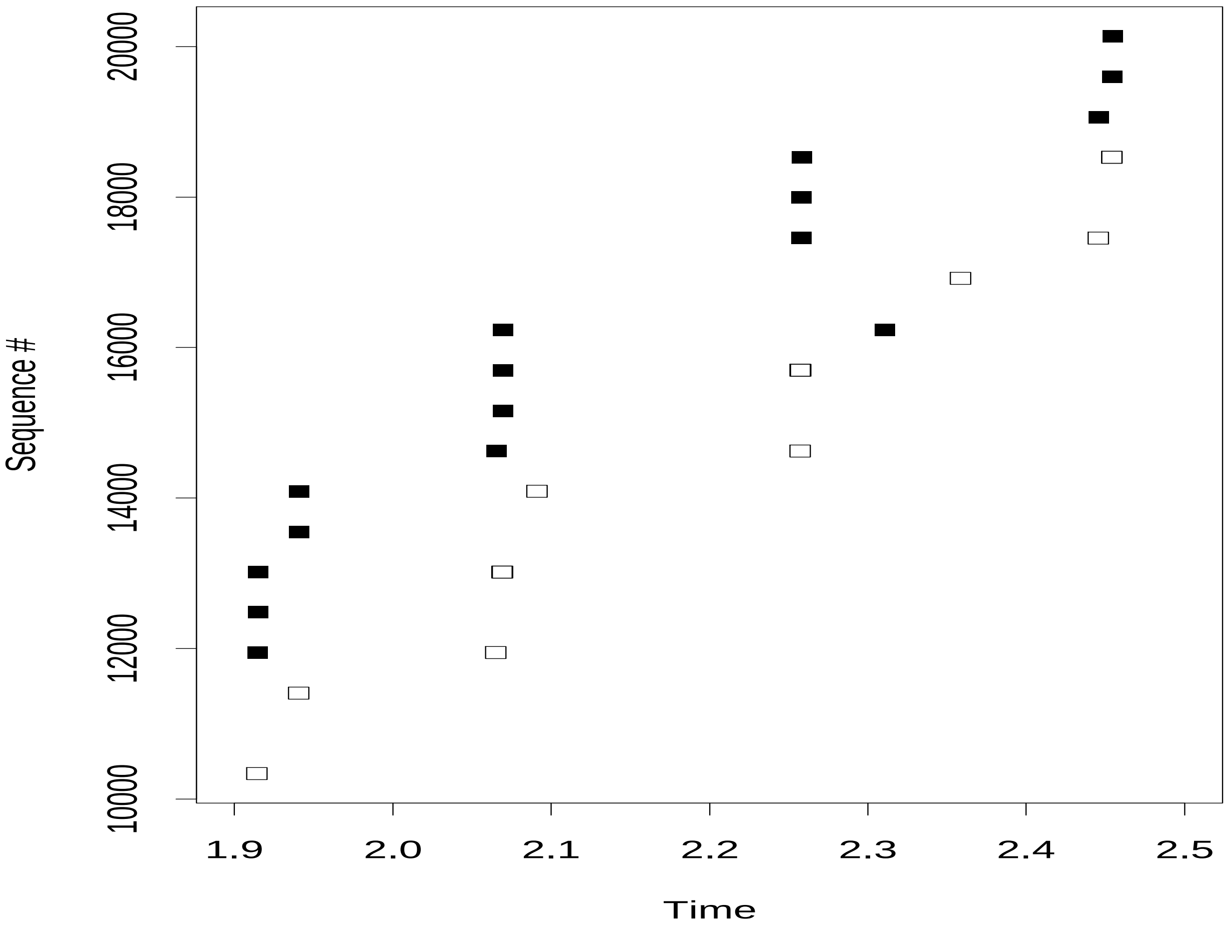
Packet lost—where was it dropped?

Serious ack compression.



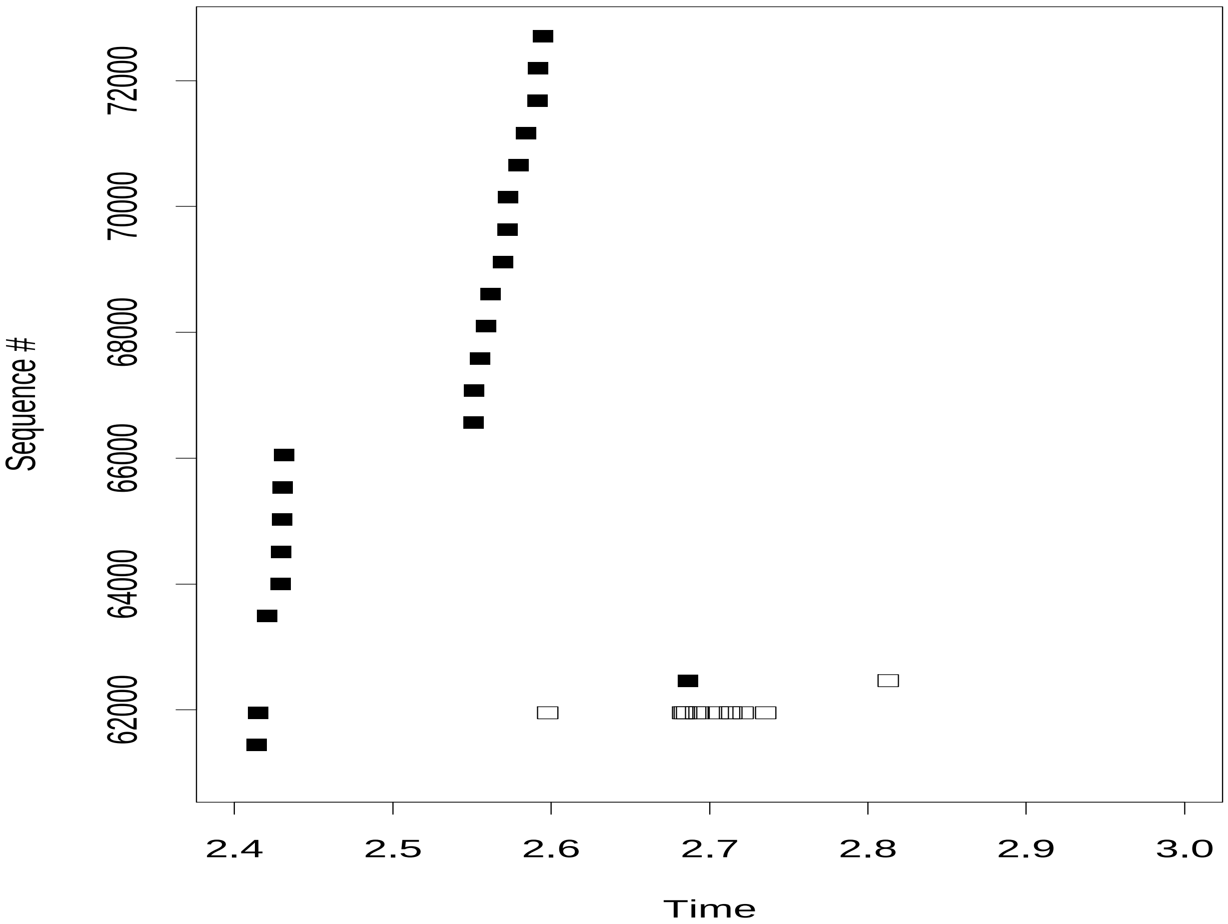
Sustained throughput: 160 KB/sec.
Prior to multiple losses: 240 KB/sec.

Packet filter measurement — Precautionary tales:



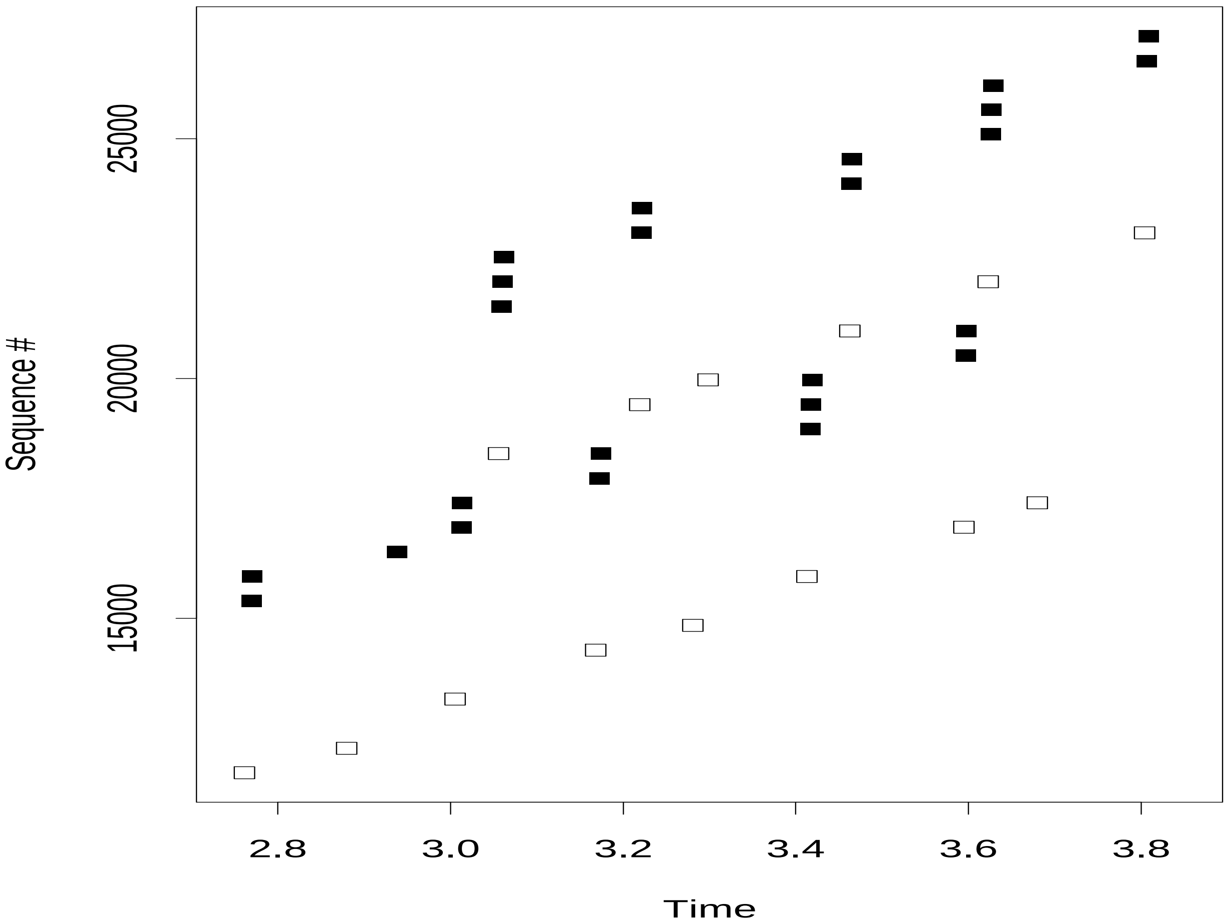
What's wrong with this picture?

Precautionary tales, con't:



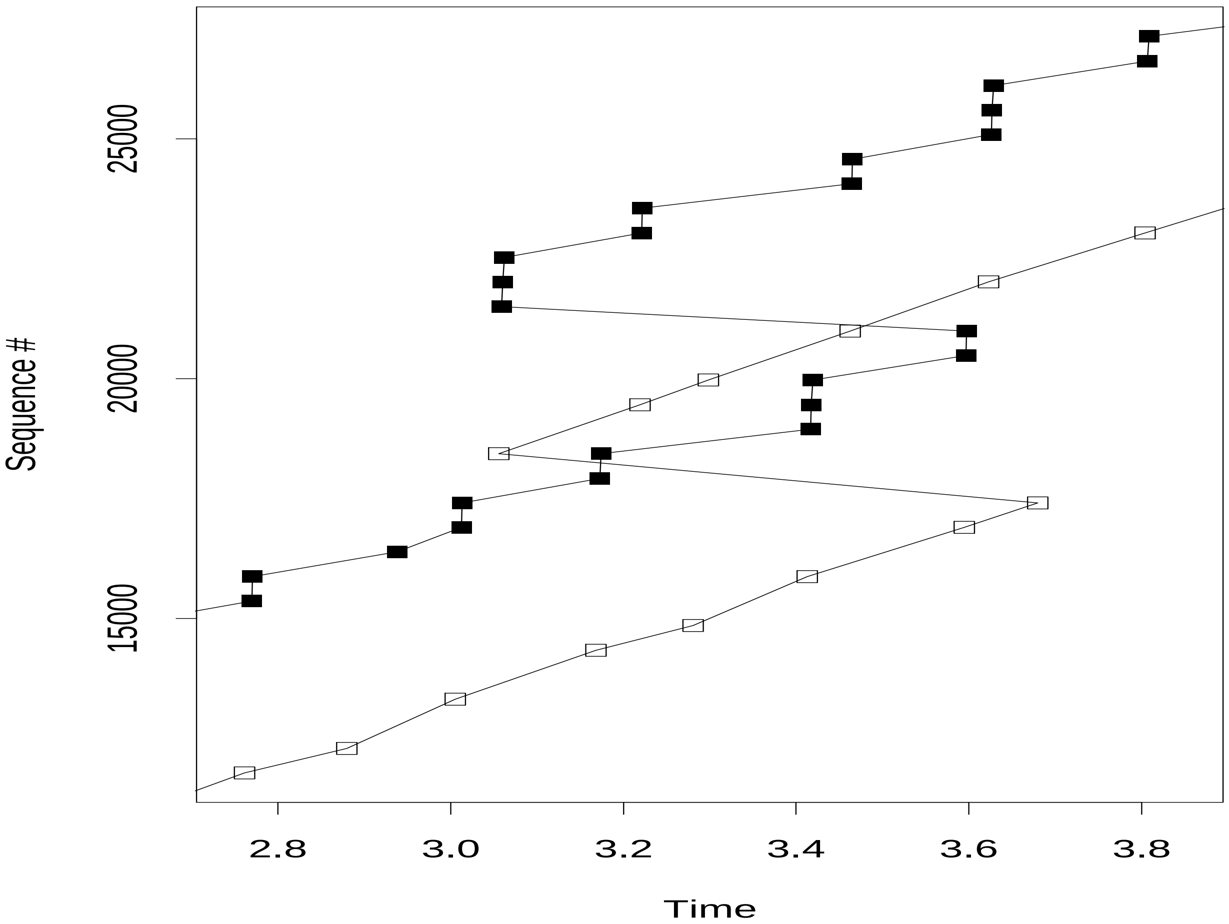
Trace recorded at sender — what's going on?

Precautionary tales, con't:

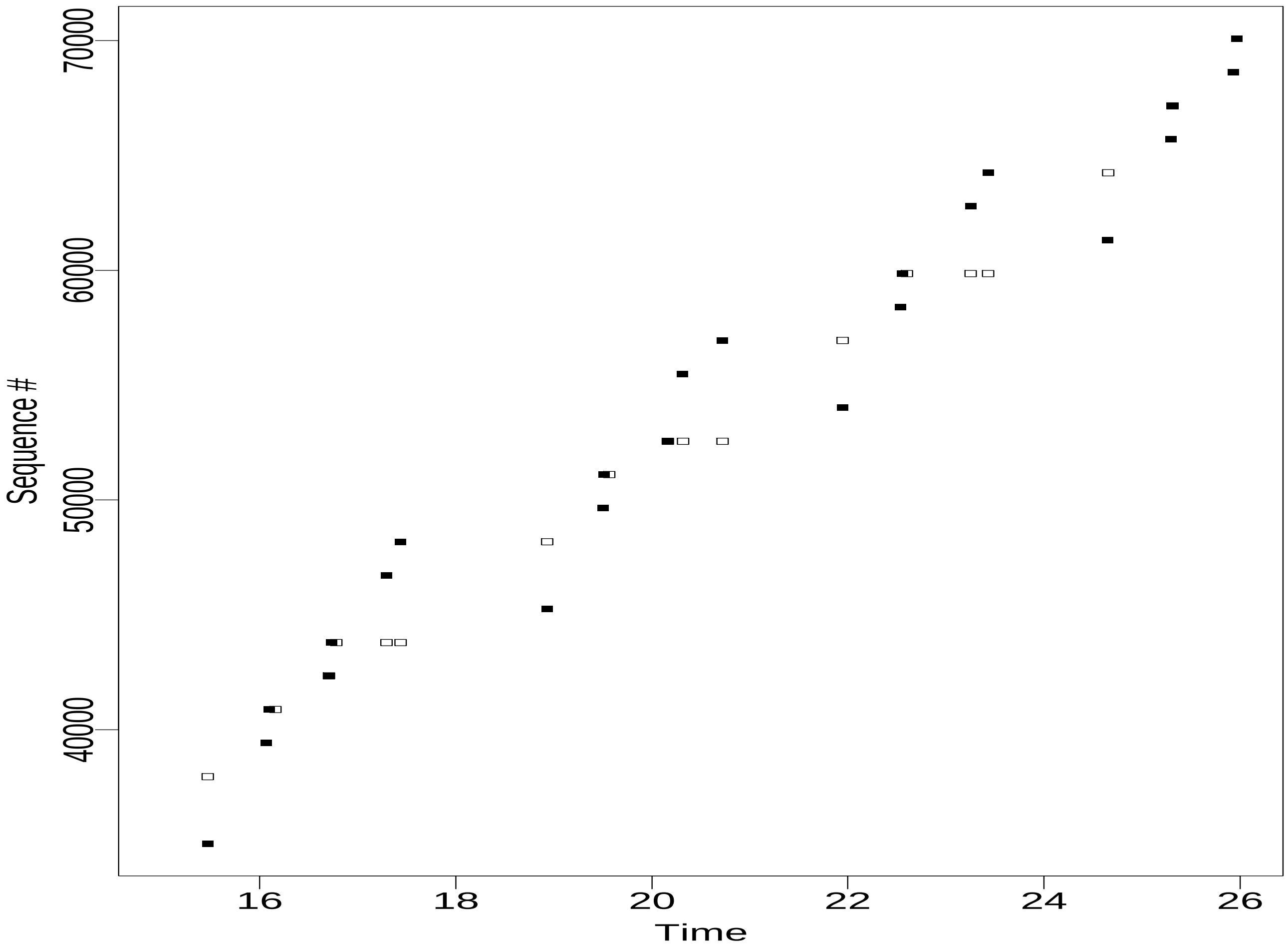


Why two separate sets of acks & packets?

Precautionary tales, con't:

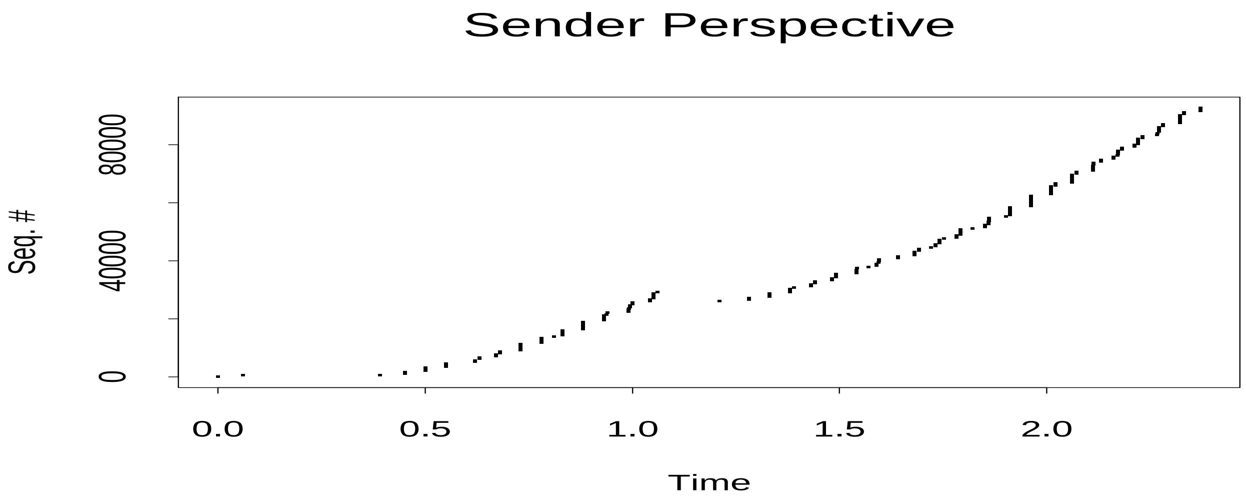


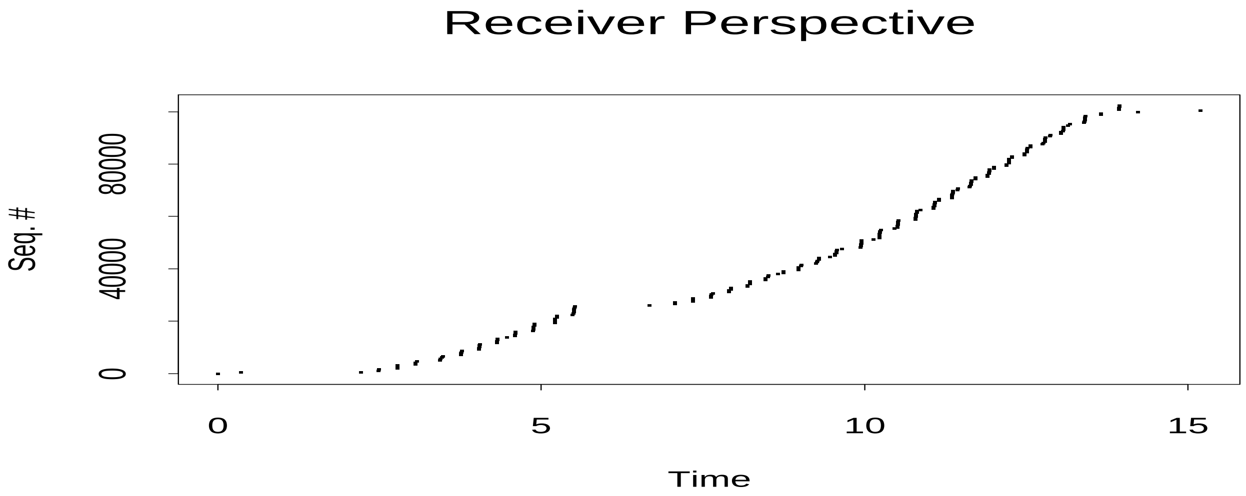
Precautionary tales, con't:



See the extra 400 msec? No!

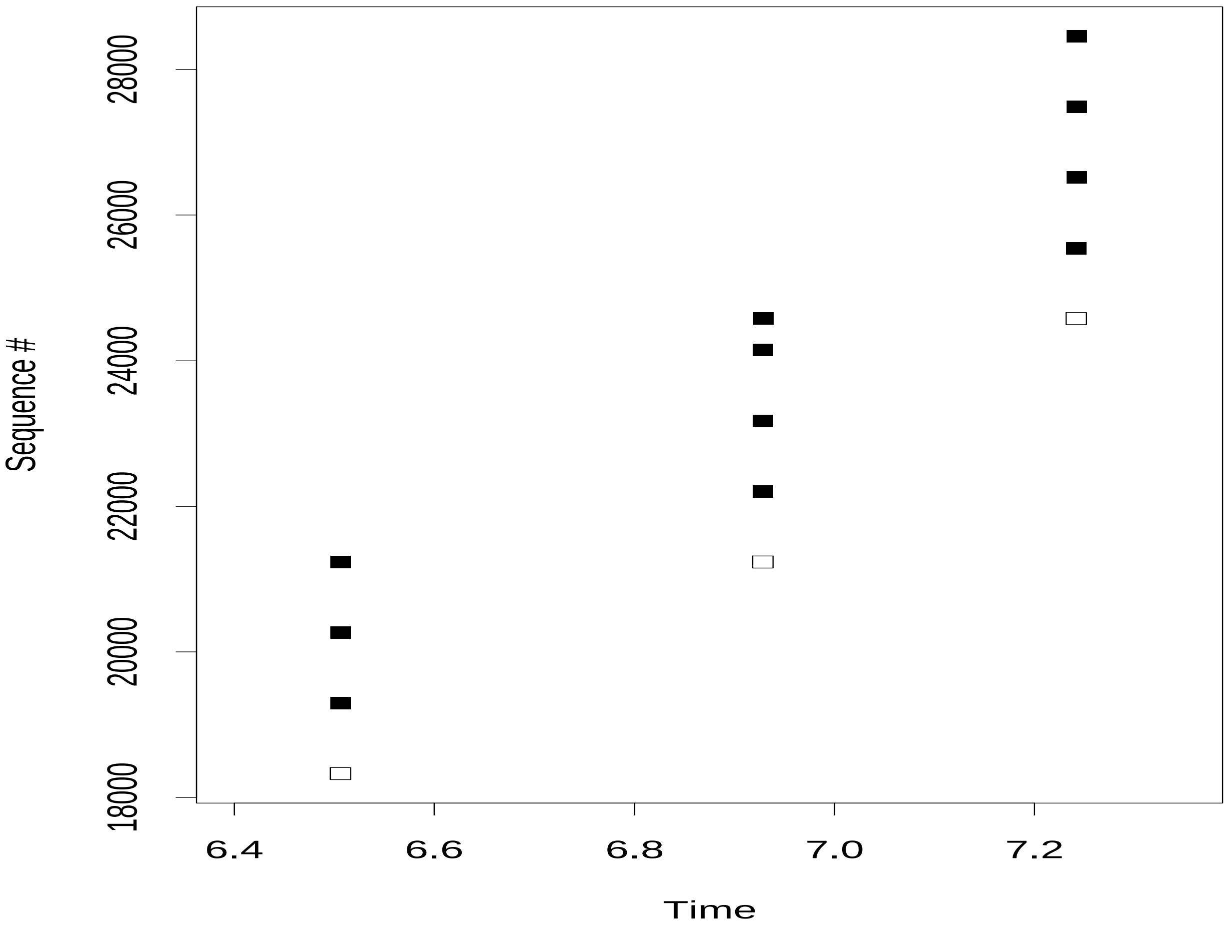
Precautionary tales, con't:





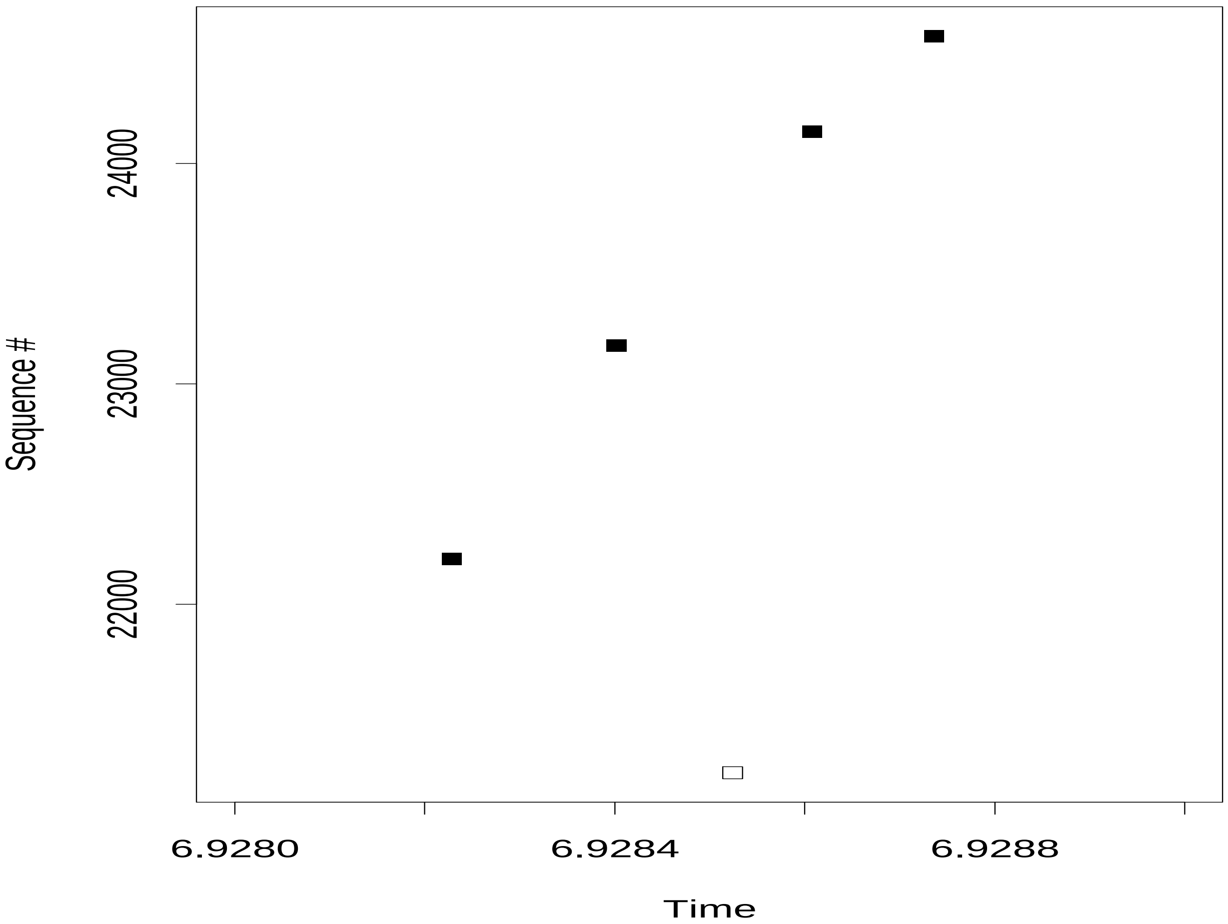
Who do you believe?

Precautionary tales, con't:



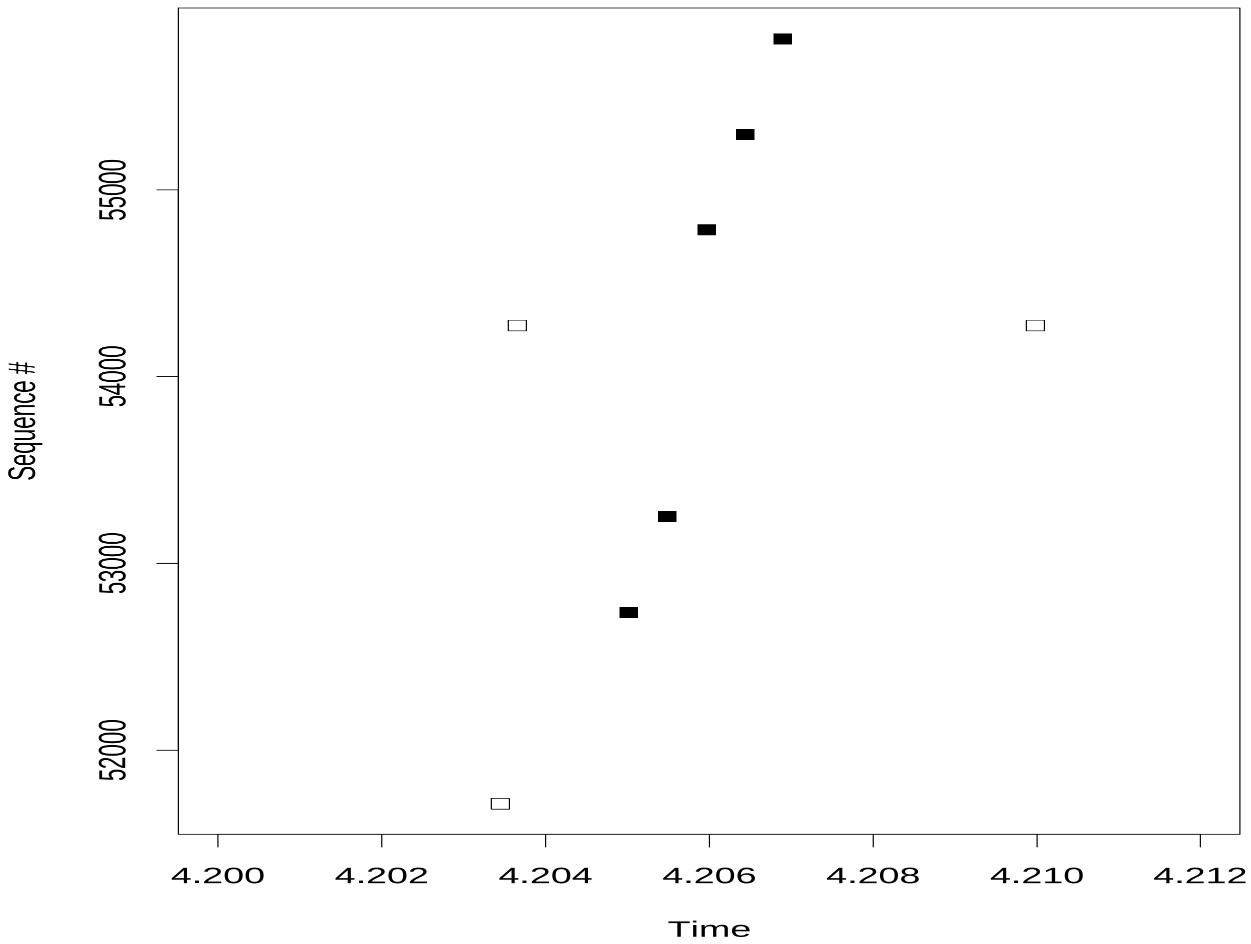
Ack at $T = 6.93$ liberates four packets . . .

Precautionary tales, con't:



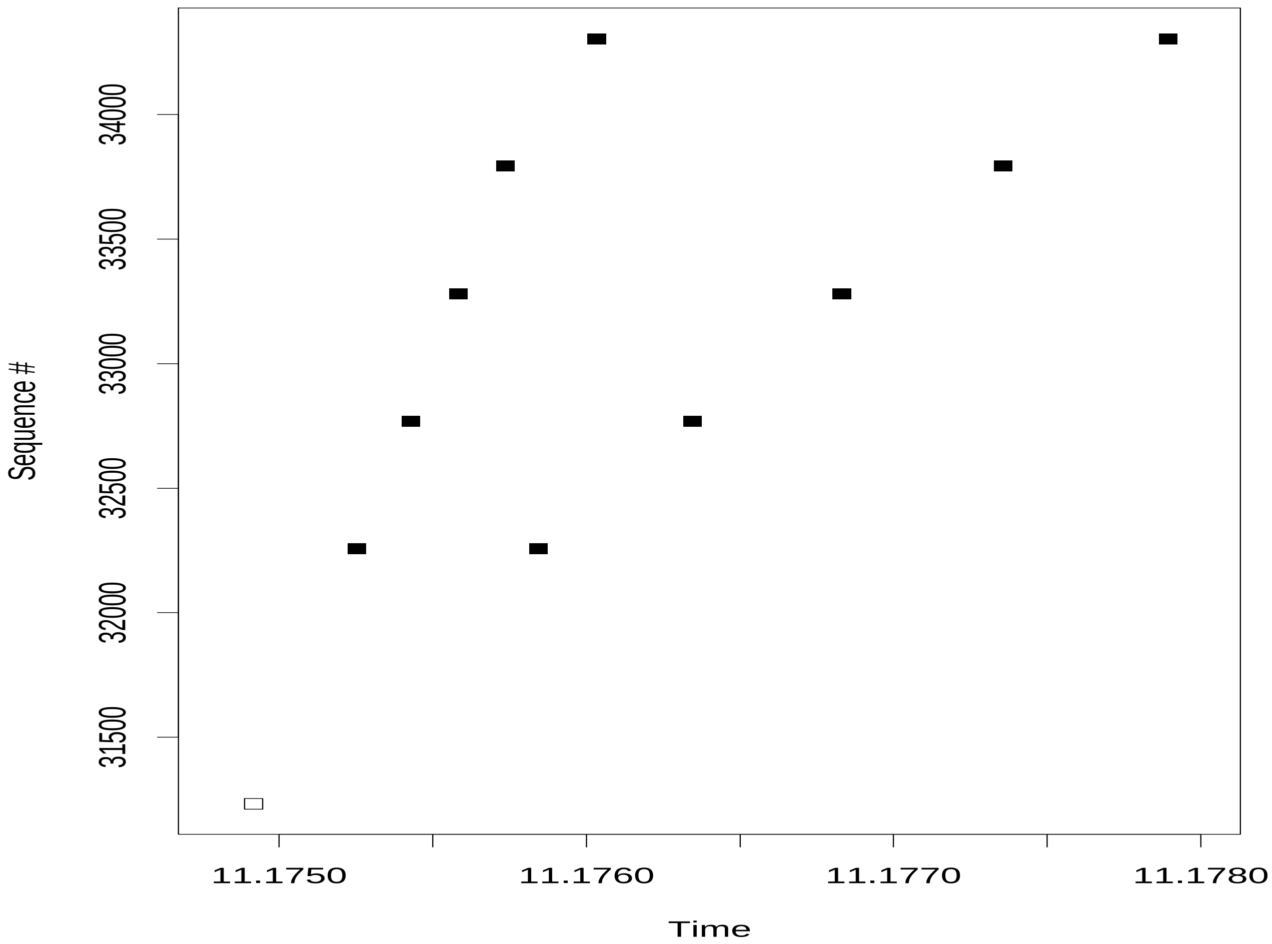
Why did two of them depart before it arrived?

Precautionary tales, con't:



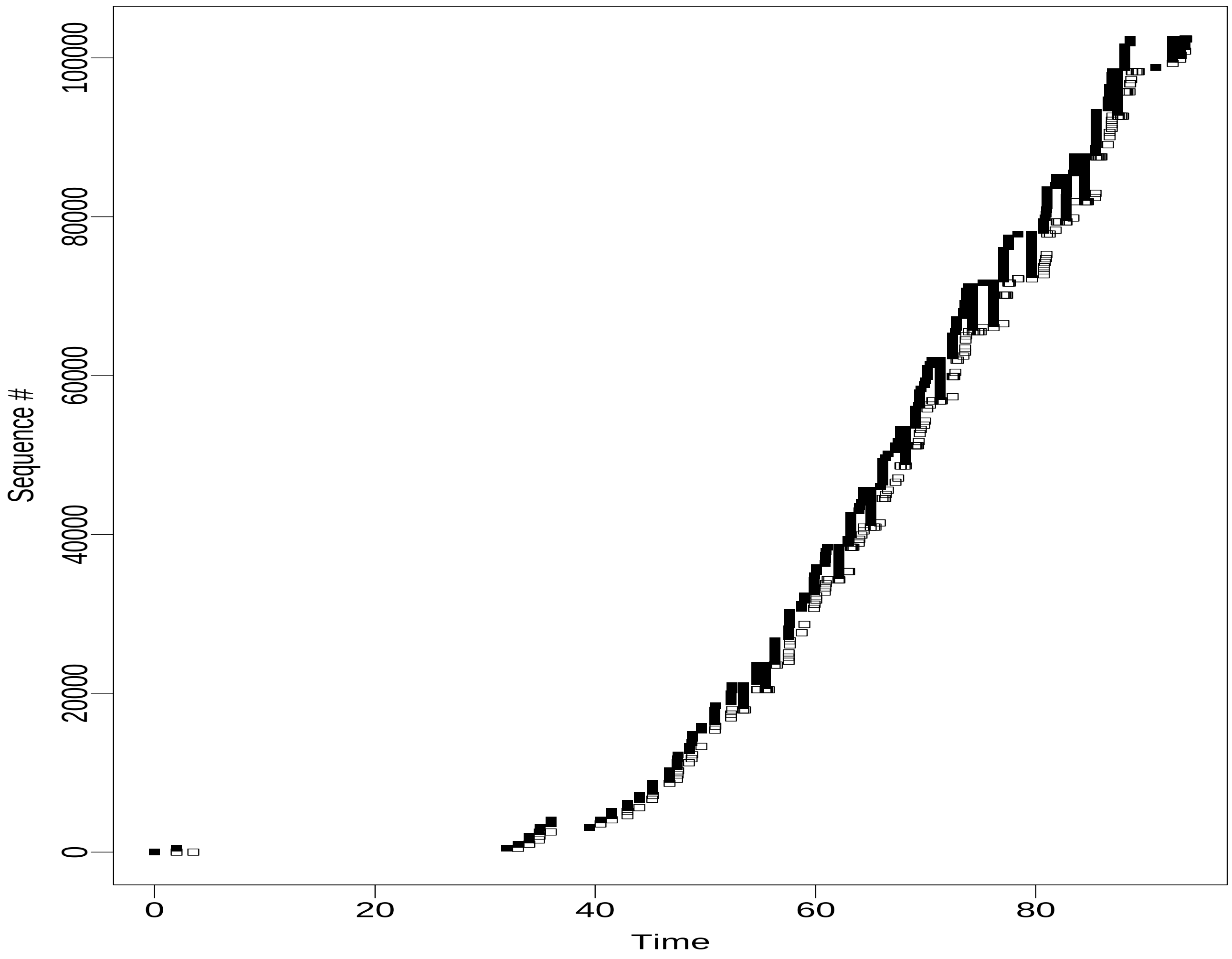
Why were the first two packets sent?

Precautionary tales, con't:



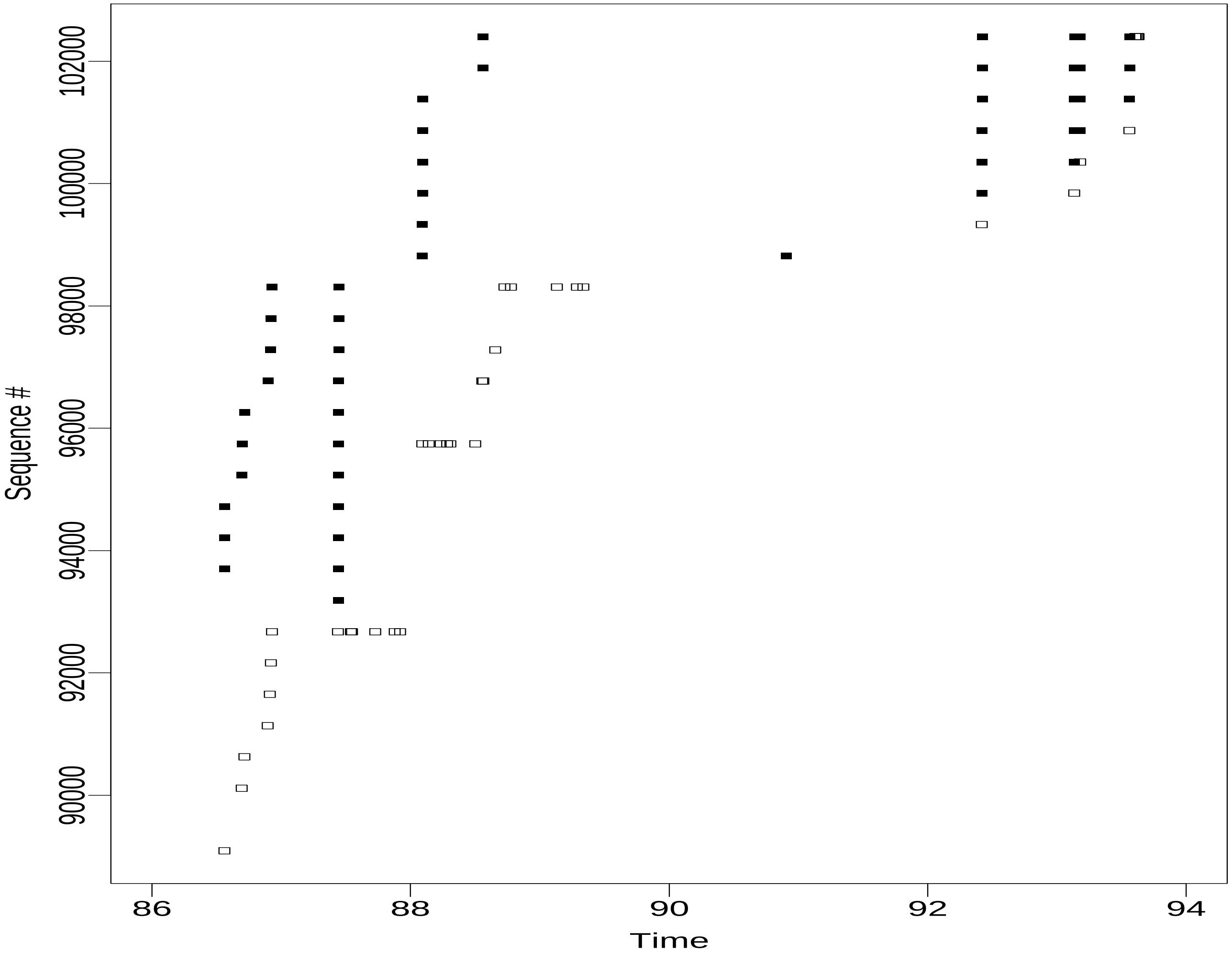
Why are packets sent twice?

Misleading throughput — Linux 1.0:



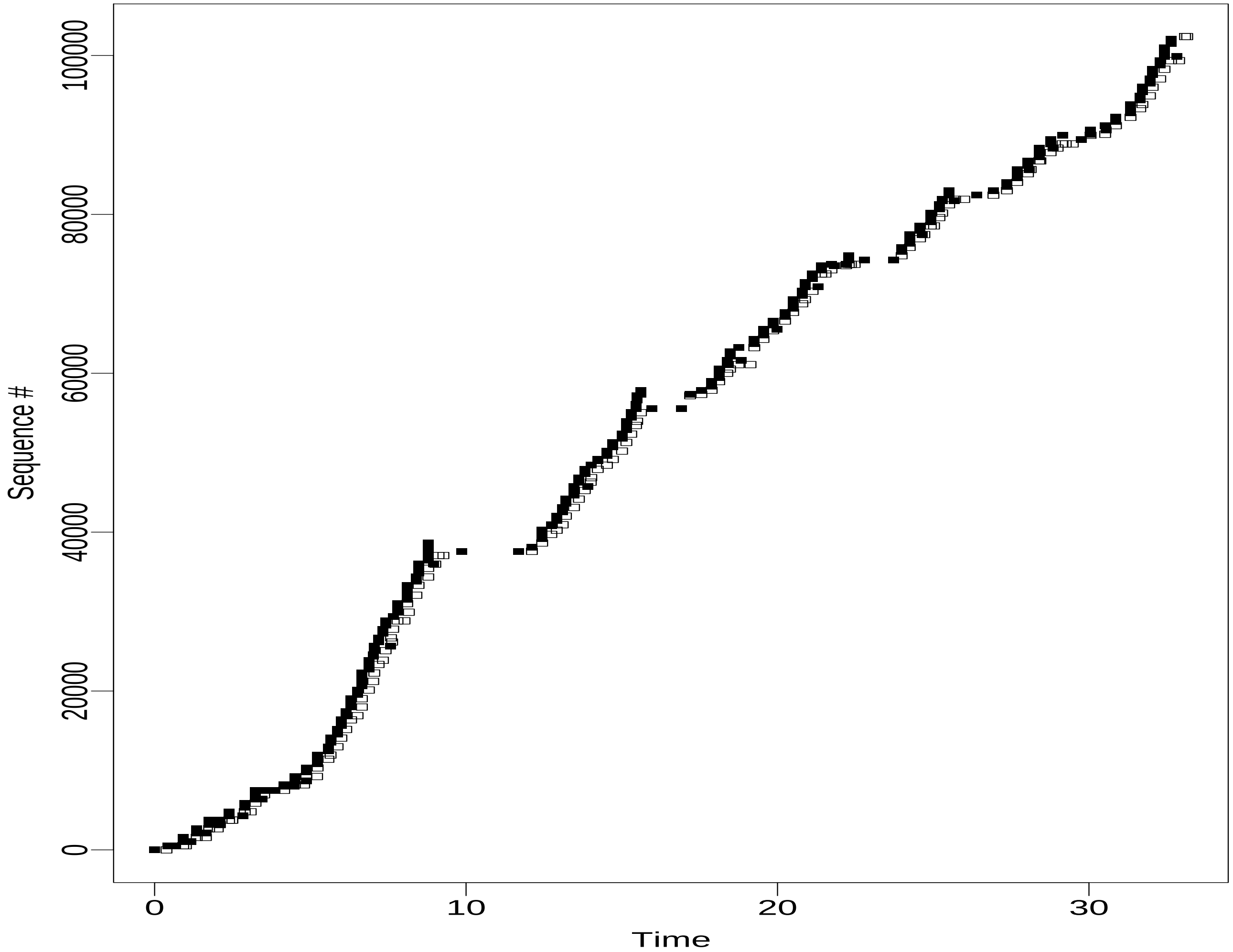
Sustained throughput $\approx$ 2 KB/S.

Misleading throughput — Linux 1.0, con't:



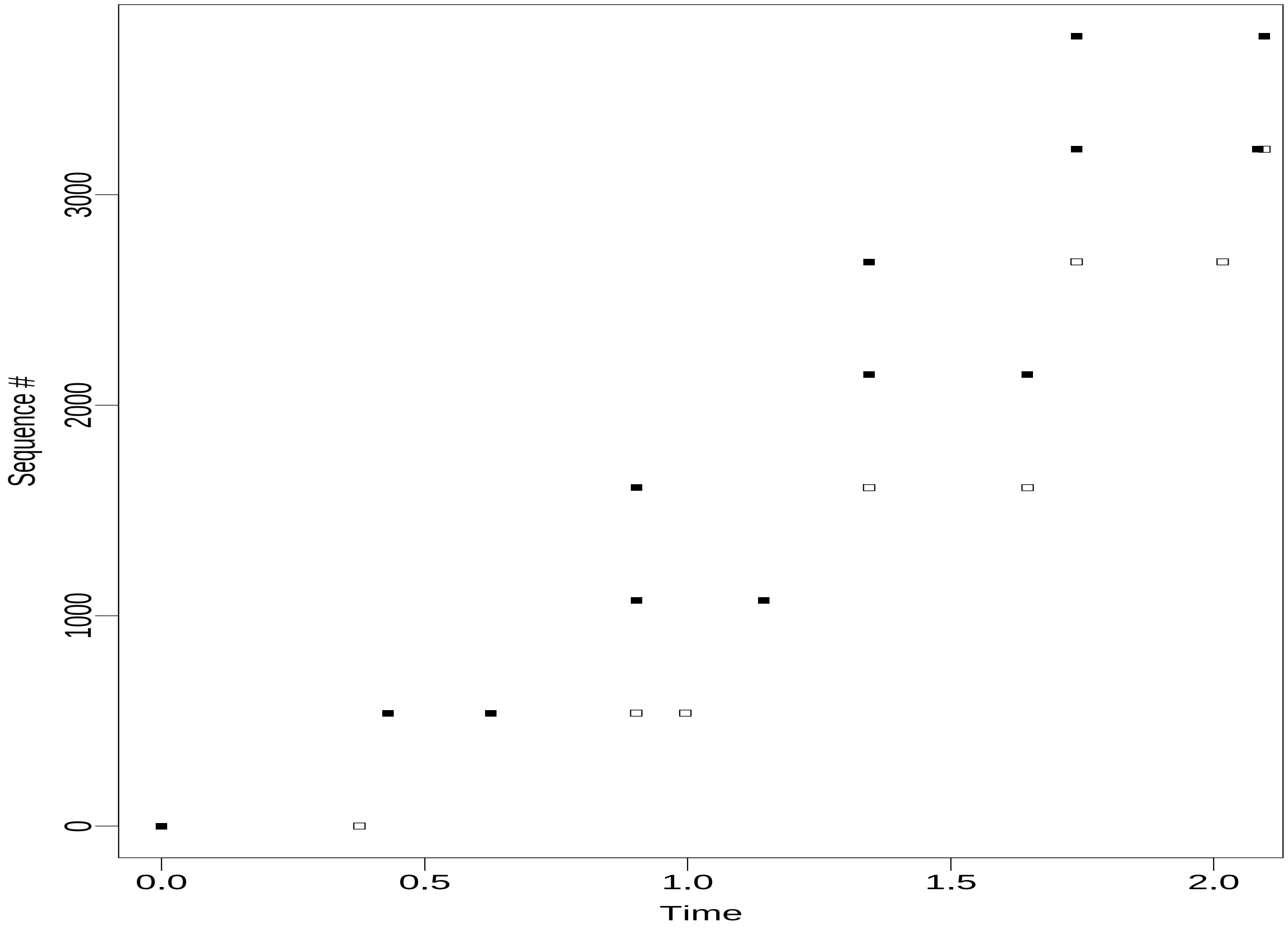
Entire window sent again on each retransmission! (FIXED.)

Misleading throughput — Solaris 2.3/2.4:



Sustained throughput $\approx$ 3 KB/S.

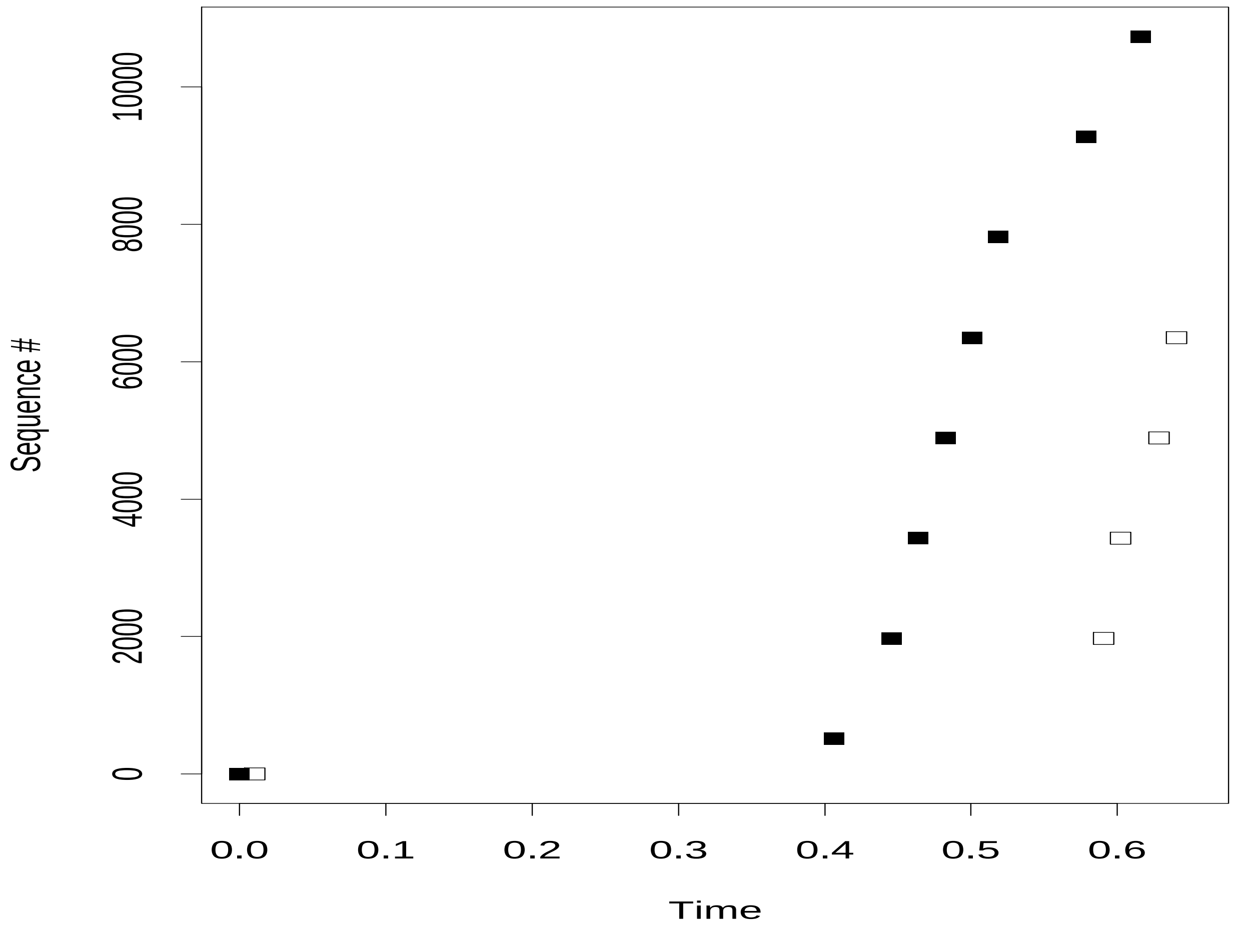
Misleading throughput — Solaris 2.3/2.4, con't:



RTT $\approx$ 350 msec, RTO $\approx$ 170 msec.

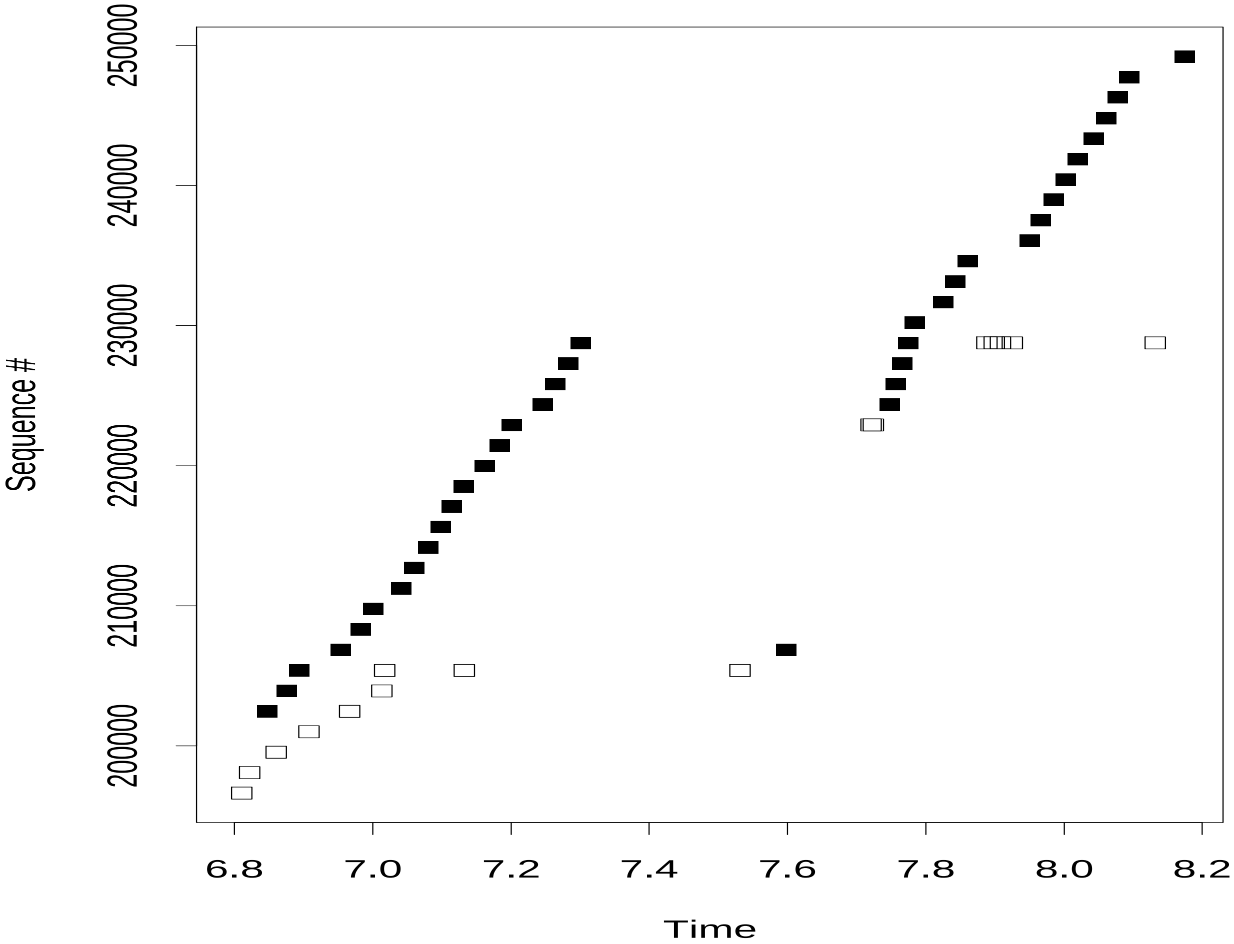
Retransmit timer too low, and not adapting! (FIXED.)

Misleading throughput — Trumpet/Winsock 3.0:



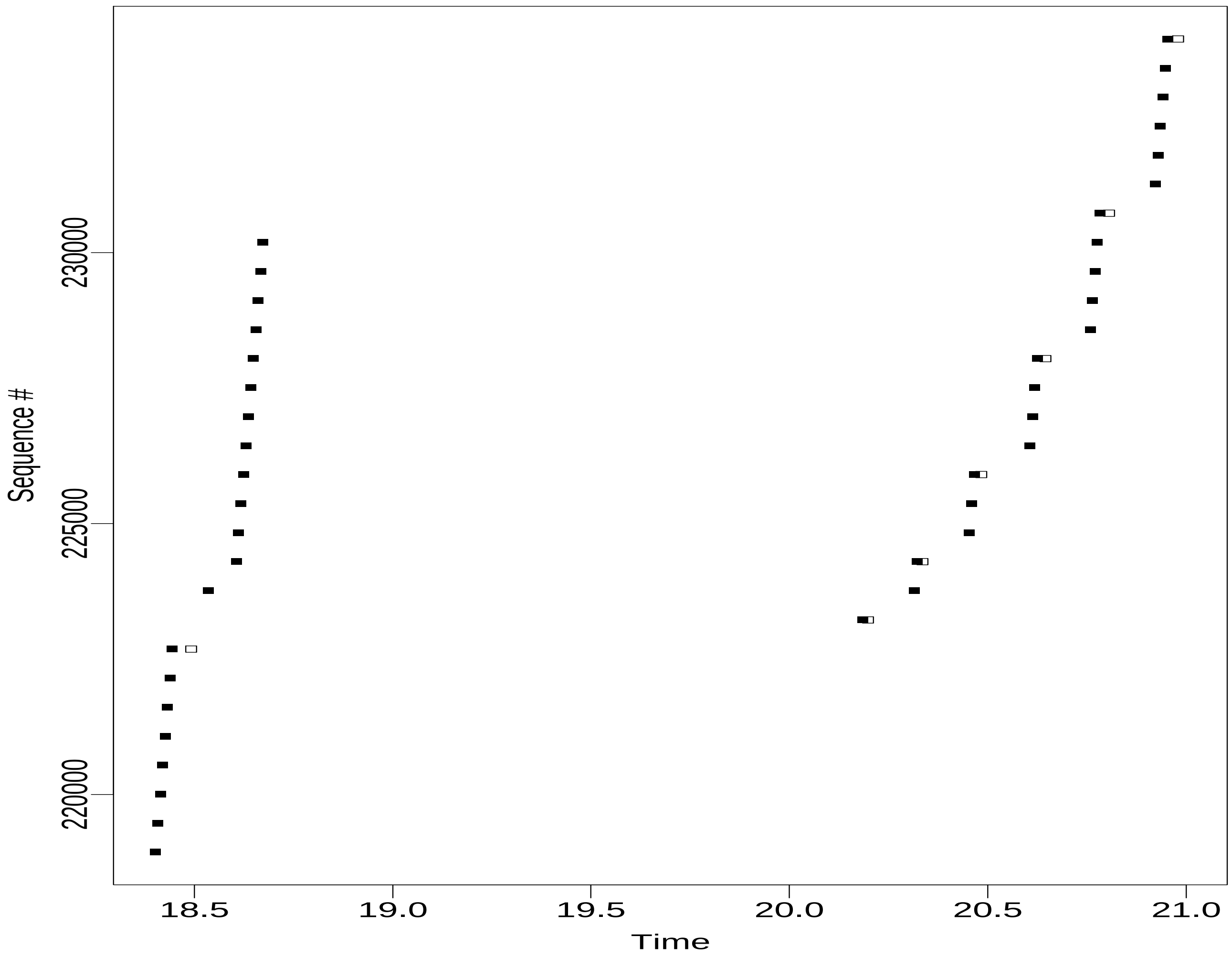
What's unusual about this connection startup?

Misleading throughput — Trumpet/Winsock 3.0, con't:



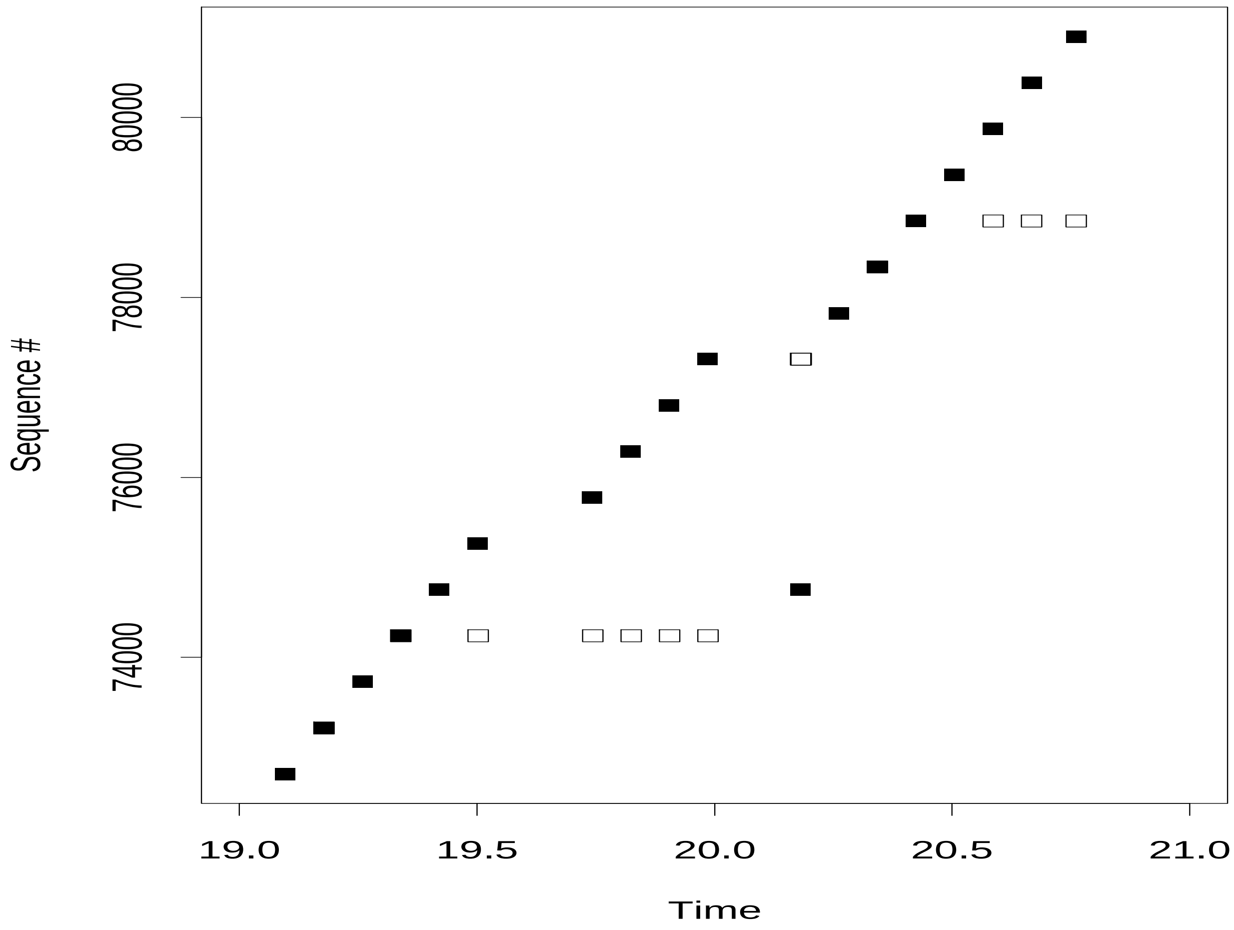
What's unusual about this retransmission sequence?

Misleading throughput — Trumpet/Winsock 3.0, con't:



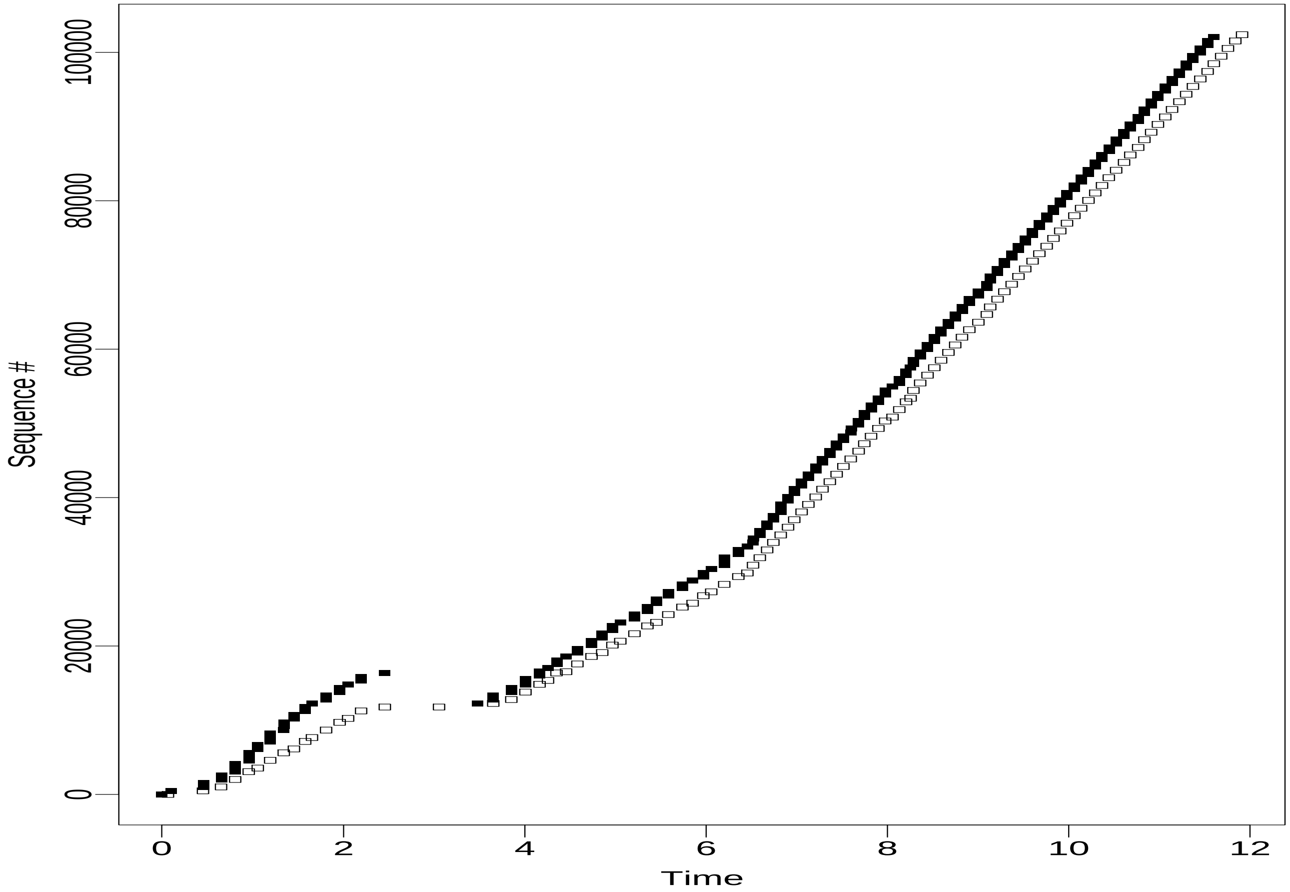
What's unusual about this trace at a Trumpet/Winsock 3.0 <u>receiver</u>?

Misleading throughput — (generic receiver):



What's going on in this receiver trace?

Misleading throughput — ISDN:



Overall throughput: 8.5 KB/sec.

Part of this is $\approx$ 6.7 KB/sec, remainder $\approx$ 13.3 KB/sec.

Which figure is correct?

Summary:

You must use a packet filter if you want to have any chance of understanding what's really happening.

Or, more generally, you need the timing of each packet — you can get this if your measurement technique allows you to craft all the packets and directly detect the responses (e.g., `treno` [MM96]).

But even if you do have all the timing, you still have to think!

Major themes:

1. The Internet is tremendously diverse, complex, and dynamic.

2. Because it's so diverse, we are desperate for invariants and parsimonious models.

3. Sound measurement requires discipline and a willingness to consider that you might be wrong.

4. Similarly, assessing someone else's work requires skepticism.

Major themes, con't:

5. Whenever possible, calibrate what you measure.

6. To do so, testing for self-consistency is extremely powerful.

7. Network traffic is highly structured: it pays to know as much about the structure as possible.

8. You don't know anything about what's going on unless you measure!