

# The S-Net System for Internet Packet Streams: Strategies for Stream Analysis and System Architecture

Jin Cao, William S. Cleveland, and Don X. Sun  
Statistics and Data Mining Research  
Bell Labs, Murray Hill, NJ  
cao@bell-labs.com, wsc@bell-labs.com, dxsun@optonline.com

**KEYWORDS:** Streaming data; Internet traffic modeling; Data mining; Time series; Long-range dependence; Data visualization; Trellis display; Lattice graphics; Statistical multiplexing; Packet traces; Network simulation

## Abstract

The traffic on an Internet link is a packet stream: packets of varying sizes arriving for transmission on the link. Each packet has an arrival time, and contained within the packet are headers that carry many critical variables. Packet traces, which consist of captured headers and measurements of the arrival times, convey substantial information about the Internet — security, usage, network performance, and the performance of engineering protocols. This paper discusses strategies for the analysis of very large databases of packet traces, and the architecture of a software system that facilitates the use of these strategies. The system has a pipeline: (1) raw packet traces; (2) a database with objects tailored to ensuing analyses; and (3) an environment with tools for data analysis: statistical methods, model fitting, and visualization. The pipeline addresses the full set of tasks in the study of packet streams, from the initial processing of raw packet traces to the final output, often a visual display. S-Net — an extensible, open-source software implementation of this architecture — is based on the R implementation of the S language for graphics and data analysis, and has been developed on Linux.

## 1 Introduction

### 1.1 Internet Technology

The Internet is a computer network with links and nodes covering the globe. The links are communication links — fiber, cable, or telephone wires — each connecting two nodes. Each link has a speed, or bandwidth, measured in units such as megabits per second (mpbs) or gigabits per second (gbps). The end nodes are host computers. The Internet exists to transfer information from one host to another. The transferred information can be files with text, images, voice, or video; the latter two can be real-time. The internal nodes are routers that forward information to another router or to a host. So we can think of the Internet as a large network in which pairs of hosts establish connections to transfer files between them. When one host in the pair, the source, sends information to the other, the destination, the information travels along a path across the Internet — from the source over a first link to a first router, from there through a succession of routers and links, and finally over a link from a last router to a destination host. The Internet protocol running on the hosts that enables this routing is IP (Internet Protocol).

A transferred file is broken up into packets of 1460 bytes or less, and the packets are sent across the path. To carry this out, the two hosts establish a connection: software running on each of the hosts managing the sending and receiving of the packets on the two hosts. The software executes one of two Internet protocols to carry out this transport, TCP (Transport Control Protocol) or UDP (User Datagram Protocol).

IP adds a 20 byte header to each packet, TCP, a 20 byte header (sometimes slightly bigger, but we will ignore the extra fields), and UDP, a 12 byte header. If TCP is used for transport, the headers add up to 40 bytes, and if UDP is used, they are 32 bytes. Each protocol uses its headers to carry out its mission. For example, an IP header contains, among other things, the IP addresses and the software port numbers of the source host and the destination host.

UDP is a very simple protocol that just transfers the packets. TCP is much more complex. The connection is managed by the headers added to the packets carrying the contents of the file being transferred, and by 40-byte control packets that have just headers but no file content. Consider the downloading of a file from a server to a client that requests the file through

a Web browser, an application managed by the application protocol HTTP. Three control packets open the connection, a *three-way handshake*: the client sends a SYN (synchronization) packet to the server, the server replies with a SYN/ACK (synchronization/acknowledgment) packet, and the client replies to that with an ACK (acknowledgment) packet. Then packets with file contents start flowing from the client to the server with information about the file being requested from the server, and then packets start flowing from the server to the client carrying the contents of the requested file. Through the period during which the files are being transferred, TCP control through the headers provides re-transmission of packets that are lost, changes the rate at which data packets are sent out based on an assessment of how congested the path is across the Internet, and provides information needed to re-assemble the files. Once the sending of the files is completed, more control packets are exchanged to close the connection. Note that because TCP reacts to lost packets by reducing the sending rate, the resulting packet traffic is closed-loop, that is, there is feedback.

## 1.2 Statistical Multiplexing

At any given moment, an Internet link has a number of simultaneous active connections; this is the number of connections between pairs of computers using the link to communicate. The packets of the different connections are intermingled on the link; for example, if there are three active connections, the arrival order of 10 consecutive packets by connection number might be 1, 1, 2, 3, 1, 1, 3, 3, 2, and 3. The intermingling is referred to as “statistical multiplexing”. As we will see, the statistical properties of the packet marked point process is heavily influenced by the magnitude of the multiplexing, that is, the number of active connections.

## 1.3 Measurement: Packet Traces

A monitor can be attached to an Internet link in such a way that the transmitted packets on the link are mirrored on the monitor without interfering with the transmission. This *passive monitoring* provides a measurement mechanism: time stamping and packet header capture. The monitor records the arrival time of the packet on the link and records the IP, TCP, and UDP headers. The time-stamp is 8 bytes and the headers are 40 bytes for TCP and 32 bytes for UDP. The sequence of such packet measurements over a time interval is a *packet trace*.

Packet trace collection on a link is an extremely powerful measurement mechanism that provides substantial information about the Internet — security, usage, network performance, and the performance of engineering protocols (Claffy, Braun, and Polyzos 1995; Paxson 1997; Fraleigh, Moon, Lyles, Cotton, Khan, Moll, Rockell, Seely, and Diot 2003; Micheel, Donnelly, and Graham 2001). The power of packet trace analysis comes, in part, because it provides not just a view of the link itself where the device is attached, but a view across the Internet along all of the paths taken by the connections whose packets use the link. Here is one example. Consider the three-way handshake that opens a TCP connection. The first arriving SYN packet starts out at the client, arrives at the monitored link where it is time-stamped, and then proceeds to the server. The server immediately sends a SYN/ACK packet that arrives at the link, where it is time-stamped, and proceeds to the client. The client immediately sends an ACK packet that arrives at the link, where it is time-stamped, and proceeds to the server. The time between the SYN time-stamp and the SYN/ACK time-stamp is a measure of the round-trip time between the link and the server; studying the round-trip time through time gives information about congestion along the path that might be slowing down the traveling of packets. Similarly, the time between the SYN/ACK time-stamp and the ACK time-stamp gives the round-trip time between the link and the client.

## 1.4 Comprehensive Analysis of Large Databases of Packet Streams

Packet traces can very quickly get large. Our current database has traces from 19 Internet links. One of them, BELL, has a peak load of 1000 packets/sec which means a packet trace accumulates at the rate of about 48 kilobytes/sec, which is 162 megabytes/hr. This is not too bad, partly because off-peak loads are much less; we have continuously monitored the link for about four years. But for another link where we have data, a core link on the network of Metropolitan Fiber Networks, the peak load is about 70000 packets/sec which is about 11 gigabytes/hr; the collection periods on such a link must be carefully designed.

One approach to the analysis of large databases is to use only summary analysis: carry out queries that retrieve functions of the detailed data that summarize behavior, and then analyze the resulting reduced information. While the study of summaries is important, it often does not succeed in exploiting fully the information in a large database.

For packet streams, summarization has been the standard in the statistical analysis of packet arrivals and sizes, which form a marked point process. The majority of studies of the marked point process has used reduced information: packet

counts, the number of packets arriving in a fixed interval of time such as 10 ms, and byte counts, the sum of the sizes of the packets arriving in the 10 ms intervals. To illustrate the data reduction, consider the above example of data from the MFN link. With an arrival rate of 70,000 packets/sec, over a 15 minute interval there are 63 million arrival times and 63 million sizes, so measurement of the marked point process consists of  $126 \times 10^6$  million observations. However, the number of packet counts plus byte counts in the 10ms interval is  $18 \times 10^4$ . The complete data set is 700 times larger.

The study of packet counts and byte counts in the past has led to important discoveries about Internet traffic, for example, long-range dependence and multi-fractal behavior (Leland, Taqqu, Willinger, and Wilson 1994; Paxson and Floyd 1995; Crovella and Bestavros 1996; Park, Kim, and Crovella 1996; Feldman, Gilbert, and Willinger 1998; Ribeiro, Riedi, Crouse, and Baraniuk 1999). But to fully understand the packet marked point process, it is necessary to study the complete arrival and size data (Ribeiro, Riedi, Crouse, and Baraniuk 1999; Cao, Cleveland, Lin, and Sun 2001; Gao and Rubin 2001). For example, doing so resulted in an important discovery about packet streams that has critical implications for engineering the Internet (Cao, Cleveland, Lin, and Sun 2002).

Our goal was to design a software and hardware system for the study of very large packet trace databases, but at the same time enable comprehensive analysis: detailed characterization, visualization, and modeling of the data, down to the level of individual observations.

## 1.5 Strategies of Analysis and S-Net

Achieving the goal of comprehensive analysis of packet streams is not simply a matter of highly efficient software and high performance hardware. Packet traces can be arbitrarily large, so any combined software and hardware combination is eventually defeated. One needs to develop strategies for analysis as well. The purpose of this article is to describe a collection of strategies for analysis that we have found to be useful, and to describe the components of an open-source software system, S-Net, based on these strategies [stat.bell-labs.com/s-net/]. S-Net is extensible but has a collection of tools that address tasks common to many packet stream studies. The S-Net tools can be *out-of-box*, which means S functions are used without modification; they can be *modified out-of-box*, which means simple changes to functions are carried out; or they can be *roll-your-own*, which means major changes are made or wholly new functions are written.

In this article, the strategies and S-Net are conveyed through their use in two studies of packet streams. Both studies contributed to PackMime, a collection of statistical models to generate Internet traffic variables in simulations to assess Internet technology. Sections 2 and 3 present the examples. Section 4 introduces S-Net and discusses software and hardware issues. Section 5 discusses strategies of analysis.

## 2 PackMime FSD: Open-Loop Internet Packet Traffic Generation

In a series of studies using S-Net, we investigated the statistical properties, including the long-range dependence (LRD), of four packet traffic variables: arrival times, inter-arrival times, packet counts, and byte counts (Cao, Cleveland, Lin, and Sun 2001; Cao, Cleveland, Lin, and Sun 2002). We discovered that the magnitude of statistical multiplexing has a dramatic effect on the statistical properties; for example, the LRD of the inter-arrivals and sizes dissipate as the magnitude increased.

We found that very simple statistical models, FSD models, which consist of a fractional ARIMA plus white noise, do an excellent job of describing the properties and how they change with the magnitude (Cao, Cleveland, and Sun 2003). Recall that TCP is closed-loop, creating traffic with feedback by reducing the sending rate when packets must be re-transmitted. But this feedback can be ignored if the rate of re-transmitted packets is small; in other words, the system can be treated as open-loop. In engineering studies where this is the case, the FSD models can be used for open-loop generation of packet arrivals and sizes.

In one phase of our study, we analyzed 2526 packet traces, 5 min or 90 sec in duration. The traces come from 6 monitors that measure packets from 15 links. Table 1 gives information about the traces. COS and AIX are from the NLANR database [pma.nlanr.net/PMA/], NZIX is from the NZIX-II database [wand.cs.waikato.ac.nz/wand/wits/nzix/2], and BELL is from the Bell Labs database [stat.bell-labs.com/InternetTraffic]. For NLANR, each link has its own monitor. At BELL, two local links, the two directions on the wire, are multiplexed because transmission is half-duplex. At NZIX, monitoring is carried out on a port that receives multiplexed mirrored streams from the 9 links on the switch. For these two monitors, we study the full stream, but also the individual link streams because queueing is minor in both cases. (2 links for NZIX are ignored because of very low loads.) The table shows the time interval for the traces of each link, the link speed (bandwidth), the link layer protocol, and the mean of the log base 2 of the average number of active connections for the traces in the group.

## 2.1 Comprehensive Variables and Summary Variables

A comprehensive variable has many values per trace and a summary variable has one value per trace that summarizes the behavior of the trace. We studied both comprehensive and summary variables in our open-loop modeling of packet traffic.

Consider packets arriving on a link for a single trace where  $v = 1$  corresponds to the first arriving packet,  $v = 2$  to the second, and so forth. We suppose that the time-stamps occur at the beginning moment of transmission. For packet counting purposes we divide the time block of each trace into sub-blocks of equal length. The length of each time sub-block here is 100 ms. The following are comprehensive variables ( $\log_2$  denotes log base 2):

- $q_v$  = packet sizes, headers plus file data but not link layer encapsulation
- $q_v^* = q_v$  normalized to have mean 0 and variance 1
- $q_v^{cap}$  = encapsulated packet sizes, headers plus data plus link layer encapsulation
- $a_v$  = arrival times
- $t_v = a_{v+1} - a_v$  = the inter-arrival times
- $t_v^* = \log_2 t_v$  normalized to have mean 0 and variance 1
- $p_i$  = the number of  $a_v$  falling into the  $i$ th sub-block
- $p_i^* = \log_2 p_i$  normalized to have mean 0 and variance 1
- $t_v^{min} = q_v^{cap} / link.speed$  = the time to transmit packet  $j$
- $b_{1v} = q_v^{cap} / t_v$  = the one-packet bandwidth.

The  $j$ -th packet, whose encapsulated size is  $q_v^{cap}$ , is transmitted during the beginning of the interval from  $a_v$  to  $a_{v+1}$ , whose length is  $t_v$ . So  $q_v^{cap} / t_v$ , the one-packet bandwidth, is the fraction of the capacity of the link that is used during the time interval  $a_v$  to  $a_{v+1}$ .

If there is no measurement error,  $t_v \geq t_v^{min}$ , and if in addition the  $(v + 1)$ -st packet is back-to-back with the  $v$ -th,  $t_v = t_v^{min}$ . Violations of the inequality  $t_v \geq t_v^{min}$  can occur from measurement error resulting in  $t_v < t_v^{min}$ . There are two types of error. One is simply time-stamp resolution,  $res$  sec. So for any case where we actually have  $t_v = t_v^{min}$ , and resolution is the only error, the measured inter-arrival lies in the interval  $t_v^{min} \pm 2res$ . The other error is a delay in writing a time-stamp if packets queue up on the monitor, move into memory, and then are time-stamped as a block. Because the processor is fast, if  $t_v < t_v^{min} + 2res$ , it is most likely that packets have queued up on the router. Because of these considerations, packet  $v$  is taken to be back-to-back with packet  $v + 1$  if  $t_v < t_v^{min} + 2res$ .

The following are trace summary variables:

- $\psi_k$  = the percent of packets that are back to back with  $k$  or more following packets
- $\nu$  = the coefficient of variation of the  $p_i$ , the standard deviation divided by the mean
- $c$  = the average number of active connections, where we average across all times in the trace
- $c^* = \log_2(c)$
- $\tau_1$  = the one-step entropy of a time series, the variance of the error of linear prediction one step ahead from the infinite past.

The  $\psi_k$  provide information about queueing on in the link input buffer.  $\nu$  is an important variable because it provides one characterization of the variability of the packet trace.  $c$  is important, because it measures the magnitude of the multiplexing, which has a large effect on the statistical properties of the arrivals and sizes.  $\tau_1$  is applied to a time series after a normalization that makes the mean 0 and the variance 1, so we have  $0 < \tau_1 \leq 1$ . If  $\tau_1 = 1$ , the series is independent (uncorrelated); if  $\tau_1$  is close to 0, the series is highly dependent in the sense that it can be predicted reliably from the past.

## 2.2 Time-Stamps and Back-To-Back Packets

Figure 1 applies a comprehensive visualization tool of S-Net to one trace from the NZIX(5min) link;  $\log_2(t_v)$  is graphed against  $\log_2(q_v^{cap})$  for each packet. There are two curves at the lower envelope of the plot:  $\log_2(t_v^{min} \pm 2res)$  vs.  $\log(q_v^{cap})$ . Points between the two curves could have arisen from cases where  $t_v = t_v^{min}$ , but resolution error jitters them by up to  $\pm 2res$ . A point can lie below the lower curve if monitor delay results in  $t_v < t_v^{min} - 2res$ , but does not in this case. Figure 2 shows a related comprehensive tool, a log one-packet bandwidth quantile plot. On the plot, the  $i$ -th largest value of the log one-packet bandwidth is graphed against  $(i - 0.5) / number.of.observations$ . If  $t_v < t_v^{min}$ , which can happen due to resolution error or monitor delay or both, a point has a one-packet bandwidth larger than  $link.speed$ , 100 mbps in this case.

In Figure 1, no point is below the lower boundary, so the measurements do not appear to have significant monitor delay. This is consistent with direct test results on the performance of the monitor. In Figure 2, a small fraction of points are slightly above 100 mbps, which is due to timer resolution. NZIX-II was the top monitor in our study with very few points

below the lower boundary on the inter-arrival-size plot, across all 100 traces. This is no surprise given the high quality of the measurement operation. The remaining monitors did show violations, the effects of monitors getting behind, but the effects were minor.

Queueing at the link input causes packets to be back-to-back; if the amount of queueing is large enough, this causes significant changes in the statistical properties of the  $t_v$  and the  $p_i$  because the time the packet arrives on the link is affected, but not the  $q_v$  because the packet order is not affected. We explored values of  $\psi_k$ , for  $k = 1, 3, \text{ and } 5$  to determine the amount of link queueing. Figure 3 is summary tool, a trellis display (Becker, Cleveland, and Shyu 1996) of  $\psi_1$  against  $\log_2(c)$  for each of the 15 links. There are 15 panels, one per link, and each point on a panel displays  $\psi_1$  against  $\log_2(c)$  for one trace. The panels are ordered, left to right and then bottom to top by the link means of  $\log_2(c)$ , shown in column 4 of Table 1. For all links,  $\psi_1$  increases with  $\log_2(c)$  as expected. The values of  $\psi_1$  vary substantially from panel to panel, reflecting very different amounts of queueing on the links.

### 2.3 Statistical Variability of Packet Counts

The mathematical theory of multiplexed point processes stipulates that the statistical variability of the  $p_i$  relative to the mean should decrease with the trace summary measure,  $c$ , the average number of active connections (Erramilli, Narayan, and Willinger 1996; Cao, Cleveland, Lin, and Sun 2002). This means that the variation in  $p_i/\bar{p}$  for a trace, where  $\bar{p}$  is the mean of the  $p_i$ , should decline across traces as  $c$  increases. The comprehensive tool in Figure 4 graphs the log base 2 of  $p_i/\bar{p}$  for two traces from the NZIX(5min) link. For the bottom panel,  $c$ , shown in the strip label at the top, is 1057 connections, and for the top panel, it is 3168 connections. Clearly the variability is much greater in the bottom panel than in the top.

Our trace summary measure of the amount of statistical variability is the coefficient of variation,  $\nu$ , the standard deviation of the  $p_i/\bar{p}$ . Theory says  $\nu$  should decline like  $c^{-0.5}$  until non-trivial queueing sets in. The summary tool in Figure 5 graphs  $\log_2(\nu)$  against  $\log_2(c)$  for each of the 15 links. The line on each panel is the least-squares line with slope  $-0.5$ . If the theory applies, the pattern of the points should follow the line. In fact, agreement is quite good.

### 2.4 Long-Range Dependence (LRD)

The mathematical theory of multiplexed point processes stipulates that the LRD of the packet inter-arrivals,  $t_v$ , and the packet sizes,  $q_v$ , should always be present, but should have less and less influence as the magnitude of multiplexing increases (Daley and Vere-Jones 1988).  $c$ , the average number of active connections, is a summary measure of the magnitude for each trace. The theory also stipulates that the LRD of the packet counts,  $p_i$ , should remain the same with  $c$  until the amount of input queueing becomes non-trivial; but because the statistical variability of the  $p_i$  relative to the mean  $\bar{p}$  decreases with  $c$ , the LRD becomes less and less salient because the magnitude of the bursts due to LRD decreases. Let  $q_v^*$ ,  $t_v^*$ , and  $p_i^*$  be the time series  $q_v$ ,  $\log(t_v)$ , and  $\log(p_i)$ , respectively, normalized to have mean 0 and variance 1. We use the power spectrum and the one-step entropy summary measure,  $\tau_1$ , to study the LRD of the three series.

Figure 6, a summary tool, graphs  $\tau_1$  against  $c^* = \log_2(c)$ , for the  $t_v^*$ . Each panel shows the values for one link. Figures 7 and 8 do the same for  $q_v^*$  and  $p_i^*$ , respectively. The theory holds up well on these plots. For the  $q_v^*$ ,  $\tau_1$  increases toward 1. For the  $p_i^*$ , there is no consistent pattern. For the  $t_v^*$ ,  $\tau_1$  increases toward 1 for most links except AIX1(90sec), AIX2(90sec), NZIX7(5min), and NZIX(5min), where it decreases a bit; estimates of the power spectrum will explain this behavior.

Figure 9, a comprehensive tool, graphs 3 power spectrum estimates for  $q_v^*$  from each of the 15 links. There are 45 panels on the display broken into 15 sets of three panels, one set per link. Going through the links from left to right and then bottom to top, the order is the same as that of the links in Figure 7. Within each set of three panels, the value of  $c^*$  for which the spectrum is estimated increases left to right and is shown at the top middle of each panel. In the figure, the general pattern for smaller values of  $c^*$  is very large power at low frequencies, a rapid rise as the frequency tends to 0, and an overall monotone decrease in power as the frequency increases from 0 cycles/packet to 0.5 cycles/packet. This is a result of a persistent long-range dependence in the size time series. But this pattern changes with  $c^*$  as follows: as  $c^*$  increases, the fraction of low-frequency power decreases and the fraction of high-frequency power increases; this means the long-range dependence decreases, and the estimates tend toward the constant spectrum of white noise. This happens for each link individually. It also happens across links as well; the links toward the top with higher average  $c^*$  have less long-range dependence than those at the bottom with lower values.

Figure 10 graphs the 3 power spectrum estimates for  $t_v^*$  using the same method as in Figure 9. For each link other than AIX1, AIX2, NZIX7, and NZIX, the general pattern is very large power at low frequencies, a rapid rise as the

frequency tends to 0, and an overall monotone decrease in power as the frequency increases from 0 cycles/inter-arrival to 0.5 cycles/inter-arrival. This is a result of the persistent long-range dependence of the  $t_v^*$ . But as  $c^*$  increases, the fraction of low-frequency power decreases, the fraction of high-frequency power increases, and the spectra approach the spectrum of white noise. For AIX1, AIX2, NZIX7, and NZIX, as  $c^*$  increases, the spectrum near 0 decreases, but the spectrum at high frequencies decreases as well so that the end result is a log spectrum that is not flat and decreases slowly as the frequency increases. This is the spectrum of a time series with very short range dependence. This accounts for the slight decrease in the entropy for these links.

Figure 11 graphs the estimates of the power spectrum of  $p_i^*$  using the same method as in Figure 9. They exhibit the expected persistent long-range dependence as for the other series. For each link, the spectra show little change with  $c^*$ . Furthermore, the spectra for different links are very similar.

### 3 PackMime HTTP: Transfer Request Models for Closed-Loop Internet Packet Traffic Generation

In another study using S-Net, we developed models for HTTP application connection variables (Cao, Cleveland, Lin, and Sun 2001; Cao, Cleveland, Gao, Jeffay, Smith, and Weigle pear). For example, two of the connection variables are the size of the file sent by the client and the size of the file sent by the sender. Another two are the client-side round-trip time measured by the time from the server SYN/ACK to the client ACK in the three-way handshake, and the server-side round-trip time measured by the time from the client SYN to the server SYN/ACK in the three-way handshake. The purpose of the modeling is to provide stochastic generation of application requests for network simulators such as NS-2 (McCanne and Floyd 1998). The architecture consists of two parts: (1) the statistical models, which stochastically generate requests for TCP connections among a cloud of hosts; and (2) a simulated network carrying packets resulting from the generated connections. Actual TCP implementations are used to deliver the packets across the simulated network and provide TCP feedback. Thus the packet generation is closed-loop.

As in the study of Section 2, we discovered that the magnitude of statistical multiplexing also has a major effect on the statistical properties of application connection variables, but in this case it is convenient and natural to use the number of new HTTP connections per second — that is, the SYN packet arrival rate — as the measure of the magnitude of multiplexing. There are sources of the HTTP requests, single user sessions, and the requests arriving at the link are a multiplexing of the source requests on the link. As the number of sources increases, the request rate tends to increase.

Here, we show a few tools applied to one connection variable: HTTP start times. The data are the arrival times of client HTTP SYN packets on the link for the 500 BELL(5min) traces in Table 1; each arrival is the beginning of the usage of the link for a Web file transfer by TCP.

#### 3.1 Comprehensive Variables and Summary Variables

As for the packet traffic in Section 2, there are comprehensive variables with many values per trace, and summary variables with one value per trace. The following are comprehensive variables:

- $a_v$  = HTTP connection start times
- $t_v = a_{v+1} - a_v$  = inter-arrival times
- $t_v^* = \log_2(t_v)$  normalized to have mean 0 and variance 1.

The following are summary variables:

- $\rho$  = the average number of new HTTP connections per second, the inverse of the mean inter-arrival time
- $\rho^* = \log_2(\rho)$ .

#### 3.2 Marginal Distribution

We studied the marginal distribution of the  $t_v$  of each trace by a comprehensive tool, a Weibull quantile plot, to see if the Weibull provides a good fit to the marginal distribution. Let  $w$  be a random variable that has a Weibull distribution with shape parameter  $\lambda$  and scale parameter  $\alpha$ . Then  $(w/\alpha)^\lambda = u$  where  $u$  is a unit exponential. So if  $\lambda = 1$ ,  $w$  has an exponential distribution. For  $\lambda < 1$ , the upper tail of  $w$  is heavier than that of the exponential, but as  $\lambda$  increases to 1, the upper tail becomes less heavy.

Figure 12 shows the Weibull plot for two BELL(5min) traces. The  $t_v^*$  are graphed against the log quantiles of the exponential. The oblique line is drawn through the 25% and 75% quantiles. The vertical lines show the 5%, 10%, 25%,

75%, 90%, and 95% quantiles. The value of  $\rho$  is shown in the strip label at the top of each panel. If the pattern of the points is a line, then the  $t_v$  are well approximated by the Weibull. In such a case, the inverse of the slope of the pattern is the Weibull shape parameter.

In Figure 12, the pattern of the points follows the quartile line well on both panels. There are deviations at the bottom end of the distribution. The empirical distribution is truncated to the transmit time of a minimum-size packet, 40 bytes plus encapsulation. This effect, which increases with the magnitude of the multiplexing, is never more than minor for the 500 Bell(5min) traces; for the two plots of the figure, the vertical lines show the truncation affects no more than 5% of the data. However, the overall conclusion is that the Weibull provides an excellent fit.

In Figure 12, the inverse slopes of the patterns of the points, the Weibull shape parameters, are both less than 1, but in the top panel the shape parameter is considerably greater, so the distribution is closer to an exponential. This is what theory would predict. As the multiplexing increasing, the arrivals tend toward Poisson, so the marginal distribution tends toward exponential. The display shows just two traces, but we can check the theoretical result across all traces. We fitted the Weibull distribution to the 500 BELL(5min) traces; for each trace there is an estimate,  $\hat{\lambda}$ , of  $\lambda$ , a trace summary variable. Figure 13, a summary tool, is a plot of  $\hat{\lambda}$  against  $\rho^*$ . The plot shows the effect of the magnitude of the multiplexing on  $\lambda$ . As  $\rho^*$  increases,  $\hat{\lambda}$  tends to 1, so the inter-arrival marginal tends toward the exponential as the theory predicts.

### 3.3 Long-Range Dependence (LRD)

Like the packet inter-arrivals, the HTTP inter-arrivals have a component of LRD that is reduced with the magnitude of multiplexing. The comprehensive tool in Figure 14 reveals the changing properties of the  $t_v$  with  $\rho$ . In the bottom panel,  $t_v^*$  is graphed against  $a_v$  for one BELL(5min) trace. The value of  $\rho$  is 1.99 connections/sec. The log on the vertical scale is vital because the  $t_v$  vary by several orders of magnitude. The horizontal scale, however, conveys arrivals and inter-arrivals on the original scale. The top panel of Figure 14 plots another trace with  $\rho$  equal to 32.6 connections/sec.

In the bottom panel of Figure 14, the data form distinct vertical bands, taking values in the middle of the distribution of values. These are bursts of connections caused by single clicks of individual users; a single click can lead to the opening of many connections. The larger values of the  $t_v^*$  on the plot tend to be quiescent times until the click of some user occurs. The resulting burstiness gives rise to the dependence in  $t_v^*$ . In the top panel of Figure 14, the bursty behavior at the small time scales has disappeared. Because  $\rho$  is much larger, the SYN's of more users intermingle, and the behavior of individual users is broken up. The  $t_v$  are headed toward independence, their behavior when they are the inter-arrivals of a Poisson process.

## 4 S-Net: Software and Hardware Issues

S-Net runs on Unix and is written in C, Perl, Unix commands, and the S language for visualization and data analysis (Becker, Chambers, and Wilks 1988; Chambers 1998). The system addresses the full set of tasks in the study of packet streams, from the initial processing of raw packet traces to the final output, often a visual display.

S-Net has a pipeline: (1) raw packet traces; (2) a database with objects tailored to ensuing analyses; and (3) an environment with tools for data analysis: statistical methods, model fitting, and visualization. The analysis environment is S; there are tools for converting database objects into S objects, and tools for data analysis. The database consists of Unix flat files; software written in C and Perl converts raw packet traces in a variety of standard formats into database objects and manages the database. Computationally intensive data processing and database management are carried out using the Perl programs, C programs, and Unix commands. Comprehensive and summary analysis, and less computationally intensive database management, are carried out using S.

### 4.1 Open Source

The S-Net software is open-source. We did this by using the Linux version of Unix [www.linux.org], and the R version of the S language [www.r-project.org] (Ihaka and Gentleman 1996; Venables and Smith 2002), both open-source systems. S-Net software is also open-source.

## 4.2 Extensibility

One important goal in designing S-Net was to make comprehensive analysis flexible. It is important to provide a collection of tools that accommodate widespread tasks carried out by researchers who study packet streams, but in addition, provide an environment that allows an analyst to readily alter tools or build entirely new ones. In other words, the system must be extensible.

Extensibility in S-Net is provided by the design of Unix and S, which allow ready extensibility, so the user is not bound by only those tools in the system. This gives the analyst the flexibility to tailor analyses to the specific questions being asked of the data. S-Net tools can be *out-of-box*, which means S functions are used without modification; they can be *modified out-of-box*, which means simple changes to functions are carried out; or they can be *roll-your-own*, which means major changes are made or wholly new functions are written. The tools used in the examples of Sections 2 and 3 each fall into one of these three categories that are cited in the figure legends.

S has another dimension of extensibility. Tools written in C or using Unix system commands can be called from within S-Net by S commands and have the results returned as S objects.

There have been many useful tools developed for analyzing Internet packet streams. A survey is available at CAIDA [[www.caida.org/tools/](http://www.caida.org/tools/)]. Most of these available tools provide summary reporting for variables such as packet counts, byte counts, and flow characteristics. S-Net is the first comprehensive, extensible system that provides for the processing of raw packet stream traces in many commonly used formats, tools for database management, and tools for comprehensive, detailed data analysis. In fact, the tools of the CAIDA survey could be called from the S-Net system.

## 4.3 Cluster Hardware

As we will see in Section 5, the strategy of analysis underlying S-Net results in use of the same computation over different blocks of data, so this computation is “embarrassingly parallel.” Thus the strategy lends itself to implementation on a hardware environment consisting of a cluster of computers distributed over a high-speed network or single switch, each computer storing flat-file objects and S objects on attached disk. The disks of the individual computers are shared through the Network File System (NFS). This storage architecture helps to reduce the burden on individual disks when large amounts of data are accessed by multiple processors from different disks on different computers.

S-Net has so far been implemented on two Linux clusters over fast networks, one with 100 mbps and 1 gbps links, and another with 1 gbps and 2.5 gbps links.

# 5 Analysis Strategy for Comprehensive Analysis

Comprehensive analysis, detailed and summary, is achieved by five aspects of S-Net: (1) a dataflow that progresses from raw packet trace files, to primary and secondary UNIX files, and then to S objects; (2) time blocking with repetitive analysis — break traces from a single monitor into many time blocks, use the same analysis for all blocks, and then combine the results of the analyses; (3) a combination of comprehensive tools that show the detail in the data and tools that summarize behavior; (4) mechanisms for taking into account, the immense impact that the magnitude of statistical multiplexing has on the properties of many traffic variables; (5) very large visual displays using trellis display, a visualization framework that allows the analyst to create a single display with hundreds or thousands of pages and tens of panels per page (Becker, Cleveland, and Shyu 1996); (In R, this framework is implemented as lattice graphics (Murrell 2001; Sarkar 2003). Collectively, these aspects amount to a strategy for approaching the analysis of packet streams.

## 5.1 Data Flow

Figure 15 shows the data flow of S-Net. Primary flat-file objects are compressed flat ASCII files that form the master database in the S-Net analysis system. One example of primary files is the data of a single monitor broken into time blocks (by packet arrival time) with headers in a block ordered by their time-stamps. This is what was accessed to carry out the analysis described in Section 2. A second example, again for a single monitor, is headers of TCP packets organized by TCP connection; each time block has the packets of all connections that start in the block, the packets in the block are organized by connection, the connections are ordered by their start times (arrival times of SYN packets), and for each connection the headers are ordered by their time-stamps. This is what was accessed to carry out the analysis described in Section 3.

Secondary flat-file objects are compressed flat ASCII files containing derived quantities from the primary flat-file objects. For example, for the TCP connection primary files, secondary files might be the counts of connection starts in 10 ms

intervals. Secondary flat files or pieces of secondary flat files are read into S and become S objects.

Primary flat files underlie very large numbers of data analyses. The secondary flat files are created with more specific analyses in mind. Further database management occurs through selecting just some information to be read into S. The resulting reads create S objects that typically form the basis for a specific analysis; but in the course of the analysis many new S data objects are often created from these basic S objects to provide more focused access of data.

This design of the data flows contributes to the ease of comprehensive analysis. The flow from primary packet files to S objects makes it far more convenient to carry out detailed study because the analyst is typically spending time altering analyses through access of S objects and secondary flat files, rather than going all the way back up to the primary flat files (or the raw packet header traces), which is tedious and time-consuming. Of course, this approach does require planning: forecasting the likely course of an analysis and then tactical thinking about how to form objects to enable more efficient execution of this likely course.

## 5.2 Time Blocking with Repetitive Analysis

We have found that a useful strategy for achieving comprehensive, detailed analysis is time blocking with repetitive analysis. Suppose we have a long trace from a monitor covering many days, weeks, or months. We break the trace into time blocks, and apply the same analysis to all blocks; this strategy is time blocking with repetitive analysis. The number of blocks can (and often should) be in the hundreds or thousands. Sections 2 and 3 are two examples.

A critical issue is the duration of the time blocks. Some issues call for longer blocks, others shorter. First, we want stationarity of phenomena over a block. This pushes toward shorter blocks. But we need to have a time block last long enough that the phenomena under study can come to completion. For this purpose we can in a sense extend beyond the block; for the example of Section 3, we study all flows that begin in a 5 min block, but measure aspects of the flow that extend beyond the block, sometimes hours or even days. Finally, we need to take the computational burden of the tools into account; more burdensome tools call for shorter blocks.

For the examples in Sections 2 and 3, 5 min was a reasonable choice. The AIX and COS links were only 90 sec, less than ideal, but the longest available; to make sure that this did not bias our results, we analyzed 90 sec traces for BELL and NZIX, and found results were similar to those for 5 min except there was greater statistical variability. The major issue that pushes toward shorter blocks is diurnal variation; the usage of a link has a strong 24 hour period, and getting much beyond about 15 min runs the risk of interference from this changing level of usage, which means a changing magnitude of multiplexing. In both cases, we want the mean magnitude of statistical multiplexing — the mean number of active connections or the mean number of new connections per second — to be constant within the block to ensure stationarity.

Of course, characteristics can and do change over time scales greater than the block length. This can be investigated by studying how the results of the block analyses change with time. Statistically, the blocks represent repeat samples that have a changing magnitude of multiplexing.

## 5.3 Comprehensive and Summary Study

It is important to have comprehensive tools that allow study in the finest possible detail. In effect we seek tools that allow us to see every individual measurement. For example, in Section 3, the Weibull quantile plot of inter-arrival times and the plot of log inter-arrivals against arrivals, both show every arrival in a trace. In Section 2, the plot of log inter-arrival against log encapsulated size, the plot of one-packet bandwidth, and the plot of log packet size minus the log mean against time all show individual measurements of the variables under study. Anything less runs the risk of missed effects. Summaries, with one value per trace, are indeed useful. In Sections 2 and 3, plots of summaries against measures of the magnitude of multiplexing are critical to the analyses. But the summaries are not nearly enough.

## 5.4 Multiplexing

Statistical multiplexing needs to be at the core of many studies of Internet packet streams. The magnitude of statistical multiplexing has a large impact on the statistical properties of many variables, both packet variables and application connection variables. This is amply demonstrated in the studies cited in Sections 2 and 3. Properties need to be related to measures of the magnitude of multiplexing such as the number of active connections or the number of connections per second. S-Net provides ample capabilities to take the magnitude of multiplexing into account.

## 5.5 Trellis and Very Large Displays

Time-blocking and trellis display (Becker, Cleveland, and Shyu 1996) go well together. Trellis is a framework for the visualization of multivariable databases. In R, this framework is implemented as lattice graphics (Murrell 2001; Sarkar 2003).

The most prominent aspect of trellis display and lattice graphics is an overall visual design, reminiscent of a garden lattice or trelliswork, in which panels are laid out into columns, rows, and pages. On each panel, a subset of the data is graphed by a display method such as a scatterplot, curve plot, boxplot, 3-D wireframe, Weibull quantile plot, or dot plot. Each panel shows the relationship of certain variables, the panel variables, conditional on the values of other variables, the conditioning variables. The strip labels at the tops of panels are indicators of the values of conditioning variables. A number of display methods employed in the visual design of trellis display enable it to uncover the structure of data even when the structure is quite complicated. All visualizations shown in Sections 2 and 3 used trellis software implemented in the S language. For example, Figures 9, 10, and 11 are the pages of a 3-page trellis display, each page with 45 panels.

Sending a display across many pages is necessary to study a very large trace database in detail. There can be hundreds or thousands of pages. Figures 1, 2, 4, 12, and 14 are single-panel or two-panel trellis displays with a single page. But in the actual displays of our analyses, there are as many panels across all pages as there are traces. So for the first three tools there are 2526 panels, the number of traces in the PackMime FSD study, and for the last two there are 500, the number in the PackMime HTTP study. This means, of course, there are a very large number of pages. Our practice is to break the panels into separate displays by link, and produce a single display for each link.

We find it convenient to store displays as a PostScript document and study them using Ghostview. Even though the document manipulation of Ghostview is simplistic, it nevertheless quite useful, but an important and interesting topic for future work is much more effective methods of manipulating the layout of panels and pages.

## References

- Becker, R. A., J. M. Chambers, and A. R. Wilks (1988). *The New S Language*. London: Chapman & Hall.
- Becker, R. A., W. S. Cleveland, and M. J. Shyu (1996). The Design and Control of Trellis Display. *Journal of Computational and Statistical Graphics* 5, 123–155.
- Cao, J., W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. Weigle (2004, to appear). Stochastic Models for Generating Synthetic HTTP Source Traffic. In *Infocom 2004. Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies*.
- Cao, J., W. S. Cleveland, D. Lin, and D. X. Sun (2001). On the Nonstationarity of Internet Traffic. *ACM SIGMETRICS* 29(1), 102–112.
- Cao, J., W. S. Cleveland, D. Lin, and D. X. Sun (2002). Internet Traffic Tends *Toward* Poisson and Independent as the Load Increases. In C. Holmes, D. Denison, M. Hansen, B. Yu, and B. Mallick (Eds.), *Nonlinear Estimation and Classification*, pp. 83–109. New York: Springer.
- Cao, J., W. S. Cleveland, and D. X. Sun (2003). Fractional Sum-Difference Models for Open-Loop Generation of Internet Packet Traffic. Technical report, Bell Labs, stat.bell-labs.com.
- Chambers, J. M. (1998). *Programming with Data*. New York: Springer.
- Claffy, K., H.-W. Braun, and G. Polyzos (1995). A Parameterizable Methodology for Internet Traffic Flow Profiling. *IEEE Journal on Selected Areas in Communications* 13, 1481–1494.
- Crovella, M. E. and A. Bestavros (1996). Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *ACM SIGMETRICS* 24(1), 160–169.
- Daley, D. J. and D. Vere-Jones (1988). *An Introduction to the Theory of Point Processes*. New York: Springer-Verlag.
- Erramilli, A., O. Narayan, and W. Willinger (1996). Experimental Queueing Analysis with Long-Range Dependent Packet Traffic. *IEEE/ACM Transactions on Networking* 4, 209–223.
- Feldman, A., A. A. Gilbert, and W. Willinger (1998). Data Networks as Cascades: Explaining the Multifractal Nature of Internet WAN Traffic. In *Proceedings ACM SIGCOMM*, pp. 42–55.

- Fraleigh, C., S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot (2003). Packet-Level Traffic Measurements from the Sprint IP Backbone. *IEEE Network* 17, to appear.
- Gao, J. and I. Rubin (2001). Multiplicative Multifractal Modeling of Long-Range-Dependent Network Traffic. *International Journal of Communications Systems* 14, 783–201.
- Ihaka, R. and R. Gentleman (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5(3), 299–314.
- Leland, W., M. Taqqu, W. Willinger, and D. Wilson (1994). On the Self-Similar Nature of Ethernet Traffic. *IEEE/ACM Transactions on Networking* 2, 1–15.
- McCanne, S. and S. Floyd (1998). UCB/LBNL Network Simulator - ns (version 2). [www-mash.cs.berkeley.edu/ns/](http://www-mash.cs.berkeley.edu/ns/).
- Micheel, J., S. Donnelly, and I. Graham (2001). Timestamping Network Packets. In *First ACM SIGCOMM Workshop on Internet Measurement Workshop*, San Francisco.
- Murrell, P. (2001). R Lattice Graphics. In *Proceedings of the 2nd International Workstop on Distributed Statistical Computing*, Vienna. [www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/](http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/).
- Park, K., G. Kim, and M. Crovella (1996). On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic. In *Proceedings of the IEEE International Conference on Network Protocols*.
- Paxson, V. (1997). End-to-End Internet Packet Dynamics. In *Proceedings ACM SIGCOMM*, pp. 139–152.
- Paxson, V. and S. Floyd (1995). Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking* 3, 226–244.
- Ribeiro, V. J., R. H. Riedi, M. S. Crouse, and R. G. Baraniuk (1999). Simulation of NonGaussian Long-Range-Dependent Traffic Using Wavelets. *ACM SIGMETRICS* 27, 1–12.
- Sarkar, D. (2003). Some Notes on Lattice. In *Proceedings of the 3rd International Workstop on Distributed Statistical Computing*, Vienna. [www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/](http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/).
- Venables, W. N. and D. M. Smith (2002). *An Introduction to R*. Bristol: Network Theory.

Name	Number	Link	$\log_2(c)$
AIX1(90sec)	23	622mbps PoS	13.09
AIX2(90sec)	23	622mbps PoS	13.06
COS1(90sec)	90	156mbps ATM	10.83
COS2(90sec)	90	156mbps ATM	10.81
NZIX(5min)	100	100mbps Eth	10.75
NZIX7(5min)	100	100mbps Eth	9.60
NZIX5(5min)	100	100mbps Eth	8.66
NZIX6(5min)	100	100mbps Eth	7.85
NZIX2(5min)	100	100mbps Eth	7.32
NZIX4(5min)	100	100mbps Eth	7.17
BELL(5min)	500	100mbps Eth	6.97
NZIX3(5min)	100	100mbps Eth	6.54
BELL-IN(5min)	500	100mbps Eth	5.98
BELL-OUT(5min)	500	100mbps Eth	5.94
NZIX1(5min)	100	100mbps Eth	4.42

Table 1: Name: link name including length of trace interval • Number: number of traces • Link: link speed and link-layer protocol •  $\log_2(c)$ : mean log base 2 average number of active connections

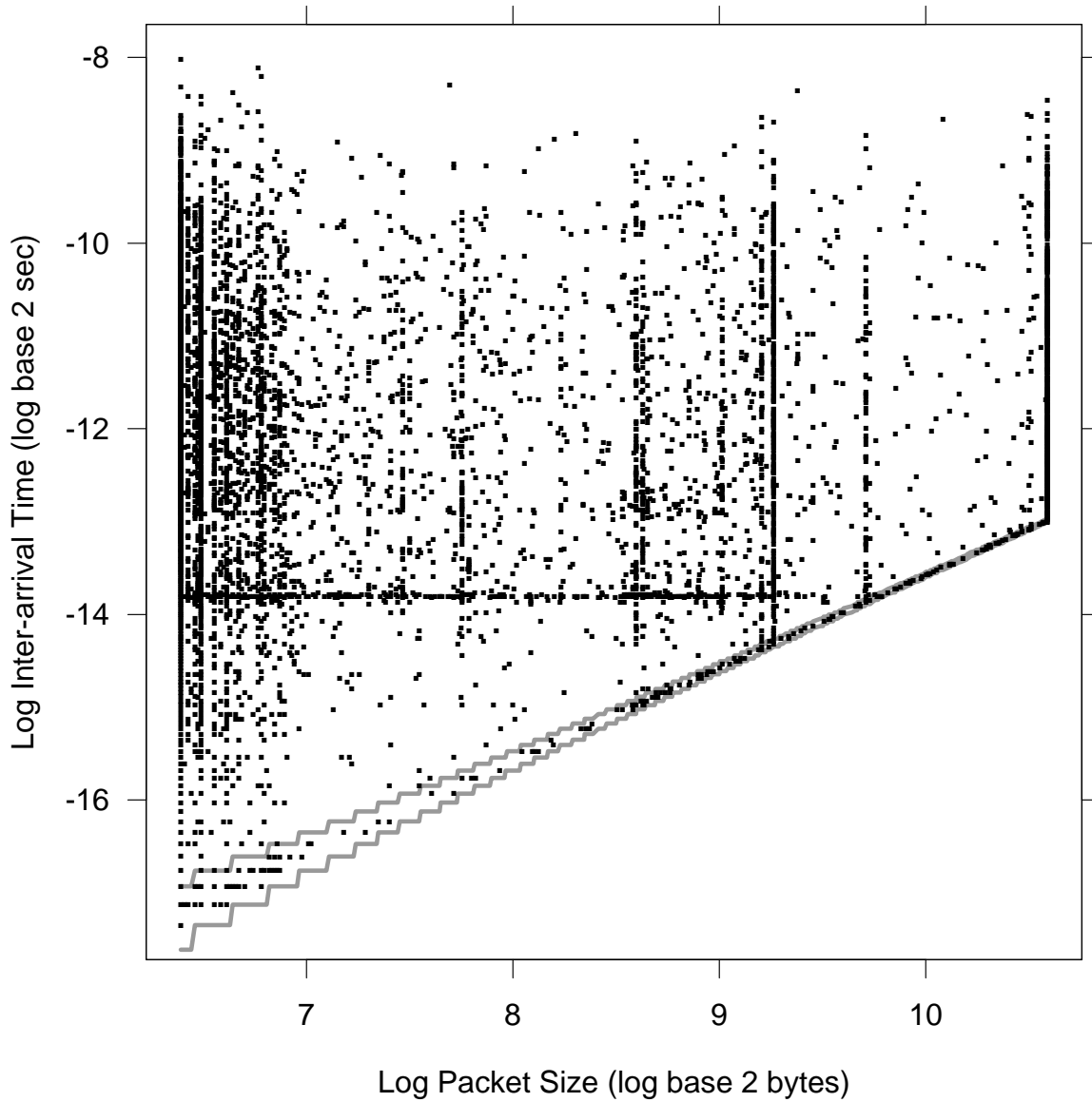


Figure 1: Log inter-arrival time is graphed against log encapsulated packet size for an NZIX(5min) trace. This out-of-box comprehensive tool is described in Section 2.2.

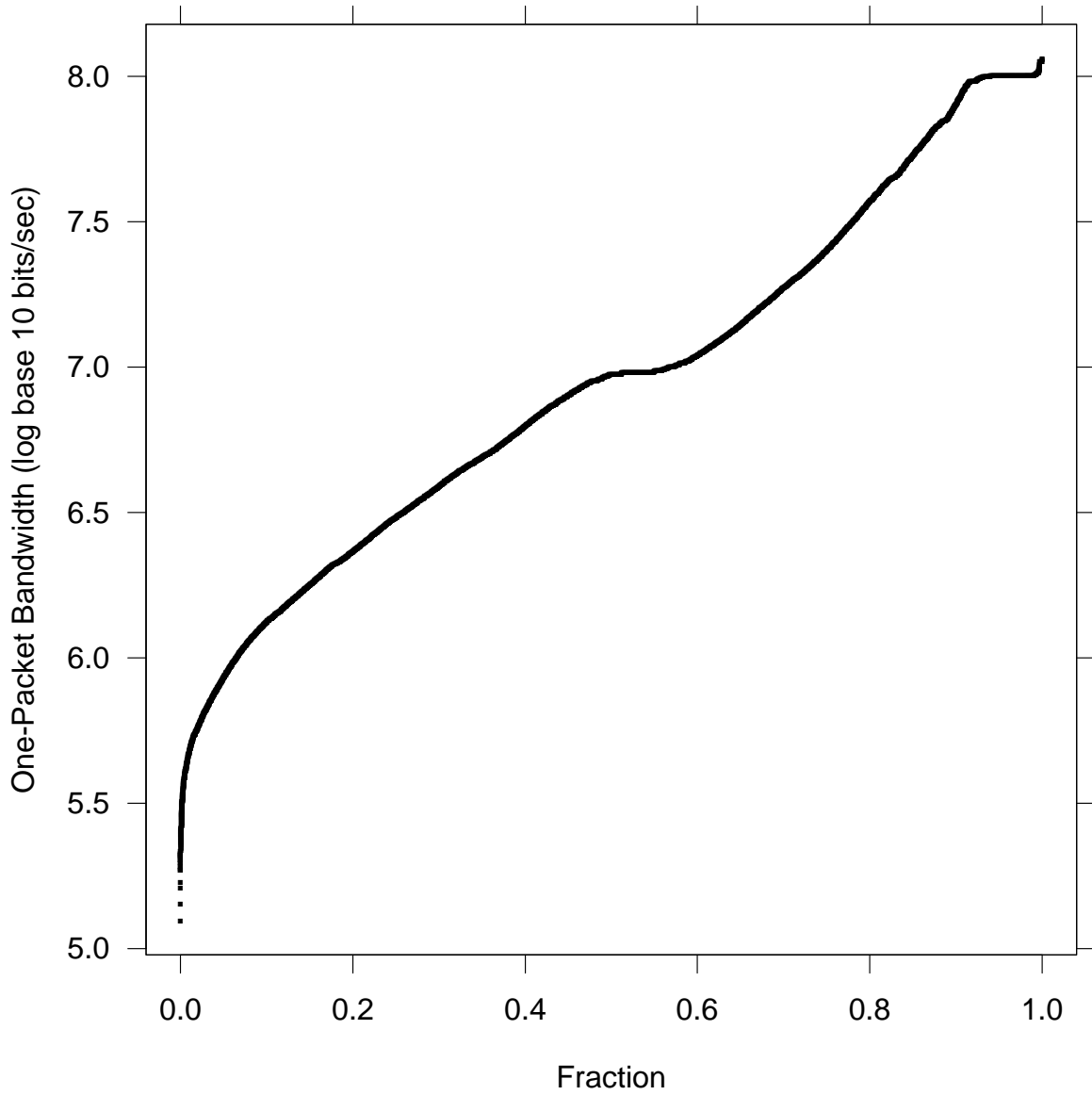


Figure 2: Quantiles of log one-packet bandwidth are graphed against the quantile fraction for the same trace as in Figure 1. This out-of-box comprehensive tool is described in Section 2.2.

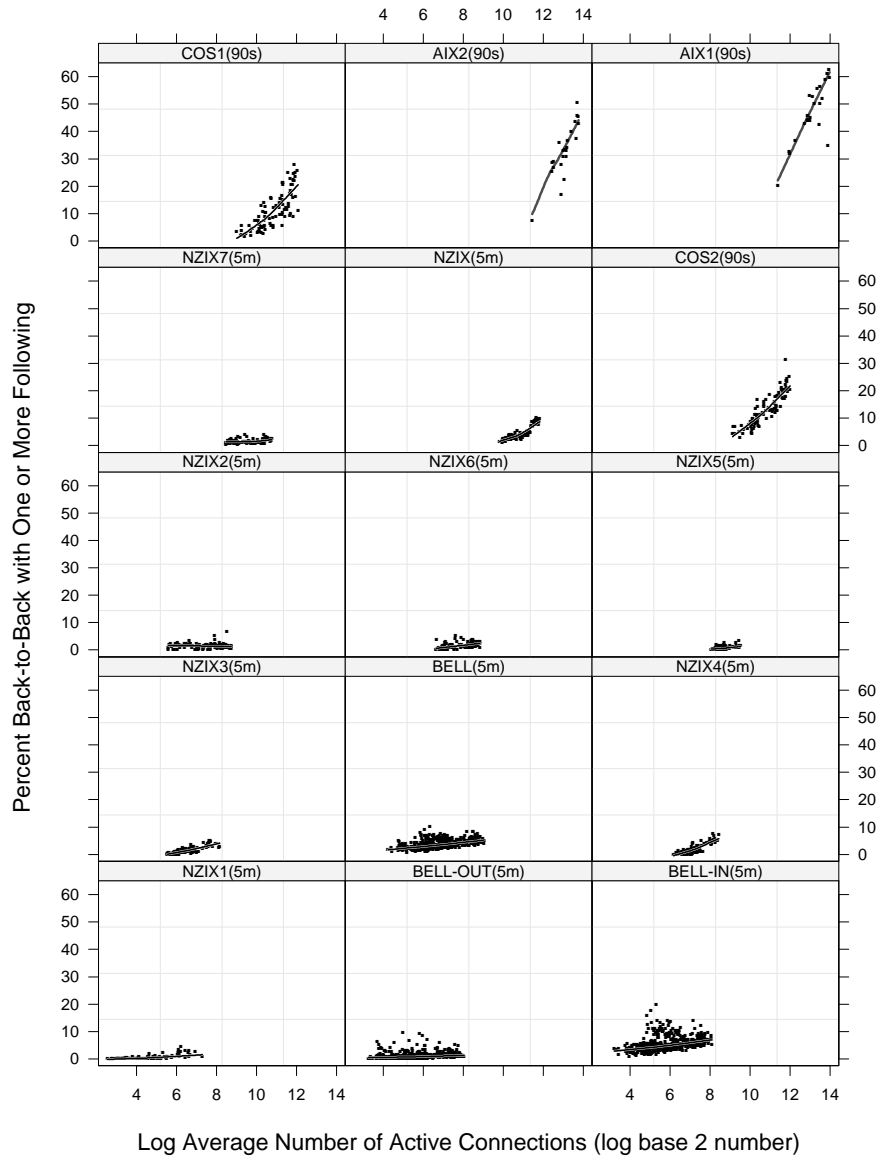


Figure 3: Percent of packets back-to-back with one or more following packets is plotted against log average number of active connections. This modified out-of-box summary tool is described in Section 2.2.

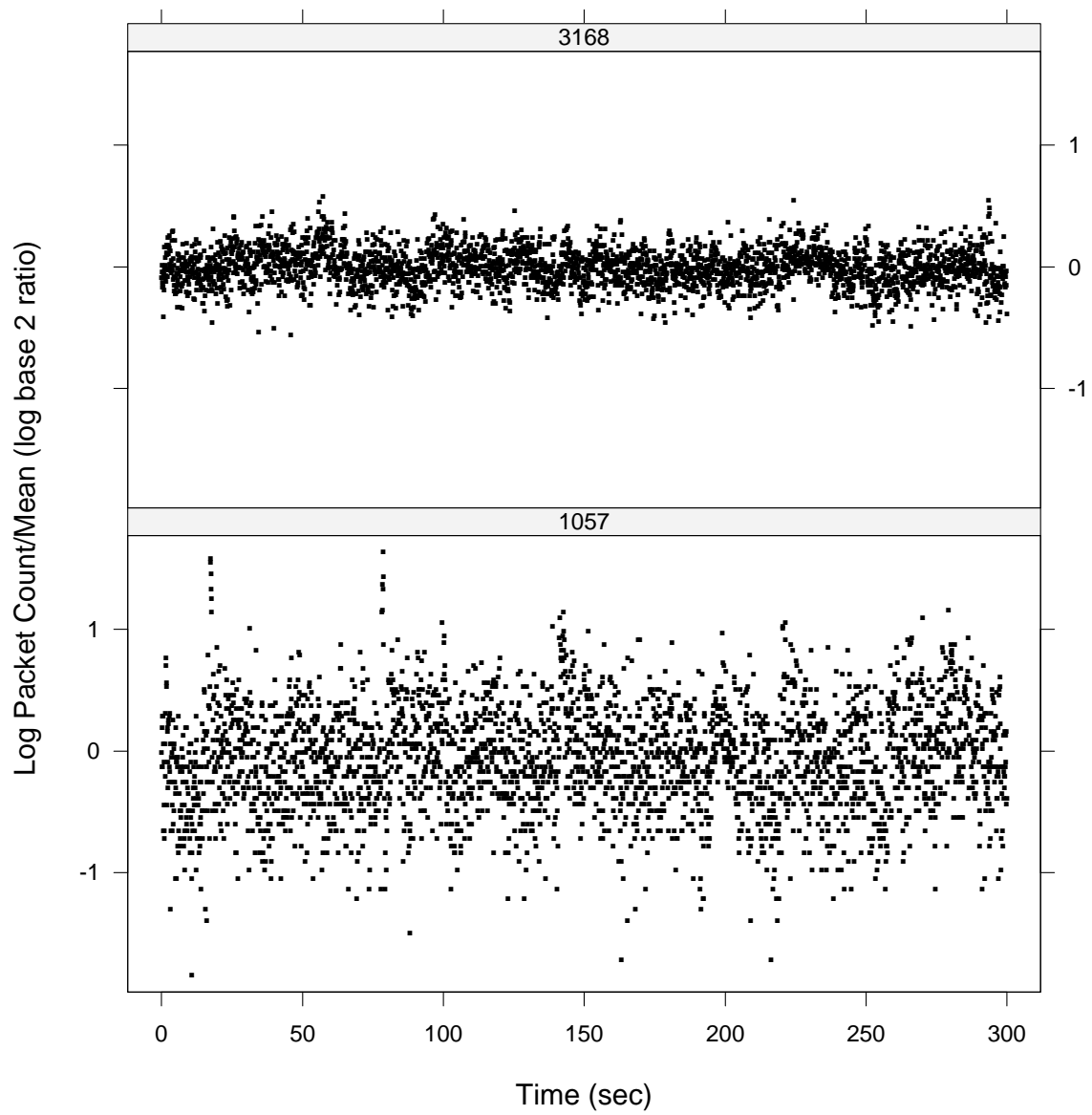


Figure 4: Log 100-ms packet count divided by the mean is graphed against time for two NZIX(5min) traces. This modified out-of-box comprehensive tool is described in Section 2.3.

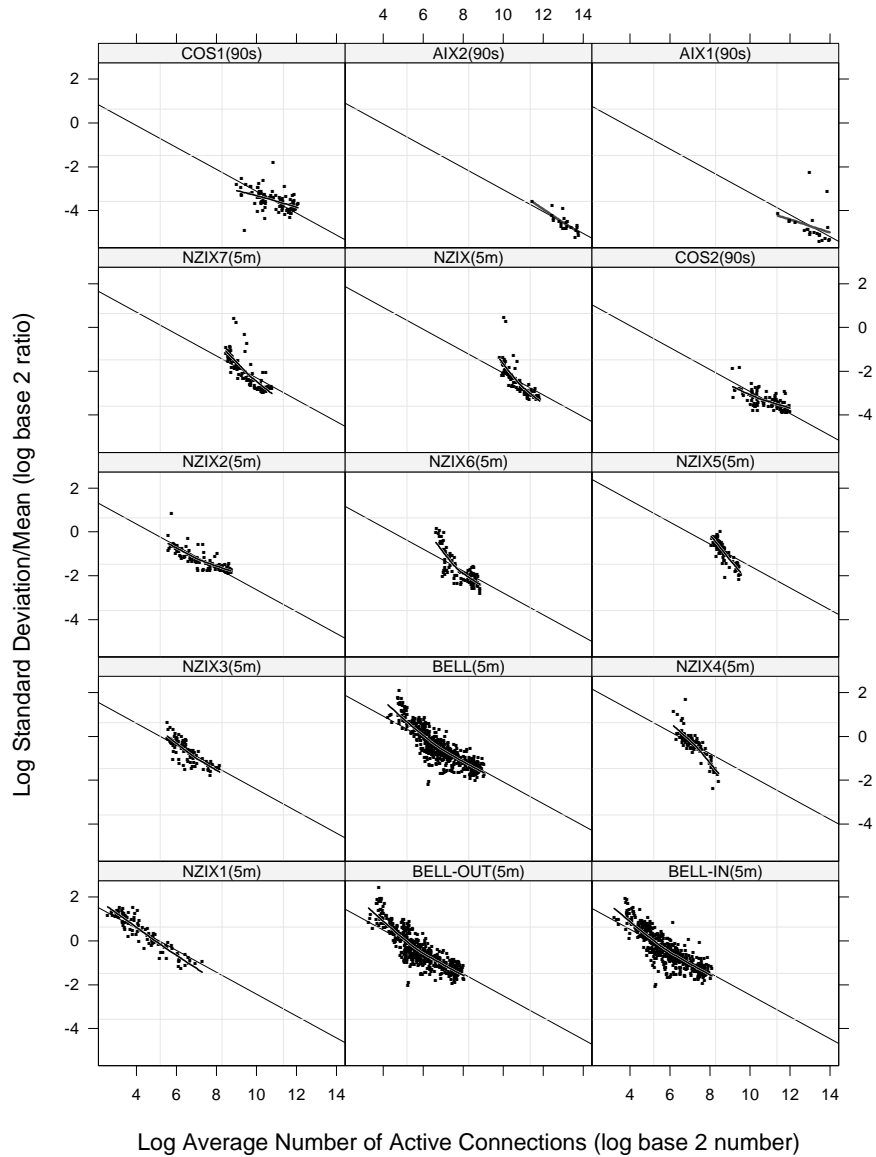


Figure 5: Log coefficient of variation for the packet counts is plotted against log average number of active connections. This modified out-of-box summary tool is described in Section 2.3.

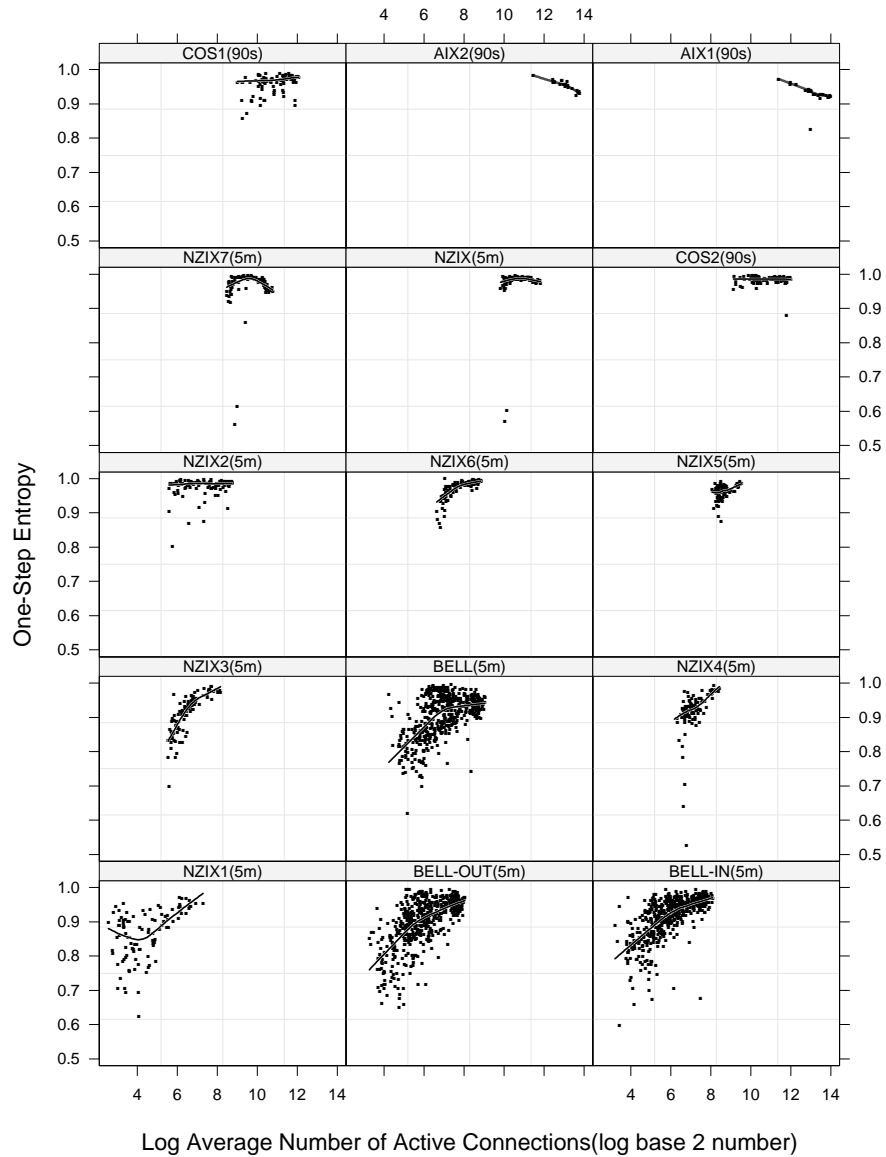


Figure 6: One-step entropy for the normalized log inter-arrivals is plotted against log average number of active connections. This modified out-of-box summary tool is described in Section 2.4.

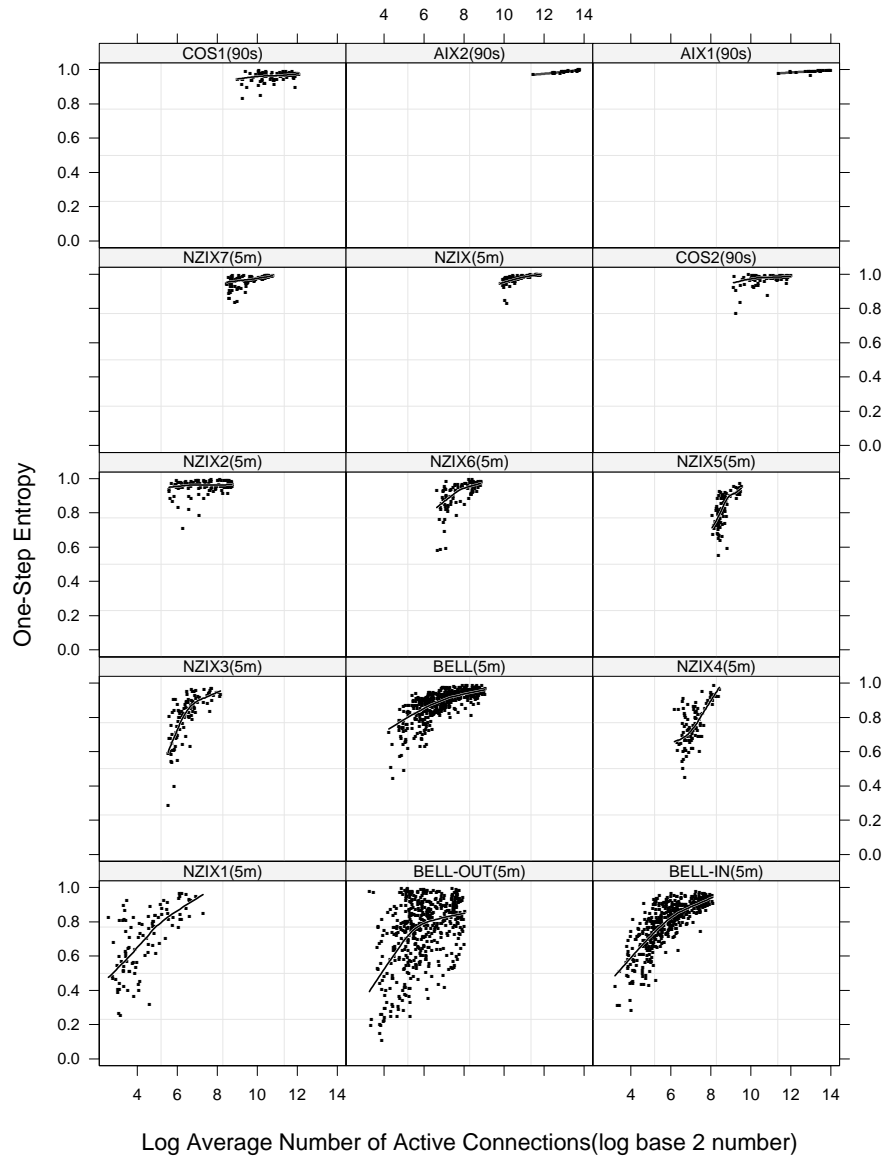


Figure 7: One-step entropy for normalized the packet sizes is plotted against log average number of active connections. This modified out-of-box summary tool is described in Section 2.4.

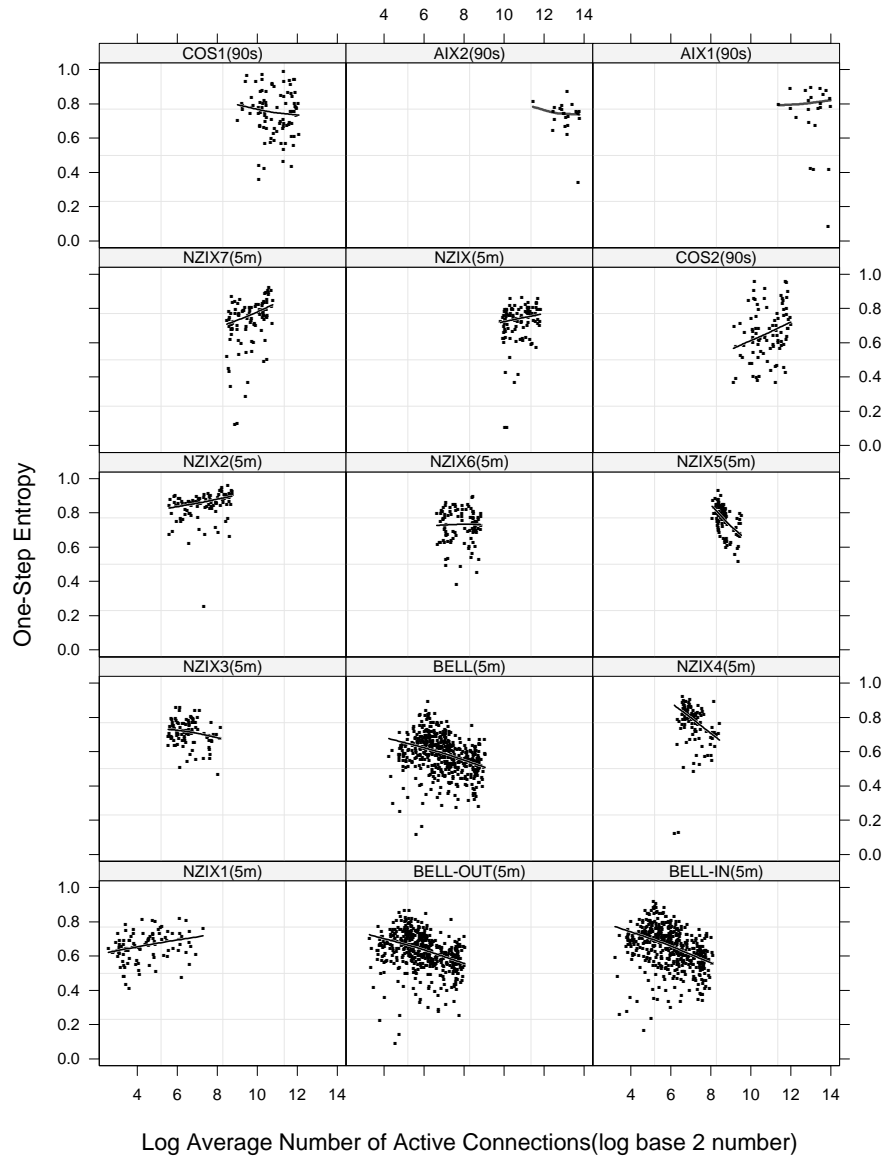


Figure 8: One-step entropy for the normalized log 100-ms packet counts is plotted against log average number of active connections. This modified out-of-box summary tool is described in Section 2.4.

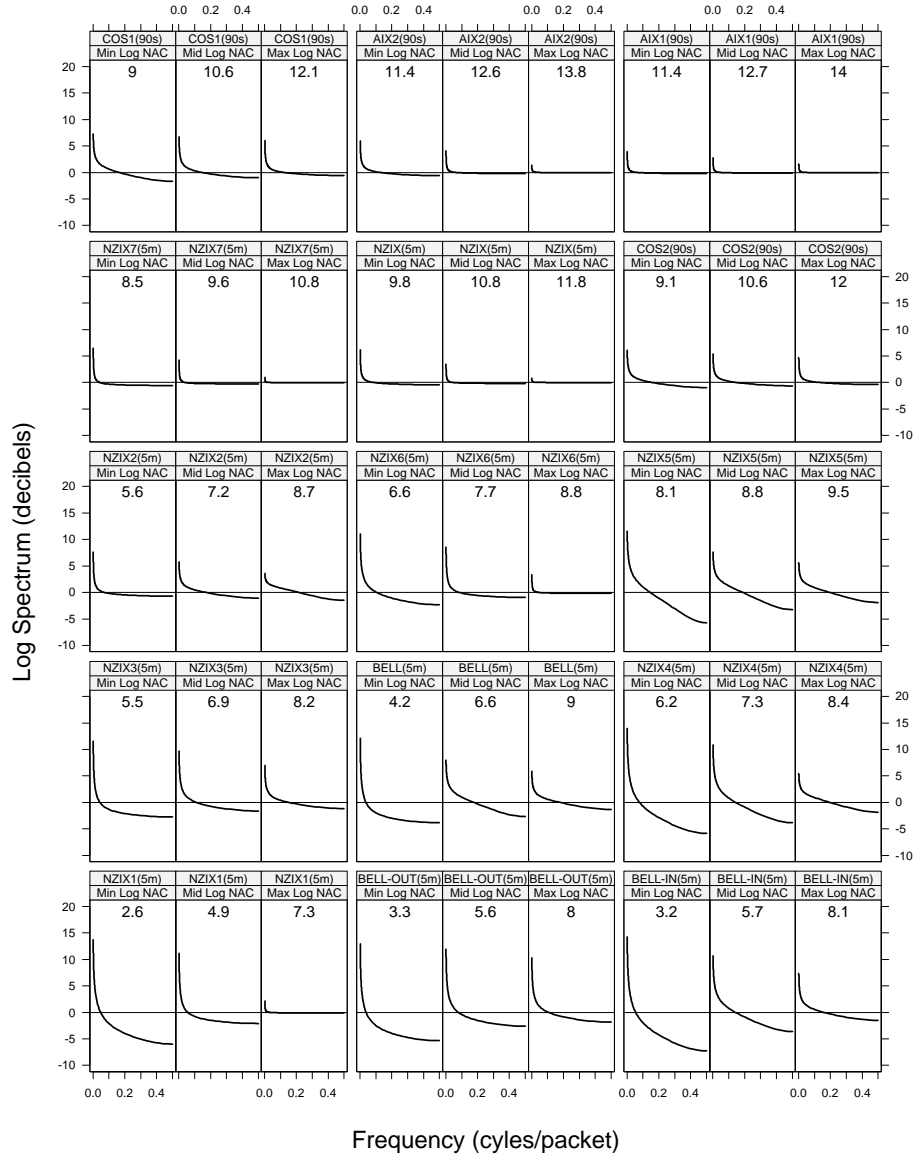


Figure 9: For each of the 15 links, the power spectrum for the normalized packet sizes is plotted against frequency for 3 traces. This roll-your-own comprehensive tool is described in Section 2.4.

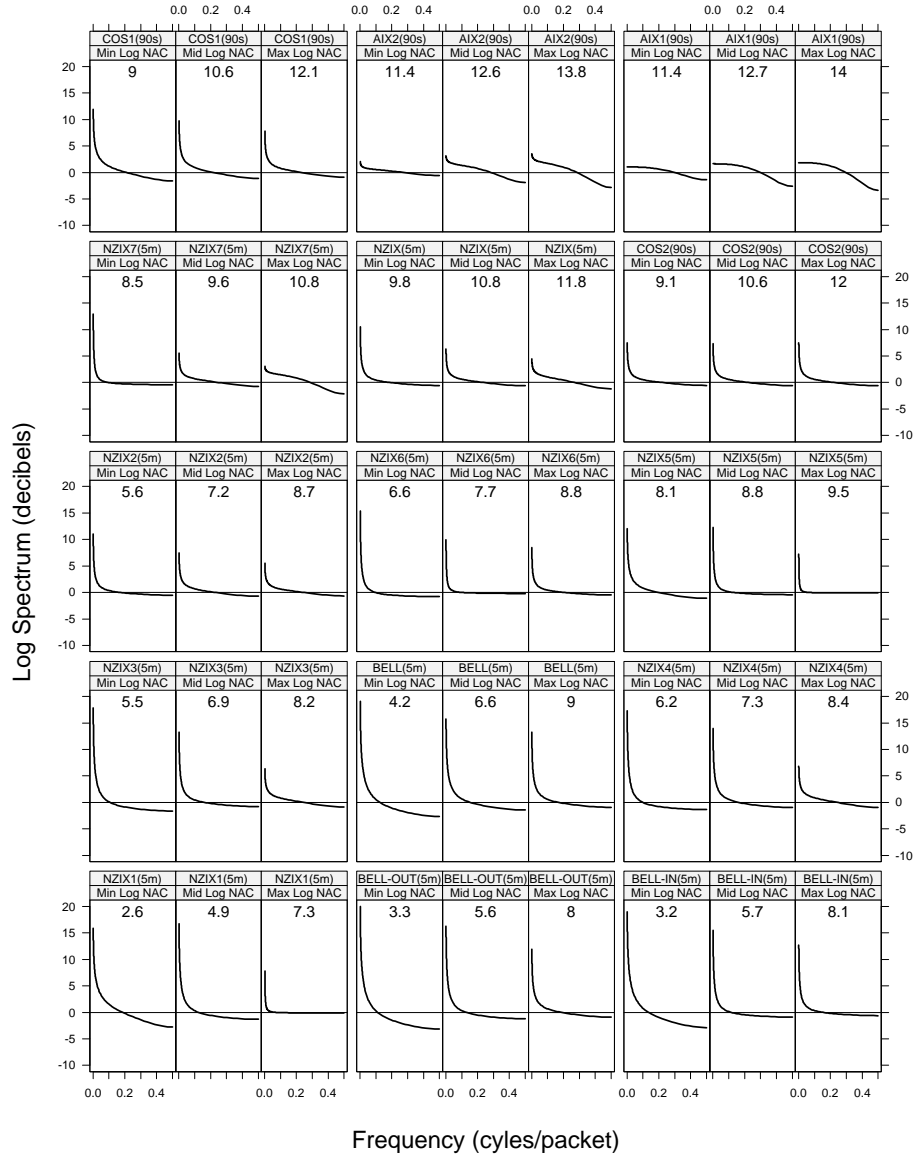


Figure 10: For each of the 15 links, the power spectrum for the normalized log inter-arrivals is plotted against frequency for 3 traces. This roll-your-own comprehensive tool is described in Section 2.4.

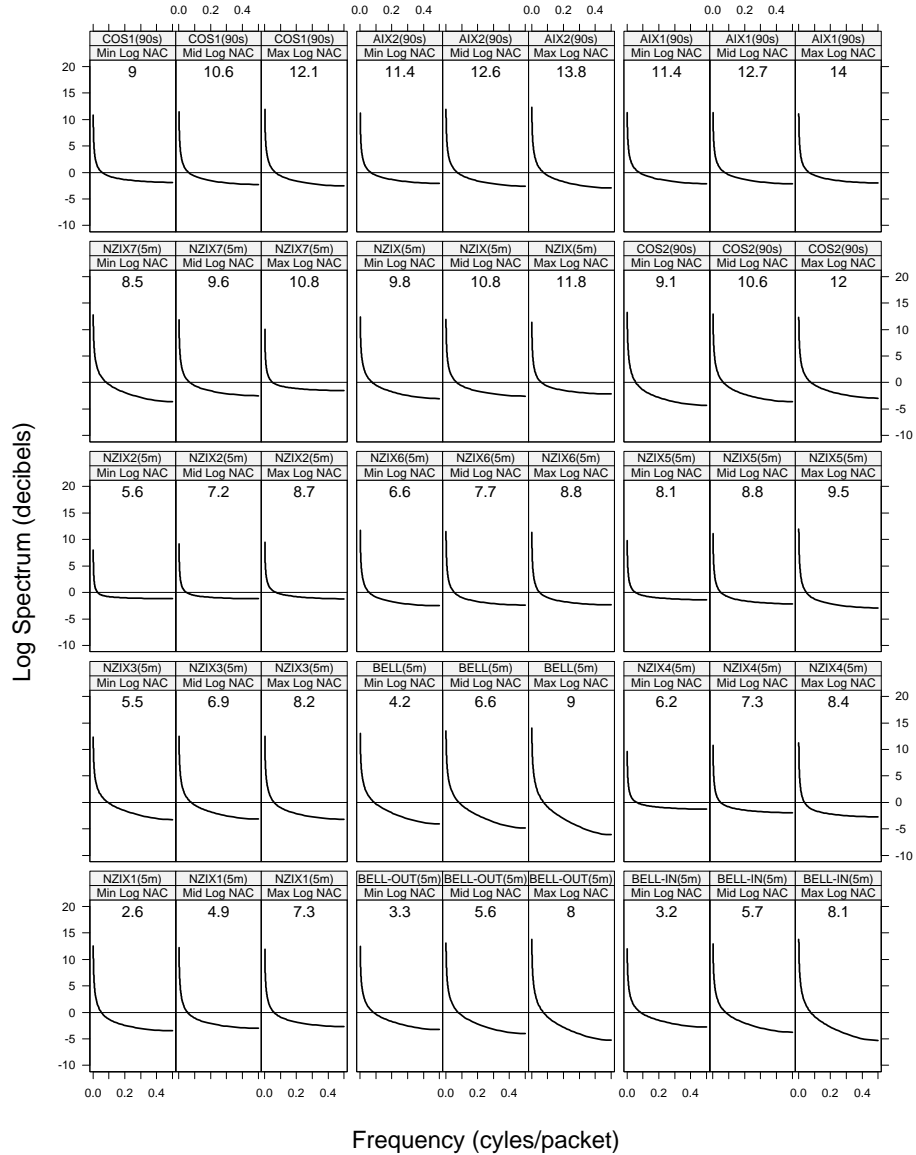


Figure 11: For each of the 15 links, the power spectrum for the normalized log packet counts is plotted against frequency for a set of 3 traces. This roll-your-own comprehensive tool is described in Section 2.4.

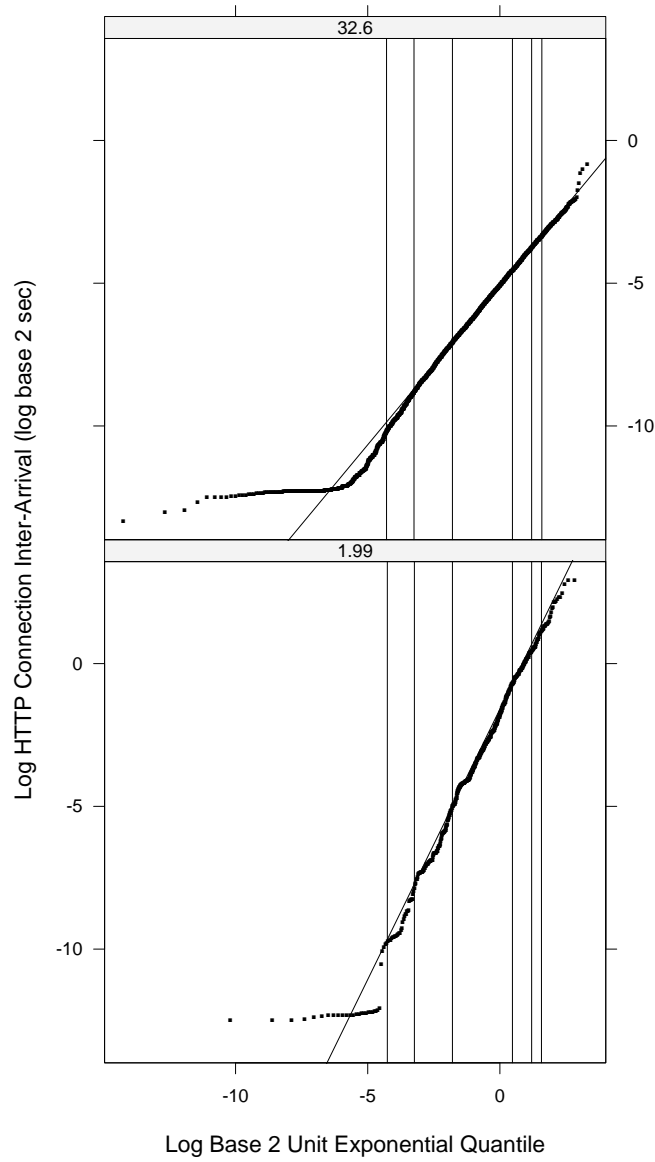


Figure 12: Quantiles of the log HTTP connection inter-arrivals are graphed against log quantiles of an exponential for two BELL(5min) traces. This out-of-box comprehensive tool is described in Section 3.2

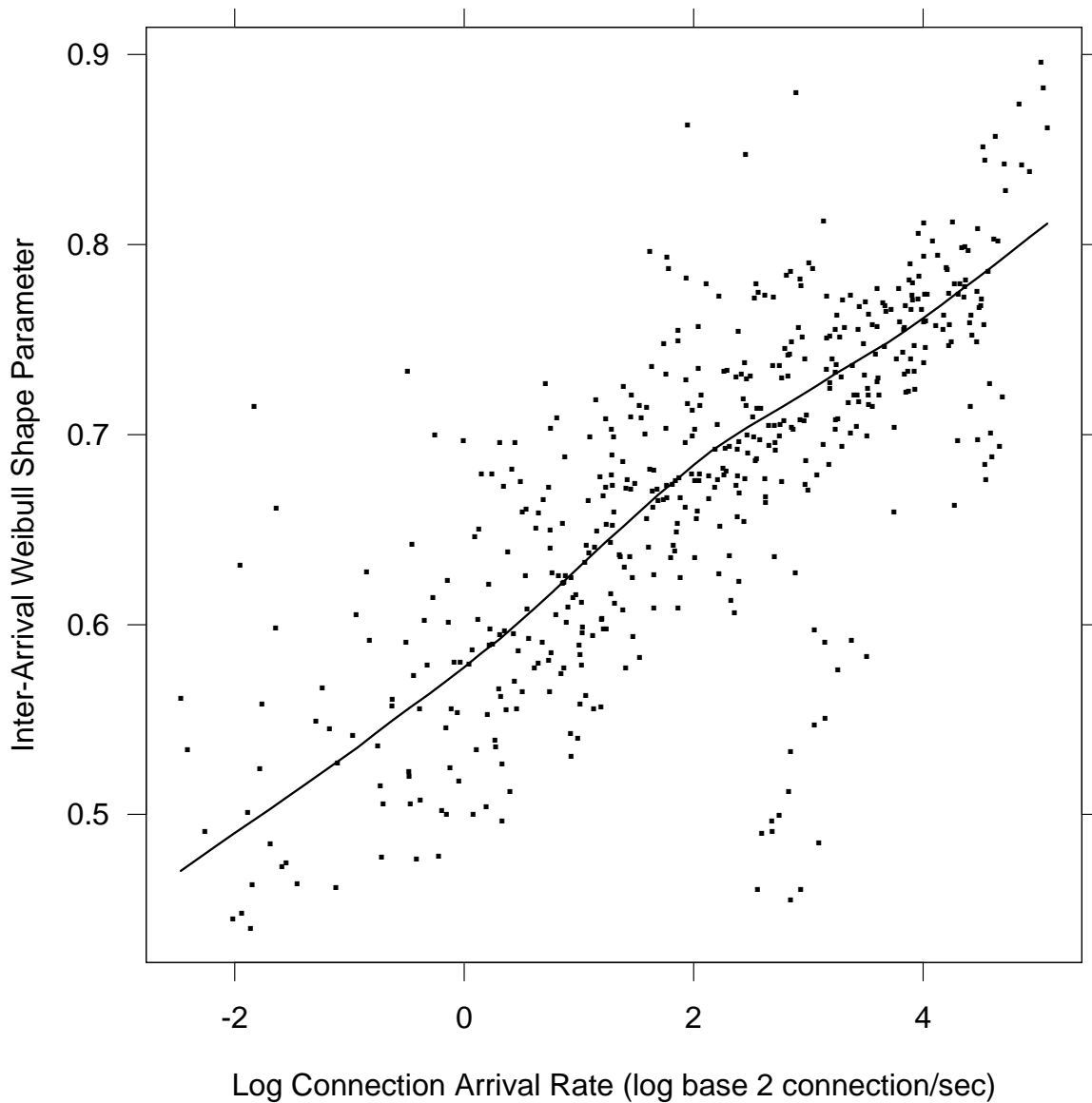


Figure 13: Estimates of the Weibull shape parameter are graphed against log number of new connections per second for the BELL(5min) traces. This out-of-box summary tool is described in Section 3.2.

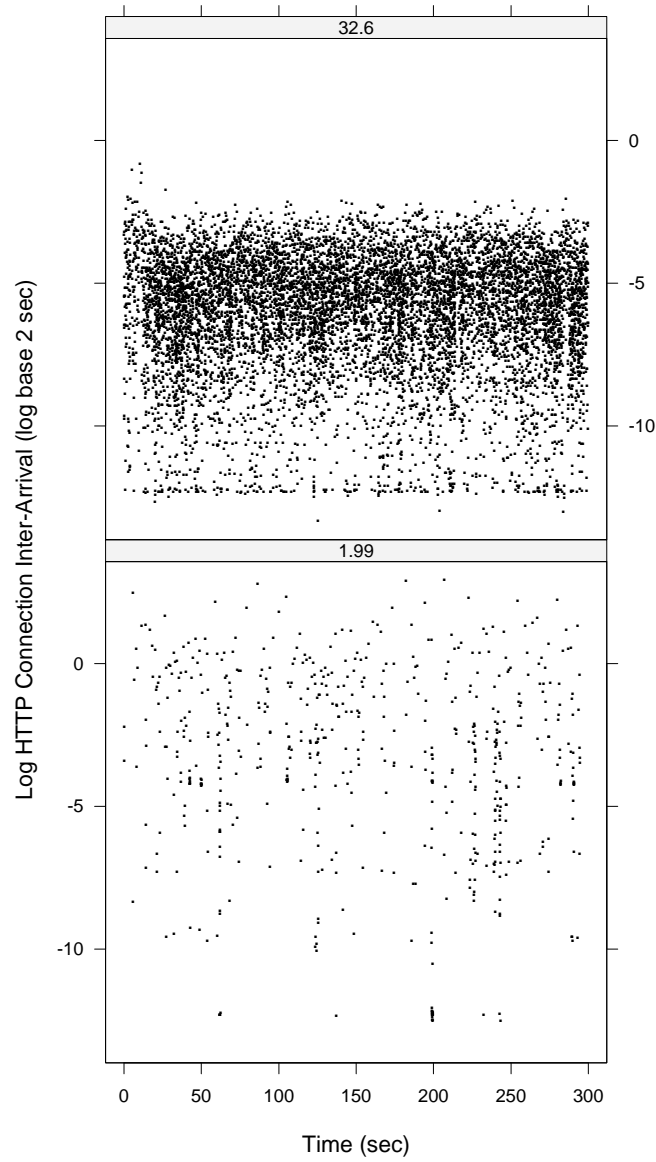


Figure 14: Log HTTP connection inter-arrival time is graphed against arrival time of the first packet of the pair that forms the inter-arrival time. This out-of-box comprehensive tool is described in Section 3.3.

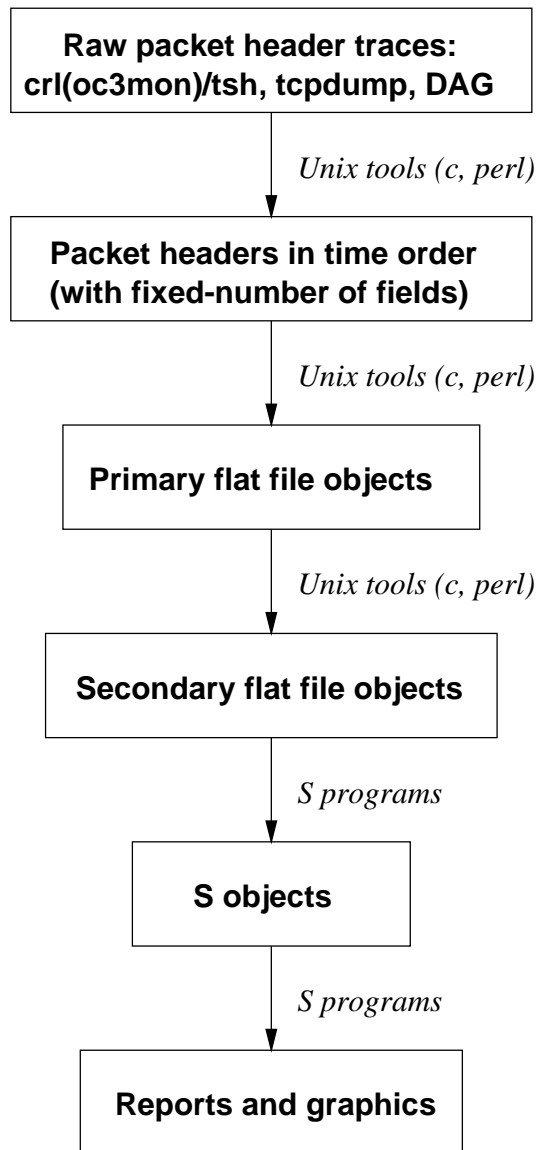


Figure 15: Data Flow in the S-Net environment.