Measurement and Modelling of Self-Similar Traffic in Computer Networks

Tony Field, Uli Harder & Peter Harrison[1]

Department of Computing

Imperial College London

Huxley Building

180 Queen's Gate

London SW7 2AZ

England

Abstract

We report results from the analysis of traffic measured over a departmental switched Ethernet. Self-similar characteristics are seen throughout the network, for example at the compute servers, web server and intermediate routers. We show that data shipped by the web server (i.e. including both static files from a filer server and dynamically-generated data) has a heavy-tailed distribution which is matched extremely well by a Cauchy distribution. We also show that the fragmentation of the data (i.e. into Ethernet frames) leads to a departure process whose power spectrum is shown to follow a power law very similar to that of the observed traffic. Importantly, the power law appears to be largely independent of the input process - self-similar behaviour is observed even with Poisson arrivals. This supports the suggested link between file/request size distribution and self-similarity in network traffic. The resulting implication that self similarity and

[1]e-mail: ajf@doc.ic.ac.uk, uh@doc.ic.ac.uk, pgh@doc.ic.ac.uk

heavy tails are primarily due to server-nodes, rather than being inherent in offered traffic, leads to the possibility of using conventional queueing network models of performance.

1. Introduction

The statistical characteristics of network traffic has been of interest to researchers for many years, not least to obtain a better understanding of the factors that effect the performance and scalability of large systems such as the Internet. Early studies of Internet traffic proved particularly interesting as they exposed self-similar characteristics that were not previously commonplace[1, 2]. There are many papers debating the reason for the apparent self-similarity in modern communications traffic. These range from ON/OFF models of heavy-tail distributions [1, 2], file size distributions in file systems and web servers [3, 4], user behaviour, network protocols, back-off algorithms in the Ethernet [5], buffers in routers and the TCP congestion avoidance algorithms [6, 7, 8, 9, 10].

Like many previous studies we have analysed our own local intranet (a switched Ethernet system in a University Computing department) with a view to understanding better the nature of the traffic in terms of the underlying distributions, correlations, power laws etc. However, our long-term objective is to build analytical models of communication networks, including the external and internal processes which govern their behaviour. We therefore seek also to explain observed behaviour, in particular the root cause of self-similarity in the network traffic.

This paper contributes to the body of knowledge on self-similarity and network traffic modelling in three respects:

- We confirm that the sizes of files shipped by file servers and web servers have heavy-tailed distributions but propose a Cauchy distribution as a better approximation to those file sizes than the family of Pareto distributions previously suggested, particularly for small files.

- We show, via a simulation model of the fragmentation process that packages file content into Ethernet frames, that the departure process from a file/web server with the above heavy-tailed characteristics, follows a power law and that this closely resembles the power law observed in real networks.
- We demonstrate, through the same model, that this self-similar behaviour is induced by the server, rather than the arrival process that feeds it. For example, we show that self-similarity is observed in the output process even when the arrival process is Poisson.

This last observation is particularly interesting in the sense that it opens up the possibility of using product-form queueing network models (specifically those with arbitrary service times and processor sharing) as approximate models of real networks with self-similar traffic.

In addition, we report our experiences with using `/proc/net/dev` as an alternative to `tcpdump` and demonstrate some interesting statistical properties of the packet rate changes at different points in the network.

The paper extends our earlier work [11] in which the web server traffic was monitored *indirectly* as it left the router, rather than at the server itself. Here we employ new data gathered directly at the server that enables us to study more accurately the arrival and departure processes at the web server.

The measurements we have performed are cheap and easy to reproduce at other sites. The reader is referred to [12] which makes available all the data used in this paper, along with other data which may be useful to those conducting similar research.

In the remainder of this paper Section 2 discusses the mechanisms we employed for real-time data capture. Section 3 describes the network architecture and summarises the techniques used to monitor the network traffic. Section 4 describes the analysis methods that have been applied to the captured traces and Section 5 presents the results of the various analyses. A simulation model of the Ethernet packetisation process at a web server output link is detailed in Section 6 together with a discussion of the observed behaviour of the simulated link output. The possible role of queueing models in the analysis of internet components is discussed in Section 7 and the paper's conclusions and ideas for future work are given in Section 8.

2. Data Capture

We explored two approaches to data capture: `tcpdump` which traces traffic at the packet level and the file `/proc/net/dev` which maintains aggregate traffic counts.

2.1. `tcpdump`.

The `tcpdump` [13] utility attaches itself to the network socket in the Linux kernel and makes a copy of each network packet that arrives at the network interface. To be more precise it allows all Ethernet packets arriving at the kernel to be logged. We have used the program to obtain, for every frame that is transmitted:

- A timestamp indicating when the packet reaches the kernel;
- Source and destination IP addresses and port numbers;
- The size of the frame (only the user data is reported, the headers for various protocol layers have to be added on to recover the actual size of the frame);
- The traffic type (`tcp`, `udp`, `icmp`, etc).

The timestamp is the time the kernel first "sees" a packet rather than the time it seen by the Ethernet card.

2.2. `/proc/net/dev`. The Linux operating system keeps track of the number of packets and bytes sent and received in a virtual file called `/proc/net/dev` using counters. The counters get reset at machine reboot; the values are modulo 2^{32}. We have used a `PERL` program to query this file at regular intervals of 1 second and record the value of the counters and a time stamp. Occasionally, in less than 0.1% of all cases, this process fails to record the counter within 1 second, because the program is run in user space. In the processing phase we then linearly interpolate the missing values and obtain a time series of the counter values for every second. This method has three advantages over `tcpdump` : it usually requires no special operating system privileges, it can be run for a much longer time as only summary data is collected, and it incurs lower intrusion overheads. The tradeoff between resolution and efficiency, both space and time, is addressed later in this paper.

All raw data discussed in this paper has been anonymised and is freely available for inspection and use in other research—see [12] for the url.

3. Network Architecture and Monitoring

The monitored system is the Department of Computing's internal Ethernet system at Imperial College London. We focus on three components of this network for the purposes of this paper: The central router, which connects the network to the outside world, an arbitrarily chosen CPU server (named MOA), and the departmental web server which services both internal and external web page requests.

3.1. **Router.** The department is connected to the Internet via a Black Diamond router from Extreme Networks [14]. This is used as both a top-level switch for the internal Ethernet and as a router for all external traffic. Those two functions are separate. All packets going to, and coming from, external locations are duplicated at media access control (MAC) level to the second network interface card (NIC) of a dedicated monitoring machine. Log files are transferred to a different machine (GAUSS) for analysis.

We measured the router network traffic on 10 April 2003 between 15:45 and 16:45.

3.2. **CPU server.** On one of the departmental CPU servers, MOA, we ran the `/proc/net/dev` monitor for twelve days at a measurement interval of one second. The measurement started at 1 February 2002 at 16:24:47. The majority of the traffic generated by the CPU servers is targeted at the internal file and web servers.

3.3. **Web Server.** We ran `tcpdump` on the web server to get information on the request arrival process over the same observation period as that of the router. The web server itself additionally provides information on the individual web requests, in particular bytes shipped per request. Data from the server is sent to the outside world via the router (Black Diamond) and individual (www) packets are monitored as they pass through the same router. We thus obtain the measured request size distribution and a time series representing the instants the corresponding stream of Ethernet packets pass through the router.

4. Analysis Methods

We concentrate here on the point processes formed by packet departure events happening at times t_i, $i \in I \subseteq I\!N$. An event that occurs at time $t \in I\!R$ is called E_t.

We assume that there are $n \in I\!N$ events (or observations of events), the first happening at t_1 and the last one at t_n. The observation period may begin before the first event and end after the last, so we define it to be $T = [t_0, t_{n+1}] \subset I\!R$ with $t_0 \leq t_1 \leq \ldots \leq t_n \leq t_{n+1}$, for arbitrary t_0, t_{n+1} bounding the set of event-instants.

The inter-event times, Δt_i, $1 \leq i \leq n-1$, are defined as

$$\Delta t_i = t_{i+1} - t_i$$

For a finite observation period of a point process we can easily generate a histogram that, when correctly scaled, approximates the probability density function (pdf) of inter-event times.

4.1. **Power Laws.** The probability density function $p(x)$ is said to follow a power law if

$$p(x) \propto x^{\gamma}$$

as $x \rightarrow \infty$, for $\gamma < -1$. When investigating the existence or otherwise of a power law we use exponentially growing bin sizes for the histograms. Apart from the histogram, we also compute the mean and variance of the inter-event times which are useful for distribution fitting.

4.2. **Aggregation.** Starting from a point process one can investigate the behaviour of the corresponding *aggregated* time series. The observation period T is divided into N contiguous intervals of size $T_N = T/N$. In each of these intervals, we count the number of events or, if appropriate, we aggregate the properties of the events. So the time series consists of N values

$$a_i = \Big|\{E_t \mid t_0 + iT_N \leq t < t_0 + (i+1)T_N\}\Big|.$$

for $i = 1, 2, \ldots, N$. Sometimes it is preferred to use the quantity $A_i = a_i/T_N$.

4.3. **Self Similarity.** An aggregated time series can be subjected to many analyses, one of which is its *scaling behaviour*. This is sometimes known as "testing self-similarity". Two of the first investigations of the statistical nature of network traffic were [1, 2]. The authors found evidence that the observed traffic was distinctly non-Poisson and thus not amenable to conventional traffic modelling techniques. Amongst others they used Hurst's rescaled range statistic R/S to analyse long-range dependence of the data [15, 16]. There are many other statistics that can be used to investigate long-range dependence (LRD). A good review of the estimators and their relationships can be found in [17, 18, 19]. Other methods for investigating the correlation of the data are the log-variance plot, [15, 16, 20] detrended fluctuation analysis, wavelet transformations [21],the Fano factor and the Allan factor [17, 18, 19, 21].

In this investigation, we will use the *power spectrum* to analyse the correlation of the monitored data and inter-event histograms to analyse the inter-arrival time distribution.

If one assumes that a time series is generated by a stochastic process, such as Brownian motion, for example, one can investigate the distribution of the *changes* in the time series values a_i, $i = 1, 2, ..., N$, from one time step to the next, i.e. the differenced time series $\{\Delta a_i = a_{i+1} - a_i \mid 1 \leq i \leq N - 1\}$. Alternatively one can work out the distribution of the $a_i, 1 \leq i \leq N$ as they are the changes of the underlying counting process. Depending on the statistical nature, one also expects a different kind of scaling law for these distributions with respect to the length of the aggregation interval T_N.

4.4. **Power Spectrum.** For a time series $X(t)$ with zero mean the power spectrum (density) is defined as the Fourier transform of the auto correlation function. By the Wiener-Khintchine theorem [22, 23], under certain assumptions, the power spectrum is the same as the absolute square of the Fourier transform of the time series signal[2] which can be efficiently calculated using, for instance, the Fast Fourier Transformation. For details we refer the reader to [24]. The power spectrum, or spectral density, shows the distribution of the signal strength at different frequencies. Note that this expression can be discretised for discrete time series. Since the point process tends to be rather sparse it is better to use an aggregate time series of counts. In our investigation we have used 10ms bins for the aggregation in line with previous research [1, 2]. Again one is looking for power laws where the power spectrum $S(f)$ behaves like $S(f) \propto 1/f^{\alpha}$, where f is the frequency. The exponent α turns out to be 0 for white noise and 2 for a Brownian motion.

Because the autocorrelation function is the inverse Fourier transform of the power spectrum, $1/f^{\alpha}$ noise corresponds to the following autocorrelation function

$$C(\tau) \propto |\tau|^{\alpha-1} \quad \text{for } 0 < \alpha < 1 \quad \text{see [24]}$$

where $\tau \in I\!R$. It follows that an exponent α close to but smaller than 1 implies long term correlations.

The power spectra were computed using standard methods published in the *Numerical Recipes in C* [25] with overlapping windows to make them a consistent estimator. In addition we averaged the values of the power spectra in logarithmic intervals to reduce the noise that is usually seen for higher frequencies.

When interpreting a power spectrum it is useful to keep in mind that the rightmost (highest) frequency is determined by sampling rate. This is also known

[2]We should note that this is not a very rigorous way of defining the power spectrum, as the time series has to fulfil certain criteria for the integral to be well defined, for example.

as Nyquist frequency. The leftmost (lowest) frequency is determined by the length of the observation period.

5. Measurements

5.1. **Router Traffic.** The outgoing and incoming data of the router are collected independently. We investigate the nature of the entire in and outgoing traffic and also the traffic caused by external requests made to the departmental web server; we will use this later in our network traffic model. As we can see in figure 1 there is no power law in the inter arrival time distribution. It is also clearly non-exponential as the coefficient of variation of the inter arrival times is 1.8 for the overall traffic and the incoming web requests and 3.8 for the outgoing web server traffic.

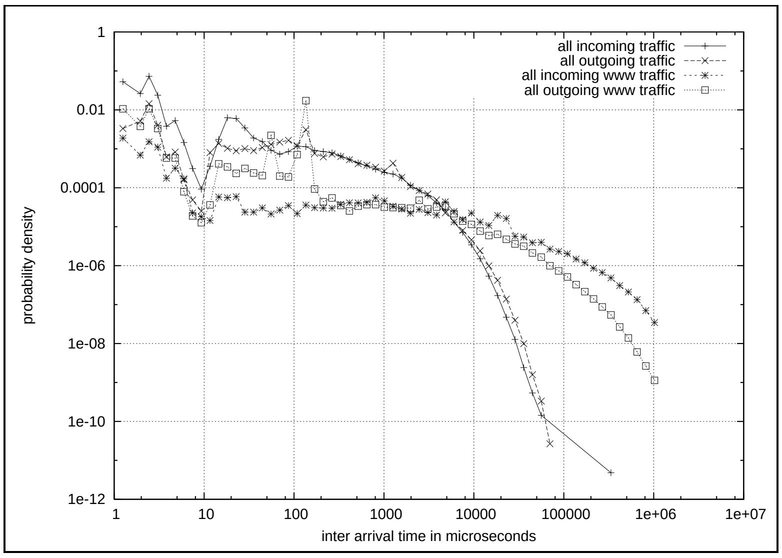

Figure 1. The inter event time histogram for the outgoing network traffic on a log-log scale. The measurements were taken on the router using `tcpdump`. Note that a heavy-tailed distribution would appear as a straight line on a log-log plot such as this.

Although there is no evidence of a power law in the inter-event time distribution (arrival process) there is strong evidence of a power law in the power spectrum of the network traffic, suggesting that something other than the inter arrival times is causing long-range dependence in network traffic, see figure 2. The total traffic shows a power law from approximately $10^{0.5}$ to $10^{-2.5}$ Hz, equivalent to a frequency range from 1/3 seconds to 6 minutes. The outgoing traffic of the web server covers the same range with a similar gradient of -1, but the incoming requests to the web server show far less long-range dependence. They start to flatten out to white noise below $10^{-1.0}$Hz or 10 seconds. Overall the web server

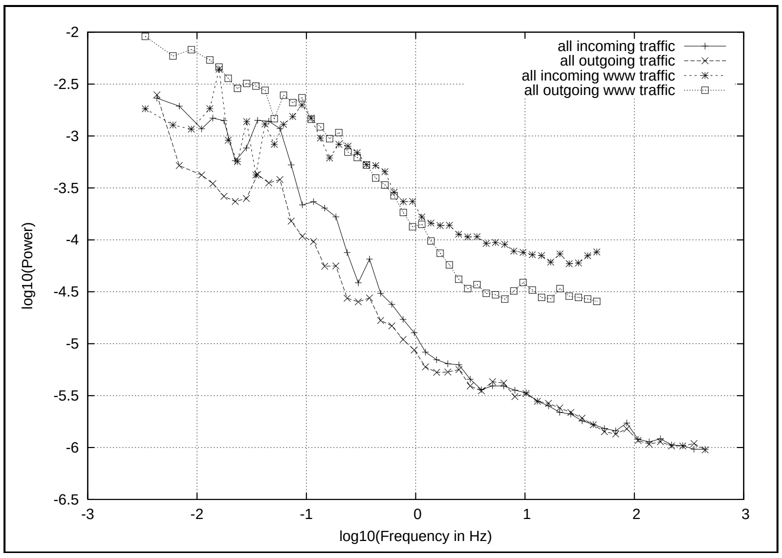

FIGURE 2. The power spectrum of the router network traffic time series measured in packets per second, aggregated over either 0.01 or 0.001 second intervals. These measurements were taken on the router using `tcpdump`.

traffic is about 1% of the total network traffic.

5.2. **CPU Server.** The power spectrum of a twelve-day observation period (figure 3) shows peaks corresponding to the daily cycles ($86,000 \approx 10^{4.93}$seconds or

$10^{-4.93}$Hz), and also has peaks between one and two hours ($7,200 \approx 10^{3.85}$seconds or $10^{-3.85}$Hz and $3,600 \approx 10^{3.55}$seconds or $10^{-3.55}$Hz), an interval that will correspond to typical session lengths of student users during term time. The main interest lies in the plots which show the *changes* (defined in section 4.2) in the observed packet rate. We see a distribution that is distinctly leptokurtic and

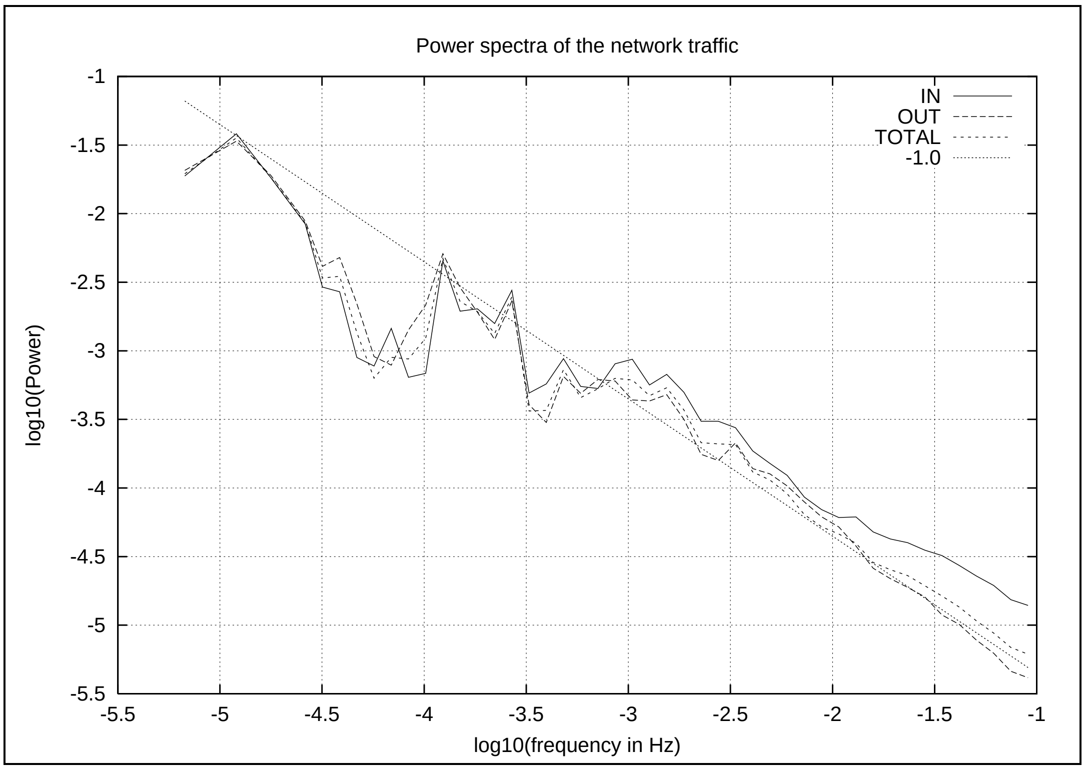

FIGURE 3. Power spectrum of the network traffic time series measured in packets per second, aggregated over 5 second intervals.

certainly non-Gaussian, see figures 4, 5 and 6.

In fact figure 5 shows that the asymptotic power law has a gradient of -2, and figure 6 shows that the distributions at different aggregation levels (i.e. different bin sizes) fall into one *master* curve when plotted on top of each other, as they

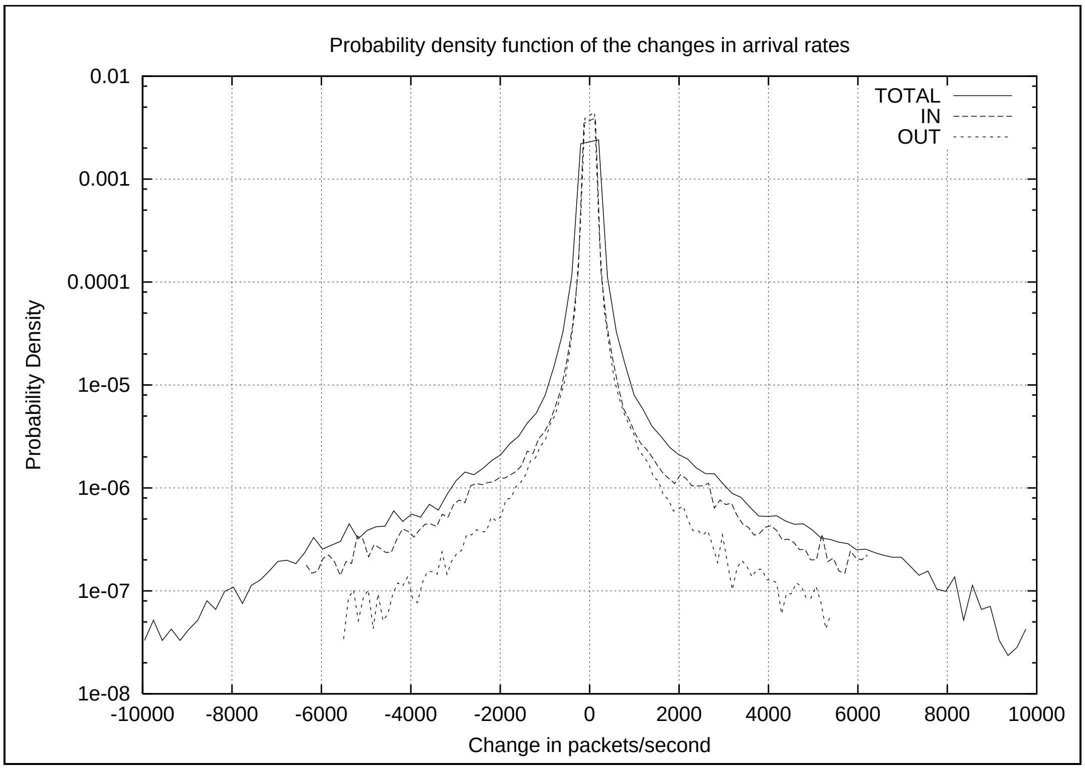

FIGURE 4. Probability density function of the changes in the event rates of packets for the incoming, outgoing and total traffic in a log-linear plot.

should do for a Cauchy distribution[3]. These plots were inspired by related work in the analysis of stock prices where changes in prices have been shown to follow similar patterns, see for example [26, 27].

In order to trace possible causes of this behaviour we next study the characteristics of the file sizes stored at the file server and the request sizes generated by the web server; requests to these two servers constitute the main source of network traffic to/from the CPU server.

5.3. **Web server.**

[3]See p. 92 in Voit's book and p. 71 in Mantegna's book [26, 27] for details of the scaling behaviour of Lévy distributions.

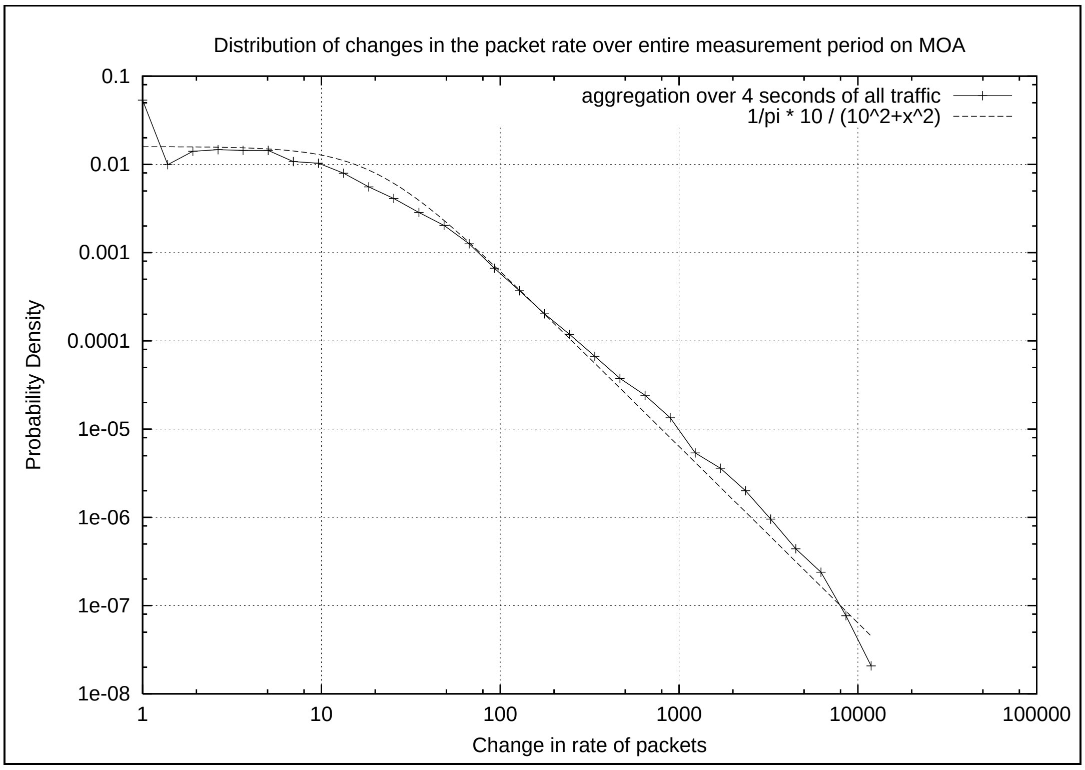

FIGURE 5. Probability density function of the changes in the event rates of packets of the entire traffic compared to a Cauchy distribution in double-logarithmic plot.

5.3.1. *Network Traffic.* We monitored the network traffic at packet level on the web server, compared the inter arrival times of all network packets (i.e. including packets for control flow etc.) and identified those belonging specifically to requests and served documents (figure 7). None of these exhibits a power law. However, the overall traffic, and that caused by the served documents, shows a peak at about 10 microseconds which corresponds to the maximum packets length on a 1Gbps network. The corresponding peak for the minimum size is invisible as `tcpdump` can not resolve times that small.

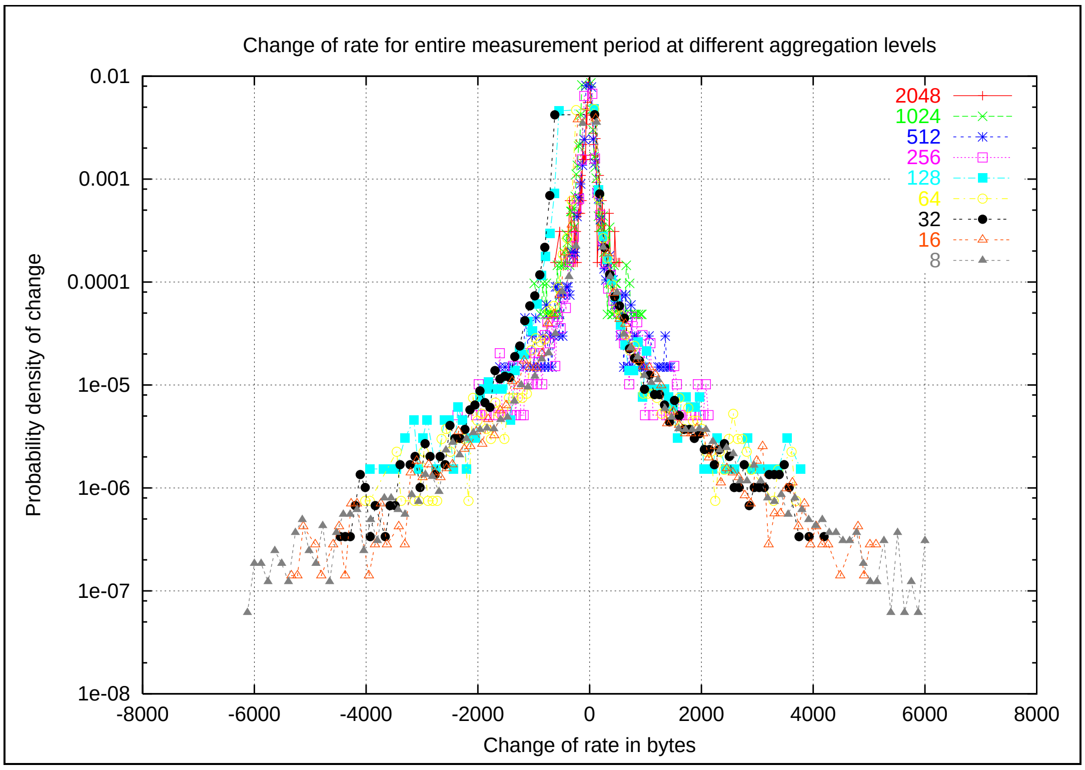

FIGURE 6. Probability density function of the changes in the event rates of packets of the entire traffic at different aggregation levels.

The power spectra (figure 8) look similar to that of the router traffic; again the overall traffic and the served documents show a much stronger power law over a wider frequency range.

5.3.2. *Webserver Request Distribution.* Many authors have repeatedly shown that these distributions follow Zipf's law to a good approximation, which says that

$$P(\text{file or request size} > x) \approx \frac{1}{x}.$$

One pdf that can exhibit this behaviour, and which has been used elsewhere, is the Pareto distribution

$$p(x) = \alpha k^{\alpha} x^{-\alpha-1}$$

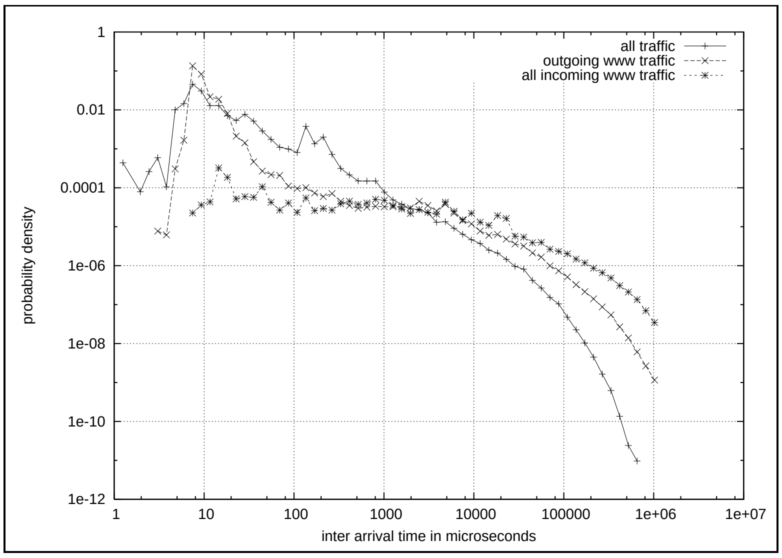

FIGURE 7. The inter event time histogram for the outgoing network traffic on a log-log scale. The measurements were taken on the router using `tcpdump`.

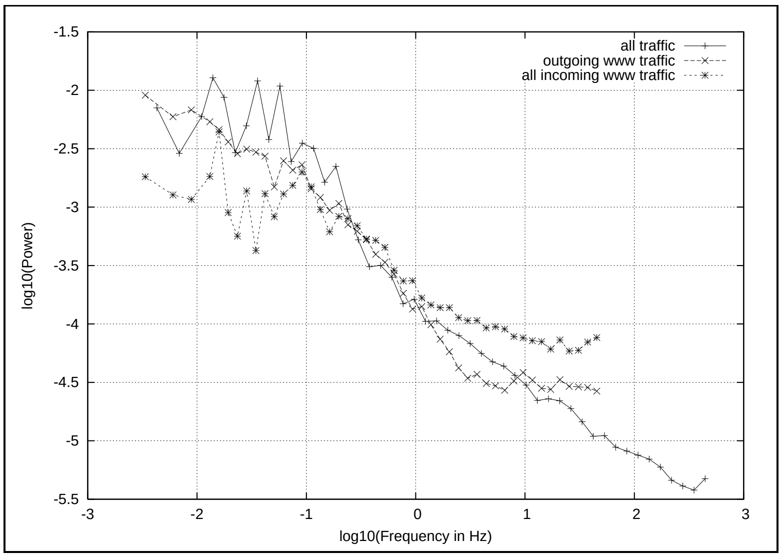

FIGURE 8. The power spectrum of the router network traffic time series measured in packets per second, aggregated over either 0.01 or 0.001 second intervals. These measurements were taken on the router using `tcpdump`.

where $\alpha, k > 0$ and $x \geq k$. If $\alpha = 1$ the Pareto distribution shows the behaviour of Zipf's law for large x. In a double logarithmic plot, this distribution is a straight line with gradient $-(1 + \alpha)$. In [4] the authors use the *log-log complementary distribution* plots to estimate the distribution of request sizes.

To parametrise our model we plotted the pdf of the request size distribution of the requests made to the web server. We plotted the pdf using exponentially increasing bin sizes. The results are similar to those published in [11] where we also showed that file size distributions behave similarly.

These measurements are distinctly non-Pareto but we find that they are extremely well approximated by a Cauchy distribution. The positive Cauchy distribution has a pdf given by

$$p(x) = \frac{2}{\pi} \frac{s}{s^2 + x^2} \tag{1}$$

where $s > 0$. The plot in figure 9 show a Cauchy distribution truncated at the largest file or request and corresponding parameters $s = 4100$ and $s = 29000$ respectively. For those values of s the distributions match the measured mean and variance well. For large x the density behaves like $1/x^2$ so the complementary distribution function (cdf) therefore obeys Zipf's law. The distribution is distinguished from the Pareto distribution by its behaviour for small x. This is easily missed when linearly sized bins are used to create the histogram – hence the value of choosing exponentially scaled bin sizes. Although qualitatively the approximation for small file sizes is not brilliant, it is significantly better than the Pareto distribution. On a log-log plot, this is just a straight line. So, although it can be used to model well requests above approximately 1Kbyte (figure 9) it cannot capture the smaller request sizes at all. This has already been pointed out by Downey [28] who puts forward a very simple and elegant model for file size

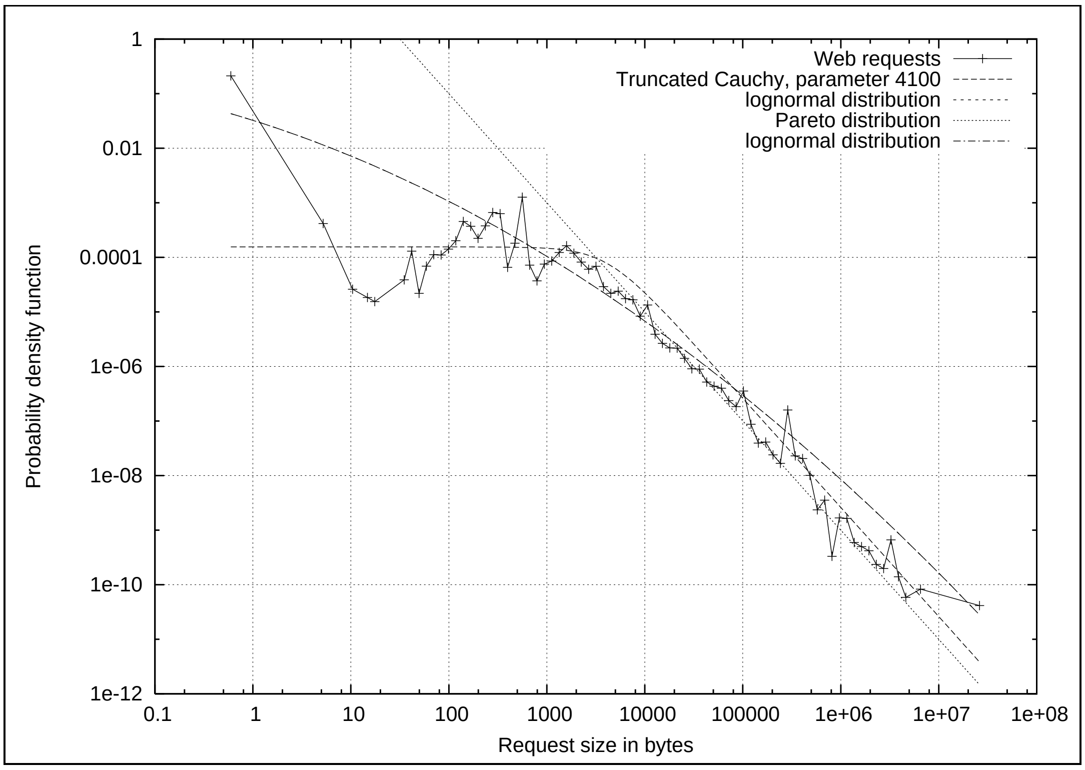

FIGURE 9. This plot compares the request size distribution of external requests on 10 April 2003 between 15:45 and 16:45 to a Cauchy distribution. Though it is clearly not a perfect fit, it describes the nature of the distribution fairly well, including the feature of requests below one kilobyte. In addition, the corresponding log-normal distribution and Pareto distribution are shown.

distributions that has a log-normal distribution. Figure 9 show lognormal distributions superimposed. These were parametrised using the values of logarithmic mean and variance of the observed data.

For requests to the web site, the log-normal model does much worse than the Cauchy distribution.

Recently Marron et al. [29] have developed a model where they use an extension of the Pareto model and overlay up to four Pareto distributions to fit the measured distribution. The numerical results appear to be good, but the fit has to be performed by hand each time. Also, a parsimonious model just having one or two parameters like the Cauchy or log-normal distribution seems more appealing due to its simplicity.

6. Modelling the Traffic

We have now characterised the measured network traffic with its inter-event time histogram, the power spectrum of its aggregated time series and the change in the rates of packets seen on the network.

In this section we describe a simulation study that makes simple assumptions in line with our measurements. We focus attention on the network link at the output of the web server. We model this link using a single server queue where jobs arrive according to a specified arrival process and where the service requirement (network transmission) is distributed according to an exponential or truncated Cauchy distribution.

For the arrival process we have used a deterministic, Poisson or truncated Cauchy process, by first matching the mean number of arrivals. In addition we have sampled the actual arrival time distribution. In all four cases we have used a truncated Cauchy distribution for the service requirements by matching mean and variance. For the Cauchy arrival process we have also ran the simulation with an exponential service requirement.

The truncated Cauchy distribution has a pdf $c(x)$ defined by:

$$c(x) = \begin{cases} p(x)/C & 0 \le x \le x_{\max} \\ 0 & \text{else} \end{cases}$$

where $p(x)$ is given by equation 1 and C is a normalisation constant

$$C = \int_0^{x_{\max}} p(x)\, dx.$$

Note that the truncation of the Cauchy distribution gets rid of its usually prohibiting features like infinite moments.

It is important to understand that we are *not* trying to model a web server here. We are not concerned with multi-threading, process scheduling, disk and CPU accesses etc. caused by a request. Also, we do not take the TCP connection build-up, close down or acknowledgements into account and do not distinguish UDP-based (like most NFS implementations) and TCP-based (like web servers) systems. Instead we focus on the server output where, ultimately, requests are blocked into frames which are added to the input queue of a separate server that represents the network link. Crucially, what we have done is model explicitly the packetisation of requested files into individual frames prior to transmission. In this sense it is *not* a conventional M/G/1 queueing model.

The server (link) removes work from the queue in blocks (Ethernet frames) and waits after each frame for a short period (inter-frame gap). The blocks are all of the maximum size of 1518 bytes or less. The link server is busy proportional to the size of the block and then waits before the next block is processed, similar to the effect of the inter-frame gap on an Ethernet. We assume that all blocks leave the server at a fixed rate determined by the network speed. The model is illustrated in figure 10 and has been implemented in JAVA.

We measure the cumulative queue length at the server to make sure the system is in equilibrium. Each departure from the system is logged and the time series of departures is analysed like those of real network traffic in previous sections. For the power spectra, we aggregated packet counts in 0.001 second bins.

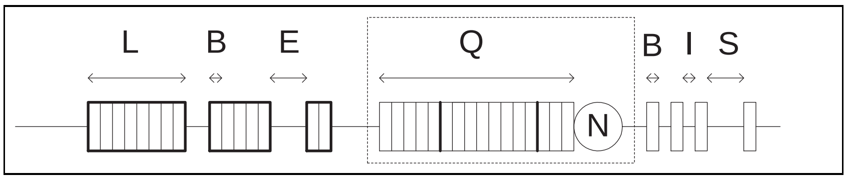

FIGURE 10. This is a sketch of our model. Requests arrive at the left. Their sizes L are measured in bytes and are distributed according to a Cauchy or exponential distribution. Each request is then packaged into blocks (frames) of size 1518 bytes or less. The time between requests E is distributed according to one of our four choices. The server N emits each block after a service time corresponding to the network speed. The minimum time between blocks after service is the inter-frame gap I of twelve bytes. The time between output blocks can be larger than I, namely S if the server was idle. The queue-length Q is measured as the number of blocks awaiting service.

To parametrise the model we used the log file of the departmental webserver covering the period from 15:45 to 16:45 on 10 April 2003. As the reader will recall, we have already presented an analysis of the traffic going through the Black Diamond router. Figure 9 shows the request size distribution for that period and compares it to a Cauchy distribution with $s = 4100$. If we truncate this distribution at 34 Mbytes we get a mean of 23.5 kbytes per request and a standard deviation of 290 kbytes. This compares to a mean 23.5 kbytes and a standard deviation of 554 kbytes with the measurements. We chose this value of s as it fits the mean request size extremely well and produces a standard deviation that is in the same order of magnitude as the observed one. Unfortunately we do not have access to all the logfiles of the departmental webserver. Also the logfiles tell us only when a particular request has been fulfilled, not when it arrived. And to make matters worse this is only logged down to the full second. In order to gain more knowledge about the arrival process of requests we have looked into the `tcpdump` files collected on the webserver and the departmental router. They tell us that in our observation period the mean inter arrival time of requests was 0.101 seconds with a standard deviation of 0.187 seconds. There is no significant difference between the data gathered on the router and the web server.

We ran the simulations 10 times with a simulation time of about one day at a network speed of 1Gbps for different arrival and service process combinations. For the simulation with Cauchy service request and empirical, deterministic and exponential inter arrival time distribution we find that the mean queue-length is [2.1,2.5],[2.1,2.5] and [1.4,2.8] packets respectively. For the combination of Cauchy arrivals and Cauchy service requests the queue length is [3.8,4.2] packets. The by far smallest queue-length arises from the Cauchy arrival time distribution and exponential service requests with [0.0554,.0556] packets.

We analysed the last two hours of each simulation run similarly to the analysis of real network traffic before. To compare our simulation at the network level with the measurements made at the Black Diamond router, we filtered the traffic trace discussed in section 5 for packets originating from the internal webserver which are bigger than one Kbyte.

6.1. **Departure Process.** A log-log plot of the inter-event time histogram is shown in figure 11. This is worthy of some explanation. Both simulation and real traffic show a peak around the time it takes to transmit the maximum possible Ethernet packet, 11.592μseconds (largest frame + inter-frame gap). The slight shift may be caused by inaccuracies of the `tcpdump` program. The peak corresponding to the smallest packet is not visible as the time resolution is insufficient. For very short inter-event times, the real data contains timing errors in the `tcpdump` timings. This has been verified by comparing `tcpdump` timings with those obtained by instrumenting the Linux kernel directly using the GILK tool [30].

Beyond 11.592μseconds the inter-event times correspond to times where the queue empties and hence includes a request inter-event time. Therefore it is not surprising to see that the simulation models produce graphs that resemble the input process in that region.

Superficially the real traffic follows a similar trend with, interestingly, a tail that appears *not* to be consistent with a heavy-tailed distribution. Quantitatively, however, it is evidently not a particularly accurate model of the departure process.

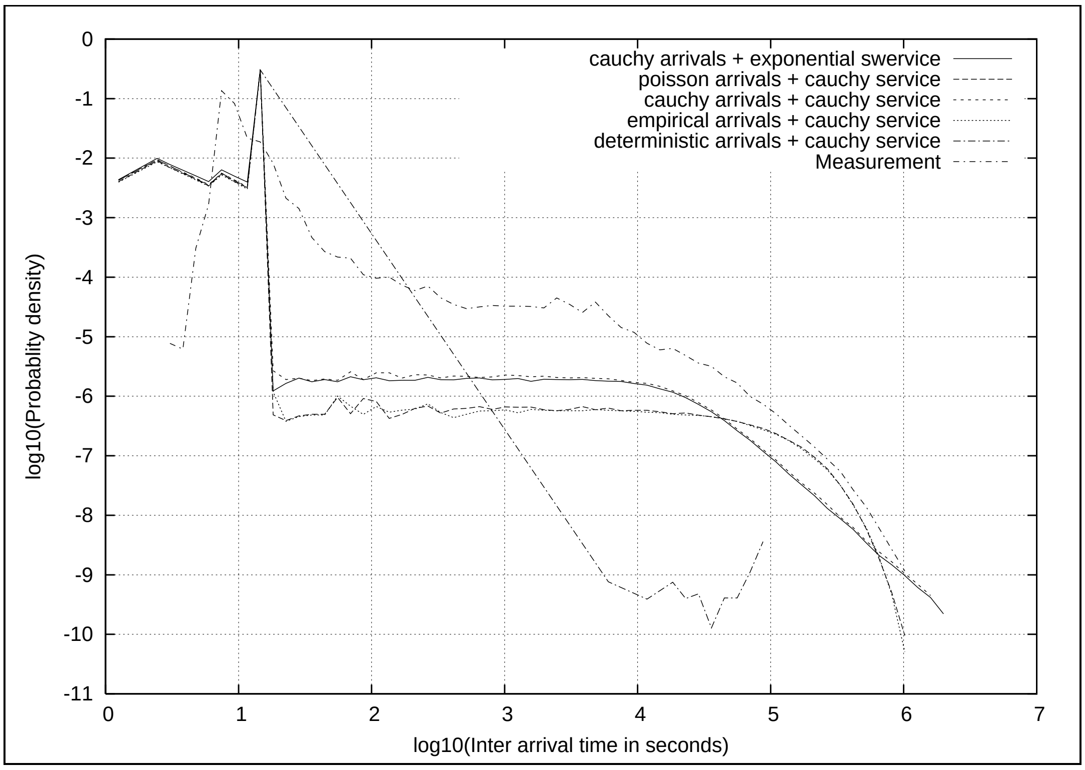

FIGURE 11. Plot comparing the inter event times of the simulation model and the real traffic.

What is particularly interesting, however, is the nature of the power spectra of the departure process for each simulated process combination simulated. These are shown in Figure 12. Four of the five simulation experiments assumed Cauchy (i.e. heavy-tailed) file size distributions. For each of these there is clear evidence

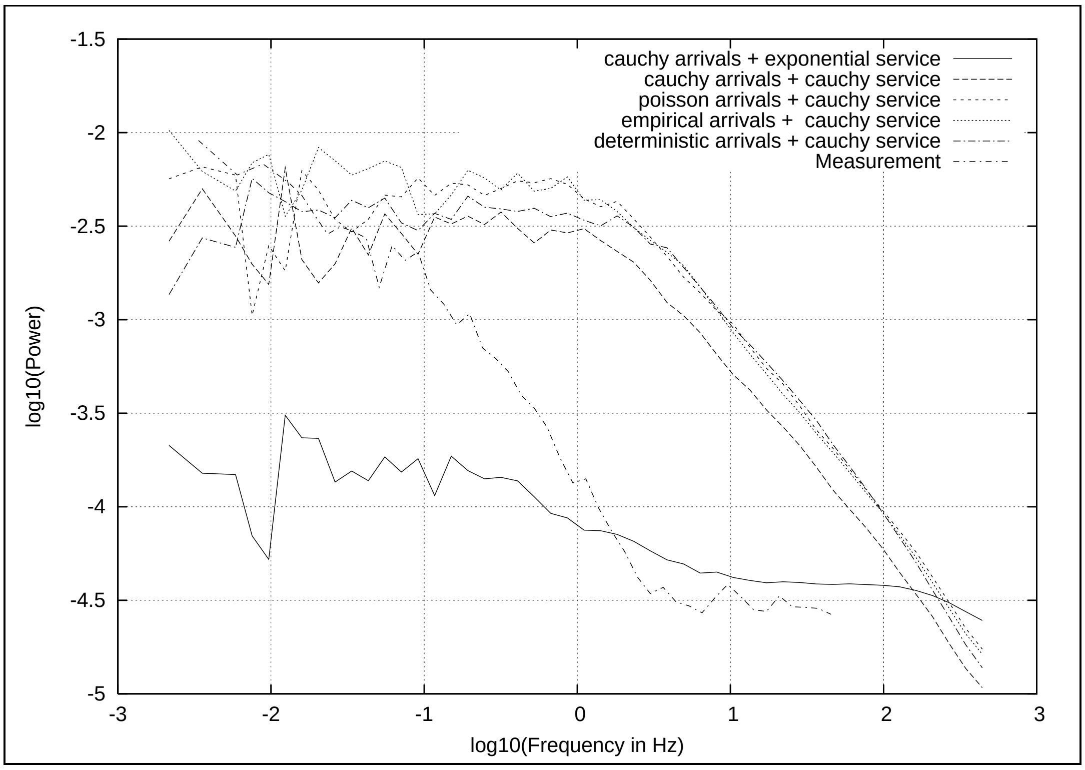

FIGURE 12. The plot compares the power spectra of the departure process of the simulation for different network speeds with that of the real traffic. The real traffic was measured with `tcpdump` on 10 April 2003 between 15:45 and 16:45.

of a power law. Furthermore, the nature of the power spectra are affected very little by the arrival process. The additional experiment assumed exponentially distributed file sizes and for this we see no power law at all. The only way to introduce a power law in this case is to use unreasonable parameters for the maximum inter arrival times of the requests, i.e. assume that there tens of minutes between requests. By and large the simulated traffic is similar to that of the real system, although shifted to the right. Shifts to the right are caused by increasing network speed as the departure events become more frequent and higher x-values correspond to higher frequencies. This suggests that the perceived network of the webserver is slower than the nominal speed of its network connection—a highly probable situation in a real network.

Note also that the simulations show a more distinct power law than the measured data. This is likely to be related to factors that our model has neglected, like CPU, I/O waits of the webserver and competition with other network traffic. Such traffic might also account for the fact that the empirical data would agree more with a 1-10Mbps simulation than with the 1Gbps, even though the webserver has a 1Gbps network connection. The filtered traffic used to parametrise the simulation accounted for about seven to eight percent of the total traffic.

7. Analytical modelling

The comparison between the properties of the traffic generated by the simulation and that observed at the Black Diamond router suggest that file size distribution can be a sufficient cause of power laws, self-similarity and long range correlation. Significantly, this is the case even for Poisson external arrivals. Although we have not tested the hypothesis that external request instants approximate Poisson streams (we have not been able to measure this directly), this has been found to be the case for many decades in a wide range of teletraffic systems. In fact, the hypothesis has sound theoretical foundations. It can be proved that a superposition of arrival processes is asymptotically Poisson under appropriate conditions; for example a large number of independent renewal processes of which none is dominant, i.e. has a much higher rate than the average – see [31, 32] for example.

In any packet-based communication network, the output traffic from a server comprises (a) a sequence of constant-size packets separated by the small inter-frame gap when the server is busy (with *any* work); (b) a null stream, when the server is idle. Put in queueing terms, the output is an on-off process with streamed packet arrivals during the on-periods, which are the *busy periods* of the queue, and off-periods distributed according to the inter-arrival time distribution. Clearly, for heavy-tailed service times, the busy period must also be heavy tailed, although it is hard to calculate precisely the busy period distribution in general.

If we consider the output from the simulation model above, each busy period comprises a consecutive sequence of packet streams, one for each request in the (FIFO) queue. When we measure real data on a network the packets we see are an *interleaving* of the output packet representations of each input request—an artifact of the operation of the servers themselves. If it is indeed the case that requests to the server are approximately Poisson and that the request sizes are approximately Cauchy then the observed packet stream and that produced by the simulation should be approximately the same in terms of these on/off processes.

The instrumentation in our network precludes direct measurement of the output process so it is not possible to measure directly the busy periods at the server output. However, if they can be shown to match the busy period of an M/G/1 queue with Cauchy-distributed service times [4] then a conventional M/G/1 queue with processor sharing would appear to be a good model of the server as a whole. Moreover, if the external arrivals really are well approximated by Poisson processes, and if the nodes' service times are independent, then whole networks would be amenable to analysis using conventional queueing networks with *processor sharing* (PS) queueing discipline. It is because of the interleaving of packet streams, referred to above, that we believe PS discipline to be the most appropriate. In these circumstances, by the BCMP Theorem [33], product-form solutions will exist, from which performance measures like mean queue length, throughput

[4]The fragmentation of requests into frames (packets) introduces a small gap between frames but this adds less than 1% to the busy period.

and mean response time follow via recursive algorithms that can be implemented efficiently. Such models can easily accommodate multiple classes of traffic (with different demands at each node and different paths through the network). The extent to which BCMP queueing networks provide adequate approximate models of networks of this sort is a key next step in our proposed research.

8. Conclusions and Future Work

In this paper we have reported results from the analysis of output traffic at the central router of a departmental switched Ethernet and have investigated the possibility of using queueing models to characterise the behaviour of such networks.

The measurement and analysis reported has shown that the power spectrum of the output process conforms to a power law, consistent with observations reported elsewhere in the literature. However, we have postulated that the output behaviour may be governed primarily by the nature of the service processes involved, rather than the arrival processes. We have built a very simple simulation model of a web server with two important characteristics: firstly, we have used the Cauchy distribution to model the distribution of the sizes of files shipped by the server – this is shown to model extremely accurately the observed file size distribution, for both small and large file sizes; secondly we have modelled explicitly the transmission of these files by the server at the level of Ethernet frames. We have verified that power laws very similar to those observed in practice can be produced by the simulation model and that the power spectra produced are influenced very little by the arrival process. Conversely, we have also shown that this power law disappears when we replace the heavy-tailed (Cauchy) file size distribution with an exponential distribution.

A detailed investigation into the applicability of BCMP queueing networks, in particular an assessment of the accuracy of the processor sharing and independence of nodes' service times assumptions, constitutes a major next step in our proposed research. In practice, of course, we would not always expect service times to be independent, particularly in situations where files are propagated through a series of routing nodes. However, it may be a reasonable approximation. We suspect that processor sharing is likely to provide a very good approximation of the behaviour of the nodes. This is certainly true for web servers and, very likely, file servers which typically operate NFS. These architectures are heavily multi-threaded, with the sharing of the available network resources among the various users being an important design objective.

Finally, we would like to investigate how we can incorporate results in [7, 8, 9, 10], which indicate that the congestion avoidance algorithm of TCP [34] introduces power laws, into to our model. Perhaps combining our findings with a model based on dynamical systems, e.g. [35], will shed more light on the nature of network traffic. In Mathematical and Theoretical Physics self-similarity has been studied widely over the last few decades, motivated by the widespread occurrence of $1/f$ noise in natural phenomena. There are many areas in science where $1/f$-noise has been seen: see for instance Jensen's book [36] for a good overview of the topic. A possible interpretation of $1/f$ noise is the notion of self-organised criticality (SOC) [37], where a critical state of a system is related to an infinite correlation length and the system going through a phase transition.

Acknowledgements

The authors would like to thank the Computer Support Group in helping with the data capture. We would also like to thank Jörn Davidsen of the Chemical

Physics Theory Group (Max-Planck-Institute for Physics of Complex Systems, Dresden, Germany) for fruitful discussions on self-similarity and criticality, and Will Knottenbelt and David Thornley for stimulating conversations.

The research was funded by EPSRC (research grant QUAINT, GR/M80826).

References

[1] Will E. Leland, Murad S. Taqq, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. In Deepinder P. Sidhu, editor, *ACM SIGCOMM*, pages 183–193, San Francisco, California, 1993.

[2] Ashok Erramilli, Onuttom Narayan, and Walter Willinger. Experimental queueing analysis with long-range dependent packet traffic. *IEEE/ACM Transactions on Networking*, 4(2):209–223, 1996.

[3] Gordon Irlam. Unix file size survey. `http://www.base.com/gordoni/ufs93.html`.

[4] Mark Crovella and Azer Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. In *Proceedings of SIGMETRICS'96: The ACM International Conference on Measurement and Modelling of Computer Systems.*, Philadelphia, Pennsylvania, May 1996. Also, in Performance evaluation review, May 1996, 24(1):160-169.

[5] Kensuke Fukuda, Hideki Takayasu, and Misako Takayasu. Origin of critical behaviour in ethernet traffic. *cond-mat/0007435*.

[6] Torsten Huisinga, Robert Barlovic, Wolfgang Knospe, Andreas Schadschneider, and Michael Schreckenberg. A microscopic model for packet transport in the internet. *Physica A*, 294:249–256, 2001.

[7] Andras Veres and Miklos Boda. The chaotic nature of TCP congestion control. *INFOCOM*, (3):1715–1723, 2000.

[8] Hans-Peter Schwefel. Behavior of TCP-like elastic traffic at a buffered bottleneck router. In *INFOCOM*, pages 1698–1705, 2001.

[9] B. Sikdar and K. Vastola. The effect of tcp on the selfsimilarity of network traffic. *Proceedings of the 35th Conference on Information Sciences and Systems, Baltimore, MD, March 2001. http://citeseer.nj.nec.com/sikdar01effect.html.*

[10] D.E. Newman, Nathaniel D. Sizemore, B.A. Carreras, and V.E. Lynch. Growth and propagation of disturbances in a communication network model. *Hawaii International Conference on Systems Sciences*, January 2002.

[11] Tony Field, Uli Harder, and Peter Harrison. Network Traffic Behaviour in Switched Ethernet Systems. *MASCOTS Proceedings*, pages 32–42, 2002.

[12] Measurement data. *http://www.doc.ic.ac.uk/~uh/QUAINT/data/*.

[13] Tcpdump and website. `http://www.tcpdump.org/`.

[14] ExtremeNetworks and documentation. `http://www.extremenetworks.com`.

[15] Benoit Mandelbrot. *The Fractal Geometry of Nature.* Freeman, 1982.

[16] Jens Feder. *Fractals.* Plenum, 1988.

[17] B. Pilgram and D. T. Kaplan. A comparison of estimators of $1/f$ noise. *Physica*, D 114:108–122, 1998.

[18] M.S. Taqqu, V. Teverovski, and W. Willinger. Estimators for long-range dependence: an empirical study. *Fractals*, 3(4):785–798, 1995.

[19] S. Thurner, S. B. Lowen, M. C. Feurstein, C. Heneghan, H. G. Feichtinger, and M. C. Teich. Analysis, synthesis, and estimation of fractal-rate stochastic point processes. *Fractals*, 5(4):565–595, 1997.

[20] Jan Beran. *Statistics for Long-Memory Processes.* Chapman & Hall, 1994.

[21] P. Abry and D. Veitch. Wavelet analysis of long-range-dependent traffic. *IEEE Transactions on Information Theory*, 44(1):2–15, 1998.

[22] N. Wiener. Generalized harmonic analysis. *Acta Mathematica*, 55:117, 1930.

[23] A. Khintchine. Korrelationtheorie der stationären Prozesse. *Mathematische Annalen*, 109:604, 1934.

[24] J. Honerkamp. *Statistical Physics, 2nd ed.* Springer, ISBN 3 540 43020 2, 2002.

[25] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C 2nd ed.* CUP, 1993.

[26] R. N. Mantegna and H. E. Stanley. *An Introduction to Econophysics.* CUP, ISBN 0 521 62008 2, 2000.

[27] J. Voit. *The Statistical Mechanics of Financial Market.* Springer, ISBN 3 540 41409 6, 2001.

[28] Allen B Downey. The structural cause of file size distributions. *SIGMETRICS/Performance*, pages 328–329, 2001.

[29] F. Hernández-Campos, J.S. Marron, G. Samorodnitsky, and F.D. Smith. Variable heavy tailed durations in internet traffic part I: Understanding heavy tails. *MASCOTS Proceedings*, pages 43–50, 2002.

[30] David Pearce, Paul Kelly, Uli Harder, and Tony Field. GILK: A dynamic instrumentation tool for the Linux Kernel. *Modelling Techniques and Tools 12th International Conference, TOOLS 2002 London, UK, April 14-17, Proceedings (ISBN 3 540 43539 5)*, pages pp.220–226, April 2002.

[31] H.J. Larson and B.O Shubert. *Probabilistic models in engineering sciences.* Wiley, New York, 1979.

[32] P.G. Harrison and N.M. Patel. *Performance Modelling of Communication Networks and Computer Architectures.* Addison-Wesley, 1993.

[33] Forest Baskett, K. Mani Chandy, Richard R. Muntz, and Fernando G Palacios. Closed and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, April 1975.

[34] Van Jacobson. Congestion Avoidance and Control. *ACM Computer Communication Review Proceedings of the Sigcomm '88 Symposium in Stanford, CA*, 18(4):314–329, August 1988.

[35] Frank Kelly. Mathematical modelling of the internet. *Mathematics Unlimited - 2001 and Beyond, B. Engquist and W. Schmid (eds.)*, pages 685–702, 2001 Springer-Verlag, Berlin.

[36] H.J. Jensen. *Self-Organised Criticality.* CUP, 1998.

[37] P. Bak, C. Tang, and K. Wiesenfeld. Self organised criticality: an explanation of 1/f noise. *Physical Review Letters*, 59:381, 1987.