Micro-Time-Scale Network Measurements and Harmonic Effects

Mark Carson and Darrin Santay

National Institute of Standards and Technology (NIST) *
carson@nist.gov, santay@nist.gov

Abstract. As network transmission speeds increase, packet streams increasingly uncover fine details of the interior behavior of router hardware and software. This behavior reveals itself as a set of harmonic effects, as interior clocks periodically interrupt packet forwarding, and interior queues periodically empty and fill. We examine this behavior with a Linux-based router employing a variety of gigabit Ethernet interfaces, with a view toward two goals: the creation of harmonic models of router forwarding performance which are accurate and yet mathematically simple and fast; and the analysis of the potential for an undesirable succession of positively reinforcing router "reverberations."

1 Introduction

It has been observed that network behavior differs significantly at a fine (millisecond level) time scale from that seen at coarser (one second or larger) time scales. [1] [2] Now, with transmission speeds reaching the gigabits/second range and beyond, network behavior at a micro scale (microsecond level) becomes relevant.

In the course of performing a high-speed calibration analysis of the NIST Net network emulator [3], we encountered a number of interesting phenomena (independent of NIST Net) in the underlying Linux-based router, which are only apparent at these very short time scales.

In effect, the high-speed uniform packet streams used to perform the analysis act as a sort of "network ultrasound probe," exposing underlying characteristics of the forwarding node. When packets arrive in patterns which correlate strongly with periodic functions such as timer ticks, queue flushing or internal status updating, they may experience unusually low or high latencies, compared with those arriving randomly.

As an analytical tool, then, we can employ a range of such packet probes to uncover these fundamental node "frequencies." Once isolated, these frequencies

* Certain software, equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose. Product names mentioned in this paper are trademarks of their respective manufacturers

can be used to create computationally simple approximate models of node behavior. In this paper, we subject our Linux-based router to such probing and uncover three distinct fundamental frequencies of this sort; we then present analyses and generative mathematical models for these behaviors.

It should be noted that beyond their use as an artificially-created analytical tool, there is evidence that such relatively uniform high-speed packet streams occur naturally. At sufficiently high data rates, not only do interpacket gaps in packet trains [4] shrink, but their variance tends to decrease as well. This is due to a variety of factors: increased use of constant-rate UDP-based packet streams for audio and video transmission and other such uses [5] [6], increased use of persistent TCP connections [7] [8] which decrease the variability due to TCP's slow start, new hardware characteristics, such as *interrupt coalescing*, [9] featured by high-speed network interfaces, and other possible statistical reasons. [10] [11]

Given this situation, there is the possibility that such streams may inadvertently reveal intermediate node frequencies, resulting in increased jitter (latency variance). In the worst case, successive nodes may reinforce this effect, leading to an undesirable network "reverberation." Hence, understanding fine-grained individual node behavior should provide better insight into overall network dynamics.

2 Methods

The data described here were generated using a SmartBits 6000B Performance Analysis System [12]. This device was used to drive uniform loads of up to 1 Gbit/sec through a Linux-based router (Figure 1). Individual packet latencies were then measured with the 10 MHz (0.1 μsec precision) SmartBits timer.

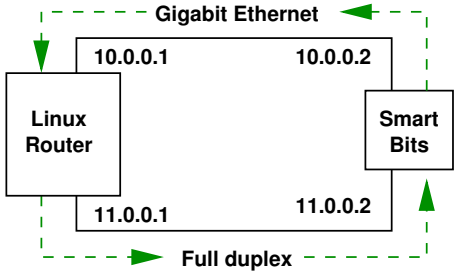

Fig. 1. Router testbed configuration

Given the influence of system bus design on gigabit Ethernet performance, [13] we conducted all tests using 64 bit/66 MHz PCI-based cards in 64 bit/66 MHz slots. The Linux system under test had a 1666 MHz Athlon processor and

1 GB RAM. It was based on Red Hat 7.3, with a stock 2.4.18 kernel installed. A variety of gigabit Ethernet interfaces were tested:

- SysKonnect SK-9844, one of the original 1000Base SX (fiber) interfaces with an Xaqli XMAC II controller.
- NetGear GA621, another 1000Base SX (fiber) interface with a National Semiconductor DP83820 controller.
- NetGear GA622T, also based on a National Semiconductor DP83820 controller. but with a 1000Base TX (copper) interface.

The range of tests covered offered loads from 100 kbps to 1 Gbps, with packet sizes from 128 to 1500 bytes. The complete set of results is available at the NIST Net web site. [14] In this paper, we will consider only a representative subset of tests, with loads from 100 Mbps to 900 Mbps, and packet sizes of 1024 bytes.

For all tests, we brought the load up to the specified level for ten seconds, then captured and recorded the latencies of approximately 200,000 packets. (At lower loads, some of the tests did not run long enough to generate that many packets, but in all cases at least 125,000 were captured.) For the analyses below, a representative slice of 30,000 packet latencies was taken out of the middle of the capture range; this was done to avoid any possible sampling "edge" effects.

3 Results

Figure 2 shows sample results, for 100, 300, 500 and 700 Mb/sec loads with NetGear GA621 adapters. Here, and in all the charts, latencies (y-axis) are expressed in units of 0.1 μsec (100 nanosec).

As may be seen, packet latency behavior changes dramatically as the load increases. Looking at the corresponding latency distributions (Figure 3), we can identify three distinct performance regimes: At low loads, latencies are fairly uniformly distributed over a relatively narrow range, with an overlay of horizontal banding (seen as peaks in the distribution plots). As the load increases, the banding becomes increasingly pronounced, until the latency trace takes on a "quantum" appearance, with certain latency levels "allowed" and common, with other intermediate levels being "forbidden" and nearly absent. Finally, when the load reaches sufficiently high levels, the pattern changes again; the latency traces take on a vertical, peak and valley appearance, rather than a horizontal, banded one, and the latency distribution begins to resemble a normal curve (in the statistical sense).

The other Ethernet adapters exhibit similar behavior, but with some notable differences at higher offered loads. In the case of the NetGear Ga622T, the newest and, presumably, highest-performing adapter of the group, the transition from the horizontal quantum regime to the vertical normal regime happens at a higher load level; at 70% offered load (700 Mb/sec), it is in a transitional state between the two regimes (Figure 4).

For the SysKonnect SK-9844, the high-load latency traces take on a distinctive appearance. While still quasi-normal in terms of their distribution curves, they add a new, long-term periodicity to the trace curves (Figure 5).

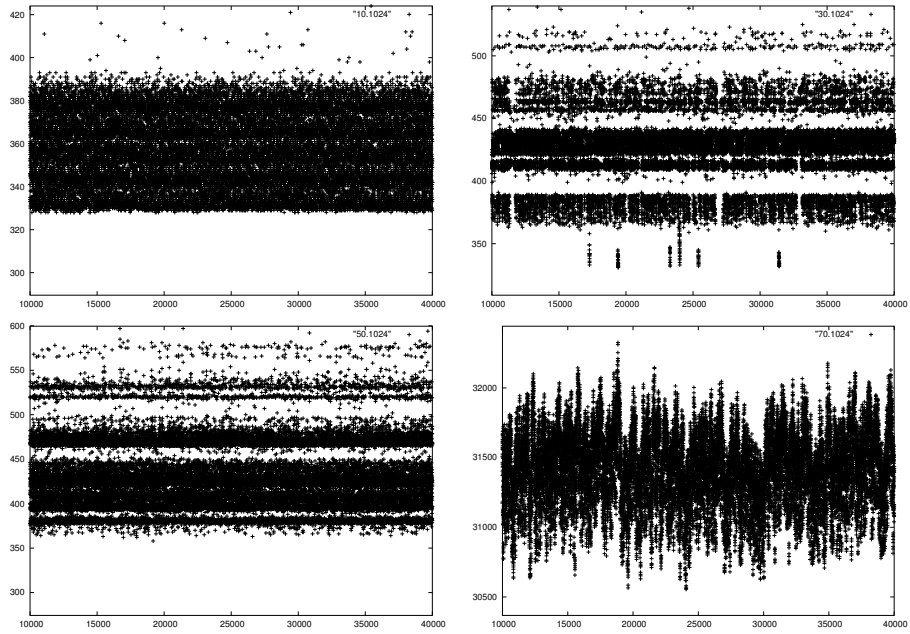

Fig. 2. GA621 latency traces for 100, 300, 500 and 700Mb/sec loads

4 Analysis

How can we explain the latency behavior exhibited by the various gigabit interfaces? The horizontal banding seen at low-to-mid load levels is a typical indication of an underlying clock during some point of processing; only when a clock tick occurs can packet processing advance. For example, latency traces when NIST Net-imposed delays are added (available at the NIST Net site [14]) show clear indications of the 8192Hz (122 μsec) NIST Net clock.

In these traces, banding seems to occur at multiples of approximately 1.4 μsec, or a bit over 700kHz. The likeliest explanation is found in the interrupt coalescing used by the gigabit interfaces to reduce the interrupt load on the CPU; [9] this acts in effect like a clock, bunching packets in certain latency ranges.

Banding would then be expected to be less distinct at lower loads, because such coalescing would occur only on input; once a packet is received, it can be handled all the way through the kernel forwarding code through output to the egress interface with no further delays, and hence no further coordination imposed. In this situation, the natural variation in processing time in the relatively long forwarding path would tend to smear the banding out.

By contrast, at medium loads, coordination is also required on output. With a packet size of 1024 bytes, when the load reaches 300 Mbps, interpacket arrival intervals are approximately 27 μsec, which is less than the minimum latency for

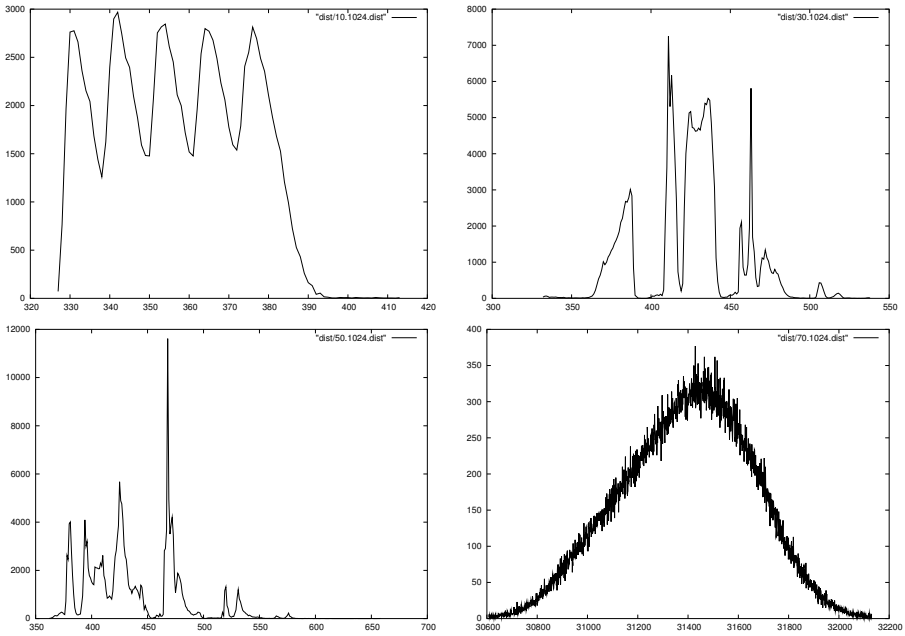

Fig. 3. GA621 latency distributions

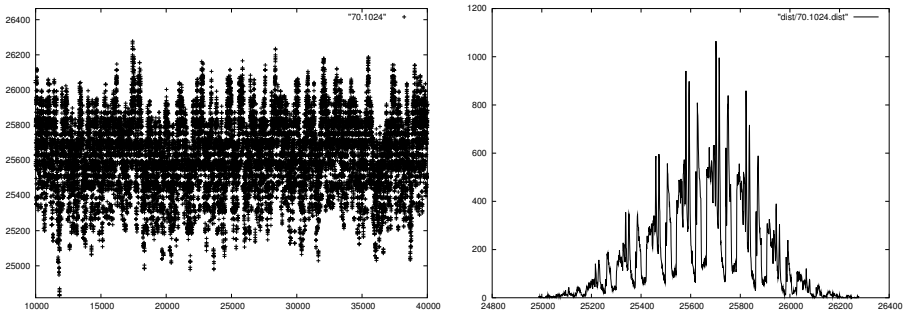

Fig. 4. GA622T latency trace and distribution

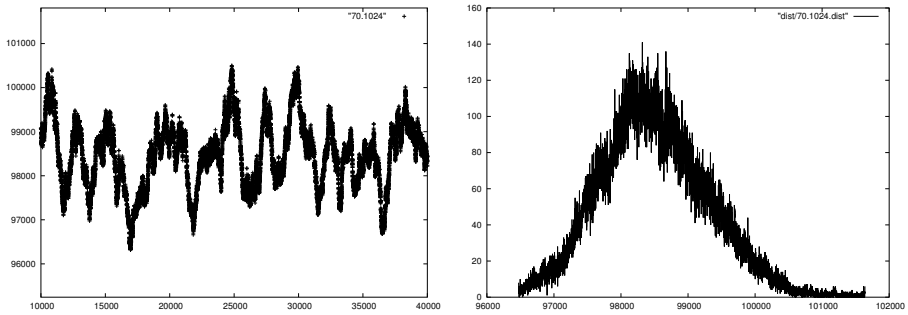

Fig. 5. sk9844 latency trace and distribution

all the gigabit adapters (around 33 μsec for 1024 byte packets). Hence, input processing can begin to interfere with output processing, forcing coordination between the two. This results in the quantized distinct bands seen in traces in this range.

At the very highest loads, queuing effects become evident. When packets arrive at rates higher than they can be forwarded, they tend to queue up until a critical point is reached, whereupon the queue is flushed, all packets are either forwarded or lost, and the cycle repeats. This pattern gives a ladder-like, vertical structure to the latency traces.

Looking at more detailed traces for 700Mb/sec and 900Mb/sec loads (Figure 6), it appears this cycle occurs around every 37 packets at 700Mb/sec and every 50 packets at 900Mb/sec, or in both cases approximately every 450 μsec (2.2kHz). During this cycle, latency increases approximately 70 μsec. A gap of approximately 50 μsec is then required to flush out all packets; during this time several packets are lost (around 4-5 at 700Mb/sec and 6-7 at 900Mb/sec).

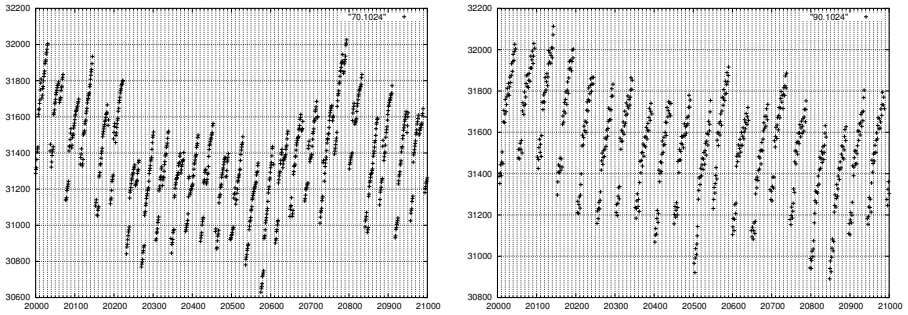

Fig. 6. GA621 detail traces for 700 and 900Mb/sec loads

The behavior of the SK-9844 card under high load differs in several respects. While its average latency is higher (985 μsec at 700Mb/sec *vs.* 314 μsec for the GA621), its loss rate is correspondingly lower (5% *vs.* 16%). Presumably the differences are due to the much larger buffering provided by the SK-9844 (enough for nearly 1000 packets). This allows the queuing behavior to play out over a much larger range.

5 Modeling

The relatively simple, repetitive structures exhibited by the latency traces facilitates the development of relatively simple, fast computational models of this behavior. The harmonic nature of the latency data encourages the notion of using Fourier transform-based models.

Our basic approach is to use a truncated Fourier representation; that is, given a representation of the latency data $l(t)$ as a Fourier series

$$l(t) = \sum_{i}^{N} c_i f_i(t)$$

we simplify it to

$$\overline{l}(t) = \sum_{i \in max(N)} c_i f_i(t)$$

where $max(N)$ is the subset of (indices of) Fourier coefficients c_i of greatest absolute value. For the models presented here, we chose $N = 30000$ (that is, began with a 30,000-point Fourier representation), and then truncated this to the largest 500 or 200 values.

Such modeling works quite well in unmodified form for higher data rates, where the queuing behavior overwhelms all other structure. The left side of Figure 7 shows the results of a truncated 500-coefficient Fourier representation of the GA621 data trace. The SK9844 behavior, show on the right, is even simpler, being perfectly well modeled by a 200-coefficient Fourier representation.

At lower data rates, some provision must be made for the "quantum" nature of the latency distributions. The most straightforward method of doing so is to generate points using a simpler distribution (in this case, the nearly-normal distribution produced by a truncated Fourier representation), and then "nudge" them into the desired distribution as described in, for example [15] and [16]. Mathematically, if $F(x)$ and $G(x)$ are the cumulative distribution functions of the current and desired distributions, we replace a generated value x by

$$y = G^{-1}(F(x))$$

.

The nudging process need not be terribly precise; in this case, we found that 100-point tabular approximations to the cumulative distribution functions

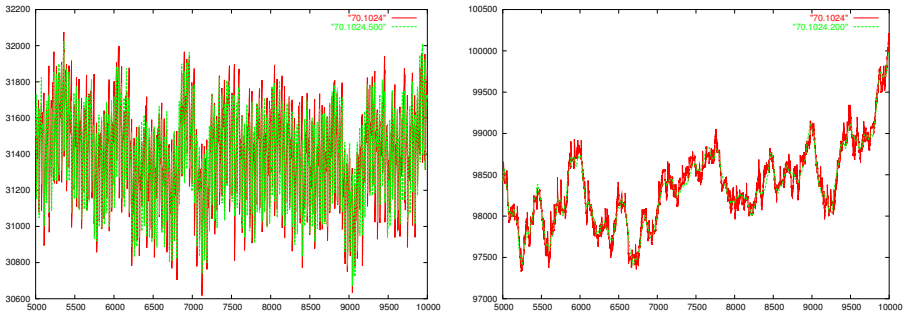

Fig. 7. GA621 and SK9844 modeled 700Mb/sec traces

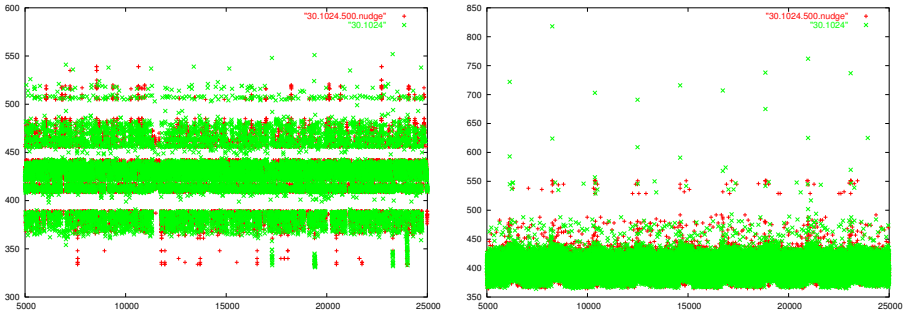

Fig. 8. GA621 and SK9844 modeled 300Mb/sec traces

are more than adequate. Applying these to a 500-coefficient truncated Fourier representations yields the modeled traces shown in 8.

Overall, then, all the observed latency distributions can be well-modeled with relatively compact representations, which allows for very fast latency value generation. In our first, totally unoptimized attempts, all representations require at most 700 data values (500 Fourier coefficients and two sets of 100-point distribution approximations); total analysis and regeneration time for 30,000 values was under 0.1 sec (over 300,000 values analyzed and regenerated per second). This can undoubtedly be improved greatly; in particular, it does not seem necessary to generate all 30,000 coefficients, since the desired few hundred all tend to be found within the first thousand or so.

When analysis has already been completed, values can be generated at rates well in excess of 1 million/second.

6 Conclusions

In this paper, we have presented empirical descriptions at the sub-microsecond level of the latency behavior of a variety of gigabit Ethernet interfaces in a Linux-based router. Based on these observations, we have posited some tentative analyses, and then created simple Fourier transform-based generative models capable of recreating the observed behaviors.

The phenomena observed here suggest several further avenues of exploration. As one example, the quantization and queuing effects are clearly sources of increased jitter (latency variance). When multiple routers of similar characteristics are connected, these effects may tend to reinforce each other, or tend to cancel each other out, depending on the details of their interactions. In a simulation of two interconnected routers passing traffic at 300Mb/sec, we found jitter could vary more than 20%, with no real change in overall average latency, depending on the exact alignment of the two router's clocks. Interestingly, in this case overall jitter can be reduced, at the expense of a slight increase in average latency, by having the first router *add* a small amount of jitter to its output timings.

The data and analytical results described in this paper are all in the public domain and are available through the NIST Net web site [14].

7 Acknowledgments

The work described here was supported in part by the Defense Advanced Research Project Agency's Network Measurement and Simulation (NMS) project. We would also like to thank our colleagues at NIST, in particular Doug Montgomery, John Lu and Kevin Mills, for their valuable advice throughout the course of this project and during the preparation of this manuscript.

References

1. Agrawala, A., Shankar, U.: Fine-time-scale measurement of the Internet (1993–2000) http://www.cs.umd.edu/projects/sdag/netcalliper/.

2. Sanghi, D., Gudmundsson, O., Agrawala, A., Jain, B.: Experimental assessment of end-to-end behavior on Internet. Technical Report CS-TR 2909, University of Maryland (1992) ftp://ftp.cs.umd.edu/pub/netdyn/infocom93.ps.
3. Carson, M., Santay, D.: NIST Net: a Linux-based network emulation tool. SIGCOMM Comput. Commun. Rev. **33** (2003) 111–126
4. Jain, R., Routhier, S.A.: Packet trains: measurements and a new model for computer network traffic. IEEE Journal on Selected Areas in Communications **4** (1986) 986–995
5. Schulzrinne, H.: RTP profile for audio and video conferences with minimal control. RFC 1890 (1996) http://www.ietf.org/rfc/rfc1890.txt.
6. Schulzrinne, H., Rao, A., Lanphier, R.: Real time streaming protocol (RTSP). RFC 2326 (1998) http://www.ietf.org/rfc/rfc2326.txt.
7. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Berners-Lee, T.: Hypertext transfer protocol – HTTP/1.1. RFC 2068 (1997) http://www.ietf.org/rfc/rfc2068.txt.
8. Cohen, E., Kaplan, H., Oldham, J.: Managing TCP connections under persistent HTTP. Computer Networks (Amsterdam, Netherlands: 1999) **31** (1999) 1709–1723
9. Jin, G., Tierney, B.L.: System capability effects on algorithms for network bandwidth measurement. In: Internet Measurement Conference 2003 (IMC 2003). (2003)
10. Grover, W.: Self-organizing broad-band transport networks. In: Special Issue on Communications in the 21st Century. Volume 85. (1997) 1582–1611
11. Partridge, C. In: Gigabit Networking. Addison-Wesley (1994) 109, 231, 270
12. Spirent Communications: SmartBits 6000B Performance Analysis System (1996) http://www.spirentcom.com/.
13. Gray, P., Betz, A.: Performance evaluation of copper-based gigabit ethernet interfaces. In: Proceedings of the 27th Annual IEEE Conference on Local Computer Networks, IEEE Computer Society (2002) 679–690 http://www.cs.uni.edu/ gray/gig-over-copper/hsln-lcn.ps.
14. Santay, D., Carson, M.: NIST Net web site; calibration data (2003–2004) http://www.antd.nist.gov/nistnet/calibration/.
15. Chen, H., Asau, Y.: On generating random variates from an empirical distribution. AIEE Transactions **6** (1974) 163–166
16. Bratley, P., Fox, B.L., Schrage, L.E. In: A Guide To Simulation. Springer Verlag (1987) 149