Throughput and Delay Analysis of Interrupt-Driven Kernels under Poisson and Bursty Traffic

K. Salah[**] and K. Elbadawi
Department of Information and Computer Science
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
Email: {salah,elbadawi}@kfupm.edu.sa

Abstract

This paper studies the performance of interrupt-driven kernels when subjected to heavy network traffic such as that of Gigabit Ethernet. Under heavy network traffic, the kernel performance will be negatively affected due to interrupt overhead caused by the incoming traffic. In particular, excessive latency and significant degradation in system throughput can be experienced. In this paper, we present analytical models to study the performance in terms of two key kernel performance metrics: throughput and delay. The performance is also studied using simulation. Both Poisson and bursty traffic with empirical packet size distribution are considered.

KEYWORDS: High-Speed Networks, Operating Systems, Interrupts, Bursty Traffic, Receive Livelock, Modeling and Analysis, Performance Evaluation.

1. Introduction

The arrival rate for incoming packets in a Gigabit network link can surpass the system packet processing rate of network protocol stack processing and interrupt cost. In fact even with today's powerful multi gigahertz processors, the cost of per-packet interrupt *alone* surpasses the inter-arrival time of packets. With Gigabit Ethernet and the highest possible rate of 1.23 million interrupts per second for a minimum sized packet of 64 bytes, the CPU must process a packet in less than 0.82 μs in order to keep up with such a rate. According to reported measurements in [1], an incoming-packet interrupt cost, on a 450MHz Pentium-III machine running Linux 2.2.10, was 13.23 μs. With the presence of more powerful multi gigahertz processors these days, it is expected the interrupt cost will not be decreased linearly by the speed frequency of the processor, as I/O and memory speed limits dominate [2]. In [2] it was concluded that the performance of 2.4GHz processor only scales to approximately 60% of the performance of an 800MHz processor.

Under heavy traffic load such as that of Gigabit networks, the performance of interrupt-driven systems can be degraded significantly, and thus resulting in a poor host performance perceived by the user. For one thing,

[**] Corresponding Author: Prof. K. Salah, PO Box 5066, ICS Department, KFUPM, Dhahran 31261, Saudi Arabia

every hardware interrupt, for every incoming packet, is associated with context switching of saving and restoring processor's state as well as potential cache/TLB pollution. More importantly, interrupt-level handling, by definition, has absolute priority over all other tasks. If interrupt rate is high enough, the system will spend all of its time responding to interrupts, and nothing else will be performed; and hence, the system throughput will drop to zero. This situation is called *receive livelock* [2]. In this situation, the system is not deadlocked, but it makes no progress on any of its tasks, causing any task scheduled at a lower priority to starve or not have a chance to run.

The receive livelock was established by experimental work on real systems in [2-4]. A number of solutions have been proposed in the literature [1,3,5-13] to address network and system overhead and improve the OS (Operating System) performance. Some of these solutions include interrupt coalescing, OS-bypass protocol, zero-copying, jumbo frames, polling, pushing some or all protocol processing to hardware, etc. In most cases, published performance results are based on research prototypes and experiments. However little or no research has been done to study analytically the impact of interrupt overhead on OS performance. In [9,13], a simple calculation of the interrupt overhead was presented. In [9], a mathematical equation was given directly for application throughput based on packet length and cost of interrupt overhead per byte and per packet. In [13], the interrupt overhead was computed based on the arrival rate, ISR (Interrupt Service Routine) time, and a fixed cost of interrupt overhead. Both of these calculations are very simple. The calculations fail to consider complex cases such as interrupt masking, CPU utilization, and effect of ISR and its overhead on packet processing at OS and application levels. Moreover, the calculations fail to capture the receive livelock phenomenon and identify the saturation point of the host.

In this paper, we study the performance in terms of two key system performance metrics: throughput and delay. Also, equations are given for CPU utilization and saturation conditions. Since our analysis is based on queueing theory, our analytical work can be easily extended to study mean waiting time, mean number of packets in system and queues, blocking probability, etc. [14]. In addition, our analytical work can be important for engineering and designing various NIC (Network Interface Card) and system parameters. These parameters may include the proper service times for ISR handling and protocol processing, buffer sizes, CPU bandwidth allocation for protocol process and application, etc.

In previous work [15], we presented a preliminary analytical study. The system performance was studied primarily based on throughput. A detailed simulation models for hosts with PIO and DMA were given in [16]. In sharp contrast to our previous work presented in [15,16], this paper is different in significant ways. First, the host performance is studied and compared in terms of two key performance indicators which include system throughput and system latency. Second, we utilize DES simulation to model and examine the performance when host is subjected to not only to Poisson traffic but also to bursty traffic with variable packet sizes. Third and as opposed to our previous work, we consider more realistic values for system parameters that suit modern Gigabit network environment and hosts. In our previous work we used system parameters of 400 MHz Pentium III machines and with network traffic of 10 and 100 Mbps. In this article we consider system

parameters of today's state-of-the-art CPU cores such as the 2.53 GHz Pentium-IV machines and with network traffic of 1Gbps.

The rest of the paper is organized as follows. Section 2 presents an analytical model to study system behavior and derives equations for various system performance metrics. Numerical examples showing both analysis and simulation results under Poisson traffic are given in Section 3. Section 4 examines performance impact when hosts are subjected to bursty traffic with empirical variable-size packets. The impact is studied using simulation. Finally, Section 5 concludes the study and identifies future work.

2. Analysis

In this section we present an analytical study to examine the impact of interrupt overhead on system performance. First we define the system parameters. Let λ be the mean incoming packet arrival rate, and μ be the mean protocol processing rate carried out by the kernel. Therefore $1/\mu$ is the time the system takes to process the incoming packet and deliver it to the application process. This time includes primarily the network protocol stack processing carried out by the kernel, excluding any interrupt handling. Let T_{ISR} be the interrupt handling time, which is basically the interrupt service routine time for handling incoming packets. We will also define $\rho = \lambda/\mu$. ρ is as a measure of the traffic intensity or system load. We study the system performance in terms of system throughput (γ) and delay (R). System throughput is the rate at which packets are delivered by the kernel to the application process. System delay is the latency of delivering one packet by the kernel to the application once received by the network adapter. In addition, we investigated two important and critical operating points for the system, mainly receive lock and saturation. Receive-livelock point is the point at which system throughput becomes zero. Saturation point is the point at which the system can not keep up with the offered load.

Throughout our analysis, we assume the times for protocol processing or ISR handling to be *not* constant. Both of these service times change due to various system activities. For example ISR handling for incoming packets can be interrupted by other interrupts of higher priority, e.g. timer interrupts. Also, protocol processing can be interrupted by higher priority kernel tasks, e.g. scheduler. For our analysis, we assume both of these service times to be exponential. Also we assume the network traffic follows a Poisson process, i.e. the packet interarrival times are exponentially distributed with fixed packet sizes. This assumption is valid for Constant Bit Rate (CBR) traffic such as uncompressed interactive audio and video conferencing.

Our analytical models assume Poisson traffic with fixed-size packets. In practice, network traffic is not always Poisson and packets are not always fixed in size. In [17-19], it was shown that Ethernet traffic is bursty and characterized as self-similar with long range dependence. An analytical solution becomes intractable when considering variable-size packets and non-Poisson arrivals. In Section 4, we use simulation to study and compare the impact of bursty traffic with empirical variable packet sizes on system performance.

2.1. Ideal System

In the ideal system, we assume the overhead involved in generating interrupts is totally ignored. With our assumptions, we can simply model such a system as an *M/M/1/B* queue with a Poisson packet arrival rate λ and a mean protocol processing time of $1/\mu$ that has an exponential distribution. *B* is the maximum size the system buffer can hold. *M/M/1/B* queueing model is chosen as opposed to *M/M/1* since we can have the arrival rate go beyond the service rate, i.e., $\rho > 1$. This assumption is a must for Gigabit environment where under heavy load λ can be very high compared to μ.

2.2. DMA Option

For our hosts, we assume that the NIC (Network Interface Card) is equipped with DMA engines. However, a NIC adapter can be designed with a PIO-based option. A NIC adapter with PIO-based design can be an attractive option when considering factors such as cost, simplicity, and speed and efficiency in copying relatively small-size packets [20]. However, a major drawback for a PIO-based design is burdening the CPU with copying incoming packets from the NIC to kernel memory. In order to save CPU cycles consumed in copying packets, major network vendors equip high-speed NICs with DMA engines. These vendors include Intel, 3Com, HP, Alteon owned now by Nortel, Sundace, and NetGear. NICs are equipped with a receive Rx DMA engine and a transmit Tx DMA engine. A Rx DMA engine handles transparently the movement of packets from the NIC internal buffer to the host system memory. A Tx DMA engine handles transparently the movement of packets from the host memory to the NIC internal buffer. Both DMA engines operate in a bus-master fashion, i.e. the engines request access to the PCI bus instead of waiting to be polled. It is worth noting that the transfer rate of incoming traffic into the kernel memory across the PCI bus is not limited by the throughput of the DMA channel. These days a typical DMA engine can sustain over 1 Gbps of throughput for PCI 32/33 MHz bus and over 4 Gbps for PCI 64/66 MHz bus [21,22].

After the notification of the arrival of a new packet, the kernel will process the packet by first examining the type of frame being received and then invoking immediately the proper handling stack function or protocol, e.g. ARP, IP, TCP, UDP, etc. The packet will remain in the kernel or system memory until it is discarded or delivered to the user application. The network protocol processing for packets carried out by the kernel will continue as long as there are packets available in the system memory buffer. However, this protocol processing of packets can be interrupted by ISR executions as a result of new packet arrivals. This is so because packet processing by the kernel runs at a lower priority than the ISR.

There are two possible system delivery options of packet to user applications. The first option is to perform an extra copy of packet from kernel space to user space. This is done as part of the OS protection and isolation of user space and kernel space. This option will stretch the time of protocol processing for each incoming packet. A second option eliminates this extra copy using different techniques described in [6-8,23-25]. The kernel is written such that the packet is delivered to the application using pointer manipulations. Our analytical model

captures both options. The only difference is in the protocol processing time. The second option will have a smaller processing time than the first.

It is important to note that the NIC is typically configured such that an interrupt is generated after the incoming packet has been completely DMA'd into the host system memory. In order to minimize the time for ISR execution, ISR handling mainly sets a software interrupt to trigger the protocol processing for the incoming packet. In this situation if two or more packets arrive during an ISR handling, the ISR time for servicing all of these packets will be the ISR time for servicing a single packet, with no extra time introduced.

Let us assume that the T_{ISR} for handling an interrupt is exponentially distributed with a mean of $1/r$. One can express the mean effective service rate as

μ' = Rate at which packets gets processed by the kernel's network protocol given that the CPU is available for protocol processing, i.e. the CPU is not handling ISR.

$$\mu' = \mu \cdot (\% \text{ CPU availability for protocol processing}). \quad (1)$$

The DMA-based design option can be modeled as an *M/G/1/B* queue with a Poisson packet arrival rate of λ and a mean effective service time of $1/\mu'$ that has a general distribution. In order to determine the mean effective service time $1/\mu'$, we need to determine the CPU availability and usage percentages for protocol processing and interrupt handling, respectively. We use a Markov process to model the CPU usage, as illustrated in Figure 1. The process has state (0,0) and states (1,*n*). State (0,0) represents the state where the CPU is available for protocol processing. States (1,*n*) with $0 \le n < \infty$ represents the state where the CPU is busy handling interrupts. *n* denotes the number of interrupts that are batched or masked off during T_{ISR}. Note that *n*+1 denotes the number of packet arrivals during ISR handling. Therefore, state (1,0) means there are no interrupts being masked off and the CPU is busy handling an ISR with one packet arrival. State (1,1) means that one interrupt has been masked off and the CPU is busy handling an ISR with two packet arrivals. Both of these packets will be serviced together with a mean rate *r* of servicing only one packet.

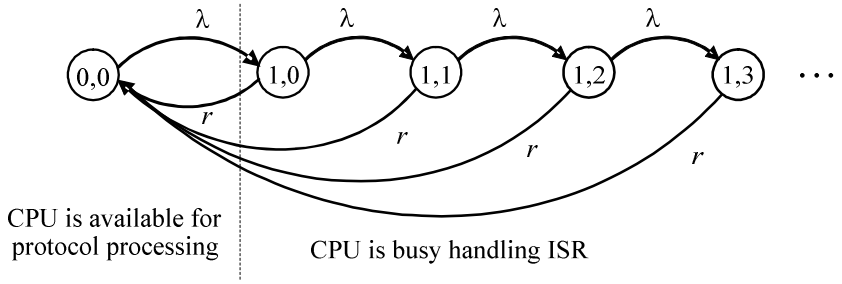


Figure 1. Markov state transition diagram for modeling CPU usage due to ISR with DMA

The steady-state difference equations can be derived from $\mathbf{0} = \boldsymbol{pQ}$, where $\boldsymbol{p} = \{p_{0,0}, p_{1,0}, p_{1,1}, p_{1,2}, \cdots\}$ and $\boldsymbol{Q}$ is the rate-transition matrix and is defined as follows:

$$\boldsymbol{Q} = \begin{bmatrix} -\lambda & \lambda & 0 & 0 & 0 & \cdots \\ r & -(\lambda + r) & \lambda & 0 & 0 & \cdots \\ r & 0 & -(\lambda + r) & \lambda & 0 & \cdots \\ r & 0 & 0 & -(\lambda + r) & \lambda & \cdots \\ r & 0 & 0 & 0 & -(\lambda + r) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

This will yield $-\lambda p_{0,0} + r\,(p_{1,0} + p_{1,1} + p_{1,2} + \cdots) = 0.$

Since we know that $p_{0,0} + \sum_{n=0}^{\infty} p_{1,n} = 1$, then

$$-\lambda p_{0,0} + r\,(1 - p_{0,0}) = 0.$$

Solving for $p_{0,0}$, we thus have

$$p_{0,0} = \frac{r}{\lambda + r},$$

and

$$1 - p_{0,0} = \frac{\lambda}{\lambda + r}.$$

Therefore, the CPU availability percentage for protocol processing and the CPU utilization for handling interrupts are $r/(\lambda + r)$ and $\lambda/(\lambda + r)$, respectively.

Thus, the mean effective service rate can be expressed as

$$\mu' = \mu \cdot \frac{r}{\lambda + r}. \qquad (2)$$

The term $\frac{r}{\lambda + r}$ is the percentage of CPU bandwidth available for protocol processing, and is equal to $1 - \frac{\lambda}{\lambda + r}$. Please note that by examining equation (1), the disrupted or effective service time with a mean $1/\mu'$ is exponentially distributed since the service time with a mean of $1/\mu$ is exponential, and thus the *M/G/1/B* queueing model becomes the known Markovian queue of *M/M/1/B*.

2.3. Throughput Analysis

In an *M/M/1/B* model or the ideal system, the mean system throughput γ can be expressed as

$$\gamma = \mu\,(1 - p_0), \qquad (3)$$

where p_0 is the probability of zero packets being processed by the kernel's network protocol stack. p_0 is given by

$$p_0 = \begin{cases} \dfrac{1-\rho}{1-\rho^{B+1}} & (\rho \neq 1), \\ \dfrac{1}{B+1} & (\rho = 1). \end{cases} \qquad (4)$$

The system throughput can be expressed by equation (3). However, the mean service rate μ of equation (3) must be replaced by the mean effective service rate μ' of equation (2).

2.4. Receive-Livelock Point

A critical operating point for a system is the condition at which the system throughput becomes zero. This condition is where receive livelock occurs. By examining equation (3), system throughput (γ) can be zero if p_0 = 1 or the mean service rate μ is zero.

The mean effective service rate μ' of equation (2) depends only on the values of λ and *r*, as copying has been eliminated. From theoretical aspect as given by equation (2), μ' does not become zero unless $\lambda \to \infty$. This means that the receive-livelock point for DMA occurs *theoretically* when $\lambda \to \infty$. However from practical aspect, having an incoming packet arrival rate λ to reach infinity is not possible due to the physical limitation of host hardware as well as the capacity limit of network link. Host physical limitation is related to PCI bus, DMA engines, memory speed, etc. Nowadays the transfer bit rate of a 64/66 MHz PCI bus goes little over 4 Gbps [21,22]. Also the current exiting Ethernet links have a bandwidth of up to 10 Gbps. Of course the incoming packet rate λ is much less than such a bit rate as it gets reduced by the packet size and payload overhead. Therefore from practical aspect, the receive livelock should not occur when employing DMA.

2.5. Saturation Point

Another critical operating point for a system is the situation at which the system can not keep up with the offered network load. This is referred to as the saturation point where $\rho = 1$, or is defined as the "cliff" point of system throughput. It is where the throughput starts falling as the network load increases.

The saturation point can be expressed as

$$\rho = 1 \quad or \quad \lambda = \mu \cdot \frac{r}{\lambda + r}.$$

Solving for λ, we get

$$\lambda(\lambda + r) = \mu r \quad \Rightarrow \quad \lambda^2 + r\lambda - \mu r = 0.$$

The roots of the quadratic equation $\lambda^2 + r\lambda - \mu r = 0$ are

$$\lambda = \frac{-r \mp \sqrt{r^2 + 4\mu r}}{2} = \frac{-r \mp r\sqrt{1 + 4\dfrac{\mu}{r}}}{2}.$$

Since the term under the square root is always greater than one then the negative sign is neglected. Therefore, the saturation point for this system occurs when

$$\lambda = \frac{r}{2}\left(\sqrt{1+4\frac{\mu}{r}} - 1\right). \tag{5}$$

It is to be noted that equation (5) can also be derived by finding the maximum point of system throughput. This can be done by taking the derivative of the system throughput of equation (3) with respect to λ and setting it to zero, i.e. $\frac{d\gamma}{d\lambda} = 0$, and then solving for λ.

2.6. Delay Analysis

In *M/M/1/B* model or the ideal system, the mean system packet processing delay $E(r)$ can be given by

$$E(r) = \frac{E(n)}{\lambda(1-p_B)},$$

where $E(n) = \frac{\rho}{1-\rho} - \frac{B+1}{1-\rho^{B+1}}\rho^{B+1}$, $\rho = \lambda/\mu$, and

The mean system delay per packet is affected by both ISR handling and protocol processing. An incoming packet experiences a delay due to interrupt handling and due to the delay of protocol processing. To determine this delay analytically, we apply Jackson's queueing theorem in which the mean system delay is approximated to be the sum of the mean delay of interrupt handling plus the mean delay of protocol processing. Hence the total mean system delay, $E(r)$, according to Jackson theorem can be expressed as

$$E(r) = E_{ISR}(r) + E_{IP}(r), \tag{6}$$

where $E_{ISR}(r)$ is the mean delay due to ISR and $E_{IP}(r)$ is mean delay due to protocol processing.

$E_{ISR}(r)$ is simply $1/r$. This is so due to the nature of servicing packets during ISR handling. The mean ISR handling time for one packet or many packets is the same, i.e. $1/r$. This delay can also be computed using the Markov chain depicted in Figure 1. First we compute $p_{1,n}$ from Figure 1. Using mathematical induction and the iterative method of solving the steady-state difference equations, $p_{1,n} = \frac{r}{\lambda}\left(\frac{\lambda}{\lambda+r}\right)^{n+2}$. The average number of packets being serviced by one ISR, $E_{ISR}(N)$, can be computed using the Markov chain depicted in Figure 1, and consequently can be expressed as

$$E_{ISR}(N) = \sum_{n=0}^{\infty}(n+1)p_{1,n} = \sum_{n=1}^{\infty} np_{1,n} + \sum_{n=0}^{\infty} p_{1,n} \ .$$

With further simplification,

$$E_{ISR}(N) = \frac{\lambda}{r}.$$

And therefore, the average ISR delay per packet, $E_{ISR}(r)$, according to Little's law, is

$$E_{ISR}(r) = \frac{E_{ISR}(N)}{\lambda} = \frac{1}{r}. \tag{7}$$

As for the mean delay caused by protocol processing, $E_{IP}(r)$, it is simply the mean delay encountered in the *M/M/1/B* queueing system with $\rho = \left(\frac{\lambda}{\mu'}\right)$. According to [14], such delay can be expressed as

$$E_{IP}(r) = \frac{E_{IP}(N)}{\lambda'}, \tag{8}$$

where $E_{IP}(N) = \frac{\rho}{1-\rho} - \frac{(B+1)\rho^{B+1}}{1-\rho^{B+1}}$ and λ' is the mean effective arrival rate. λ' is the same as the mean system throughput γ given by equation (3) with $\mu = \mu'$. Also remember that the μ', to be substituted in the equations of λ' and ρ, is given by equation (2) in this case.

3. Analysis Verification and Validation

In this section we verify and validate our analysis two ways: by considering some special cases and by utilizing a DES simulation.

3.1. Special Cases

We consider a special case when interrupt handling is ignored, i.e., the ideal system when $T_{ISR} = 0$. In this situation when $T_{ISR} = 0$, $r \to \infty$. With this condition, we prove that the equations for the mean effective service rate, saturation point and mean system latency become the same equations for the ideal system.

Mean effective service time. We prove that equation (2) yields the ideal system mean service rate μ as follows:

$$\mu' = \lim_{r\to\infty} \mu \cdot \left(\frac{r}{\lambda + r} \right) = \lim_{r\to\infty} \mu \cdot \left(\frac{1}{\lambda/r + 1} \right) = \mu \ .$$

Saturation Point. We prove that equation (5) yields the ideal system saturation point of $\lambda = \mu$ as follows:

$$\lambda = \lim_{r\to\infty} \left(\frac{r}{2} \sqrt{1 + 4\frac{\mu}{r}} - \frac{r}{2} \right) = \lim_{r\to\infty} \left(\frac{\sqrt{1 + 4\frac{\mu}{r}} - 1}{\frac{2}{r}} \right).$$

Applying L'Hopital Rule, we get

$$\lambda = \lim_{r\to\infty} \left(\frac{-2\mu}{r^2 \sqrt{1 + 4\mu/r}} \Big/ \frac{-2}{r^2} \right) = \lim_{r\to\infty} \left(\frac{\mu}{\sqrt{1 + 2\mu/r}} \right) = \mu \cdot$$

Mean system delay. When the interrupt is ignored, it is expected the delays encountered by the interrupt overhead resort to zero. When examining $E_{ISR}(r)$ equation (7), it can be concluded easily that $E_{ISR}(r)$, when $r \to \infty$, becomes zero. This is in line with intuition and our expectation. Consequently, the mean system delay becomes that of an *M/M/1/B* queueing system, i.e., the ideal system.

3.2. Simulation

In order to verify and validate our analytical models, a discrete-event simulation model was developed and written in C language. A detailed description and flowcharts of the simulation model for normal interruption can be found in [16]. The assumptions of analysis were used. The simulation followed closely and carefully the guidelines given in [26]. We used the PMMLCG as our random number generator [26]. The simulation was automated to produce independent replications with different initial seeds that were ten million apart. During the simulation run, we checked for overlapping in the random number streams and ascertain that such a condition did not exist. The simulation was terminated when achieving a precision of no more than 10% of the mean with a confidence of 95%. We employed and implemented dynamically the *replication/deletion* approach for means discussed in [26]. We computed the length of the initial transient period using the MCR (Marginal Confidence Rule) heuristic developed by White [27]. Each replication run lasts for five times of the length of the initial transient period. Analytical and simulation results, as will be demonstrated in Section 3.3, were very much in line.

3.3. Numerical Examples

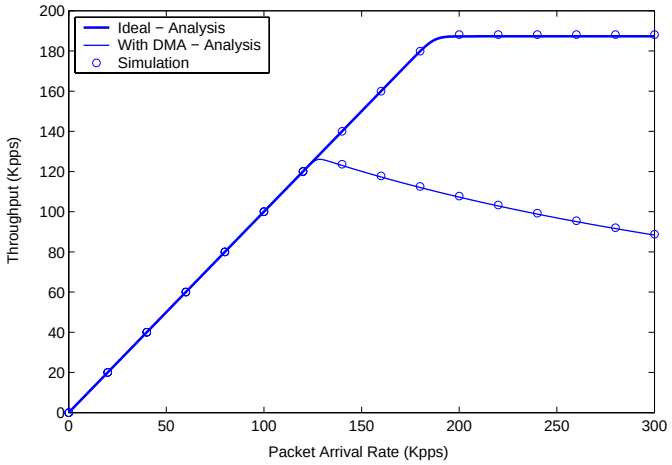
In this section, we report and compare results of analysis and simulation. Numerical results are given for two key performance indicators: mean system throughput and latency. For our numerical examples, realistic values for system parameters, that suit modern Gigabit network environment and hosts, must be used. Experimental

study is the best approach to give accurate measurements, as well as the underlying probability distributions. Such experimental is beyond the scope of this paper and left for future work. However, for the sole purpose of comparison, we base our values on modern credible experimental measurements reported in the literature.

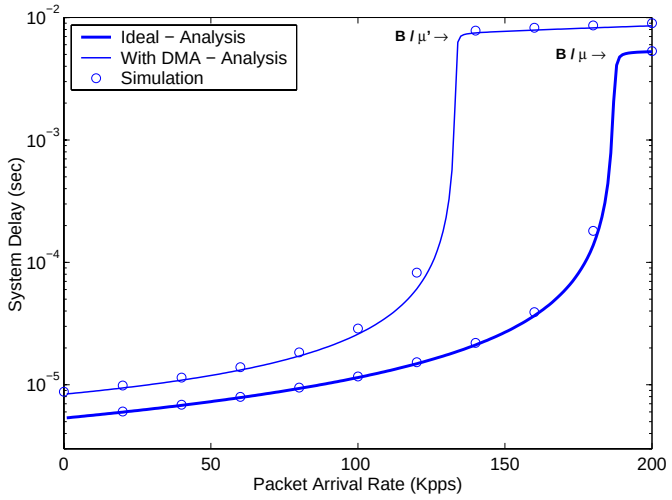
The overall interrupt cost $1/r$ includes both interrupt overhead and handling. In [28], the interrupt overhead for an off-chip timer interrupt with a null event handler was measured to be in the vicinity of 4.36 μs on a 500MHz Pentium-III machine running for FreeBSD-2.2.6. A similar result of 7.7 μs was reported by [4] on a 450MHz Pentium-III machine running Linux 2.2.10. For a modern 2.53GHz Pentium-IV machine, it is expected this overhead will not be decreased linearly by the speed frequency of the processor, as I/O and memory speed limits dominate [29]. In [29] it was concluded that the performance of 2.4GHz processor only scales to approximately 60% of the performance of an 800MHz processor. Consequently the NIC interrupt overhead with null handler for a modern 2.53GHz Pentium-IV machine can be roughly 60% of 4.36, which is 2.62 μs. In [4] the interrupt handling was measured to be 5.53 μs on a 450MHz Pentium-III machine running Linux 2.2.10. The measurement of interrupt handling included substantial work and a major cache pollution. The handling included appending the packet from DMA ring to protocol buffer, replenishment of the DMA ring, and finally notifying the protocol processing. In our case, considering the speed of the processor and limited work for the interrupt handling, which primarily includes notification of protocol processing with minimal cache pollution, we assume the handling cost is 20% of 5.53, or 1.11 μs. Hence for a modern 2.53GHz Pentium-IV machine, the overall interrupt cost $1/r = 2.62 + 1.11 = 3.73$ μs.

For protocol processing, we use the TCP processing values measured by *lmbench* [30] on a 2.53GHz Pentium-IV running Linux 2.4.18. The results are reported in [31]. Also results for different machines are reported in [32]. From the results in [31], it is reported that the average local *loopback* latency for one TCP token (i.e., 4-byte data packet) is 10.5 μs. This time, of course, includes OS overhead as well as TCP actual processing. Ideally, the TCP latency of the receiving path would be approximately half of 10.5, that is 5.25 μs. TCP processing also includes copying of packet payload to user application. [31] reports that the average TCP bandwidth (buffering and copying) is 748 Mbytes/s. Therefore, for a minimum packet size of 64 bytes, the cost of copying and buffering is 64/748= 0.086 μs. Hence the mean TCP processing time $1/\mu$ (for a fixed size packet of 64 bytes) can be summed up to approximately 5.34 μs. In all of our examples, we fix the kernel's protocol processing buffer B to a size of 1000 packets, which occupies about 1.5M bytes of host memory when assuming a maximum of 1538 bytes per packet in accordance to IEEE802.3 standards. This buffer size is a configurable parameter [33].

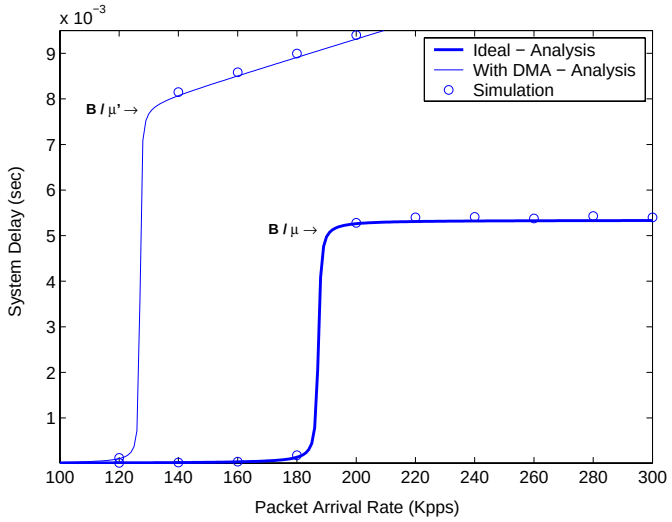
Figure 2a, 2b, and 2c plot the mean system throughput, mean system latency at low load, and mean system latency at high load, respectively, as a function of the system load represented by the packet arrival rate. The packet arrival rate is Poisson for both analysis and simulation. The load and throughput are both expressed in pps (packets per second). Both of these measures can easily be expressed in bits per second by multiplying the packet rate by the packet size. For clarity and interpretations of results, we chose to express these two measures in pps. The figures exhibit a very close agreement between discrete-event simulation results and analysis results.



(a)



(b)



(c)

Figure 2.System throughput and delay in relation to a Poisson arrival rate

From the figures, it is observed that the maximum throughput occurs at 187 Kpps. It can be noted that the saturation or cliff point for the system occurs at 127 Kpps. At this point, the corresponding CPU utilization (for ISR handling plus protocol processing) is at 100%, and thus resulting in a CPU availability of zero. Therefore, user applications will starve and livelock at this point. Figure 2a shows that as the arrival rate increases after the cliff point the system throughput starts to decline. Figure 2c shows the mean system delay also continues to increase after reaching the saturation point. Theoretically, the latency should flatten off at B/μ', but rather we find it slowly shoots to infinity. The decline in the throughput and the sharp increase in latency is due to the fact that the mean effective service rate μ' decreases as the arrival rate increases right after the saturation point. See equation (2). Intuitively, CPU availability for protocol processing decreases as CPU becomes more utilized handling ISR.

4. Impact of Bursty Traffic

Assuming Poisson arrival for network traffic can be valid for modeling real-time voice and video traffic [34]. However such a Poisson traffic fails for modeling Ethernet traffic. It was shown that Ethernet traffic is bursty and characterized as self-similar with long-range dependence [18,19]. A comprehensive summary and review of the topic of self-similar network traffic can be found in [35]. For examining the impact of bursty traffic on the system performance, we modified our discrete-event simulation accordingly. To generate such a bursty traffic, we implemented the method described in [36]. This method follows fractional Gaussian noise such as the resulting self-similar traffic is obtained by aggregating multiple streams (one stream per source) each

consisting of alternating Pareto-distributed ON/OFF periods. In our simulation we used 100 streams. Figure 3 illustrates graphically the aggregation of multiple streams.

Figure 3. Self-similar traffic generation model

Pareto distribution is a heavy-tailed distribution with PDF of $f(x) = ab^{\alpha} / x^{\alpha+1}$, $x \geq b$, where α is a shape parameter and b is a location parameter. We use this distribution to generate both the ON and OFF periods with shape parameters of $\alpha_{ON} = 1.3$ and $\alpha_{OFF} = 1.5$, respectively. The choice of the values of these shape parameters are commonly used and promoted by measurements on actual Ethernet traffic performed in [19]. The location parameter of b_{ON} is the minimum ON period and depends on the minimum Ethernet frame size of 64 bytes. This is fixed to 64x8 bit times or 512 ns. The calculation of b_{OFF} can be computed from the desired total load for all sources, i.e., $\rho_{Total} = \sum_i \rho_i$. We assume equal loads for all sources. The individual load of a single source is measured as $\rho_i = E[ON]/(E[ON] + E[OFF])$. Note that the load ρ_{Total} takes on values between 0 to 1. In Pareto, $E[ON] = (\alpha_{ON} b_{ON})/(\alpha_{ON} - 1)$ and $E[ON] = (\alpha_{ON} b_{ON})/(\alpha_{ON} - 1)$. Solving these simple formulas, b_{OFF} can be expressed (in terms of the known parameters of ρ_{Total}, b_{ON}, α_{ON}, and α_{OFF}) as

$$b_{OFF} = \left(\frac{\alpha_{ON} b_{ON}}{\alpha_{OFF}} \right) \left(\frac{\alpha_{OFF} - 1}{\alpha_{ON} - 1} \right) \left(\frac{1 - \rho_i}{\rho_i} \right).$$

During the ON period, packets are generated back-to-back with a rate of 1 Gbps. The number of packets generated in the ON period depends on the ON period, the packet size, and the inter-packet size. The inter-packet size is 20 bytes which comprises of the standard minimum Ethernet IFG of 12 bytes plus 8 bytes for the preamble. The packet sizes are not fixed and follow an empirical distribution, which are real measurements of packet sizes from MCI backbone. The measurements are reported in [37] and available online at http://www.caida.org. In [37], the reported packet size distribution represents IP datagram sizes. To obtain Ethernet frame size distribution, the packet sizes were modified to include 18-byte header (12 bytes for

destination and source addresses, 2 bytes for length/type, and 4 bytes for FCS). In addition all bytes shorter than 46 bytes were padded to 46 bytes, so that the minimum Ethernet frame size is equal to 64 bytes. The histogram and CDF of the resulting Ethernet frame sizes are shown in Figure 4. The figure shows dominating frame sizes of 64, 570, 594, and 1518 bytes.

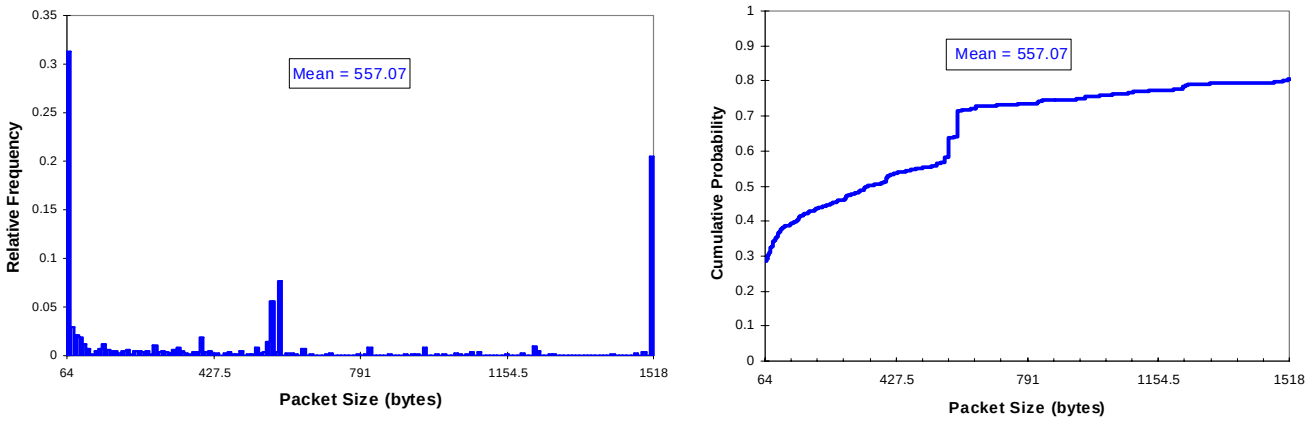
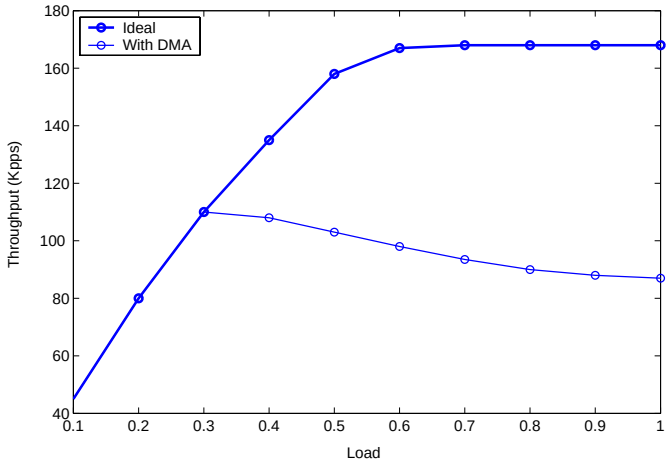


Figure 4. Histogram of empirical packet sizes and their corresponding CDF

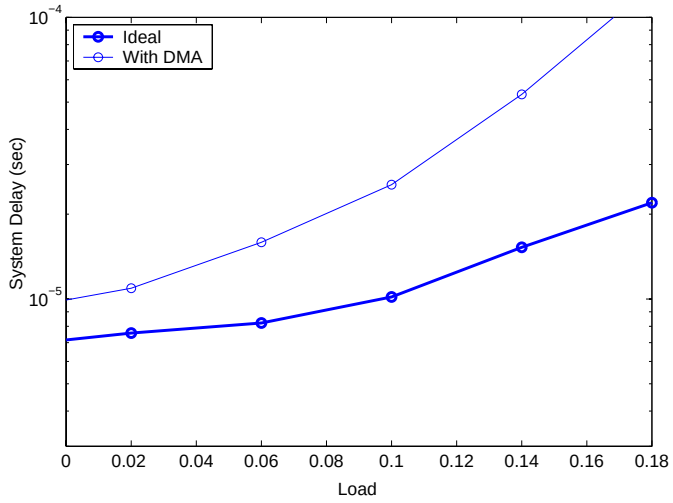
It is to be noted when packet sizes are variable, the service time for protocol processing is strongly correlated with the packet size, primarily due to CRC checksum calculation and copying to application layer. In this case, the service time for protocol reprocessing is comprised of packet overhead which is exponentially distributed with a mean of 5.25 μs plus a fixed per-byte overhead of PacketSize/(748 Mbytes/s).

The simulation results shown in Figure 5 represent the estimated mean of 10 simulation replications. We had to fix the number of replications when generating bursty traffic. As opposed to the simulation carried out with Poisson traffic, a simulation run with bursty traffic can not be automated to stop when achieving a desired precision for the estimated mean. This is because of the presence of irregular incoming traffic. The irregularity of traffic is due to the use of empirical variable packet sizes and the superposition of multiple streams with each stream producing ON and OFF periods (with huge variance) modeled by heavy tailed distribution such as that of Pareto. The problem is exacerbated when the Pareto's shape parameter α is close to 1 (as is the case for the shape parameters for our ON and OFF periods). Therefore simulation with such traffic will be very slow to converge to steady state and thus the CI (Confidence Interval) can be very long [38]. In order to obtain relatively acceptable accuracy and precision, simulation has to produce a huge number of samples [38]. For our simulation, each replication generated for each source at least 10 million samples for its ON period and another 10 millions for its OFF period. Care was taken to make sure that there is no overlapping in the random number streams of simulation.

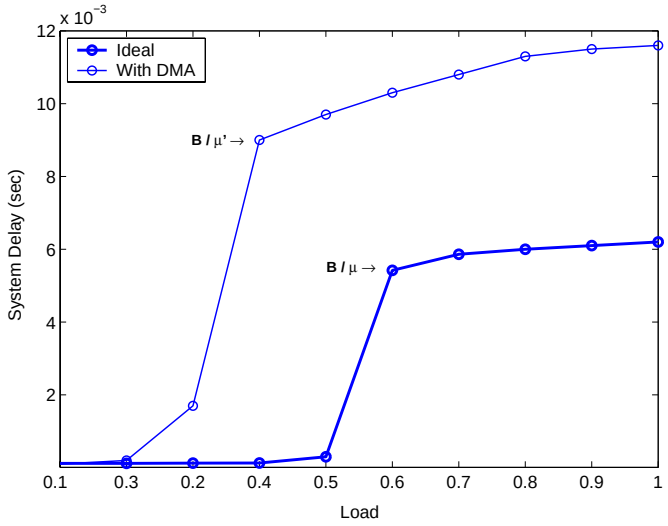
Figure 5a, 5b, and 5c plot the mean system throughput, mean system latency at low load, and mean system latency at high load, respectively, as a function of the aggregated self-similar traffic load ρ_{Total}. By subjectively eyeballing the performance curves and comparing their shapes to the performance curves of Poisson traffic of Figure 2, we find the shapes of the curves are very similar for the most part. Figure 5a shows at low load a curved shape for the achievable system throughput as opposed to straight one. This is because the units for the throughput and the load are not the same. Remember that the load ρ_{Total} takes a value between 0 to 1. Also note that the highest average system throughput is around 167 Kpps. This is expected as the average empirical packet size, based on the distribution presented in Figure 4, is 557 bytes. The average protocol processing service time is the sum of fixed packet overhead of 5.25 µs plus a fixed per-byte overhead of PacketSize/(748 Mbytes/s). This yields an average service time of 5.99 µs, or a throughput of 167Kpps.



(a)



(b)



(c)

Figure 5. Key performance indicators in relation to system load of bursty traffic

Identifying the cliff or saturation point is a key design parameter. For Poisson traffic, Equation (5) gives a cliff point at 127 Kpps. For bursty traffic, Equation (5) surprisingly also gives an adequate approximation for the cliff point. The cliff point can be computed if the average packet size is measured. Based on the empirical packet size distribution, the average packet size is 557bytes. As discussed earlier, this yields an average protocol service time of 5.99 μs, or a rate of 167Kpps. Consequently and applying Equation (5), the cliff point λ_{cliff} for bursty traffic is 116 Kpps, which is very much in line with the simulation results of Figure 5a.

5. Conclusion

We developed analytical models to study the impact of interrupt overhead caused by high-speed network traffic on OS performance. We presented a throughput-delay analysis for hosts when subjected to light and heavy network loads. We considered high-speed network hosts with the DMA design option. We also investigated two critical system operating points, mainly receive livelock and saturation. We considered both Poisson and bursty traffic with variable packet sizes. It was concluded that the system performance under bursty traffic was similar to that of Poisson traffic. As noted, degraded throughput and excessive latency can be encountered due to heavy network loads. Our analysis effort provided equations that can be used to easily and quickly predict the system performance and behavior when engineering and designing network adapters, device drivers, and OS network and communication software. An acceptable performance level varies from one system requirement to another and depends on the worst tolerable system throughput and latency. As a further work,

we will study and evaluate the performance of the different proposed solutions for minimizing and eliminating the interrupt overhead caused by heavy network loads.

Acknowledgement

The authors acknowledge the support of King Fahd University of Petroleum and Minerals in the development of this work.

References

[1] I. Kim, J. Moon, and H. Y. Yeom, "Timer-Based Interrupt Mitigation for High Performance Packet Processing, " Proceedings of 5th International Conference on High-Performance Computing in the Asia-Pacific Region, Gold Coast, Australia, September 2001.

[2] K. Ramakrishnan, "Performance Consideration in Designing Network Interfaces," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 2, February 1993, pp. 203-219.

[3] J. Mogul, and K. Ramakrishnan, "Eliminating Receive Livelock In An Interrupt-Driven Kernel," *ACM Trans. Computer Systems*, vol. 15, no. 3, August 1997, pp. 217-252.

[4] R. Morris, E. Kohler, J. Jannotti, and M. Kaashoek, "The Click Modular Router," *ACM Transactions on Computer Systems*, vol. 8, no. 3, August 2000, pp. 263-297.

[5] A. Indiresan, A. Mehra, and K. G. Shin, "Receive Livelock Elimination via Intelligent Interface Backoff," TCL Technical Report, University of Michigan, 1998.

[6] P. Druschel, "Operating System Support for High-Speed Communication," *Communications of the ACM*, vol. 39, no. 9, September 1996, pp. 41-51.

[7] P. Druschel, and G. Banga, "Lazy Receive Processing (LRP): A Network Subsystem Architecture for Server Systems," Proceedings Second USENIX Symposium. on Operating Systems Design and Implementation, October 1996, pp. 261-276.

[8] P. Shivan, P. Wyckoff, and D. Panda, "EMP: Zero-copy OS-bypass NIC-driven Gigabit Ethernet Message Passing," Proceedings of SC2001, Denver, Colorado, USA, November 2001.

[9] C. Dovrolis, B. Thayer, and P. Ramanathan, "HIP: Hybrid Interrupt-Polling for the Network Interface," *ACM Operating Systems Reviews*, vol. 35, October 2001, pp. 50-60.

[10] Alteon WebSystems Inc., "Jumbo Frames," http://www.alteonwebsystems.com/products/white_papers/jumbo.htm

[11] A. Gallatin, J. Chase, and K. Yocum, "Trapeze/IP: TCP/IP at Near-Gigabit Speeds", Annual USENIX Technical Conference, Monterey, Canada, June 1999.

[12] C. Traw, and J. Smith, "Hardware/software Organization of a High Performance ATM Host Interface," *IEEE JSAC*, vol.11, no. 2, February 1993.

[13] C. Traw, and J. Smith, "Giving Applications Access to Gb/s Networking," IEEE Network, vol. 7, no. 4, July 1993, pp. 44-52.

[14] L. Kleinrock, Queueing Systems: Theory, vol 1, Wiley, 1975

[15] K. Salah and K. El-Badawi, "Performance Evaluation of Interrupt-Driven Kernels in Gigabit Networks", IEEE Globecom 2003, San Francisco, USA, December 1-5, 2003, pp. 3953-3957

[16] K. Salah and K. El-Badawi, "Analysis and Simulation of Interrupt Overhead Impact on OS Throughput in High-Speed Networks," *International Journal of Communication Systems*, vol. 18, no. 5, Wiley Publisher, June 2005, pp. 501-526

[17] W. Leland, M. Taqqu, W. Willinger, D. Wilson, "On the Self-Similar Nature of Ethernet Traffic", *IEEE/ACM Transaction on Networking*, vol. 2, no. 1, February 1994, pp. 1-15

[18] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling," IEEE/ACM Transactions on Networking, vol. 3, no. 3, June 1995, pp. 226-244

[19] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson, "Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," Proceedings of ACM SIGCOMM, Cambridge, Massachusetts, August 1995, pp. 100-113.

[20] P. Steenkiste, "A Systematic Approach to Host Interface Design for High-Speed Networks," *IEEE Computer magazine*, Vol. 24, No. 3, March 1994, pp. 44-52.

[21] K. Kochetkov, "Intel PRO/1000 T Desktop Adapter Review," http://www.digit-life.com/articles/intelpro1000t

[22] 3Com Corporation, "Gigabit Server Network Interface Cards 7100xx Family," http://www.costcentral.com/pdf/DS/3COMBC/DS3COMBC109285.PDF

[23] J. Brustoloni and P. Steenkiste, "Effects of Buffering Semantics on I/O Performance ," Proceedings Second USENIX Symposium. on Operating Systems Design and Implementation, October 1996, pp. 277-291.

[24] Z. Ditta, G. Parulkar, and J. Cox, "The APIC Approach to High Performance Network Interface Design: Protected DMA and Other Techniques," Proceeding of IEEE INFOCOM 1997, Kobe, Japan, April 1997, pp. 179-187.

[25] H. Keng and J. Chu, "Zero-copy TCP in Solraris," Proceedings of the USENIX 1996 Annual Technical Conference, January 1996.

[26] A. Law and W. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, 2nd Edition, 1991.

[27] J. White, "An Effective Truncation Heuristic for Bias Reduction in Simulation Output," *Simulation Journal*, vol. 69, no. 6, December 1997, pp. 323-334

[28] M. Aron and P. Druschel, "Soft Timers: Efficient Microsecond Software Timer Support for Network Processing," *ACM Transactions on Computer Systems*, vol. 18, no. 3, August 2000, pp. 197-228.

[29] A. Foong, T. Huff, H. Hum, J. Patwardhan, and G. Regnier, "TCP Performance Re-Visited," IEEE Symposium on Performance of Systems and Software, March 2003, pp. 70-79

[30] L. McVoy and C. Staelin, "lmbench: Portable Tools for Performance Analysis," Proceedings of the 1996 USENIX Technical Conference, San Diego, CA, January 1996, pp. 279-295.

[31] Ashford Computer Consulting Service, "GigaBit Ethernet to the Desktop – Client1 System Benchmarks", 2004, http://www.accs.com/p_and_p/GigaBit/Client1.html

[32] T. Dunigan, "ORNL Opteron Evaluation – lmbench 2.0 Summary", http://www.csm.ornl.gov/~dunigan/opteron-1.5/lm.rpt

[33] A. Rubini and J. Corbet, “The Linux Device Drivers,” O’Reilly, 2001.

[34] M. Karam and F. Tobagi, “Analysis of Delay and Delay Jitter of Voice Traffic in the Internet,” *Computer Networks Magazine*, vol. 40, no. 6, December 2002, pp. 711-726.

[35] K. Park and W. Willinger, “Self-Similar Network Traffic and Performance Evaluation,” John Wiley & Sons, Inc, 2002.

[36] M. S. Taqqu, W. Willinger, and R. Sherman, “Proof of a Fundamental Result in Self-Similar Traffic Modeling” ACM/SIGCOMM Computer Communication Review, vol. 24, no. 2, 1997, pp. 5-23.

[37] K. C. Claffy, G. Miller, and K. Thompson, “The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone,” In the Proceedings of INET 1998, Geneva, Switzerland, July 1998.

[38] M. Crovella and L. Lipsky, “Long-Lasting Transient conditions in Simulations with Heavy-Tailed Workloads,” Proceedings of the 1997 Winter Simulation Conference, Atlanta, GA, pp. 1005-1012.