An Analytical Model for Evaluating Interrupt-Driven System Performance of Gigabit Ethernet Hosts with Finite Buffer

Khaled Salah
Department of Information and Computer Science
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
Email: salah@ccse.kfupm.edu.sa

Abstract

A novel analytical model based on Markov processes is developed to study the impact of interrupt overhead on operating system performance of network hosts such as PC-based routers, servers, and end hosts when subjected to Gigabit network traffic. Under heavy network traffic, the system performance will be negatively affected due to interrupt overhead caused by incoming traffic. In particular, excessive latency and significant degradation in system throughput can be experienced. Also, user applications may livelock as the CPU power is mostly consumed by interrupt handling and protocol processing. In this paper, we present an analytical model to evaluate system performance. The system performance is studied in terms of throughput, latency, stability condition, CPU utilizations of interrupt handling and protocol processing, and CPU availability for user applications. The analysis can be instrumental in choosing system design parameters offline, therefore allows capacity planning and system diagnosis. Analytical results are compared with the ideal limited buffer queueing system without interrupt overhead.

1. Introduction

These days a massive deployment of 1-Gigabit and 10-Gigabit Ethernet network devices has taken place. With such large network bandwidth, the network packets arrive very close to each other, causing the kernel to spend most of its time processing the incoming packets. In Gigabit networks, the arrival rate of the incoming packet can exceed the host's processing rate. If no CPU power is left for the lower priority user applications, the perceived performance by the user will significantly be degraded. This user-perceived performance can be reflected in a router that is not transmitting any packets or an interactive application that is not responding at all.

Interrupt-driven systems tend to perform very badly under such heavy network loads. 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. At low packet arrival rates, the cost of interrupt overhead for handling incoming packets are low. However, interrupt overhead cost directly increases with an increase of packet arrival rate, causing *receive livelock*.

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 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 [17], a preliminary simple throughput analysis was presented for interrupt-driven kernels when utilizing PIO and DMA in high-speed networks such as that of Gigabit Ethernet. In this paper, we present a novel Markovian analytical model to study a number of important performance metrics. Compared with the model presented in [17], the novel analytical model proposed in this paper is based on pure Markovian processes and more accurate. The presented Markovian model does not depend on the computation of the effective or disrupted protocol processing rate. In this paper, we substantially expand and study the performance in terms system throughput, system latency, host saturation point and system stability condition, CPU utilizations of ISR handling and protocol processing, and CPU availability for other processing including user applications. As opposed to prototyping and simulation, this Markovian model can be utilized to give a quick and easy way of studying the receive livelock phenomenon and system performance in high-speed and Gigabit networks. This model yields insight into understanding and predicting the performance and behavior of interrupt-driven systems at low and at very-high network traffic. Our analytical work can be important for engineering and designing various NIC 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.

The rest of the paper is organized as follows. Section 2 presents an analytical model that captures the system behavior and study the performance of Gigabit Ethernet hosts. Section 3 verifies and validates the analysis. Finally, Section 4 concludes the study and identifies future work.

2. Analysis

In this section we present a Markovian analytical model to examine the impact of interrupt overhead on OS 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. Note that $1/\mu$ is the

average time the system takes to process the incoming packet and deliver it to the user application. This time includes primarily the network protocol stack processing carried out by the kernel, excluding any time disruption due to interrupt handling. Let $1/r$ be the mean interrupt handling time, which is basically the interrupt service routine time for handling incoming packets. $1/r$ includes basically the interrupt-context switching overhead and notifying the kernel to start the protocol processing for the received packet.

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.

Throughout our analysis, we assume the following:

i) It is reasonable not to assume the times for protocol processing or ISR handling to be constant. These times change due to various OS 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 these service times to be exponential. In Section 3, we demonstrate that this assumption gives an adequate approximation.

ii) The network traffic follows a Poisson process, i.e. the packet interarrival times are exponentially distributed.

iii) The packet sizes are fixed. This assumption is true for constant traffic such as uncompressed interactive audio and video conferencing.

2.1. Ideal System

We first present an ideal system in which the overhead involved in generating interrupts is totally ignored. This system is used for comparison and verification of the analysis of the real system in which interrupt overhead is considered. 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 protocol 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., $\frac{\lambda}{\mu} > 1$. This assumption is a must for Gigabit environment where under heavy load λ can be very high compared to μ. Another reason is that hosts practically and realistically has a finite amount of buffer space reserved for protocol processing.

CPU Utilization and Availability. Using the *M/M/1/B* queueing model, the CPU utilization due to protocol processing alone, when the interrupt overhead is ignored, can be expressed as $\rho_{Idea} = \left(\frac{\lambda}{\mu}\right)$. The CPU availability, V, for other processing can be expressed as $1 - \rho_{Idea}$. Note this is true when $\frac{\lambda}{\mu} < 1$. If $\frac{\lambda}{\mu} > 1$, then CPU utilization is at 100% and $V = 0$. For $0 < \frac{\lambda}{\mu} < \infty$, one can express CPU utilization as $(1 - p_0)$, and the CPU availability as $V = p_0$. p_0 is the probability of not queueing.

Mean System Throughput. The mean system throughput, γ, in *M/M/1/B* queueing model is the rate at which packets are successfully being processed by the kernel's protocol stack and can be expressed as

$$\gamma = \mu(1 - p_0). \quad (1)$$

Saturation Point. A critical operating point for the system is computing the saturation point. It is the point at which the system can not keep up with the offered network load. This is also referred to the "cliff" point of system throughput, i.e. $\lambda = \mu$. It is where the throughput starts falling as the network load increases. Also the system will become unstable causing dropping of packets, excessive latencies and timeouts. In addition, the user applications will livelock at this point as the CPU power is at 100%, and thus resulting in $V = 0$. The CPU power is being consumed by ISR handling and protocol processing.

Mean System Latency. The mean system latency, $E(r)$, it simply the mean delay encountered in the *M/M/1/B* queueing system with $\rho_{Ideal} = \left(\frac{\lambda}{\mu}\right)$.

2.2. DMA-Based Design

For our hosts, we assume that the NIC 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. 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].

It is important to note that the device driver for the network adapters 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. Please note in this situation if two or more packets

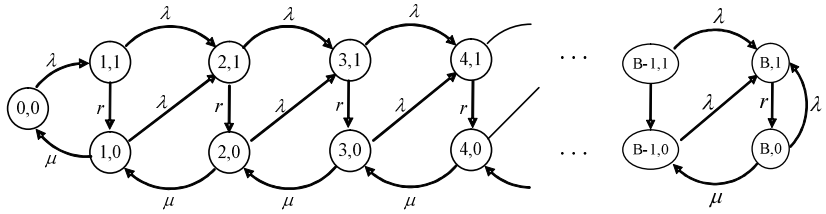

Figure 1. Markov state transition diagram for interrupt-driven system with finite buffer

arrive during an ISR handling, the interrupt servicing or acknowledgement of all of these packets will be done at the end of the current ISR handling. I.e., the ISR time for servicing all of these packets will be the ISR time for servicing a single packet.

2.3. Markov Chain Analytical Model

The interrupt-driven system with DMA can be modeled as a pure Markov chain with a state space $S = \{(n,m), 0 \le n \le \infty, m \in \{0,1\}\}$, where n denotes the number of packets in the buffer and m denotes the type of activity the CPU is performing. State (0,0) represents the state where the CPU is idle. States $(n,1)$ represent the states where the CPU is busy handling interrupts. States $(n,0)$ represent the states where the CPU is busy processing protocol. The rate transition diagram is shown in Figure1.

Let $p_{n,m}$ be the steady-state probability at state(n,m). A system of difference equations can be derived for the stationary probabilities as follows:

$$\begin{aligned} 0 &= -\lambda p_{0,0} + \mu p_{1,0} \\ 0 &= -(\lambda + r) p_{1,1} + \lambda p_{0,0} \\ 0 &= -(\lambda + \mu) p_{n,0} + r p_{n,1} + \mu p_{n+1,0} \quad \text{for } n \ge 1 \\ 0 &= -(\lambda + r) p_{n,1} + \lambda p_{n-1,0} + \lambda p_{n-1,1} \quad \text{for } n \ge 2 . \end{aligned} \tag{2}$$

The first two equations constitute the initial values. The last two equations constitute the system of difference equations. These equations can be written in the vector form as

$$p(n+1) = A\, p(n),$$

where

$$A = \begin{bmatrix} \dfrac{\lambda+\mu}{\mu} & -\dfrac{r}{\mu} \\ \dfrac{\lambda}{\lambda+r} & \dfrac{\lambda}{\lambda+r} \end{bmatrix}, \; p(n) = \begin{bmatrix} p_{n,0} \\ p_{n,1} \end{bmatrix}, \text{ and } p(n+1) = \begin{bmatrix} p_{n+1,0} \\ p_{n+1,1} \end{bmatrix}.$$ Th

erefore, our equations have been nicely converted to a system of first order difference equation, in which we can apply Putzer algorithm to obtain the solution [24].

Before we proceed further, let us denote $\alpha = \lambda/\mu$, and $\beta = \lambda/(\lambda + r)$. The eigenvalues of matrix A can be obtained by solving the characteristic equation $\det(A - zI) = 0$ where z is the eigenvalue, and I is the identity matrix. Now

$$\begin{aligned} \det(A - zI) &= \det\begin{bmatrix} \alpha + 1 - z & -\alpha(1-\beta)/\beta \\ \beta & \beta - z \end{bmatrix} \\ &= (1-z)(z - \alpha - \beta) = 0. \end{aligned}$$

Hence, the eigenvalues of matrix A are $z_1 = 1$ and $z_2 = \alpha + \beta$. So, according to Putzer Algorithm,

$$M(0) = I, \quad \text{and} \quad M(1) = A - z_1 I = \begin{bmatrix} \alpha & -\alpha(1-\beta)/\beta \\ \beta & \beta - 1 \end{bmatrix}$$

Then, $u_1(n) = 1^n = 1$, and

$$u_2(n) = \sum_{i=0}^{n-1} (\alpha+\beta)^{n-1-i} (1^i) = \frac{1-(\alpha+\beta)^n}{1-(\alpha+\beta)}.$$

Finally, we have

$$\begin{aligned} A^n &= u_1(n) \times M(0) \;+\; u_2(n) \times M(1) \\ &= \begin{bmatrix} \dfrac{1-\beta-\alpha(\alpha+\beta)^n}{1-(\alpha+\beta)} & \dfrac{\alpha(1-\beta)(1-(\alpha+\beta)^n)}{\beta(1-(\alpha+\beta))} \\ \dfrac{\beta(1-(\alpha+\beta)^n)}{1-(\alpha+\beta)} & \dfrac{-\alpha+(1-\beta)(\alpha+\beta)^n}{1-(\alpha+\beta)} \end{bmatrix}. \end{aligned}$$

The solution of the difference equation is given by $p(n+1) = A^n p(1)$. Subsequently, this solution can be nicely simplified to

$$\left.\begin{aligned} p_{n,0} &= \alpha\, p_{0,0} (\alpha+\beta)^{n-1} \\ p_{n,1} &= \beta\, p_{0,0} (\alpha+\beta)^{n-1} \end{aligned}\right\} \quad n \ge 1 \tag{3}$$

The boundary probabilities at state (B, m) are

$$-(\lambda+\mu) p_{B,0} + r\, p_{B,1} = 0, \tag{4}$$

$$-r\, p_{B,1} + \lambda\, p_{B-1,1} + \lambda\, p_{B-1,0} + \lambda\, p_{B,0} = 0. \tag{5}$$

Substituting equation (5) into (4), we get

$$p_{B,0} = \frac{\lambda}{\mu}(p_{B-1,0} + p_{B-1,1}).$$

Using equations (2) to obtain $p_{B-1,0}$ and $p_{B-1,1}$, and then substituting them into the above equation, we get

$$p_{B,0} = \alpha\, p_{0,0} (\alpha+\beta)^{B-1}. \tag{6}$$

Now substitute equation (6) into equation (4), we have

$$p_{B,1} = \frac{\lambda}{r} p_{0,0} (\alpha+1)(\alpha+\beta)^{B-1}. \tag{7}$$

Since the summation of all probabilities is equal to 1, we get

$p_{0,0} + \sum_{n=1}^{B-1} (p_{n,0} + p_{n,1}) + p_{B,0} + p_{B,1} = 1$, and thus

$$p_{0,0} = \left[1 + \frac{(\alpha+\beta) - (\alpha+\beta)^B}{1-(\alpha+\beta)} + (\alpha + \frac{\lambda}{r}(\alpha+1))(\alpha+\beta)^{B-1} \right]^{-1}.$$

Now, let $\rho = \alpha + \beta$ and $\alpha + \lambda / r \cdot (\alpha + 1) = \rho / (1 - \beta)$, then

$$p_{0,0} = \frac{1 - \rho}{1 - \frac{\alpha}{1 - \beta} \rho^B}. \tag{8}$$

CPU Utilization and Availability. From the Markovian model, the CPU utilization for ISR handling can be derived as

$$\begin{aligned} U_{ISR} &= \sum_{n=1}^{B} p_{n,1} = \left(\sum_{n=1}^{B-1} p_{n,1} \right) + p_{B,1} \\ &= \beta \, p_{0,0} \left(\sum_{n=1}^{B-1} \rho^{n-1} \right) + \frac{\beta}{1 - \beta} (\alpha + 1) \, p_{0,0} \, \rho^{B-1} \\ &= \beta \, p_{0,0} \left(\frac{1 - \rho^{B-1}}{1 - \rho} + \frac{(\alpha + 1) \rho^{B-1}}{1 - \beta} \right) = \beta. \end{aligned}$$

Similarly, the CPU utilization for protocol processing can be derived as

$$\begin{aligned} U_{IP} &= \sum_{n=1}^{B} p_{n,0} = \alpha \, p_{0,0} \sum_{n=0}^{B} \rho^{n-1} \\ &= \alpha \left(\frac{1 - \rho^B}{1 - \rho} \right) \left(\frac{(1 - \beta)(1 - \rho)}{1 - \beta - \alpha \rho^B} \right) = \frac{1 - \rho^B}{\frac{1}{\alpha} - \frac{1}{1 - \beta} \rho^B}. \end{aligned}$$

Hence, the CPU availability for other processing can be expressed as $V = p_{0,0}$. The CPU utilization, U_{ISR+IP}, for both ISR handling and protocol processing can be expressed as $U_{ISR+IP} = 1 - V$. Also U_{ISR+IP} is equal to the sum of U_{ISR} and U_{IIP}. As a verification point, it can also be proven that $1 - p_{0,0} = U_{ISR} + U_{IP}$. When simplified, both sides of the equations yield the same term.

$$1 - p_{0,0} = U_{ISR} + U_{IP} = \frac{\rho(1 - \beta) - \alpha \rho^B}{1 - \beta - \alpha \rho^B}.$$

Mean System Throughput. The mean system throughput, γ, is the rate at which packets are successfully being processed by the kernel's protocol stack. According to [23], γ can be expressed as $\mu \sum_{n=1}^{B} p_{n,0}$. Therefore, γ can be derived as follows

$$\gamma = \mu \sum_{n=1}^{B} p_{n,0} = \mu \alpha \, p_{0,0} \sum_{n=1}^{B} \rho^{n-1} = \mu \alpha \, p_{0,0} \times \frac{1 - \rho^B}{1 - \rho}. \tag{9}$$

Saturation Point. The saturation or the cliff point occurs when

$$V = 0 \quad or \quad p_{0,0} = 0.$$

Substituting in equation (8), we get

$$\rho = 1 \quad or \quad \lambda / \mu + \lambda / (\lambda + r) = 1.$$

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

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

Since the term under the square root is always greater than one then the negative sign is neglected. Therefore, the system will be stable whenever

$$\lambda = \frac{r}{2} \left(\sqrt{1 + 4\frac{\mu}{r}} - 1 \right). \tag{10}$$

It is to be noted that this equation 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 (9) with respect to λ and setting it to zero, i.e. $\frac{d\gamma}{d\lambda} = 0$, and then solving for λ.

Mean System Latency. The mean system latency, $E(r)$, can be computed as follows

$$E(r) = \frac{E(n)}{\lambda'}, \tag{11}$$

where $E(n)$ is the expected number of packets in the system and λ' is the mean effective arrival rate. λ' is expressed in equation (9). However $E(n)$ can be derived as follows

$$\begin{aligned} E(n) &= \sum_{n=1}^{B} n (p_{n,0} + p_{n,1}) \\ &= \sum_{n=1}^{B-1} n (p_{n,0} + p_{n,1}) + B \times (p_{B,0} + p_{B,1}) \\ &= p_{0,0} \sum_{n=1}^{B-1} n \rho^n + \frac{B}{1 - \beta} p_{0,0} \, \rho^B \\ &= \left(\frac{\rho}{(1 - \rho)^2} - \frac{B(1 - \rho) + \rho}{(1 - \rho)^2} \rho^B + \frac{B}{1 - \beta} \rho^B \right) \times p_{0,0}. \end{aligned}$$

Simplifying the above equation, we get

$$E(n) = \frac{\rho(1 - \beta)(1 - \rho^B)}{(1 - \rho)(1 - \beta - \alpha \rho^B)} - \frac{\alpha B}{1 - \beta - \alpha \rho^B}.$$

3. Verification and Validation of Analysis

We verify our analysis using simulation. Also we compare our analysis results to results of a real lab experiment reported in literature. In addition, as a verification of the derived analytical equations in Section 2.3, we examine the equations when ignoring the interrupt overhead, i.e. when $r \to \infty$. It can be proven that when $r \to \infty$, $p_{0,0} = p_0$, $U_{ISR} = 0$, and $U_{ISR+IP} = 1 - V = 1 - p_0$, which mathematically equivalent to the equations of the ideal system. With a bit more equation manipulation, it is possible to verify that equation (9) for mean system throughput is also mathematically equivalent to the corresponding equation of throughput of the ideal system when $r \to \infty$. As for the mean system latency of equation (11), when $r \to \infty$, equation (11) is not mathematically equivalent to the corresponding equation of latency of the ideal system. This is due to the fact of having two series summations in deriving $E(n)$ in the pure Markovian process, as opposed to only one series summation in deriving $E(n)$ in the *M/M/1/B* queueing model. The derivation of $E(n)$ for an *M/M/1/B* queueing model

is shown in [18]. However and as expected, both equations when plotted using MATLAB are very exactly matching.

3.1. Simulation

In order to verify our analytical model, we built a discrete-event simulation using C programming and ran a wide number of simulation runs. In all cases, a perfect accordance has been verified based on the assumptions of analysis using exponential service times and Poisson arrivals. Due to the allowed space of this publication, a complete description of the simulation models will be presented in a future publication.

3.2. Numerical Examples

In this section, we report and compare results of analysis and simulation. Numerical results are given for the mean system throughput, latency, and CPU utilization. For validation, we compare our analysis results of system throughput to the DMA experimental results reported in [4]. In [4], the lab experiment basically consisted of a PC-based router, 450 MHz Pentium III, running Linux 2.2.10 OS with two Fast-Ethernet NICs with DMA. A traffic of fixed-size packets was generated back-to-back to the router. As measured by [4], the mean service time for ISR ($1/r$) was 7.7 μ seconds and the mean protocol processing time ($1/\mu$) was 9.7 μ seconds.

For all analysis and simulation runs, we fix B to a size of 1000 packets. Figure 2a, Figure 2b, and Figure 2c plot the mean system throughput, CPU utilization, and mean system latency, respectively, as a function of packet arrival rate. As for validation, we compare the experimental results for system throughput to those of analysis. CPU utilization and availability as well as system latency using real experiments were not measured in [4]. From Figure 2a, it is clear that the analysis results give an adequate approximation to real experimental measurements. The results match exactly those of experimental at light load. At high load, there is a slight difference. This difference can be contributed to the arrival characteristics of incoming packets. Other factors can also be related to internal caching, system background processes, user-to-kernel mode switching, and measurement overhead.

When examining the mean system throughput of Figure 2a and the corresponding CPU utilization and mean system latency of Figure 2b and Figure 2c, it can be noted that the saturation point for the system occurs at $\lambda = 67{,}750$ pps. At this point, the corresponding CPU utilizations for both ISR handling and protocol processing is at 100%, with CPU availability of zero. Therefore, user applications will starve and livelock at this point. In addition, the mean system delay will increase rapidly. It can also be observed from Figure 2b that if the incoming traffic increases beyond the saturation point that the CPU utilizations for ISR handling keeps increasing and IP processing keeps decreasing, as expected. This causes more instability for the system as throughput worsens and packets start being dropped, causing a significant collapse in both the system and user-perceived performance.

Figure 2c shows the mean delays for ISR handling, protocol processing, and system. In general, the figure shows that at light load the mean system latency is small but increases considerably when the system is overloaded. This will result in an undesirable excessive latency in which packets start timing out. The mean system delay takes a snake shape. For the ideal system, Figure 2c shows the mean system delay reaches the cliff point of an arrival rate of 103,093 pps, and then the delay sharply increases until it reaches a high value. After that the mean system delay stays flat. The high value is determined by the size of the queue and the mean service rate, and it is equal to B/μ. For the ideal system this value is 9.7 msec. Figure 2c shows that the mean delay also increases sharply when reaching the cliff point of an arrival rate of 67,750 pps. The delay then does not flatten off but rather slowly shoots to infinity. This is because as the arrival rate increases right after the cliff point, the mean protocol processing rate decreases, and thus resulting in a large delay.

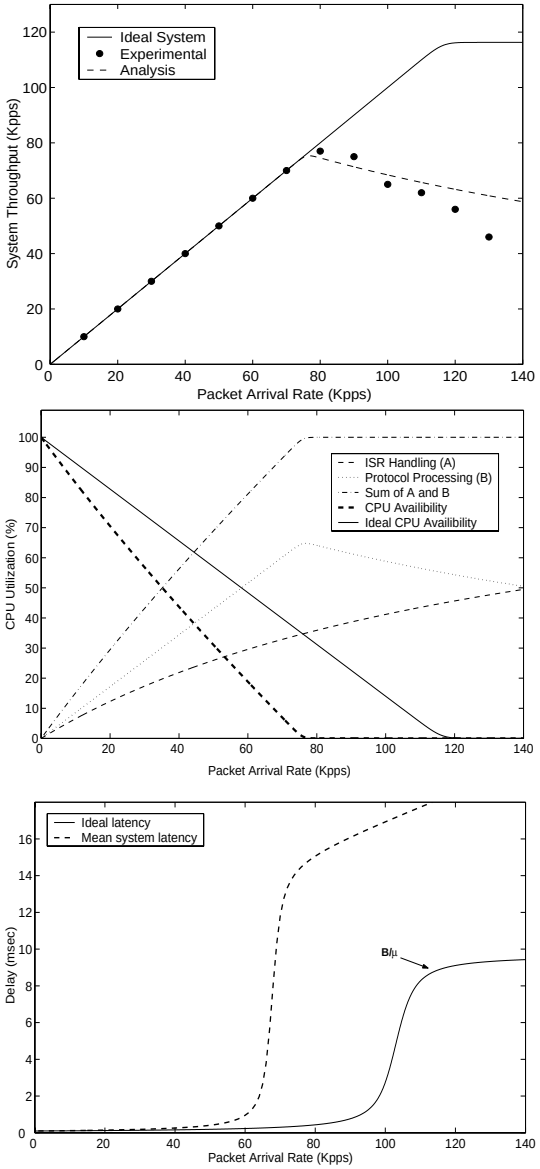

Figure 2. (a) Mean system throughput (b) CPU utilization and availability (c) Mean system latency

4. Conclusion

We developed a Markovian model to study the impact of interrupt overhead caused by Gigabit Ethernet network traffic on OS performance. Using this analytical model, we were able to conduct a throughput-delay analysis to evaluate the system performance of Gigabit Ethernet hosts when subjected to light and heavy network loads. We also investigated the system saturation point and stability condition, CPU utilizations of ISR handling and protocol processing, and CPU availability for other processing including user applications. 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. Given a worst-case network load, acceptable performance levels for throughput, delay, and CPU availability can be reached by choosing the proper system parameters for protocol processing and ISR times. An acceptable performance level varies from one system requirement to another and depends on user-application requirements and the worst tolerable system throughput and latency. The impact of generating variable-size packets instead of fixed-size and bursty traffic instead of Poisson is being studied using simulation, and results are expected to be reported in the near future. A lab experiment of 1-Gigabit links is also being set up to measure and compare the performance of different system metrics. As a further work, we are currently studying and evaluating the performance of the different proposed schemes for minimizing and eliminating the interrupt overhead caused by heavy network loads.

Acknowledgement

The author wishes to thank Mr. Khalid El-Badawi for his assistance in the simulation work. The author also acknowledges 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] 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.

[15] 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.

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

[17] K. Salah and K. Badawi, "Evaluating System Performance in Gigabit Networks", The 28th IEEE Local Computer Networks (LCN), Bonn/Königswinter, Germany, October 20-24, 2003, pp. 498-505

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

[19] W. Leland, M. Taqqu, W. Willinger, D. Wilson, "On the Self-Similar Nature of Ethernet Traffic", *IEEE/ACM Transaction on Networking*, vol. 2, pp. 1-15, 1994.

[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-ife.com/articles/intelpro1000t

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

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

[24] S. N. Elaydi, S. N., *An Introduction to Difference Equations*, Springer-Verlag, 1996, pg 113.