Performance Evaluation of Interrupt-Driven Kernels in Gigabit Networks

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

Abstract—**The paper presents models and analytical techniques for studying system behavior of an interrupt-driven kernel due to high packet arrival rate found in gigabit networks. An analytical study is presented describing the impact of high interrupt rate on system performance. The performance is studied in terms of throughput, latency, and system power. Equations are derived for system throughput, latency, power, and stability condition. Results from both reported experimental findings and simulations show that our analytical model is valid and give a good approximation. To the best of authors' knowledge, the impact of interrupts on system performance had never been studied analytically in the past, and this analytical work is the first of its kind.**

Keywords: Interrupts, Receive Livelock, Gigabit Networks, Modeling, Analysis, Performance, Operating Systems.

1. INTRODUCTION

Interrupt overhead of Gigabit network devices can have a significant negative impact on system performance. Traditional operating systems were designed to handle network devices that interrupt on a rate of around 1000 packets per second, as is the case for 10Mbps Ethernet. The cost of handling interrupts in these traditional systems was low enough that any normal system would spend only a fraction of its CPU time handling interrupts.

For 100Mbps Ethernet, the interrupt rate increases to about 8000 interrupts per second using the standard maximum 1500 byte packets. However for Gigabit Ethernet, the interrupt rate for the maximum sized-packet of 1500 bytes increases to 80,000 interrupts per second. Of course with 10 Gigabit Ethernet and considering smaller packets, the problem is much worse.

In Gigabit networks, the packet arrival rate surpasses the system packet processing rate which includes network protocol stack processing and interrupt handling. With Gigabit Ethernet and a rate of 80,000 interrupts per second for a minimum sized packet of 512 bytes, the CPU must handle an interrupt in less than 4 μs in order to keep up with such a rate. According to [1], a *null* system call (not an interrupt) on a typical 666 MHz Intel Pentium III takes on the order of 10 μs! Also, a typical latency for handling interrupt due to a packet arrival in Linux is in the order of 50 μs!

Interrupt-driven systems tend to perform very badly under such heavy load conditions. 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 and latency for handling incoming packets are low. However, interrupt overhead cost directly increases with an increasing of packet arrival rates, causing *receive livelock*.

The receive livelock condition was shown by experiments and measurements in real systems [3,4]. In this paper we present a model for the receive livelock phenomenon and show its analytical solution. These models can be utilized to understand and predict the performance and behavior of interrupt-driven systems and can be served as a reference model for comparing the performance of these proposed solutions to resolve the receive livelock condition. More importantly, the paper presents an analytical study of system performance in terms of throughput, latency, and system power due to high rate of interrupts found in Gigabit networks.

A number of solutions have been proposed to minimize the interrupt overhead and resolve receive livelock condition. Such solutions include interrupt coalescing, OS-bypass protocol, zero-copy, jumbo frames, polling, pushing some or all protocol processing to hardware, etc. Some of these solutions are listed in [2,3,4,5,6]. However none of these solutions or others, to the best of our knowledge, modeled and studied analytically the performance and behavior of system performance under heavy network loads.

The rest of the paper is organized as follows. Section 2 presents analysis for two models: an ideal system that ignores the impact of interrupts on system performance, and a second model that captures the system behavior under low and high network traffic intensity. Numerical examples are given in Section 3. A note on the accuracy of the analysis is given in Section 4. Finally, Section 5 has the conclusion and identifies future work.

2. ANALYSIS

In this section we present an analytical study to examine the impact of interrupts on system performance. At first we define system parameters. Let λ be the average incoming packet arrival rate, and μ be the average protocol processing rate by the kernel. Therefore $1/\mu$ is the time it takes the system to process the incoming packet and deliver it to the application program. This time includes primarily the network protocol stack processing by the kernel, excluding any interrupt handling. However, the interrupt handling time will be denoted as T_{ISR}, which is basically the interrupt service routine time for handling incoming packet. We will also denote ρ as a measure of the traffic intensity or system load and is defined as λ/μ.

We study the system performance in terms of three commonly-used performance metrics. These metrics include throughput, latency, and system power. System throughput (γ) is the rate at which packets are delivered by the kernel to the application program. Latency or the mean response time *(R)* which is the time duration between a packet arrival at the NIC and its delivery to the application program. Since an improvement in system throughput would a have a negative impact on latency, and vice versa, system power (*P)* was proposed in [8] which resolves this contradiction. System power gives the correct operating point that maximizes throughput and minimizes latency.

2.1 Ideal System

This section presents analysis for the ideal situation in which the overhead involved in generating interrupts is totally ignored. Assuming packets are all of fixed sizes, 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 arrival rate go beyond the service rate, i.e. $\rho > 1$. This assumption is true in Gigabit environment where under heavy load λ can be very high compared to μ.

Note. It is worth mentioning that in our analysis we assume a Poisson arrival for network traffic. It is has to be stated that that network traffic is not always Poisson in nature. However, such assumption makes analysis tractable. As we will demonstrate in Section 4 and 5, it turns out that our model with those assumptions including that of a Poisson arrival is a good approximation to an experimental model with real network traffic.

In M/M/1/B model, the system throughput can be expressed as

$$\gamma = \mu(1 - p_0), \tag{2.1}$$

where p_0 is the probability that the system is idle and given by $p_0 = \begin{cases} \dfrac{1-\rho}{1-\rho^{B+1}} & (\rho \neq 1), \\ \dfrac{1}{B+1} & (\rho = 1). \end{cases}$

System packet processing latency *R* can be given by

$$R = \frac{E(n)}{\lambda(1 - p_B)}, \tag{2.2}$$

where $E(n) = \dfrac{\rho}{1-\rho} - \dfrac{B+1}{1-\rho^{B+1}}\rho^{B+1}$, and p_B is the probability of packet being dropped due to buffer being full.

And system power is expressed by [8] as

$$P = \gamma^{\alpha}/R, \tag{2.3}$$

where α is a positive real number and is a tunable parameter. Normally, α = 1 where increasing throughput and decreasing latency are given equal weight. For our study we will set α = 1.

2.2 Impact of Gigabit-Network Interrupts

Modeling an interrupt-driven system is a challenging task especially when we consider the Gigabit networking environment where $\rho > 1$. For every incoming packet, an interrupt is initiated. The system processes the packet by first executing the ISR and then handing it to the protocol stack where it gets processed. Hence, the system protocol processing time per packet is simply equal to $T_{ISR} + 1/\mu$. However the value of this processing time is not true all the time and it depends on the arrival time of the next packet. If the next packet arrives while handling the interrupt of a previous packet, i.e., while the system execution has not finished the current ISR, the value of this process time will be $T_{ISR} + 2/\mu$. This is true since the new interrupt is being masked off because another interrupt of the same interrupt priority level is being serviced. So a new T_{ISR} is not incurred. However, kernel time to process 2 packets by the protocol stack will be $2/\mu$.

As a good design practice, we would like to minimize the execution time of the ISR as much as possible. Therefore, we assume the primary job of the ISR is to notify the kernel of the arrival of a new packet. The notification only happens after the packet is copied by the DMA to the system host memory. This assumption is valid since in gigabit networking environment, the use of DMA becomes necessary in order to eliminate any CPU overhead involved in copying packets from the NIC to kernel memory. Major network vendors equip Gigabit NICs with DMA engines. These suppliers include 3Com, HP, Alteon owned now by Nortel, Sundace, and NetGear.

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, etc. The packet will remain in the kernel or system host memory until it is discarded or delivered to the user program or application.

We also assume that the protocol processing for packets 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.

One may think that such an interrupt-driven system can be simply modeled as a priority queueing system with preemption in which there are two arrivals of different priorities. The first arrival constitutes that for ISRs and has the higher priority. The second arrival is the arrival for incoming packets, and has the lower priority. As noted the ISR execution preempts protocol processing. However this is an invalid model because ISR handling is not counted for every packet arrival. ISR handling is ignored if the system.is servicing another interrupt of the same level. In other words, if the system is currently executing another ISR, the new ISR which is of the same priority interrupt level will be masked off and there will be no service for it.

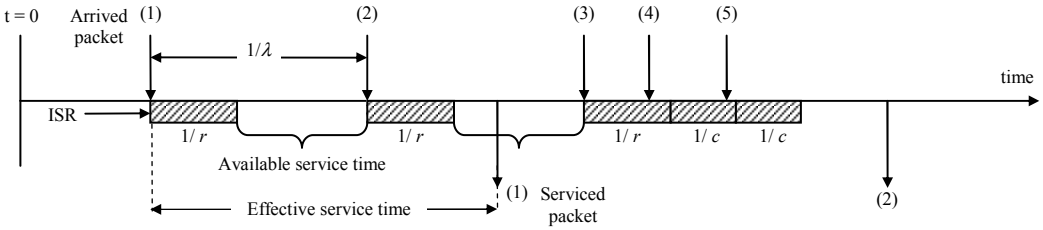

Figure 1. Effective service time

2.3 Mean Effective Service Time

In this section, we find the mean effective service time for processing packets in the kernel protocol stack. We first find the formula for the mean effective service time. Knowing this formula, the system can be modeled as an M/G/1 queue with a Poisson packet arrival rate of λ and a mean effective service rate of μ' that takes a general distribution.

As illustrated in Figure 1, the effective service time is the actual time available for servicing a packet, exclusive of T_{ISR} disruption. The available service time is the available time between successive T_{ISR}'s. If a packet or multiple packets arrive during T_{ISR}, we will have batched or masked-off interrupts and the packets will be queued into the system with effectively one T_{ISR} disrupting the service time. Therefore, the disruption of the service time is mainly influenced by the arrival rate of the packets λ and T_{ISR}.

Let us assume that T_{ISR} is exponentially distributed with mean $T_{ISR} = 1/r$. One can express the mean effective service rate as:

μ' = Rate at which packets are processed by the kernel's network protocol with no interrupt disruption.

Therefore,

$$\mu' = \mu \cdot (\%\text{ CPU availability for protocol processing}). \quad (2.4)$$

In order to determine the CPU availability percentage for protocol processing and interrupt handling, we use a Markov process to model the CPU usage, as illustrated in Figure 2. 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 $0 < n < \infty$ represent the state where the CPU is busy handling interrupts. n denotes the number of packet arrivals that are being batched or masked off during T_{ISR}. Note that when process is in state (1,0), this means there are no interrupts being masked off and the CPU is handling a single interrupt.

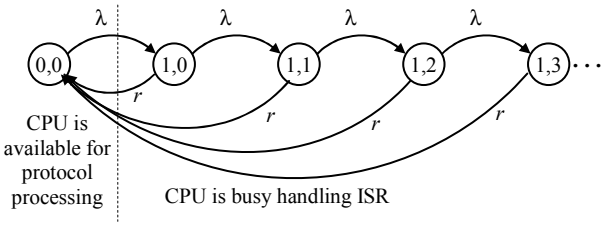

Figure 2. Markov state transition diagram for modeling CPU usage

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 $p_{0,0} + \sum_{i=0}^{\infty} p_{1,i} = 1$, 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 percentage of the CPU available for protocol processing packets and handling interrupts are $r / (\lambda + r)$ and $\lambda / (\lambda + r)$, respectively. And thus, the mean effective service rate can be expressed as:

$$\mu' = \mu \cdot \frac{r}{\lambda + r}. \quad (2.5)$$

It is to be noted from equation (2.4) that the mean effective service rate μ' is exponential. Therefore, we can model the system as M/M/1/B queue as is the case for the ideal system. However, the mean service rate μ will be replaced by the mean effective service rate μ'. Hence, the system throughput γ, latency R, and power P are expressed by equations (2.1), (2.2), and (2.3), respectively.

A particular point of interest is finding the stability condition for the system. The stability condition is the situation where $\rho < 1$, or is defined as the "cliff" point for system throughput. It is where the throughput starts falling to zero as the system load increases. The stability condition for the system 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

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

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

$$\lambda < \frac{r}{2}\left(\sqrt{1 + 4\frac{\mu}{r}} - 1\right). \qquad (2.6)$$

Another interesting point is finding the maximum system power point. This point is also the system correct operating point which gives maximum throughput and the minimum latency. In order to accomplish this, we take the derivative of the power function with respect to λ, and solving the derivative after making it equal to zero. From [9], the maximum power point occurs when $\rho < 1$. Hence, it is suitable to model the system in this case only as M/M/1, since there is no need to consider the case when $\rho > 1$ as we all along assumed. For this case, the throughput and latency as a function of λ are denoted by $\gamma(\lambda)$ and $R(\lambda)$, respectively.

$$\gamma(\lambda) = \mu'(1 - p_0) = \mu'\left(1 - \left(1 - \frac{\lambda}{\mu'}\right)\right) = \mu'\left(\frac{\lambda}{\mu'}\right) = \lambda .$$

$$R(\lambda) = \frac{E(n)}{\lambda} = \frac{\frac{\lambda}{\mu' - \lambda}}{\lambda} = \frac{1}{\mu' - \lambda} .$$

$$\therefore P(\lambda) = \frac{\lambda(\gamma)}{R(\lambda)} = \frac{\lambda}{1/(\mu' - \lambda)} = \lambda(\mu' - \lambda) .$$

Taking the derivative of $P(\lambda)$,

$$\frac{dP(\lambda)}{d\lambda} = \mu' - 2\lambda$$

Setting $dP/d\lambda = 0$, we get $\lambda = \frac{1}{2}\mu'$.

Thus, the maximum power point occurs when

$$\lambda = \frac{r}{2}\left(\sqrt{1 + 2\frac{\mu}{r}} - 1\right). \qquad (2.7)$$

2.4 Special Case

We consider a special case when interrupt handling is ignored, i.e., when $T_{ISR} = 0$. In this situation when $T_{ISR} = 0$, $r \to \infty$. We prove that equations (2.5), (2.6), and (2.7) yield the same equations of the ideal system model, i.e., M/M/1/B queueing system, as follows:

For mean effective service rate of equation (2.5),

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

For finding λ for stability condition of equation (2.6),

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

Applying L'Hopital Rule, we get

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

And finally for finding λ that gives the maximum system power point of equation (2.7),

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

By applying L'Hopital Rule, we get

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

3. NUMERICAL EXAMPLES

In this section, we report some numerical results of our analytical model to study the behavior of the system and the impact of interrupts on system performance. The system performance is studied as a function of traffic intensity ρ. Numerical results are also given for the ideal system when ignoring interrupts. For all of these results, we fix μ to 1 and B to a size of 1000.

We first examine the system throughput as a function of traffic intensity ρ. We study this relation with three T_{ISR} time units 0.2, 0.3, and 0.5. A T_{ISR} time unit of 0.2 means that the interrupt service duration is 20% of the duration of the packet protocol processing time $1/\mu$.

Figure 3 depicts the impact of high and low traffic intensity on system throughput. We note for the ideal system, the throughput is the expected one and matches very closely to the behavior of receive livelock. However, the throughput is different when considering interrupts impact, i.e., the receive livelock phenomenon. We note that the throughput doesn't fall rapidly to zero due to interrupt batching as illustrated in Section 2.3. Figure 3 shows the system throughput for three cases of T_{ISR} 0.2, 0.3, and 0.5. It is noted that as the interrupt overhead increases, i.e., increasing the value of T_{ISR}, the system throughput is degraded and the livelock phenomenon occurs earlier.

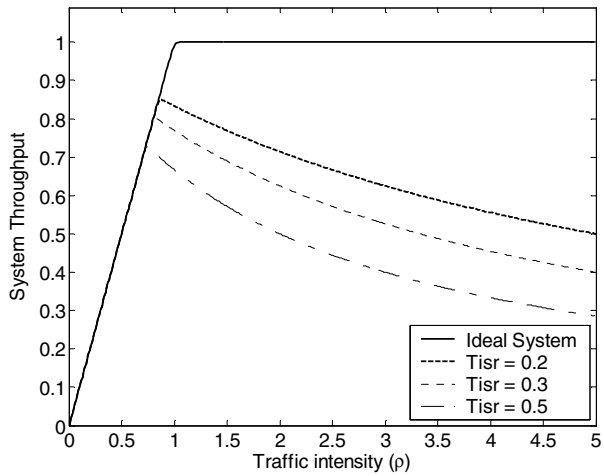

Figure 3. System throughput vs. traffic intensity

Figure 3 also shows the cliff points for the system throughput. As previously defined, the cliff points are those points where system throughput starts falling to zero as the system load increases. As shown, the cliff points in terms of traffic intensity ρ for T_{ISR} of 0.2, 0.3, and 0.5 are 0.85, 0.81, and 0.73, respectively. Since we are fixing μ to 1, the cliff points are the same for the system throughput, traffic intensity, and packet arrival rate. These points match exactly the points derived by equation (2.6) for finding the stability condition.

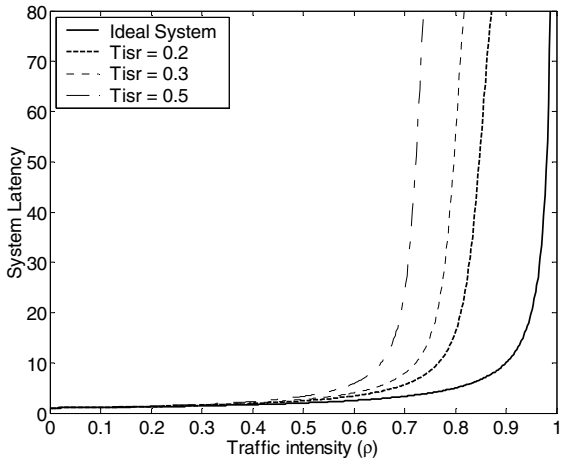

Figure 4. System latency vs. traffic intensity

Figure 4 illustrates the relation between packet latency and traffic intensity for the same system parameter values considered for system throughput. It is shown that the latency for the ideal system is the least and it is the worst when T_{ISR} is equal to 0.5.

The impact of low and high traffic intensity on system power is shown in Figure 5. In the ideal system, the maximum system power is when $\rho = 0.5$. However, the maximum system power decreases with different values of $T_{ISR,}$, giving the least value for T_{ISR} = 0.5. In addition the figure shows that the maximum power point for the system for T_{ISR} of 0.2, 0.3, and 0.5 are for λ of 0.46, 0.45, and 0.41, respectively. These points match also exactly the points derived by equation (2.7) for finding λ that gives the maximum power point.

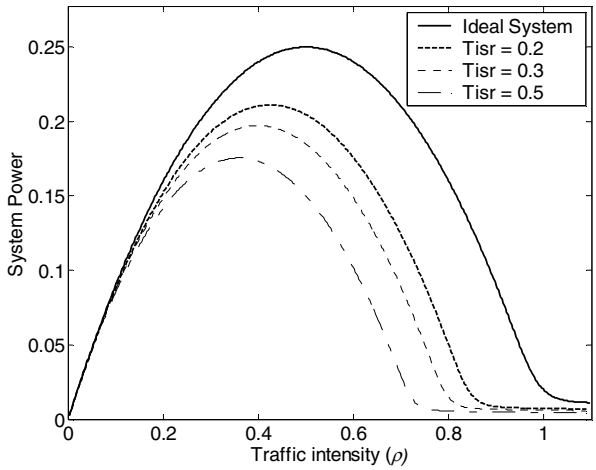

Figure 5. System power vs. traffic intensity

4. VERIFICATION AND VALIDATION OF ANALYSIS

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. The analysis was also verified by proving that all derived equations yield the same as these of the ideal system model when considering the special case of ignoring the handling of interrupts. In addition, our analytical results were compared to results from experimental findings reported by [3,4], in particular for system throughput. Our analytical results are very much inline with these reported experimental results.

5. CONCLUSION

We presented a valid analytical model that captures the behavior of interrupt-driven systems when subjected to high interrupt rates. We proposed and studied two models: an ideal system that ignores the impact of interrupts on system performance, and a second model which captures the system behavior under low and high traffic intensity. Simulation and reported experimental results show that our analytical model is valid and give a good approximation. As a further study, we will evaluate the performance of the different proposed solutions for decreasing interrupt overhead and resolving the receive livelock problem.

ACKNOWLEDGMENT

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

REFERENCES

[1] W. Feng, "Is TCP an Adequate Protocol for High-Performance Computing Needs?" *Proceedings of SC2000*, Dallas, Texas, USA, November 2000.

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

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

[6] 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.

[7] 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.

[8] A. Giessler, J. Haanle, A. Konig, and E. Pade, "Free Buffer Allocation – An Investigation by Simulation," *Computer Networks*, vol 1., no. 3, July 1978, pp. 191-204

[9] L. Kleinrock, "On the Modeling and Analysis of Computer Networks," *Proceedings of the IEEE*, vol. 81, no. 8, August 1993, pp 1179-1191.