Available online at www.sciencedirect.com
ScienceDirect
Int. J. Electron. Commun. (AEÜ) 61 (2007) 215–225
International Journal AEÜ of Electronics and Communications
www.elsevier.de/aeue

To coalesce or not to coalesce

Khaled Salah

Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

Received 12 January 2006

Abstract

System performance of Gigabit network hosts can be severely degraded due to interrupt overhead caused by heavy incoming traffic. One of the most popular solutions to mitigate such overhead is interrupt coalescing in which a single interrupt is generated for multiple incoming packets. This is opposed to normal interruption in which an interrupt is generated for every incoming packet. In this paper we investigate the performance of interrupt coalescing analytically and compare it with that of normal interruption. We consider two types of coalescing (viz. count-based and time-based). The system performance is studied in terms of throughput, CPU availability for user applications, latency and packet loss.

Keywords: High-speed networks; Operating systems; Interrupts; Interrupt coalescing; Modeling and analysis; Performance evaluation

1. Introduction

Under heavy traffic load, such as that of Gigabit networks, the performance of interrupt-driven systems can be degraded significantly resulting in a poor host performance perceived by the user. 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 has absolute priority over all other tasks by definition. If interrupt rate is high enough, the system will spend all of its time responding to interrupts, and hence the system throughput will drop to zero. This situation is called *receive livelock* [1]. In this situation, the system is not deadlocked but causing tasks scheduled at a lower priority to starve.

A number of solutions has been proposed in the literature [1–13] to mitigate interrupt overhead and improve 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. One of the most popular solutions to mitigate the interrupt overhead for Gigabit network hosts is interrupt coalescing. In recent years, almost all network adapters or network interface cards (NICs) are manufactured to have interrupt coalescing (IC). IC is a feature in which the NIC generates a single interrupt for a group of incoming packets. This is opposed to normal interruption in which the NIC generates an interrupt for every incoming packet.

Although interrupt coalescing is an important feature that is widely available to mitigate interrupt overhead and improve performance, little research has been conducted to study its performance. In [2], a time-based interrupt coalescing was studied using NIC emulation. The OS provided overload conditions to the NIC to adjust the coalescing time. In [5,6], the impact of hosts using interrupt coalescing on the overall bandwidth and latency of IP networks was investigated experimentally. In [7], the performance of interrupt coalescing was analyzed using an experiment that consisted of modifying the Linux kernel of a gateway.

A preliminary brief analytical work was presented in [14] to address the issue of latency in IC and compare it with that of normal interruption. Also an integrated performance evaluating criterion was introduced in [14] to select between IC

E-mail address: salah@kfupm.edu.sa.

1434-8411/$ - see front matter © 2006 Elsevier GmbH. All rights reserved.
doi:10.1016/j.aeue.2006.04.007

and normal interruption. In sharp contrast to the work of [14], this paper is different in significant ways. This paper extends and enhances significantly the analytical work. In this paper we study and analyze two coalescing types of IC (viz. time-based IC and count-time IC). In [14] only count-based IC was briefly studied. The numerical examples given in this paper are more realistic and based on modern hardware parameters. This paper provides in-depth analysis (and recommendations) on selecting the proper value for coalescing parameters for both IC types. The analytical models presented in this paper are based on queueing theory and Markov processes. The host performance is studied in terms of system throughput, system latency, host saturation point and system stability condition, CPU utilizations of Interrupt Service Routine (ISR) handling and protocol processing, and CPU availability for user applications.

The rest of the paper is organized as follows. Section 2 presents analytical models that capture the coalescing behavior and study the performance of Gigabit Ethernet hosts. Section 3 shows numerical examples to compare and validate analysis. Finally, Section 4 concludes the study and identifies future work.

2. Analysis

With almost all today's Gigabit NICs and under normal interruption, an incoming packet gets transferred (or DMA'd) through the PCI bus from the NIC to the protocol processing buffer of the kernel. After the packet has been successfully DMA'd, the NIC generates an interrupt to notify the kernel to start protocol processing on the incoming packet. Protocol processing typically involves TCP/IP processing of the incoming packet and delivering it to user applications. During protocol processing other packets may arrive and get queued. Protocol processing time is affected by the interrupts of incoming packets. Interrupt handling has an absolute priority over protocol processing. If an interrupt occurs during protocol processing, the protocol processing will be disrupted or preempted (i.e. protocol processing will stall until the completion of interrupt handling).

2.1. Ideal and normal interruption

In previous work [14], we presented analytical models to study two types of interrupt handling schemes (viz. ideal scheme and normal interruption). In ideal scheme, the overhead involved in generating interrupts is totally ignored. The ideal scheme gives the best performance that can possibly be obtained when employing interrupts, thus serving as a reference (or a benchmark) to compare with. In normal interruption, every incoming packet causes an interrupt. Closed-form solutions for a number of performance metrics can be found in [14].

2.2. Interrupt-coalescing

There are two types of interrupt coalescing to mitigate the rate of interrupts (viz. count-based and time-based). In this section, we first present analysis for count-based IC and then discuss the analysis for time-based IC. In count-based IC, the NIC generates an interrupt when a predefined number of packets has been received. In time-based IC, the NIC waits a predefined time period before generating an interrupt. During this time period, multiple packets can be received. It is very important to recognize that the time period gets restarted only when the previous time period has expired and a fresh packet has been received.

Our analytical approach is based on first determining the portion of CPU power (or CPU utilization) consumed by interrupt handling. A Markov process is used to compute such utilization. Knowing the CPU utilization of interrupt handling, one can then find the mean effective protocol processing rate. The mean effective protocol processing rate is actually the protocol processing rate taking into account the disruption factor of interrupt handling. Lastly, a discrete-state continuous-time Markov process is used to model a finite-buffer queueing system with this mean effective processing rate. In addition, the Markov process captures the coalescing behavior.

2.2.1. Count-based IC

For comparison purposes, we will use the same assumptions and notations presented in [14]. We assume Poisson incoming traffic, fixed packet sizes, and exponential times for interrupt handling and protocol processing.

Let λ denote the mean incoming packet arrival rate,
μ denote the mean protocol processing rate carried out by the kernel, and thus $1/\mu$ becomes 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, and
$1/r$ denote the mean ISR or interrupt handling time (i.e. the interrupt service routine time for handling incoming packets). $1/r$ basically includes the interrupt-context switching overhead as well as the ISR handling. The main function of ISR handling is to notify the kernel to start protocol processing of the received packet.
τ denote the coalescing parameter for the predefined number of packets to be coalesced before initiating an interrupt.

Thus, the interrupt frequency or rate is mitigated to $I_{\text{freq}} = \lambda/\tau$.

It is important to notice that when $\tau = 1$, an interrupt is generated per packet (i.e. the NIC resorts to normal interrupting with $I_{\text{freq}} = \lambda$).