Design and Evaluation of New Power Management Methods to Reduce Direct and Induced Energy Use of the Internet

by

Priyanga Chamara Gunaratne

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Kenneth J. Christensen, Ph.D.
Rafael A. Perez, Ph.D.
Miguel A. Labrador, Ph.D.
Stephen W. Suen, Ph.D.
Hüseyin Arslan, Ph.D.

Date of Approval:
December 8, 2006

Keywords: Power Consumption, Networks, Ethernet, Adaptive Link Rate, Performance Evaluation

Acknowledgements

I would like to express my gratitude to my graduate adviser, Dr. Kenneth J. Christensen, for his guidance, encouragement, and support over the last four years. Being advised and mentored by Dr. Christensen has been a very rewarding learning experience for me. My sincere thanks go to Dr. Rafael A. Perez, Dr. Miguel A. Labrador, Dr. Stephen W. Suen, and Dr. Hüseyin Arslan for serving on my committee. I am deeply grateful to Dr. Suen for his kind advice and assistance with the derivations in Chapter 5. I would also like to thank Dr. Gihan V. Dias at the University of Moratuwa in Sri Lanka for kindling my interest in a research career and Dr. Manjriker Gunaratne at the University of South Florida for his help and advice.

I am grateful for the encouragement and support I have received from my wife and my parents. I truly am grateful for my wife's patience, trust, and kindness.

Table of Contents

List of Tables

List of Figures

Design and Evaluation of New Power Management Methods to Reduce Direct and Induced Energy Use of the Internet

Priyanga Chamara Gunaratne

ABSTRACT

The amount of electricity consumed by devices connected to the Internet in the U.S. has rapidly increased and now amounts to over 2% of total electricity used, which is about 74 TWh/yr costing over $6 billion annually. This energy use can be categorized as direct and induced. Much of this energy powers idle links, switches, and network-connected hosts and is thus wasted.

This dissertation contains the first-ever investigation into the energy efficiency of Ethernet networks. A method for matching Ethernet link data rate with link utilization, called Adaptive Link Rate (ALR), is designed and evaluated. ALR consists of a mechanism to change the link data rate and a policy to determine when to change the data rate. The focus of this dissertation is on the analysis and simulation evaluation of two ALR policies. The simplest ALR policy uses output buffer thresholds to determine when to change data rate. This policy is modeled using a Markov chain. A specific challenge was modeling a state-dependent service rate queue with rate transition only at service completion. This policy was shown to be unstable in some cases, and an improved policy based on explicit utilization measurement was investigated. This more complex policy was evaluated using simulation. A synthetic traffic generator was developed to create realistic synthetic network traffic traces for the simulation evaluation. Finally, an improved method for detecting long idle periods using quantile estimation was investigated.

Characterization of network traffic showed that proxying by a low power device for a high power device is feasible. A prototype proxy for a Web server was developed and studied. To maintain TCP connections during sleep time periods of a host, a new split TCP connection method was designed. The split connection method was prototyped and shown to be invisible to a telnet session.

This research has contributed to the formation of an IEEE 802.3 Energy Efficient Ethernet study group. It is thus very likely that ALR will become a standard and will achieve industry implementation and widespread deployment. This will result in energy savings of hundreds of millions of dollars per year in the U.S. alone.

Chapter 1: Introduction

1.1 Background

In 1990, the Internet was unknown to those beyond the handful of researchers and universities developing and using it. Yet, 15 years later, the Internet has grown to encompass over 450 million hosts [71]. The amount of electricity consumed by devices connected to the Internet subsequently increased from zero to a significant and rapidly increasing percentage of the energy consumption of the U.S.

The growth of the Internet has been such that the consequent increase in energy consumption has been the subject of some alarmism and controversy. In "Dig More Coal – The PCs Are Coming", an article in *Forbes* magazine in May 1999 by Huber and Mills [59], it was claimed that the Internet and the devices connected to it account for 13% of the energy consumption in the U.S. This was much greater than previous estimates and led to testimony and questions during the U.S. congressional hearings regarding the Kyoto Protocol. It is now accepted that the assumptions behind the estimates in the article by Huber and Mills are invalid, and a rebuttal can be found in "Sorry, Wrong Number" by Koomey [88].

An estimate at the end of 1999 from the Lawrence Berkeley National Laboratory (LBNL) is possibly the most authoritative estimate available currently. It estimates that the total energy consumption of office and network equipment is about 74 TWh/yr (2% of total power consumption in the U.S.) [83]. This increase in energy consumption has both financial and environmental consequences. According to the U.S. Energy Information Administration [33], the average cost of a kWh to the end-user in April 2006 was 8.5 cents [30]. At this rate, 74 TWh costs approximately $6.29 billion. Global warming due to gasses emitted by burning fossil fuels is also an issue of great concern to scientists today. In generating 1 TWh of electricity, a coal fired power plant will emit 1.05 million tons of CO_2 [18]. As the "always-connected" digital lifestyle becomes the norm and the network infrastructure needed to support such needs is built, it can be expected that significant growth in electricity consumption will occur in the future.

Power consumption caused by computer networks can be classified into two major categories – direct power consumption and induced power consumption. Direct power consumption is power consumed by the network links, hubs, switches, and routers that constitute the physical infrastructure (hereafter referred to as intermediate nodes) of the Internet. In 1999, the direct power consumption of these intermediate nodes was estimated to be 6.15 TWh [112] and was projected to increase by another 1 TWh by 2005 [112]. Network induced power consumption is by far a more subtle problem than direct power consumption. It is caused when a network host such as a desktop PC is not actively being used but is prevented from powering off or using available power management features in order to maintain network connectivity. An otherwise idle device needs to maintain network connectivity in order to serve any requests for services that might be running in that device, to keep connections to remote hosts alive by responding to TCP and application layer keep-alive messages, and to maintain ownership of dynamically assigned resources such as IP (Internet protocol) addresses assigned by a DHCP server. A common example of induced power consumption is a Microsoft Windows PC in an office that is left powered on with the power management features disabled so that the user can access data in that PC from elsewhere when needed.

The focus of this dissertation is reducing the direct and induced power consumption of networks. Specifically, this dissertation addresses the following:

1. Reducing the direct power consumption of Ethernet links.
2. Reducing the induced power consumption of network hosts by the use of protocol proxying and split TCP connections.

1.2 Motivation

This work into reducing the power consumption of network hosts and intermediate nodes is motivated by two principal factors. First, the power consumption of network links and intermediate nodes does not decrease significantly with reduced utilization. Most network links are utilized at a small fraction of capacity. Second, network hosts need to be powered on in order to maintain network connectivity and be accessible over the network thus causing the hosts to be powered on at all times. These factors are described in detail in the following sub-sections.

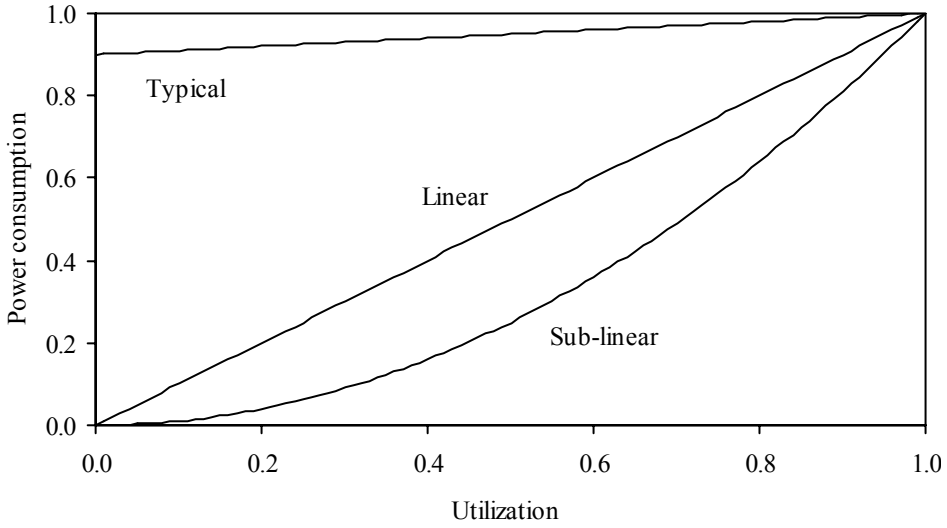


Figure 1.1. Power Consumption vs. Utilization

1.2.1 Energy Waste Due to Underutilized Ethernet Links

The power consumption of network intermediate nodes and links (and most electronic devices for that matter) does not linearly increase with utilization. As illustrated in Figure 1.1, similar levels of power consumption are typically observed whether the device is lightly or heavily utilized. For example, a Dell OptiPlex GX270 PC system unit consumes approximately 65 W when idle (measured at the wall outlet using an AC power meter), while power consumption when playing a flight simulation game with high quality graphics is approximately 110 W. Ideally, the power consumption would increase linearly (or even sub-linearly) with utilization.

The direct power consumption of Ethernet links is considered in this work. Ethernet links operate at a constant link data rate once powered on. Power consumption measurements of a Cisco Catalyst 2970 network switch and Intel PRO 1000/MT NIC (network interface controller) show that there is negligible difference in power consumption whether an Ethernet link is idle or fully utilized [49]. Ethernet NICs compatible with the 1000BASE-T specification are designed to operate at 10 Mb/s, 100 Mb/s and 1 Gb/s in order to maintain backward compatibility with previous specifications. It was observed that running the Ethernet links at lower data rates caused the power consumption of the NIC and switch to decrease [49].

Several studies [25, 100, 101] have shown that network links, especially edge links from the switch to the desktop, are lightly utilized with negligible levels of background “chatter” traffic. In the study detailed

in [100], link utilization in the Ethernet network links in the University of Toronto network was shown to be approximately 1% during a one week time period in 1998. In the study detailed in [25], Cisco NetFlow traces of the top 100 users (by traffic volume) on the University of South Florida (USF) network were collected for March 27, 2003. The link utilization (assuming 100 Mb/s link data rate) of the user with the greatest traffic volume was 7.2%. In the study described in [101], network traffic traces of the LBNL enterprise network from December 2004 and January 2005 were studied. It was shown that average link utilization is "2-3 orders of magnitude less than the capacity of the network". The full capacity of the link is rarely used and then only used to transfer brief bursts of data. This calls into question the necessity of maintaining high data rates at all times in network links.

The magnitude of possible energy savings by operating Ethernet links at a lower data rate when not fully utilized can be tens of millions of U.S. dollars. The power savings from operating an Ethernet link at 100 Mb/s instead of 1 Gb/s is about 4 W [49]. Assuming that the link operates at 100 Mb/s for 80% or more of the time, with a reasonable 100 million Ethernet links operating for 8 hours per day, an energy savings of about 0.93 TWh/yr can be achievable. This would represent a savings of $80 million/yr at 8.5 cents/kWh. In this dissertation, answers to the following questions are addressed:

1. Is it possible to transition between Ethernet link data rates while the link is active to reduce energy consumption without a user-perceivable delay? That is, can bursts of data be serviced at the full link rate while background "chatter" traffic is serviced at a lower (and energy saving) rate?
2. What policies are possible for deciding when to change data rates?
3. What is the effect of changing data rates upon packet delay and the variability of inter-packet times?

1.2.2 Energy Waste Due to Idle, But Powered-On Network Hosts

Network hosts need to maintain network connectivity during extended periods in low power states. Inability to do so leads to the host being powered on at all times. In studies conducted in 2001 and 2003, office and administrative buildings in several states were surveyed at nighttime by the Energy Analysis Program Group at the LBNL [109, 131]. It was observed on both surveys that more than 50% of desktop

PCs were powered on while less than 5% of desktop PCs had power management features enabled. A major reason for not enabling power management features in desktop PCs is that extended periods in low power "sleep" states causes numerous network connectivity problems. As mentioned in the previous section, network protocols operate under the assumption that a network host is always powered on and capable of responding to requests. A failure to do so promptly is assumed to be a failure with the host or with the network itself. It is estimated that complete enabling of power management features in office equipment could save 17 TWh/yr ($1.4 billion at 8.5 cents/kWh) – the majority of this savings is expected from desktop PCs [83]. In this dissertation, answers to the following questions are examined:

1. Can a low power consumption device be used to proxy for a high power consumption network host to enable the host to power off for extended periods while preserving service availability?
2. Can application and network protocol behavior be modified at the operating system level (with no application or protocol modifications) to make them more "power management friendly"?

1.3 Contributions

The primary research contributions of this dissertation are as follows:

1. Design, analysis, and simulation based evaluation of Adaptive Link Rate (ALR) for Ethernet. ALR adaptively varies the data rate as a function of link utilization and is the first method designed to reduce the direct energy use of Ethernet network links. In July 2005 a tutorial session on ALR was conducted at the IEEE 802 LAN/MAN Standards Committee Plenary Session [98]. This work was presented at IEEE LCN 2006 [47] and IEEE GLOBECOM 2006 [50]. At the November 2006 meeting of the IEEE 802 LAN/MAN Standards Committee Plenary Session a Call For Interest was approved with the intention of standardizing ALR.
2. Exact Markov model of a dual-threshold state-dependent service rate single server queue where service rate changes at the end of the service interval. This Markov model was used to study the performance of ALR.

3. A new method (and its implementation) for generating synthetic Ethernet network traffic traces. This synthetic traffic generator uses bounded Pareto-distributed bursts and was shown to generate synthetic traffic that closely matches the characteristics of actual (traced) traffic.
4. Design and evaluation of a new predictive device power off method that uses online quantile estimation to determine when to power off in order to reduce the direct energy use of network hosts. This work was published in [48].
5. First traffic characterization of network traffic traces captured from the Ethernet link to an idle desktop PC in order to identify and classify network traffic by relevance to proxying. This work led to the design and experimental evaluation of a prototype proxy for a web server and a prototype "shim" layer that permits the operating system to close and resume TCP connections without application awareness. This work was published in [25] and [49].

1.4 Organization of This Dissertation

The remainder of this dissertation is organized as follows.

1. Chapter 2 presents an overview of the energy use of computer and communication equipment and discusses possible energy savings.
2. Chapter 3 reviews the principles of power management and the current state-of-the-art in power management in operating systems, disk drives, and processors.
3. Chapter 4 reviews existing work and literature in power management in wired networks and briefly describes existing work in power management in wireless networks.
4. Chapter 5 describes the derivation of the Markov models for the threshold based ALR policies.
5. Chapter 6 presents the simulation results for an output-queued switch with ALR capability connected to ALR capable desktop PCs. The trade-off in energy saved versus packet delay is investigated.
6. Chapter 7 covers the quantile estimation based method for detecting long inter-packet idle periods and discusses possible energy savings.

7. Chapter 8 describes methods for reducing induced power consumption of network hosts. A detailed traffic characterization of desktop PC's network link is presented and a prototype proxy for HTTP is described. Further, split TCP connection to enable PCs to power off is described. The split TCP connection is evaluated by implementing it within a telnet client and server.
8. Chapter 9 concludes this dissertation and describes future directions for research.

Chapter 2: Energy Consumption of the Internet

Understanding the magnitude and ramifications of a problem is a necessary prerequisite to designing and evaluating possible solutions. Measurements and third-party estimates regarding the present and future power consumption of Ethernet links are presented in Section 2.1. Network induced power consumption is estimated in Section 2.2. In Section 2.3, the magnitude of possible energy savings by the use of Ethernet links capable of low data rates in response to low utilization as well as the use of desktop PCs capable of maintaining network connectivity when in low power "sleep" state is estimated.

2.1 Direct Power Consumption of Ethernet Links

The power consumption of a typical PC system unit and LAN switch were measured at the wall socket using an AC power meter. The test bed is illustrated in Figure 2.1. The present and future power consumption of Ethernet links is inferred from these measurements and publicly available data.

2.1.1 Power Consumption Measurements of Desktop PCs and Switches

The power consumption of a Dell OptiPlex GX270 PC system unit was measured at the wall socket using an Electronic Product Design, Inc PLM-1-LP Power Line AC power meter. The PC system unit was equipped with an Intel PRO 1000/MT NIC in the motherboard, which was disabled using the BIOS when the other NICs were installed. The measurements were taken with the installed NIC set to 10 Mb/s,

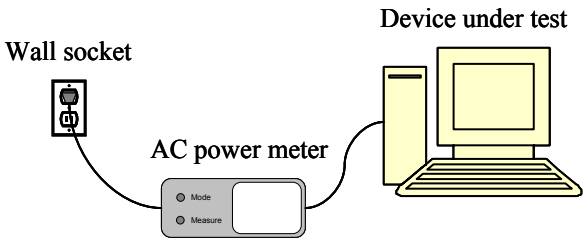


Figure 2.1. Test Bed for Measuring Power Consumption

Table 2.1. Power Consumption of PC System Unit with Different NICs

	10 Mb/s		100 Mb/s		1 Gb/s	
	On	Sleep	On	Sleep	On	Sleep
Intel PRO 1000/MT (on motherboard)	57.9 W	3.2 W	58.2 W	3.6 W	60.6 W	7.0 W
NetGear GA 311 (PCI card)	59.6	5.3	59.6	5.7	62.9	5.8
LinkSys EG1032 (PCI card)	59.1	5.6	59.6	5.5	62.0	5.6
Average change from 10 Mb/s (On=58.9 W, Sleep = 4.7 W)	N/A	N/A	0.3	0.2	2.9	1.4

100 Mb/s, and 1 Gb/s when the PC system unit was powered on and also when the unit was in Microsoft Windows XP standby sleep state. The results are summarized in Table 2.1. It can be observed that the average difference in power consumption when running the same NIC at 100 Mb/s and 1 Gb/s was approximately 2.9 W when powered on and 1.4 W when in sleep state.

The power consumption of a 24-port Cisco Catalyst 2970 switch was measured at the wall socket using an Electronic Product Design, Inc PLM-1-LP Power Line AC power meter. Measurements were taken while increasing the number of Ethernet links attached to the switch. The switch was connected to an IXIA 400T automated network traffic generator [80]. The generator interfaces were set to 10 Mb/s, 100 Mb/s, and 1 Gb/s for each successive set of measurements. 1 Gb/s data rate measurements were taken at both 100% and 0% utilization, while the other link data rate measurements were taken at 100% utilization only. The measurements are given in Figure 2.2. It can be observed that with no links attached (no load) the

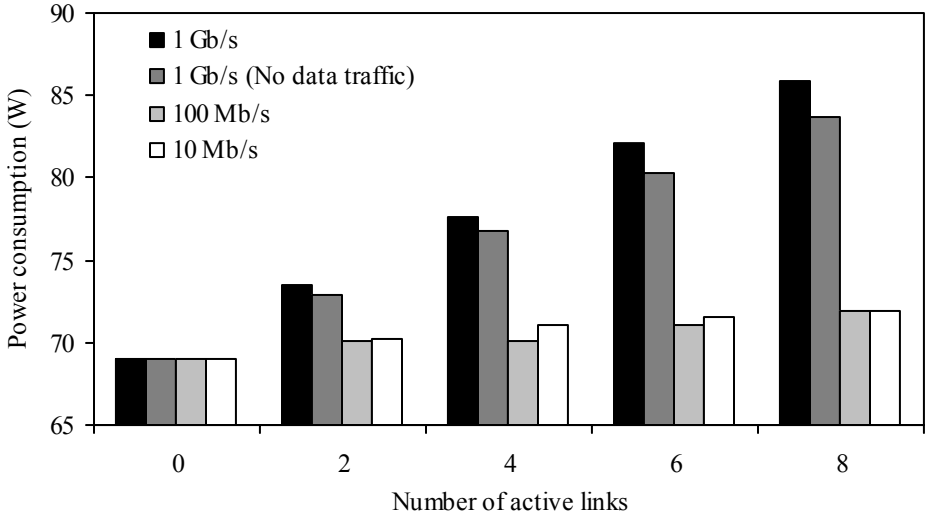


Figure 2.2. Power Consumption of Cisco Catalyst 2970 Switch

switch consumed approximately 68 W – this is power consumption with zero utilization. Looking at the columns for the 1 Gb/s data rate, it can be observed that the switch power consumption is increased by connecting a new link, even though there is no data being transmitted on this link. It can also be observed that there is no appreciable difference in power consumption whether running at 10 Mb/s or 100 Mb/s. However, increasing the data rate from 100 Mb/s to 1 Gb/s causes switch power consumption to increase by approximately 1.8 W per link. Based upon these measurements, it can be plausibly stated that an Ethernet link from a desktop PC to a switch interface consumes at least an extra 4 W of power when the link data rate is 1 Gb/s compared to when the link data rate is 100 Mb/s. This increase is consistent whether the link is fully utilized or idle.

The next step up from the 1 Gb/s link data rate for Ethernet over copper unshielded twisted pair (UTP) transmission media, the 10 Gb/s data rate, is currently under development. Ratification of standards for 10 Gb/s link data rate over copper media is expected to be completed by late 2006 [62]. Therefore, the power consumption of an S2io 10 Gb/s NIC for fiber optic transmission media was estimated by measuring the power consumption of the PC system unit with and without the NIC at the wall socket using an Electronic Product Design, Inc PLM-1-LP Power Line AC power meter. It was observed that the power consumption of the PC system unit increased by approximately 18 W with the NIC. It is thus quite possible that the next generation of 10 Gb/s Ethernet NICs will represent an increase in power consumption of at least 10 W over 1 Gb/s Ethernet NICs.

2.1.2 Reasons for the Increase in Power Consumption

Possible reasons for the increase in power consumption of NICs as data rates are increased are considered in this section. The previously mentioned measurements were taken with Ethernet NICs using UTP copper cable media. Each UTP cable consists of 4 pairs of copper wires with each wire in a pair twisted around the other (“twisted pair”). Power consumption models of Ethernet NICs are not publicly available to draw exact conclusions, but the 802.3 standards provide sufficient data to draw general conclusions. Ethernet link data rates have increased and each increase has been accomplished by:

1. Increasing the number of wires used for transmission.
2. Increasing the symbol transmission frequency.
3. Increasing the complexity of the coding scheme (more bits are transmitted per symbol).

For example, in 100 Mb/s Ethernet (100BASE-TX), each link partner transmits on one wire pair and receives on one wire pair (out of the 4 pairs in an UTP cable) at a frequency of 125 MHz, with 4B/5B symbol encoding where 5 symbols represent 4 bits [61, Clause 25]. However, for 1 Gb/s Ethernet (1000BASE-T) each link partner simultaneously transmits on 4 wires and receives on 4 wires at a frequency of 125 MHz, with each symbol representing 2 bits [61, Clause 40]. Preliminary standards for 10 Gb/s Ethernet over UTP (10GBASE-T) call for a frequency of 833 MHz with each symbol representing 3 bits [63]. Power consumption is increased by adding more transmit/receive circuits, increasing the clock frequency of the circuits, and increasing the complexity of the circuits requiring a larger number of transistors. When no data is transmitted on the link, idle bit patterns are continuously transmitted in order to keep the link partner NICs synchronized. This leads to similar NIC power consumption levels whether transmitting data or not.

Increasing power consumption as capabilities are increased can be observed in other devices as well including central processing units (CPU) and graphics processing units (GPU). With each generation of new processors, the number of transistors has grown in order to accommodate more features and the clock frequency has also increased, increasing the power consumption. For example, the Intel Pentium II 266 MHz processor introduced in 1997 has a thermal design power (TDP) of 10W [69] while the newer Intel Pentium 4 3.06 GHZ processor (introduced in 2002) has a TDP of 89W [70]. The dual-core Intel Itanium 2 1.6 GHz processor has a TDP of 104W [67]. The Pentium 4 is composed of 42 million transistors while the Itanium 2 processor is composed of 220 million transistors, leading to greater power consumption despite a slower clock frequency [68].

With the advent of graphical user interfaces and games with 3D graphics, the power consumption of GPUs has increased significantly due to the greater capabilities required. Multiple parallel rendering units are common in high-end GPUs and in some cases the power consumption of the GPU exceeds the power consumption of the CPU. The Radeon X1900 XTX consumes over 120W processing 3D graphics and the Nvidia 7950 GX2 has a reported power consumption of over 143W [37].

2.1.3 Estimates of Energy Use by Ethernet-Based LAN Switches and Hubs

Network intermediate nodes are the network links, hubs, switches, and routers that together comprise the physical infrastructure of the Internet. These are generally powered-on continuously and have no power management features. Of these device classes, hubs and LAN switches utilize Ethernet network links. A detailed study of the energy consumed by the U.S. network infrastructure was conducted by the Arthur D. Little firm for the US Department of Energy in late 1999, and the results were presented in [112]. Table 2.2 summarizes the estimated annual energy consumption (AEC) in TWh for hubs and LAN switches from the Arthur D. Little study [112]. The direct power consumption of hubs and LAN switches in the U.S. in 1999 was estimated to be 4.9 TWh. It was also estimated that the number of LAN switch ports (which are mostly Ethernet based) would increase by 17% from 2000 to 2005 [112] leading to a commensurate increase in energy consumption.

2.2 Induced Energy Consumption by the Internet

Measuring and estimating induced power consumption is a far more difficult and subtle problem than measuring and estimating the direct power consumption of the Internet. As previously mentioned, induced power consumption is defined as power consumption when a network host is not actively being used but prevented from powering off or enabling power management due to the need to maintain network connectivity. The measurements in Table 2.1 show that the power consumption of the PC system unit is reduced by over 50 W to at most 7 W in Microsoft Windows XP standby state. A PC system unit and display with a total power consumption of 100 W consumes 857kWh/yr if left always powered on, and increases the average annual household electricity consumption of 10,219 kWh by more than 8.5% (100W/hr is 0.1 kWh and 0.1 kWh × 24 × 365 / 10,219 kWh = 8.6%). However, if power management is

Table 2.2. Direct Energy Consumption of Hubs and LAN Switches

Intermediate node class	AEC (TWh)
Hubs	1.6
LAN switches	3.3
Total	4.9

used and the PC is in a low power consumption sleep state for 16 hours out of 24 and the power consumption in this state is only 10 W, the annual energy consumption drops to 350 kWh/yr, which is an increase of 3.4% of the average annual household electricity consumption. Detailed statistics on residential computer use is available for the year 2001 at the latest [107]. The average annual energy consumption of a desktop PC is 262 kWh. Note, however, that this is before the proliferation of residential broadband access to the Internet and widespread use of Peer-to-Peer file sharing applications. Participating nodes in Peer-to-Peer file sharing networks are continuously powered on.

Desktop PCs are kept powered-on in offices for many reasons. Employees need to access the data and resources on their office desktop PCs while physically being elsewhere and outside normal office hours. Office PCs are often left powered on during nighttime so that they can be accessed for upgrading and maintenance by system administrators. PCs also need to be powered on to keep dynamically allocated resources such as IP addresses assigned by DHCP and to keep connections to remote servers alive.

Results of the surveys into nighttime equipment shutdown in office buildings conducted by the LBNL in 2000 [131] and 2003 [109] indicate that the use of power management in office desktop PCs is minuscule and that the majority are powered on continuously. In [131], 11 office buildings in San Francisco and Washington, DC were surveyed at nighttime. It was observed that of 1280 desktop PCs, 54% were powered on, 44% were powered off and 3% were in low power state. In [109], 12 office buildings in San Francisco, Pittsburgh, and Atlanta were surveyed at nighttime and it was observed that of 1453 desktop PCs, 60% were powered on, 36% were powered off and just 4% were in low power state. In both surveys, less than 5% of desktop PCs in office buildings had power management features enabled while at least 50% of them were left powered on at all times.

The best available estimate of the costs of network induced power consumption at present can be found in Kawamoto et al. [83] – a detailed study of electricity use by office and network equipment in the U.S. at the end of 1999. It was found that the annual power consumption of office and network equipment was 74 TWh/yr and that complete saturation and proper functioning of power management features would save a total of 17 TWh/yr, mainly from desktop PCs, laser printers, and copiers. At 8.5 cents/kWh, 17 TWh equals to an end-user cost of $1.4 billion. However, this study was based upon certain assumptions that now appear questionable, including the following:

1. It was assumed that a desktop PC in low power state would consume 25 W, but actual values are closer to 5 W.
2. It was assumed that 25% of desktop PCs already have power management features enabled, but Roberson et al. [131] and Webber et al. [109] show that the actual rate is closer to 5% than 25%.

Therefore, the magnitude of possible energy savings by the complete saturation and proper functioning of power management features could be much greater than the 17 TWh estimated in Kawamoto et al. [83].

2.3 Potential for Energy Savings

In this section, savings in energy consumption made possible by Ethernet links capable of operating at lower data rates and full deployment of methods for allowing networked PCs to sleep while maintaining network connectivity are estimated. The estimates presented here are the work of Bruce Nordman, staff scientist at the LBNL, and were published in [49]. They are included in this dissertation for completeness.

PC usage is divided into five segments: high-traffic on, low-traffic on, possible-sleep, sleep and off. ALR only affects low-traffic on and possible-sleep time segments. Protocol proxying only affects the possible-sleep time segment. The combined residential and commercial desktop PC population in 2007 is extrapolated from current stocks to be 160 million units. The following assumptions are made:

1. 50% of all PCs are continuously powered-on machines that are potentially power managed and in a low power sleep state 75% of the time.
2. Of the remaining 50%, they are assumed to be on 20% of the time and potentially in a low power sleep state for half that time.
3. It is estimated that 80% of the in-use time is low-traffic time where low Ethernet link data rates can be used with no performance impact.
4. An average electricity price of 7.2 cents/kWh is assumed.

Table 2.3. Energy Savings Assumptions and Potentials

Parameter			Value
PC Power	On (W)		60.0
	Sleep (W) with low power links enabled		5.0
Savings	Sleep savings (W)		55.0
	Low power links savings (W)		4.0
Average PC – operating time			
Off (%)			37.5
On – high traffic plus low traffic (%)			17.5
	High traffic (%)		3.5
	Low traffic (%)		14.0
Possible sleep (%)			45.0
	Savings	From links	From PCs
	% of yr	59.0	45.0
	Hours/yr	5170.0	3940.0
	kWh/yr	21.0	217.0
	$/yr per PC	1.5	5.6
	US savings ($ billion/yr)	0.24	2.5

The calculations of possible energy savings are given in Table 2.3, and it can be observed that energy savings of $240 million is possible through the use of Ethernet links capable of low data rates in response to low utilization and energy savings worth $2.5 billion is possible through the use of methods that permit computers to maintain network connectivity while in low-power states.

Chapter 3: Overview of Power Management

According to Nordman et al., "Power-management does not reduce the performance of a computer, but simply adds features to reduce their power consumption when not in use" [99]. Through the use of power management, energy consumption of electronic devices can be reduced with minimal effect upon utility. For example, with non-mobile network hosts that draw power from wall electrical outlets, use of power management results in reduced energy consumption, less dissipated energy, and lower energy costs to the operators. With mobile network hosts that draw power from batteries included within the device itself, power management results in reduced power consumption and therefore in increased time before batteries need replacing or recharging. Ultimately, use times are increased and device weight and size are reduced due to the smaller battery capacity required for similar powered on times.

This section provides a broad overview of power management. The basic principles are described with examples of their application at several different scales – processor, disk drive, and server clusters within a data center. Existing regulatory efforts aimed at increasing the deployment of power management features are also described.

3.1 Basic Principles of Dynamic Power Management

The goals of dynamic power management are to trade-off a loss in responsiveness to user requests for a reduction in energy consumption. The basic principles behind the mechanisms and policies in use today to make electronic devices remain responsive to user requests while saving energy are:

1. Slowing down task completion rate during idle/low utilization periods.
2. Powering off during idle periods.
3. Proxying.

Generally, slowing down is applied at the circuit level, powering off during idle periods is applied at the system, component, or processor level, and proxying is applied at the system level. These are described in detail in the following sections.

Note that dynamic power management principles can be used simultaneously at different time scales. Circuits can take advantage of idle periods lasting at most microseconds or milliseconds while a desktop PC can be powered on and off during idle periods lasting at least several minutes.

3.1.1 Slowing Down

With electronic devices, reducing the rate at which a task is completed can result in lower energy expenditure to complete the task, though the time to completion is longer. Most electronic devices use synchronous logic and operate at a given clock frequency and voltage. Higher clock frequencies generally lead to greater throughput, but the increase in frequency requires a proportional increase in supply voltage. The main components of dynamic power consumption in an integrated circuit are switching power consumption, power consumption caused by short-circuit current and power consumption caused by leakage current [97]. The major component is switching power consumption. The relationship between power consumption (P), frequency (f) and voltage (V) can be expressed as $P \propto fV^2$. Since $f \propto V$, this relationship can be further simplified to $P \propto f^3$. Thus, switching power consumption of a circuit increases in proportion to the cube of the clock frequency. Therefore, reducing the clock frequency by $1/2$ will theoretically cause the power consumption to decrease to $1/8$ of its previous value. Therefore, while the time to complete a given task will double, the energy expenditure will be reduced to $1/4$ of the energy required at the higher frequency.

When idle periods or low utilization periods are detected, clock frequency and supply voltage can be reduced in order to reduce power consumption. Frequency and voltage scaling was first proposed in [132] and [19] in the context of the operating system scheduler picking the clock frequency that gives the least energy consumption while meeting the completion deadline of the job being processed. A modern

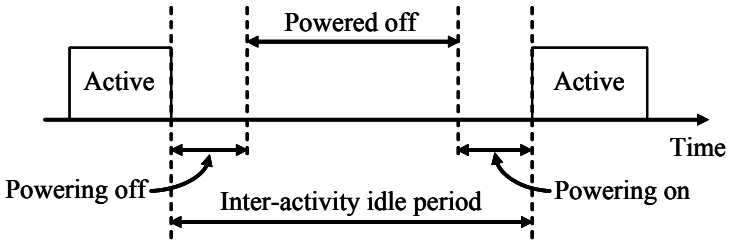


Figure 3.1. Saving Energy During Idle Periods

implementation is the SpeedStep technology in the Intel Pentium M processor [43]. Frequency is scaled from 600 MHz to 1.6 GHz in 100 MHz increments and the core voltage is scaled from 0.9 V to 1.5 V.

3.1.2 Powering Off During Idle Periods

Device use is not uniform. Periods of activity are followed by periods of idleness. By powering off components during idle periods, power consumption can be reduced with no perceivable delay to the user and over time, the overall energy consumption can be significantly reduced. When a new user request or prompt arrives, the powered off components are powered on. Powering off and powering on a device is shown in Figure 3.1. At the end of the active period, the device powers off and ideally, powers on just before the next active period begins. Since powering off and powering on components take a measurable non-zero time the key challenge here is to identify the beginning and end of idle periods so that optimal use of idle periods can be made. During the powering on time period, requests cannot be responded to and if the device powers on when a new request arrives, the time to respond (delay) is further increased.

The simplest policy is timeout-based power off, where after a given time period elapses with no activity, the device powers off and remains powered off until the next request for service arrives. An example of this is the power management method used in the Microsoft Windows XP operating system. However, there are several drawbacks with this policy. Once the device powers off, a user request arrival that causes the device to power on is not serviced until the device is powered-on In a PC, the time to power on can be several seconds. Furthermore, this policy is non-adaptive. The distribution of idle and active periods cannot be expected to be stationary. If powering off and powering on consumes extra energy, using dynamic power management under pathological conditions could lead to additional power consumption compared to leaving the device constantly powered on. The concept of *predictive power management* was

first introduced by Srivastava et al. [119]. Previous activity and idleness history is used to predict the duration of idle and active periods. A set of conditions are evaluated at the beginning of each idle period based upon the predicted duration of the idle period, and then a decision is made whether to stay powered on or power off. This concept was refined in [60] to include predictive power on in order to avoid a latency penalty. The duration of the idle period is predicted and the device is powered on by the predicted end of the idle period.

The problem of determining an optimal policy for power management was defined as a stochastic model based upon Markov decision processes and solved for stationary arrivals in [16]. This is the first work to regard power management as a stochastic optimization problem subject to constraints where past history is used to determine the transition probabilities between states. Problems with this approach include the significant computational overhead associated with computing the optimal policy and the non-stationary nature of arrivals in a realistic environment. This method was subsequently improved upon and extended by using continuous-time instead of discrete-time Markov chains [105], stochastic Petri nets [106], time indexed semi-Markov processes [115], and event windows for dealing with non-stationary arrivals [26]. It was also shown that adaptive learning trees can be used to predict arrivals [27].

In [89] an adaptive dynamic power management algorithm for wireless devices is presented. Entitled "Bounded Slowdown Protocol", it varies the sleep time period between powering on to listen to the access point and when compared with the standard 802.11 [64] power management scheme it reduces both network latency and power consumption. Irani et al. [72] consider the problem of optimal scheduling for reducing power consumption, where power consumption is convex with operating speed (increasing operating speed causes increasing power consumption) and arrivals need to be completed before a given deadline. In [73] a dynamic power management scheme that determines the probability distribution of request inter-arrival times online and determines the optimal strategy is presented.

3.1.3 Proxying

Proxying, where one device acts or assumes some functions carried out by another device, can be used in several contexts to facilitate dynamic power management in order to save energy. One approach is to use

a low power consumption proxy device capable of carrying out a subset of the functions that a higher power consumption device is capable of. The lower power proxy device can substitute for the high power device for routine tasks that do not require extensive resources thus enabling the high power device to power off for extended periods of time. Energy is saved due to the difference in power consumption between the two devices.

The first investigation of how PCs can be powered off and network connectivity be maintained was presented in [74], where the design and evaluation of a new TCP connection sleep option in a "Green TCP/IP" was described. This work was followed by an investigation of how a proxy server for TCP/IP connections could allow multiple proxy-managed clients (e.g., desktop PCs) to remain in a sleep state for long periods of time [24]. The development and experimental evaluation of a prototype proxy for the HTTP protocol was described in [25] and for the Universal Plug and Play (UPnP) protocol in [86]. In both cases the proxy would respond to routine messages (ARP requests, SSDP discover messages, etc) and maintain the "network presence" of the proxied device, waking it up only when a request for the services contained within it arrived.

The wireless network interface in mobile devices consume a significant amount of power when compared to the overall power consumption of the device. While it is possible to save energy by powering off the network interface when not transmitting data, this leads to lost packets and unpredictable latency due to the non-stationary inter-arrival times of incoming packets. A proxy connected to the wired network can be used to intercept and cache TCP and UDP packets destined for the mobile device. The mobile device powers on the network interface at periodic intervals and probes and retrieves packets from the proxy [51, 111].

A proxy can also be used to off-load energy intensive computations and data storage from battery-powered mobile network devices. In [84], a generic framework for off-loading program execution is presented; and in [40], energy efficient network storage for mobile network devices is described.

3.2 Power Management in End-User IT Equipment

End-user IT equipment, such as desktop PCs, are made up of differing components manufactured by many vendors and operate a wide variety of software. Enabling dynamic power management in such a diverse environment can be a challenge. Advanced Power Management (APM) [12] and Advanced Configuration and Power Interface (ACPI) [1] frameworks described next enable the components and software in a desktop PC to communicate with each other and enable dynamic power management for components individually as well as the whole system itself when not actively used. The operating system coordinates and controls the power management of the components.

3.2.1 APM and ACPI Frameworks for Dynamic Power Management

APM was the computer industry's first attempt to develop a standardized interface for reducing the power consumption of components within the personal computer [12]. APM was mainly developed and promoted by Intel and Microsoft. Implemented in the BIOS, APM defines several operational states for the PC. An inactivity timer is used to transition between states. For example, the CPU would be powered off in one state, while power to the random access memory (RAM) would be maintained. In a more advanced

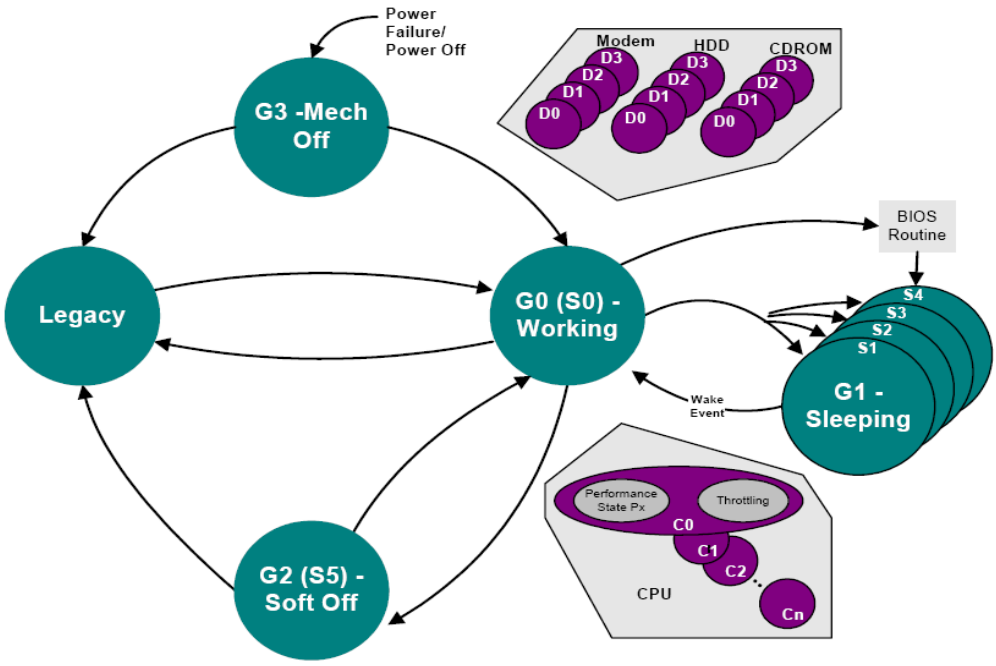


Figure 3.2. ACPI Power States (This is Figure 3-1 in [1])

sleep state, the contents of the RAM would be written to the hard drive and the RAM and the hard drive would then be powered off. However, end-user acceptance of APM was low because it was implemented solely within the BIOS and made no provision for fine-grained control by the operating system or for application requirements.

ACPI [1] was developed and released in 1996 by HP, Intel, Microsoft, Phoenix, and Toshiba to overcome the weaknesses of APM. In ACPI, power management is controlled by the operating system. Individual devices and any running programs in the computer can register their capabilities and requirements through the ACPI framework. ACPI defines states for the computer and the devices within it. For example, as shown in Figure 3.2, the states G0 to G3 define the global power state of the computer, where G0 denotes the normal powered-on state and G3 denotes the powered-off state.

3.2.2 Power Management Support in Operating Systems

Within the ACPI framework, the operating system is responsible for control and decision making for dynamic power management. When the computer is powered on, the components in the computer register their capabilities and requirements with the operating system. Control of power management decisions by the operating system via ACPI allows for fine grained control of component power consumption – components that are not being operated can be powered off or can be placed in low power states. Based on operating conditions, the operating system can decide upon various power management policies. For example, with a notebook computer that can be powered by both mains power and on-board batteries, if mains power is not available, aggressive power management policies aimed at reducing power consumption and increasing battery life can be used. Modern versions of Microsoft Windows such as Windows 2000 and Windows XP conform to the ACPI specification, while the Linux 2.6 kernel partially conforms to the ACPI specification.

Academic research in this area has tended to focus on trading off application performance in order to enhance battery life for mobile devices. In [31], Ellis proposes that power management policy be decided by application requirements and platform capabilities and that reducing energy consumption should be a design goal for operating system design. In [129], the authors reiterate that higher reductions in power

consumption are possible by designing operating systems with energy efficiency as a primary goal rather than optimizing sub-systems such as disk drives and processors for energy efficiency. Designed and developed by the authors of [129], ECOSystem [136] is a prototype operating system based upon Linux that shares energy among the processes running in a computer with the aim of reducing energy consumption. Adaptive applications that cooperate with the operating system to gracefully degrade performance as energy resources are depleted are explored in [39] using the Odyssey experimental operating system. In [14] and [15], Bellosa et al. use per-thread energy monitoring and accounting to meet thermal performance and battery life goals by energy-aware thread scheduling.

Within the context of mobile systems, there is existing work on operating system based connection resumption after failover [7] and migration of connections for mobile and distributed hosts [135, 137] by using intermediate layers in the operating system to isolate applications from the network protocol stack. These are conceptually similar to the split TCP connection method developed and evaluated in Chapter 8 of this dissertation for closing and resuming TCP connections without application awareness to avoid losing application state associated with the connection.

3.3 Examples of Dynamic Power Management in Use

The main principles of dynamic power management were described in Section 3.1 and the frameworks that permit the operating system to power manage individual components within a computer were described in Section 3.2. The application of dynamic power management principles to processors, disk drives, and individual servers in server clusters are described in the following sections.

3.3.1 Dynamic Power Management for Processors

In this section, power management mechanisms used in processors including design and compiler time (static power management) methods and dynamic power management methods are described. Powering off components in the processor that are unused by the currently executing task has been explored experimentally and commercially. The Pentium M processor predicts which units will be idle and disables the clock signal to those units [43]. The main dynamic power management technique used by commercial

processors is dynamic frequency and voltage scaling described in Section 3.1.1. The first "mainstream" processor to make reducing power consumption a priority was the Crusoe design by Transmeta [85]. Though conforming to the x86 software interfaces, the Crusoe processor was designed for low power consumption. Each x86 instruction was translated into its native instruction set before execution, and the processor was capable of adapting the operating voltage and frequency to meet performance targets. The Pentium M processor, which is designed for notebook computers, uses multi-state frequency-voltage combinations (Intel SpeedStep technology) [43] to adapt power consumption to operating conditions. The processor state can be changed by the operating system or by executing application programs. AMD PowerNow! [10] is a similar technology that has also been deployed in processors designed for server environments. Future processor architectures are likely to be variants of chip-multi-processor designs, where multiple simple, low power consumption processor cores exist within a single die. In [45], dynamically powering on and off entire processor cores in response to demand is explored in order to reduce power consumption.

3.3.2 Dynamic Power Management for Disk Drives

Initial policies for power management of disk drives were based upon time-out intervals where if no disk activity was detected for a fixed time period, the disk drive was powered off. This concept was extended in [2] where components of the disk drive are powered off in stages. Disk drives typically take several seconds to spin up and powering off frequently can be inconvenient and time consuming to users. In [29] adaptive time out periods where the idle time period before powering off is adaptively varied by adding or subtracting a constant value from the idle time period was explored. In [58] machine learning techniques that utilize multiple predictors where varying weights are assigned to the predictions depending upon the accuracy of predictions in previous time periods are used to adaptively vary the idle time period before powering off the disk drive. A detailed taxonomy and performance evaluation of adaptive methods for the dynamic power management of disk drives can be found in [44]. However, completely powering off a disk drive can result in unacceptable latencies for servers (spin up time can be several seconds). This issue is addressed by a proposed Dynamic Rotations per Minute (DRPM) mechanism [55]. A disk drive

capable of rotating at several speeds consumes less power in the slower rotational speeds. In the DRPM mechanism, the disk drive rational speed is increased or decreased based upon the latency for previous requests. Another technique is to group disk accesses together in order to enhance the predictability of the idle periods and increase the idle time periods [103].

Though disk drives contain massive amounts of data, the set of pages that have non-zero probability of being accessed in the near future can be quite small. This observation is the basis for hybrid disk drives [133] that contain several gigabytes of non-volatile low-power consumption flash memory. Pages with a high probability of being accessed are copied to the flash memory and the mechanical motors and spindle drives are then powered off. Any requests for data are serviced from the flash memory and the disk drive is powered on only if the requested data is not available in the flash memory. This technology is expected to debut in Microsoft Windows Vista as ReadyDrive [133].

Both disk drive activity and end-user network link activity are user-driven and operate over similar time scales. The adaptive policies used in the dynamic power management of disk drives may possibly be used successfully within the context of end-user network links. Note that powering on or changing the data rate of an Ethernet network link is possibly a magnitude faster than powering on a disk drive.

3.3.3 Dynamic Power Management for Server Clusters

Due to economies of scale and ease of administration, the large numbers of servers and high-bandwidth communication links required by modern e-commerce enterprises are densely packed together in locations generally referred to as data centers or server farms. Each data center hosts thousands of servers with a total power consumption measured in Megawatts [96].

For redundancy in case of node failure and for greater throughput, groups of servers in data centers are organized as clusters. Except at peak load times, the load on the cluster is likely to be significantly less than its overall capacity. In contrast to the two previous examples, rather than applying dynamic power management to components within the server, the focus in server clusters is on powering on and off individual servers based upon the total load upon the cluster. A framework for managing server cluster while optimizing energy consumption is presented in [21] and [22]. Energy savings of over 30% are

claimed for typical web workloads. Algorithms for load balancing in server clusters in order to maximize energy savings are explored in [104].

3.4 Regulatory Directions

The energy crisis of the late 1970's highlighted the need for reducing energy consumption through energy efficiency and conservation, and led to the development of government standards and mandates regarding energy efficiency in automobiles, buildings, and appliances in most industrialized nations. Currently, there are existing regulatory standards regarding the energy efficiency of electrical and electronic devices. These increasingly stringent standards are a factor driving the adoption of dynamic power management mechanisms and policies as manufacturers attempt to provide increased performance and features at lower energy cost. Regulatory efforts in the U.S. and the European Union (EU) are briefly described in the following sections.

3.4.1 EPA Energy Star

The use of power management in desktop PCs was promoted by the Energy Star [32] program initiated by the U.S. Environmental Protection Agency (EPA) in 1992. This program's first specifications were issued for PC system units and monitors. Today, Energy Star sets energy efficiency targets for numerous device classes. This is a voluntary program that sets targets for energy efficiency, and devices that meet these targets are issued an Energy Star label. In most cases, the U.S. Government must purchase equipment certified to be energy-efficient, and the Energy Star label enables consumers to identify energy-efficient devices when making purchases. Since its inception, this program has expanded to include office equipment, lighting, heating and cooling equipment, home appliances and electronics, and numerous other categories. In order to qualify for the Energy Star label, computers shipped after July 1, 2000 must enter a sleep state after 30 minutes of inactivity, in which the power consumption is less than 15% of the power supply's maximum rating (Guideline "B" [32]), while still being capable of responding to network wake events.

The EPA Energy Star Program Requirements for Computers: Draft 3 of Version 4.0 that is presently being developed states that, "All computers shall reduce their network link speeds during times of low data traffic levels in accordance with any industry standards that provides for quick transitions among link rates". Draft 2 further noted that "With such capabilities in place, reduced link rates are expected to be heavily used on Tier 2 computers, and have the potential for significant savings" [108]. ALR for Ethernet described in Chapters 5 and 6 of this dissertation is well suited for providing these capabilities.

3.4.2 European Regulatory Efforts

Similar programs to the Energy Star [32] program exist in most industrialized nations. With respect to desktop PCs and other office equipment, the European regulatory standards are identical to the U.S. standards [34]. The European Commission's Joint Research Council oversees a separate effort, named the EU Stand-by Initiative [35], which focuses on reducing the stand-by power consumption of end-use electrical devices. It is claimed that stand-by power consumption is an estimated 10% of the electricity use in homes and offices within the EU. Presently, the EU Stand-by Initiative has resulted in Codes of Conduct for regulating the power consumption of digital TV receivers, broadband communication equipment on links to homes and offices, and external power supplies. Power consumption targets for broadband communications equipment have driven the development of ADSL2+ [78] and VDSL [79] with multiple power modes, which are expected to be the next generation last-mile-to-home communications technology within the EU [36].

The regulatory efforts undertaken by the EU Stand-by Initiative are of special significance when considering the ALR for Ethernet described in this dissertation. Broadband and Ethernet links offer power savings of the same magnitude and Ethernet LAN links are far more numerous. Similar regulatory efforts for Ethernet LAN links could possibly yield much greater energy savings.

Chapter 4: Power Management in Networks

Communications networks can be differentiated by the transmission media used to propagate signals: wired or wireless. Devices connected to a wired network are usually mains powered, while devices connected to wireless networks are generally mobile, and therefore, battery-powered. The application of dynamic power management to wired and wireless networks is described in this chapter.

Dynamic power management in wireless networks and devices is an area of very active research interest with a tremendous body of existing work. Contributions and trends in this area are described in Section 4.1. Dynamic power management of wired networks and reducing the induced power consumption of devices connected to such networks has not been an active research area until quite recently [54]. The major contributions and trends in this research area are described in detail in Section 4.2.

4.1 Power Management in Wireless Networks

Wireless networks are usually used for communicating with mobile network hosts such as cell phones, notebook PCs, and in wireless sensor networks. These devices are mostly battery-powered and batteries are recharged or replaced at infrequent intervals. By minimizing the power consumption of these devices the battery lifetime can be extended and/or the size of the battery required can be reduced (thereby reducing the weight and size of the device). Sensor devices are typically a few cubic centimeters in size and are expected to report back sensor data for weeks and months once deployed. Usually their power supplies cannot be renewed once deployed and extending battery lifetime is of paramount importance. The wireless network interface is a significant power consumer [121] in all these devices. Thus reducing the power consumption is a very active area of research.

The lack of legacy protocols and products in this new research area has enabled novel “clean slate” solutions. Wireless networks are either self-contained or interface with the wider Internet through a gateway that translates between the wireless network and the wired network. The need for inter-operability

with existing protocols and products does not exist. However, this factor also limits the relevance and applicability of these solutions to wired networks. For dynamic power management solutions for wired networks, interoperability with existing transport layer protocols, network layer protocols, and routing protocols is essential. Widespread adoption of new solutions cannot be expected otherwise. Another factor to consider is that wireless networks mostly operate at significantly low data rates compared to wired networks. Correspondingly, the need for buffer capacity can be lower and acceptable latencies (especially for sensor networks) can be higher.

Proposed power management solutions in the area of wireless networks include:

1. Novel energy-efficient transport layer protocols.
2. Energy-efficient network layer and routing protocols that enables nodes to power off for extended time periods.
3. Medium access control schemes designed to permit network interfaces to power off when not transmitting or receiving data.

Advances in these areas are briefly described in the following sub-sections.

4.1.1 Reducing the Energy Cost of TCP/IP

The TCP/IP protocol suite was originally developed for wired networks, but since the advent and widespread use of wireless networks, it has been extended for use with wireless networks as well. TCP/IP is designed to operate in an environment where the main cause of packet loss is network link congestion and buffer overflows. However, packet loss in wireless networks is also caused by the high bit-error-rates of wireless links. TCP responds to losses by retransmitting lost packets and by reducing the transmission rate leading to greater energy cost and delay. Research efforts have focused on adapting existing TCP versions and on developing alternative transport layer protocols specifically designed for wireless networks.

Wang and Singh experimentally measure the computational cost of using TCP in mobile devices in [130]. The components are categorized as: 1) user-to-kernel copy cost; 2) kernel-to-NIC copy cost; and 3) TCP processing cost. It is shown that the kernel-to-NIC copying cost is the largest component (at 70%) of

the total computational cost. Furthermore, for a notebook PC using 802.11 [64], the computational cost is approximately 70% of the total cost of sending or receiving data over TCP/IP, while the radio receiving/transmitting cost is approximately 30%. For a Compaq iPAQ PDA, the ratios are reversed with the computational cost being approximately 30% of the total cost and the radio receiving/transmitting cost being approximately 70%. The authors propose that TCP packets be buffered in the NIC so that when a packet retransmission is required, it does not need to be copied from the kernel. Experimental studies to determine which TCP variant has lower energy consumption over a single link and multiple links are described in [4] and [116], respectively. In [4], it is shown that turning off selective acknowledgement (SACK) and increasing the maximum transmission unit (MTU) to 2296 bytes (the maximum allowed by the 802.11 [64] data link layer protocol) results in an increase of energy efficiency of almost 25%. Contradicting the results in [4], it is shown in [116] that over multiple hops TCP SACK gives better results than either TCP Reno or TCP NewReno. The energy efficiency of TCP in a wireless environment is estimated through analytical means in [138]. The wireless channel is modeled using a two-state Markov model. The authors conclude that the TCP congestion control mechanism works as intended when the wireless channel has long error bursts.

New variants of TCP and novel transport layer protocols more suitable for wireless networks have been proposed. In [125], it is shown that the error and congestion control algorithms of the existing TCP variants perform similarly in wireless networks where energy efficiency and throughput are concerned. It is also shown that TCP NewReno is more efficient where the errors are of short duration, while TCP Tahoe is more efficient where the errors are of long duration. In [126], the same authors propose a new TCP variant named TCP Probing, which probes the network after encountering an error in order to ascertain channel status. This new TCP variant is shown to be capable of greater throughput with greater energy efficiency than existing TCP variants. Some environments such as wireless sensor networks do not require real-time transmission of data, and in [127], a new transport layer protocol suitable for energy constrained environments is proposed and described. The Wave and Wait Protocol (WWP) utilizes probing packets to identify channel conditions and reduces or increases the number of packets transmitted in a burst based upon channel conditions. It is shown that the WWP performs significantly better with several times more throughput than TCP in noisy environments. In [11], a power saving network architecture is proposed in

which a connectionless transport layer protocol is used for transmitting and receiving from the wireless node to the access point, while normal TCP is used from the access point to the destination node. The access point estimates the time necessary to transmit the data received from the wireless node to the destination node and communicates this time to the wireless node. The wireless node then powers off the network interface for the duration of that time period. The proposed protocol is based upon the concept of Indirect TCP described in [13].

4.1.2 Energy Aware Routing in Ad Hoc/Sensor Networks

In wired networks, routing protocols are designed to select routes with the least delay. However, with wireless sensor or ad hoc networks, there are no dedicated routers, but rather data is routed through participating nodes. Nodes are energy constrained, and in some cases, have short transmission ranges. In such situations, maintaining network connectivity to all nodes and reducing the power consumed by forwarding data for other nodes are of greater importance than minimizing packet delay. New "clean slate" energy efficient routing protocols have been developed for wireless sensor and ad hoc networks in order to satisfy these conditions. A detailed survey of wireless sensor networks is given in [6], and a survey of routing protocols in wireless sensor networks is given in [8] and [5]. Several significant protocols are described briefly here.

In one approach, it is proposed that the data sink transmit queries with attributes such as the required data type and geographic area of interest. The sensor nodes would transmit the data messages in response to the query, and messages could then be aggregated together for greater efficiency. The SPIN protocol [57] and the Directed Diffusion protocol [65] are examples of this approach. Another approach is to cluster sensor nodes that are geographically co-located and select a single node (named the cluster head) to transmit the data gathered by the nodes in the cluster. The cluster head would relay messages from outside the cluster towards the data sink. Once the energy level of the cluster head reaches a certain threshold the role is assumed by another node in the cluster. LEACH [56] was the first protocol to use cluster-based routing, and PEGASIS [92] is an improved version of LEACH. Other approaches used for routing in

wireless sensor networks include geographic location-based routing [110] and selecting routes based upon energy levels as a metric [20].

4.1.3 Reducing the Power Consumption of the Network Interface

A wireless interface can be transmitting data, receiving data, or in standby. Measurements have shown that a PCMCIA card wireless network interface consumes 1.82 W when transmitting data, 1.80 W when receiving data, and 0.18 W when in standby state [82]. Power saving methods and policies attempt to maximize the time the wireless interface spends in the standby state, and numerous examples have been proposed over time. Scheduling the transmission time for each node and powering off the interface when not transmitting or receiving is proposed in [117] and [118]. The widely used 802.11 wireless LAN protocol [64] uses a beaconing method, where the mobile node informs the access point that it is powering off the network interface. Subsequently, if the access point receives any packets addressed to that node, the access point will periodically transmit a beacon packet. The wireless interface powers on periodically and listens for the beacon, and if necessary, will retrieve any buffered packets from the access point. One weakness with this method is that packets are delayed by at most the inter-beacon time. The authors of [89] propose an improved method that provides performance guarantees in order to eliminate this.

A novel solution to minimizing the powered on time is proposed in [52]. The authors propose that two transmission channels be employed, one for high-bandwidth data and the other for low-bandwidth control signals. The mobile node's wireless interface would have a normally powered transceiver for the high-bandwidth channel and a very low powered receiver for the low-bandwidth control signal channel. If the wireless interface in the mobile node is powered off, the access point would transmit a "wake-up signal" in the control channel before transmitting the data in the data channel. This idea has significant merit and Shih et al. [114] have implemented this technique in a Compaq iPAQ PDA which demonstrated an increase in battery lifetime of over 115%. A detailed survey of energy efficient medium access protocols for wireless networks is available in [82].

4.2 Power Management in Wired Networks

The lack of legacy protocols and products has motivated the design and development of novel solutions for reducing power consumption in wireless networks. In contrast, the need for interoperability and compatibility with existing protocols and products is of vital importance in wired networks.

Network induced power consumption was first noticed in centrally-controlled corporate networks. In order to carry out automated administrative tasks during the nighttime, network connected desktop PCs had to be kept powered on continuously. Consequently, the first attempts were directed at reducing the induced power consumption of the network hosts in corporate networks. As the Internet grew to encompass millions of hosts in homes and offices, the power consumption of network intermediate nodes reached many TWh/yr and research interests were focused on reducing the direct power consumption of wired networks as well. This section describes existing work on reducing the direct and induced power consumption of network hosts, LAN switches, Ethernet network links, and NICs.

4.2.1 Wake-On-LAN for Wake-Up of "Sleeping" Hosts

It was observed that in order to update, patch or perform routine backups and maintenance on network connected PCs in corporate networks during the nighttime, either it was necessary to leave the PC powered on at all times or a technician had to physically power on each PC. In the early 1990's, IBM and Intel proposed a novel technology named Wake-On-LAN (WOL) (named Magic Packet by AMD) [9]. This new

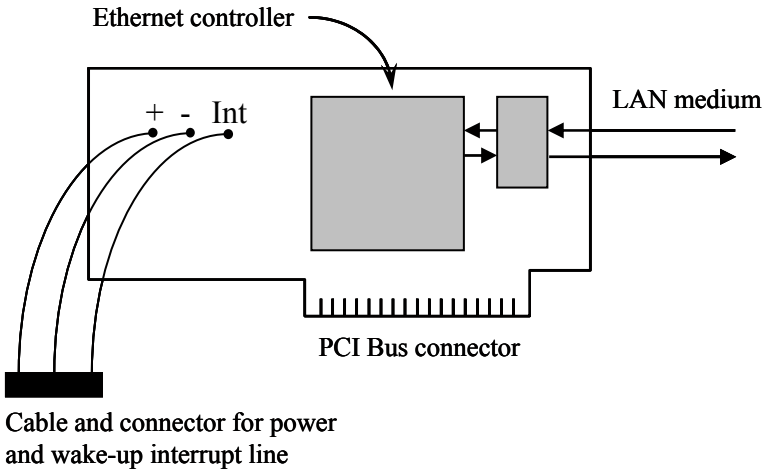


Figure 4.1. WOL-Capable Ethernet NIC

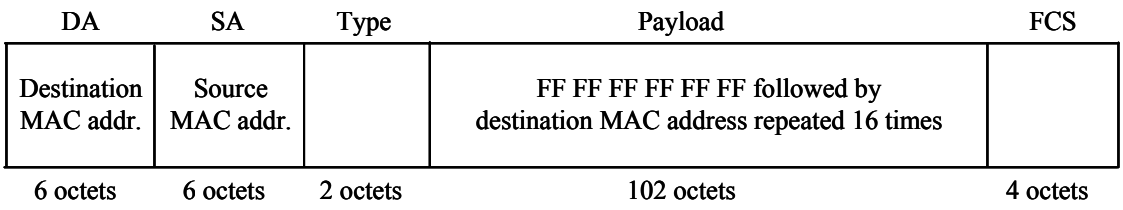


Figure 4.2. Wake-On-LAN Packet Format

technology woke up a PC that was powered off to an intermediate state between powered on and powered off by sending a special WOL packet through the network. To accomplish this feat, motherboards were designed to keep the network interface controller powered-on although the PC was nominally powered off. A host that needs to access a WOL-capable host would first send a WOL packet. The always powered on NIC would receive the packet and recognize from the special format of the packet that it is a request to power on the host. The NIC would trigger a wake-up signal to the PC causing it to power on and be accessible over the network. Figure 4.1 illustrates a WOL-capable Ethernet NIC.

A WOL packet is identified by the destination host's medium access control (MAC) address being repeated 16 times in the Ethernet packet payload area, as shown in Figure 4.2. Reliance upon the destination host's MAC address limits the scalability of WOL. MAC addresses are data link layer addresses, and as such, are not propagated beyond the local broadcast domain. Since there is no mechanism similar to DNS for resolving MAC addresses, a network host that is outside the local area network (LAN) of the targeted host has no means of identifying the targeted host's MAC address and thus cannot send a WOL packet.

Advanced Configuration and Programming Interface (ACPI) was reviewed in Chapter 3. Under the ACPI power management scheme, the network device class can wake-up the system upon receipt of a wake packet from the network. Wake packets include the original WOL packet, any packet with the host's IP address in the destination IP address field, and any packet that matches a predefined set of bit-patterns. By responding to the IP address, which is known outside the local network of the targeted host, it was expected that targeted hosts could be reached from outside the local network.

WOL capable NICs cannot respond to routine network messages while the host is in a low power state. Also, it is implicitly assumed that a wake packet will be repeated after a time interval sufficient for the network host to power on. This might very well not be the case though. The address resolution protocol

(ARP) is used in LAN environments to identify the MAC address of the host with a particular IP address. To find the MAC address, an ARP query broadcast message is transmitted with the IP address and the host with that particular IP address would send an ARP reply to the host broadcasting the query. Lack of responses to ARP query messages can result in the host being unreachable (since the MAC address that corresponds to the IP address is not known) and waking up the host for each and every packet addressed to it can result in reduced powered off time.

4.2.2 Power Management of LAN Switches

One of the first works on dynamic power management of wired computer networks is [54] by Gupta and Singh. Wireless networks broadcast data in all directions, and it was shown that it takes more energy (in bits per Joule) to transmit data through wired networks than wireless networks. Gupta and Singh propose two solutions to reduce the power consumption of wired networks. First, they proposed to power off network interfaces in LAN switches during packet inter-arrival times. It is shown that by adding extra memory buffers at the interfaces, packet loss can be minimized. Second, they proposed that routing protocols be modified so that all traffic on parallel routes are aggregated into one route during low-traffic times in order to allow idle network links be powered off to save energy. In [53], the first idea is expanded further, and it is shown that significant percentages of idle time exist by analyzing traces of realistic links. The analysis in [53] contains an error: in the simulation model, the value of the current inter-packet idle period is used instead of the value of the previous inter-packet idle period in predicting the value of the current inter-packet idle period This gives results that are near-optimal. The proposed method, when correctly calculated, can result in energy savings of up to 50% of the interface power consumption.

The feasibility of powering off network interfaces during inter-packet idle periods is contingent upon the development of a mechanism to power off and power on the components in the NIC within sub-millisecond timescales.

4.2.3 Low Power Modes in ADSL2 and ADSL2+

Asynchronous Digital Subscriber Line (ADSL) [76] is a last-mile-to-home communication technology that is used to provide broadband data connections from switching stations to residences. ADSL competes with metro-Ethernet and cable modems in this market. Newer versions of ADSL, ADSL2 [77] and ADSL2+ [78], support several power modes: L0 (powered on and transmitting or receiving at maximum possible data rate), L1 (low power consumption mode with less than the maximum data rate), and L2 (powered off) [42]. These power consumption modes are standardized in ITU recommendations G.992.3 [77] and G.992.5 [78]. The primary goal is to reduce the power consumption of each link and thus reduce the aggregate heat dissipation at local switching stations where hundreds or even thousands of ADSL2 lines are terminated. An ADSL2+ link is expected to consume 3 W in L0 mode, 0.5 W in L1 mode, and 0.3 W in L2 mode.

Energy savings are achieved by reducing the transmission power. Policies that determine when a link should transition to another power mode are apparently not standardized and are left to the manufacturers of ADSL2 equipment. Policies considered include idle time length based triggers [42] and counting the number of cells transmitted and received [128].

4.2.4 Reducing Ethernet Data Rates When in Sleep States

In order to maintain inter-operability with previous-generation products, products conforming to newer Ethernet over UTP standards (100BASE-TX, 1000BASE-T) are capable of operating at lower (previous-generation) link data rates. A mechanism named “Auto-Negotiation” [61, Clause 28] is used to select the highest compatible data rate at link initialization and the Ethernet link is operated at this data rate. Due to the necessity of compatibility with 10BASE-T network interfaces, during the Auto-Negotiation process a burst of pulses named Fast Link Pulse (FLP) burst is transmitted every 16 ms, which is the interval between 10BASE-T Normal Link Pulses. Each FLP burst consists of 33 pulse positions and contains 17 clock pulses and 16 data pulses. The data pulses encode a 16-bit Link Code Word (LCW) containing NIC capability information. Clock and data pulses within a FLP burst are illustrated in Figure 4.3. Each transmitted Link

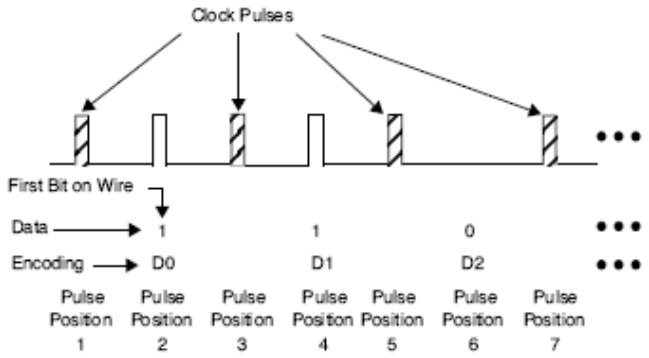


Figure 4.3. Clock and Data Pulses in FLP Burst (This is Figure 28-4 from [61])

Code Word is acknowledged by the link partner after three consecutive consistent FLP bursts have been received.

A sequence of FLP bursts is necessary for transmitting link capability information and for selecting the link data rate. In the case of 1000BASE-T compatible NICs, a Base page LCW, a Message page LCW and two Unformatted Next Page LCWs are required [61, Clause 40.5]. Transmitting and acknowledging this sequence can take a minimum of 256 ms. Data cannot be transmitted while the Auto-Negotiation process is in progress [61, Clause 28, Figure 28-3]. Therefore, using Auto-Negotiation for changing the link data rate while the link is in use is not feasible and may even result in packet loss in switches and hubs due to buffer capacity being exhausted. However, Auto-Negotiation is used by some Ethernet NICs to reduce the Ethernet link data rate when the network host (desktop PC) enters a low power “sleep” state [66]. When the network host transits back to powered on state the Ethernet link data rate is set back to the highest possible rate using Auto-Negotiation. The possibility of packet loss is immaterial in this situation since the network host is in a low power sleep state. Auto-Negotiation is also used in some notebook PCs to reduce the link data rate when transitioning from mains power to battery power.

Chapter 5: Reducing Direct Energy Use – ALR Part I

Measurements of Ethernet NIC and switch power consumption show that 1 Gb/s Ethernet links consume about 4 W more than 100 Mb/s links and that for 10 Gb/s links the increase in power consumption may be in the tens of Watts (this is described in Section 2.1). It was also shown that idle and fully utilized Ethernet links consume about the same amount of power, that is, Ethernet power consumption is independent of link utilization. Furthermore, it was shown that utilization of Ethernet links to desktop PCs is very low, typically less than 5% of link data rate. Thus, there is opportunity for significant energy savings by operating links at a lower data rate during low utilization periods.

In this chapter it is proposed to adaptively vary the link rate to match utilization (offered load) in order to make Ethernet link power consumption proportional to utilization. Adaptive Link Rate (ALR) is a method for automatically switching the data rate of an Ethernet link to match link utilization and is designed to use existing data rates (10 Mb/s, 100 Mb/s, 1 Gb/s, and 10 Gb/s – this work is focused on NICs capable of 10 Mb/s, 100 Mb/s, and 1 Gbs/ data rates). The savings in direct energy consumption is achieved by operation at a lower link data rate at the expense of an increase in packet delay. Key challenges in realizing ALR are defining a mechanism for quickly switching link data rates and creating a policy to change the link data rate without adversely affecting packet delay while maximizing energy savings. The ALR *mechanism* determines *how* link data rates are switched while the ALR *policy* determines *when* link data rates are switched. A good policy should maximize the time spent in a low data rate (energy savings) while minimizing the increased packet delay. However, a good policy should also favor low packet delay over energy savings to minimize the effect of ALR on applications.

In this chapter, Section 5.1 presents a system design for ALR. Section 5.2 describes the ALR MAC frame handshake mechanism as well as the ALR dual-threshold policy. Section 5.3 develops a Markov model for the dual-threshold policy. In Section 5.4, numerical results from the Markov model and the development and verification of a simulation model for the ALR dual-threshold policy are described.

Results from a performance evaluation of representative link configurations are also provided in Section 5.4 with both trace and Poisson traffic as input. Finally, section 5.5 models and analyzes link rate oscillations caused by the ALR dual-threshold policy.

5.1 ALR Mechanisms and Policies

The ALR MAC frame handshake mechanism is described in Section 5.1.1 and the ALR dual-threshold policy is described in Section 5.1.2.

5.1.1 ALR MAC Frame Handshake Mechanism

A fast mechanism for initiating and agreeing upon a link data rate change is necessary in order to minimize packet delay and to minimize the possibility of packet loss due to buffer overflow. Either end of a link, (i.e., the NIC in a desktop PC and the switch port in a LAN switch) must be able to initiate a request to change the rate. The desktop PC could be running an application that is generating data at greater than the present link data rate for transmission to a remote host and the switch port could receive a burst of data from a remote host at a rate greater than the present link data rate. Therefore, both ends of the link must be ALR capable.

It is necessary at link establishment to negotiate capabilities including support of ALR at both ends and the possible data rates that the link partner NICs have in common. Existing 802.3 Auto-Negotiation [61, Clause 28] negotiates the data rates and other capabilities the link partners have in common at link initialization. This can be extended for negotiation of ALR capability as well (e.g., using Unformatted Link Code Word pages used for exchanging additional data). Auto-Negotiation could also be used to switch data rates during link operation, but as detailed in Section 4.2.4, the Link Code Word exchange alone requires a

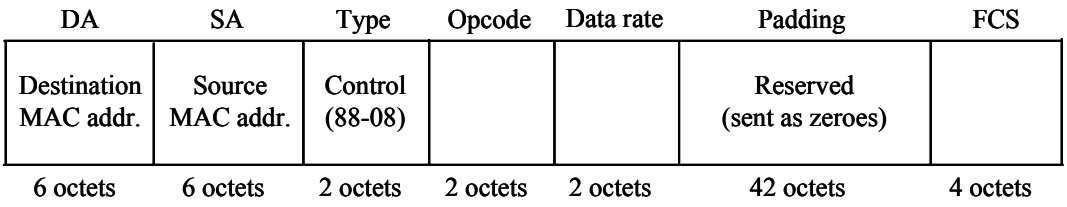


Figure 5.1. ALR MAC Frame

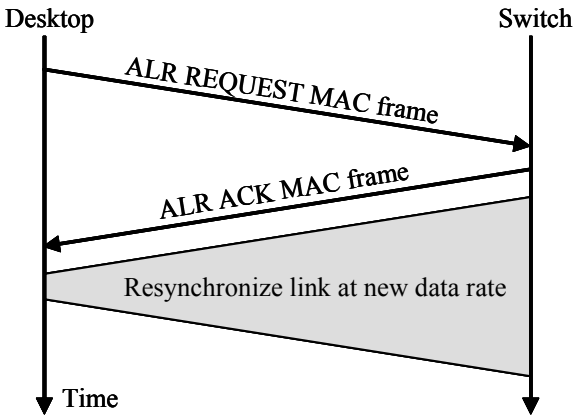


Figure 5.2. ALR MAC Frame Handshake

minimum of 256 ms for a 1000BASE-T compatible implementation. A faster handshaking could be implemented using Ethernet MAC frames. Figure 5.1 illustrates an ALR MAC frame based upon the existing 802.3 MAC control frame [61, Clause 31]. The *Opcode* field identifies the ALR message and the *Data rate* field identifies the requested data rate. A two-way handshake could be implemented as:

1. The end of the link that determines a need to increase or decrease its data rate requests a new link data rate change using an ALR REQUEST MAC frame. The request could be "goto low" or "goto high" data rate.
2. The receiving link partner acknowledges the data rate change request with either an ALR ACK (agrees to change the data rate) or ALR NACK (does not agree) MAC frame.

Following an agreement to switch the link data rate, the link is resynchronized at the new data rate. Figure 5.2 shows a MAC frame handshake followed by a link resynchronization. For operating at 10 Gb/s, the noise environment and required signal strength are determined at link establishment and is expected to take several seconds. More work needs to be done to understand if link retraining needs to be done every time a link is resynchronized, or only at link establishment. However, this is a physical layer problem and is outside the scope of this dissertation.

An ALR ACK response would trigger a link rate switch (rate transition) and link resynchronization. Based on 10,000 clock cycles needed for link resynchronization, the total time for handshake plus resynchronization can be less than 100 μs for 1 Gb/s Ethernet. However, it is quite likely that link resynchronization can be achieved in much less than 10,000 clock cycles. The time taken to switch data

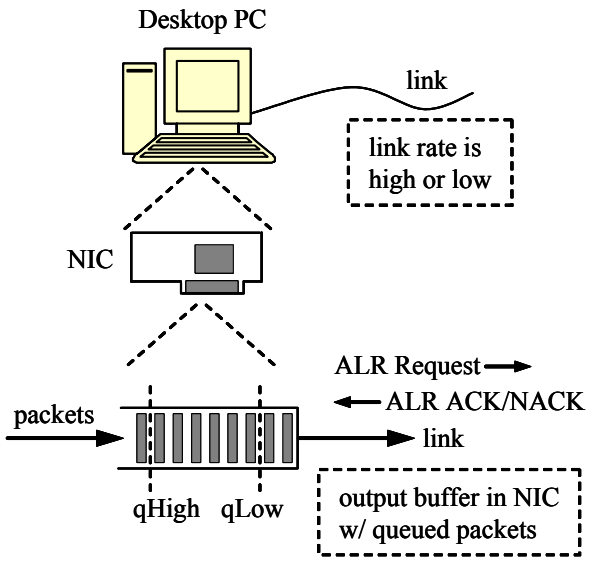


Figure 5.3. Output Buffer with Thresholds

rates is of critical importance to the performance of ALR. In most of the ALR simulation experiments described in this dissertation, rate switching time (T_{switch}) is conservatively assumed to be 1 ms.

5.1.2 ALR Dual-Threshold Policy

The simplest ALR policy is based on output buffer queue length threshold crossing. Figure 5.3 shows a NIC output buffer with high and low thresholds. The NIC is contained within a desktop PC, either as an expansion card (shown) or built into the motherboard. Note that the NIC could also be in a LAN switch port. Two thresholds are used to introduce hysteresis into the system and minimize oscillation between rates. A detailed system design of the dual-threshold policy as executed in the NIC is shown in the finite state machine (FSM) given in Figure 5.4 where the output buffer queue thresholds are denoted as qLow

Table 5.1. FSM Timers, Parameters and Variables for Dual-Threshold

Name	Description
tRetry	Timer for handshake retry (ACK, NACK lost)
qLow	Output buffer queue length low threshold in bytes
qHigh	Output buffer queue length high threshold in bytes
qLen	Present output buffer queue length in bytes
reqLow	Request low rate flag. A NIC can request a low rate only if this is set to true and is designed to prevent one link partner repeatedly requesting a low rate.

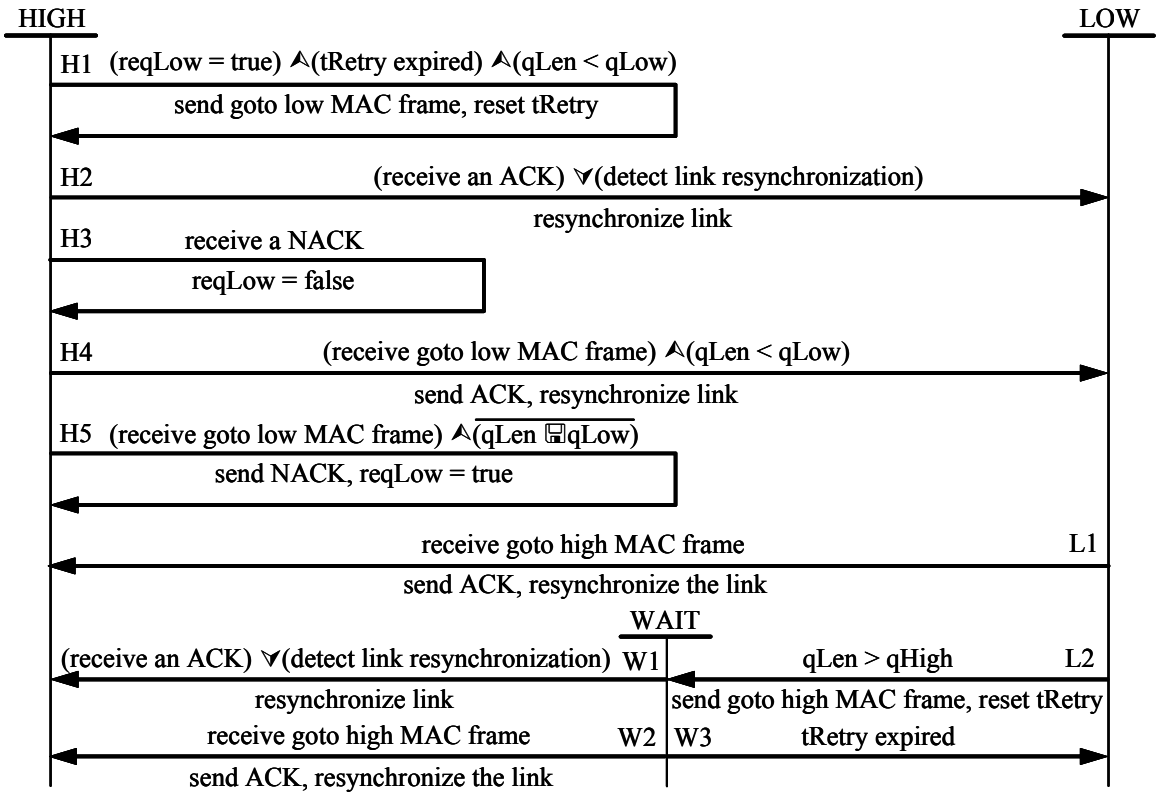


Figure 5.4. FSM for ALR Dual-Threshold Policy

(low threshold) and qHigh (high threshold). The FSM timers, parameters, and variables are described in Table 5.1. Timers are expired when not timing and they begin to count down to their expired condition when reset. It is assumed that link resynchronization does not fail once initiated by both link partners. It is assumed that one side starts (e.g., at link initialization) with reqLow as true and the other side with it as false, and that both sides always start in the HIGH state. A detailed explanation for each of the FSM states and transitions follows:

State HIGH: NIC is operating in high link data rate.

Transition H1: The output buffer queue length (qLen) is less than the low queue threshold (qLow). The timer tRetry has expired and it is the NIC's turn to request a low data rate (reqLow = true). A goto low ALR REQUEST frame is sent to the link partner NIC. The timer tRetry is reset.

Transition H2: An ALR ACK frame is received (in response to a previously sent goto low ALR REQUEST frame) or link synchronization is detected. The NIC resynchronizes the link to the low data rate.

Transition H3: An ALR NACK (negative acknowledgement) frame is received (in response to a previously sent goto low ALR REQUEST frame). The variable reqLow is set to false indicating that the link partner NIC is to request a low rate next.

Transition H4: A goto low ALR REQUEST frame is received and qLen is less than qLow. An ALR ACK frame is transmitted and the link is resynchronized at the low rate.

Transition H5: A goto low ALR REQUEST frame is received and qLen is not less than qLow. An ALR NACK frame is transmitted and the variable reqLow is set to true.

State LOW: NIC is operating in low link data rate.

Transition L1: A goto high ALR REQUEST frame is received. The only possible action is to transmit an ALR ACK frame and resynchronize the link at the high data rate.

Transition L2: Output buffer queue length (qLen) exceeds the high threshold qHigh. A goto high ALR REQUEST frame is transmitted and the NIC transits to the WAIT state.

State WAIT: A goto high ALR REQUEST frame has been transmitted. The NIC is waiting to either receive and ACK frame from link partner NIC or for the tRetry timer to expire. NIC is operating in low link data rate.

Transition W1: An ALR ACK frame is received or a link resynchronization is detected. The NIC resynchronizes the link at the high rate.

Transition W2: A goto high ALR REQUEST frame is received (i.e. the link partner NIC has independently decided that high data rate is required). An ALR ACK frame is transmitted and the link is resynchronized at the high rate.

Transition W3: No response is received from the link partner NIC during time tRetry, and therefore transitions back to the LOW state and resends goto high ALR REQUEST frame if conditions are still valid.

```
if (qLen is greater than qHigh threshold) then
  handshake for a high link data rate
```

(a) Process for state LOW

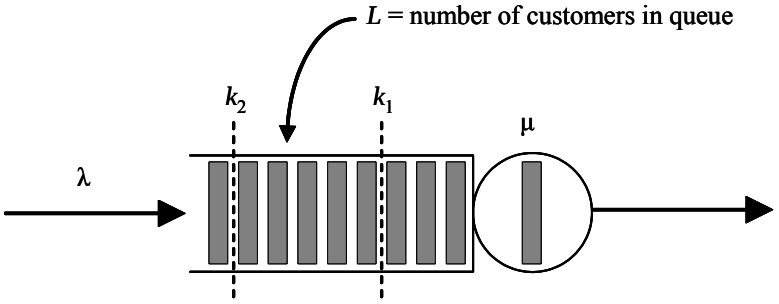
```
if (qLen is less than qLow threshold) then
  handshake for a low link data rate
```

(b) Process for state HIGH

Figure 5.5. Pseudocode Description of the Dual-Threshold Policy

The internal variable reqLow in the FSM is used to prevent one side from repeatedly requesting to reduce the data rate when the other side cannot agree with the request (that is, if one side has a high utilization and the other side is idle). If a NACK to a goto low REQEST is lost and both sides simultaneously request a low data rate and also simultaneously reply with a NACK to each other, there exists a possibility that both sides could get stuck in the HIGH state with reqLow as false. This condition is not shown in the FSM but would be resolved when the link is next initialized.

The dual-threshold policy can also be described as two processes that execute at each end of the link. Figure 5.5 shows a pseudocode description of the dual-threshold policy where (a) shows the process that



Rate change during service:

$$\mu = \begin{cases} \mu_1 \text{ if } (\mu = \mu_2) \text{ and } (L < k_1) \\ \mu_2 \text{ if } (\mu = \mu_1) \text{ and } (L > k_2) \end{cases}$$

Rate change at service completion:

$$\mu = \begin{cases} \mu_1 \text{ if } (\mu = \mu_2) \text{ and } (L < k_1) \text{ and (service center empty)} \\ \mu_2 \text{ if } (\mu = \mu_1) \text{ and } (L > k_2) \text{ and (service center empty)} \end{cases}$$

Figure 5.6 . Single Server Queue with State Dependent Service Rate

corresponds to the LOW state in the FSM and (b) the process that corresponds to the HIGH state. Thus (a) contains FSM transitions L1, L2, W1, W2, and W3 and (b) contains transitions H1, H2, H3, H4, and H5. The pseudocode description omits details on the handshake procedure. The process in (a) is triggered on receiving a frame and the process in (b) is triggered when sending a frame.

5.2 Markov Model for the ALR Dual-Threshold Policy

Assuming Poisson arrivals and exponential service times, the ALR dual-threshold policy can be modeled as a state-dependent service-rate, single server queue where rate transition cannot occur during a service period (that is, service of the customer in the service center must complete before the service rate can change). A single server, state-dependent service rate queue with two service rates, μ_1 and μ_2, is shown in Figure 5.6 where $\mu_1 < \mu_2$. In this section, single and dual-threshold systems with rate transition possible either during a service period or only on service period completion are modeled. In the case of an Ethernet link, rate transition clearly cannot occur during a packet transmission (a service period); it can only occur after the completion of packet transmission.

Packets arrive at a rate λ and are serviced at a low service rate μ_1 or high service rate μ ($\mu_1 < \mu$) with respective utilizations as $\rho_1 = \lambda/\mu_1$ and $\rho = \lambda/\mu$ where $\rho < 1$ for stability. In the single-threshold policy, the output buffer occupancy level equaling or exceeding threshold value k causes the service rate to switch from μ_1 to μ. An output buffer queue length dropping below *k* causes the service rate to switch from μ to μ_1. In the dual-threshold policy, two buffer occupancy thresholds, k_1 and k_2, are defined where $k_1 < k_2$ Thus k_1 corresponds to qLow and k_2 to qHigh in the FSM of Figure 5.4. An output buffer occupancy level equaling or exceeding threshold value k_2 causes the service rate to switch from μ_1 to μ if the present rate is μ_1. An output buffer occupancy level dropping below threshold value k_1 causes the service rate to switch from μ to μ_1 if the present service rate is μ.

Of interest is the packet delay and time spent in low service rate as a function of threshold values (k_1 and k_2), arrival and services rates (λ, μ_1, and μ), and rate switching time (T_{switch}). Performance models are developed in order to gain insight into the effects of these parameters on packet delay and energy

savings. Energy is saved when the link is in low service rate (low data rate). Expressions are derived for the steady state probability of *n* customers (packets) in the system, P_n, and the mean number of customers in the system, *L*. In the derivations, π_n is used to denote the steady state probability of state *n* and unless explicitly stated, P_n does not equal π_n. Relative time in low rate is the sum of steady state probabilities of states with service rate μ_1.

5.2.1 Continuous Service – Single and Dual-Threshold

The simplest system is that of a single-threshold queue with state-dependent service rate where a service rate transition can occur during a service period. Figure 5.7 is the well known Markov model for this system. For this Markov model and the others, the number of customers in the system is shown below the Markov chain and for certain states, the state identifier differs from the number of customers in the system in that state. For this system, closed-form expressions are readily available [46]:

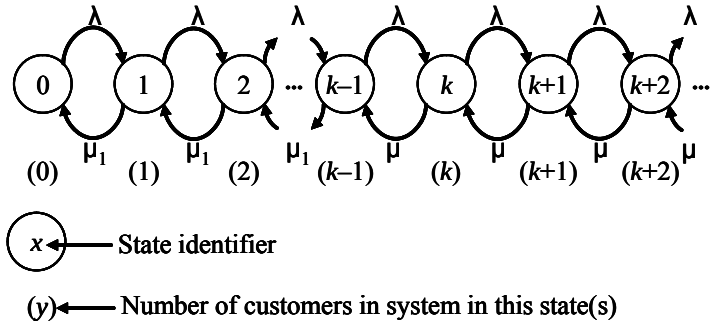


Figure 5.7. Single Threshold, Transition During Service

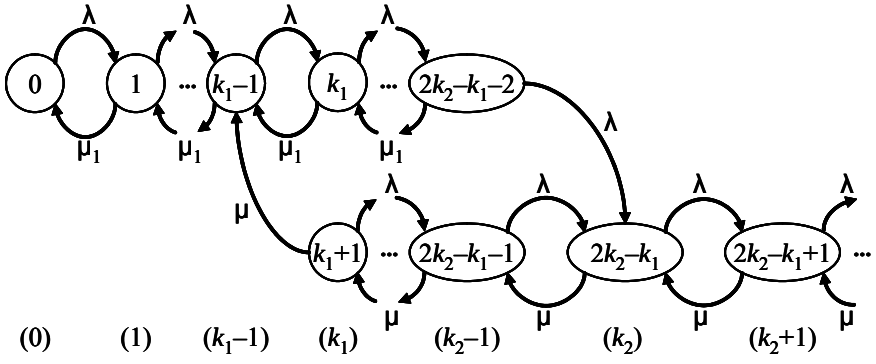


Figure 5.8. Dual-Threshold, Transition During Service

$$P_n = \begin{cases} P_0 \rho_1^n & (0 \le n < k) \\ P_0 \rho_1^{k-1} \rho^{n-k+1} & (n \ge k) \end{cases} \tag{5.1}$$

$$P_0 = \left(\frac{1-\rho_1^k}{1-\rho_1} + \frac{\rho \rho_1^{k-1}}{1-\rho} \right)^{-1} \tag{5.2}$$

$$L = P_0 \left(\frac{\rho_1 \left(1 + (k-1)\rho_1^k - k\rho_1^{k-1}\right)}{(1-\rho_1)^2} + \frac{\rho \rho_1^{k-1} \left(k - (k-1)\rho\right)}{(1-\rho)^2} \right) \tag{5.3}$$

Extending the single-threshold model to two thresholds was first accomplished by Gebhard [41]. The Markov chain for this system is shown in Figure 5.8. The two rows of states for k_1 to $k_2 - 1$ customers in the system model the cases for the system being in rate μ_1 or μ depending on the previous threshold crossing and rate. In states k_1 to $2k_2 - k_1 - 2$ (the upper row of states), the service rate is μ_1, and will only change when the number of customers increases to k_2. In states $k_1 + 1$ to $2k_2 - k_1 - 1$ (the lower row of states), the service rate is μ, and will only change when the number of customers decreases below k_1. The following closed-form expressions are from [41] and are converted to the notation used in this dissertation:

$$P_n = \begin{cases} P_0 \rho_1^n & (0 \le n < k_1) \\ \dfrac{P_0 \rho_1^{k_1-1}(1-\rho_1)}{\left(1-\rho_1^{k_2-k_1+1}\right)} \left(\dfrac{\rho_1^{n-k_1+1} - \rho_1^{k_2-k_1+1}}{1-\rho_1} + \dfrac{\rho_1^{k_2-k_1} \rho \left(1-\rho^{n-k_1+1}\right)}{1-\rho} \right) & (k_1 \le n < k_2) \\ \dfrac{P_0 \rho_1^{k_2-1} \rho^{n-k_2+1} (1-\rho_1)\left(1-\rho^{k_2-k_1+1}\right)}{(1-\rho)\left(1-\rho_1^{k_2-k_1+1}\right)} & (n \ge k_2) \end{cases} \tag{5.4}$$

$$P_0 = \left(\frac{1}{1-\rho_1} - \frac{(1+k_2-k_1)\rho_1^{k_2-1}(\rho_1 - \rho)}{\left(1-\rho_1^{k_2-k_1+1}\right)(1-\rho)} \right)^{-1} \tag{5.5}$$

5.2.2 Discrete Service – Single and Dual-Threshold

To correctly model ALR for Ethernet, rate change can only occur at a service completion (that is, at the completion of sending a packet). Figure 5.9 depicts the Markov chain for this single-threshold rate change at service completion system, which is similar to the Markov chain presented by Chong and Zhao in [23] and solved with the use of transforms. In this dissertation, the Markov chain is solved without the use of

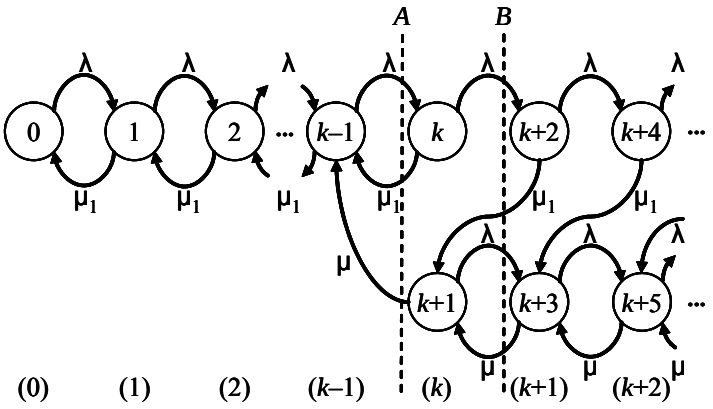


Figure 5.9. Single Threshold, Transition at Completion

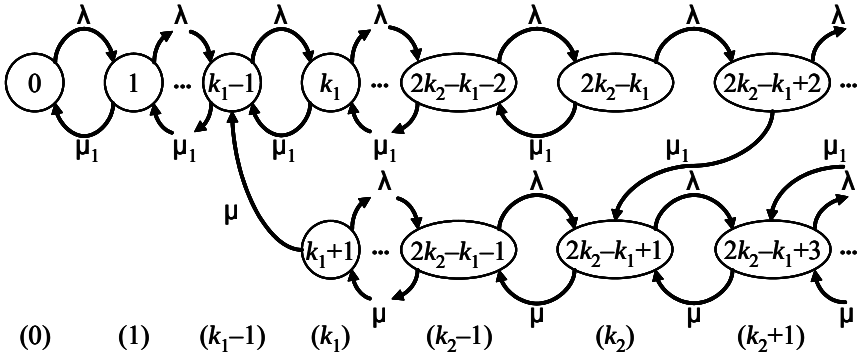


Figure 5.10. Dual-Threshold, Transition at Completion

transforms [122] and this method is then extended to solve the Markov chain with two thresholds (as given in Figure 5.10).

The states k, $k+2$, $k+4$, … $k+2n$ ($n \geq 0$) in Figure 5.9 model the case where arrivals causing a threshold crossing occur during a service period. Only upon completion of the service period during which the number of customers in the system exceeded k does the service rate change from μ_1 to μ, as indicated by the transitions from the upper row of states to the lower row of states. Note that at the end of the service period in state k, though the number of customers in the system has reached k, the number of customers in the system is $k-1$, and the service rate remains unchanged at μ_1. This is modeled in Figure 5.9 with the upper row of states denoting service at rate μ_1 and the lower row of states denoting service at rate μ. The steady state probabilities of the states k, $k+2$, $k+4$, … $k+2n$ ($n \geq 0$) are given by the probability of receiving n arrivals during a service interval. The cumulative probability of receiving an arrival *before* time t is

$$F(t) = \int_0^t \lambda e^{-\lambda y} dy = 1 - e^{-\lambda t} . \tag{5.6}$$

Similarly, for exponentially distributed service times with rate of service of μ_1, the probability of completing service at time x is

$$f(x) = \mu_1 e^{-\mu_1 x} . \tag{5.7}$$

Therefore, the probability of receiving an arrival before service completes and of completing service at time x is given by the expression:

$$\left(\int_0^x \lambda e^{-\lambda y} dy \right) \mu_1 e^{-\mu_1 x} \tag{5.8}$$

Since the time taken to complete a service interval can range between 0 and ∞, the probability of receiving an arrival before the current service is completed is

$$\int_0^\infty \left(\int_0^t \lambda e^{-\lambda y} dy \right) \mu_1 e^{-\mu_1 t} dt = \frac{\lambda}{\lambda + \mu_1} . \tag{5.9}$$

In view of (5.9), given the memoryless property of exponential distributed arrivals, the steady state probabilities for states k, $k+2$, $k+4$, … are related by the following equations:

$$\pi_k = \left(\frac{\lambda}{\lambda + \mu_1} \right) \pi_{k-1} , \tag{5.10}$$

and for $n \geq 1$,

$$\pi_{k+2n} = \left(\frac{\lambda}{\lambda + \mu_1} \right) \pi_{k+2n-2} , \tag{5.11}$$

giving for $n \geq 1$,

$$\pi_{k+2n} = \left(\frac{\lambda}{\lambda + \mu_1} \right)^{n+1} \pi_{k-1} . \tag{5.12}$$

By partitioning the Markov chain into two subsets with no shared states and writing the balance equations for the state transitions between the subsets, it is possible to derive the steady state probabilities for the

number of customers in the system. In Figure 5.9, it can be observed that up to state $k-1$ the model is similar to an M/M/1 queue and therefore

$$P_{k-1} = \pi_{k-1} = P_0 \rho_1^{k-1}. \tag{5.13}$$

Given that the steady state probabilities of the states k, $k+2$, $k+4$, … $k+2n$ ($n \geq 0$) are known, the Markov chain needs to be partitioned such that when the balance equation is written the number of states with unknown steady state probabilities is minimized. Balance equations along partition *A* yields

$$\lambda \pi_{k-1} = \mu_1 \pi_k + \mu \pi_{k+1}, \tag{5.14}$$

where the only unknown is π_{k+1}. Balance equations along partition *B* yields

$$\lambda \pi_k + \lambda \pi_{k+1} = \mu_1 \pi_{k+2} + \mu \pi_{k+3}. \tag{5.15}$$

From Figure 5.9, it can be observed that

$$P_k = \pi_k + \pi_{k+1}. \tag{5.16}$$

By substituting from (5.12) and (5.13), (5.16) can be written as

$$\pi_{k+1} = P_k - P_{k-1}\left(\frac{\lambda}{\lambda + \mu_1}\right). \tag{5.17}$$

Similarly, the following can be written

$$\pi_{k+3} = P_{k+1} - P_{k-1}\left(\frac{\lambda}{\lambda + \mu_1}\right)^2. \tag{5.18}$$

By substituting from (5.12) and (5.17) in (5.14), the following is obtained

$$\lambda P_{k-1} = \mu_1 P_{k-1}\left(\frac{\lambda}{\lambda + \mu_1}\right) + \mu\left(P_k - P_{k-1}\left(\frac{\lambda}{\lambda + \mu_1}\right)\right). \tag{5.19}$$

By substituting from (5.12), (5.17), and (5.18) in (5.15), the following is obtained:

$$\lambda P_k = \mu_1 P_{k-1}\left(\frac{\lambda}{\lambda + \mu_1}\right)^2 + \mu\left(P_{k+1} - P_{k-1}\left(\frac{\lambda}{\lambda + \mu_1}\right)^2\right). \tag{5.20}$$

Finally, from (5.19) and (5.20) the following general equation for steady state probabilities where $n \geq k$ is generated:

$$\lambda P_{n-1} = \mu_1 P_{k-1}\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k+1} + \mu\left(P_n - P_{k-1}\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k+1}\right). \tag{5.21}$$

Equation (5.21) can be refined to yield the following recursive equation for the steady state probability of having n packets in the system where $n \geq k$

$$P_n = \rho P_{n-1} + \left(1-\frac{\rho}{\rho_1}\right)P_{k-1}\left(\frac{\rho_1}{1+\rho_1}\right)^{n-k+1}. \tag{5.22}$$

Equation (5.22) can be iteratively solved and the final closed-form equations for the steady state probabilities and P_0 are:

$$P_n = \begin{cases} P_0\rho_1^n & (0 \leq n < k) \\ \dfrac{P_0\rho_1^{k-1}}{\rho+\rho/\rho_1-1}\left(\rho^{n-k+2} - \left(1-\dfrac{\rho}{\rho_1}\right)\left(\dfrac{\rho_1}{1+\rho_1}\right)^{n-k+1}\right) & (n \geq k) \end{cases} \tag{5.23}$$

$$P_0 = \left(\frac{1-\rho_1^k}{1-\rho_1} + \frac{\rho_1^k}{1-\rho}\right)^{-1} \tag{5.24}$$

Extending the single-threshold with rate transition at service completion model to a dual-threshold model is shown in Figure 5.10. Though more complex than the Markov model for the single-threshold system, similar techniques can be used to derive the steady state probabilities. The complete derivation is provided in Appendix A. The steady state probabilities are given by

$$P_n = \begin{cases} P_0\rho_1^n & (0 \leq n < k_1) \\ \dfrac{P_0{\rho_1}^{k_1-1}(1-1/\rho_1)}{1-1/\rho_1^{k_2-k_1+2}}\left(\dfrac{1-1/\rho_1^{k_2-n+1}}{1-1/\rho_1} + \dfrac{\rho\left(1-\rho^{n-k_1+1}\right)}{1-\rho}\right) & (k_1 \leq n < k_2) \\ \dfrac{P_0{\rho_1}^{k_1-1}}{1-1/\rho_1^{k_2-k_1+2}}\left(\rho^{n-k_2+1}\left(\dfrac{(1-1/\rho_1)(1-\rho^{k_2-k_1})}{1-\rho} - \dfrac{1-1/\rho_1^2}{1-\rho-\rho/\rho_1}\right) + \dfrac{(1-1/\rho_1^2)(1-\rho/\rho_1)}{1-\rho-\rho/\rho_1}\left(\dfrac{\rho_1}{1+\rho_1}\right)^{n-k_2+1}\right) & (n \geq k_2) \end{cases} \tag{5.25}$$

and P_0 is given by

$$P_0 = \left(\frac{1-\rho_1^{k_1}}{1-\rho_1} + \frac{1}{1-\rho_1^{k_2-k_1+2}}\left(\frac{\rho_1^{k_2}-\rho_1^{k_1}}{\rho_1-1} + \frac{\rho_1^{k_2}\left((k_2-k_1)(\rho_1-\rho)+\rho_1^2-1\right)}{\rho-1}\right)\right)^{-1}. \tag{5.26}$$

5.3 Performance Evaluation of the Dual-Threshold Policy

With the Markov models developed in the previous section the performance of ALR can be evaluated for Poisson arrivals, exponential services times, and a rate switching time of zero seconds. A simulation model of a single-server queue with state-dependent service time was developed using the CSIM19 [113] process-oriented discrete event simulation function library. The queue is implemented as a CSIM function and other monitoring and measuring functions are also implemented as CSIM functions. Similarly to the analytical model, the service rate changes depending upon number of customers in the queue. The arrivals to the queue are either generated exponentially distributed or read-in from a network trace file. The simulation model allows performance evaluation for the cases of 1) non-zero rate switching times and 2) actual (traced) Ethernet traffic. Two general assumptions made in the simulation model are:

1. An Ethernet link supports two data rates: 100 Mb/s as the low rate and 1 Gb/s as the high rate.
2. Time spent changing from low rate to high rate is considered (for power use) as time spent in high rate, and time spent switching from high to low rate is considered as time spent in low rate.

The simulation model is illustrated in Figure 5.11. The performance metrics of interest are mean packet delay and percentage of time spent in the (energy saving) low data rate.

5.3.1 Performance Evaluation with Markov Assumptions

The fraction of time in low rate and the mean packet delay are measured using the dual-threshold, rate transition at service completion Markov model in Figure 5.10. The low service rate μ_1 was set to 0.1, and the high service rate μ was set to 1.0. This models Ethernet data rates, which increase in multiples of 10.

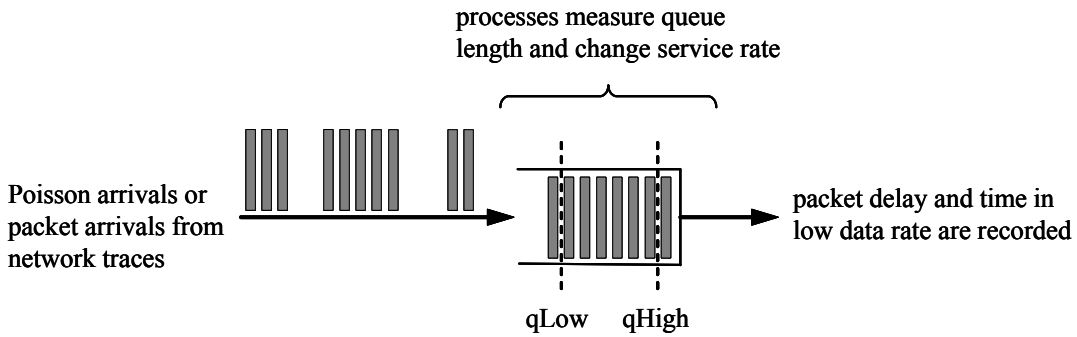


Figure 5.11. Simulation Model of State-Dependent Service Rate Queue

The threshold values k_1 and k_2 were set to 15 and 30 customers, respectively. Output buffer size for desktop NICs generally range from 32 KiB to 128 KiB [17] with the lower end of the range being more common. Ethernet frames are at most 1518 bytes and 30 frames sum up to approximately 44 KiB, which is a reasonable NIC buffer size. The parameter k_1 was arbitrarily set to half of k_2. The rate of arrivals λ was varied from 1% to 25% of the high rate μ. The numerical results for this case are shown in Figure 5.12, where the bars show the percentage of time in low rate and the line shows mean packet delay. The time units for measuring delay are units of mean service interval at high service rate μ. It is observed that initially time in low rate is 100% and the queue length and hence the mean response time increases with increasing λ (the queue length is insufficient to trigger a change to the high service rate). However, as λ exceeds μ_1, the queue length increases during service intervals at rate μ_1 and exceeds the upper threshold k_2 triggering a service rate change to μ As the difference between λ and μ_1 increases, the proportion of packets serviced at the higher rate increases and consequently, the mean response time decreases.

The simulation model (which was first validated by exactly reproducing the numerical results in Figure 5.12) was used to repeat the previous experiment with non-zero switching rate transition times of 10 and 100 mean service intervals at the high service rate μ. The simulation results are shown in Figure 5.13 and Figure 5.14. The time in low rate and the mean response time are shown in Figure 5.13 and the number of service rate transitions per 1000 time units for switching times of 0, 10, and 100 time units as the rate of

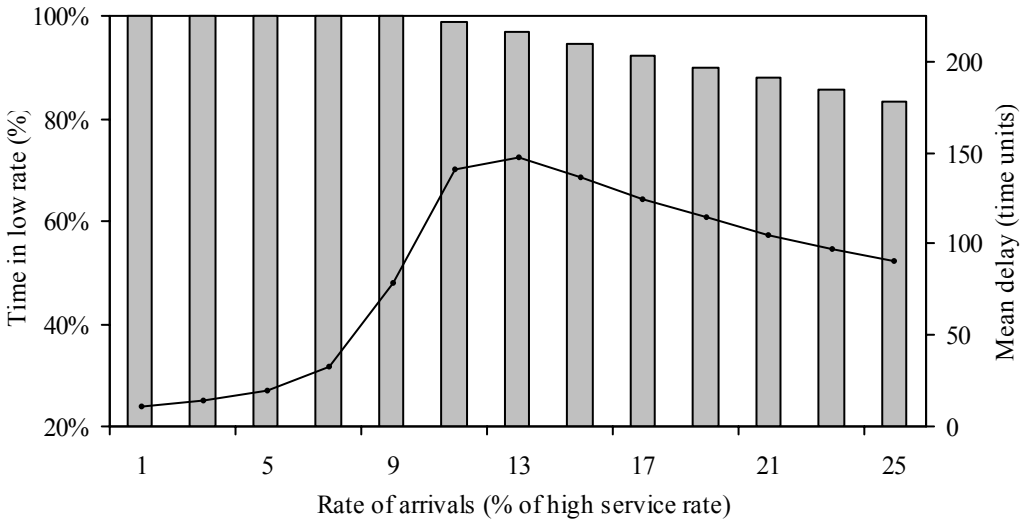


Figure 5.12. Numerical Results from Markov Model

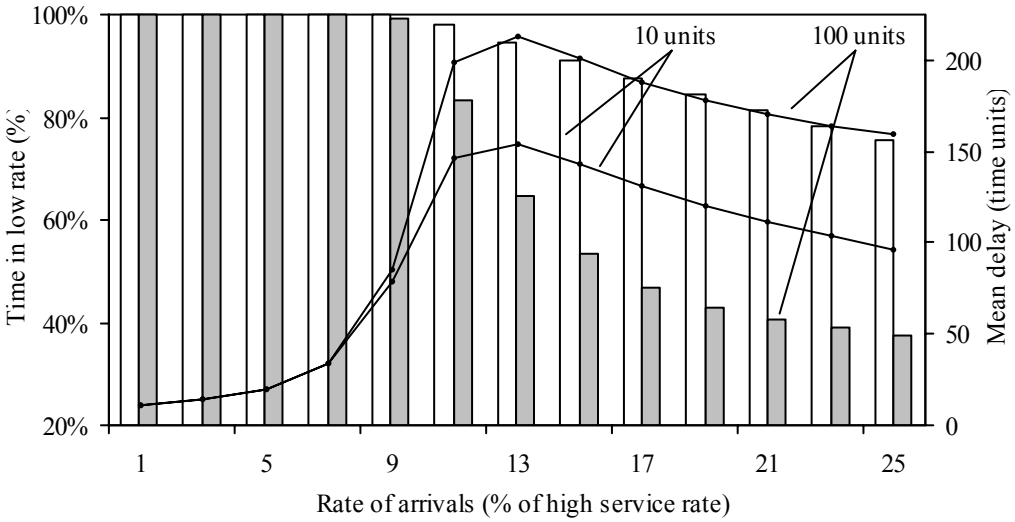


Figure 5.13. Simulation Results with Non-Zero Switching Time

arrivals is increased is shown in Figure 5.14. In Figure 5.12 and Figure 5.13 it can be observed that for up to 9% offered load (offered load is relative to the high service rate) the percentage of time in low rate is approximately 100% and is constant. This is not surprising since in this region the system is an M/M/1 with low rate service time at 90% offered load (with mean queue length of 9 customers). With an increase in the rate of arrivals, the fraction of time in low rate decreases with the biggest decrease shown (not surprisingly) for the largest switching time. For zero switching time with offered load at 25%, over 80% of the time is in low data rate.

In Figure 5.14, the number of rate switches is shown for non-zero switching time using the validated simulation model. For switching times of 0 and 10 time units, the number of switches increases with the increasing λ as the link oscillates between high and low service rates. However, for a switching time of 100 time units, the number of rate switches increases initially and then starts to decrease because of the greater number of arrivals queued during the longer rate switching time. This reduces the time spent servicing arrivals at rate μ_1 as the upper threshold k_2 is exceeded quicker than in the case of 0 or 10 time unit switching times. And as λ increases, a greater number of packets are queued during each rate switch and the time to drain the queue at rate μ increases causing less switching activity. Eventually, the time spent servicing arrivals at low service rate μ_1 becomes insignificant with increasing λ as by the time the service rate change is complete the queue length is again greater than k_2. At a rate of arrivals of 25% of the

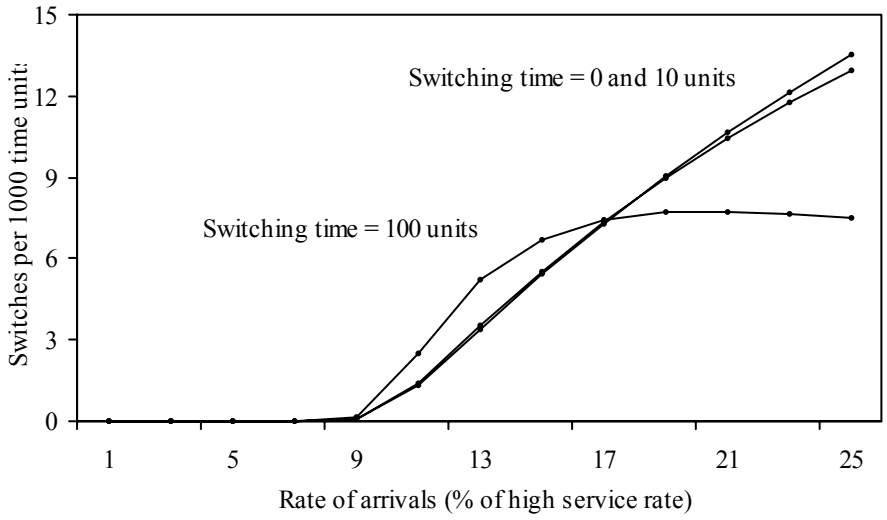


Figure 5.14. Rate Switches/1000 Time Units

high service rate, the low rate time is less than 40% (Figure 5.13), and the majority of that time is spent in rate switching rather than servicing arrivals at the low rate. Taking the practical example of an ALR-capable Ethernet link implementing the dual-threshold policy, for data rates of 100 Mb/s and 1 Gb/s, at 15% offered load the link would switch data rates more than 400 times per second. Assuming a T_{switch} of 1 ms, this is more than 40% of the time spent on switching data rates. Therefore, rate oscillations caused by the ALR policies is of interest and is addressed in Section 5.4.

5.3.2 Performance Evaluation with Traced Traffic

Six traffic traces were collected from 100 Mb/s Ethernet links. USF #1 through USF #3 were collected from the University of South Florida network in September and October 2004 using Ethereal by Mr. Joe Rogers. The USF "busy" traces are from links to student dormitory rooms and consist mainly of Peer-to-Peer traffic. The USF "average" trace is from the link to an office workstation and consists mainly of Microsoft Windows workgroup and web traffic. Three traces were collected in 2004 from the Portland State University network (PSU #1 through PSU #3) using tcpdump and are described in [53]. Table 5.2 summarizes the trace characteristics. Average link utilization is less than 5% in all cases.

The mean packet delay at fixed 10 Mb/s and 100 Mb/s data rates was measured using the simulation model. The results for packet delay are shown in Table 5.3. Operation at 10 Mb/s only results in excessive packet delays. The simulation model was used to measure the mean packet delay and percentage of time in

Table 5.2. Summary of Traces

Trace	Duration	Description	Avg. util.[1]	Num. of packets
USF #1	0.5 hours	Link to "busiest" user in USF	4.11 %	1,637,855
USF #2	0.5	Link to 10th busiest user	2.63	900,329
USF #3	0.5	Link to an average user	0.03	15,985
PSU #1	2.0	Link to a Windows desktop PC	0.13	77,163
PSU #2	2.0	Link connecting two switches	1.01	617,697
PSU #3	2.0	Link connecting switch to router	1.03	634,019

[1] Calculated at 100 Mb/s with 1518 byte packets assumed for PSU traces.

low rate when switching between 10 Mb/s and 100 Mb/s using ALR. The rate switching time was set to 1 ms and the threshold values k_1 and k_2 were set to 15 and 30 packets, respectively. The results for both packet delay and percentage of time in 10 Mb/s are shown in Table 5.3.

5.4 Rate Oscillation Problem with the Dual-Threshold Policy

A significant open problem with the dual-threshold ALR policy is how to handle smooth traffic at high utilization levels. Oscillation of link rate will occur if the packet arrival rate at the low link data rate is high enough to cause a high threshold crossing (at the low data rate), but not high enough to maintain the queue length above the low threshold at the high data rate. The number of rate switches during unit time period for the dual-threshold policy ($N_{oscilDual}$) can be computed using the Markov model given in Figure 5.10. The number of rate switches during a unit time period is given by the number of state transitions between

Table 5.3. Mean Packet Delay of Traces

Trace	10 Mb/s	100 Mb/s	ALR[1]	
USF #1	7.60 ms	0.09 ms	2.79 ms	99.42 %
USF #2	3.95	0.08	1.81	99.81
USF #3	196.29	0.05	1.48	99.99
PSU #1[2]	33.51	0.18	5.63	99.98
PSU #2[2]	2321.31	0.12	9.55	99.12
PSU #3[2]	1147.83	0.51	4.07	99.83

[1] First column is mean packet delay and second column is time in low rate.
[2] Packet size is assumed to be 1518 bytes for PSU traces.

states with differing rates of service. The number of state transitions during a unit time period out of a given state is given by the steady state probability of being in that state multiplied by the rate of leaving that state. With respect to the Markov model given in Figure 5.10, the service rate changes from μ to μ_1 when transitioning from state $k_1 + 1$ to $k_1 - 1$ (first term in 5.27) and changes from μ_1 to μ when transitioning from states $2k_2 - k_1 + 2n$ to $2k_2 - k_1 + 2n - 1$ ($n \geq 1$)(second term in 5.27). Therefore, the number of rate transitions is given by

$$N_{oscilDual} = \mu\pi_{k_1+1} + \sum_{n=1}^{\infty} \mu_1 \pi_{2k_2-k_1+2n} \ . \tag{5.27}$$

The number of rate switches for an ALR-capable link with data rates of 1 Gb/s and 100 Mb/s was computed using the Markov model and measured using the simulation model. Assuming a mean packet size of 1500 bytes, μ_1 was set to 8333 (this is the number of 1500 byte packets per second at 100 Mb/s), and the high service rate μ was set to 83,333 (this is the number of 1500 byte packets per second at 1 Gb/s). The rate of arrivals is varied as a fraction of μ. Results from both the Markov model and the simulation model are given in Figure 5.15 (the lines overlap) together with the results from the simulation model for a fixed packet size of 1500 bytes with exponentially distributed arrivals. Once the rate of arrivals is greater than 100 Mb/s, a rate switch to 100 Mb/s will inevitably result in a switch back to the high service rate of 1 Gb/s. At a link utilization level of 50%, the number of rate switches is 1062 per second for

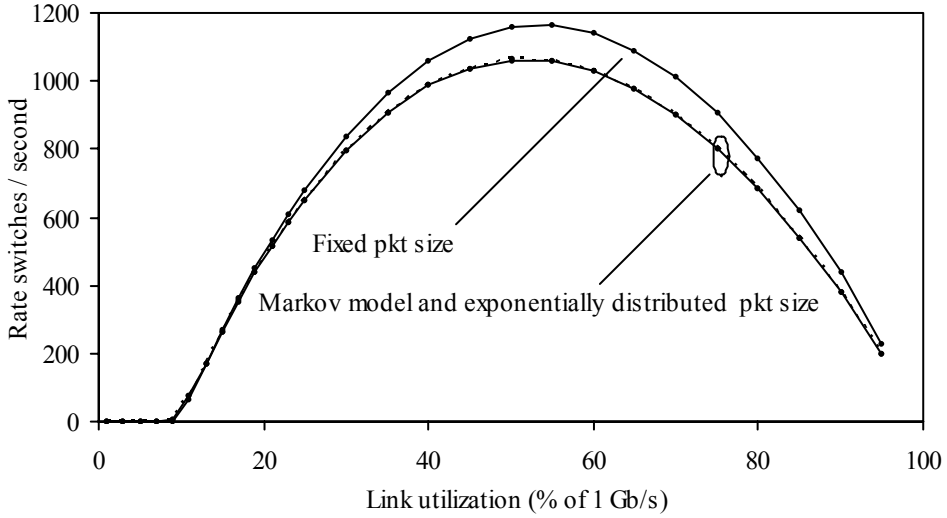


Figure 5.15. Rate Switches/Second for Dual-Threshold Policy

exponentially distributed packets and 1156 for fixed size packets. This is an increase of approximately 9% for fixed size packets. Using fixed size packets is similar to a M/D/1 queue and the queue length is smaller for fixed size packet and results in less time per oscillation and greater number of oscillations. Since in practice the rate switching time would be non-zero, link data rate oscillations results in additional delay and perhaps even in extra energy expenditure.

Additionally, the effect of link data rate oscillation upon the mean response time is explored. Similar to above, system with high link data rate of 1 Gb/s and a low link data rate of 100 Mb/s is considered. In this case, packets arrive into the output buffer consisting of Poisson (smooth) arrivals with packets of constant length 1500 bytes and the system (without ALR) is thus modeled as an M/D/1 queue. The two dashed line curves in Figure 5.16 show mean packet delay (for M/D/1) as a function of increasing arrival rate for 100 Mb/s and 1 Gb/s data rates. As the arrival rate (the utilization) increases when in low link data rate, the packet delay will reach an unacceptable level. At some utilization level less than 10%, the data rate should be switched to the high rate and remain there. Using the simulation model of the dual-threshold policy ALR controlled queue, the mean packet delay and number of data rate transitions for qLow = 0 KiB, qHigh = 32 KiB, and T_{switch} = 1 ms is measured. The mean packet delay for the ALR-controlled queue is the solid line curve in Figure 5.16. It can be seen that mean response time increases similarly to M/D/1 curve for 100 Mb/s but does not decline once the arrival rate (utilization) is greater than 100 Mb/s (which occurs at 10% utilization at 1 Gb/s). This is due to the dual-threshold policy causing the link data rate to oscillate

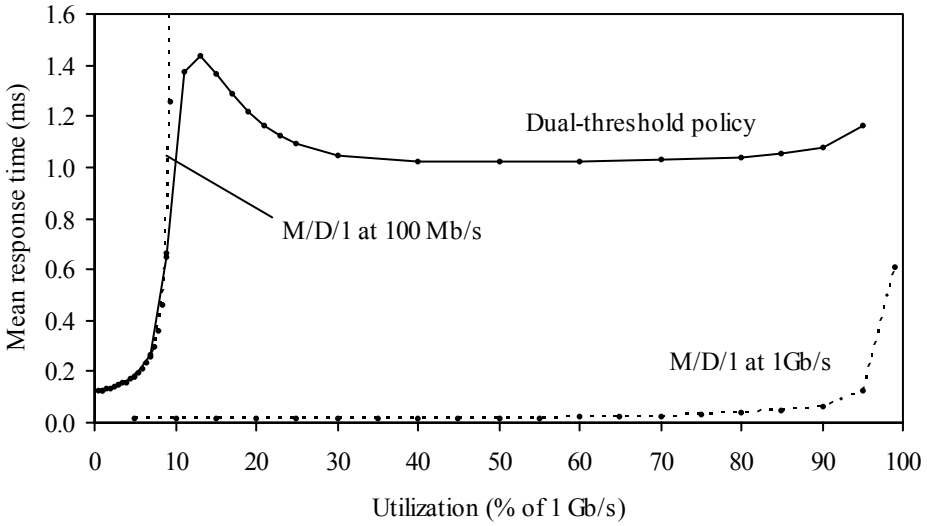


Figure 5.16. Mean Response Time of Dual-Threshold Policy

between 100 Mb/s and 1 Gb/s. For example, at 50% utilization the dual-threshold ALR policy results in a mean packet delay of about 1 s (i.e., approximately the value of T_{switch}) when operation at 1 Gb/s would result in a mean packet delay of about 0.04 ms. At the 50% utilization level with ALR, 49% of the time is spent in rate switching and an insignificant amount of time is spent in processing packets at the 100 Mb/s (energy saving) data rate (by the time the data rate switches from high to low, the queue length has exceeded qHigh).

Link rate oscillation presents a significant problem that needs to be overcome for implementation of ALR to be feasible under all possible conditions. A control policy that minimizes data rate oscillations by explicitly monitoring oscillations is presented in Chapter 6.

Chapter 6: Reducing Direct Energy Use – ALR Part 2

The dual-threshold policy described in Chapter 5 leads to link data rate oscillation during long data bursts at rates greater than the low link data rate. The utilization-threshold policy described and evaluated in this chapter is developed in order to prevent data rate oscillation – link utilization is explicitly monitored and used in the decision to transition between data rates.

In this chapter, Section 6.1 describes the utilization threshold policy and evaluates data rate oscillations when using this policy. The optimal utilization monitoring time for Poisson traffic is derived in Section 6.2 and a new method for generating synthetic Ethernet network traces is described in Section 6.3. Performance evaluation of the ALR utilization-threshold policy using synthetic traces is described in Section 6.4. The effects of the increased packet delay caused by ALR is described in Section 6.5 and extending ALR to more than two link data rates is described in Section 6.6.

6.1 ALR Utilization-Threshold Policy

The FSM in Figure 6.1 shows the detailed design for utilization-threshold policy and the FSM given in Figure 6.2 shows monitoring for link utilization by counting the number of bytes transmitted during a time period. Table 6.1 shows the timers, parameters, and variables associated with the new policy. Utilization monitoring is based on counting bytes sent in an interval tUtil. If the number of bytes transmitted is greater than the threshold value, data rate is not changed to a low rate though the queue length may be less than qLow. The FSM in Figure 6.1 is similar to the dual-threshold policy FSM shown in Figure 5.3 except for the transitions with the grey background. A detailed explanation for each of the FSM states and transitions follows:

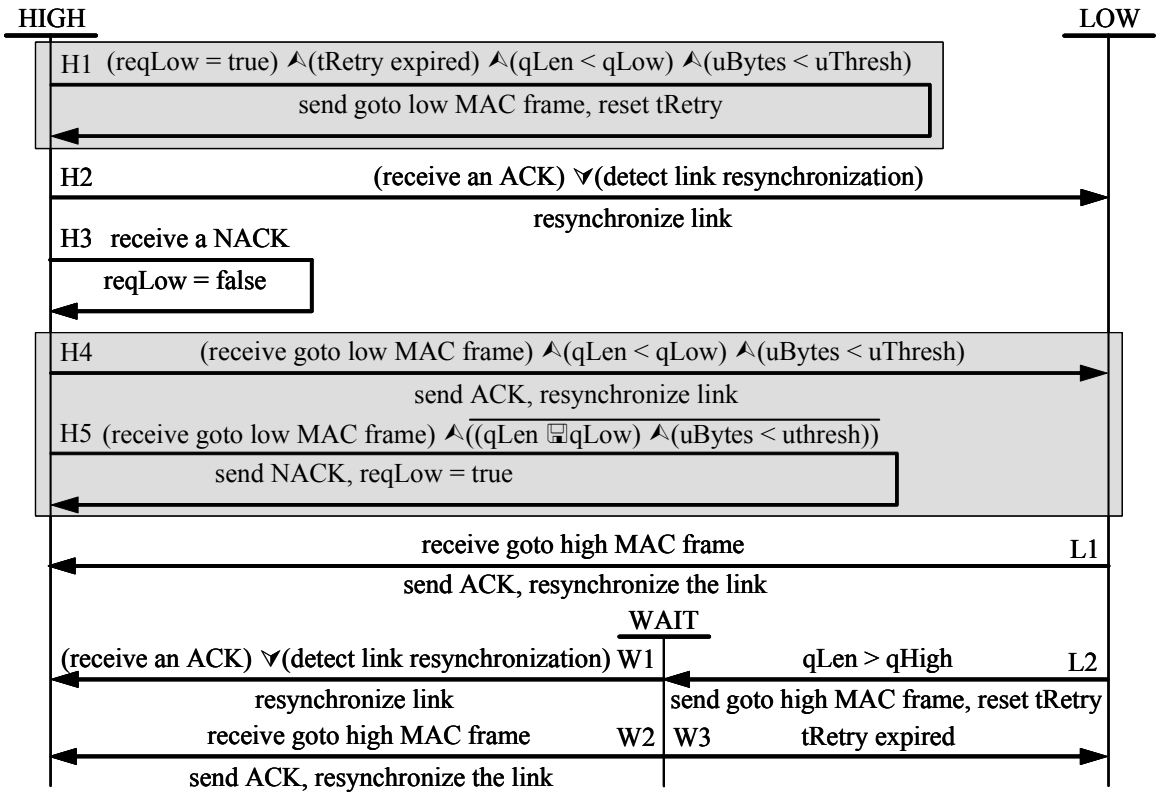


Figure 6.1. Modified FSM for ALR Utilization-Threshold Policy

State HIGH (FSM in Figure 6.1): NIC is operating in high link data rate.

Transition H1: The output buffer queue length (qLen) is less than the low queue threshold (qLow) and the number of bytes transmitted during the previous tUtil time period (uBytes) is less than the threshold value (uThresh). The timer tRetry has expired and it is the NIC's turn to request a low data rate (reqLow = true). A goto low ALR REQUEST frame is sent to the link partner NIC. The timer tRetry is reset.

Transition H4: A goto low ALR REQUEST frame is received from the link partner NIC. qLen is less than qLow and the number of bytes transmitted during the previous tUtil time period (uBytes) is less than the

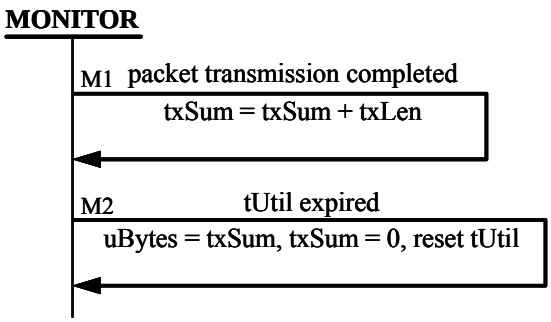


Figure 6.2. New FSM for Utilization Monitoring

Table 6.1. FSM Timers, Parameters and Variables for Utilization-Threshold

Name	Description
tRetry	Timer for handshake retry (ACK, NACK lost)
qLow	Output buffer queue length low threshold in bytes
qHigh	Output buffer queue length high threshold in bytes
qLen	Present output buffer queue length in bytes
reqLow	Request low rate flag. A NIC can request a low rate only if this is set to true and is designed to prevent one link partner repeatedly requesting a low rate.
tUtil	Timer for utilization measurement interval. This timer is reset to denote the beginning of a new interval.
uBytes	Number of bytes transmitted during the last tUtil interval
uThresh	Utilization threshold in bytes. This value is pre-computed.
txLen	Transmitted packet length in bytes
txSum	Variable for accumulating the number of bytes transmitted during the current tUtil interval

threshold value (uThresh). An ALR ACK frame is transmitted, and the link is resynchronized at low rate.

Transition H5: A goto low ALR REQUEST frame is received from the link partner NIC. qLen is not less than qLow or the number of bytes transmitted during the previous tUtil time period (uBytes) is not less than the threshold value (uThresh). An ALR NACK frame is transmitted and the variable reqLow is set to true.

State MONITOR (FSM in Figure 6.2): Process for monitoring link utilization

Transition M1: A packet transmission is completed. The number of bytes transmitted (txLen) is added to the number of bytes transmitted so far during the current tUtil time period (txSum).

Transition M2: End of tUtil time period (timer has expired). The number of bytes transmitted during the previous tUtil time period (txSum) is copied to the variable uBytes and txSum is initialized. tUtil is reset in order to begin the new tUtil time period.

Figure 6.3 shows a pseudocode description of the utilization-threshold policy. This is similar to the pseudocode description of the dual-threshold policy (in Figure 5.5) with one additional test in (b) for uBytes less than uThresh. Similar to the pseudocode description in Figure 5.5, (a) contains FSM transitions L1, L2, W1, W2, and W3 and (b) contains transitions H1, H2, H3, H4, and H5. The pseudocode

```
if (qLen is greater than qHigh threshold) then
  handshake for a high link data rate
```

(a) Process for state LOW

```
if (qLen is less than qLow threshold) then
  if (uBytes less than uThresh) then
    handshake for a low link data rate
```

(b) Process for state HIGH

Figure 6.3. Pseudocode Description of the Utilization-Threshold Policy

description omits details on the handshake procedure. The process in (a) is triggered on receiving a frame and the process in (b) is triggered when sending a frame.

The simulation model described in Chapter 5 is modified by implementing the new policy. Utilization monitoring is implemented as a separate process running in parallel with the variable service rate queue process. The number of rate changes per second for the utilization threshold policy is modeled for smooth (Poisson) traffic using the simulation model and an analytical model. Developing an analytical model for the utilization-threshold policy is much more complicated than for the dual-threshold policy. This policy cannot be modeled as a Markov chain due to the fixed tUtil time period. However, the oscillations can be modeled by considering two separate M/M/1 queues – one with service rate μ_1 and the other with service rate μ. For a given rate of arrivals λ, the mean time taken for a oscillation can be considered as the sum of three separate components. They are:

1. Mean time in low service rate μ_1, which is the mean passage time ($T_{passLow}$) for the number of packets in the system to increase from k_1 to the average number of packets in the system when changing to high rate. The method described in [90] for computing the mean passage time for continuous time Markov chains was used in this calculation.
2. The mean passage time for the number of packets in the system to decrease from average number of packets in the system when changing to high rate to k_1 ($T_{passHigh}$). This was also calculated using the method described in [90].
3. The mean time in service rate μ once the queue length has dropped to k_1 is determined by the probability of uBytes being less than uThresh during a tUtil time period (q) and the probability of

the queue length being less than the threshold k_1 at the end of a tUtil time period and this is given by $tUtil \Big/ q\sum_{n=0}^{k_1-1}\pi_n$ where π_n is the steady state probability of having n customers in the system for a M/M/1 queue where the rate of service is μ.

Thus, the total time taken for a single oscillation ($T_{oscilUtil}$) can be given by

$$T_{oscilUtil} = T_{passLow} + T_{passHigh} + \frac{tUtil}{q\sum_{n=0}^{k_1-1}\pi_n}. \tag{6.1}$$

Since each oscillation consists of two rate switches, the number of rate switches during a unit time period for the utilization-threshold policy ($N_{oscilUtil}$) can be given by

$$N_{oscilUtil} = \frac{2}{T_{oscilUtil}}. \tag{6.2}$$

Note that (6.2) gives an approximate answer since it is assumed that the long term steady state probabilities from a M/M/1 chain can be used for the computation when the service rate is μ, though the systems changes service rates periodically.

The number of rate switches per second for an ALR capable link with the utilization-threshold policy is evaluated for smooth (Poisson) traffic using both the analytical model and the simulation model. A mean packet size of 1500 bytes is assumed and the service rate μ is set to 83,333 and μ_1 is set to 8,333,

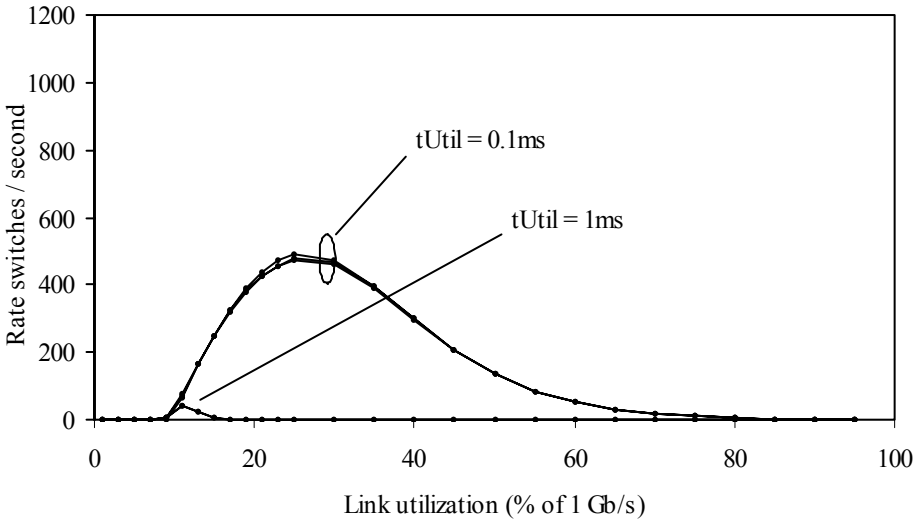


Figure 6.4. Rate Switches/Second for Utilization-Threshold Policy

corresponding to data rates of 1 Gb/s and 100 Mb/s, respectively. The time tUtil values of 0.1 ms and 1 ms are evaluated. Also, Poisson arrivals with a fixed packet size of 1500 bytes are also evaluated for a tUtil value of 0.1 ms. The results are shown in Figure 6.4. It is observed that the utilization-threshold policy is more effective at reducing data rate oscillations compared to the dual-threshold policy with approximately 500 rate switches/second compared to over a 1000 rate switches/second for the dual-threshold policy (shown in Figure 5.13), and that increasing the tUtil time period reduces oscillations. The number of oscillations appears to be virtually identical with both fixed size and exponentially distributed packets. At a link utilization of 20%, tUtil = 0.1 ms gives about 400 rate switches/second while tUtil = 1 ms gives 0 rate switches/second. The similarity of the behavior for the simulation and analytical models serves to validate the accuracy of the simulation model. Determining the optimal tUtil time period is described in Section 6.2.

6.2 Optimal Sampling Time for the Utilization-Threshold Policy

For exponentially distributed inter-arrival times (Poison traffic), the minimum value of tUtil sampling period needed to achieve a given margin of error and level of confidence can be derived. From the central limit theorem, the probability of a population mean, $1/\mu$ (μ being the mean rate of arrivals), being within a confidence interval is

$$\Pr\left(\frac{T}{n}-\frac{1}{\mu}a<\frac{1}{\mu}<\frac{T}{n}+\frac{1}{\mu}a\right)=p \tag{6.3}$$

where

$$\frac{1}{\mu}a=\frac{zs}{\sqrt{n}} \tag{6.4}$$

and T is the time period during which arrivals were observed, s the standard deviation of the sample, a the desired accuracy ($0<a<1$), z the normal variate for the desired confidence interval ($z=1.96$ for 95% confidence), and n the number of arrivals. The number of arrivals required for a margin of error of $(1/\mu)a$ is then,

$$n=\frac{z^2s^2\mu^2}{a^2}\,. \tag{6.5}$$

For exponentially distributed inter-arrival times, $s^2 = 1/\mu^2$. Therefore, (6.5) can be simplified to

$$n = \frac{z^2}{a^2}. \tag{6.6}$$

Since n is known, optimal T (optimal tUtil value) can be calculated for a given rate of arrivals:

$$T = \frac{z^2}{a^2 \mu} \tag{6.7}$$

With μ defined as a mean rate of packet arrivals per second, T will correspond to the value of tUtil in seconds. For a link date rate of 1 Gb/s, fixed length 1500 byte packets, and 5% link utilization the mean rate of arrivals (μ) is 4166.67 packet arrivals per second. With $a = 0.1$ and $z = 1.96$, tUtil = 92.2 ms. In 92.2 ms (at 5% utilization of 1 Gb/s), uThresh is 576,250 bytes.

In [87] it was shown that the *network power* metric, which is defined as throughput divided by delay, is maximized at a throughput of 50% link capacity. At throughputs greater than 50%, the packet delay increases rapidly. Therefore, the target is to operate the link at the low data rate for at most 50% utilization (measured at the low data rate, which is 5% utilization at the high data rate). If the link utilization is greater than 50% at low rate, the link is operated at the high rate.

6.3 Generating Synthetic Ethernet Link Traffic

The policies developed for ALR need to be effective for future 1 and 10 Gb/s Ethernet traffic. Characterizations of existing 100 Mb/s desktop to LAN switch Ethernet links show that traffic is very bursty. Table 6.2 shows the traffic characteristics of six traffic traces, each of duration 30 minutes or greater, taken from operational Ethernet links in 2004 at USF and Portland State University (PSU) [53]. These traces were previously described in Section 5.3.2. In Table 6.2 a burst is defined as one or more consecutive 10 millisecond time periods with 5% or greater utilization. Future 1 and 10 Gb/s Ethernet traffic will likely have very similar characteristics to the traffic on current 100 Mb/s links where the majority of traffic volume is in bursts and background "network chatter" (e.g., ARP broadcasts) is negligible compared to the link data rate. File transfer applications, such as Peer-to-Peer file sharing, are one common source of bursty traffic.

Table 6.2. Characteristics of Actual (Traced) Traffic

Trace	Burst Len	CoV	Hurst	Util.[1]
USF #1	17.5 KiB	1.76	0.66	4.11%
USF #2	14.6	2.18	0.76	2.63
USF #3	24.1	13.95	0.82	0.03
PSU #1	26.7	2.21	0.73	0.13
PSU #2	642.8	6.07	0.91	1.01
PSU #3	24.4	3.54	0.91	1.03

[1] Computed at 100 Mb/s (1518 byte packets assumed for PSU)

In [38] Ethernet traffic is modeled as bursty with Cauchy distributed burst lengths and exponentially distributed idle inter-burst times. Due to the lack of a mean, the Cauchy distribution, though useful when modeling an existing trace, can be intractable to use for generating synthetic traces with pre-determined levels of utilization. In [28] it is shown that traffic bursts follow a bounded Pareto distribution which follows from the distribution of file sizes of files stored in web servers. The probability distribution for bounded Pareto is

$$f(x) = \frac{\alpha k^{\alpha}}{1-(k/p)^{\alpha}} x^{-(\alpha+1)} \tag{6.8}$$

where k and p are the lower and upper bounds on possible values of $f(x)$ and α is the Pareto index.

A synthetic traffic generator was implemented to generate a text file containing time-stamps and packet length pairs as a synthetic traffic trace. Burst size (in bytes) is bounded Pareto distributed and inter-burst idle time is exponentially distributed. The input parameters to the traffic generator are:

1. Data rate (100 Mb/s, 1 Gb/s, or 10 Gb/s)
2. Minimum burst size in bytes (k)
3. Maximum burst size in bytes (p)
4. Pareto index for bursts (α)
5. Burst intensity (as a percentage of link data rate)
6. Packet size distribution (packet size is fixed or empirically distributed)
7. Mean utilization or mean inter-burst time (if mean utilization is given, then this value is used to determine the mean inter-burst time)

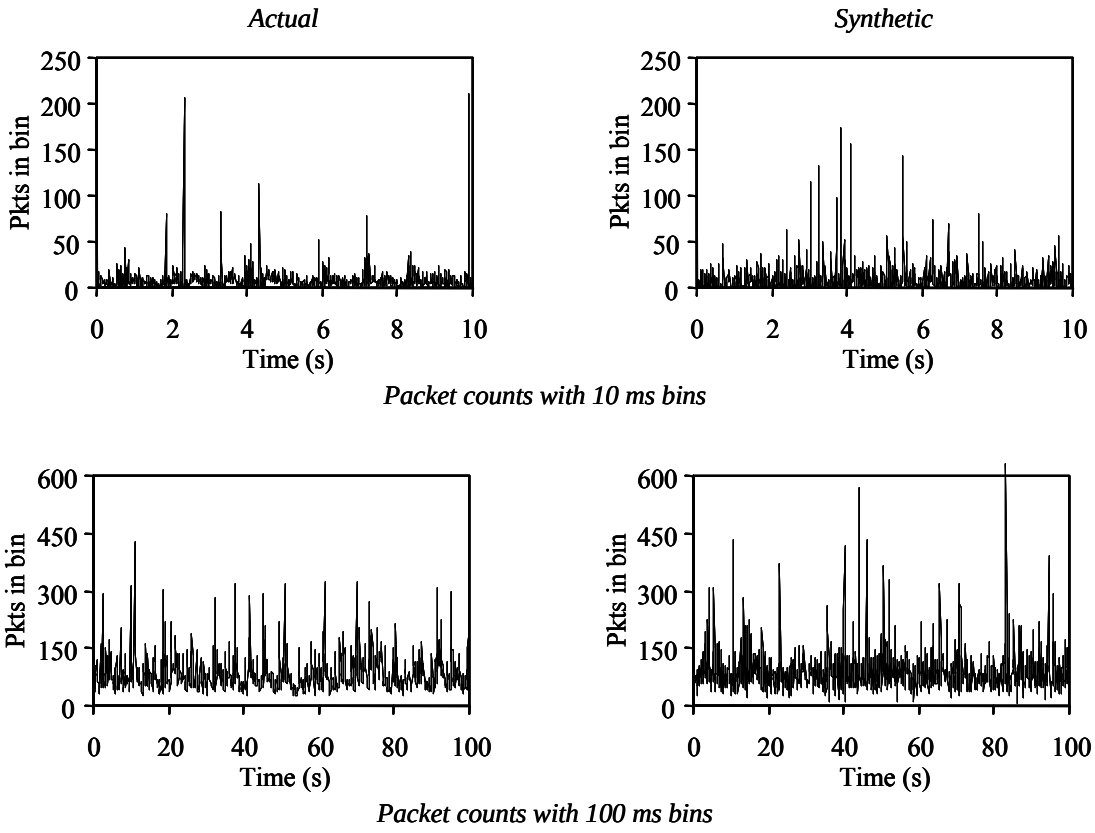


Figure 6.5. Packet Counts for Different Time Scales

Figure 6.5 shows a time plot of packet counts for actual and synthetic 100 Mb/s traffic. The trace USF #1 with a utilization of 4% was used for the actual traffic. Two time scales (10 ms and 100 ms bins) are shown. The parameter values used for the synthetic traffic were $k = 1518$ bytes, $p = 1$ GiB, $\alpha = 1.5$, burst intensity of 80%, target utilization of 4%, and empirical packet size distribution based upon the actual packet size distribution of trace USF #1. The actual and synthetic traffic visually appear to be the same in

Table 6.3. Actual Versus Synthetic Traffic

Characteristic	USF #1	Synthetic
Mean inter-packet time (ms)	1.10	1.06
CoV of inter-packet times	1.76	3.81
Mean packet size (bytes)	577	526
CoV of packet size	1.16	1.15
Hurst parameter of packet counts	0.66	0.64
Utilization (% of 100 Mb/s)	4.11	3.95

Figure 6.5. Table 6.3 shows the summary statistics of the two traces further demonstrating the ability of the traffic generator to generate synthetic traffic with realistic measures. Note the large Hurst parameter value that indicates a high degree of self similarity in the traffic.

6.4 ALR Simulation Results with Synthetic 1 Gb/S Ethernet Link Traffic

The behavior and performance of the utilization-threshold policy is studied using a simulation model of an Ethernet NIC and a simulation model of an output queued switch. The metrics of interest are mean response time (packet delay) and time in low rate (energy savings). The parameter uThresh is a link utilization percentage during time period tUtil and is defined to be the number of bytes that can be transmitted at 5% link utilization during time period tUtil at high data rate. The link will be prevented from switching to low rate if the number of bytes transmitted during a tUtil time period exceeds the value of uThresh.

A single-server, variable service rate queue models an Ethernet NIC and is adapted from the model validated in Chapter 5. The modifications to implement the utilization-threshold policy were validated in Section 6.1 (see Figure 6.4) by comparing the number of rate switches against an analytical model. The utilization-threshold policy was also implemented in a simulation model of a 16-port output queued switch. The switch model was developed and validated in [134]. For this study, ALR has been added to the output buffers. The inputs to the models are described in the experiment descriptions.

6.4.1 Simulation Experiments

Simulation experiments were designed to evaluate the utilization-threshold policy and to compare the performance to the dual-threshold policy. For the experiments described next, unless stated otherwise, qLow = 0 bytes, qHigh = 32 KiB, high and low data rates were 1 Gb/s and 100 Mb/s, and T_{switch} = 1 ms. As previously mentioned in Section 5.3.1, 32 KiB is a reasonable output buffer size and hysteresis is maximized by setting qLow to 0 bytes. And as described in Section 5.1.1, 1 ms is a conservative estimate for T_{switch}. It was assumed that all ALR data rate switch requests are successfully ACKed. All experiments

were run for at least 10 million packet arrivals. Unless otherwise noted, the values of tUtil used were 0.1, 1, 10, and 100 ms.

Two scenarios for future high-bandwidth Ethernet traffic were considered. Multiplayer gaming, virtual reality environments, and collaborative file sharing applications generate relatively small bursts of data with millisecond-scale inter-burst times. This scenario was evaluated in *Bursty traffic experiment #1*. A corporate or scientific network where gigabyte-scale datasets and backups are transferred several times per day (i.e., with many minutes or hours between bursts) was evaluated in *Bursty traffic experiment #2*. Network "background chatter" between bursts was considered to be negligible compared to link data rate. The experiments are described as follows:

1. *Smooth traffic experiment:* The effect of varying tUtil for smooth (Poisson) traffic was studied in this experiment. A fixed packet size of 1500 bytes was used. The traffic utilization ranged from 1% to 95%. The metric of interest is mean response time (mean packet delay).
2. *Single burst experiment #1:* The transient effects of rate switching were studied in this experiment. A single burst comprised of Poisson traffic at 80% utilization and 0.4 seconds in duration was studied. The metric of interest is packet delay.
3. *Single burst experiment #2:* The effect of rate switching upon the total time taken to transmit a burst was studied in this experiment. A single burst comprised of Poisson traffic at 80% utilization was transmitted with the link initially in the low data rate. Burst size was varied from 10 KiB to 1 GiB and values for T_{switch} were 1 ms, 10 ms, and 100 ms.
4. *Bursty traffic experiment #1:* The traffic model described in Section 6.3 was used to generate synthetic traces with mean burst size of 8.4 KiB with target utilization ranging from 1% to 25%. The parameter values were k = 1518 bytes, p = 2.5 GiB, α = 1.5, and burst intensity of 80%. The resulting Hurst parameter and CoV range from respectively 0.80 and 3.35 (1% utilization) to 0.81 and 2.33 (25% utilization). The metrics of interest were mean response time, time in (energy saving) low rate and the CoV of the packet inter-departure times.
5. *Bursty traffic experiment #2:* The traffic model was used to generate synthetic traces with mean burst size of 867 MiB (mean burst duration is 9.09 seconds at 1 Gb/s) with inter-burst times ranging from 10 seconds to 100,000 seconds (approximately 1 day). The parameter values were

k = 250 MiB, p = 10 GiB, α = 1.1, and burst intensity of 80%. The tUtil values studied were 1 ms, 10 ms, and 100 ms.

6. *Switch experiment:* This experiment studied measurable energy savings in a LAN switch for a desktop to switch link. In [49] the power consumption of a 24-port Cisco Catalyst 2970 switch with no Ethernet links attached was measured, and when scaled to 16 ports, is 46 W. Each Ethernet link operating at 100 Mb/s or 1 Gb/s added an additional (measured) 0.3 W or 1.8 W, respectively, to the switch power consumption. All power measurements were made at the wall socket using an AC power meter. A switch with 16 full-duplex Ethernet ports was modeled where input traffic streams were generated with the parameters used for *Bursty traffic experiment #1*. Utilization (switch offered load) was increased from 1% to 15% of the 1 Gb/s high data rate. All packets in a burst had the same destination port in the switch. The metrics of interest were power consumption (W) and mean switch delay for packets (time difference between entry and exit for a packet). The values of tUtil studied were 10 ms and 100 ms.

6.4.2 Results from Simulation Experiments

The results from the *Smooth traffic experiment* are shown in Figure 6.6. Previously, In Section 6.2, it was calculated that the optimal tUtil value for detecting 5% link utilization at 1 Gb/s with Poisson traffic was 92 ms. In Figure 6.6 it can be observed that for tUtil = 100 ms, the mean response time was reduced

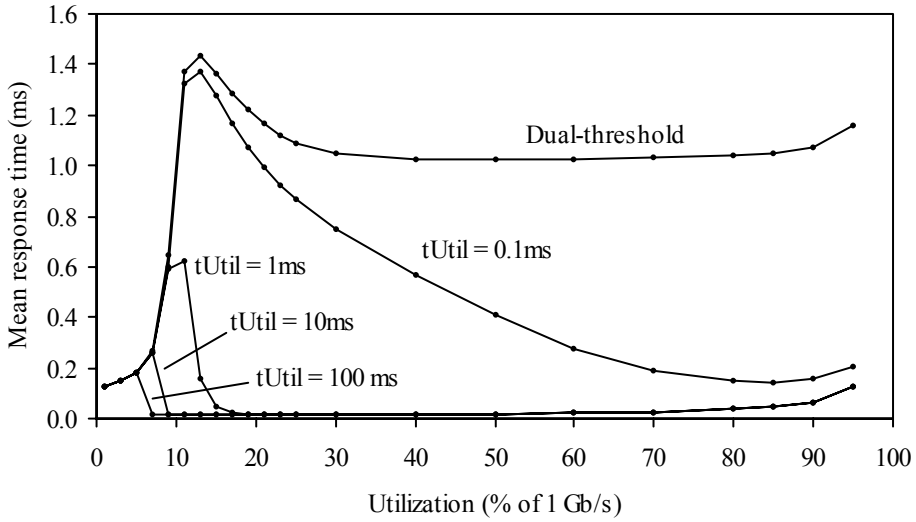


Figure 6.6. Results from Smooth Traffic Experiment

when utilization exceeded 5% (at high data rate) due to the utilization-threshold policy detecting that the link utilization was greater than 5% and maintaining a constant 1 Gb/s link data rate. It can be seen that as the tUtil value decreased to 0.1 ms, the link utilization level at which the mean response time reaches its peak increases. This was due to the length of tUtil being insufficient to receive the number of bytes that denotes link utilization greater than 5% during all tUtil time periods. Therefore, at the end of time periods where the uBytes < uThresh, the link rate was set to 100 Mb/s and this resulted in a greater response time, both due to the lower service rate and T_{switch} time. These experimental results validate the analysis in Section 6.2.

The results for the *Single burst experiment #1* are shown in Figure 6.7 (only for tUtil = 0.1 ms) and Table 6.4. A spike in the response time denotes a rate switch. The initial rate switch occurred at time = 0.1 second (which is the burst start time). For tUtil = 0.1 ms it is observed that five rate switch oscillations occurred. These oscillations increased the mean response time. The mean response time was

Table 6.4. Results from Single Burst Experiment #1

Characteristic	tUtil value (ms)			
	0.1	1	10	100
Rate changes	12	2	2	2
Mean response time (ms)	0.19	0.05	0.05	0.05
Post burst lag (ms)	0.19	1.19	12.2	102

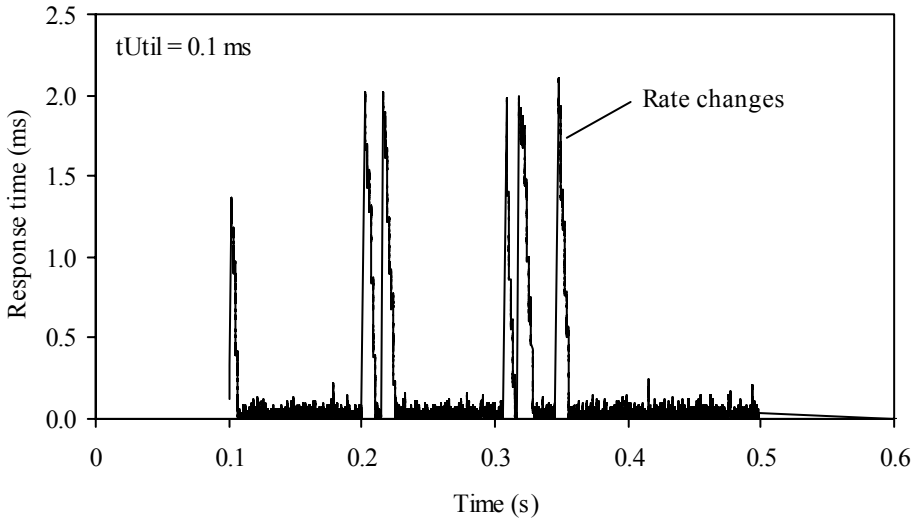


Figure 6.7. Results from Single Burst Experiment #1

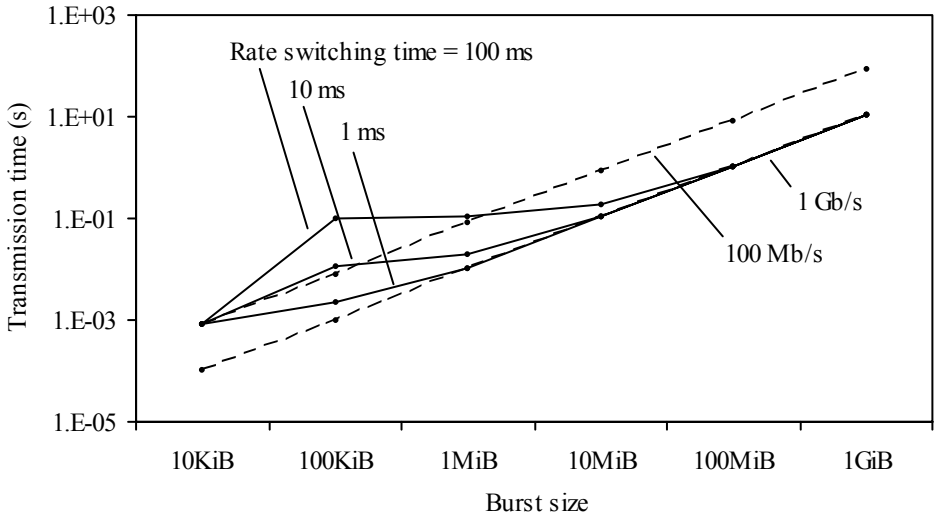


Figure 6.8. Results from Single Burst Experiment #2

0.19 ms with the 90th percentile at 0.73 ms and the 99th percentile at 1.88 ms. In Table 6.4, the number of rate switches, the mean response time, and the post burst lag for varying tUtil values are shown. The post burst lag time gives the time difference between the end of a burst and the final rate switch to 100 Mb/s.

The results for the *Single burst experiment #2* are shown in Figure 6.8 together with the times to transmit the burst at a constant data rate of 100 Mb/s or 1 Gb/s (the dashed lines). It can be observed that for a rate switching time of 1 ms, the impact upon burst transmission time is negligible. The transmission times with ALR and without ALR at 1 Gb/s are similar for bursts greater than 1 MiB. Since the burst is at 80% of 1 Gb/s, the cost of servicing a portion of the burst at 100 Mb/s and the cost of rate switching is amortized over the duration of the burst. For rate switching times of 10 ms and 100 ms, there is greater increase in transmission time, but as the burst size increases, again the transmission times tend to be similar. For example, time taken to transmit a 100 KiB burst through an ALR capable link with 1 ms rate switching time is 2.2 times the time taken at 1 Gb/s and is 98.9 times with 100 ms rate switching time. For a burst of 100 MiB, the time taken is 1.0 times and 1.0 times, respectively.

The results for the *Bursty traffic experiment #1* for the utilization-threshold policy are shown in Figures 6.9, 6.10, and 6.11. As tUtil and link utilization increased, it can be observed that the mean response time and time in low rate decreased. From the *Single burst experiment #1* it is known that the post burst lag increases with an increasing tUtil value. With greater lag time, the possibility that the next burst will begin before the link data rate switches to 100 Mb/s increases and therefore, time in low rate decreases.

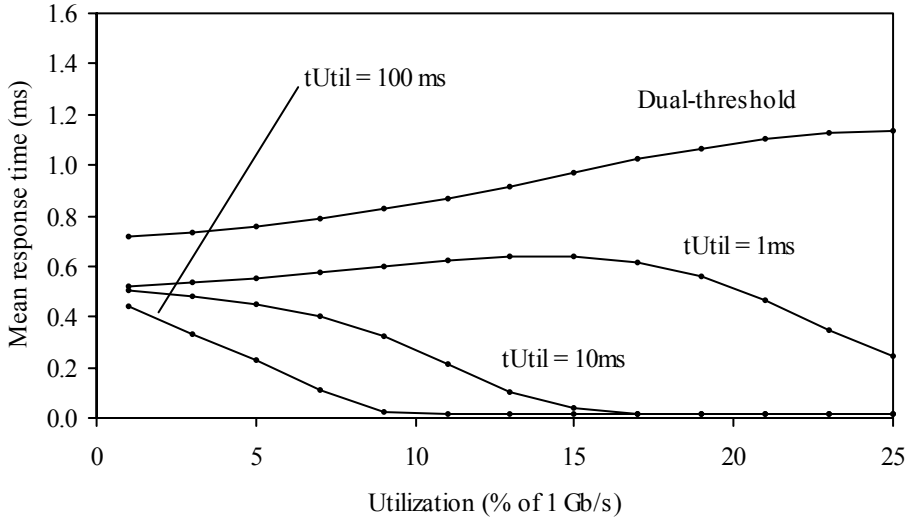


Figure 6.9. Mean Response Time from Bursty Traffic Experiment #1

With increasing utilization, the inter-burst time decreases and therefore, the possibility that for a given tUtil value the next burst will start before the data rate changes to 100 Mb/s increases. Consequently, fewer packets are serviced at the lower data rate and mean delay is decreased. With the dual-threshold policy it can be observed that rate switch oscillations caused the mean response time to increase as the utilization increased.

Regarding the CoV of the inter-packet departure times shown in Figure 6.11, it can be inferred that queueing while in low rate helps reduce the variance when link utilization is low – the packets in a burst are "spread out" due to queueing and transmission at the lower rate. As the link utilization increases, for tUtil

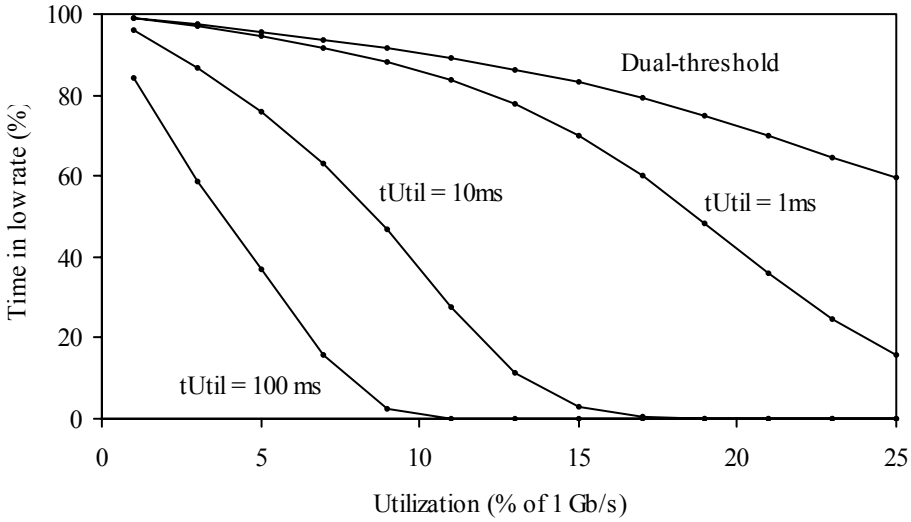


Figure 6.10. Time in Low Rate from Bursty Traffic Experiment #1

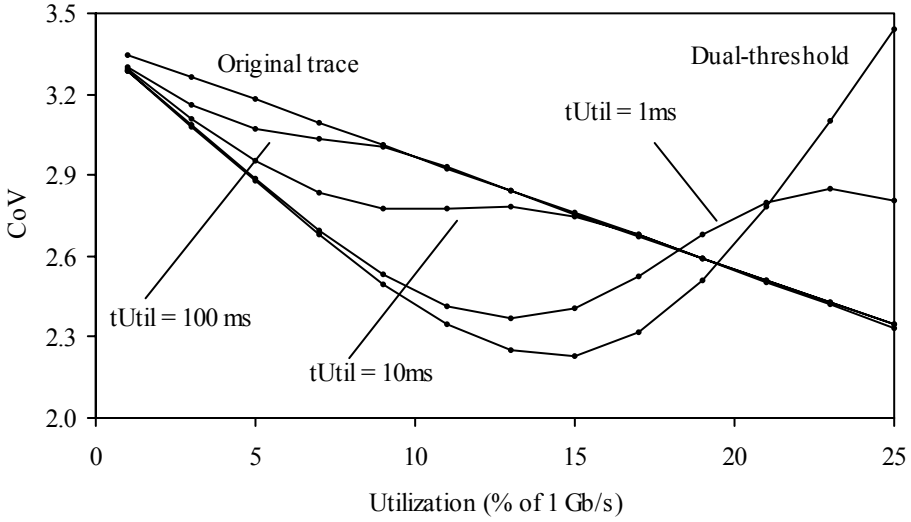


Figure 6.11. Packet Inter-Departure Time CoV from Bursty Traffic Experiment #1

values of 100 ms and 10 ms, as the time in low rate tends to zero, the inter-packet departure time variance tends to the variance of the inter-packet arrival times – packets are serviced as they arrive. Results for the dual-threshold policy and the utilization-threshold policy with tUtil = 1 ms show that the continued oscillations between the link data rates result in increased variability in the inter-packet departure times compared to the arrival times.

For the *Bursty traffic experiment #2* it was found that for the utilization-threshold policy the data rate switched from 100 Mb/s to 1 Gb/s at the beginning of a burst and returned to 100 Mb/s data rate at the end of the burst for the tested tUtil values. With the utilization-threshold policy there were no rate switches during the bursts. The post burst lag time was insignificant compared to the inter-burst times of 10 seconds to 100,000 seconds and thus virtually all non-burst time was spent in the low (and energy saving) 100 Mb/s data rate. With the dual-threshold policy, it was found that the link data rate oscillated between 100 Mb/s and 1 Gb/s during the bursts leading to a mean response time of over 1 ms.

The results for the *Switch experiment* are shown in Figure 6.12. The switch power consumption is given for tUtil = 10 ms and 100 ms (shown as bars) with the dashed line indicating switch power consumption without ALR. The solid lines show the mean response time for tUtil = 10 ms and 100 ms, and with ALR disabled (constant 1 Gb/s data rate). It can be observed that with an average utilization of 5%, a power savings of about 15 W (20%) is possible for tUtil = 10 ms. The increase in mean response time

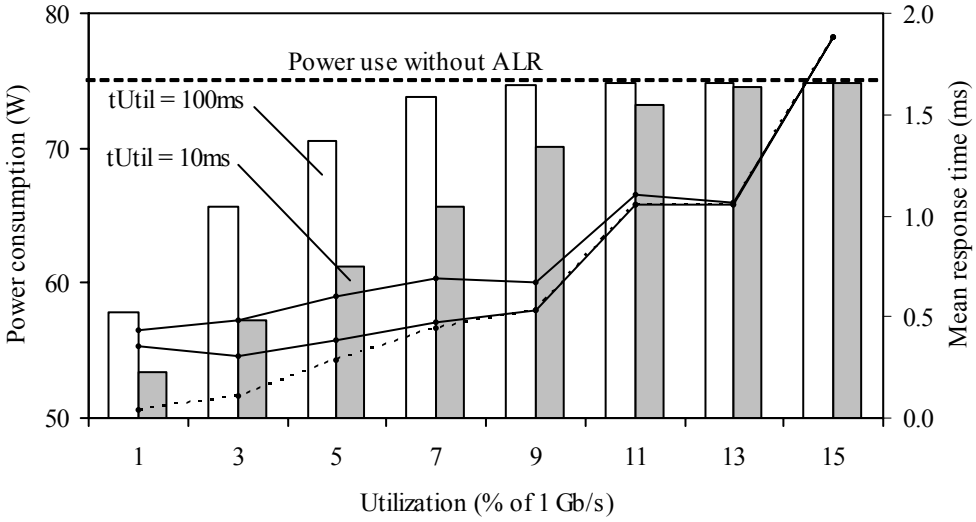


Figure 6.12. Results from Switch Experiment

when ALR is enabled for both values of tUtil was less than 0.5 ms, which can be considered negligible for most applications.

In summary, the improved ALR utilization-threshold policy based upon explicit utilization monitoring is shown to be capable of minimizing rate switch oscillations with smooth traffic. It is shown that with bursty traffic, at an average link utilization of 5% or less (at 1 Gb/s) it is possible to operate a 1 Gb/s Ethernet link at 100 Mb/s for 80% of the time or more at the expense of a packet delay of less than 0.5 ms (see Figures 6.9 and 6.10 for tUtil = 10 ms). This level of delay would be imperceptible for end-users of desktop PCs at offices and at residences. Considering that the measured link utilization of Ethernet links is generally much lower than 5%, an ALR capable 1 Gb/s Ethernet link would be operating at the energy saving 100 Mb/s data rate for the vast majority of the time.

Modeling the power consumption of a first-level Ethernet switch with ALR-capable switch ports shows that at typical Ethernet link utilization levels of less than 5%, the power consumption of the switch is reduced by more than 20%. This is without considering any power consumption savings in switch internal components.

6.5 Effect of Increased Delay on Applications

ALR causes an increase in packet delay when compared to operating a link at the highest possible data rate at all times. The effect of this increased packet delay on network users needs to be considered. ALR is

primarily intended for desktop links, thus the increased packet delay caused by ALR affects only a single desktop PC user. The possible impact of the additional delay caused by ALR can be evaluated within the context of the effect of this additional delay on existing end-to-end delays. For LANs end-to-end delays are typically less than 1 ms. For long distance connections on the Internet, end-to-end delays are 10s to 100s of milliseconds.

Studies of media synchronization and jitter have shown that delays in excess of 80 ms between audio and video streams in motion pictures are perceptible to humans [120]. For IP telephony, ITU Recommendation G.114 requires a mouth-to-ear delay of 150 ms for "excellent" voice quality [102] and studies of interactive computer game players have shown that delays greater than 100 ms affect the player accuracy [75]. The experiments in Chapters 5 and 6 showed that the largest mean packet delay when using ALR is approximately 10 ms (for the case of 100 Mb/s and 10 Mb/s using trace PSU#2 – in Table 5.3) and this value is less than the absolute delay perceptible to humans. Studies of virtual reality environments have shown that for differences between latencies to be perceptible, the differences must be greater than 20 to 30 milliseconds [3, 93]. The largest increase in mean packet delay measured for ALR is approximately 10 ms and this value does not exceed the just-noticeable-difference (JND) threshold for delays.

Typical applications for desktop PCs can be categorized into three groups:

1. Data transfer – applications such as web browsing and file download
2. Playback streaming – applications such as video and audio playback
3. Real time streaming – applications such as videoconferencing and VoIP telephony

The use of ALR in Ethernet links to desktop PCs results result in an increased packet delay that is most probably not noticeable to end-users for these applications for both LAN and WAN end-to-end connections.

6.6 Extending ALR to More Than Two Link Data Rates

ALR can be extended to support more than two link rates. The current design of ALR is focused on 1 Gb/s Ethernet. For 1 Gb/s NICs (that also support 10 Mb/s and 100 Mb/s), it was found that 10 Mb/s and 100 Mb/s consume approximately the same power (see Section 2.1.1). Thus, switching between two link

rates – 100 Mb/s and 1 Gb/s was sufficient. For future 10 Gb/s (and even 100 Gb/s) link data rates, it may be beneficial to extend ALR to support switching between more than two link rates.

The existing MAC frame handshake mechanism already includes the requested link data rate (see Section 5.1.1). Thus, no changes are required in the ALR mechanism. The utilization-threshold policy can be changed to support switching between more than two link data rates. Two possible policy changes are:

1. Multiple high queue thresholds for an incremental increase in data rate. The link partner NICs at each end of the link would switch the link to a higher data rate upon a threshold crossing and successively greater data rates would have correspondingly greater threshold values.
2. Multiple utilization thresholds for switching down to optimum lower rate. Utilization sampling would be used to determine the present link utilization and multiple utilization thresholds would be used to determine the lowest data rate with sufficient capacity.

Implementing both policy changes would maximize energy savings at a possible cost in increased delay. Implementing only the second change and otherwise always switching to the highest rate on a high threshold crossing would minimize delay at a possible expense in reduced energy savings. A multiple data rate ALR policy based on the utilization-threshold policy with only the second policy change is shown in Figure 6.13 as a pseudocode description (an FSM description is not shown due to space limitations). A

```
if (qLen is greater than qHigh threshold) then
  handshake for a high link data rate
```

(a) Process for states LOW and MEDIUM

```
if (qLen is less than qLow threshold) then
  if (uBytes less than uThreshLow) then
    handshake for a low link data rate
  else if (uBytes less than uThreshMedium) then
    handshake for a medium data rate
```

(b) Process for state HIGH

```
if (qLen is less than qLow threshold) then
  if (uBytes less than uThreshLow) then
    handshake for a low link data rate
```

(c) Process for state MEDIUM

Figure 6.13. Pseudocode Description of a Multiple Data Rate Utilization-Threshold Policy

multiple data rate ALR policy that supports three data rates will have states HIGH, MEDIUM, and LOW. This policy requires two utilization thresholds – uThreshMedium and uThreshLow – to determine whether to switch to a low or medium data rate when in the HIGH state. This multiple threshold testing implements the support for multiple data rates. Figure 6.13 (a) shows the process that corresponds to the LOW and MEDIUM states, (b) for the HIGH state, and (c) an additional process for the MEDIUM state. The pseudocode description omits details on the handshake procedure. The process in (a) is triggered on receiving a frame and the processes in (b) and (c) are triggered when sending a frame.

Chapter 7: Reducing Direct Energy Use – Predictive Shutdown

Network interfaces consist of several components: physical layer modulation and demodulation circuits (PHY), medium access control circuits, input and output buffers, packet processing circuits, and in the case of a switch or router interface, circuits for packet classification and for finding the output port by routing table lookup. Energy savings in a network interface can be achieved if the complete network interface or its components can power off to a low power sleep state during the idle periods between packet arrivals. The key challenge is predicting the length of the inter-packet idle period accurately. A packet arrival to a network interface with sleeping components triggers a forced wake-up (to power up the sleeping components). Since wake-up time is non-negligible, forced wake-up results in added delay to packet service time and should therefore be minimized. Analysis of network traffic (in Table 7.1) shows that the majority of inter-packet idle time is concentrated on a small fraction of idle periods. Based on this analysis, a new predictive power management method that uses online quantile estimation to detect long idle periods is developed.

7.1 Quantile Estimation for Detecting Idle Periods

Network traffic is well known to be bursty and self similar [91]. As illustrated in Figure 7.1, inter-packet idle periods within a "burst" of packets are small compared to the idle periods between bursts. Self similarity can result in the majority of idle periods being small (and uniform), while the large part of the

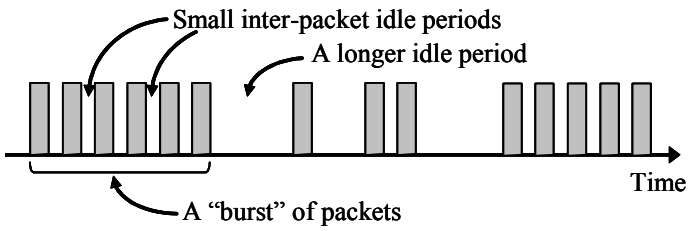


Figure 7.1. Inter-Packet Idle Periods Over Time

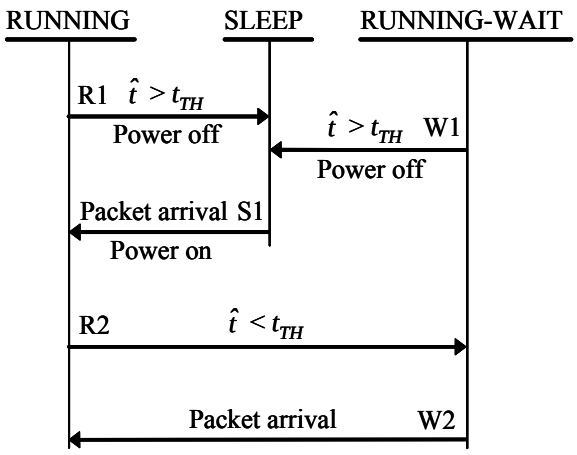


Figure 7.2. FSM for Hwang-Wu Method (Without Pre-Wakeup)

total idle time occurring in a few long idle periods. Detecting and exploiting long idle periods is a difficult challenge. Though numerous methods have been developed, as described in Section 3.1.2, many use complex computations. A method that can be implemented with insignificant additional power consuming circuitry is required.

The method proposed by Hwang and Wu [60] addresses this challenge. Figure 7.2 shows the finite state machine for this method. The states and transitions of the FSM representing the Hwang and Wu method are:

State RUNNING: The device is powered-on and may be processing a packet.

Transition R1: The prediction for the next idle period is greater than the threshold value. The device powers off to a low power sleep state.

Transition R2: The prediction for the next idle period is less than the threshold value. The device transits to the RUNNING-WAIT state.

State SLEEP**:** The device is in a low power sleep state and will wake-up on packet arrival.

Transition S1: A packet arrival occurs while in SLEEP state. The device powers on.

State RUNNING-WAIT**:** The device is powered-on and waiting to transit to either SLEEP state or RUNNING state. At the end of every threshold time period a new prediction is computed.

Transition W1: The prediction is greater than the threshold value and the device powers off to a low power sleep state.

Transition W2: A packet arrival occurs and the device transits to the RUNNING state.

In the method proposed by Hwang and Wu, exponentially weighted moving average (EWMA), or exponential smoothing, is used to predict the duration of the next idle period. The value of the next idle period is predicted using $\hat{t}_{i+1} = \hat{t}_i \alpha + t_i (1 - \alpha)$ where $\hat{t}_{i+1}$ is the new prediction, $\hat{t}_i$ is the previous prediction, t_i is the time of previous idle period, and α is a constant ($0 \le \alpha \le 1$). If the predicted idle period is greater than a previously calculated threshold (t_{TH}), the device will power off or enter a low power sleep state until an event occurs that triggers a forced wake up. If the predicted idle period is less than t_{TH}, the device stays powered-on in a RUNNING-WAIT state and a new prediction is made after every t_{TH} time interval until either an idle period greater than t_{TH} is predicted or wake-up event (a packet arrival) occurs. A prediction is limited to a maximum of C times the previous prediction. In order to minimize the additional delay caused by forced wake-up, this method has a feature named pre-wake-up. If this feature is enabled, if no wake-up event occurs by the end of the predicted idle period and the device is sleeping, the device will wake-up at the end of the predicted idle period.

The new ExpQ method improves upon the method of Hwang and Wu by using quantile estimation to detect and take advantage of long idle periods that occur after many small idle periods. A series of small idle periods biases the predictions made by the EWMA method towards small idle periods. Table 7.1 shows the sum of idle periods that exceed the given quantile as a percentage of the sum of all idle periods for the

Table 7.1. Sum of Idle Periods Exceeding Quantile

Quantile	Oct89Ext	Oct89Ext4	pAug89	pOct89
0.60	95.49%	97.77%	83.59%	94.88%
0.70	91.89	96.09	73.87	90.88
0.80	85.27	92.11	61.29	81.17
0.90	73.23	79.93	46.61	60. 7
0.95	62.34	66.30	36.45	44.23

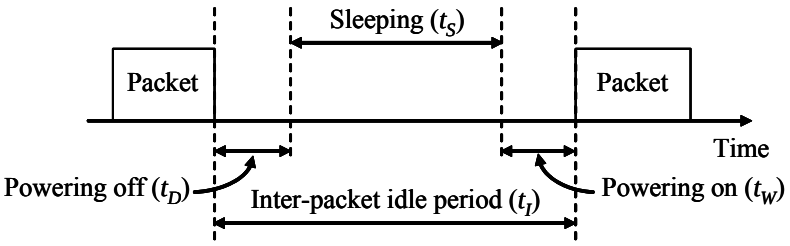


Figure 7.3. Powering Off During Inter-Packet Idle Periods

four well known Bellcore traces from [91]. This motivates the use of quantiles as means of dynamically finding thresholds for identifying difficult to predict long idle periods. The Bellcore traces, each of which contains a million packet arrival events, are from an Ethernet network at the Bellcore Morris Research and Engineering Center and are known to exhibit a large degree of self-similarity and to be representative of real network traffic. They have been extensively studied by the networking community and contain arrivals spanning over several hours and days.

Figure 7.3 illustrates an inter-packet idle period. For an inter-packet idle period (t_I) with non-zero powering on time (t_W) and non-zero powering off time (t_D), the possible sleep time (t_S) is $t_S = t_I - (t_W + t_D)$. If $(t_W + t_D) < t_I$, it is possible to power off the device and save energy with no increase in delay. Figure 7.4 shows the finite state machine (FSM) of the new EWMA plus quantile estimation method (named ExpQ). The states and transitions of the FSM representing the ExpQ method are:

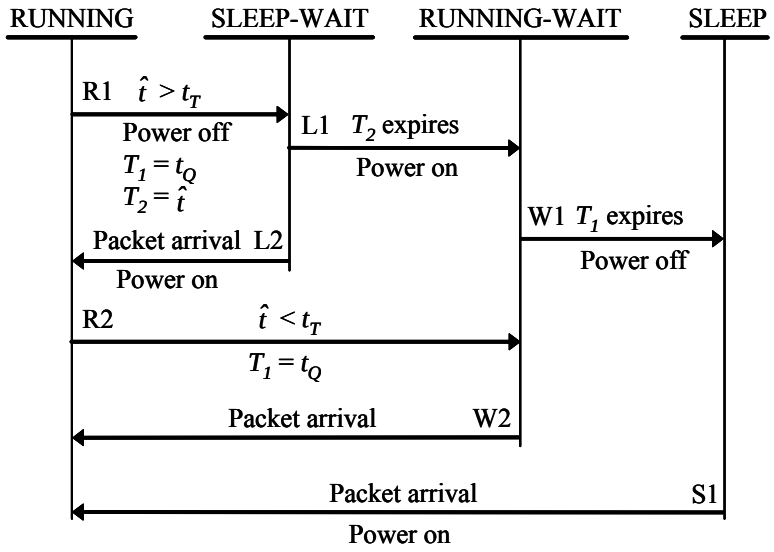


Figure 7.4. FSM for ExpQ Method

State RUNNING: The device is powered-on and may be processing a packet.

Transition R1: The prediction for the next idle period is greater than the threshold value. The device powers off to a low power sleep state. The timer T_1 is set to the estimated quantile time t_Q and the timer T_2 is set to the predicted idle period time $\hat{t}$.

Transition R2: The prediction for the next idle period is less than the threshold value. The device transits to the RUNNING-WAIT state. The timer T_1 is set to the estimated quantile time t_Q .

State SLEEP-WAIT: The device is in a low power sleep state and will wake-up on either packet arrival or expiry of timer T_2.

Transition L1: The timer T_2 expires. The device powers on and transits to the RUNNING-WAIT state.

Transition L2: Packet arrival occurs. The device powers on.

State RUNNING-WAIT: The device is powered-on and waiting to transit to either SLEEP state or RUNNING state.

Transition W1: The timer T_1 expires before a packet arrival occurs. A long idle period is assumed and the device powers off to a low power sleep state.

Transition W2: A packet arrival occurs and the device transits to the RUNNING state.

State SLEEP: The device is in a low power sleep state and will wake-up on packet arrival only.

Transition S1: A packet arrival occurs while in SLEEP state. The device powers on.

As mentioned previously, a prediction of the next idle period is $\hat{t}_{i+1} = \hat{t}_i \alpha + t_i (1-\alpha)$, where $\hat{t}_{i+1}$ is the new prediction, $\hat{t}_i$ is the previous prediction, t_i is the time of previous idle period, and α is a constant ($0 \le \alpha \le 1$). Threshold value t_T is defined as $t_T = t_W + t_D$. If $\hat{t} < t_T$ the device will set the timer T_1 to the estimated quantile time t_Q . If no packet arrival occurs before T_1 expires, the device will power off and sleep until a packet arrives and triggers a forced wake-up. If $\hat{t} > t_T$, the device sets timer T_1 to the

estimated quantile time t_Q and sets T_2 to the predicted idle period time $\hat{t}$ and then transits to the SLEEP-WAIT state. If no packet arrival occurs before T_2 expires, the device will power on and wait for T_1 to expire in the RUNNING-WAIT state. Though the ExpQ method contains four states compared to three states in the method of Hwang and Wu, the computational complexity is similar.

The estimated quantile time t_Q is calculated using the P2 algorithm proposed by Jain and Chlamtac [81]. This algorithm does not require the storing of observations and maintains five marker values that approximate the observations by a piece-wise parabolic curve. The marker values are dynamically updated (each update requires a finite number of steps regardless of the number of observations) as new observations are generated. Storage is only required for the marker values and for several intermediate variables. It is suggested in [81] that this algorithm is suitable for implementing in an integrated circuit.

7.2 Performance Evaluation of ExpQ Method

Trace-driven simulation was used to compare the ExpQ method to the Hwang and Wu method. Traffic input consisted of the Bellcore traces [91] and synthetic traces. The shortest mean idle period among the four Bellcore traces is 1.249 ms for the BC-pOct89 trace (this trace gives the worst performance compared to the other traces based on the *K* metric – defined in the next paragraph – when using the ExpQ method). The synthetic traces also approximate this value. The four synthetic traces are composed of the following:

1. Poisson arrivals (exponential idle periods), labeled as Exp in Table 7.3, with the rate of arrivals equal to 801 packets per second.
2. Interrupted Poisson Process arrivals, labeled as IPP in Table 7.3, with rate of arrivals set to 10,000 packets per second and with the on-to-off rate set to 12.5 per second and the off-to-on rate set to 1 per second.
3. Two traces of unbounded Pareto distributed arrivals with the Pareto parameters α = 1.2, β = 0.0002083 (labeled as Par12 in Table 7.3) and α = 1.5, β = 0.0004163 (labeled as Par15 in Table 7.3), respectively.

The performance measures were sleep time as a percentage of the total idle time and forced wake-ups as a percentage of the total number of packet arrivals in the trace. A new metric $K = L(1 - F)$, is defined,

where L is sleep time as a fraction of the total idle time and F is the number of forced wake-ups minus forced wake-ups that occur at the end of the longest 5% of idle periods as a fraction of the number of arrivals in the trace. The possible values for K range from 0 to 1, where a larger value indicates better performance. Because the energy saved by sleeping during a long idle period mitigates the cost of the forced wake-up, thus this metric ignores forced wake-ups that occur at the end of longest 5% of idle periods. This metric was also calculated without ignoring forced wake-ups that occur at the end of long idle periods and by ignoring forced wake-ups that occur at the end of the longest 5%, 10%, 15%, and 20% of idle periods. When taking the average value for all traces the ExpQ method outperformed the other methods for all cases. As a greater percentage of idle periods are ignored, the differences between the K values for all methods narrow.

The method proposed by Hwang and Wu was evaluated with both pre-wake-up disabled (labeled as HW in tables 7.2 and 7.3) and pre-wake-up enabled (labeled as HWP in tables 7.2 and 7.3). For the HW and HWP methods, t_{TH} = 1.5 ms and C = 2 (as given in [60]). For the ExpQ method, $t_W + t_D = 1.5$ ms and t_Q = 0.90 quantile. For all methods, the constant $\alpha = 0.5$. Tables 7.2 and 7.3 show the performance of the ExpQ method compared with the Hwang and Wu's HW and HWP methods for the traffic inputs described

Table 7.2. Performance Metrics for the Bellcore Traces

	Oct89Ext	Oct89Ext4	pAug89	pOct89
Sleep time as percentage of all idle time				
HW	99.99	99.78	85.91	64.96
HWP	21.21	25.12	50.98	36.97
ExpQ	53.93	55.62	59.66	54.99
Forced wake-ups as percentage of all arrivals				
HW	99.52	93.19	64.06	28.20
HWP	54.12	53.23	38.81	19.28
ExpQ	46.50	47.78	30.37	21.48
K metric				
HW	0.055	0.118	0.352	0.497
HWP	0.097	0.118	0.313	0.301
ExpQ	0.310	0.320	0.450	0.450

Table 7.3. Performance Metrics for Synthetic Traces

	Exp	IPP	Par12	Par15
Sleep time as percentage of all idle time				
HW	39.84	92.14	51.01	38.73
HWP	32.31	0.22	9.56	17.31
ExpQ	32.87	93.53	59.04	41.76
Forced wake-ups as percentage of all arrivals				
HW	39.83	0.39	13.73	22.17
HWP	32.27	0.26	11.29	18.63
ExpQ	32.84	10.69	19.01	23.65
K metric				
HW	0.259	0.919	0.460	0.321
HWP	0.226	0.002	0.086	0.145
ExpQ	0.240	0.880	0.510	0.340

above. On average, the ExpQ method spends 20% less time than HW sleeping, but has 35% less forced wake-ups. ExpQ spends 133% more time sleeping than the HWP method, but has 3% more forced wake-ups. Using the *K* metric, the ExpQ method outperforms both the HW and HWP methods for three out of four of the Bellcore traces (performance for trace BC-pOct89 is worse). For the synthetic traces modeled upon BC-pOct89, the ExpQ method has better performance on the Par12 and Par15 traces.

For the IPP synthetic trace, the sleep time for the HWP method is surprisingly low when compared with the other methods. The IPP trace consists of bursts of arrivals where the inter-arrival idle period during bursts is small compared to the idle periods between the bursts. Therefore, when a burst ends and a long idle period between bursts occurs, the HWP method would still predict a very small idle period and power-on at the predicted arrival time.

Possible savings in energy consumption of a device using these three methods are evaluated. An energy cost was assigned to powering on and powering off. Instead of the fixed 1.5 ms value used in the previous experiments, the threshold values t_{TH} and t_T were set to the minimum time necessary for the energy savings incurred during the powered-off time to exceed the energy cost of powering off and powering on. The power consumption when powered on was set to 1.0 W and the power consumption when powered off was set to 0.1 W. The energy cost of powering off or powering on was modeled as a

Table 7.4. Energy Consumption for the Bellcore Traces

	Oct89Ext	Oct89Ext4	pAug89	pOct89
1 ms power consumption spike				
HW	13.39%	14.48%	86.10%	98.13%
HWP	83.75	81.15	101.60	104.43
ExpQ	54.58	53.74	86.38	96.31
10 ms power consumption spike				
HW	39.21%	37.93%	100.71%	100.02%
HWP	96.75	94.28	100.74	100.02
ExpQ	53.07	49.08	100.94	100.20
100 ms power consumption spike				
HW	83.13%	95.10%	100.00%	100.00%
HWP	105.08	109.60	100.00	100.00
ExpQ	88.53	102.31	100.00	100.00

power consumption "spike" at a rate of 2.0 W and the duration of the spike was set to 1 ms, 10 ms and 100 ms, respectively. The values of the other parameters are unchanged from the previous experiments.

Table 7.4 shows the energy consumed when using each method as a percentage of the energy consumed when no power management methods are used for the Bellcore traces. Smaller values signify better results and 100% or greater indicates zero or negative energy savings. The trace BC-Oct89Ext has the largest mean inter-arrival time (122.68 ms) and the trace BC-pOct89 has a smallest mean inter-arrival time (1.25 ms) and correspondingly, the energy savings are better for the former trace as powered off times are greater. For the traces BC-pAug89 and BC-pOct89, for most inter-arrival times, the duration of the inter-arrival time is less than the threshold values t_{TH} and t_T. Thus, it is not possible save energy by powering off and the energy consumed whether using power management methods or not is approximately the same. The HW method has better energy savings in all but one case (BC-pOct89 with 1 ms spike where the ExpQ method is better) and the HWP method has the worst performance. However, as shown in Table 7.2, the HW method has a greater percentage of forced wake-ups compared to the other two methods and therefore the adverse effect on response time would be greater for the HW method.

7.3 Application of ExpQ Method to HAS-CP Architecture

Local area network (LAN) switches in use today are not capable of dynamically varying their power consumption as a function of utilization. In [53], Gupta, Grover, and Singh first introduced the concept of power management in LAN switches. The following power management methods were proposed for switch network interfaces: Simple Sleep, Hardware Assisted Sleep (HAS), and Hardware Assisted Buffered Sleep (HABS). With Simple Sleep, the Ethernet interface (e.g., as implemented in a line card) powers on periodically and any packets received when not powered-on are lost. With HAS, the interface remains powered off during inter-packet idle periods until a packet is detected. The packet causing the wake-up is lost and any packets arriving during the transition to powered-on are also lost. With HABS, input (RX) buffers in the switch line cards are assumed to be always powered-on and any packets received when sleeping are buffered. Figure 7.5 shows the Gupta et al. switch sleep model taxonomy including the new HAS-CP (Hardware Assisted Sleep – Cache Packets) architecture.

The HABS model is based upon the assumption that each switch interface is on a line card and is input buffered. A line card based switch contains several line cards in a chassis. Each line card typically contains several network interfaces, buffers and packet processing circuitry for packet classification, table lookup, checking for bit errors, updating packet header values, etc. Packets are forwarded from the line card into the switch fabric for forwarding to the output port. Figure 7.6 shows the HABS switch model. The physical

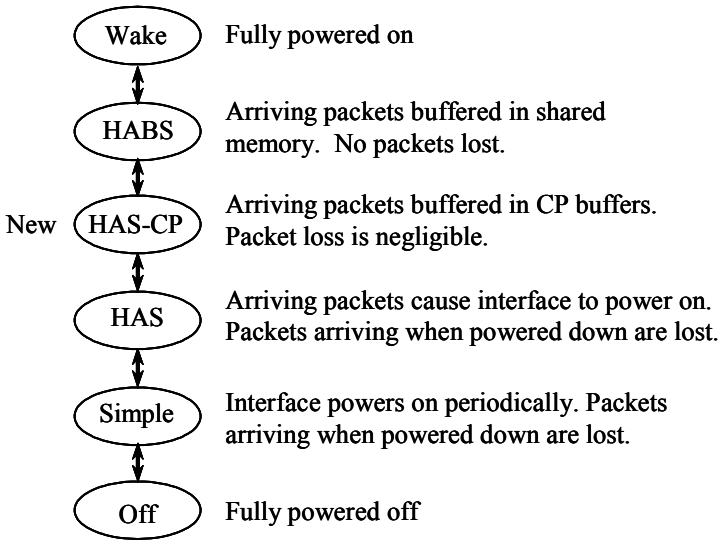


Figure 7.5. Switch Sleep Model Taxonomy

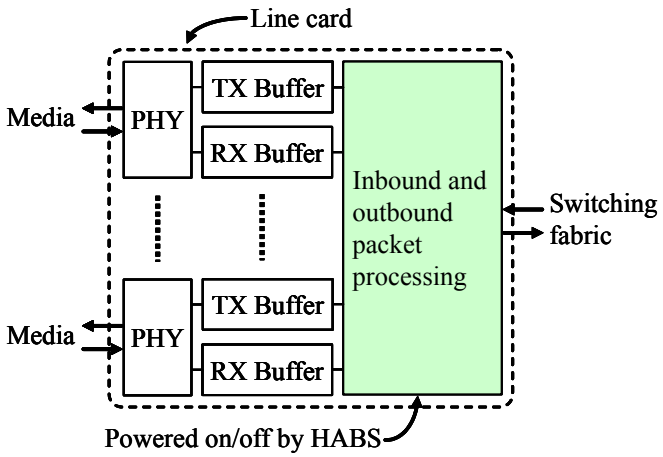


Figure 7.6. Powering Off Components in a Line Card Using HABS

layer interface circuits and the buffers are kept powered on at all times while the inbound and outbound packet processing circuits are assumed to be powered off during inter-packet idle periods. Low cost LAN switches used to connect desktop PCs to the network, however, are mostly based on shared memory architecture with a single central switch processor that handles packet classification, look-up, buffering, and forwarding. The core switch processor is connected to a number of Ethernet physical interfaces [123]. In shared memory switches, packets are sent to the shared memory after the destination port is determined through accessing forwarding tables. This makes it difficult to power off the interface components, which is necessary for HABS operation. Thus, the HABS model and assumptions in [53] cannot be easily applied to the shared memory switch, and as a result, an intermediate architecture (the HAS-CP architecture) becomes necessary.

In the HAS-CP architecture, a small cache memory (possibly a FIFO) is inserted between the Ethernet physical interface and the switch processor. Received packets pass through the CP buffers. During inter-packet idle periods the medium access controller, data bus controllers, forwarding table lookup circuits, and any components not necessary for receiving incoming packets in the switch processor can be powered off. Any packets received when components are powered off or are powering on are cached in the CP buffer. Power consumption incurred by the additional CP buffers needs to be considered however. The power consumption of a modern 32 MiB DDR2 memory chip is about 366 mW [94]. From this, it is calculated that the power consumption of a 4 MiB CP buffer would be approximately 50 mW. The switch power consumption could be reduced further. Assuming that the switch processor is designed to run at a sufficient

Table 7.5. Table of Parameters

Symbol	Parameter
$\hat{t}$	Predicted value for next idle period
t_I	Duration of idle period
t_D	Time to power off
t_W	Time to power on
t_S	Duration of low power "sleep" time
t_Q	Quantile estimate
t_{TH}	Threshold time for Hwang and Wu methods
t_T	Threshold time for ExpQ method
T_T	Total trace time
T_X	Total packet transmission time
T_S	Total low power "sleep" time
P_I	Power consumption when idle
P_P	Power consumption when processing packet
P_D	Power consumption when powering off
P_S	Power consumption when sleeping

speed to service requests coming through N_1 interfaces and if at a given time N_2 interfaces are powered off, the switching fabric could be run at sufficient speed to serve requests coming through ($N_1 - N_2$) interfaces, thus reducing power consumption.

Given an inter-packet idle period (t_I), with time for power off (t_D) and time for power on (t_W), the possible sleep time (t_S) is $t_S = t_I - (t_D + t_W)$. Using the conventions from [53], it is assumed that powering on and powering off consumes energy and that the power consumption during these transitions is greater than when powered off or powered on and idle. The power consumption when powering off is denoted by P_D, when powered off (sleeping) by P_S, when powered on but idle by P_I, when powered on and processing

a packet by P_P, and when powering on by P_W. If $(P_W t_W + P_D t_D + P_S t_S) < P_I t_I$, energy savings can be achieved by powering off. The duration of an idle period is not known in advance and the ExpQ method (with t_Q = 0.75 quantile) is used to predict the duration. The prediction of the next idle period is $\hat{t}_{i+1} = \hat{t}_i \alpha + t_i (1 - \alpha)$ where $\hat{t}_{i+1}$ is the new prediction, $\hat{t}_i$ is the previous prediction, t_i is the measured time of previous idle period, and α is a constant ($0 \le \alpha \le 1$). If the predicted idle period is of sufficient length to yield energy savings, the switch interface powers off until the next packet arrival event. Within a given time period, energy used by a switch interface (E_S) is given by

$$E_S = P_I (T_T - t_W N_W - t_D N_D - T_S - T_X) + P_W t_W N_W + P_D t_D N_D + P_S T_S + P_P T_X + P_B T_T$$

where N_W is the number of power-on transitions, N_D is the number of power-off transitions, T_T is the total time, T_S is the total sleep time, T_X is the total time spent processing packets and P_B is the power consumption of the buffer used by HAS-CP. The normal energy consumption is $E = P_I (T_T - T_X) + P_P T_X$. A list of parameters used is given in Table 7.5.

7.4 Evaluation of Possible Energy Savings

Trace-driven simulation was used to evaluate the energy savings of the HAS-CP architecture. Traffic input consisted of the well known Bellcore traces [91] and synthetic traces. The mean idle periods for the four Bellcore traces are in decreasing order, 122.6 ms (BC-Oct89Ext), 75.7 ms (BC-Oct89Ext4), 2.7 ms (BC-pAug89), and 1.2 ms (BC-pOct89). Two synthetic traces were generated with the mean idle period of the BC-Oct89Ext Bellcore trace. The two synthetic traces consist of Poisson arrivals and unbounded Pareto distributed arrivals with α = 1.4 and β = 0.03505. All traces contain approximately 1 million packet arrival events. Similar to [53], the performance measure used was the ratio of energy consumed when HAS-CP was not used (E) to energy consumed when HAS-CP was used (E_S) (i.e., E/E_S). A value greater than 1 indicates energy savings and the higher the value the greater the energy savings. The energy savings achieved when using the HAS-CP architecture is compared with an "oracle". The oracle has complete knowledge of the future and only powers off during idle periods when the energy cost can be amortized, and also always powers on before a packet arrives. It is noted (though not shown here) that this metric does

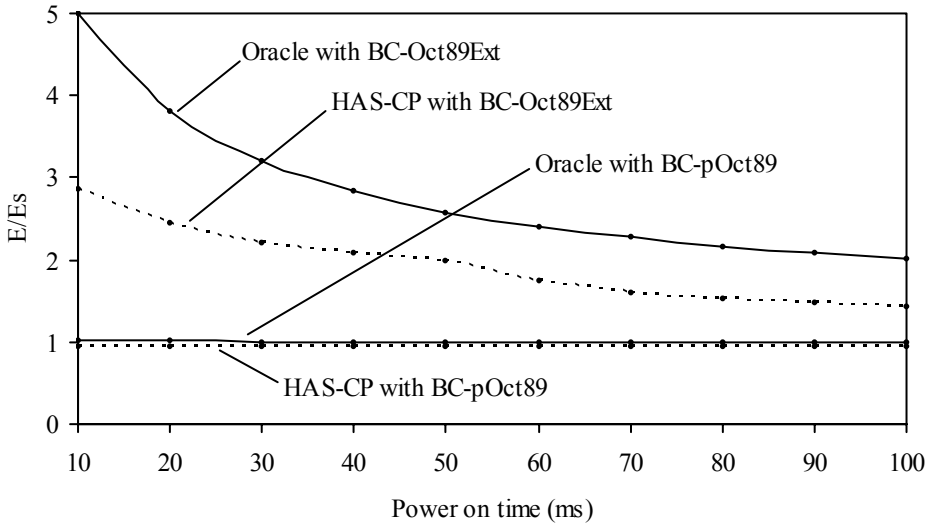


Figure 7.7. Performance of HAS-CP with Bellcore Traces

not consider delays caused by forced wake-ups and that the HW method, which was described previously and shown to be inferior to the ExpQ method when considering both sleep time and delay, shows higher performance in this metric due to the greater sleep time.

The following values from [53] were used in the simulations: P_I = 1 W, P_S = 100 mW, P_W = 2 W, P_P = 2 W and P_D = 0 W. Furthermore, P_B = 50 mW, t_D = 0 s, α = 0.875, and t_W ranges from 10 ms to 100 ms in increments of 10 ms. α = 0.875 was found to give consistently good results and is used for this reason. Figure 7.7 shows the results when using the BC-Oct89Ext and BC-pOct89 traces. These two traces give the best and worst energy savings of the four Bellcore traces, respectively. HAS-CP with the

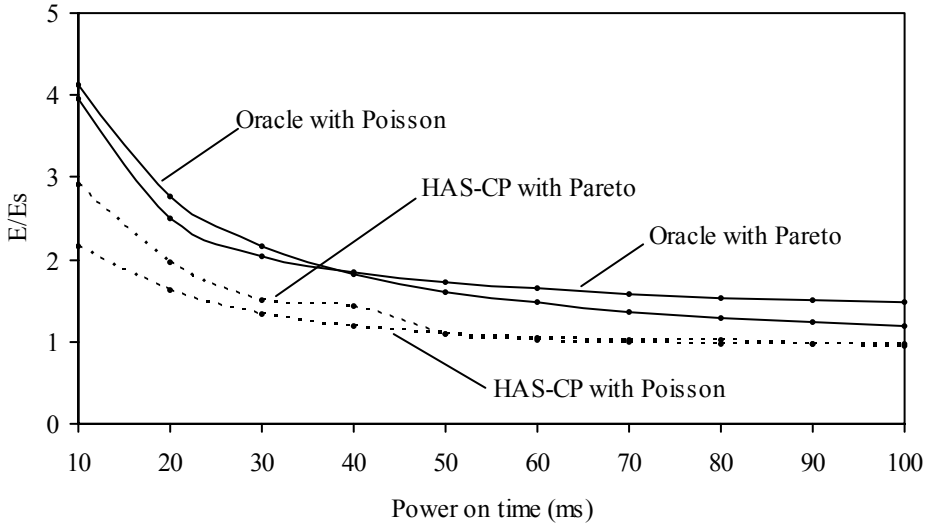


Figure 7.8. Performance of HAS-CP with Synthetic Traces

BC-pOct89 trace has a small negative energy savings due to the extra power consumption of the memory buffer. No opportunity for powering off exists with the latter trace due to small inter-packet idle periods. Figure 7.8 gives the results for the synthetic traces and in all cases, energy savings greater than 50% appear feasible for a power on time of 10 ms. It can be seen that as the power on time increases, less idle periods of sufficient length exist and the possible energy savings decreases.

Chapter 8: Reducing Induced Energy Use – Initial Work

Network induced energy consumption is caused when an otherwise idle network host is prevented from powering off or transitioning to a low-power consumption operating state in order to maintain network connectivity and be responsive to network events. Routine network protocol messages that maintain a desktop PC's "network presence" and network accessibility need to be responded to, and it is not possible to pre-determine when network access to the resources within the desktop PC will be needed. Furthermore, client/server applications maintain long-lived TCP connections (e.g., telnet, SSH). If the client desktop PC transits to a low-power consumption state and does not respond to TCP keep-alive messages, the connection can be dropped leading to loss of application state associated with the TCP connection. This chapter describes initial work in reducing induced power consumption of network hosts.

In this chapter, Section 8.1 details a characterization of the traffic on the Ethernet network link to an idle desktop PC with respect to protocol proxying. Proxying, introduced in Section 3.1.3, uses a low cost and low power consumption device to represent a high power consumption device, thus enabling the high power consumption device to reduce its power consumption by transiting to a low power consumption sleep state. If the resources in the high power consumption device (e.g., a desktop PC) are required, the proxy sends a wake-up signal. Within the context of protocol proxying for a desktop PC, the proxy is expected to do the following:

1. Proxy for the desktop PC for simple routine protocol events
2. Wake up the desktop PC when its resources are needed
3. Possibly exchange state information with the operating system of the PC

The proxy can be an enhanced Ethernet NIC with additional processing and storage capacity or a separate device connected to the network and capable of proxying for multiple desktop PC's. Figure 8.1 illustrates a proxy for a desktop PC in an Ethernet NIC. In all cases, the initial cost must be low and power consumption must be minimized. Therefore, the key challenge is to determine which functions can be

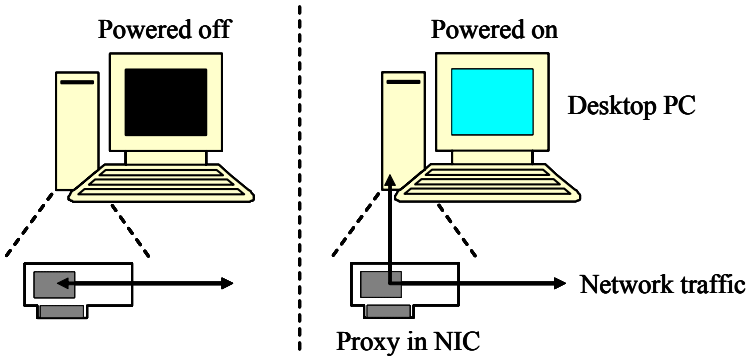


Figure 8.1. Desktop PC Proxy in an Augmented NIC

transparently (or nearly transparently) off-loaded to the proxy with reduced capabilities compared to a desktop PC. Protocols and events that can be proxied without a host wake-up are identified, and improved wake-up semantics that prevent spurious wake-ups are described in sub-sections of Section 8.1. In Section 8.2, an emulation based prototype of a proxy for a Hyper Text Transfer Protocol (HTTP) server is presented. It is experimentally shown that with proxying, the "network presence" of a sleeping network host can be maintained and that resources within the host (i.e., web pages) can be accessed when required. In Section 8.3, split TCP connection that permits the operating system to disconnect and reconnect TCP connections without application awareness is covered.

8.1 Protocol-Level Characterization of Desktop Traffic

Developing a protocol proxy for responding to routine network traffic without powering on the host desktop PC as well as developing more selective wake-up methods requires knowledge of packets received on the links. An idle desktop PC receives packets on its link at all times. With respect to proxying and host system wake-up, packets can be categorized as:

1. *No response required:* Packets that require no actions or response and are discarded by the protocol stack in the operating system. This includes broadcast bridging and routing protocol packets and this also includes "hacker" traffic such as port scans.
2. *Minimal response required:* Packets that require minimal action or a response that a proxy could handle. This includes ARP and ICMP echo request packets.

3. *Wake-up required:* Packets that require an operating system or application-level response. This includes TCP SYN packets for connection requests to applications with open ports and SNMP GET requests.

In addition, some network protocols generate packets from a client. For example, DHCP lease renewal requests are generated locally from a PC or other device holding a DHCP granted IP address. Packets of the first two types (no response required and minimal response required) can be called "network chatter". Table 8.1 summarizes 296,387 packets received in 12 hours and 40 minutes by an idle Dell OptiPlex GX270 desktop PC running the Microsoft Windows XP operating system connected to the University of South Florida network– this is just over 6 packets per second. This was collected on an August 2004 weekday during nighttime. ARP packets constitute the slight majority of received packets with Universal Plug and Play, routing, and bridge protocol packets comprising about 30% of the total. The packets summarized in Table 8.1 were captured using a network packet trace capturing tool running on another PC on a shared Ethernet hub. Figure 8.2 illustrates the test bed used to capture network traces. The NIC was configured to wake-up the PC upon receiving either a WOL packet or a directed packet. The system was configured for 10 minutes of inactivity time before transitioning into the Microsoft Windows XP standby (low-power sleep) state. PCs and most other electrical devices consume more energy during the wake-up transition – a wake-up power spike – than during steady state operation. Thus, unnecessary wake-ups waste energy in two ways: by the wake-up power spike as well as during the steady-state on time before re-entering the sleep state.

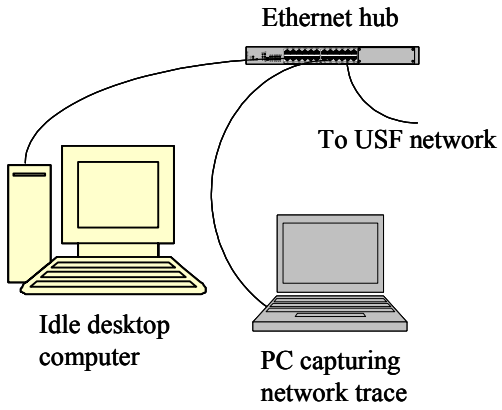


Figure 8.2. Test Bed for Capturing Network Traces

Table 8.1. Breakdown of Packets Received on Link to Idle PC

Protocol	% in trace	Discard	Proxy	Wake-up	Comments
ARP Request and Reply	52.50 %	Most	Yes	No	Requests for IP address answered by proxy
Universal Plug & Play	6.50	Most	Yes	No	Discovery messages answered by proxy
BRIDGE "Hello"	7.80	All	No	No	Can be discarded by proxy
Cisco Discovery	6.90	All	No	No	Can be discarded by proxy
NetBIOS Datagram	4.40	Some	No	Possible	Directed packets need a PC wake-up
NetBIOS Name Service	3.60	Some	Yes	Possible	Directed packets need a PC wake-up
Banyan System	1.80	All	No	No	Can be discarded by proxy
OSPF	1.60	All	No	No	Can be discarded by proxy
DHCP	1.20	All	No	No	Can be discarded by proxy
IP Multicasts	1.00	All	No	No	Can be discarded by proxy
RIP	0.50	All	No	No	Can be discarded by proxy
SMB	0.40	Some	No	Possible	Directed packets needs a PC wake-up
NetBEUI	0.31	All	No	No	Can be discarded (deprecated protocol)
Unknown port scans	0.30	All	No	No	Contains TCP SYNs (unneeded wakeups)
BOOTP	0.25	All	No	No	Can be discarded by proxy
NTP	0.20	All	No	No	Can be discarded by proxy
NetBIOS Session service	0.12	Some	No	Possible	Directed packets need a PC wake-up
ICMP (including echo)	0.08	Some	Yes	No	Echo requests answered by the proxy
DEC Remote Console	0.08	All	No	No	Can be discarded by proxy
SNMP	0.06	Some	No	Possible	Can be discarded unless SNMP agent
ISAKMP	0.04	All	No	No	Can be discarded by proxy
X Display Manager	0.02	All	No	No	Can be discarded by proxy

8.1.1 Types of Traffic That Can Be Proxied

The protocol proxy could drop all packets categorized as "No response required" and respond to packets categorized as "Minimal response required" without waking up the host PC. The proxy would filter packets that require no response, reply to packets that require a minimal response, and only wake-up the system for packets requiring a non-trivial response. Of the packet types shown in Table 8.1, responding to Address Resolution Protocol (ARP) packets, Internet Control Message Protocol (ICMP) packets, Universal Plug and Play resource discovery packets, and NetBIOS Name Service packets is required to maintain the "network presence" of the host and can be trivially responded to by a proxy.

The proxy could also generate packets for simple protocols such as DHCP based on a timer interrupt. This response to an internal event is similar in complexity to a minimal response to an external event such as a received packet. Interfaces need to be defined to pass state information from a PC to its proxy. Possibly, existing ACPI interfaces could be extended to achieve this. This state information would include the IP addresses of the host, Microsoft Windows local area networking workgroup name and which TCP ports are open for listening.

With proxying, fully 91% of the nearly 300,000 packets summarized in Table 8.1 would be filtered out or trivially responded to by a proxy. The remaining 9% of packets include TCP SYN packets, most for non-existent open ports, and Microsoft Windows workgroup file sharing protocol traffic, such as NetBIOS Datagram and Session Service and SMB. The SYN packets require a response from the system only when there is a running application with an open port. If the proxied host does not participate in Microsoft Windows workgroup file sharing, that traffic can also be discarded.

8.1.2 Improved Wake-Up Semantics

As described previously in Chapter 4, existing wake-up methods rely on a special WOL format packet, trigger on the appearance of the host IP address, or match and trigger on pre-programmed patterns found in a received packet. Wake-up on IP address match often results in many unnecessary wake-ups with the system being powered on unnecessarily. A more selective and intelligent wake-up would reduce the amount of time a system is awake by preventing spurious wake-ups. The traced PC of Table 8.1 was awake for 2 hours and 40 minutes (21%) of the 12 hour 40 minute trace period due to wake-ups caused by incoming TCP SYN packets (after each wake-up the PC would stay awake for approximately 2 minutes). However, apart from Microsoft Windows workgroup networking ports, there were no open TCP ports (application-level listens).

Of the wake-up causing TCP SYN packets recorded in the trace, 53% were for open local area networking ports and the rest were simply port scan traffic. Adding knowledge of the open TCP port numbers to triggering wake-ups on TCP SYN packets would reduce the observed system's powered-on

time to 1 hour 24 minutes (11% of total trace time). Open TCP ports are created by network applications when they issue a listen command for incoming TCP connection requests.

8.2 Design and Evaluation of Prototype Proxy for an HTTP Server

A prototype proxy is designed, developed, and experimentally evaluated for an HTTP server. Figure 8.3 shows the control flow for a packet received by the proxy, i.e., the control flow for a program or an application specific integrated circuit (ASIC) operating within the proxy for processing a packet received on the network interface of the proxy. If the host PC is awake, the proxy operates normally (i.e., packets are passed to the PC as shown in first three lines of Figure 8.3). If the proxied desktop PC is sleeping, packets are examined and:

1. Discarded if no action is required.
2. Handled by the proxy if a routine reply is required.
3. Triggers a wake-up signal to the desktop PC

If it is decided to wake-up the desktop PC, the packet is passed on to the PC after wake-up. By observing specific bytes within the packet, it is possible to determine the protocol and the purpose of the packet.

In order to experimentally evaluate protocol proxying, a second PC is used to emulate a proxy. Figure 8.4 shows a photograph of the proxy test bed with a Test System and Proxy Emulator. The Test System is the power managed Dell desktop PC with Microsoft Windows XP and an Ethernet NIC that supports WOL.

```
receive a packet
if (system is awake) then
  pass packet to computer
else
  if (packet requires no action) then
    discard packet
  else
    if (“simple” request) then
      handle locally on the adapter
      discard packet
    else
      trigger computer wake-up
      wait for computer to wake-up
      pass packet to computer
```

Figure 8.3. Control Flow for Packet Proxy

Figure 8.4. HTTP Proxy Emulator and Test System

The Proxy Emulator is another Dell PC that emulates the HTTP proxy. The Test System and Proxy Emulator are connected together through an Ethernet hub.

The Test System has its power management features enabled and automatically goes into Microsoft Windows XP standby state after a period of inactivity (set to 120 seconds). When the Test System awakes, the Proxy Emulator checks whether the Test System is powered on or sleeping by sending ARP Request packets at a rate of one every five seconds to the Test System and waiting for replies. If three consecutive ARP Requests do not elicit a reply from the Test System, it is assumed that the Test System has transitioned to a sleep state. The Proxy Emulator then broadcasts an ARP Reply packet linking its Ethernet MAC address to the Test System's IP address, which causes the computers and network devices on the laboratory LAN and USF backbone to update their ARP tables with the new address association. Subsequently, any data packet sent to the Test System's IP address will be received by the Proxy Emulator. Figure 8.5 illustrates the operating modes for the Test System and the Proxy Emulator.

The Proxy Emulator replies to any ARP Request broadcasts for the Test System's MAC address with its own MAC address, ensuring that any data packets for the Test System will be redirected to it. As a demonstration, the capability to respond to ICMP Echo requests is encoded. The Proxy Emulator will receive any ping echo packets sent to the Test System's IP address and will reply with an ICMP Echo reply

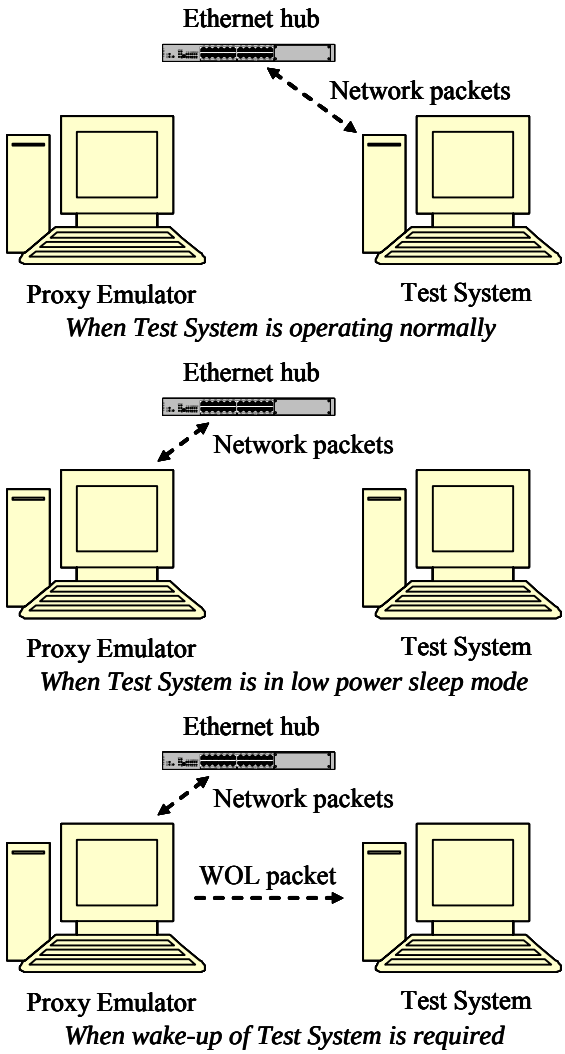


Figure 8.5. Operating Modes for Proxy Emulator and Test System

packet that appears to come from the Test System. Since the Proxy Emulator is responding on behalf of the sleeping Test System, the Test System does not need to wake-up.

An HTTP server is installed and is operating in the Test System. If the redirected packet received by the Proxy Emulator is a TCP SYN packet for the HTTP port, the emulator will transmit a WOL packet to the Test System to wake-up the computer. At this point, the proxy emulator will again start probing the Test System to determine whether it is powered-on or in sleep state. The Microsoft Windows XP Test System used for testing can be woken-up from the Microsoft Windows XP standby state to a powered on state in approximately 4 seconds by a WOL packet. It was found that this wake-up time was sufficient for

an HTTP browser sending a TCP SYN request not to time out. Once the Test System is powered on, the HTTP request is serviced by the HTTP server within.

8.3 Splitting TCP Connections to Enable Power-Off

Many applications maintain a persistent TCP connection between a client and server. For example, a client/server database application may leave TCP connections open at all times. Telnet, SSH, and other shared resource applications also require permanent connections. Even when the TCP connection is idle – that is, no application data is flowing on the connection – both end-points must generate and respond to periodic TCP-generated keep-alive messages that occur at least once every two hours. Many applications also generate application-level keep-alive messages that must be responded to if the application is to maintain its TCP connection. Thus, the key challenge is how to respond to keep-alive messages without requiring the full resources of a client or server. In addition, there must be a means to quickly and fully resume the TCP connection when application data needs to flow either from client to server or server to client. There are three possible ways to address persistent TCP connections:

1. Rewrite applications at the client side and/or server side that normally maintain persistent TCP connections to only be connected while actively requiring data transfer. HTTP 1.0 uses this approach.
2. Use proxying at the client to answer keep-alive messages and wake-up the client PC on resumption of data transfer within a connection
3. Split the TCP connection such that though applications "see" a connection, the operating system has closed the connection at the TCP layer.

The first method is outside the scope of this dissertation and would require the rewriting off all applications that use persistent TCP connections. The second method based on proxying would require no changes in the server. The widely used method for sending TCP keep-alive messages is to send packets with connection sequence numbers that have already been acknowledged by the receiver. Upon receiving these already acknowledged sequence numbers, the receiver would then transmit a duplicate acknowledgement for the last data received from the sender. Implementing this in a proxy would require

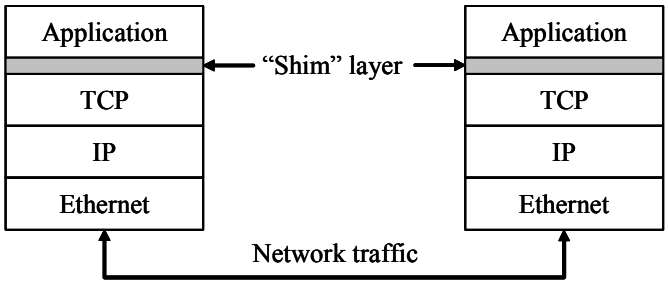


Figure 8.6. A "Shim" Layer for Power Management

the proxy to be aware of the connection sequence numbers and acknowledgements for every established TCP connection. This would be non-trivial for a proxy to implement and does not consider application-level keep-alive messages, which cannot be proxied. The third method based on splitting a TCP connection within the client and server would require modifications to the operating system in both the server and the client, but no changes to the way TCP works or to the multitude of applications that use TCP. This last method is explored in this dissertation through prototyping and experimental evaluation.

To avoid changes to applications and the TCP protocol implementation, a TCP connection is "split" by the addition of a "shim" layer between the sockets interface and the application as illustrated in Figure 8.6. This shim presents a sockets interface to the application (thus, the application does not change) and uses the existing sockets layer of the TCP software implementation. Today, when a client powers-off, the connection is dropped and the server cleans up all resources (state) associated with the connection. The application is then notified that the connection is closed resulting in an error if the application expects a persistent connection. The shim layer enables applications to see an established connection at all times. Figure 8.7 shows the semantics of a split TCP connection. It can be observed that the shim layer's functionality is used when power management events are triggered. The following is accomplished by the shim layer:

1. When the client at one end of the connection transits to low-power sleep state, the shim layer informs the opposing shim layer in the server to drop the TCP connection.
2. When the client transits back to powered on state (e.g., due to keyboard or mouse activity), the shim layer reestablishes the previously dropped TCP connection by informing the opposite shim layer which socket connection to connect to.

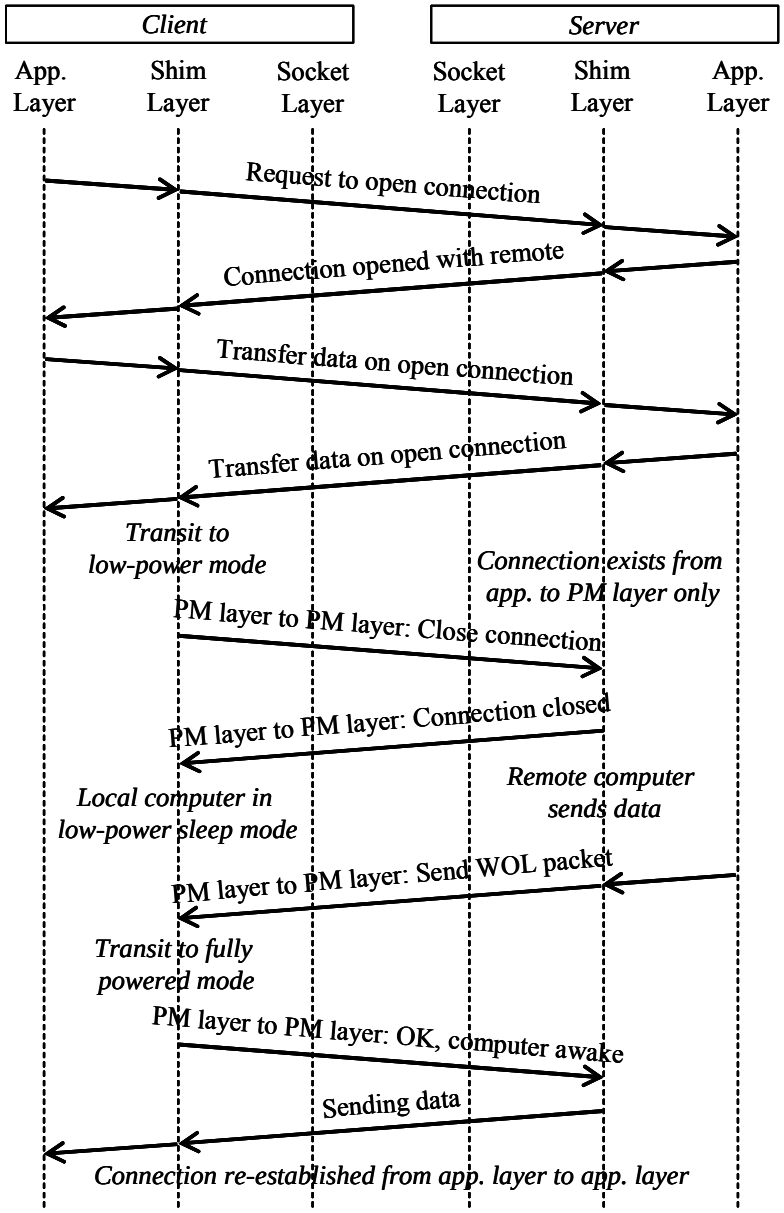


Figure 8.7. Spilt TCP Connection

3. If a server application wishes to send data to a client that is in low-power sleep state, the shim layer first wakes-up the sleeping client (e.g., using one of the wake-up mechanisms already described) and then reestablishes the TCP connection.

To evaluate the feasibility of this method, a prototype was developed using a telnet client and a telnet server running on Microsoft Windows XP. Since it is impossible to modify the Windows socket library source code, the shim layer was implemented in the application space and compiled in with the application source code. Specifically, the socket library function calls in the applications were renamed (e.g., in the

telnet client source code, the `connect` function call was renamed `txcConnect` and the `send` function call was renamed `txcSend`) and the shim layer functionality was implemented within these new functions. All calls to the socket library functions would "pass-through" the shim layer functions. By modifying the source code in this manner, it is possible to include the shim layer functionality without access to the operating system source code.

When initially setting up socket connections, all parameters in the function calls would be recorded in a data structure within the shim layer. This data would then be used for subsequent connection reestablishments. For example, to send data to the server, the telnet client would call the function `txcSend` with the socket identifier of the initial connection. Within the `txcSend` function, the underlying connection's availability is checked and if the connection to the server has previously been dropped, a new connection is established to the shim layer in the server. Then, the `send` function would be called from within the `txcSend` function with the new TCP connection's socket id in order to send the data to the server. Obviously, if the initial TCP connection was available, then there would be no need for connection reestablishment. All socket function calls and Windows sockets architecture functions were renamed and modified similarly.

It was not possible to trap and respond to the Microsoft Windows WM_POWERBROADCAST [95] power management event notifications without major changes to the telnet client and server applications. Therefore, manual event triggers were used in order to test the prototype shim. Timers were used to drop the existing connection between the telnet client and the server after a period of inactivity. This was observed and confirmed using the TCPView [124] utility program. Then the desktop PC running the telnet client was manually set to the Microsoft Windows XP standby state. After an extended time period in the standby state the PC was powered on manually and a telnet command was keyed in at the telnet client window. The connection reestablishment was observed using the TCPView utility program. It was observed that it was possible to power off the desktop PC running the telnet client to the low power consumption Microsoft Windows XP standby state and power on, causing the underlying connections to be dropped and reestablished without losing the telnet session (which exists at the application layer) between the telnet client and the telnet server.

Therefore, split TCP connections provide a solution for powering off desktop PCs even though the applications running in the PC has connections open to remote hosts without having to modify all applications installed in the PC. Legacy applications are updated infrequently. It can be years before updated versions are released and not all applications are supplied by the same vendor and updated at the same time. However, the operating system in all the desktop PCs in an organization is usually the same and is updated at the same time. Therefore, the split TCP connection provides a "quick fix" for making legacy applications more power management friendly.

Chapter 9: Conclusions and Future Research Directions

The Internet and the devices that connect to it consume over $6 billion annually in electricity in the U.S. alone. Much of this energy powers idle links, switches, and network-connected hosts and is thus wasted. In this dissertation, the direct energy use of Ethernet links and induced energy use of network-connected hosts has been addressed. New methods to reduce energy waste have been investigated.

This dissertation proposed, designed, and evaluated a new method called Adaptive Link Rate (ALR) for matching Ethernet link data rate with link utilization. This is the first ever investigation into energy efficiency of Ethernet links. The ALR method – in the context of a control policy to determine when to transition link data rate – was studied using Markov models and simulation. A simple dual-threshold ALR policy was shown to result in unnecessary oscillation of link data rate for smooth (Poisson) traffic. This oscillation caused excessive packet delay. An improved policy that explicitly measures link utilization was developed and evaluated. This improved policy does not cause unnecessary oscillation of link data rate. This dissertation also investigated detection of long idle periods, characterization of network traffic for identifying relevance for proxying, prototyping of a proxy for a Web server, and development of a "shim" layer for splitting TCP connections to enable inactive hosts with idle TCP connections to go to sleep.

The expected benefits from this research are:

1. A large reduction of energy used by Ethernet links is now achievable with ALR. This was shown with both analytical and simulation modeling. The engineering trade-offs of energy saved versus increased packet delay were investigated. Additional packet delay was shown to be minor (and imperceptible to a user in virtually all cases of interest) compared to the expected energy savings.
2. Modeling of state-dependent service rate queues with service rate change at the end of a service interval is now possible for multiple thresholds. The developed Markov model was applied to the evaluation of the ALR dual-threshold policy and made possible a detailed understanding of link

rate oscillation. This new Markov model also has benefits to modeling of computer networks and other queueing systems where the only realistic case is service rate change at service completion.

3. As part of the simulation modeling of ALR a method and implementation for generating synthetic network traffic was developed. This synthetic traffic generated uses Pareto-distributed bursts and was shown to generate synthetic traffic that very closely matches the characteristics of actual (traced) traffic. This synthetic traffic generator can be used by other researchers who simulate computer networks and need to drive their models with realistic traffic.
4. A new method for detecting long idle periods for powering off network interfaces using quantile estimation has been developed. This new method does not require complex off-line calculations. When applied to power management, the resulting sleep time is less than that achieved by comparable methods (such as the method proposed by Hwang and Wu [60]); however the number of forced wakeups is greatly reduced.
5. A demonstration of feasibility of proxying and splitting of TCP connections has been achieved. A detailed characterization of traffic on an Ethernet link to an idle desktop PC showed that the majority of packets can be disregarded or trivially responded to (i.e., proxied by a low-power device). A prototype proxy for a web server was developed and demonstrated. This contribution shows proxying and split TCP connections are feasible methods for reducing induced energy use.

An IEEE 802.3 Energy Efficient Ethernet study group has been formed with the intention of standardizing ALR. The research described in this dissertation has directly contributed to the formation of this study group (in conjunction with researchers from Lawrence Berkeley National Laboratory). It is thus very likely that ALR will become a standard and will achieve industry implementation and widespread deployment. Annual savings in electricity costs of over $200 million is expected after full deployment of ALR for Ethernet (Table 2.3 on page 15). Successful deployment of methods that reduce induced power consumption is expected to result in savings of $2.5 billion annually (Table 2.3 on page 15). These methods are still in the formative stages, so these expected savings will require more research and thus may take many years to be achieved.

It was shown in Chapter 6 that the power consumption of an ALR-capable edge level switch can be reduced by over 20% at typical Ethernet desktop link utilization levels. Further significant energy savings

can be expected by reducing the power consumption of switch internal components The key next step for this research in energy efficient networks should be to investigate how ALR can enable much greater energy saving within a switch, including in the switch line card, switch main processor and routing tables, and switching fabric. An Ethernet switch with ALR-capable switch ports would have a certain number of ports operating at a lower data rate at any given time. This knowledge can be utilized to operate the switch main processor, routing table lookup circuitry, and packet processing circuitry (e.g., in the switching fabric) at lower clock frequencies thereby saving energy.

This dissertation has been the first work in energy efficiency of the Internet and the hosts that connect to it. Many new questions have arisen as a result of this research. Future research directions include:

1. ALR policies based upon higher-layer information. The ALR policies explored in this dissertation are based on link-layer information only. Can application and transport layer information (e.g., type of application currently running in PC, current TCP congestion window size, etc.) be used to determine when high or low link data rates should be used?
2. Network traffic shaping at network hosts for energy efficiency. Can non-urgent or non-real time network traffic (e.g. automated backups of desktop PCs, SMTP traffic, etc) be transmitted at lower rates so that network links can be operated at energy saving low data rates? What is the trade-off in energy savings at the network links versus longer operating times for network hosts?
3. Sleeping when not actively used while maintaining resource accessibility for desktop PCs. In this dissertation it was shown that, through proxying, it is possible for a web server to be powered off while maintaining accessibility by responding to a limited set of network protocols. Can a general purpose proxy be designed and developed that is capable of proxying for the numerous applications and network protocols in use in a LAN environment? What is the trade-off in capability and computing resources (e.g., processor, memory, etc.) necessary to achieve this? Must “always on” imply that a network host must always be powered on?

References

[1] ACPI – Advanced Configuration & Power Interface Specification.
URL: http://www.acpi.info/spec30.htm.

[2] "Adaptive Power Management for Mobile Hard Drives," Technical report, IBM Corporation, Storage Systems Division, April 1999.
URL: http://www.almaden.ibm.com/almaden/pbwhitepaper.pdf.

[3] B. Adelstein, T. Lee, and S. Ellis, "Head Tracking Latency in Virtual Environments: Psychophysics and a Model," *Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society*, pp. 2083-2087, 2003.

[4] S. Agrawal and S. Singh, "An Experimental Study of TCP's Energy Consumption over a Wireless Link," *Proceedings of the 4th European Personal Mobile Communications Conference*, February 2001.

[5] K. Akkaya and M. Younis, "A Survey of Routing Protocols in Wireless Sensor Networks," *Elsevier Ad Hoc Networks Journal*, Vol. 3, No. 3, pp. 325-349, 2005.

[6] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, Vol. 40, No. 8, pp. 102-114, August 2002.

[7] L. Alvisi, T. Bressoud, A. El-Khashab, K. Marzullo, and D. Zagorodnov, "Wrapping Server-side TCP to Mask Connection Failures," *Proceedings of INFOCOM 2001*, pp. 329-337, April 2001.

[8] J. Al-Karaki and A. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey," *IEEE Wireless Communications*, Vol. 11, No. 6, pp. 6-28, December 2004.

[9] AMD Magic Packet Technology.
URL: http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/20213.pdf.

[10] AMD PowerNow! Technology.
URL: http://www.amd.com/us-en/assets/content_type/DownloadableAssets/Power_Now2.pdf.

[11] G. Anastasi, M. Conti, and W. Lapenna, "A Power Saving Network Architecture for Accessing the Internet from Mobile Networks: Design, Implementation and Measurements," *Proceedings of the 2nd International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; and Mobile and Wireless Communications*, p.240-251, May 2002.

[12] APM – Advanced Power Management V. 1.2 specification.
URL: http://www.microsoft.com/whdc/archive/amp_12.mspx.

[13] A. Bakre and B. Badrinath, "Implementation and Performance Evaluation of Indirect TCP," *IEEE Transactions on Computers*, Vol. 46, No. 3, pp. 206-278, March 1997.

[14] F. Bellosa, "The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems," *Proceedings of 9th ACM SIGOPS European Workshop*, pp. 37-42, September 2000.

[15] F. Bellosa, S. Kellner, M. Waitz, and A. Weissel, "Event-Driven Energy Accounting for Dynamic Thermal Management," *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP'03)*, September 2003.

[16] L. Benini, A. Bogliolo, G. A. Paleologo, and G. De Micheli, "Policy Optimization for Dynamic Power Management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 6, pp. 813-833, June 1999.

[17] Broadcom BCM5701 10/100/1000BASE-T controller product brief.
URL: http://www.broadcom.com/collateral/pb/5701-PB10-R.pdf.

[18] Carbon Dioxide Emissions from the Generation of Electric Power in the United States, Department of Energy and the Environmental Protection Agency, July 2000.
URL: http://www.eia.doe.gov/cneaf/electricity/page/co2_report/co2report.html.

[19] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 4, pp. 473-484, April 1992.

[20] J. Chang and L. Tassiulas, "Maximum Lifetime Routing in Wireless Sensor Networks," *Proceedings of the Advanced Telecommunications and Information Distribution Research Program (ATIRP'2000)*, March 2000.

[21] J. Chase and R. Doyle, "Balance of Power: Energy Management for Server Clusters," *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*, May 2001.

[22] J. Chase, C. Anderson, P. Thakar, R. Doyle, and A. Vahdat, "Managing Energy and Server Resources in Hosting Centers," *Proceedings of the 18th ACM Symposium on Operating System Principles*, pp. 103-116, October 2001.

[23] E. Chong and W. Zhao, "Performance Evaluation of Scheduling Algorithms for Imprecise Computer Systems," *Journal of Systems and Software*, Vol. 15, No. 3, pp. 261-277, July 1991.

[24] K. Christensen and F. Gulledge, "Enabling Power Management for Network-Attached Computers," *International Journal of Network Management*, Vol. 8, No. 2, pp. 120-130, March/April 1998.

[25] K. Christensen, C. Gunaratne, B. Nordman, and A. George, "The Next Frontier for Communications Networks: Power Management," *Computer Communications*, Vol. 27, No. 18, pp. 1758-1770, December 2004.

[26] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. De Micheli, "Dynamic Power Management for Non-stationary Service Requests," *IEEE Transactions on Computers*, Vol. 51, No. 11, pp. 1345-1361, November 2002.

[27] E.-Y. Chung, L. Benini, and G. De Micheli, "Dynamic Power Management Using Adaptive Learning Tree," *Digest of Technical Papers, 1999 IEEE/ACM International Conference on Computer-Aided Design*, pp. 274-279, November 1999.

[28] M. Crovella, M. Harchol-Balter, and C. Murta, "Task Assignment in a Distributed System: Improving Performance by Unbalancing Load," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 26, No. 1, pp. 268-269, June 1998.

[29] F. Douglis and P. Krishnan, "Adaptive Disk Spin-Down Policies for Mobile Computers," *Computing Systems*, Vol. 8, No. 4, pp. 381-413, 1995.

[30] Electric Power Monthly, July 2006 edition, Table 5.3.
URL: http://www.eia.doe.gov/cneaf/electricity/epm/epm_sum.html.

[31] C. Ellis, "The Case for Higher Level Power Management" *Proceedings of the 7th Workshop on Hot Topics in Operating Systems*, pp. 162, March 1999.

[32] ENERGY STAR.
URL: http://www.energystar.gov/.

[33] Energy Information Administration (EIA).
URL: http://www.eia.doe.gov.

[34] EU Energy Star.
URL: http://www.eu-energystar.org/en/en_000.htm.

[35] EU Stand-by Initiative.
URL: http://energyefficiency.jrc.cec.eu.int/html/standby_initiative.htm.

[36] EU stand-by initiative, "Code of Conduct on Energy Consumption of Broadband Equipment," European Commission Directorate General Joint Research Commission.
URL: http://energyefficiency.jrc.cec.eu.int/html/standby_initiative_broadbandcommunication.htm.

[37] "Faster, Quieter, Lower: Power Consumption and Noise Level of Contemporary Graphics Cards," X-bit labs, July 2006.
URL: http://www.xbitlabs.com/articles/video/display/power-noise.html.

[38] A. Field, U. Harder, and P. Harrison, "Network Traffic Behaviour in Switched Ethernet Systems," *Performance Evaluation*, Vol. 58, No. 2, pp. 243-260, 2004.

[39] J. Flinn and M. Satyanarayanan, "Energy-aware Adaptation for Mobile Applications," *Proceedings of the 17th ACM Symposium on Operating Systems Principles*, pp. 48-63, December 1999.

[40] J. Fryman, C. Huneycutt, H. Lee, K. Mackenzie, and D. Schimmel, "Energy-Efficient Network Memory for Ubiquitous Devices," *IEEE Micro*, Vol. 23, No. 5, pp. 60-70, September-October 2003.

[41] R. Gebhard, "A Queueing Process with Bilevel Hysteretic Service-Rate Control," *Naval Research Logistics Quarterly*, Vol. 14, pp. 55-68, 1967.

[42] G. Ginis, "Low Power Modes for ADSL2 and ADSL2+," Broadband Communications Group, Texas Instruments, SPAA021, January 2005.
URL: http://focus.ti.com/lit/an/spaa021/spaa021.pdf.

[43] S. Gochman, et al., "The Intel Pentium M Processor: Microarchitecture and Performance," *Intel Technology Journal*, Vol. 7, No. 2, pp. 21-36, May 2003.

[44] R. Golding, P. Bosch, and J. Wilkes, "Idleness is Not Sloth," *Technical Report HPL-96-140*, HP Laboratories, October 1996.

[45] D. Greenhill and J. Alabado, "Power Savings in the UltraSPARC T1 Processor," December 2005.
URL: http://www.sun.com/processors/whitepapers/UST1_pwrsav_v1.0.pdf.

[46] D. Gross and C. Harris, *Fundamentals of Queueing Theory*, 3rd ed., New York: John Wiley & Sons, 1998, pp. 89.

[47] C. Gunaratne and K. Christensen, "Ethernet Adaptive Link Rate: System Design and Performance Evaluation," *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pp. 28-35, November 2006.

[48] C. Gunaratne and K. Christensen, "A New Predictive Power Management Method for Network Devices," *IEE Electronics Letters*, Vol. 41, No. 13, pp.775-777, June 2005.

[49] C. Gunaratne, K. Christensen, and B. Nordman, "Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed," *International Journal of Network Management*, Vol. 15, No. 5, September/October 2005, pp. 297-310.

[50] C. Gunaratne, K. Christensen, and S. Suen, "Ethernet Adaptive Link Rate (ALR): Analysis of a Buffer Threshold Policy," *Proceedings of IEEE GLOBECOM 2006*, November 2006.

[51] M. Gundlach, S. Doster, H. Yan, D. Lowenthal, S. Watterson, and C. Surendar, "Dynamic, Power-Aware Scheduling For Mobile Clients Using a Transparent Proxy," *Proceedings of International Conference on Parallel Processing 2004*, Vol. 1, pp. 557-565, August 2004.

[52] C. Guo, L. Zhong, and J. Rabaey, "Low Power Distributed MAC for Ad Hoc Sensor Radio Networks," *Proceedings of IEEE GLOBECOM 2001*, pp. 2944-294, November 2001.

[53] M. Gupta, S. Grover, and S. Singh, "A Feasibility Study for Power Management in LAN Switches," *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04)*, pp. 361-371, October 2004.

[54] M. Gupta and S. Singh, "Greening of the Internet," *Proceedings of the ACM SIGCOMM 2003*, pp. 19-26, August 2003.

[55] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "Reducing Disk Power Consumption in Servers with DRPM," *IEEE Computer Special Issue on Power-Aware and Temperature-Aware Computing*, Vol. 36, No. 12, pp. 59-66, December, 2003.

[56] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *Proceeding of the Hawaii International Conference System Sciences*, Vol. 8, pp. 8020, January 2000.

[57] W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 174-185, August 1999.

[58] D. Helmbold, D. Long, T. Sconyers, and B. Sherrod, "Adaptive Disk Spin-Down for Mobile Computers," *ACM/Baltzer Mobile Networks and Applications (MONET) Journal*, Vol. 5, No. 4, pp. 285-297, December 2000.

[59] P. Huber and M. Mills, "Dig More Coal – The PCs Are Coming," *Forbes*, May 1999.

[60] C-H. Hwang and W. Wu, "A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 5, No. 2, pp. 226-241, April 2000.

[61] IEEE 802.3 LAN/MAN CSMA/CD Access Method.
URL: http://standards.ieee.org/getieee802/802.3.html.

[62] IEEE 802.3an (10GBASE-T) Task Force.
URL: http://www.ieee802.org/3/an/index.html.

[63] IEEE 802.3 10GBASE-T Tutorial.
URL: http://www.ieee802.org/3/10GBT/public/nov03/10GBASE-T_tutorial.pdf.

[64] IEEE 802.11 Wireless LAN Medium Access Control and Physical Layer Specification.
URL: http://standards.ieee.org/getieee802/802.11.html.

[65] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 56-67, August 2000.

[66] Intel 82541PI Gigabit Ethernet Controller – Overview.
URL: http://www.intel.com/design/network/products/lan/controllers/82541pi.htm.

[67] Intel Dual-Core Itanium 2 Processor 9000 Series – Product Brief.
URL: http://download.intel.com/products/processor/itanium2/dc_prod_brief.pdf.

[68] Intel – Moore's Law 40th Anniversary.
URL: http://www.intel.com/pressroom/kits/events/moores_law_40th/.

[69] Intel Pentium II Processors for Embedded Computing.
URL: http://www.intel.com/design/intarch/pentiumii/pentiumii.htm.

[70] Intel Pentium IV Processor on 90nm Process – Datasheet.
URL: http://download.intel.com/design/Pentium4/datashts/30056103.pdf.

[71] Internet Systems Consortium, Inc.
URL: http://www.isc.org.

[72] S. Irani, S. Shukla, and R. Gupta, "Algorithms for Power Savings," *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms*, pp. 37-46, January 2003.

[73] S. Irani, S. Shukla, and R. Gupta, "Online Strategies for Dynamic Power Management in Systems with Multiple Power Saving States," *ACM Transactions on Embedded Computing Systems, special issue on Power Aware Embedded Computing*, Vol. 2, No, 3, pp. 325-346, August 2005.

[74] L. Irish and K. Christensen, "A 'Green TCP/IP' to Reduce Electricity Consumed by Computers," *Proceedings of IEEE Southeastcon*, pp. 302-305, April 1998.

[75] ITU Recommendation G.114: One-way transmission time.
URL: http://www.itu.int/rec/T-REC-G.114-200305-I/en.

[76] ITU Recommendation G.992.1: Asymmetric digital subscriber line (ADSL) transceivers.
URL: http://www.itu.int/rec/T-REC-G.992.1/en.

[77] ITU Recommendation G.992.3: Asymmetric digital subscriber line transceivers 2 (ADSL2).
URL: http://www.itu.int/rec/T-REC-G.992.3/en.

[78] ITU Recommendation G.992.5: Asymmetric digital subscriber line (ADSL) transceivers.
URL: http://www.itu.int/rec/T-REC-G.992.5/en.

[79] ITU Recommendation G.993.1: Very high speed digital subscriber line transceivers. URL: http://www.itu.int/rec/T-REC-G.993.1/en.

[80] IXIA 400T. URL: http://www.ixiacom.com/pdfs/datasheets/ch_1600t_400t.pdf.

[81] R. Jain and I. Chlamtac, "The P2 Algorithm for Dynamic Calculation of Quantiles and Histograms without Storing Observations," *Communications of the ACM*, Vol. 28, No. 10, pp. 1076-1085, October 1985.

[82] C. Jones, K. Sivalingam, P. Agrawal, and J. Chen, "A Survey of Energy Efficient Network Protocols for Wireless Networks," *Wireless Networks*, Vol. 7, No. 4, pp. 343-358, July 2001.

[83] K. Kawamoto, J. Koomey, B. Nordman, R. Brown, M. Piette, M. Ting, and A. Meier, "Electricity Used by Office Equipment and Network Equipment in the U.S.: Detailed Report and Appendices," *Technical Report LBNL-45917*, Energy Analysis Department, Lawrence Berkeley National Laboratory, February 2001.

[84] R. Kehr, A. Zeidler, and H. Vogt, "Towards a Generic Proxy Execution Service for Small Devices," *Presented at FuSeNetD (Future Services For Networked Devices) Workshop at Heidelberg*, November 1999.

[85] A. Klaiber, "The Technology Behind Crusoe Processors," January 2000. URL: http://www.transmeta.com/pdfs/paper_aklaiber_19jan00.pdf.

[86] J. Klamra, M. Olsson, K. Christensen, and B. Nordman, "Design and Implementation of a Power Management Proxy for Universal Plug and Play," *Proceedings of the Swedish National Computer Networking Workshop (SNCW 2005)*, September 2005.

[87] L. Kleinrock, "On Flow Control in Computer Networks," *Proceedings of the 1978 International Conference on Communications*, Vol. 2, pp. 27.2.1-27.2.5, June 1978.

[88] J. Koomey, "Sorry, Wrong Number – How to Separate Fact From Fiction in the Information Age," *IEEE Spectrum*, Vol. 40, No. 6, pp. 11-12, June 2003.

[89] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown," *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking (Mobicom) 2002*, pp. 119-130, September 2002.

[90] V. Kulkarni, *Modeling, Analysis, Design, and Control of Stochastic Systems*, New York: Springer Verlag, 1999, pp. 199.

[91] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE/ACM Transactions on Networking*, Vol. 2, No. 1, pp. 1-15, February 1994. URL: http://ita.ee.lbl.gov/html/contrib/BC.html.

[92] S. Lindsey and C. Raghavendra, "PEGASIS: Power Efficient GAthering in Sensor Information Systems," *Proceedings of the IEEE Aerospace Conference*, Vol. 3, pp. 1125-1130, March 2002.

[93] K. Mania, B. Adelstein, S. Ellis, and M. Hill, "Perceptual Sensitivity to Head Tracking Latency in Virtual Environments with Varying Degrees of Scene Complexity," *Proceedings of 1st ACM SIGGRAPH Symposium on Applied Perception in Graphics and Visualization*, pp. 39-47, August 2004.

[94] Micron® System-Power Calculator.
URL: http://www.micron.com/support/designsupport/tools/powercalc/powercalc.

[95] Microsoft Developer Network: WM_POWERBROADCAST Messages.
URL: http://windowssdk.msdn.microsoft.com/en-us/library/ms714828.aspx.

[96] J. Mitchell-Jackson, "Energy Needs in an Internet Economy: A Closer Look at Data Centers," *Master's Thesis*, Energy and Resources Group, University of California at Berkeley, May 2001.

[97] T. Mudge, "Power: A First-Class Architectural Design Constraint," *IEEE Computer*, Vol. 34, No. 4, pp. 52-58, April 2001.

[98] B. Nordman and K. Christensen, "Reducing the Energy Consumption of Network Devices," Tutorial presented at the July 2005 IEEE 802 LAN/MAN Standards Committee Plenary Session, July 2005.
URL: http://www.csee.usf.edu/~christen/energy/pubs.html.

[99] B. Nordman, M. Piette, K. Kinney, and C. Webber, "User Guide to Power Management in PCs and Monitors," *Technical report LBNL-39466*, Environmental Energy Technologies Division, Lawrence Berkeley National Laboratory, January 1997.
URL: http://eetd.lbl.gov/EA/Reports/39466/39466.PDF.

[100] A. Odlyzko, "Data Networks are Lightly Utilized, and Will Stay That Way," *Review of Network Economics*, Vol. 2, No. 3, pp. 210-237, September 2003.

[101] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney, "A First Look at Modern Enterprise Traffic," *ACM SIGCOMM/USENIX Internet Measurement Conference*, October 2005.

[102] L. Pantel and L. Wolf, "On the Impact of Delay on Real-Time Multiplayer Games," *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 23-29, 2002.

[103] A. Papathanasiou and M. L. Scott, "Energy Efficient Prefetching and Caching," *Proceedings of the USENIX Annual Technical Conference*, June 2004.

[104] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath, "Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems", *Proceedings of the Workshop on Compilers and Operating Systems for Low Power, 10th International Conference on Parallel Architectures and Compilation Techniques*, September 2001.

[105] Q. Qiu and M. Pedram, "Dynamic Power Management Based on Continuous-Time Markov Decision Processes," *Proceedings of the 36th Design Automation Conference 1999*, pp. 555-561, June 1999.

[106] Q. Qiu, Q. Wu, and M. Pedram, "Dynamic Power Management of Complex Systems Using Generalized Stochastic Petri Nets," *Proceedings of the 37th Design Automation Conference 2000*, pp. 352-356, June 2000.

[107] Residential Energy Consumption Survey for year 2001, Energy Information Administration.
URL: http://www.eia.doe.gov/emeu/recs/.

[108] Revisions to existing EPA Energy Star specifications: Computer Specification.
URL: http://www.energystar.gov/index.cfm?c=revisions.computer_spec.

[109] J. Roberson, C. Webber, M. McWhinney, R. Brown, M. Pinckard, and J. Busch, "After-hours Power Status of Office Equipment and Inventory of Miscellaneous Plug-Load Equipment," *Technical report LBNL-53729*, Energy Analysis Department, Lawrence Berkeley National Laboratory, January 2004.

[110] V. Rodoplu and T. Ming, "Minimum Energy Mobile Wireless Networks," *IEEE Journal of Selected Areas in Communications*, Vol. 17, No. 8, pp. 1333-1344, August 1999.

[111] M. Rosu, C. Olsen, C. Narayanaswami, and L. Luo, "PAWP: A Power Aware Web Proxy for Wireless LAN Clients," *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2004)*, pp.206-215, December 2004.

[112] K. Roth, F. Goldstein, and J. Kleinman, "Energy Consumption by Office and Telecommunications Equipment in Commercial Buildings, Volume I: Energy Consumption Baseline," *Arthur D. Little Reference No. 72895-00*, January 2002.

[113] H. Schwetman, "CSIM19: A Powerful Tool for Building System Models," *Proceedings of the 33rd Winter Simulation Conference*, pp. 250-255, December 2001.

[114] E. Shih, P. Bahl, and M. Sinclair, "Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices," *Proceedings of the 8th International Conference on Mobile Computing and Networking*, pp. 161-170, 2002.

[115] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, "Event-Driven Power Management," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 20, No. 7, pp. 840-857, July 2001.

[116] H. Singh and S. Singh, "Energy Consumption of TCP Reno, NewReno, and SACK in Multi-Hop Wireless Networks," *Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp. 206-216, 2002.

[117] S. Singh and C. Raghavendra, "PAMAS: Power Aware Multi-Access Protocol with Signaling for Ad Hoc Networks," *Computer Communications Review*, Vol. 28, No. 3, pp. 5-26, July 1998.

[118] K. Sivalingam, J. Chen, P. Agrawal, and M. Srivastava, "Design and Analysis of Low Power Access Protocols for Wireless and Mobile Networks," *Wireless Networks*, Vol. 6, No. 1, pp. 73-87, 2000.

[119] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen, "Predictive System Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 4, No. 1, pp. 42-55, March 1996.

[120] R. Steinmetz, "Human Perception of Jitter and Media Synchronization," *IEEE Journal on Selected Areas in Communications*, Vol. 14, No. 1, pp. 61-72, January 1996.

[121] M. Stemm and R. Katz, "Measuring and Reducing the Energy Consumption on Network Interfaces in Handheld Devices," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science*, pp, 1125-1131, August 1997.

[122] Stephen W. Suen, Associate Professor, Department of Mathematics and Statistics, University of South Florida; personal communication.

[123] Switchcore Xpeedium2™ CXE-1100.
URL: http://www.switchcore.se/products/cxe-1100/.

[124] TCPView for Microsoft Windows v2.4.
URL: http://www.microsoft.com/technet/sysinternals/utilities/TcpView.mspx.

[125] V. Tsaoussidis and H. Badr, "Energy/Throughput Tradeoffs of TCP Error Control Strategies," *Proceedings of the 5th IEEE Symposium on Computers and Communications (ISCC 2000)*, pp. 106-112, July 2000.

[126] V. Tsaoussidis and H. Badr, "TCP-Probing: Towards an Error Control Schema with Energy and Throughput Performance Gains," *Proceedings of the 8th IEEE International Conference on Network Protocols (ICNP'00)*, pp. 12-21, November 2000.

[127] V. Tsaoussidis and H. Badr, "Wave and Wait Protocol: An Energy-Saving Transport Protocol for Mobile-IP Devices," *Proceedings of the 7th International Conference on Network Protocols (ICNP '99)*, pp. 301-308, October 1999.

[128] M. Tzannes, "ADSL2 Helps Slash Power in Broadband Designs," *CommsDesign*, January 2003.
URL: http://www.commsdesign.com/design_corner/OEG20030130S0008.

[129] A. Vahdat, A. R. Lebeck, and C. S. Ellis, "Every Joule is Precious: A Case for Revisiting Operating System Design for Energy Efficiency," *Proceedings of the 9th ACM SIGOPS European Workshop*, pp. 31-36, September 2000.

[130] B. Wang and S. Singh, "Computational Energy Cost of TCP," *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) 2004*, Vol. 2, pp.785-795, March 2004.

[131] C. Webber, J. Roberson, R. Brown, C. Payne, B. Nordman, and J. Koomey, "Field Surveys of Office Equipment Operation Patterns," *Technical report LBNL-46930*, Energy Analysis Department, Lawrence Berkeley National Laboratory, September 2001.

[132] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," *Proceedings of the 1st Symposium on Operating Systems Design and Implementation*, pp. 13-23, November 1994.

[133] M. Williams, "Samsung Hybrid Flash-Disk Drive to Boost Laptop Performance," *www.Infoworld.com*, May 2006.
URL: http://www.infoworld.com/article/06/05/17/78429_HNhybriddrive_1.html.

[134] K. Yoshigoe and K. Christensen, "A Parallel-Polled Virtual Output Queued Switch with a Buffered Crossbar," *Proceedings of the IEEE Workshop on High Performance Switching and Routing*, pp. 271-275, May 2001.

[135] V. Zandy and B. Miller, "Reliable Network Connections," *Proceedings of the 8th ACM MobiCom*, pp. 95-106, September 2002.

[136] H. Zeng, X. Fan, C. Ellis, A. Lebeck, and A. Vahdat, "ECOSystem: Managing Energy as a First Class Operating System Resource," *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)*, October 2002.

[137] Y. Zhang and S. Dao, "A 'Persistent Connection' Model for Mobile and Distributed Systems," *Proceedings of the 4th International Conference on Computer Communications and Networks*, pp. 300-307, September 1995.

[138] M. Zorzi and R. Rao, "Energy Efficiency of TCP in a Local Wireless Environment," *Mobile Network and Applications*, Vol. 6, pp. 265-278, 2001.

Appendices

Appendix A: Derivation of Steady State Probabilities

The steady state probabilities for the dual-threshold, service rate transition at service completion model are derived in this appendix. Parameter definitions and usage is similar to Chapter 5. The symbol π_n denotes the steady state probability of state n with P_n denoting the steady state probability of having n number of customers in the system. The lower threshold for changing service rates is denoted by k_1 and the higher threshold for changing service rates is denoted by k_2. Parameters k_1 and k_2 are the number of customers in the system and $k_1 < k_2$. The rate of arrivals is given by λ and μ_1 denotes the lower rate of service and μ the high rate of service. $\mu_1 < \mu$ and for stability $\lambda < \mu$. The parameters ρ_1 and ρ are defined as $\rho_1 = \lambda/\mu_1$ and $\rho = \lambda/\mu$, respectively.

In this model (depicted in Figure A.1), if the current rate of service is μ_1, when the number of customers in the system equals or exceeds threshold value k_2, the rate of service changes to μ. If the current rate of service is μ, when the number of customers in the system is less than k_1, the rate of service changes to μ_1. The rate of service only changes at the end of the service interval where the threshold crossing occurs. The derivation is given as a series of steps.

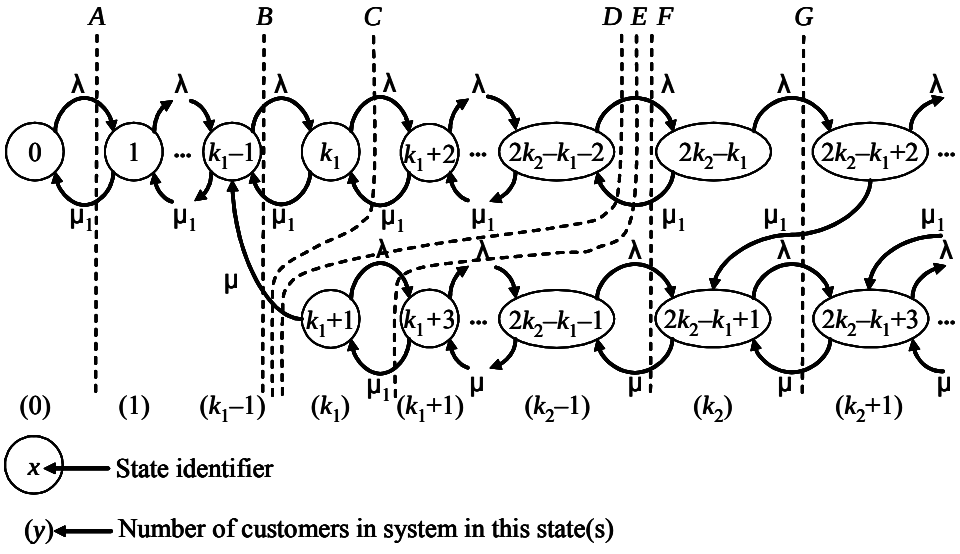


Figure A.1. Dual-Threshold, Rate Transition at Service Completion Markov Model

Appendix A: (Continued)

The steady state probabilities for having $(k_1 - 1)$ customers or less than $(k_1 - 1)$ customers in the system are derived in this section. The balance equations along partition *A* in Figure A.1 yields the following:

$$\pi_0 \lambda = \pi_1 \mu_1 . \tag{A.1}$$

Similarly,

$$\pi_1 \lambda = \pi_2 \mu_1 . \tag{A.2}$$

It can be observed from Figure A.1 that where $n \le k_1 - 1$

$$\pi_{n-1} \lambda = \pi_n \mu_1 . \tag{A.3}$$

Therefore,

$$\pi_n = \pi_0 \rho_1^n \ (n \le k_1 - 1). \tag{A.4}$$

The steady state probabilities of the 'parallel' states with service rate μ_1 are derived in this section. Relationships are constructed between the states using balance equations. Upon inspection of Figure A.1, the following can be observed:

1. $\pi_{k_1 - 1}$ is known and the balance equation along partition *B* will yield a relationship between $\pi_{k_1 - 1}$, π_{k_1} and $\pi_{k_1 + 1}$.
2. From Section 5.2.2, it is known that $\pi_{2k_2 - k_1} = \pi_{2k_2 - k_1 - 2} \left(\dfrac{\lambda}{\lambda + \mu_1} \right)$.
3. Balance equation along partition *D* will yield a relationship between $\pi_{k_1 + 1}$ and $\pi_{2k_2 - k_1 - 2}$.
4. Balance equations along partition *C* and similar partitions will possibly yield relationships between $\pi_{2k_2 - k_1 - 2}$, π_{k_1} and the intermediate states.

Appendix A: (Continued)

Balance equation along partition *D* in Figure A.1 gives the following relationship:

$$\lambda\pi_{2k_2-k_1-2} = \mu\pi_{k_1+1} + \mu_1\pi_{2k_2-k_1} . \tag{A.5}$$

Since $\pi_{2k_2-k_1} = \pi_{2k_2-k_1-2}\left(\frac{\lambda}{\lambda+\mu_1}\right)$,

Equation (A.5) can be expressed as

$$\lambda.\pi_{2k_2-k_1-2} = \mu.\pi_{k_1+1} + \mu_1\pi_{2k_2-k_1-2}\left(\frac{\lambda}{\lambda+\mu_1}\right). \tag{A.6}$$

This simplifies to

$$\left(\frac{\lambda^2}{\lambda+\mu_1}\right)\pi_{2k_2-k_1-2} = \mu.\pi_{k_1+1} . \tag{A.7}$$

Balance equations along a partition similar to partition *C* in Figure A.1 and running between states $2k_2 - k_1 - 2$ and $2k_2 - k_1 - 4$ yield the following equation:

$$\lambda\pi_{2k_2-k_1-4} = \mu_1\pi_{2k_2-k_1-2} + \mu\pi_{k_1+1} . \tag{A.8}$$

From (A.7), it is possible to substitute for π_{k_1+1} giving:

$$\lambda\pi_{2k_2-k_1-4} = \mu_1\pi_{2k_2-k_1-2} + \left(\frac{\lambda^2}{\lambda+\mu_1}\right)\pi_{2k_2-k_1-2} . \tag{A.9}$$

This can be simplified to

$$\pi_{2k_2-k_1-4} = \pi_{2k_2-k_1-2}\left(\frac{1}{\lambda}\right)\left(\frac{{\mu_1}^2 + \lambda\mu_1 + \lambda^2}{\lambda+\mu_1}\right). \tag{A.10}$$

Similarly, balance equations along a partition similar to partition *C* in Figure A.1 between states $2k_2 - k_1 - 4$ and $2k_2 - k_1 - 6$ yield the following equation:

$$\lambda\pi_{2k_2-k_1-6} = \mu_1\pi_{2k_2-k_1-4} + \mu\pi_{k_1+1} . \tag{A.11}$$

Appendix A: (Continued)

And it is possible to substitute and simplify this equation to:

$$\pi_{2k_2-k_1-6} = \pi_{2k_2-k_1-2}\left(\frac{1}{\lambda}\right)\left(\frac{\frac{\mu_1^{\,3}}{\lambda} + \mu_1^{\,2} + \lambda\mu_1 + \lambda^2}{\lambda + \mu_1}\right). \tag{A.12}$$

Similarly, balance equations along a partition similar to partition *C* in Figure A.1 between states $2k_2 - k_1 - 6$ and $2k_2 - k_1 - 8$ and simplification yield the following equation:

$$\pi_{2k_2-k_1-8} = \pi_{2k_2-k_1-2}\left(\frac{1}{\lambda}\right)\left(\frac{\frac{\mu_1^{\,4}}{\lambda^2} + \frac{\mu_1^{\,3}}{\lambda} + \mu_1^{\,2} + \lambda\mu_1 + \lambda^2}{\lambda + \mu_1}\right). \tag{A.13}$$

It can be observed that there exists a pattern in the equations and this pattern can be generalized to:

$$\pi_{2k_2-k_1-2n} = \pi_{2k_2-k_1-2}\left(\sum_{i=0}^{n}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)\ (n>0). \tag{A.14}$$

Recall that the only known value is π_{k_1-1}. It is required to establish relationships between the known and the unknown terms in order to express the unknown terms using the known terms. Therefore, balance equation along partition *B* in Figure A.1 yields the following relationship

$$\lambda.\pi_{k_1-1} = \mu_1.\pi_{k_1} + \mu.\pi_{k_1+1}\,. \tag{A.15}$$

From (A.14) it is possible to derive the following expression for $\pi_{2k_2-k_1-2}$

$$.\pi_{k_1} = \pi_{2k_2-k_1-2}\left(\sum_{i=0}^{k_2-k_1}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right). \tag{A.16}$$

From (A.7) it is possible to express $\pi_{2k_2-k_1-2}$ in terms of $\pi_{2k_2-k_1-2}$. Therefore, it is possible to write the above expression as

$$\pi_{k_1-1} = \left(\frac{\mu_1}{\lambda}\right)\pi_{2k_2-k_1-2}\cdot\left(\sum_{i=0}^{k_2-k_1}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right) + \left(\frac{\lambda^2}{\lambda+\mu_1}\right)\pi_{2k_2-k_1-2}\left(\frac{1}{\lambda}\right) \tag{A.17}$$

Appendix A: (Continued)

This can be simplified and used to express $\pi_{2k_2-k_1-2}$ in terms of π_{k_1-1} (the known value) giving

$$\pi_{2k_2-k_1-2} = \frac{\pi_{k_1-1}}{\left(\sum_{i=0}^{k2-k1+1}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)}. \tag{A.18}$$

Therefore, the steady state probability $\pi_{2k_2-k_1-2}$ is now known. Recall that in (A.14) it was possible to derive a generalized expression in terms of $\pi_{2k_2-k_1-2}$. With $\pi_{2k_2-k_1-2}$ known, the generalized expression can be used. First, (A.14) is rewritten so that the states are numbered from k_1 forwards rather than backwards from k_2

$$\pi_{k_1+2n} = \pi_{2k_2-k_1-2}\left(\sum_{i=0}^{k_2-n}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right) \tag{A.19}$$

Then, by substituting (A.18) it is possible to obtain a general expression for steady state probabilities for the 'parallel' states where the service rate is μ_1

$$\pi_{k_1+2n} = \frac{\pi_{k_1-1}}{\left(\sum_{i=0}^{k2-k1+1}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)}\left(\sum_{i=0}^{k_2-n}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right) \tag{A.20}$$

π_{k_1-1} can be expressed as $\pi_{k_1-1} = \pi_0\rho_1^{k_1-1}$. Therefore:

$$\pi_{k_1+2n} = \frac{\pi_0\rho_1^{k_1-1}}{\left(\sum_{i=0}^{k2-k1+1}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)}\left(\sum_{i=0}^{k_2-n}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right). \tag{A.21}$$

And it is possible to use summation of series formulas to derive the following closed form expression:

$$\pi_{k_1+2n} = \frac{\pi_0\rho_1^{k_1-1}}{\left(1-\left(\frac{1}{\rho_1}\right)^{k_2-k_1+2}\right)}\left(1-\left(\frac{1}{\rho_1}\right)^{k_2-n+1}\right)\left(0 \le n \le (k_2-k_1-1)\right). \tag{A.22}$$

Appendix A: (Continued)

And it is possible to express this formula with n giving the number of customers in the system:

$$\pi_{k_1+2n} = \frac{\pi_0 \rho_1^{k_1-1}}{\left(1-\left(\frac{1}{\rho_1}\right)^{k_2-k_1+2}\right)}\left(1-\left(\frac{1}{\rho_1}\right)^{k_2-n+1}\right)\left(k_1 \le n < k_2\right). \tag{A.23}$$

The steady state probabilities of the 'parallel' states with service rate μ are derived in this section. The value of π_{k_1+1} and the relationship to $\pi_{2k_2-k_1-2}$ is known, but the values of $\pi_{k_2-k_1-1}$ and the steady state probabilities of the intermediate states are not known. By partitioning the Markov chain along partition *E* and similarly to *E*, it is possible to develop a generalized expression for the steady state probabilities. Balance equations along a partition similar to partition *E* in Figure A.1 and running between states $2k_2 - k_1 - 3$ and $2k_2 - k_1 - 1$ yield the following equation:

$$\mu\pi_{2k_2-k_1-1} + \mu_1\pi_{2k_2-k_1} = \lambda\pi_{2k_2-k_1-3} + \lambda\pi_{2k_2-k_1-2}. \tag{A.24}$$

Since it is known that

$$\pi_{2k_2-k_1} = \pi_{2k_2-k_1-2}\left(\frac{\lambda}{\lambda+\mu_1}\right), \tag{A.25}$$

$$\mu\pi_{2k_2-k_1-1} + \mu_1\pi_{2k_2-k_1-2}\left(\frac{\lambda}{\lambda+\mu_1}\right) = \lambda\pi_{2k_2-k_1-3} + \lambda\pi_{2k_2-k_1-2}. \tag{A.26}$$

This can then be simplified to yield:

$$\mu\pi_{2k_2-k_1-1} = \lambda\pi_{2k_2-k_1-3} + \pi_{2k_2-k_1-2}\left(\frac{\lambda^2}{\lambda+\mu_1}\right). \tag{A.27}$$

And then further simplified to yield

$$\pi_{2k_2-k_1-1} = \left(\frac{\lambda}{\mu}\right)\pi_{2k_2-k_1-3} + \left(\frac{1}{\mu}\right)\pi_{2k_2-k_1-2}\left(\frac{\lambda^2}{\lambda+\mu_1}\right). \tag{A.28}$$

Appendix A: (Continued)

Similarly, balance equations along a partition similar to partition *E* in Figure A.1 and running between states $2k_2 - k_1 - 5$ and $2k_2 - k_1 - 3$ yield the following equation:

$$\pi_{2k_2-k_1-3} = \left(\frac{\lambda}{\mu}\right)\pi_{2k_2-k_1-5} + \left(\frac{1}{\mu}\right)\pi_{2k_2-k_1-2}\left(\frac{\lambda^2}{\lambda+\mu_1}\right). \tag{A.29}$$

The above equation is substituted in (A.28), giving

$$\pi_{2k_2-k_1-1} = \left(\frac{\lambda}{\mu}\right)^2 \pi_{2k_2-k_1-5} + \left(\frac{\lambda}{\mu}+1\right)\pi_{2k_2-k_1-2}\left(\frac{1}{\mu}\right)\left(\frac{\lambda^2}{\lambda+\mu_1}\right). \tag{A.30}$$

A pattern can be observed and it is possible to rewrite above in general terms:

$$\pi_{2k_2-k_1-1} = \left(\frac{\lambda}{\mu}\right)^n \pi_{2k_2-k_1-1-2n} + \left(\sum_{i=0}^{n-1}\left(\frac{\lambda}{\mu}\right)^i\right)\pi_{2k_2-k_1-2}\left(\frac{1}{\mu}\right)\left(\frac{\lambda^2}{\lambda+\mu_1}\right). \tag{A.31}$$

The equation above is numbered back from state k_2. Therefore, it is rewritten so that the equation numbered forward from state k_1, giving

$$\pi_{k_1+1+2n} = \pi_{k_1+1+2(n-i)}\left(\frac{\lambda}{\mu}\right)^i + \left(\sum_{i=1}^{n-1}\left(\frac{\lambda}{\mu}\right)^i\right)\pi_{2k_2-k_1-2}\left(\frac{1}{\mu}\right)\left[\frac{\lambda^2}{\lambda+\mu_1}\right]. \tag{A.32}$$

Now, $\pi_{k_1+1+2(n-i)}$ is not known and π_{k_1+1} is known and therefore, it is possible to iterate and write the above expression as

$$\pi_{k_1+1+2n} = \pi_{k_1+1}\left(\frac{\lambda}{\mu}\right)^n + \left(\sum_{i=0}^{n-1}\left(\frac{\lambda}{\mu}\right)^i\right)\pi_{2k_2-k_1-2}\left(\frac{1}{\mu}\right)\left[\frac{\lambda^2}{\lambda+\mu_1}\right]. \tag{A.33}$$

From (A.7), the following substitution is possible:

$$\left(\frac{\lambda^2}{\lambda+\mu_1}\right)\pi_{2k_2-k_1-2} = \mu\pi_{k_1+1}. \tag{A.34}$$

And it is possible obtain the following general equation

$$\pi_{k_1+1+2n} = \pi_{2k_2-k_1-2}\left(\frac{1}{\mu}\right)\left(\frac{\lambda^2}{\lambda+\mu_1}\right)\left(\frac{\lambda}{\mu}\right)^n + \left(\sum_{i=0}^{n-1}\left(\frac{\lambda}{\mu}\right)^i\right)\pi_{2k_2-k_1-2}\left(\frac{1}{\mu}\right)\left(\frac{\lambda^2}{\lambda+\mu_1}\right). \tag{A.35}$$

Appendix A: (Continued)

This can be simplified as given in the following steps:

$$\pi_{k_1+1+2n} = \pi_{2k_2-k_1-2}\left(\frac{1}{\mu}\right)\left(\frac{\lambda^2}{\lambda+\mu_1}\right)\left(\left(\frac{\lambda}{\mu}\right)^n + \sum_{i=0}^{n-1}\left(\frac{\lambda}{\mu}\right)^i\right), \tag{A.36}$$

$$\pi_{k_1+1+2n} = \pi_{2k_2-k_1-2}\left(\frac{1}{\mu}\right)\left(\frac{\lambda^2}{\lambda+\mu_1}\right)\left(\sum_{i=0}^{n}\left(\frac{\lambda}{\mu}\right)^i\right). \tag{A.37}$$

From (A.18) it is known that $\pi_{2k_2-k_1-2}$ is given by

$$\pi_{2k_2-k_1-2} = \frac{\pi_{k_1-1}}{\left(\sum_{i=0}^{k2-k1+1}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)}. \tag{A.38}$$

Therefore,

$$\pi_{k_1+1+2n} = \frac{\pi_{k_1-1}}{\left(\sum_{i=0}^{k2-k1+1}\left(\frac{\mu_1}{\lambda}\right)^i\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)}\left(\frac{1}{\mu}\right)\left(\frac{\lambda^2}{\lambda+\mu_1}\right)\left(\sum_{i=0}^{n}\left(\frac{\lambda}{\mu}\right)^i\right). \tag{A.39}$$

$$\pi_{k_1+1+2n} = \frac{\pi_{k_1-1}\rho}{\left(\sum_{i=0}^{k2-k1+1}\left(\frac{\mu_1}{\lambda}\right)^i\right)}\left(\sum_{i=0}^{n}\left(\frac{\lambda}{\mu}\right)^i\right). \tag{A.40}$$

This can further be simplified into the following closed-form expression

$$\pi_{k_1+1+2n} = \frac{\pi_0\rho_1^{\,k_1-1}\rho}{\left(\frac{1-\left(\frac{1}{\rho_1}\right)^{k_2-k_1+2}}{1-\left(\frac{1}{\rho_1}\right)}\right)}\left(\frac{1-\rho^{n+1}}{1-\rho}\right)\ \left(0 \le n \le \left(k_2-k_1-1\right)\right). \tag{A.41}$$

Appendix A: (Continued)

Equation (A.39) can be rewritten so that n gives the number of customers in the system

$$\pi_n = \frac{\pi_0 \rho_1^{k_1-1} \rho}{\left(\frac{1-\left(\frac{1}{\rho_1}\right)^{k_2-k_1+2}}{1-\left(\frac{1}{\rho_1}\right)} \right)} \left(\frac{1-\rho^{n-k_1+1}}{1-\rho} \right) \left(k_1 \le n < k_2 \right) \tag{A.42}$$

From (A.22) and (A.42) it is possible to construct an equation for the steady state probabilities of having $k_1 \le n < k_2$ customers in the system where n is the number of customers in the system. From Figure A.1, it can be observed that

$$P_n = \pi_{k_1+2n} + \pi_{k_1+1+2n}. \tag{A.43}$$

Therefore,

$$P_n = \frac{\pi_0 \rho_1^{k_1-1} \left(\rho_1 - 1\right)}{\rho_1 \left(1-\left(\frac{1}{\rho_1}\right)^{k_2-k_1+2} \right)} \left(\frac{\left(1-\left(\frac{1}{\rho_1}\right)^{k_2-n+1} \right)}{1-\left(\frac{1}{\rho_1}\right)} + \rho \left(\frac{1-\rho^{n-k_1+1}}{1-\rho} \right) \right). \tag{A.44}$$

The steady state probabilities for having k_2 or more than k_2 customers in the system are derived in this section. From local balance equation along partition *F* in Figure A.1, it can be written that

$$\lambda \pi_{2k_2-k_1-2} + \lambda \pi_{2k_2-k_1-1} = \mu_1 \pi_{2k_2-k_1} + \mu \pi_{2k_2-k_1+1}. \tag{A.45}$$

From Section 5.2.2 it is known that

$$\pi_{2k_2-k_1} = \pi_{2k_2-k_1-2} \left(\frac{\lambda}{\lambda+\mu_1} \right). \tag{A.46}$$

Therefore, by observation from Figure A.1, (A.45) can be rewritten as

$$\lambda P_{k_2-1} = \mu_1 \pi_{2k_2-k_1-2} \left(\frac{\lambda}{\lambda+\mu_1} \right) + \mu \left(P_{k_2} - \pi_{2k_2-k_1-2} \left(\frac{\lambda}{\lambda+\mu_1} \right) \right). \tag{A.47}$$

Appendix A: (Continued)

From local balance equations along partition **G** in Figure A.1, it can be written that

$$\lambda\pi_{2k_2-k_1} + \lambda\pi_{2k_2-k_1+1} = \mu_1\pi_{2k_2-k_1+2} + \mu\pi_{2k_2-k_1+3}\,. \tag{A.48}$$

It is known (from Section 5.2.2.) that

$$\pi_{2k_2-k_1+2} = \pi_{2k_2-k_1-2}\left(\frac{\lambda}{\lambda+\mu_1}\right)^2. \tag{A.49}$$

And by observation from Figure A.1, (A.48) can be rewritten as

$$\lambda P_{k_2} = \mu_1\pi_{2k_2-k_1-2}\left(\frac{\lambda}{\lambda+\mu_1}\right)^2 + \mu\left(P_{k_2} - \pi_{2k_2-k_1-2}\left(\frac{\lambda}{\lambda+\mu_1}\right)^2\right). \tag{A.50}$$

From (A.45) and (A.48) it is possible to write a general expression

$$\lambda P_{n-1} = \mu_1\pi_{2k_2-k_1-2}\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k_2+1} + \mu\left(P_n - \pi_{2k_2-k_1-2}\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k_2+1}\right)\left(n \geq k_2\right). \tag{A.51}$$

The above equation can be rewritten so that P_n is isolated on one side of the equation

$$P_n = P_{n-1}\left(\frac{\lambda}{\mu}\right) + \pi_{2k_2-k_1-2}\left(1-\frac{\mu_1}{\mu}\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k_2+1}\left(n \geq k_2\right). \tag{A.52}$$

By iterating the following equation can be derived

$$P_n = P_{k_2-1}\left(\frac{\lambda}{\mu}\right)^{n-k_2+1} + \pi_{2k_2-k_1-2}\left(1-\frac{\mu_1}{\mu}\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k_2+1}\left(1+\frac{\left(\frac{\lambda}{\mu}\right)}{\left(\frac{\lambda}{\lambda+\mu_1}\right)}+\frac{\left(\frac{\lambda}{\mu}\right)^2}{\left(\frac{\lambda}{\lambda+\mu_1}\right)^2}+.....+\frac{\left(\frac{\lambda}{\mu}\right)^{n-k_2}}{\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k_2}}\right). \tag{A.53}$$

This can be simplified further:

$$P_n = P_{k_2-1}\left(\frac{\lambda}{\mu}\right)^{n-k_2+1} + \pi_{2k_2-k_1-2}\left(1-\frac{\mu_1}{\mu}\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k_2+1}\left(\sum_{i=0}^{n-k_2}\frac{\left(\frac{\lambda}{\mu}\right)^i}{\left(\frac{\lambda}{\lambda+\mu_1}\right)^i}\right), \tag{A.54}$$

Appendix A: (Continued)

$$P_n = P_{k_2-1}\left(\frac{\lambda}{\mu}\right)^{n-k_2+1} + \pi_{2k_2-k_1-2}\left(1-\frac{\mu_1}{\mu}\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k_2+1}\left(\sum_{i=0}^{n-k_2}\left(\frac{\lambda+\mu_1}{\mu}\right)^i\right), \tag{A.55}$$

$$P_n = P_{k_2-1}\left(\frac{\lambda}{\mu}\right)^{n-k_2+1} + \pi_{2k_2-k_1-2}\left(1-\frac{\mu_1}{\mu}\right)\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k_2+1}\left(\frac{1-\left(\frac{\lambda+\mu_1}{\mu}\right)^{n-k_2+1}}{1-\left(\frac{\lambda+\mu_1}{\mu}\right)}\right). \tag{A.56}$$

The final closed-form formula is:

$$P_n = \frac{P_0\rho^{k_1-1}}{\left(\frac{1}{\rho 1}\right)^{k_2-k_1+2}-1}\left(\rho^{n-k_2+1}\left(\left(\frac{1}{\rho}\right)^2-1+\frac{\rho\left(\rho^{k_2-k_1}-1\right)\left(\frac{1}{\rho_1}-1\right)}{\rho-1}\right)+\left(\left(\left(\frac{1}{\rho}\right)^2-1\right)(\mu-\mu_1)\left(\frac{\left(\frac{\lambda}{\lambda+\mu_1}\right)^{n-k_2+1}-\left(\frac{\lambda}{\mu}\right)^{n-k_2+1}}{\mu-\lambda-\mu_1}\right)\right)\right). \tag{A.57}$$

To derive P_0, the sum of the steady state probabilities from (A.4), (A.44), and (A.57) is set to equal 1. After simplification, it is possible to obtain the following expression for P_0:

$$P_0 = \left(\frac{1-\rho_1^{k_1}}{1-\rho_1}+\frac{1}{1-\rho_1^{k_2-k_1+2}}\left(\frac{\rho_1^{k_2}-\rho_1^{k_1}}{\rho_1-1}+\frac{\rho_1^{k_2}\left(\left(k_2-k_1\right)\left(\rho_1-\rho\right)+\rho_1^2-1\right)}{\rho-1}\right)\right)^{-1}. \tag{A.58}$$

In summary, the final closed-form formulas for the steady state probabilities are given by:

$$P_n = \begin{cases} P_0\rho_1^n & \left(0 \le n < k_1\right) \\ \frac{P_0{\rho_1}^{k_1-1}\left(1-1/\rho_1\right)}{1-1/\rho_1^{k_2-k_1+2}}\left(\frac{1-1/\rho_1^{k_2-n+1}}{1-1/\rho_1}+\frac{\rho\left(1-\rho^{n-k_1+1}\right)}{1-\rho}\right) & \left(k_1 \le n < k_2\right) \\ \frac{P_0{\rho_1}^{k_1-1}}{1-1/\rho_1^{k_2-k_1+2}}\left(\rho^{n-k_2+1}\left(\frac{\left(1-1/\rho_1\right)\left(1-\rho^{k_2-k_1}\right)}{1-\rho}-\frac{1-1/\rho_1^2}{1-\rho-\rho/\rho_1}\right)+\frac{\left(1-1/\rho_1^2\right)\left(1-\rho/\rho_1\right)}{1-\rho-\rho/\rho_1}\left(\frac{\rho_1}{1+\rho_1}\right)^{n-k_2+1}\right) & \left(n \ge k_2\right) \end{cases} \tag{A.59}$$

About the Author

Priyanga Chamara Gunaratne received the B. Sc. in Computer Science and Engineering degree from the University of Moratuwa in Moratuwa, Sri Lanka in 2000. He is presently a Ph.D. candidate in the Department of Computer Science and Engineering at the University of South Florida in Tampa, Florida. His research interests are in performance evaluation of computer networks and dynamic power management of networked devices. He is a student member of the IEEE and IEEE Computer and Communications Societies.