Improving Energy Efficiency of Internet Equipment

ZHI ZHAO

School of Electrical Engineering and Telecommunications

The University of New South Wales

A thesis submitted for the degree of

Master of Engineering (Research)

August 2012

Originality Statement

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Signed _____

Date _____

Copyright Statement

Signed _____

Date _____

Authenticity Statement

Signed _____

Date _____

Acknowledgements

It has been a wonderful experience for me to conduct research at School of Electrical Engineering and Telecommunications, University of New South Wales in the past two years. With many people's help and inspiration I reached my research goal and learnt skills that will benefit me during the rest of my life. This thesis would not have been possible without their great help and I would like to take this opportunity to express my gratitude to all of them.

First and utmost, I would like to thank my supervisor Vijay Sivaraman for his constant guidance, patience, inspiring ideas and encouragement throughout my candidature. He is very good at visualizing problems and providing in-depth analysis. He spent a lot of time to chat with me and offered his constructive comments and advices on my work. I sincerely thank him for providing me with this research opportunity. It has been a great honour for me to conduct my research under his supervision. Without his great support and assistance I would not have been able to overcome all difficulties in my study. I hope to have other opportunities to cooperate with him in the future.

Second, I would like to thank my co-supervisor Craig Russell in the Commonwealth Scientific and Industrial Research Organisation (CSIRO). He helped me build all experiment networks and provided me with unlimited access to the facilities related to my work. With his help I could complete all experiments in NetFPGA routing platform in CSIRO's lab. The whole thesis would not have been possible without his continuous assistance. In addition I am also grateful to CSIRO for their student scholarship, which helped me support my life during the study.

I have had a very good time to cooperate with Arun Vishwanath in the past years. With his solid support I solved many problems in my research. He is an excellent research student and has many wonderful ideas in his area, which also inspires me very often. I am really grateful to him and feel very lucky to have such partner during my candidature.

ACKNOWLEDGEMENTS

I also want to express my appreciation to Phil Allen for processing all my computer related requirements. His quick response always helped me solve problems in using lab facilities and setting up test environments. Thanks also go to the fellow students in office 318, I benefit a lot from the discussion and communication with them. Their friendship has made my research life a very happy experience.

Furthermore, I sincerely thank the Australian Federal Government for its financial help —the Australian Postgraduate Awards (APA) scholarship, with which I did not have any financial problems during my study.

Finally, I would like to thank my family for their unconditional support. I would not have been able to complete my study without their love, encouragement and assistance. I sincerely thank my parents for their endless love and help for me to pursue my happiness. I am also grateful to my beloved wife for her sacrifices for the whole family and hard work in raising our baby. I feel very lucky to have her to be my lifetime partner.

Abstract

Improving energy efficiency of Internet equipment is becoming an increasingly important research topic, motivated by the need to reduce energy costs (and Carbon footprint) for Internet Service Providers, as well as increase of power density to achieve more switching capacity per-rack. To clearly understand how power consumption of Internet equipment is determined by the network elements such as sleep or active states of the device components, number of active ports or links, traffic patterns and network architectures, researchers have profiled the energy demand of commercial routing platforms and proposed ways to reduce network power consumption. However, these early works usually profile only coarse-grained power models (i.e., at the granularity of per line-card or per port) and the proposed power saving solutions involve significant architectural and/or protocol changes in the network. The cost and risk associated with such drastic changes increase the barrier to adoption by network operators, thus stretching the time-horizon at which they become practical for wide-scale deployment.

In this thesis we profile fine-grained power models for Internet equipment and propose a power saving scheme that requires minimal changes to existing router design, carries little risk of impacting network performance, is almost entirely transparent to network operators and is ready for incremental deployment.

We first profile power consumption of NetFPGA, an increasingly popular routing platform for networking research due to its versatility and low-cost, by conducting several experiments that allow us to decompose the power consumption of the NetFPGA routing card into fine-grained per-packet and per-byte components with reasonable accuracy. This work opens the doors for estimating network-wide energy footprints at the granularity of traffic sessions and applications (e.g., due to TCP file transfers), and provides a benchmark against which energy improvements arising from new architectures and protocols can be evaluated.

Second, we analyse the power consumption of Energy Efficient Ethernet (EEE)

switches in several experiments and based on the results propose a power model to profile them. Energy Efficient Ethernet is an IEEE standard (802.3az) for improving energy efficiency in Ethernet devices, which was newly issued in Sep. 2010. Our work is the first evaluation of power consumption of EEE switches. The proposed model can be used to predict the energy savings when deploying the new switches and also for research on further power saving techniques such as energy efficient routing or dynamic link shutdown.

Third, we propose a simple and practical algorithm for activating buffers in backbone router line-cards incrementally as needed and putting them to sleep when not in use. We evaluate our algorithm on traffic traces from carrier and enterprise networks, via simulations in ns2, and by implementing it on a programmable-router test-bed. Our study shows that much of the energy associated with off-chip packet buffers can be eliminated with negligible impact on traffic performance. Dynamic adjustment of active router buffer size provides a low-complexity low-risk mechanism of saving energy that is amenable for incremental deployment in networks today.

Contents

List of Figures

LIST OF FIGURES

List of Tables

List of Publications

1. A. Vishwanath, Z. Zhao, V. Sivaraman and C. Russell "An Empirical Model of Power Consumption in the NetFPGA Gigabit Router", IEEE Advanced Networks and Telecommunication Systems (ANTS), Mumbai, India, Dec 2010.

2. V. Sivaraman, A. Vishwanath, Z. Zhao and C. Russell, "Profiling Per-Packet and Per-Byte Power Consumption in the NetFPGA Gigabit Router ", IEEE INFOCOM Workshop on Green Communications and Networking (GCN), Shanghai, China, Apr 2011.

3. P. Riviriego, V. Sivaraman, Z. Zhao, J.A. Maestro, A. Vishwanath, A. Sanchez-Macian and C. Russell, "An Energy Consumption Model for Energy Efficient Ethernet Switches", International Workshop on Optimization Issues in Energy Efficient Distributed Systems (OPTIM 2012), Madrid, Spain, Jul 2012.

4. A. Vishwanath, V. Sivaraman, Z. Zhao, C. Russell and M. Thottan, "Adapting Router Buffers for Energy Efficiency", ACM SIGCOMM CoNEXT, Tokyo, Japan, Dec 2011.

5. Z. Zhao, V. Sivaraman, H. Habibi, C. Russell, "Joint Buffer & Link Bundle Adaptation for Energy Efficiency", 2012 (Under Preparation)

Chapter 1

Introduction

1.1 Background and Motivation

Internet traffic has witnessed exponential growth over the past decade, currently in the Exabyte range (10^{18} bytes) per year, and projected to reach Zettabytes (10^{21} bytes) in the next 5 years [5]. To cope with such high traffic demand, Internet Service Providers (ISPs) deploy routers that can today switch data at hundreds of gigabits-per-second, drawing tens of KiloWatts of power [6]. Not only does this power account for a significant fraction of the ISPs operational expenses, but also the high power density necessitates complex cooling systems to manage heat dissipation. Furthermore, though routing equipment is becoming more power efficient, the increase in efficiency is outpaced by annual increase in throughput capacity [7], meaning that the problem is likely to worsen with time. This trend presents grave concerns, mounting pressure on ISPs and equipment vendors to go green and energy efficiency in core networks has become a hot research topic in recent years.

The energy efficiency problem has motivated major chip vendors, equipment manufacturers, service providers and academic researchers world-wide to collectively find ways to manage and reduce the power consumption of core networks. The problem needs solutions at multiple levels, ranging from more efficient chips and components, to higher-level power management techniques that turn off (or underclock) components and sub-systems at certain times, or even redesign the network structures and protocols for power efficiency:

- Routers and switches are main equipment used widely in many networks. A simple router usually consists of a central CPU, some interface cards, shared memories and PCI bus. More advanced routers are composed of line cards

that have packet processing engines and memory buffers, switching fabrics and central processor [8]. To reduce the power consumption of a router, power efficiency in all router's components should be considered. An approach named power efficient designing was proposed in 2009, which suggested a series of power saving concerns in router hardware [9]. The approach argued that better power efficiency can be achieved by adopting aggregated semiconductor chips as packet processing engines, high speed memories to store tables, high speed router backplane transmission technology to increase the bandwidth between packet processing engines and interface cards, and a centralised switching fabric with multiple packet processing engines to reduce the power consumed during packet processing. Another study analysed the power efficiency of four widely-used switch fabric architectures and summarized the relationship between power consumption of switch fabrics and router throughput, number of active ports and fabric architectures [10], which revealed the principles to choose switch fabrics with power efficiency awareness.

- Besides the efforts to reduce hardware power consumption, researchers also focus on developing high-level power management techniques that turn off router components under low utilisation. To turn off components or sub-systems of routers, it is very important to decide what parts can be switched off, when the shutting-down operation should be taken and how long the sleeping status should last. Furthermore, the impact that a sleeping operation brings to the performance and the quality of service (QoS) of the whole network is also a key factor to consider. In their study about Internet energy efficiency [8], Gupta and Singh analysed the possibility to turn off line cards, memories and switch fabric or clock the main processor slower in a router and reached the conclusion that it is feasible to put these components to sleep but serious consideration about the impact to network protocols should be taken. In a subsequent research they proposed an algorithm that can dynamically shut down the Ethernet links according to buffer occupancy and mean inter-arrival time of packets [11], which presents a significant power savings when the traffic load is lower than 5% in most cases. Since there are more than three billion Ethernet interfaces in use worldwide today, improving energy efficiency of Ethernet ports will lead to huge financial benefits and power savings [3]. To fulfill this purpose a new IEEE standard 802.3az named Energy Efficient Ethernet (EEE) was issued in Sep. 2010, of which the main idea is to put the very

low utilisation link into a sleeping state called Low Power Idle (LPI) to save significant power. With the implementation of EEE technology in more and more Ethernet devices after the standard was released, the idea of turning off low-utilised components or sub-systems will bring huge power saving benefits that is estimated over $410 million in U.S. or over $1 billion globally per year [3].

- In addition to the aforementioned power saving approaches, energy efficiency can also be improved by redesigning the network structures and protocols. The power consumption of four conventional network architectures were analysed in [12]: electronic circuit-switched nodes, electronic packet-switched nodes, optical circuit-switched nodes and optical packet-switched nodes. The conclusion of this study suggested that generally circuit-switched nodes consume much less power than packet-switched nodes so that an efficient combination of circuit & packet switching in core nodes can achieve better energy efficiency than pure packet-switched networks. In a study on involving energy efficiency in both network design and protocol implementations [13], the authors investigated how to deploy routers with different chassis/line card configurations to minimise the power consumption of whole network while all performance requirements are still met, and how to manage network-wide traffic routing to put low-utilised hardware to sleep in protocol design. Their work showed the necessity of considering power efficiency during the period of designing network configurations and protocols, and also described the directions and methods for power awareness implementations.

In this thesis we make contributions on power saving in core networks by improving energy efficiency of Internet equipment. We believe that huge energy and operation cost can be saved by deploying Internet equipment that involve power saving concerns in their design and producing procedures. We create fine-grained power models for NetFPGA - an increasingly popular routing platform for networking research, profile newly-produced Energy Efficient Ethernet switches and propose an practical algorithm for activating router buffers incrementally as needed and putting them to sleep when not in use.

Our work is motivated by three challenges in energy efficiency research: first, many researchers have measured energy consumption of commercial routing platforms, using which they have developed many models for revealing the factors that determine the power consumption of a router. But these models usually are coarse-

grained and at the granularity of device, line-card or port, which is not suitable for the research requirement at the granularity of traffic sessions and applications (e.g., TCP file transfers). Our work focuses on obtaining fine-grained energy measurements of per-packet and per-byte energy for a routing system, which typically cannot be measured because of high confidence in commercial routers today and provides a benchmark against which energy improvements arising from new architectures and protocols can be evaluated. Second, the power saving schemes proposed in the previous literature usually involve significant architectural and/or protocol changes in the network, which will lead to huge amount of implementation cost and service quality risk to network operators and thus is less practical for wide-scale deployment. The algorithm we propose for buffer adaptation in this thesis requires minimal changes to existing router design, carry little risk of impacting network performance, are almost entirely transparent to network operators, and are ready for incremental deployment. Third, commercial routers do not offer sufficient flexibility (or mechanisms) for experimenting with new power-optimised architectures and algorithms. To maintain the stability of the commercial networks, ISPs are also very negative in implementing any experiments in their networks. Therefore, most researchers can only prove their ideas with pure theoretical analysis and/or simulations, not real-time implementations in hardware that can provide their research with solid support. To strengthen our arguments, we prove our research innovation with not only common methods like analysis and simulations, but also real-time hardware implementations in NetFPGA routing platform so that the feasibility, complexity and implementing risk of our proposals can be clearly evaluated by network operators.

1.2 Contributions of This Thesis

In the context of improving energy efficiency of Internet equipment, our primary contributions in this thesis are summarized below:

1. We decompose the energy consumption of the NetFPGA routing card into fine-grained per-packet and per-byte components with reasonable accuracy. We devise a series of experiments that allow us to quantify the per-packet processing energy, per-byte energy for receipt and storage at the ingress to the router, as well as per-byte energy for queueing and transmission at the egress of the router. To the best of our knowledge, ours is the first work that gives

such a fine-grained profile of energy consumption on the NetFPGA platform. This profile opens the doors to deducing network-wide energy footprints at various levels such as traffic sessions, applications, or user-groups. Researchers who prototype new architectures (e.g., dynamic speed scaling or sleeping) and protocols (e.g., power-aware TCP) for energy-savings in the NetFPGA routing platform can use our profile as a benchmark to quantify the improvements their mechanisms can realize.

2. We present an in-depth analysis of the power consumption profile of small EEE switches. We measure the power consumption of EEE switches in different scenarios such as changing number of active ports when no traffic, changing traffic loads, varying burstiness while keeping data rate unchanged, and connecting an EEE switch to a EEE-enabled or non-EEE switch. Based on the results a model for the energy consumption of EEE enabled Ethernet switches is proposed. This model can be used by researchers who want to explore higher layer energy saving mechanisms like energy efficient routing or selective link deactivation. This would allow them to make realistic estimates of the savings that would be achieved by those techniques in Ethernet networks. To the best of our knowledge, ours is the first evaluation of power consumption of EEE switches.

3. We argue that router buffer size can be adapted dynamically to track buffer usage, allowing much of the off-chip buffer memory to be put to sleep when not needed, thus saving energy. We first argue that the energy costs associated with always-on packet buffers in todays routers are nearly 10%. We then present empirical evidence from several carrier and enterprise networks that link loads (and by inference buffer occupancies) are low for a large proportion of the time, which presents an opportunity to adapt router buffer size dynamically to save energy. We propose a practical mechanism for dynamic adjustment of buffer size. We quantify the impact of our scheme in terms of energy savings and loss/throughput. We apply the algorithm off-line to traffic traces derived from Internet link data. We then simulate it in ns2, and profile its impact on TCP performance. Finally, we implement dynamic buffer adaptation in the gateware of the NetFPGA-based Gigabit router platform, and demonstrate its feasibility in a test-bed with realistic traffic.

1.3 Thesis Organizations

The rest of this thesis is organized as follows.

Chapter 2 presents an overview of previous works related to this thesis. We first introduce some general solutions to improve energy efficiency of Internet equipment in core networks. Following this we review the achievements on new IEEE standard Energy Efficient Ethernet. Then we highlight the contributions to set appropriate buffer size in Internet routers.

In chapter 3 we profile per-packet and per-byte energy consumption of NetFPGA routing platform. We first introduce NetFPGA system with detailed description of hardware/software architectures, then propose a fine-grained power consumption model. Further, we present experiment setup and analysis of power measurement and finally achieve power consumption of all model elements.

In chapter 4 we conduct a series of experiments to measure power consumption of EEE switches and based on the results we propose the first power model for EEE switches.

In chapter 5 we discuss a dynamic router buffer adaptation algorithm for power saving purpose. We investigate energy consumption of router buffers, analyse link utilisation in operational networks and buffer occupancy fluctuation, and reach a conclusion that link loads in operation networks are very low for much of time and buffer occupancy seldom exceeds a few tens of KB, which leads to the idea of saving power by putting low-utilised buffer to sleep. We then introduce a three-level hierarchical buffer structure and a simple energy model for it. Further, we discuss in detail an algorithm for dynamic buffer adjustment. In the last part of this chapter we evaluate this algorithm with offline trace analysis, online ns2 TCP simulation and real-time implementation in NetFPGA routing platform.

Finally, we reach the conclusion of this thesis in chapter 6 and discuss the directions for future work.

Chapter 2

Literature Review

2.1 Introduction

In recent years there have been many proposals to reduce the energy consumption of Internet equipment. The achievements covering multi levels of the networks range from the adoption of energy efficient hardware to optimised mechanisms to turn on/off certain components like ports or buffer, or even network wide modifications on routing protocols and architectures. In this chapter we comprehensively review the literature related to our work in three aspects:

- General solutions to improve energy efficiency in core networks: The review introduces several proposals to involve power saving concerns in design of network architecture, protocol and implementation. Then the methods to save power by changing link rates, shutting down idle links and adapting link bundles are also reviewed.
- Energy Efficient Ethernet: The review involves the introduction of the Energy Efficient Ethernet standard, the efforts to evaluate the performance of EEE before and after the standard release, the methods proposed to improve energy efficiency of EEE and real power measurement analysis of EEE equipment.
- Sizing router buffers: The review covers several well-known rules to set buffer size based on link utilisation requirement, then other proposals to relate buffer size with packet loss probability. The efforts to adapt buffer size according to link utilisation or incoming traffic are also reviewed.

The rest of the chapter is organized as follows. In section 2.2 we take an overview of previous achievements in energy efficient network design and Ethernet link adap-

tation. Then we present detailed reviews on the literature specifically related to our work. In section 2.3 we introduce new IEEE standard 802.3az and achievements to improve its energy efficiency. In sec 2.4 the research to set a fixed buffer size under certain constraints is summarized and the dynamic adjustment of buffer size is also reviewed. Finally we summarize this chapter in section 2.5.

2.2 General Solutions to Improve Energy Efficiency in Core Networks

To improve energy efficiency in core networks, researchers make their contributions in different areas. Some studies focus on deploying energy efficient components in routers. Some research proposes energy awareness in design of network architecture and protocol. Some efforts are put to change link rates for power saving and much attention is paid to shut down idle links under low utilisation.

2.2.1 Power Awareness in Design of Network Architecture and Protocols

Constructing networks with an energy efficient architecture will significantly save power. In an analysis of network architectures [12], the author proposed four architectures to be considered as representatives of future high-capacity network nodes: packet-switched electronic core node, packet/burst-switched optical core node, circuit-switched electronic core node and circuit-switched optical core node. He analysed the power consumption of four nodes and the results showed that generally electronic nodes consume more power than optical nodes; among the four proposed architectures the optical circuit-switched architectures based on micro-electromechanical system (MEMS) switching devices are the best one to choose; the power consumed by optical circuit-switched nodes is less than 10% of that consumed by optical packet-switched architectures; the power consumed by electronic circuit-switched nodes is 43% less than that consumed by electronic packet-switched architectures. To reduce power consumption at core nodes, his conclusion suggested that the dynamic circuit switching or a hybrid switching (combining packet, circuit and burst switching) should be considered to form the network. The work in [10] presents the effort to save power by choosing energy efficient switch fabrics. The authors proposed power models for three key components of switch fabrics: node

switch, internal buffer and interconnect wires, and then analysed four widely used switch fabrics with these models: crossbar switch fabrics, fully-connected network, banyan network and batcher-banyan Network. The simulation result showed that the structure of fully connected switch has the best energy efficiency among the four architectures; the power consumption of Banyan network concentrates on internal buffers while the power consumed in other three architectures has a linear relationship with the traffic throughput; generally the power consumption of switch fabrics is dominated by internal node switches when number of ports are small and if number of ports is large it will be dominated by interconnect wires.

In addition to focus on power saving in network architectures and switch fabrics, considering energy efficiency as a key factor in network and protocol design is another hot subject which attracts much research attention. The work in [13] proposes the idea of involving power awareness in network and protocol design and provides an experimental method to estimate power consumption in some test networks with power-aware implementation. In their tests for power awareness in network design, the authors changed provisioning requirement by scaling the traffic to find the right hardware configuration that can satisfy the provisioning requirement and minimise the power consumption. For power awareness in protocol design, the authors fixed the hardware configuration and tried to find out how traffic flows should be routed in order to turn line cards or chassis into sleep when utilisation is low. The analysis in [13] did not lead to any concrete mechanisms on how to implement power awareness in network and protocol design, while the studies in [9; 14] suggested some practical approaches. In [14] a distributed routing protocol named General Distributed Routing Protocol for Power Saving (GDRP-PS) was proposed to put some underutilised routers into power saving mode without any impact on network performance and connectivity. GDPR-PS uses total link utilisation of all links connected to one router as the input to make sleep decision. It is designed to be compatible to any existing distributed routing protocols. As an example the authors analysed in detail how GDPR-PS can cooperate with Open Shortest Path First (OSPF) protocol . Simulations were also conducted to evaluate its performance and results showed that up to 47.5% power saving can be achieved for a router and this number is up to 18% for the whole network. The work in [9] suggested to improve power saving of routers not only by adopting more energy efficient components but also by involving some energy saving considerations in network design. The authors presented power saving ideas of turning off unused slots (or ports) in a router and switching working

frequency of line card engines to a lower level. They also mentioned Energy Efficient Ethernet and link bundle adaptation, which we will discuss in detail next.

2.2.2 The Adaptation of Link Rates and Bundle for Energy Efficiency

The widely adoption of Ethernet devices in Internet all over the world has led to great energy consumption of Ethernet links that is growing rapidly because the default link speed increases from 100Mbps to 1 Gbps now and to 10Gbps in the future. Researchers have found that almost same amount of power is consumed when a link is idle or fully utilised, which motivates the research efforts to reduce power consumption of Ethernet links by lowing the link rate instead of decreasing the link utilisation.

2.2.2.1 Link Rate Adaptation

The work in [15] shows that Operating Ethernet links in 10Mbps or 100Mbs data rate will save about 4W per link comparing to the power consumed by 1Gbps links, while this number is 10-20W savings per link for a 10Gbps case. This result motivated the authors in [15] to propose a method to scale the link rate according to the queue length in the buffer and link utilisation. The method simply judges the queue occupancy with two thresholds (qHigh and qLow) and one link utilisation threshold. If the queue length is bigger than qHigh then the low link rate is adjusted to a high value (e.g., from 10Mbps to 100Mbps); if the queue length is less than qLow while the link utilisation is also lower than the utilisation threshold then the low link rate is set (e.g., from 100Mbps to 10Mbps). The simulation to evaluate the method presented that it is possible to make the link work in a low rate for a big portion of time without significant influence to the network performance.

The idea in [15] is only an initial proposal of adapting link rate by buffer size. The advanced version of the method was proposed as Adaptive Link Rate (ALR) in another study of the same group [16]. The authors in [16] first suggested using Ethernet MAC frame instead of auto-negotiation to be the mechanism that both ends of one link adopt to inform the rate change to the other side. Then ALR was studied in two cases: single threshold and dual thresholds. The authors set up Markov models to evaluate both ALR cases and simulated ALR as a state-dependent service rate single-server queue with Poisson arrivals and exponential service time.

Another simulation was also conducted with trace traffic from two real networks for two link rates (10Mbps and 100Mbps). The result for trace traffic showed that the link with ALR operates in low rate in more than 99% of the total running time and causes negligible packet delay.

When the packet arrival rate is high enough to make queue length exceed qHigh in low link rate, but not high enough to keep the queue length bigger than qLow under high link rate, the dual-threshold ALR will have oscillation between link rates that will increase packet delay. To solve this problem the authors improved ALR by proposing two new policies: the utilisation-threshold policy and time-out-threshold policies [17]. The utilisation-threshold policy introduces a new threshold used to monitor the number of bytes sent in a certain interval tUtil, and triggers the link rate transition when (qLen $<$ qLow) and (uBytes $<$ uThresh). This policy requires modifications to the existing devices to count the packet arrival numbers, which is not acceptable to some users. The authors then suggested the time-out-threshold policy, which keeps the link stay in low or high rates for a certain time (tMinLow in low rate and tMinHigh in high rate) before switching to other rates. The group designed A series of simulation experiments to evaluate dual-threshold, utilisation-threshold, and time-out-threshold ALR policies. They finally concluded that generally the utilisation-threshold policy should be used, but if this policy is not applicable due to the implementation complexity the time-out-threshold should be adopted. If the traffic is smooth the dual-threshold policy should never be used to avoid its inherent oscillation.

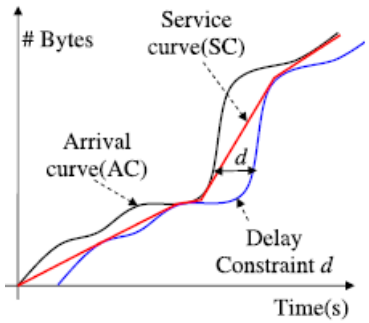


Figure 2.1: An illustration of delay-constrained service curves and the service curve minimising energy [1]

The above work adapts link rate mainly according to the buffer length. A study analysing the efficiency of sleep and rate-adaptation suggested a new method called practical rate adaptation (practRA) [1], which serves the packet at a constant rate

and only change link rate when the arrival curve (AC) intersects the latest-departure curve (AC+d) in Fig.2.1. This method has potential to save power but is very difficult to be implemented in the real networks.

2.2.2.2 Methods for Dynamic Link Shutdown and Link Bundle Adaptation

A special case for rate adaptation is to decrease the link rate to zero, which means putting an idle link to sleep or totally shutting down a link. In their study to reduce power consumption of Internet, Gupta and Singh [8] analysed the feasibility to put some or all components of a router to sleep state and suggested two sleeping approaches: Coordinated Sleeping in whole network that needs the routing protocol changes to concentrate traffic into several routers and enable other router links to sleep; uncoordinated sleeping that a router decides which interfaces to be put to sleep based on local decision. They discussed the possible impact of two approaches to the switch protocol, OSPF and IBGP routing protocols. The conclusion was that their proposal needs some significant modifications to the existing protocol specification and Internet architecture. In their subsequent research [11; 18], they proposed concrete methods to utilise the inactivity of the idle links. In [18] they developed an algorithm that shuts down the complex transceiver circuitry when certain buffer occupancy, traffic arrival time and maximum bounded delay are considered. They extended this algorithm in [11] and named it as Dynamic Ethernet Link Shutdown (DELS) . The result of DELS evaluation presented that DELS can save significant power when the link load is less than 5% in most cases.

The contributions in [1] include not only a link rate adaptation method but also a practB&B algorithm to put idle network components to sleep. The practB&B buffers the traffic received in an interval B ms in the edge networks into bursts destined to different egress routers one by one and sends these bursts together as a bunch so that other routers within the network can sleep during the interval of these bunches. The authors modeled the power consumption of sleep policy and link rate adaptation, compared the power savings made by two approaches and concluded that there is a utilisation threshold below which sleeping performs better than rate adaptation and above which rate adaptation can save more power.

To achieve bigger capacity without upgrading the link speed, pairs of routers are usually connected by a logical link bundle that consists of two to twenty cables. Very recently researchers started to develop methods to save power in link bundles

by shutting down cables during the low utilisation period [19; 20; 21]. The authors in [19] assumed a network management system that can selectively shut down cables in bundled links according to the traffic profile. Given the network topology and traffic matrix as input the system should solve an optimisation problem to maximize the number of cables to be powered off while still ensure enough provisioning to serve the traffic. They proposed three methods as the optimal solutions. The simulation results presents similar energy efficiency among three methods thus the authors preferred to introduce the simplest one to network operators. Different from [19], the work in [20] provided a method that locally makes the decision to shut down or bring up one link according to the link utilisation, not instructions from an extra management system. This method was named as Fixed Local Heuristic Threshold (FLHT) . It adopts one fixed link utilisation threshold (the threshold is 90% in the simulation) to decide the link operation and deactivate/activate one link each time. To evaluate FLHT the authors used real trace data from Internet2 (a research network in America) as traffic input and the simulation result based on 30-day traffic data showed that 86% energy saving can be achieved for a "90%"threshold setting but there is low overflow risk. To improve the performance of FLHT the authors proposed a better solution called Dynamic Local Heuristic Threshold-based algorithm (DLHT) [21]. DLHT considers not only the link utilisation threshold to make link operation decision, but also the number of activated sublinks (1-3) to reduce the overflow risks. The performance of different combinations of link utilisation threshold and number of activated sublinks as input in DLHT was evaluated in simulations with Internet2 trace data. The result presented a fact that different threshold settings (60%, 80%, 90%) will achieve very small difference in power saving but low threshold will reduce the number of super overflow events. Another fact was also shown in the simulation that activating more sublinks per time will greatly reduce the overflow risks but the cost is lower power efficiency as well.

2.3 Energy Efficient Ethernet

In previous section we highlight some general achievements to improve energy efficiency in core networks. In this section we will introduce the literature specifically related to our work. We first provide an overview of a new IEEE standard 802.3az - Energy Efficient Ethernet, and brief the efforts to improve the performance of this standard in Ethernet equipment.

2.3.1 Overview of Energy Efficient Ethernet

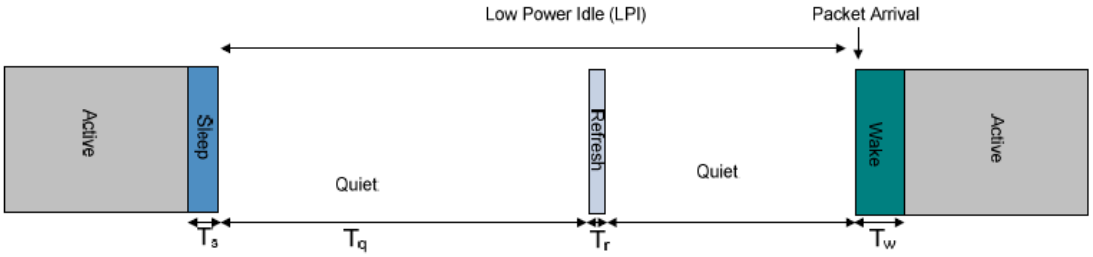


Figure 2.2: Mode transitions in Energy Efficient Ethernet

For many years, Ethernet has been the dominant technology for wire-line LANs. It is widely used in residences and commercial buildings and almost all computers include an Ethernet connection and in some cases more than one. Although Ethernet supports a variety of transmission media, most of the Ethernet ports are connected by Unshielded Twisted Pairs (UTP) , especially in homes and offices. For UTP, Ethernet currently supports four data rates: 10 Mb/s (10BASE-T), 100 Mb/s (100BASE-TX), 1 Gb/s (1000BASE-T) and 10 Gb/s (10GBASE-T). For 100 Mb/s and higher data rates, Ethernet physical layer transmitters transmit continuously to keep transmitters and receivers aligned. When there is no data to send an auxiliary signal called IDLE is sent. This means that most of the elements in the interfaces are active at all times leading to an energy consumption that is large and independent of the traffic load.

To improve the energy efficiency of Ethernet devices, IEEE issued 802.3az standard in 2010 for Energy Efficient Ethernet (EEE). This standard introduces the concept of Low Power Idle (LPI) which is used instead of the continuous IDLE signal when there is no data to transmit [22]. LPI defines large periods (T_q) over which no signal is transmitted and small periods (T_r) during which a signal is transmitted to refresh the receiver state to align it with current conditions. The operation of the LPI mode is illustrated in Fig. 2.2. The energy consumption of a physical layer device (PHY) when it is in LPI mode is expected to be significantly lower than when it is in the active mode.

The whole process to develop IEEE 802.3az is summarized in [3]. A tutorial presented to IEEE 802 Working Group in July 2005 first mentioned the idea of issuing a new standard specifically for Energy Efficiency in Ethernet. After many meetings, discussions and presentations, IEEE 802.3 working group began a new

project called Call for Interest (CFI) for EEE, which achieved a successful vote in the panel of IEEE 802.3 working group in November 2006. An EEE study group was then set up and had their first meeting in January 2007. The group finalized the decision to set up a project for creation of EEE standard with the support concluded from six meetings and 43 presentations. An authorization to the project was issued in September 2007 and the group then formed P802.3az Task Force. At that time there were two technologies to be considered: Adaptive Link Rate and Low Power Idle. The Task Force finally chose LPI and completed the first draft of the standard in October 2008. After that many modifications were made to the draft and a series of review and balloting processes were held until the final IEEE 802.3az standard was approved on September 30, 2010.

Protocol	Min T_w (µsec)	Min T_s (µsec)	Frame size (bytes)	Min T_{frame} (µsec)	Single Frame efficiency	Frame size (bytes)	Min T_{frame} (µsec)	Single Frame efficiency
100BASE-TX	30.5	200	1518	120	34.2%	64	5.1	2.2%
1000BASE-T	16.5	182	1518	12	5.6%	64	0.5	0.3
10GBASE-T	4.48	2.88	1518	1.2	14.0%	64	0.05	0.7%

Table 2.1: Minimum wake, sleep, frame transmission times and single frame efficiencies for different link speeds

The actual energy savings of EEE on a given link depend on the amount of time that the link spends in LPI mode. This time can be reduced by the transition overheads associated with activating (T_w) and putting it into LPI mode (T_s). During those transitions, there is significant energy consumption and the transition times are large compared with the frame transmission time [23; 24]. The transition times for the different speeds are summarized in Table 2.1 and compared with the frame transmission times for a 1518 byte and a 64 byte packet. To measure the efficiency of EEE, the concept of Single Frame efficiency (SF_e) which measures the efficiency of EEE for single frame transmission is introduced. When a single frame is transmitted the link has to be activated to send a frame and then deactivated after the transmission. Therefore for a frame transmission time of T_f, the link is active or in transitions for $T_w + T_s + T_f$. The ratio of both times is defined as the single frame efficiency: $SF_e = T_f/(T_w + T_s + T_f)$. It can be observed that the values for the Single Frame efficiencies in Table 2.1 are low. This results in an energy consumption versus load profile that tends to saturate at medium or low loads unless packets are

coalesced [23], thus motivates researchers to develop new mechanisms to improve the performance of EEE.

2.3.2 Methods to Improve the Performance of Energy Efficient Ethernet

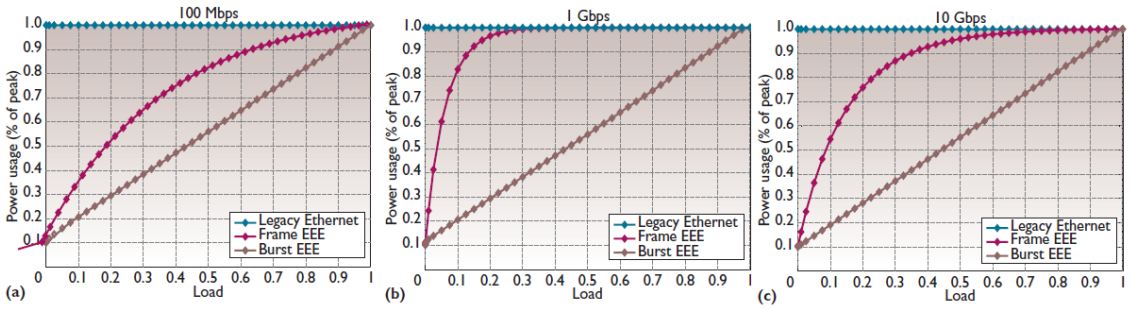


Figure 2.3: Energy consumption vs. traffic load when using burst transmission at (a) 100 Mbps, (b) 1 Gbps, and (c) 10 Gbps [2]

Before the standard for EEE was finalized in 2010, researchers had started to evaluate the performance of EEE and proposed their methods to improve it. A study made in 2009 [23] tried to reveal the relationship between the power consumption of EEE and traffic load. The authors noticed the energy efficiency reduction brought by the transition overhead of activating and putting the PHY into LPI mode. They conducted simulations on power consumption of 100Mbps, 1Gbps and 10Gbps EEE links with different traffic loads to evaluate the influence of the overhead. The result showed that EEE works well for saving power when traffic load is low and links speed is slow. For 1000BASE-T and 10GBASE-T links the power consumption of EEE will saturate to the same level as legacy links very soon even the traffic load is low because the power is consumed mainly on wake-up and sleep transition of EEE links, not transmission of the actual data frames. In additional to the simulations, the authors also evaluated the performance of EEE with several measurement-based scenarios such as video downloading by residential users, file exchanges between two users connected to the same switch and university access link analysis. The results presented the fact that EEE will have low power efficiency when the link only transmits small frames and the load is low. Based on their achievements in [23], the same group extended their research and proposed a method called burst transmission to improve the energy efficiency of EEE [2]. The purpose of burst transmission

is to extend the time the links stay in LPI mode. The authors suggested that incoming data frames can be collected in a fixed period T_{as} and a buffer threshold should be used to prevent the buffer overflow. All gathered frames should be sent out of the links in case that one of the conditions are met: timer T_{as} expires or buffer threshold is reached. To evaluate the performance of the burst transmission, the authors repeated their simulations and measurement-based scenarios previously mentioned in [23] with the parameter settings of 10-ms T_{as} and 1000 data frames as buffer threshold. The result of three simulations for different link speeds is shown in Fig. 2.3. It shows that adoption of burst transmission greatly improves the energy efficiency of EEE and the relationship between energy and traffic load is very close to be proportional.

Scenario	Direction	Speed	$Energy_{frame}$ (% of peak)*	$Energy_{burst}$ (% of peak)*	Link load (%)	Average frame size (bytes)	Energy savings (%)
1. Residential user video download	Download	100 Mbps	12.75	11.57	1.43	1,444	9.25
	Upload	100 Mbps	10.99	10.34	0.04	90	5.91
2. Residential user file transfer	File	100 Mbps	78.68	74.25	71.13	1,499	5.63
	Acknowledgments	100 Mbps	44.92	11.91	1.39	77	73.49
3. University Internet access link	Download	1 Gbps	92.80	23.47	10.94	679	74.71
	Upload	1 Gbps	96.20	27.24	17.66	919	71.68
4a. Data center: file and search server	Input	1 Gbps	65.90	12.60	1.22	87	80.88
	Output	1 Gbps	72.92	57.73	52.21	1,497	20.83
4b. Data center: search server	Input	1 Gbps	45.28	18.85	8.51	945	58.37
	Output	1 Gbps	42.30	17.73	7.23	934	58.09
4c. Data center: file and application server	Input	1 Gbps	61.37	11.77	0.65	130	80.82
	Output	1 Gbps	57.10	14.72	4.02	749	74.22

Figure 2.4: Energy consumption estimates for different measurement-based scenarios [2]

Fig. 2.4 shows the performance of burst transmission in several scenarios. In the figure $Energy_{frame}$ is the energy consumed by normal EEE links and $Energy_{burst}$ is the energy consumed by links with burst transmission. From the figure the authors concluded that normal EEE can save power in high speed links with low loads, but the energy efficiency can be further improved by adopting burst transmission in case that the average frame size is small (e.g., the scenario 4a in the figure).

Based on the work in [2; 23], Christensen et al. improved the idea of burst transmission and proposed a method called packet coalescing after the release of 802.3az standard [3]. They used a FIFO queue in Ethernet interface to coalesce incoming packets and sent them as a burst of back-to-back packet group. There

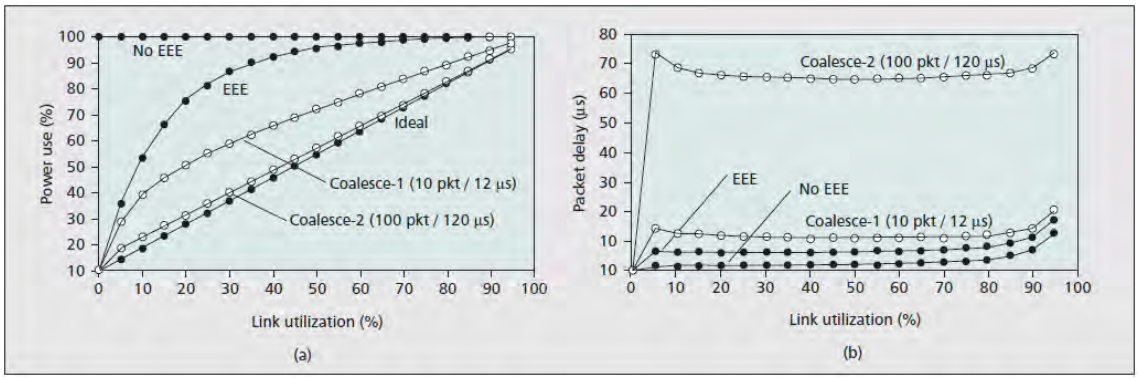


Figure 2.5: a) Energy use vs. link utilisation; b) packet delay versus link utilisation [3]

are two parameters that determine when to send the coalesced packets: a pre-defined packet counter and a timer starting from the arrival of the first packet in the coalescing queue. Either the timer expires or the maximum number of packet counter is reached will cause all coalesced packets to be sent out. The simulation of packet coalescing in a 10Gbps link shows a similar result with burst transmission in Fig. 2.5, which presents a nearly proportional relationship between energy consumption and traffic load when the parameters are big enough (case coalesce-2 in the Fig.). The result also shows that the packet coalescing will induce packet delay to the link so careful consideration should be taken when setting the parameters to balance the energy savings and performance requirements. In the analysis of the performance of packet coalescing for TCP file downloads, it shows that in the link direction that small packets like TCP ACKs are sent the EEE link will not save much power than legacy link if no packet coalescing is adopted because the link spends too much power on the overhead of wake-up and sleep transition caused by the small packets. With certain packet coalescing settings an EEE link can save 50% or more power than legacy link when transmitting small packets. Another contribution in [3] is the estimation of power saving in U.S. assuming EEE is widely adopted. The assumptions used to calculate the economic benefits was based on statistic data of link number and link utilisation in U.S. in 2008 and the authors also made some other estimations. The result presented a nearly $410 million power savings per year in U.S. by adoption of EEE and if packet coalescing is used an extra $80 million savings per year can be achieved.

2.3.3 Real Power Measurements of EEE Equipment

The researchers could only evaluated the performance of EEE with simulations and estimations before any EEE-enabled equipment available in the market. After the release of 802.3az standard Ethernet vendors started to produce new EEE-enabled equipment so that it is possible for researchers to evaluate EEE performance with real measurements from real hardware. An initial evaluation of such measurements was reported in [24]. The authors conducted a series of experiments to measure the power consumption of EEE Network Interface Cards (NICs). They first measured the power consumed when there is no traffic and full load in the EEE link. The result shows that in no traffic case EEE NIC can save significant power (over 70% for 1Gbps link and 30% for 100Mbps link). But in full load case EEE NIC consumes almost the same power as legacy NIC because the high traffic load makes the link work all time and hardly enter LPI mode. In the second experiment the authors limited the link load to a very low level and send only 250-byte packets through the link so that the packets are spaced by 200μs, which causes frequent mode transitions of EEE (entering or exiting LPI mode). The result presents a similar power consumption between EEE and legacy link, which proves the estimation in [23] that even the link load is low the EEE will not save much power in case that frequent mode transitions happen. In the third experiment the authors varied link load and measured the power. The result showed that if the load is higher than 6% the power consumption of EEE and legacy links will be the same, which is also caused by the transition overhead of EEE.

All the aforementioned work presents the fact that the performance of EEE is highly related to the traffic patterns that determine the transition overhead between active and LPI modes. Since 802.3az does not define how to make decisions to put or wake up the links into/from LPI mode, there are still a lot of opportunities open for researchers to improve energy efficiency of EEE devices.

2.4 Sizing Router Buffers

Buffers are widely used in Internet routers to store packets during the time of congestion. Buffer size is an important factor to be considered in design and performance assessment of routers. Under-buffered routers cause packet loss in the networks and decrease the application performance. Over-buffered routers increase queueing delay, consume more power and are more complex in router design and production.

There have been a lot of research efforts regarding the size of buffers in routers. One method known as the rule-of-thumb is widely used by router manufacturers and proposes that the buffer needed for a link can be calculated by the equation $B = RTT \times C$ [25], where RTT is the average round trip time and C is data rate of the link. The rule-of-thumb guarantees 100% link utilisation, which means when a link is congested there are still enough packets in the buffers to make the link busy so that no throughput is decreased. This well-known rule leads to big buffer size requirement in high speed routers because the size needed is proportional to the line rate. For example, a 40Gbps line card with 250 ms RTT will need to use $250ms \times 40Gbps = 10Gbits$ buffers (1.25Gbytes) [26]. To adopt such big buffers in high speed router design was a big challenge to vendors because if they use SRAM chips the cost is too high and the board consumes too much power; if they use DRAM chips the random access time of DRAM is too slow comparing with the packet processing speed. This challenge motivated the researchers in Stanford University to develop a new rule to reduce the required buffer size [26]. They showed that the rule-of-thumb works well for single long-lived TCP flows, but when the number of long-lived flows N in a bottleneck link is big enough (e.g., over 500) then all these flows are unsynchronized and $B = RTT \times C/\sqrt{N}$ buffers will be sufficient while the link utilisation is still kept nearly 100%. This new rule is referred as small-buffer model and it can greatly reduce the buffer size required in high speed routers.

A more aggressive proposal argued that routers with only a few dozen packet buffers can be built but the link utilisation will decrease 10-20% [27]. Since large optical buffers are not feasible nowadays, the recent achievement in optical buffer can only store a few dozen packets. Thus the tiny-buffer rule proposed in [27] is very suitable to be implemented in all-optical routers because the link capacity in optical network is very sufficient. This rule is also useful in electronic routers since less buffers will greatly reduce the complexity in router design.

The above work focuses on sizing router buffers according to the link utilisation requirement only. Packet loss rate is another constraint researchers like to consider for buffer size. The work in [28] derives the minimum buffer size of a drop-tail queue when given link utilisation, loss rate and queue delay constraints. The authors showed that for N heterogeneous TCP flows the minimum buffer size required is determined by the harmonic mean of their round-trip times and affected by degree of loss synchronization. To meet the loss rate constraint they presented that the minimum buffer size is proportional to the number of "bottlenecked" flows N if N is

bigger than a threshold (in the simulation the threshold is 30). The authors concluded their achievements to a buffer size formula named Buffer Sizing for Congested Links (BSCL) , which is applicable in the case that most of the traffic (80-90%) at the link is locally bottlenecked flows. They compared BSCL with rule-of-thumb and Stanford small buffer model, the result suggested that when the flow number is less than 20 both rule-of-thumb and small buffer model require much more buffers than BSCL. When there are large number of flows, the rule-of-thumb and small buffer model require much less buffers than BSCL but they will lead to significant loss, while BSCL can limit the loss rate in a given bound (typically 1%).

There are more studies on sizing router buffers regarding the packet loss probability in [29; 30]. In [29] Morris found that the loss rate will increase rapidly with the increase of the competing TCP flow numbers. If the buffer size is set based on rule-of-thumb it will not be sufficient and cause frequent timeouts, big variations in the throughput and transfer latency of competing TCP flows. In [30] he proposed an adaptive buffer sizing algorithm called Flow Proportional Queueing (FPQ) , which adapts buffer size in proportional to the number of active TCP connections. When traffic load is heavy, a traditional router will have low queueing delay with the cost of high loss rate. But the FPQ router will have a high queueing delay with low loss rate so that the loss rate is stable in the system.

More work on adapting buffer size subject to certain constraints (link utilisation or incoming traffic) was presented in [31; 32]. In [31] the authors proposed an Active Drop-Tail (ADT) algorithm to adapt the size of a drop-tail buffer to regulate the target utilisation while the queueing delay is minimised. The performance of ADT is related to the non-stationarity of traffic arrivals and poorly studied arrival processes. The authors solved the stability question by formulating it as a Lur'e problem so that they could transform the non-stationarity of the traffic patterns in the form of a sector bounded time-varying nonlinearity. An Adaptive Buffer Sizing (ABS) algorithm was proposed in [32], which adapts the router buffer size when utilisation and loss rate constraints are given. ABS is based on a fact that there exists a monotonic relationship between buffer size, link utilisation, loss rate and queueing delay. The authors compared ABS with Standford small buffer model and aforementioned BSCL. The results showed that when the flow number is small ABS achieves best link utilisation and when the flow number is large ABS keeps the target loss rate stable in a very low level, which still presents a better performance than the Standford model and BSCL. The authors concluded that ABS is applicable to

generic Internet traffic such as long-lived, short-lived TCP flows or non-TCP traffic.

2.5 Summary

In this chapter we provide a detailed review of the previous work to improve energy efficiency in core networks and specifically in EEE equipment and sizing router buffers. In the overview of general power saving solutions we notice that most of research efforts are spent in finding methods to switch off certain idle components (e.g., line cards and physical ports) of routers, especially Ethernet links for saving power, which attracts us to conduct research on energy efficiency in routers, Ethernet standard and buffer sizing. In the area of Energy Efficient Ethernet we find that the network interface cards are the only type of the EEE equipment whose power measurement is provided. Therefore, more power models for more types of EEE equipment are required by researchers. For sizing router buffers we find that no matter what rules are proposed to set buffer size operators in reality always build routers with large buffers. This motivates us to consider a method that adjusts buffer size in a way that will comfort the operators to use small buffers. In the rest of the thesis we will discuss these research topics in detail, beginning with power consumption analysis of the NetFPGA Gigabit router.

Chapter 3

Profiling Per-Packet and Per-Byte Energy Consumption in the NetFPGA Gigabit Router

Researchers propose power consumption models of Internet routers to provide benchmarks against which energy improvements made from novel power-optimised network architectures and algorithms can be evaluated. These previous works on power models are coarse-grained (i.e., at the granularity of per line-card or per port) and cannot meet the research requirements at the granularity of traffic sessions and applications (e.g., TCP file transfers). In this chapter we will present our fine-grained power model for NetFPGA routing platform and provide per-packet & per-byte energy components with reasonable accuracy.

3.1 Introduction

Internet traffic has witnessed exponential growth over the past decade, currently in the Exabyte range (10^{18} bytes) per year, and projected to reach Zettabytes (10^{21} bytes) in the next 5 years [5]. To cope with such high traffic demand, Internet Service Providers (ISPs) deploy routers that can today switch data at hundreds of gigabits-per-second, drawing tens of KiloWatts of power [6]. Not only does this power account for a significant fraction of the ISP's operational expenses, but also the high power density necessitates complex cooling systems to manage heat dissipation. Though routing equipment is becoming more power efficient, the increase in efficiency is outpaced by annual increase in throughput capacity [7], meaning that the problem

is likely to worsen with time. In recognition of the pressing need to address this problem, a consortium called GreenTouch [33] has recently been launched that aims to improve the ICT sector's energy efficiency by a factor of 1000 from current levels by 2015.

Early works such as [8] highlighted energy efficiency issues in wireline networks, and several recent works have proposed techniques for power-optimising network design [13], and saving energy by putting ports and line-cards to sleep or via adapting link rate [34]. Models of device energy consumption have been put forth [35] using measurements on commercial switches and routers. In this chapter, we focus instead on the NetFPGA experimental Gigabit routing platform. Our reasons for choosing this platform are twofold. **First**, commercial routers do not offer sufficient flexibility (or mechanisms) for experimenting with new power-optimised architectures and algorithms, whereas the NetFPGA router is an open reprogrammable hardware platform that is increasingly being used by networking researchers world-wide to rapidly prototype and evaluate new mechanisms. By benchmarking energy performance of this platform, it becomes possible to quantify the energy gains from new mechanisms for improving power efficiency. **Second**, commercial platforms provide only coarse-grained measurements of power (at the granularity of device, line-card or port), whereas in this chapter we focus on obtaining *fine-grained* energy measurements of per-packet and per-byte energy, which typically cannot be measured with high confidence in commercial routers today.

Our main contribution is to isolate the energy components associated with the various operations in the NetFPGA router. We use a high-precision hardware-based traffic generator and analyser that enables us to tightly control the stimulus to the router. In conjunction, we use a high-fidelity multi-channel digital oscilloscope that probes and records the instantaneous power draw of the NetFPGA router card 50 million times-per-second. Using these, we devise a series of experiments that allow us to quantify the per-packet processing energy, per-byte energy for receipt and storage at the ingress to the router, as well as per-byte energy for queueing and transmission at the egress of the router. To the best of our knowledge, ours is the first work that gives such a fine-grained profile of energy consumption on the NetFPGA platform, which will be of great value to the community. This profile opens the doors to deducing network-wide energy footprints at various levels such as traffic sessions, applications, or user-groups. Researchers who prototype new architectures (e.g., dynamic speed scaling or sleeping) and protocols (e.g., power-

aware TCP) for energy-savings in the NetFPGA routing platform can use our profile as a benchmark to quantify the improvements their mechanisms can realise.

The rest of the chapter is organised as follows. In Section 3.2, we describe the NetFPGA Gigabit router, outline the life of a packet through the router, and present a simple linear model for the power consumption. Our main contribution is in Section 3.3 that isolates the energy components based on a series of experiments, and discusses its implications. Finally the chapter is concluded in Section 3.4.

3.2 The NetFPGA Gigabit Router

3.2.1 Introduction of the NetFPGA Platform

Figure 3.1: An vertical view of the NetFPGA board [4]

The NetFPGA is a PCI-based experimental platform developed by Stanford University, which consists of a fully programmable Xilinx Field Programmable Gate Array (FPGA) based core with four Gigabit Ethernet interfaces, Static Random Access Memory (SRAM) and Double-Date Rate Dynamic Random Access Memory (DDR2 DRAM) . This platform is designed to provide a flexible academic and experimental tool which enables networking researchers and students to easily implement their research ideas in a hardware platform. All source codes of NetFPGA are open. NetFPGA users can use the reference design provided by Stanford NetFPGA group to build a complete network interface card (NIC) , switch or router, or conduct their own implementations by modifying any codes as needed.

The hardware components of the NetFPGA platform are shown in Fig. 3.1 [4]. The core of the board is a Xilinx Virtex-II Pro 50 chip that is fully programmable by the end users. The FPGA chip is connected to a Broadcom PHY chip, which controls four Gigabit Ethernet networking interfaces supporting standard Cat5E

or Cat6 copper network cables. The NetFPGA board has another Xilinx chip to be the PCI controller and communicate with the host through standard PCI bus. All operations between NEtFPGA and the host such as user design downloading, software register updating and traffic data collection are conducted through this PCI interface. To provide buffering facility when the NetFPGA router is too busy to process the traffic, the board is equipped with two Cypress SRAM chips that provide 4.5M bytes capacity. A Micron DDR2 DRAM chip of size 64 MBytes is also embedded in the board but not used in version 2 of NetFPGA reference design.

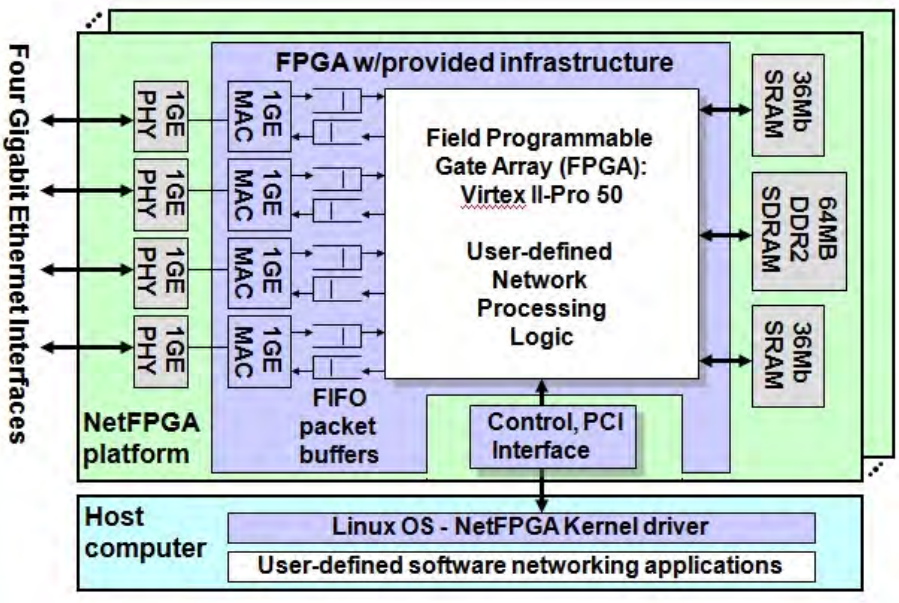


Figure 3.2: The logical architecture of the NetFPGA [4]

A block diagram of the NetFPGA is shown in Fig. 3.2 [4]. We can see that the NetFPGA system is composed of several logic blocks: user-defined networking logic is loaded into FPGA chip to realize all routing functions as a router and also to communicate with external memories(SRAM and DRAM) and four Gigabit Ethernet interfaces; user-defined software applications running on the host control the FPGA chip and collect statistical data; between the hardware and the software lies the NetFPGA kernel driver, which receives application requests and drives the hardware. The NetFPGA software package is running in Linux system. The operating system can be Centos 5.0 or Fedora 13. The software modules of the NetFPGA platform are written in four programming languages: Verilog, C, Perl and Java. User-defined networking logic in Fig. 3.2 is written in Verilog HDL, which is one of the most popular Hardware Description Language (HDL) . The user-defined

software network applications are programmed in Java, which is a class-based and object-oriented programming language and widely used in web applications. C is chosen to write kernel drivers. All debugging applications and test codes are written in Perl.

The experimental work in this chapter uses the NetFPGA revision 2 board along with the standard reference router gateware (*reference_router.bit*) taken from [4]. We use the reference router bit-file as supplied, since that is the base distribution most researchers will build upon. Needless to say, modifying the architecture (for example to store packets in DRAM rather than SRAM) would doubtless change the energy performance. We have not optimised the code for energy-efficiency, and leave that for future work.

When NetFPGA is built as a router, like commercial routers it handles all the packet-processing tasks in *hardware* (i.e., longest-prefix matching, ARP lookups, etc.). The NetFPGA has been used extensively in numerous research studies, such as for router buffer sizing, clean-slate network architecture and design, etc. Additional information can be obtained from the NetFPGA website [4].

3.2.2 Life of a Packet Through the NetFPGA Router

The core of the NetFPGA hardware design is a modular structure called Reference Pipeline, which consists of rx/tx queues and a packet processing pipeline called User Data Path [4]. The reference pipeline has four Ethernet rx queues and four CPU rx queues, correspondingly four Ethernet tx queus and four CPU tx queues. User Data Path includes three modules: Input Arbiter, Output Port Lookup and Output Queues. The input arbiter module selects which rx queue to serve and pushes incoming packets to the next module. The output port lookup module receives packets from the input arbiter, executes both routing table and address resolution protocol (ARP) lookup and decides to which output port the packets should be put. The packets from the output port lookup module are processed in the output queue module, which will decides which output queue to put the packets in according to the output port information provided by the previous module. The difference among three reference projects provided by Stanford (Reference NIC, Reference Switch and Reference Router) lies in the coding and module in the reference pipeline and it is the basic structure that all other new projects should follow.

Fig. 3.3 shows the reference pipeline structure of the NetFPGA [4]. For a packet that is *received* via the Ethernet MAC receive queue (MAC RxQ), it is first pro-

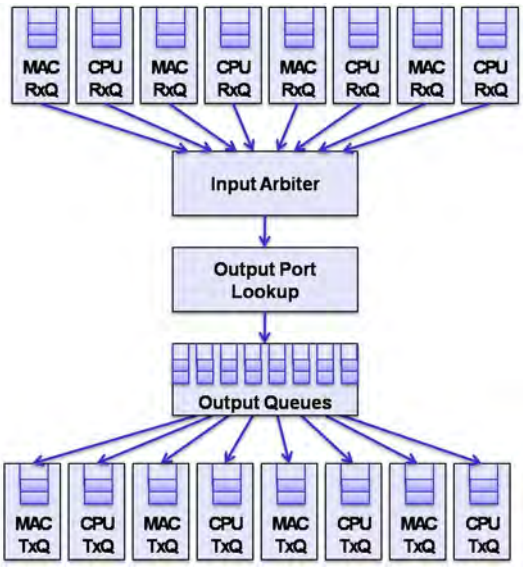


Figure 3.3: Life of a packet through the reference pipeline of the NetFPGA router [4]

cessed by the input arbiter module, stored in the Xilinx FPGA's on-chip memory while the output port lookup module inside the FPGA performs the following *packet processing* steps: (1) it does a longest prefix match on the routing table to decide the output port to which the packet must be switched to, (2) it obtains the next-hop MAC address by performing an ARP lookup, (3) it modifies the source and destination MAC addresses in the packet header, the TTL (time-to-live) is decremented, and the checksum is updated. Finally, the packet is buffered in the respective output queue in the output queue module and sent out of the NetFPGA on the wire via the Ethernet MAC transmission queue (MAC TxQ) when it is time for it to be dequeued.

3.2.3 A Simple Linear Model of Power Consumption

We use the following simple linear model of power consumption P of the NetFPGA router. When the card is powered on, but has no Ethernet ports connected (and hence no traffic), it consumes a constant power P_C. Each Ethernet port that is connected (i.e., link is up) but has no traffic flowing through it consumes an additional constant power P_E. Each packet that enters the router needs to be processed (parsing, route lookups, ARP lookup, etc.), and is assumed to require energy E_p that is independent of the packet size. Each byte of the incoming packet needs energy to be received (E_{rx}) and stored (E_{rs}) while it is processed; if the packet is

not dropped, it needs further energy to be stored in the output queue (E_{ts}) and thereafter transmitted (E_{tx}). This simple linear model can be formally expressed as below:

$$P = P_C + KP_E + N_I E_p + R_I(E_{rx} + E_{rs}) + R_O(E_{ts} + E_{tx}) \tag{3.1}$$

- P_C is a constant base-line power consumption of the NetFPGA card (without any Ethernet ports connected),
- $K \in [0, 4]$ is the number of Ethernet ports connected,
- P_E is the power consumed by each Ethernet port (without any traffic flowing),
- N_I is the input rate to the NetFPGA card in packets-per-second (pps),
- E_p is the energy required to process each packet (parsing, route lookup, etc.),
- R_I is the input rate to the NetFPGA card in bytes-per-second,
- R_O is the output rate from the NetFPGA card in bytes-per-second,
- E_{rx} is the energy required to receive a byte on the ingress Ethernet interface,
- E_{rs} is the energy required to process/store a byte on the ingress Ethernet interface,
- E_{ts} is the energy required to store/process a byte on the egress Ethernet interface, and
- E_{tx} is the energy required to transmit a byte on the egress Ethernet interface.

For a byte of a packet that does not get dropped, we find it convenient to define the term $E_b = E_{rx} + E_{rs} + E_{tx} + E_{ts}$ to denote the total per-byte energy; this term simplifies notation when the input rate R_I and output rate R_O are equal in Eq. (3.1).

3.3 Estimating Per-packet and Per-byte Energy Components

Under the assumption that the power consumption of the NetFPGA Gigabit platform running the standard reference router follows the model proposed above, the objective of this section is to determine each of the components in Eq. (3.1). In the

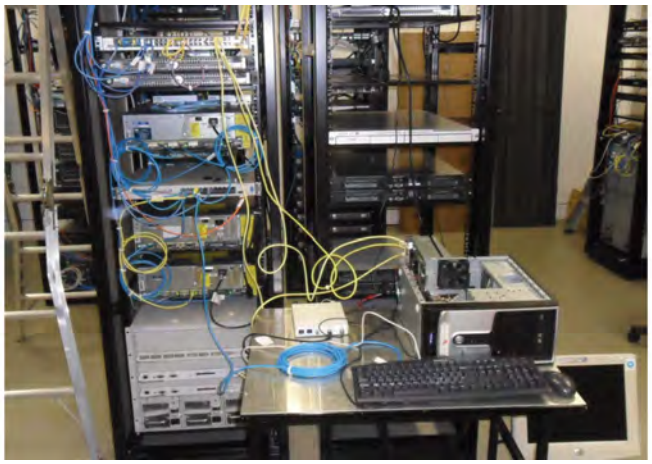


Figure 3.4: Experimental setup comprising of the NetFPGA router, riser card, oscilloscope, and traffic generator

following subsections we first outline our experimental setup and methodology, and then describe in detail each experiment that helps us estimate specific components of the energy model. Leveraging linear regression for each experimental data (shown as solid black lines in the following figures), we are able to estimate all unknown components which are not measured directly in the energy model.

3.3.1 Experimental Setup

The setup (pictured in Fig. 3.4) consists of three components: the NetFPGA router itself, a high-precision traffic generator, and a high-fidelity oscilloscope for power measurement, each described in turn next:

Router: We use a NetFPGA revision 2 board having four 1 Gbps Ethernet ports. The gateware on our NetFPGA is the supplied reference router (*reference_router.bit*, v1.0 Beta, build date 4 Jul 2009) taken from the NetFPGA website; note that we do not make any modifications to the gateware (for energy optimisation or otherwise), since researchers are likely to build upon the given base distribution. The NetFPGA card is housed in a desktop PC with a 3 GHz AMD Athlon processor with 2 GB memory, running CentOS version 5.2, along with Scone (the NetFPGA control software).

Traffic Generator: We use the IXIA [36], which is a high-precision commercial-grade hardware traffic generator with sophisticated capabilities for configuring traffic profiles, synchronising traffic on multiple ports, and accumulating statistics based on pattern filters. We will see further in this section how these capabilities are used to isolate the various energy components. We connected three Gigabit ports of the

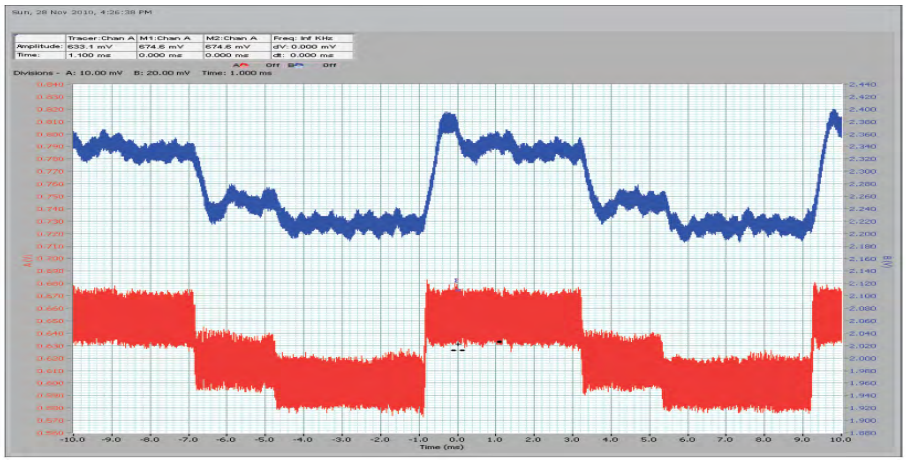


Figure 3.5: Oscilloscope output showing waveform of current draw on 3.3V (top) and 5V (bottom) supply

IXIA to three ports of the NetFPGA (the fourth port could not be connected due to a physical barrier presented by the power measurement apparatus, as described next). In our experiments Ethernet port 1 and 2 on the NetFPGA were typically the ingress ports, while port 3 was the egress feeding back to the IXIA.

Power Measurement: Early in our experimentation we noticed that the power consumed by the host PC (including the NetFPGA card) had a wide range of fluctuation (75 ± 5 Watts) even when there was no network traffic. These fluctuations most likely arise from mechanical components in the PC such as fans, disks, etc., and can mask the fluctuations we want to measure – namely changes in power drawn by the NetFPGA card as traffic patterns change. In order to accurately isolate the power consumed by the NetFPGA card alone, we therefore mounted the NetFPGA card on an Ultraview PCI Smart Extender PCIEXT-64U riser card [37], which is in turn inserted into the PCI slot of the host PC. The riser card has break-out pins for measuring current draw on the several voltage supply pins (3.3V, 5V and 12V) to the NetFPGA card. The current measurement pins (for the 3.3V and 5V supplies only, since the NetFPGA does not use the 12V supply) were connected via leads to a two-channel USB digital oscilloscope, specifically the Cleverscope CS328A [38], which samples the current draw every 20 nanoseconds and records them to a file (Fig. 3.5 shows a snapshot of the visual output). Each run of each experiment described below collected 100 million samples of the instantaneous current draw (on each supply voltage) to determine the average power consumption, and we typically performed ten runs for each experiment.

3.3.2 Baseline Power P_C

We measured several times over the day the baseline power consumed by the NetFPGA card, namely when none of the Ethernet ports are connected. On average the baseline power was $P_C = 6.936$W. However, we noticed that the standard deviation was quite large at about 67mW. In particular, we noted that as the NetFPGA operated continuously (routing traffic) for a few hours, it grew hotter and the baseline power consumption drifted. We believe this is because power consumption is affected by factors such as leakage current that vary with temperature. Since temperature can vary between our experimental runs, we minimise its impact on our estimations by concentrating on power differentials (i.e., slope of the power curve) within an experiment rather than using the absolute numbers themselves, as described in more detail further on.

3.3.3 Ethernet Per-Port Power P_E

Starting from the baseline above (with no ports connected) we successively connected each Ethernet port (to get link up but with no traffic flowing) and measured the power consumption of the NetFPGA card. The increase in power was almost perfectly linear, with each Ethernet port adding $P_E = 1.102$W. This must come from the PHY and MAC components that activate the link, perform the carrier sensing, check for an incoming preamble, etc. In all our subsequent experiments three of the NetFPGA Ethernet ports are connected, and the baseline power consumption for this 3-port configuration was found to be $P_C + 3P_E \approx 10.242$W.

3.3.4 Per-Packet Processing Energy E_p

Our next experiment is designed to estimate E_p, the incremental energy cost of processing each packet, which includes the energy required to parse the packet (to extract Ethernet/IP headers) and to perform route lookups. This energy cost is incurred on a per-packet basis, and should not depend on packet size. Our approach is as follows: We fix the packet size (to say L Bytes), and send a constant-rate stream of packets from the IXIA to the NetFPGA on port 1, which gets routed out of port 3 back to the IXIA at a constant rate. There are no packet drops or losses. The experiment is repeated for data rates of 100Mbps, 200Mbps, and so on till the maximum line-rate (close to but not exactly equal to 1Gbps due to inter-packet gaps). The whole process is repeated for several choices of fixed packet size to get

more confidence in our estimates.

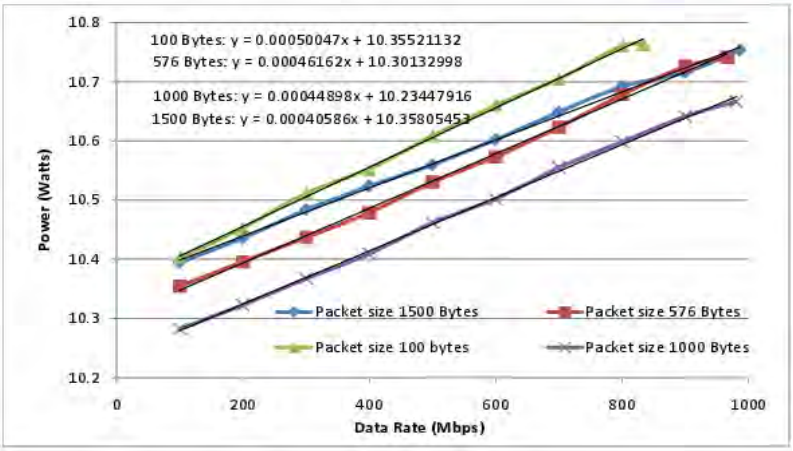


Figure 3.6: Power consumption versus data rate for fixed packet size

Fig. 3.6 shows how the power consumption P of the NetFPGA card changes as a function of data rate R_I for fixed packet size of $L = \{100, 576, 1000, 1500\}$ Bytes. Our first observation from the figure is that the power consumption varies by nearly 400mW as the traffic rate is varied from 10% to 100% of line-rate, which represents an increase of only about 4% on the base-line power consumption of 10W when the NetFPGA router is powered on (with 3 connected ports) but has no traffic. This is in-line with observations made by other researchers (e.g., [13]) that commercial routers consume most of their power just to be on, and the impact of traffic load on power consumption is relatively small. Nevertheless, there are many ongoing research efforts to make the power consumption of chips, devices and systems proportional to their workload, and our study aids their research effort by helping understand the fine-grained dependence of energy on traffic. Our second observation from the figure is that each curve (corresponding to one value of packet size L) is well approximated by a linear fit, also shown in the figure. This lends credence to the linear energy model in Eq. (3.1).

We now see how this experiment helps us estimate the per-packet energy component. We begin by taking the partial derivative of Eq. (3.1) with respect to the input rate R_I, noting that the input packet rate $N_I = R_I/L$ where L is the fixed packet size in Bytes, and that $R_O = R_I$:

$$\partial P/\partial R_I = E_p/L + E_b \tag{3.2}$$

where we have used $E_b = (E_{rx} + E_{rs}) + (E_{tx} + E_{ts})$ to denote the overall per-byte

energy. For a given value of packet size L, the left side of the above equation, namely $\partial P/\partial R_I$, can be deduced from the slope of the corresponding curve in the figure. Specifically, if the curve for packet size L is represented by a straight line of the form $m_L x + c_L$, then $\partial P/\partial R_I = 8m_L$ (the multiplier of 8 accounts for the fact that R_I is in bytes-per-second while the figure shows data rate in bits-per-second). At this point we note that the figure shows that the slope m_L decreases with packet size L, confirming that for the same increase in data rate, smaller packets incur a larger incremental energy overhead. We also note that we choose not to use the intercepts c_L since the baseline power consumption was not found to be very stable (as explained earlier in Section 3.3.2) over the several hours we needed to do the various sets of experiments. This explains the fact that actual power consumption of 1500 Byte packets is lower than the power consumption of 576 and 1000 Byte packets, even though the intercept value c_L for the curve of packet size 1500 Bytes is higher than those for 576 and 1000 Bytes in Fig. 3.6.

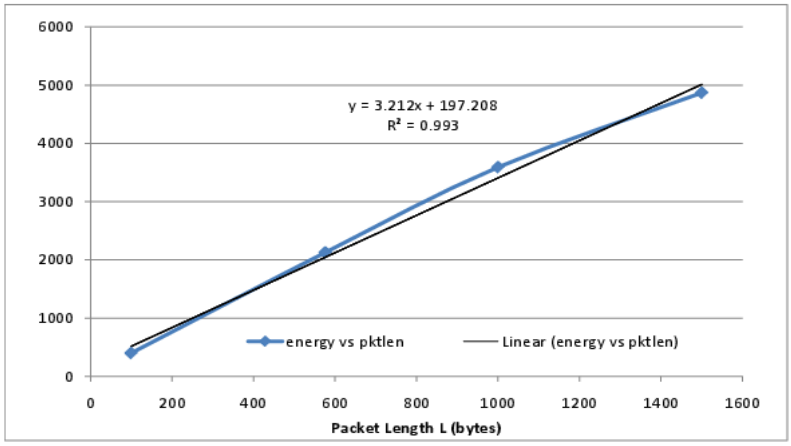


Figure 3.7: Fitting data to estimates of energy components

Eq. (3.2) can then be rewritten as the family of equations:

$$8 * L_j * m_{L_j} = E_p + L_j * E_b, \qquad j = 1, 2, 3, 4 \tag{3.3}$$

where $L_j = \{100, 576, 1000, 1500\}$ denotes the four packet sizes in our experiments, with corresponding slopes $m_{L_j} = \{0.00050047, 0.00046162, 0.00044898, 0.00040586\}$ as shown in the equations for the best linear-fit in Fig. 3.6. We solve for the energy components E_p and E_b by finding the best-fit to the equations. Specifically, we treat the above equations to be of the form $y = a + bx$, where y corresponds to the left side value in each of the equations and x is the packet length value. We plot

the (x, y) values in Fig. 3.7, and see that the data points fit almost perfectly (with $R^2 = 0.993$) to a straight line. The slope of this straight line corresponds to the per-byte energy $E_b \approx 3.212$nJ, while the intercept gives us the per-packet processing energy $E_p \approx 197.208$nJ.

3.3.5 Per-Byte Total Energy E_b

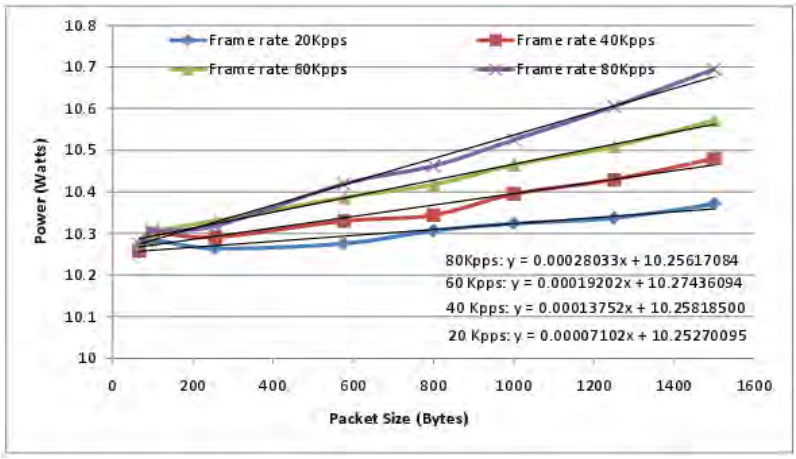


Figure 3.8: Power consumption versus packet size for fixed packet rate

Though our previous experiment also gave us an estimate for the per-byte energy $E_b = (E_{rx} + E_{rs}) + (E_{tx} + E_{ts})$, in this experiment we directly deduce the per-byte energy as follows: our setup still has one traffic stream that enters the NetFPGA on one port (port 1) and egresses on another port (port 3), and there are no losses (so $R_I = R_O$). This time we fix the packet rate N_I, and vary the packet size (from 64 up to 1500 bytes) in order to vary the data rate. We perform our experiment for four packet rates: $N_I = 20, 40, 60, 80$ Kpps, and the corresponding power consumptions of the NetFPGA card as a function of the packet size are shown by the four curves in Fig. 3.8. Once again we note that each curve is roughly linear, and the slope and intercept of the best linear-fit are also depicted in the figure.

To deduce the per-byte energy, we take the partial derivative of Eq. (3.1) with respect to packet length L, noting that N_I is a constant and $R_I = R_O = LN_I$:

$$\partial P/\partial L = N_I * E_b \tag{3.4}$$

where $E_b = (E_{rx} + E_{rs}) + (E_{tx} + E_{ts})$ denotes the total per-byte energy. Estimating the slope of each curve in Fig. 3.8 therefore yields the left side of the equation above, which when divided by the packet rate N_I directly yields an estimate of the per-byte

energy E_b. For the four curves corresponding to frame rates of $20, 40, 60, 80$ Kpps, using the slopes of the best linear-fit from the figure, we obtain estimates of E_b as $3.551, 3.438, 3.200, 3.504$nJ, which are reasonably consistent, yielding an average estimate $E_b = 3.423$nJ. This is also fairly consistent with the estimate of 3.212nJ we had obtained for E_b in the previous subsection.

3.3.6 Per-Byte Receive-Side Energy $E_{rx} + E_{rs}$

In this experiment we try to independently deduce the per-byte energy $E_{rx} + E_{rs}$ at the receive (ingress) side, which includes the energy for receiving the bytes and storing them while the packet headers are being processed and a routing decision is being made. Our methodology for estimating the receiver-side energy is as follows: we oversubscribe the output link (port 3) of the NetFPGA by feeding in traffic on two input links (ports 1 and 2). In each experiment, the packet size is fixed, and the input rate is varied such that the output link stays oversubscribed (thus the output traffic rate is fixed). Consequently, a known fraction of packets received by the NetFPGA get dropped, and in the process incur energy cost at the receiver but not at the transmitter side. This lets us isolate the receiver-side energy.

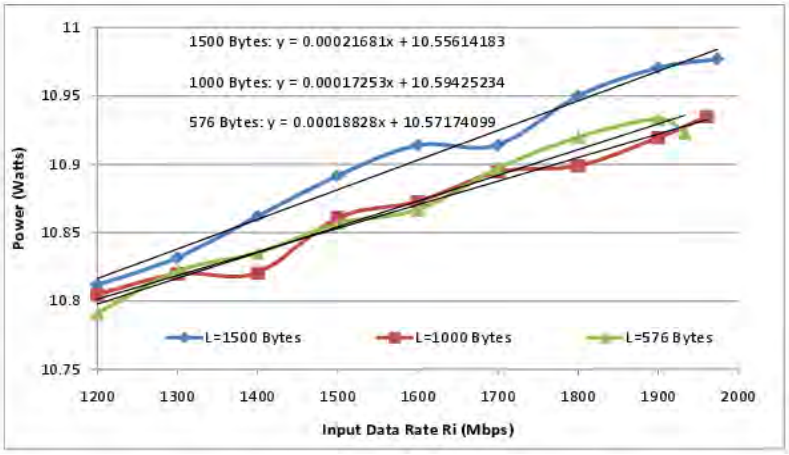


Figure 3.9: Power consumption versus input data rate for fixed output rate

More specifically, we fix packet size at each of three values $L = \{576, 100, 1500\}$ bytes. The total input rate (from two input ports) varies from 1.2 to 2 Gbps, while the output stays saturated at 1 Gbps (less the inter-packet gaps). Fig. 3.9 shows how the power of the NetFPGA card varies with input data rate R_I for three settings of packet size L. To estimate the receiver-side per-byte energy $E_{rx} + E_{rs}$, we take the partial derivative of Eq. (3.1) with respect to R_I, bearing in mind that the

output data rate R_O and the packet size L are constants, while the input packet rate $N_I = R_I/L$:

$$\partial P/\partial R_I = E_p/L + (E_{rx} + E_{rs}) \tag{3.5}$$

For each of the three packet sizes used in this experiment, the value of the left side of the above equation is deduced from the slope of the best linear-fit of the corresponding curve in the figure. Using the value of the per-packet processing energy E_p from Section 3.3.4, we deduce that $E_{rx} + E_{rs} \approx 1.317$nJ.

We are not experimentally able to isolate the individual components E_{rx} and E_{rs}, as we do not see a way by which the NetFPGA card can be made to receive the byte without storing it. However, other studies such as [39] have reported that a commercial Ethernet PHY operating at 1 Gbps consumes around 62mW for transmit and receive, which translates to per-byte receive energy of $E_{rx} = 0.496$nJ. Using this value, we can then estimate that the energy of storing a byte in the ingress side of the NetFPGA is $E_{rs} \approx 1.317 - 0.496 = 0.821$nJ.

3.3.7 Per-Byte Transmit-Side Energy $E_{ts} + E_{tx}$

In this experiment we try to directly deduce the transmit-side per-byte energy, associated with storing the byte in the output queues E_{ts} and transmitting the byte on the output link E_{tx}. To do so, we keep the input rate to the NetFPGA fixed, while trying to vary the output rate. This actually turns out to be quite tricky, and is achieved as follows: we send synchronised bursts of packets from the IXIA to the two input ports of the NetFPGA (ports 1 and 2). Each burst consists of 500 back-to-back packets of size 1000 bytes sent at line-rate (this takes about 4 msec), followed by an off-period (no packets sent) for 6 msec. The bursts are perfectly synchronised on the two input ports, so that when both ports start sending packets (for an on-period of about 4 msec), the output link is oversubscribed and the queue builds up, and when both sources stop (for an off-period of 6 msec), the queue drains out, and the entire process repeats. By adjusting the buffer size on the output link, we can vary the number of packets dropped in each on-off cycle, thereby controlling the output rate, which is measured by the IXIA. This setup therefore lets us control the output rate for fixed input rate, which helps us estimate the transmit-side per-byte energy as follows: taking partial derivates of Eq. (3.1) with respect to the output rate R_O we get:

$$\partial P/\partial R_O = (E_{ts} + E_{tx}) \tag{3.6}$$

since the input packet rate N_I and data rate R_I are fixed.

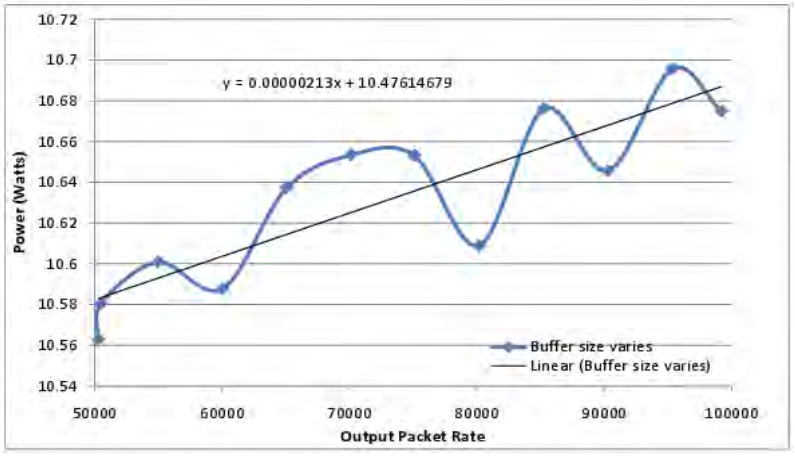


Figure 3.10: Power consumption versus output packet rate for fixed input rate

Fig. 3.10 shows how the power consumption of the NetFPGA card varies as a function of output packet rate (packet size was fixed at 1000 bytes and input packet rate was fixed at $49,603.65$ pps), achieved by varying buffer size from 3 KiloBytes to 500 KiloBytes. The curve has a small range (less than 140mW), and is quite noisy, with $R^2 \approx 0.75$ for a linear fit. Nevertheless, the slope of the curve (when converted to units of bytes rather than packets) directly yields an estimate of the transmit-side per-byte energy $E_{ts} + E_{tx} = 2.130$nJ. This is reasonably close to the estimate $E_b - (E_{rx} + E_{rs}) = 3.423 - 1.317 = 2.106$nJ from our earlier results, and thus provides independent confirmation of the accuracy of our estimates.

Once again, separating E_{ts} from E_{tx} is not possible experimentally without modifying the gateware, but we again depend on the estimate of $E_{tx} \approx 0.496$nJ from earlier work [39]. This lets us estimate that $E_{ts} \approx 1.61$nJ. We note that the storage energy E_{ts} on the transmit side is almost twice the storage energy E_{rs} on the receive side, which we believe is because the ingress queues are on-chip on the NetFPGA, whereas the transmit queues are off-chip in the external SRAM.

3.4 Summary

To support the increasing research activity in improving the energy-efficiency of Internet routing equipment, in this chapter, we experimentally derived fine-grained quantitative estimates of the per-packet and per-byte energy costs in the NetFPGA 4-port Gigabit routing platform. Table 3.1 summarises our results on the decomposition of the energy costs in the NetFPGA Gigabit router. We qualify our results

with the following notes:

- Fully loading the NetFPGA router with traffic causes its power consumption to increase by a mere 5% over being idle. However, ongoing research in "work proportional" devices and components with very low idling energy will likely amplify the relative impact of traffic load on router power consumption.
- Our experiments had only a handful of routing table entries configured, and it is likely that having hundreds of thousands of routing entries could increase the per-packet processing energy. However, we are limited in the NetFPGA hardware's current ability to store at most 32 routing table entries.
- The output queue in the NetFPGA standard reference router is currently implemented in SRAM. Modifying the gateware to use the DRAM for output queueing will likely alter the per-byte power consumption for storage at the egress.

Energy component and description	Our estimate
Power consumed by unconnected NetFPGA card (P_C)	6.936 W
Power consumed per connected Ethernet port (P_E)	1.102 W
Per-Packet processing energy (E_p)	197.2 nJ
Per-Byte energy (E_b)	3.4 nJ
Per-Byte Ingress storage energy (E_{rs})	0.8nJ
Per-Byte Egress storage energy (E_{ts})	1.6 nJ
Per-Byte transmit/receive energy (E_{tx}, E_{rx})	0.5 nJ

Table 3.1: Summary of NetFPGA power profile

Not withstanding the above caveats, we believe our work provides valuable information to the research community on the incremental energy costs of switching traffic through the NetFPGA router. We believe our work can be used in at least two ways: to compute network-wide energy footprints of TCP sessions, file transfers, user activity patterns, etc., and to evaluate the performance of new architectures and protocols (such as processor speed scaling, energy-aware transport protocols, etc.) in terms of their energy improvements. Our study informs the research community on the relative energy costs of the various components in the router, and provides a benchmark against which energy performance of new proposals can be compared.

We have profiled power consumption of the NetFPGA Gigabit router, in the next chapter, we will continue our energy efficiency analysis on another important type

of Internet equipment - switches. Our analysis will focus on switches that support newly-issued IEEE power saving standard - Energy Efficient Ethernet (802.3az) and include a power model to be referenced by further energy efficiency research.

Chapter 4

Profiling Power Consumption of Energy Efficient Ethernet Switches

Ethernet is one of the first computer networking technologies for which a standard has been developed to improve its energy efficiency. The Energy Efficient Ethernet (IEEE 802.3az) standard was approved in 2010 and is expected to enable savings of several Terawatt hours (TWh) per year. As switches that implement the standard become available in the market and are deployed in commercial networks, it is important to understand how their energy consumption depends on the number of active ports and their traffic. In this chapter the energy consumption of Energy Efficient Ethernet switches is analysed in several experiments and based on the results a model for the energy consumption of Energy Efficient Ethernet switches is proposed.

4.1 Introduction

Energy efficiency is becoming an important issue in computing and networking. This is due to both environmental concerns and economic costs associated with energy consumption. The overall consumption of computing equipment is predicted to grow substantially in this decade unless energy efficiency mechanisms are incorporated in computing systems [40]. The issue affects both high performance computing facilities such as datacenters and end user equipment. For example, the energy consumed by a data center is a key operational cost [41] and the aggregated consumption of end user devices is significant due to the large number of devices [42].It has been observed that the energy consumption is in many cases almost constant and independent of

the system load [43]. This results in poor energy efficiency for lightly loaded systems. Therefore energy efficiency of computing and networking systems can be significantly improved by making the energy consumption more proportional to system load.

The energy consumption of networking equipment is relevant and only for the Internet was estimated to be over 6 Twh in 2000 [8]. More recent studies suggest a larger consumption when end user equipment and access networks are considered [44]. Although their individual power consumption is small, most of the energy is consumed by devices in the access network and end user premises. This is explained by the large number of such devices compared to core or transport network devices.

One of the first networking technologies for which energy efficient mechanisms have been defined in a standard is Ethernet. The Energy Efficient Ethernet IEEE 802.3az standard approved in September 2010, defines a low power mode that improves the energy efficiency of Ethernet physical layer devices [22]. With an installed base of over one billion devices, the expected energy savings have been estimated in over 4 Twh [3]. Products that implement the standard are becoming common in the market and wide adoption is expected to occur in a few years. The use of Energy Efficient Ethernet (EEE) will change the energy consumption profile of switches making it more proportional to the traffic load [23]. However current products that implement EEE do not provide much information on the energy consumption profile [45; 46; 47; 48]. Typically only minimum and maximum power consumption levels are given. In this chapter an in-depth analysis of the power consumption profile of small Ethernet switches is presented. Based on the results a model for the energy consumption of EEE enabled Ethernet switches is proposed. The model can be used by researchers that want to explore higher layer energy saving mechanisms like energy efficient routing [13] or selective link deactivation [11]. This would allow them to make realistic estimates of the savings that would be achieved by those techniques in Ethernet networks.

The rest of the chapter is organized as follows, a review of the current power consumption of small Ethernet switches is provided in section 4.2. In section 4.3 the power consumption of two small Energy Efficient Ethernet switches is evaluated in a set of experiments. Based on the results a simple model for the energy consumption of Energy Efficient Ethernet switches is presented in section 4.4. Section 4.5 discusses the implications of the results and finally the summary is presented in section 4.6.

4.2 Power Consumption of Ethernet Switches

There is a wide range of Ethernet switches. From a four or five port switch used in homes and small offices to modular switches that support hundreds of ports and different transmission media [49]. Power consumption increases with the number of ports and their speed and therefore large switches consume much more energy than small ones. However since there are many more small switches than large ones, the aggregated energy consumption of the small switches to which users are connected is significant. For example the power consumption of small Ethernet switches in the US has been recently estimated in 7.9 TWh/year [50]. Small switches typically have 5,8,16 or 24 ports. This reduced number of ports enables highly integrated implementations in which only one [51] or a few integrated circuits are used [52]. The switch is composed of one physical layer device (PHY) per port, a switching fabric commonly implemented with a shared memory, control logic and a CPU [53].

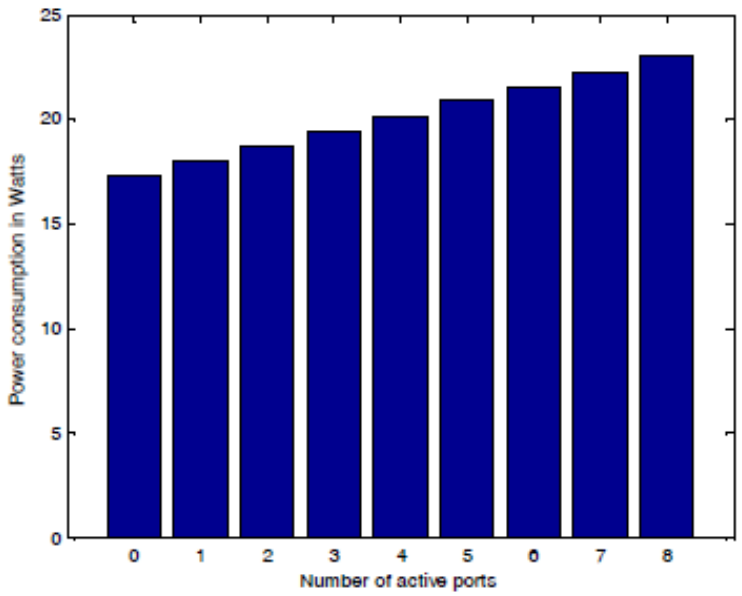


Figure 4.1: Energy consumption of a Cisco Catalyst 3560-CG switch as a function of the number of active ports

The power consumption of switches and routers has been characterized in different studies [54; 55; 56]. The results show that once the router or switch is powered on and its ports are activated the power consumption is close to its maximum value. For example in [56] that value is around 90% of the peak power consumption for a core router and also for an Ethernet router. That means that only 10% of the peak power consumption is dependent on the traffic load. This is far from the

proportional relation and results in poor energy efficiency as networks tend to be lightly loaded [57]. In [54] the energy consumption of commercial Ethernet switches is reported. In our previous work [55] an Ethernet based NetFPGA router for academic purposes was evaluated. In both cases the power consumption once all the ports are active is close to the peak power consumption. The power consumption is also analysed when the number of active ports is varied. The results show that the power consumption increases as the number of active ports grows. To corroborate the results, the power consumption of an eight port Cisco Catalyst 3560-CG switch has been measured. The results are consistent with previous studies showing that only a very small percentage of the power consumption depends on the traffic load once all ports are active. Fig. 4.1 illustrates the results in a single plot as to the measurement accuracy they were the same for no traffic and full traffic. This power consumption profile has motivated research efforts that try to reduce the number of active links when there is no traffic [11] or try to allocate traffic such that the number of links that are activated is minimised [13].

4.3 Power Consumption of Small Energy Efficient Ethernet Switches

To study how the use of Energy Efficient Ethernet affects the energy consumption profile of switches two small Ethernet switches have been characterized in a number of experiments. The models selected are D-Link DGS-1100-16 which is a 16 port Gigabit switch [45] and Level One GEU-0820 which is an 8 port Gigabit switch [47]. In this way two common configurations in terms of number of ports are tested. As mentioned before, the interest of analysing small switches lies in the fact that their aggregated energy consumption is very significant [50].

In the first experiment, the power consumption is measured when the Energy Efficient Ethernet functionality is disabled and the number of active ports is varied under no traffic and full load conditions. As in the experiment with the Cisco Catalyst 3560-CG switch, the power consumption with no traffic and at full load was almost the same. The results for no traffic are shown in Fig. 4.2 and can be directly compared with those of the Cisco Catalyst 3560-CG switch in Fig. 4.1. It can be observed that the profiles are similar but the absolute power consumption of both the DGS-1100-16 and the GEU-0820 switches is substantially lower than that of the Cisco switch. This is a result of technology scaling and shows how the use of more

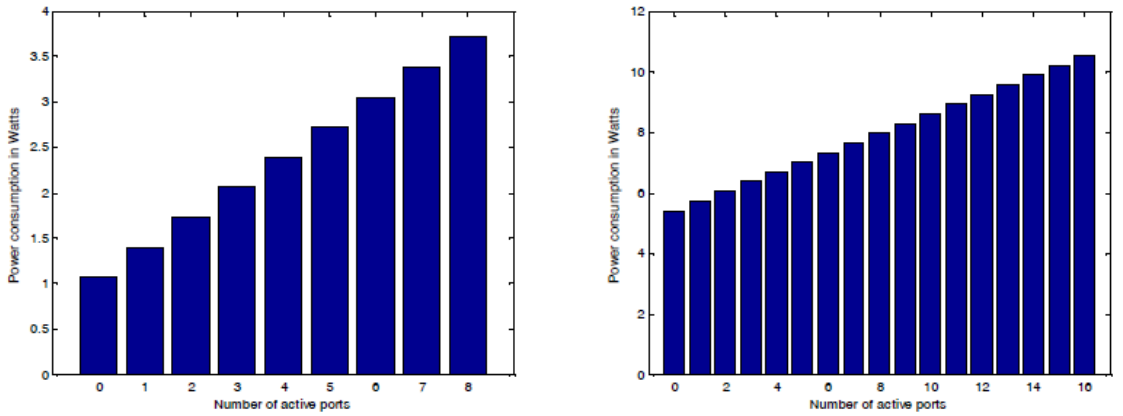


Figure 4.2: Energy consumption of Level One GEU-0820 (left) and D-Link DGS-1100-16 (right) switches as a function of the number of active ports with no traffic when EEE is disabled

advanced electronic technology can reduce the power consumption significantly. In particular the increment per port is lowered from 0.71 Watts to approximately 0.32 Watts. This power consumption reduction for 1000BASE-T PHYs has also been recently reported in NICs [24] when compared to previous studies [58]. It is also interesting to note that the power consumption of the GEU-0820 switch when no port is active is much lower than that of the DGS-1100-16 switch. This may be explained in part because the DGS-1100-16 switch has two times the number of ports in the GEU-0820 switch. However, the difference is so large that it suggests that the power consumption with no port active depends heavily on the switch implementation.

In the second experiment Energy Efficient Ethernet functionality is enabled and the same measurements done in the first experiment are repeated. For the case of full traffic load, the results are similar to those of the first experiment when EEE was disabled. The results for no traffic are shown in Fig. 4.3. It can be observed that the profile changes significantly and becomes almost independent of the number of active links. This indicates that when EEE is disabled and there is no traffic passing through the links, much power can be saved by powering off unused ports. But the situation significantly changes when the network load is light, there is less promise for energy saving by adoption of EEE while reducing the number of active links delivers little benefit. However as discussed previously, large transition overheads have been observed in EEE [24]. This means that even for low loads, the energy consumption can be significant.

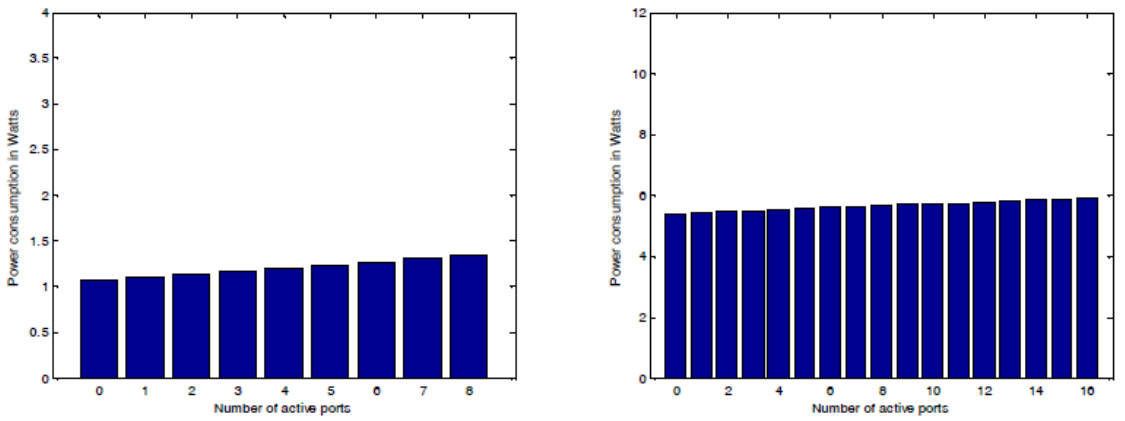


Figure 4.3: Energy consumption of Level One GEU-0820 (left) and D-Link DGS-1100-16 (right) switches as a function of the number of active ports with no traffic when EEE is enabled

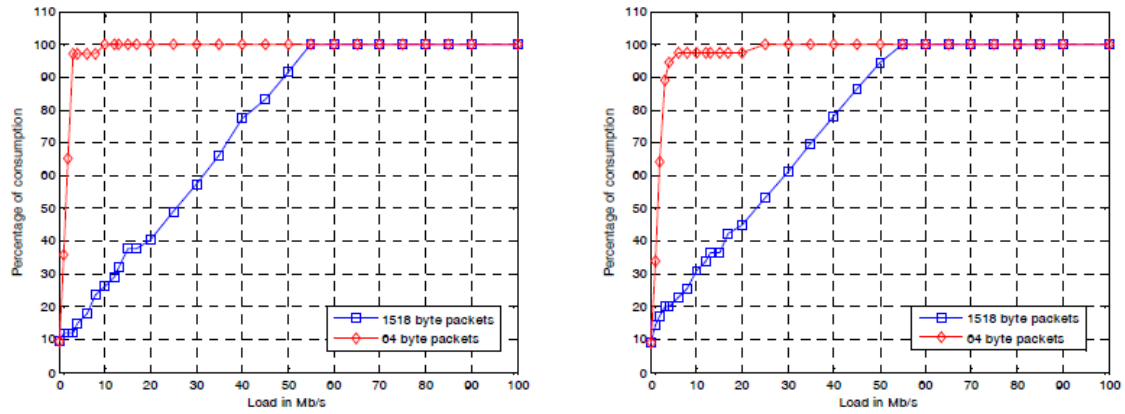


Figure 4.4: Energy consumption of a port in a Level One GEU-0820 (left) and D-Link DGS-1100-16 (right) switches as a function of the traffic load

In the third experiment, the load of one of the ports was varied from 1Mb/s per port to 1Gb/s using 1518 byte or 64 byte packets. Those packet sizes are the best and worst cases for transition overhead in EEE. The goal of the experiment is to characterize the per port power consumption versus its traffic load. The results are shown in Fig. 4.4 in terms of the percentage of power consumption for that port. It can be observed that the power consumption tends to saturate around 55 Mb/s for 1518 packets and well below 10 Mb/s for 64 byte packets. This is due to the transition overheads in EEE that are summarized for 1000BASE-T in Table 2.1. When the sending rate is limited to a few Mb/s packets are sent spaced on the

1Gb/s link thus causing frequent transitions in and out EEE low power mode that reduce the energy savings. These results are similar to the behavior reported for an EEE NIC in [24]. Since the load of small switches is low, most of the time the switch would be in the left hand side of the plot where power consumption increases linearly with the load. Furthermore, in the third experiment we also collected power consumption for intermediate packet size (i.e. 576 byte packets) and the result curve lies between the two plotted curves in Fig. 4.4. This indicates that the power performance corresponding to other packet sizes will presumably lie between two curves of the best and worst packet size cases.

4.4 An Energy Consumption Model for Small Energy Efficient Ethernet Switches

Based on the previous experiments, a model for the power consumption of small EEE switches can be proposed. Since the load for small switches is typically low, the model will focus on providing accurate energy consumption estimates for per port traffic below 50 Mb/s and a reasonable approximation for medium loads. The model is also conservative with regard to the traffic patterns and can be used to provide a lower bound on the expected energy savings.

One key observation for the model is that in many cases, traffic on the LAN is limited by a lower speed link, for example in the access to the Internet or in the interconnection with another LAN. When that is the case packets will be spaced when they are sent over that link so that for example at 50 Mb/s, 1518 byte packets are spaced by more than 200µs due to the larger transmission times at low speeds. However when they reach the LAN their transmission time at for example 1Gb/s is only 12µs which means that they are sent isolated and each packet will cause an EEE mode transition with its associated overhead.

This suggests that for end user or Small Office/Home Office (SOHO) switches and PCs that send or receive traffic from the Internet it is reasonable to assume that each frame requires an EEE transition. This behavior is captured by the model.

The assumption of a lower link limiting the rate of the port is the same as that used in the third experiment. In that experiment it was observed that for low loads the power consumption increased linearly with the load. This is consistent which the fact that each packet requires an EEE transition and therefore the increase in energy consumption is directly proportional to the increase of bandwidth (and consequently

to the number of packets). Therefore the power consumption in the proposed model is composed of a base power that is independent of the traffic load and a dynamic power that depends linearly on the traffic load.

Therefore proposed model is described by the following equation:

$$P = P_{base} + P_{dynamic} \sum_{i=1}^{N_{port}} \min(1, D \cdot \rho_i) \tag{4.1}$$

The parameters of the model are P_{base} which is the power consumption with no traffic load, $P_{dynamic}$ which is the difference between the power consumption at full load and P_{base} divided by the number of ports and D which is the increment in power consumption per increment in traffic load per port (ρ_i). D can be estimated from the measurements in the third experiment. For the switches evaluated, the value of D was close to 18 for 1518 byte packets so that power consumption saturated at a load of 55 Mb/s. For 64 byte packets, the value of D was much larger close to a value of 400. The value of D can also be estimated from the values of Table 2.1. When packets are spaced, each packet requires $T_w+T_s+T_f$ seconds to transmit but only T_f contribute to the link load. Therefore the value of D would be $(T_w+T_s+T_f)/T_f$ which is precisely the inverse of the Single Frame Efficiency (SF_e) defined in Section 2.3.1.

For a 1518 byte packet this gives a value of 17.6 and for a 64 byte packet a value of 390.2 both in line with the values estimated from the measurements.

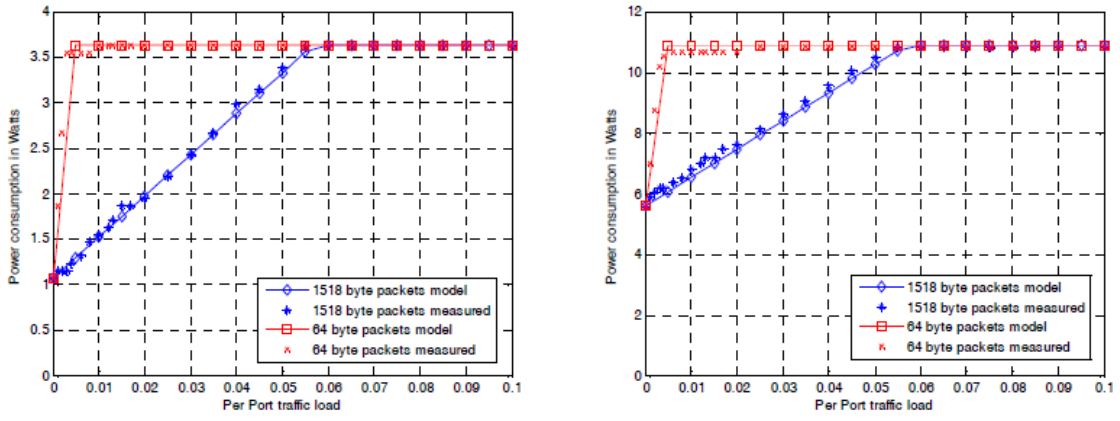


Figure 4.5: Energy consumption model for a Level One GEU-0820 (left) and D-Link DGS-1100-16 (right) switches as a function of the traffic load

The use of the model for the DGS-1100-16 and the GEU-0820 switches is illustrated in Fig. 4.5 in which it is assumed that the link load is the same for all ports.

The results of the model are also compared with the actual power measurements using different loads. It can be observed that they are in good agreement as in the experiment the load was controlled by limiting the transmission rate.

Obviously, the value of D depends on the traffic parameters such as the frame inter-arrival times and the frame size distribution. However as discussed before, for low loads and traffic sourced or destined to the Internet most frames will be transmitted isolated. When that occurs, D depends only on frame size and it can be estimated if the distribution of frame sizes is known. Let us assume that the probability of a frame having a size of L bits is p(L), then D can be calculated as follows:

$$D = \sum_{L=L_{min}}^{L_{max}} \left(P(L) \cdot \frac{1}{SF_e(L)} \right) = \sum_{L=L_{min}}^{L_{max}} \left(P(L) \cdot \frac{T_s + T_w + T_f(L)}{T_f(L)} \right) = \sum_{L=L_{min}}^{L_{max}} \left(P(L) \cdot \frac{T_s + T_w + \frac{L}{R}}{\frac{L}{R}} \right) \tag{4.2}$$

where R is the link speed. For 1000BASE-T and 10GBASE-T links $T_w + T_s \gg T_f$ and therefore D can be approximated as follows:

$$D \cong \sum_{L=L_{min}}^{L_{max}} \left(P(L) \cdot \frac{T_s + T_w}{\frac{L}{R}} \right) = R \cdot (T_s + T_w) \cdot \sum_{L=L_{min}}^{L_{max}} \left(\frac{P(L)}{L} \right) \tag{4.3}$$

where the influence of short packets on D can be clearly observed.

For end user systems, in many cases, most of the frames would also be 1518 byte frames or Acknowledgements that are correlated with data frames (an ACK reception triggers a frame transmission and the other way around). Therefore the isolated transmission of large frames can be a good approximation and in that case D would take a value of close to 18 for 1000BASE-T.

It is also worth mentioning that the proposed model is conservative in estimating the energy savings that is if the traffic patterns differ from the assumptions it would be only to increase the energy savings. Therefore the model can be also used to provide a lower bound on the expected energy savings when the assumptions on which is based are not fully met.

The model can be easily derived for other switches by measuring the maximum and minimum power consumption. Then P_{base} is the minimum power consumption and $P_{dynamic}$ is the difference between the maximum and minimum power consump-

tion divided by the number of ports. This means that only two simple power measurements are required to use the model that then enables a fast estimation of energy consumption based on very simple traffic load and frame size measurements. This can be useful when considering the adoption of EEE in a LAN. Another interesting application of the model is to predict energy savings of techniques that are being proposed to improve energy efficiency, like energy efficient routing or dynamic link shutdown [11; 13].

4.5 Discussion

The Energy Efficient Ethernet standard addresses the energy consumption of the PHY devices but can also enable savings in other system components [59]. These possible additional savings would be achieved by putting those components in a low power mode when the ports are in LPI mode as in that case no packets can arrive until the PHYs are activated. It seems that all the additional savings are not achieved in the first generation of EEE switches as the power consumption when there is no traffic on any port although lower than in the legacy switch remains significant (close to 25% for Level One switch and around 50% for D-Link switch). One possible explanation is that vendors have focused on implementing EEE on this first generation and left the optimisation of the rest of the switch elements for future releases. If that is the case one would expect further improvements in the future that will make the energy consumption of switches more proportional to traffic load. In any case, the proposed model would still be valid using different parameters (mostly reducing P_{base}). On the contrary, if there are actual limitations that make unfeasible a reduction of the power consumption when there is no traffic, then techniques that put the entire switch on a sleep mode such as the one proposed in [50] will provide significant benefits. On the other hand, techniques that only deactivate some of its links will provide insignificant energy savings in EEE switches.

Another interesting observation from the experimental results is that for no traffic EEE achieves a reduction of the port related power consumption of close to 90%. This means that the PHY consumption in LPI mode has to be around 10% that of the active mode in line with previous assumptions [23]. It is also worth mentioning the large decrease in power consumption due to technology scaling when comparing the D-Link and the Level One switches with the Cisco switch. This observation should be taken into account by network administrators as in some cases the energy

cost reductions may justify the renewal of switches.

Finally, it is also interesting to note that the experiments confirm the potential of Energy Efficient Ethernet to significantly reduce power consumption for small switches under realistic user traffic conditions. The overheads caused by mode transitions are not an issue for todays user traffic and should only be a concern for switches that have larger traffic loads, such as those used in Datacenters. In summary, Energy Efficient Ethernet will make the power consumption of small switches more proportional to system load helping to the more general goal of Energy Proportional Computing proposed in [43].

4.6 Summary

In this Chapter the first evaluation of power consumption of EEE Ethernet switches has been reported. The experiments show how EEE can reduce the PHY power consumption by 90% when there is no traffic. Based on the results, a simple model for the power consumption of small Ethernet switches has been proposed. The model provides accurate estimations for low traffic loads. It can also be used as a lower bound on the expected energy savings in other cases. To use the model in a given switch only two simple power measurements are required. As EEE is adopted over the next years, we believe that the model will be useful to estimate power savings in a simple way. It can also be used for research into new power saving techniques for Energy Efficient Ethernet LANs.

We have presented our work on profiling power consumption of Internet equipment in the previous two chapters to provide benchmarks for further energy efficiency research. In the next chapter we will demonstrate our efforts on improving energy efficiency of Internet equipment by proposing a practical buffer adaptation algorithm, which presents the possibility that power savings can be achieved by only inducing minimum changes to existing router design.

Chapter 5

Adapting Router Buffers for Energy Efficiency

Reducing the power consumption of core Internet routers is important for both ISPs and router vendors. ISPs can reduce their Carbon footprint and operational costs, while router manufacturers can achieve higher switching capacity per rack space. Today's backbone routers operate with Gigabytes of packet buffers per line-card to handle worst-case congestion scenarios. Such buffers account for nearly 10% of the power consumed by a typical router line-card, but are actively used only during transient periods of congestion. In this chapter, we argue that most of the energy associated with off-chip packet buffers can be eliminated with negligible impact on traffic performance, and propose a simple and practical algorithm for activating buffers incrementally as needed and putting them to sleep when not in use.

5.1 Introduction

The ICT sector consumed 156 GigaWatts, or about 8% of the world's total electricity consumption, in 2007. Of this, 14% is attributed to network equipment [60]. In addition to the large carbon footprint, the power density of modern core routers is becoming a serious concern for ISPs – a single telecommunications rack today consumes tens of KiloWatts of power, and requires complex cooling systems to manage heat dissipation. The high power consumption and cooling costs account for a significant fraction of the ISP's operational expenses. Though routing equipment is becoming more power efficient, the increase in efficiency is outpaced by annual increase in throughput capacity [7], meaning that the problem is likely to worsen

with time.

The gravity of the problem has motivated major chip vendors, equipment manufacturers, service providers and academic researchers world-wide to collectively [33] find ways to manage and reduce the power consumption of telecommunications networks. The problem needs solutions at multiple levels, ranging from more efficient chips and components, to higher-level power management techniques that turn off (or underclock) components and sub-systems at certain times, or even redesign the Internet for power efficiency. While several such schemes proposed in the literature have the potential to achieve (individually or in conjunction) considerable power savings, they involve significant architectural and/or protocol changes in the network. The cost and risk associated with such drastic changes increase the barrier to adoption by network operators, thus stretching the time-horizon at which they become practical for wide-scale deployment. By contrast, in this chapter we propose a power saving scheme that is admittedly more modest in its energy savings (around 10%), but requires minimal changes to existing router design, carries little risk of impacting network performance, is almost entirely transparent to network operators, and is ready for incremental deployment today.

Our specific focus is on adapting the packet buffer memories in core routers for improved energy efficiency. Today's backbone routers operate with Gigabytes of packet buffers per line-card to handle worst-case congestion scenarios. We present evidence that such buffers account for nearly 10% of the power consumed by a typical router line-card. Further, we examine data collected over several years from nearly a hundred links in carrier and enterprise networks, and find that high link-load (indicative of congestion) is a relatively rare occurrence, implying that it is wasteful in energy to keep the entire packet buffer memory always-on. We therefore propose that router buffer size be adapted dynamically to track buffer usage, allowing much of the off-chip buffer memory to be put to sleep when not needed, thus saving energy. Putting memory to sleep creates the risk of packet loss that could have been avoided with always-on buffers. Our scheme can be tuned to reduce this risk at the expense of reduced energy savings. We validate our mechanism for dynamic buffer control by analysis, simulation and experimentation. Specifically, we apply it off-line to traffic traces from operational networks, simulate it on-line in ns2 with several thousand TCP flows, and implement it to operate in real-time on an experimental platform comprising an FPGA-based programmable router and hardware-based traffic generators. Our evaluations show that dynamic buffer control

typically saves most of the energy associated with off-chip buffering (around 10% of the total energy), with negligible impact on traffic performance. Our specific contributions are:

• We argue that the energy costs associated with always-on packet buffers in today's routers are nearly 10%. We then present empirical evidence from several carrier and enterprise networks that link loads (and by inference buffer occupancies) are low for a large proportion of the time, which presents an opportunity to adapt router buffer size dynamically to save energy.

• We propose a practical mechanism for dynamic adjustment of buffer size. Our aim is to track buffer usage while still having some capability to absorb sudden bursts of packet arrivals. This trade-off between energy savings and risk protection can be controlled explicitly in our algorithm.

• We quantify the impact of our scheme in terms of energy savings and loss/throughput. We apply our algorithm off-line to traffic traces derived from Internet link data. We then simulate it in ns2, and profile its impact on TCP performance. Finally, we implement dynamic buffer adaptation in the gateware of the NetFPGA-based Gigabit router platform, and demonstrate its feasibility in a test-bed with realistic traffic.

Our aim is to persuade router manufacturers to incorporate dynamic buffer adaptation in core routers, and for network operators to trial them, as a relatively simple and safe way of reducing router power consumption.

The rest of this chapter is organised as follows: §5.2 motivates our focus on packet buffers and the opportunities to reduce their power consumption. Relevant background research is summarised in §5.3, while in §5.4 we present our dynamic buffer adjustment algorithm. §5.5 evaluates the algorithm via analysis, simulation and experimentation. The chapter is concluded in §5.6.

5.2 The Case for Reducing Router Buffer Energy

5.2.1 Energy Cost of Packet Buffers

Core Internet routers today typically have memory capacity to buffer over a million packets during periods of congestion. Moreover, at 40 Gbps, a 40-byte packet can arrive every 8 ns. To achieve large storage capacity while meeting such stringent throughput and latency requirements, routers need to employ various (off-chip) memory components in hierarchical configurations [61]. A core router line-card today has a few Gigabytes of dynamic RAM (DRAM), providing the bulk of the

packet storage, as well as several Megabytes of static RAM (SRAM), acting as the packet cache. Focusing first on DRAM, power calculators from Micron [62], a popular vendor of DRAM components, show that a state-of-the-art DDR2 SDRAM chip of 1 Gigabit capacity consumes about a Watt of power under moderate-stress conditions. At lower workloads, i.e. when there are few read/write operations, the power consumed by DRAM is lower but still non-negligible. It should also be noted that router manufacturers typically use specialised low-latency DRAMs such as Fast Cycle RAM (FCRAM) and Reduced Latency DRAM (RLDRAM), which consume about 40% more power than mass-market DDR2 or DDR3 SDRAM.

The SRAM, which implements the packet cache, typically consumes more power than the bulk DRAM buffers: for e.g., a 4 Megabyte SRAM chip (with synchronous pipelined burst and with no bus latency NoBL) from Cypress [63] consumes around 4 Watts. More importantly, the power consumption of SRAM is largely invariant to the load (i.e. frequency of read/write operations). Based on these power specifications, earlier works [12; 64; 65] have estimated packet buffers to account for between 5 and 10% of the power consumed by a router. As an example, Cisco's CRS-1 platform has reported that of the 375 Watts consumed by a line-card, memory accounts for 72 Watts (19%) [66], of which roughly 10% is attributable to packet memory.

In addition to powering the packet buffer memory, power is also required to drive the memory controller circuitry that implements the logic to move packets between main memory, cache memory, and on-chip memory. As shown in [61], this is particularly challenging in high-speed routers which typically have long pipelines, and cache misses (e.g., when a head-of-line packet ready for transmission is not available in cache) introduce non-determinism that can cause the system to lose throughput in unpredictable ways. Other complicating factors can include multiple output queues (e.g., for class-of-service support), static (e.g., circular queues) versus dynamic (e.g., linked-lists) buffer allocation, and multiple memory channels/banks across which packets are spread. Memory controllers, which have the intelligence for managing and moving packets across these buffers, are typically integrated into custom ASICs on most routing platforms, making it very difficult to estimate their energy footprint accurately. A ballpark estimate can be obtained by noting that the DRAM memory controller on the AMD Opteron 6100-series multi-core processor [67]) accounts for 15-20% of the chip's power consumption. For a network processor such as EZchip NP-4 [68] (which operates at 50 Gbps and consumes 35 Watts), DRAM controllers would conservatively account for 5-7 Watts. Bearing in mind

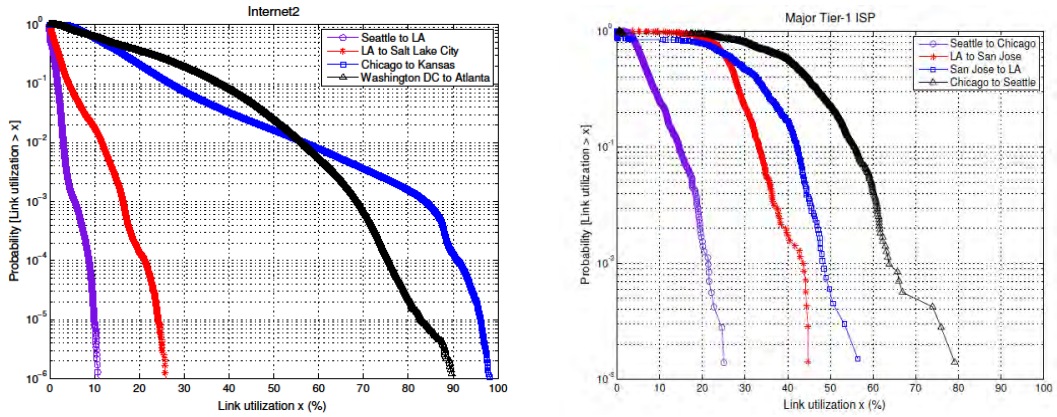
that modern routers have complex memory pipelines across DRAM, SRAM, and on-chip buffers, and often have separate ingress and egress queueing ASICs (as in the CRS-1), it would be reasonable to expect that memory controllers consume at least half as much power as the memory chips themselves.

Based on the above arguments we conservatively estimate that the power consumed by packet buffers, including the memory chips and controllers, would amount to around 10% of the total power of a line-card in a modern high-speed router. This number is high enough to motivate the study in this chapter to optimise the energy consumption of buffer memory, particularly so because buffers are meant to absorb congestion, which is a relatively infrequent event in operational networks, as discussed next.

5.2.2 Link Congestion in Operational Networks

Having seen that buffers consume a non-negligible fraction of the energy in a core router, we now present empirical evidence from carrier and enterprise networks that link loads are quite low for a vast majority of the time, implying that the buffer capacity built into routers to deal with worst-case congestion situations are needed only rarely. We have obtained and analysed traces of link loads (at granularities of seconds, minutes, hours, and days) spanning several years, over nearly a hundred links from backbone and enterprise networks. In this section, we briefly summarise our observations from two backbone networks (Internet2 and a major Tier-1 carrier ISP), and two enterprise networks (a large University of over 40,000 students and a large governmental organisation of over 6,000 employees).

The first backbone network we discuss is Internet2 [69], chosen for the comprehensive data it freely provides on its national long-distance network. We analyse load from over 50 links at 10-second granularity over the past three years (spanning Nov 2007 to Nov 2010). All links are of 10 Gbps capacity, and our general observation was that nearly all links had light loads ($< 30\%$) most of the time. To depict this, we show in Fig. 5.1(a) the complementary cumulative distribution function (CCDF) of the link utilisation, namely the probability that in a random 10-second interval, the load exceeds $x\%$ of the link capacity. The top two curves, corresponding to links from Washington DC to Atlanta and from Chicago to Kansas, were found to be amongst the most heavily loaded links in the Internet2 core. In spite of that, the chance that either of these links had load over 60% in any chosen 10-second interval was no more than one in a hundred. The other two curves corresponding to Seattle



(a) Internet2 (b) Tier-1 ISP (from CAIDA)

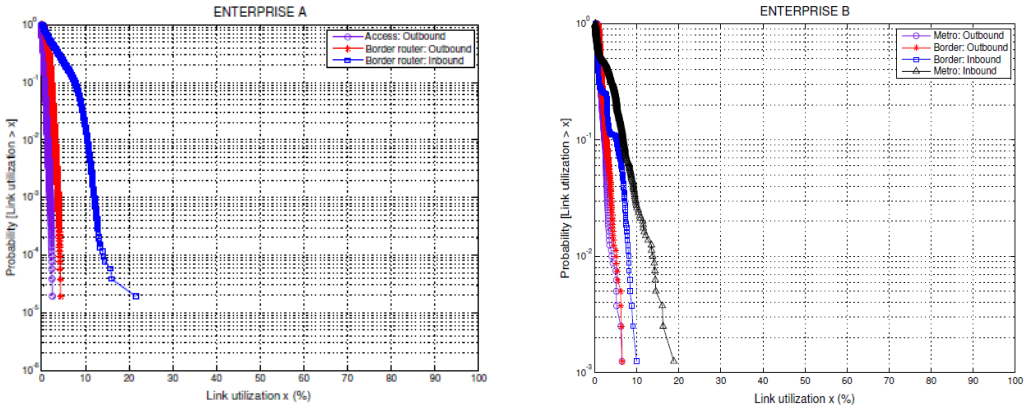
Figure 5.1: CCDF of link load from two backbone networks

to Los Angeles and Los Angeles to Salt Lake City, are more typical of most links on Internet2, with load never exceeding 30%. In fact the average load on many links was well below 20%.

At this point one may wonder that even if the load over a 10-second interval is low, there might well be many spikes in the traffic at smaller time-scales (say over a millisecond), which use the large buffers in the router over that interval. While it is very difficult to get link-load data at time-scales finer than 10 seconds from operational networks over any sustained period of time, a simple argument can be made to show that there can only be a bounded number of such traffic spikes. For example, say the link load is 20% (or lower) over a 10-second interval (we found this to be the case most often), but we are interested in traffic load at a much finer time-scale, say a millisecond. We can use the Markov inequality to bound the number of 1 millisecond slots that have high loads: the Markov inequality states that $P(X > nE[X]) \leq 1/n$ for non-negative X, and hence at most one-third of the 1 millisecond slots within that 10 second interval can have a load greater than 60%, which means that at least two-thirds of the slots have a load lower than 60% (the bound is quite loose and in reality many more slots would have a low load). Thus no matter what the time granularity we pick, there will be sufficient intervals of low load, which is a strong indicator that the buffers for that link would be occupied minimally during most periods.

In Fig. 5.1(b) we plot the CCDF of load on four links belonging to a major Tier-1 ISP. The data for these plots was obtained from CAIDA, and spans a period of two

years (2008-2010). For three of the links, the load never exceeds 60%, and it does so with a 5% probability for the fourth link (Chicago to Seattle). This should not come as a surprise, since carriers in general know their traffic profiles quite well, and provision their links to ensure that loads are not consistently high. Nevertheless, the data corroborates that links in carrier networks today typically have low load for the most part, suggesting that large buffers in core routers are used rarely. Note that this does not preclude transient spikes in link loads during which times large buffers may well be put to use.

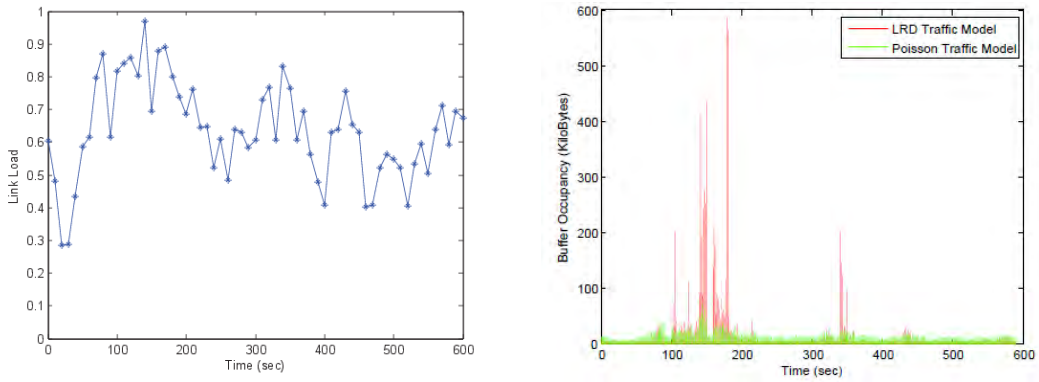


(a) Univ. of New South Wales (40,000+ students)

(b) Govt. Organisation: CSIRO (6,000+ employees)

Figure 5.2: CCDF of link load from two enterprise networks

We also obtained comprehensive load data for several links in two enterprise networks: a large University (UNSW) of 40,000+ students, and a large government organisation (CSIRO) of 6000+ employees. Fig. 5.2(a) shows the CCDF of load (in each direction) on the University's external link (to the Internet) and on an internal link within the campus (the load in the two directions was mostly symmetrical and hence only one direction is shown in the figure), measured at 5-minute intervals over a six-month period (July-Dec 2010). All links had 1 Gbps capacity, and as the figure shows, the loads never exceeded 25%. Fig. 5.2(b) shows the CCDF of link loads on external (to Internet) and internal (between sites) links in the government organisation's network spanning several cities spread across Australia, and again shows that links (of 1 Gbps) had low loads ($< 20\%$) for much of the time.



(a) Link load (b) Buffer occupancy

Figure 5.3: Trace of link load and buffer occupancy (from the Poisson and LRD models) on an Internet2 link over a 10-minute period

5.2.3 Buffer Occupancy from Analysis and Simulation

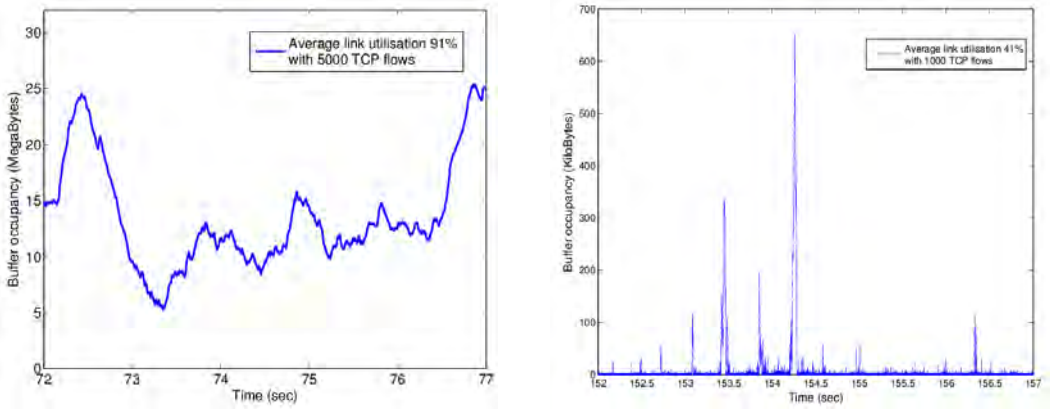
The previous subsection only depicted link load, which is an indirect measure of congestion. In this subsection we explicitly depict router buffer usage, derived from analysis of the traffic load traces and ns2 simulations.

5.2.3.1 Trace Analysis

Our first approach is to take link load data from operational networks, and generate synthetic traffic of matching load, which is then fed into a FIFO queue simulation to generate the buffer occupancy trace. We used two models for generating traffic – a simple Poisson model and a more sophisticated long-range dependent (LRD) model that uses an underlying fractional Gaussian noise (fGn) process with Hurst parameter $H = 0.85$ (the model is described in more detail in §5.5.1.1). To illustrate the outcome, we consider the Internet2 link from Chicago to Kansas, and show in Fig. 5.3(a) the load on that link over a 10-minute period of high load (reaching 96%) observed on 17 Sep, 2009. We used our Poisson and LRD models to generate traffic with matching rate (that changes every 10 seconds), and show in Fig. 5.3(b) the buffer occupancy derived by feeding the packet trace to a simulation of a FIFO queue (for computational tractability we scaled the link speed down from 10 to 1 Gbps). As the figure shows, under the Poisson model, buffer occupancy barely exceeds 40 KB, while with the LRD model, the buffer occupancy shows more burstiness, and is seen to spike occasionally to over 500 KB (which corresponds to about 4 ms of

buffering at the link rate). The point being made is that the traffic loads we are observing on links in operational networks can lead to large excursions in buffer occupancy, but these are very rare, and it would seem wasteful in energy to have all buffers active at all times to deal with such rare events.

5.2.3.2 ns2 Simulation



(a) 1 Gbps core link (load=91%) (b) 1 Gbps core link (load=41%)

Figure 5.4: Buffer occupancy from ns2 of 1000 and 5000 TCP flows sharing a 1 Gbps core link

To corroborate that buffer occupancy fluctuations can be seen with closed-loop TCP traffic, we conducted tens of simulations in ns2. Space constraints limit the observations we can report from various topologies, link speed settings, number of flows and mixes of short- and long-lived flows. We consider one specific scenario with TCP flows from 1000 users multiplexing at a 1 Gbps core link. A vast majority of the TCP flows were short-lived, transferring files with Pareto distributed sizes and having exponential think times between transfers. The scenario is described in more detail in §5.5.2. In Fig. 5.4 we plot the queue occupancy for two link loads over a five second interval: Fig. 5.4(a) has 5 flows per user (5000 flows in total) creating a load of 91%, while Fig. 5.4(b) has 1 flow per user (total 1000 flows) creating a load of 41%. In both cases the buffer size was set to 31.25 MB, corresponding to the delay-bandwidth product. The heavy load scenario shows buffer occupancy to be high much of the time (between 5-25 MB), which presents reduced opportunity for putting buffers to sleep to save energy. The low load scenario on the other hand shows that buffer occupancy seldom exceeds a few tens of KB. This scenario presents ample scope to save energy by putting off-chip buffers to sleep for much of the time.

5.3 Related Work

In recent years there have been many proposals to reduce the energy consumption of telecommunications networks. Some works recommend that energy efficiency be a major consideration in the network design stage [13; 70], such as in choosing appropriate combinations of grooming at the optical WDM layer and switching at the IP layer to reduce overall network energy consumption [70], and in choosing the configuration of interfaces and chassis to achieve the desired switching capacity [13]. Other approaches suggest selectively turning off or underclocking network elements such as interfaces and line-cards [1] to save energy during periods of low load, and yet others suggest new routing mechanisms [71; 72] to redirect traffic towards "greener" areas of the Internet (such as powered by renewable sources). We refer the reader to a recent survey paper [73] for a more comprehensive discussion of proposals for energy conservation in telecommunications networks. While all the above approaches hold promise for substantial energy savings, they require major architectural and/or protocol changes to the network (e.g., delaying packets to aggregate them into bursts, and new routing protocols). These incur high costs and/or overhead for ISPs, making them less likely to be deployed in the immediate future. By contrast, our approach in this chapter, though providing more modest energy savings (around 10% by our estimate), requires very minimal changes and is virtually transparent to operators and users of the network, and stands a much better chance of incrementally being deployed in the short term.

Our work in this chapter also draws inspiration from recent debates on buffering capacity required at core Internet routers. Studies have suggested that buffers can be safely reduced by two to three orders of magnitude [26], and even made as low as a few tens of packets [27]. While the debate about the right amount of buffering required at a router continues (we refer the reader to an article on the topic [74]), reality remains that vendors continue to build routers with large buffers. That being the case, our approach, whereby router buffers are dynamically activated only when needed (thereby conserving energy), is likely to be more palatable to operators, since it eliminates the risk of adverse impact on traffic performance while still yielding a tangible benefit in terms of energy savings. Moreover, since we adjust buffer size at run-time (rather than build-time), ISPs can gradually become comfortable with the idea of operating with reduced active buffers (which they can control in our algorithm), making them more likely to accept routers built with smaller buffers in the future.

To the best of our knowledge our work is the first to propose dynamic adaptation of router buffer size with the primary aim of reducing energy consumption. Earlier works such as [32] have proposed to adapt buffer size primarily for meeting pre-specified loss criteria, delay or utilisation bounds. Our approach is also different: we do not use feedback-based schemes (which may warrant new protocols) or operator input (since operators are quite likely to ask for zero loss and maximum throughput), but instead try to make the buffer size adaptation nearly invisible to the operator (i.e. the operator does not perceive any noticeable effect on traffic performance).

5.4 Dynamic Buffer Adjustment

5.4.1 Buffer Architecture

The focus in this work is on egress packet buffers, which absorb output link congestion, and ingress buffers, that can also absorb (link or fabric) congestion in architectures supporting back-pressure. We do not consider buffers that internally segment and reassemble packets for transmission across the switching fabric(s). The architecture of packet buffer memory varies from one platform to another, and in this chapter we consider a fairly generic three-level hierarchical model (taken from [61]). As shown in Fig. 5.5, it consists of on-chip (within the Network Processor (NP) or ASIC) buffers, off-chip cache (SRAM), and off-chip bulk memory (DRAM). We assume that the on-chip buffer memory has capacity B_I, typically a few tens or hundreds of Kilobytes (for example EZchip's 10-Gbps NP has 256 KB of on-chip packet memory while the Metro NP used in Cisco's CRS-1 router can hold 188 packets on-chip). The SRAM cache capacity is denoted by B_S, of the order of a few Megabytes, while the bulk DRAM has capacity B_D, of the order of several Gigabytes. The buffer memory can support multiple FIFO queues (per interface and/or class-of-service), and head and tail blocks of packets for each queue are moved between memory hierarchy levels as needed in a pipelined fashion.

To meet the speed and latency requirements (for example at 40 Gbps a packet can arrive every 8 ns), a number of memory banks or chips are employed in parallel. To illustrate this in the architecture, Fig. 5.5 shows four SRAM chips, each with 16 data-pins, that operate in parallel to present a 64-bit data bus to the network processor (via the controller) for increased throughput. Since DRAM is slower than SRAM (by a factor of four), DRAM is accessed via a wider data bus – the figure shows DRAM organised as a 4×4 grid, with multiple chips within a row operating

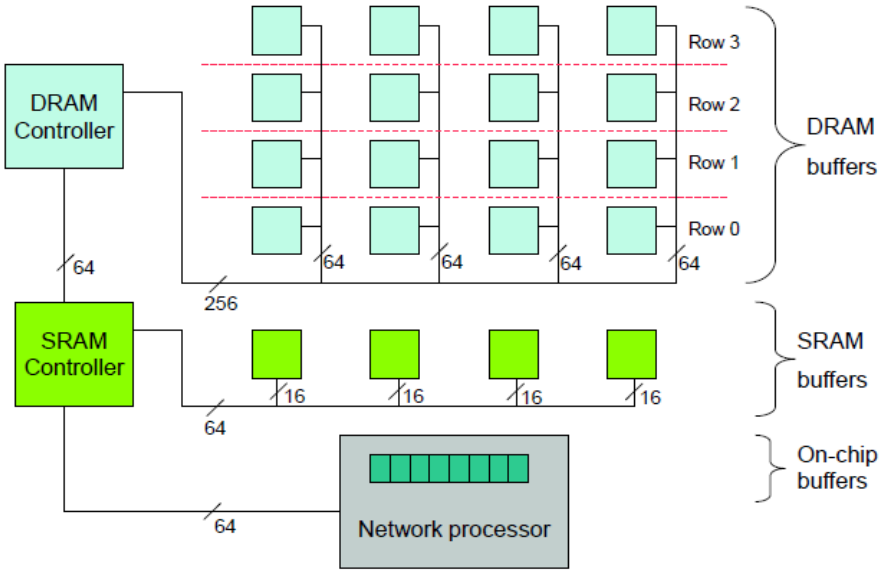


Figure 5.5: Generic model of buffer architecture

in parallel to increase data width, while each successive row adds to buffer depth. A packet stored in off-chip memory will therefore straddle all chips within one row of DRAM (or SRAM). In practice, each row (or each column, depending on the data-bus widths inside the router) of DRAM chips could be realised with a dual in-line memory module (DIMM). The data-bus widths and number of memory chips in the figure are chosen merely to illustrate the concept, and would need to be adapted when analysing a specific routing platform. To be generic, we use N_R to denote the number of rows of DRAM chips in this organisation.

Memory controllers are typically integrated into custom ASICs, and there are often several concurrent controllers; however, for ease of depiction we have shown a single aggregated controller for DRAM and similarly for SRAM. Our algorithm will put to sleep or activate an entire row of memory chips (DRAM or SRAM). As buffer occupancy falls, DRAM row-3 can be put to sleep, followed by row-2, and so on, till at some point the entire DRAM (and its controller) may be placed in the sleep state. Conversely, when buffer size needs to grow, row-0 is activated first (along with the controller), followed by row-1, and so on. We note that the FIFO nature of queues ensures that successive packets can be stored in successive memory locations – such compaction permits unused rows of memory chips to be put to sleep. The DRAM controller is put to sleep if and only if all DRAM memory chips are in the sleep state, and likewise for the SRAM. Based on data-sheets of SRAM and DRAM components, we estimate that it takes no more than 50µs to switch on

(i.e. to bring from inactive to active state) the SRAM (controller and memory), and likewise about 500μs for the DRAM.

5.4.2 Energy Model

The power consumed by DRAM is highly dependent on the frequency of read/write operations. To keep our energy model simple we approximate the DRAM as being in one of three states: active (i.e. high frequency of read/write operations), idle (i.e. little or no read/write operations), and sleep (i.e. read/write disabled). Each row of DRAM (as shown in Fig. 5.5) of capacity 2 Gigabits consumes 2W when active , 200mW when idle , and 20mW when asleep – these numbers are derived from Micron [62] DDR2 specifications. As shown in the figure, larger DRAM buffer (1 Gigabyte in this case) is realised using multiple rows of DRAM chips (4 rows of 2 Gigabits each), and the power therefore scales linearly with buffer size. The DRAM controller, which controls the entire DRAM buffers, is assumed to consume half the power of the entire DRAM memory, namely i.e. 4W when active (i.e. when any row of DRAM is active), 400mW when idle (i.e. when no row of DRAM is active), and 40mW when asleep (i.e when all DRAM rows are in sleep state). We believe this model for DRAM buffer power consumption is simple yet realistic, and can be customised to the architecture of the specific router whose buffer memory is being optimised for energy.

The power consumption of SRAM comprises two parts: a static component due to leakage current that increases with the number of transistors, and a dynamic component that is proportional to switching frequency (i.e. read/write operations). As the static power dominates [75], for simplicity our model assumes that the SRAM power is invariant to workload. We therefore consider SRAM to be in one of two states: active and sleep. For an SRAM of size 4 MB, we assume the active state power to be 4W, and the sleep state power to be 40mW. These numbers are derived from the Cypress [63] data-sheets. As before, the SRAM controller is assumed to require half the power of the SRAM (2W when active and 20mW when asleep).

In our evaluations further on in this chapter, the baseline power (i.e. one which does not employ our energy management scheme for putting packet buffer elements to sleep) is estimated by assuming that SRAM is always active, and a row of DRAM is active or idle depending on whether the buffer occupancy spills over to that row or not. Our algorithm additionally puts both SRAM and DRAM rows into sleep state, and the resulting power savings are expressed as a percentage of the baseline.

Note that we only consider the power consumption of off-chip buffer memory; the on-chip buffers internal to the network processor are assumed to be always-on, and their energy is therefore not explicitly modeled.

5.4.3 Algorithm for Dynamic Buffer Adjustment

To save maximum energy, it is desirable to have the active buffer capacity track the actual queue occupancy, and to put to sleep any off-chip buffer memory that is not needed. However, the risk in doing so is that if a sudden burst of traffic arrives, there may not be sufficient time to activate buffer memory without dropping packets from this burst. In order to control how aggressively or conservatively we want to track the buffer occupancy, we introduce a parameter $\alpha \in [0, 1)$ in our algorithm. The broad idea is to make the total active buffer capacity B at any time instant stay between the lower bound of Q (the current queue occupancy) and upper bound of $B_I + B_S + B_D$ (the maximum available buffer space). One simple way to do this is to use a linear combination of the two extremes, i.e. set $B = \alpha Q + (1 - \alpha)(B_I + B_S + B_D)$. Choosing the extremely conservative setting of $\alpha = 0$ sets active buffers to maximum available buffers, essentially disabling power control. On the other (aggressive) extreme, choosing $\alpha = 1$ would make the active buffer capacity track the exact queue occupancy – this would be equivalent to saying that buffer space is created (by activating memory) as and when a packet arrives. Since memory takes non-zero time to become active, this would result in high loss. Choosing α in $[0, 1)$ allows the energy versus loss trade-off to be controlled. Our algorithm is presented formally below, taking into account that memory can only be activated/put to sleep in discrete quantities (capacity of the SRAM or DRAM row):

The algorithm above takes as input the user parameter α and the current queue occupancy Q (in bytes). If the queue occupancy is found to be low, i.e. on-chip buffer occupancy is below fraction α (step 1), all off-chip buffers are put to sleep (step 2). Otherwise, if occupancy of the on-chip and off-chip SRAM is below fraction α (step 4), only on-chip and SRAM buffers are kept on. If it is deemed that DRAM needs to be on (step 5), the desired DRAM capacity B_A is computed as a linear combination of the total DRAM capacity (weighted by $1-\alpha$) and the current DRAM occupancy (weighted by α) in step 6. The number of rows of DRAM chips that need to be active to achieve this desired DRAM capacity is deduced in step 7, and the corresponding buffer size in bytes (including on-chip, SRAM and DRAM buffers) is

Algorithm 1 Determine active buffer size B (in bytes)

Inputs: Constants: α, B_I, B_S, B_D, N_R
Variable: current queue occupancy Q
Output: Buffer capacity B that should be active

1: **if** $Q < \alpha B_I$ **then**
2: $B = B_I$ /* on-chip buffers only */
3: **else if** $Q < \alpha(B_I + B_S)$ **then**
4: $B = B_I + B_S$ /* on-chip and SRAM buffers */
5: **else**
6: $B_A = (1 - \alpha)B_D + \alpha \max\{0, Q - B_I - B_S\}$
7: $K_D = \lceil \frac{B_A}{B_D/N_R} \rceil$ /* number of DRAM rows */
8: $B = K_D.B_D/N_R + B_I + B_S$
9: **end if**
10: output B

determined in step 8 and returned in step 10.

5.4.4 Discussion

Our algorithm is relatively easy to implement in hardware. It is executed whenever the queue occupancy Q changes, either due to packet arrivals or departures. If $1-\alpha$ is chosen to be a negative power of 2 (e.g., $\alpha = 0.75$ or 0.875), all steps can be performed without any multiplication or division operations, since the product in steps 1 and 3 can be precomputed for given α, and steps 6-8 can be realised using shift and add operations. When the algorithm returns an active buffer size B higher than is currently active, an additional memory row (i.e. SRAM or a row of DRAM) is activated. Likewise, when the required buffer size computed by the algorithm is lower than the current active buffers, the corresponding row of memory chips (i.e. SRAM or of DRAM) is put to sleep. However, to prevent memory components toggling between active and sleep states in quick succession, it is wise to have some hysteresis protection; specifically, our implementation (described in §5.5) mandates that any memory component, once active, should not be put to sleep for at least 1ms.

Though it is easy to envisage more complex algorithms for determining the best buffer size, such as by attempting to predict how queue occupancy will evolve, we believe they will be too complex for real-time hardware implementation. We have intentionally chosen to keep it simple, and have strived to have only one user input parameter α. There are however some unavoidable risks in turning buffer memory

elements on/asleep to save energy. In the worst-case, on-chip buffers of size $B_I = 100$ KB can go from zero to full occupancy within 1.25µs at input rate of 640 Gbps (e.g., if each of the 16 line-cards in the CRS-1 send traffic at 40 Gbps to the same egress line-card), which is much faster than the SRAM turn-on time of about 50µs. Likewise, SRAM of capacity $B_S = 4$ MB can fill within 50µs at 640 Gbps, which is an order of magnitude quicker than the DRAM turn-on time of 500µs. However, such worst-case scenarios are exceedingly improbable, and were never observed in the traces, simulations, and experiments we describe in §5.5. To protect against typical bursts of packets that need to be absorbed while buffer memory is being activated, we found that using $\alpha \in [0.8, 0.9]$ ensured sufficient vacant buffer space for such transients, while still saving significant energy. The router manufacturer may set α at a default value in this range, and network operators can tune it if they prefer a different trade-off point between the benefit (of energy savings) and risk (impact on traffic performance). The next section evaluates our algorithm via trace analysis, simulation, and experimentation.

5.5 Evaluation

We use three methods to evaluate our algorithm: off-line application to traffic traces generated from real Internet data (§5.5.1), on-line simulation of TCP flows in ns2 (§5.5.2), and real-time implementation on an experimental testbed of NetFPGA routers (§5.5.3).

5.5.1 Off-Line Trace Analysis

For our off-line study we generated synthetic Poisson and long range dependent (LRD) traffic traces using time-varying load obtained from empirical data in carrier and enterprise networks (as discussed in §5.2.2). The packet trace was fed into a simulation of our algorithm, and the resulting performance metrics such as power savings and packet loss ratios were obtained.

5.5.1.1 Traffic Generation

For each link considered, we generated Poisson and LRD traffic with mean rate varying as per the link load trace for that link. For example, the load on the Internet2 link from Chicago to Kansas (depicted in Fig. 5.3(a)) is measured every

10 seconds, and we therefore changed the mean rate of the generated traffic over each 10-second interval to match the measured load. The traffic generated by the Poisson model did not exhibit sufficient burstiness to cause queue occupancy to go high (as confirmed in Fig. 5.3(b)), and henceforth we concentrate on the LRD model, which generates burstiness more reflective of Internet traffic. Our LRD traffic generator is derived from Norros' self-similar traffic model [76], which combines a mean arrival rate (that is constant over each 10-second interval) with fractional Gaussian noise (fGn) having Hurst parameter $H \in [1/2, 1)$. We use the filtering method from [77] to generate long sequences of normalised fGn (zero mean and unit variance) samples with $H = 0.85$. These samples are scaled and added to the desired mean volume of traffic in each discretization interval, and the resulting fluid volume is accumulated into packets. The packet size was variable with distribution derived from CAIDA's measurements over 87 million packets at the NASA Ames Internet Exchange (AIX) [78]. We note that to generate traffic traces for sufficiently long periods ($>$ 10 minutes), we had to scale the 10 Gbps links down to 1 Gbps.

5.5.1.2 Dynamic Buffer Adaptation

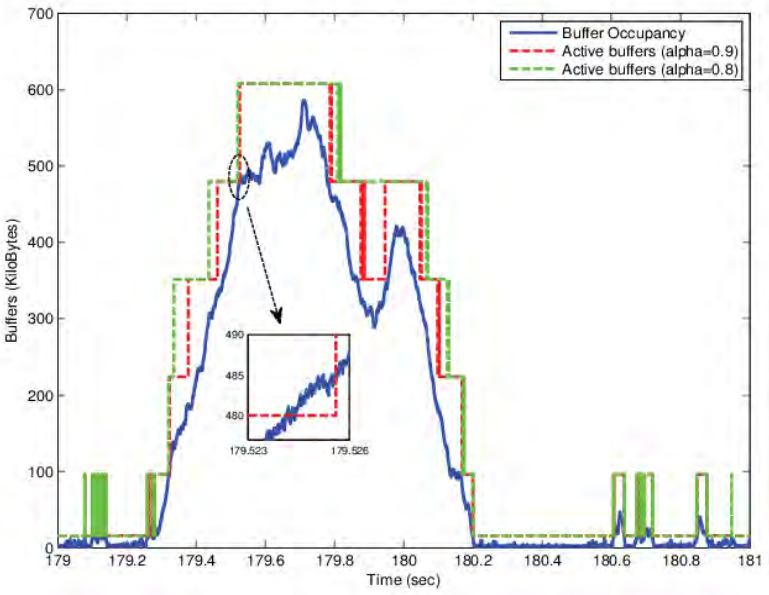


Figure 5.6: Trace of buffer occupancy and active buffers for $\alpha = 0.8$ and 0.9 from algorithm

The packet trace derived as described above from the measured link load was fed to our algorithm for dynamic buffer size adjustment. The measured link loads

were very low for the most part, and to illustrate our algorithm we pick here a 10-minute period, observed on 17 Sep, 2009, of relatively heavy load on the Chicago to Kansas Inernet2 link. The load and corresponding queue occupancy have already been shown in Fig. 5.3, and it was seen that even in this period of relatively heavy load, queue occupancy does not exceed 600 KB. This would easily fit in on-chip and off-chip SRAM, and there would not be any need for storing packets in off-chip DRAM buffers. Nevertheless, to illustrate the operation of our algorithm, we assume that the router has capacity to buffer 16 KB on-chip, 80 KB in off-chip SRAM, and 512 KB in off-chip DRAM (organised as shown in Fig. 5.5). In Fig. 5.6 we show a trace of the buffer occupancy and buffer size set by our algorithm over a chosen 2-second interval. As the figure shows, the buffer size is initially set at 16 KB (i.e. all off-chip buffers are in the sleep state). As the buffer occupancy increases (at around 179.2 seconds), the algorithm reacts by first activating SRAM and then successively each of the rows of DRAM, which subsequently become inactive when the buffer occupancy falls beyond 180.2 seconds. The impact of parameter α on how aggressively the algorithm tries to save energy can also be seen: there are several instances where the $\alpha = 0.9$ curve is seen to track the actual occupancy curve more closely than the curve for $\alpha = 0.8$. Two things to note here are: there are instants where the buffer occupancy overshoots the buffer size (e.g. at around 179.5 seconds, see inset) since it takes time to activate each memory element – and this causes loss that could have been avoided if buffers were always-on. Second, while we have chosen a narrow period of particularly high load, in general loads are quite low and much of the off-chip memory can safely be put to sleep, saving most of the energy associated with off-chip buffers. This trade-off between energy-savings and loss is quantified next.

5.5.1.3 Power vs. Loss Trade-Off

Continuing with our above example of traffic on Internet2's Chicago to Kansas link, we applied our algorithm off-line to the trace with various values of the parameter α to measure power savings and impact on packet loss. When $\alpha = 0$, the algorithm is effectively turned off. We progressively increased α, and found that for $\alpha < 0.7$, the power savings were not very significant (typically $< 50\%$ because the SRAM was active for over 80% of the time), and alongside there were no packet losses induced by the algorithm. This is because at low values of α the algorithm is very conservative, and activates off-chip memory well in advance of the on-chip buffers

overflowing. The SRAM state transition frequency (i.e. from active to sleep and vice-versa) varied between 5 and 10 times/sec, while it was < 1 for the DRAM rows.

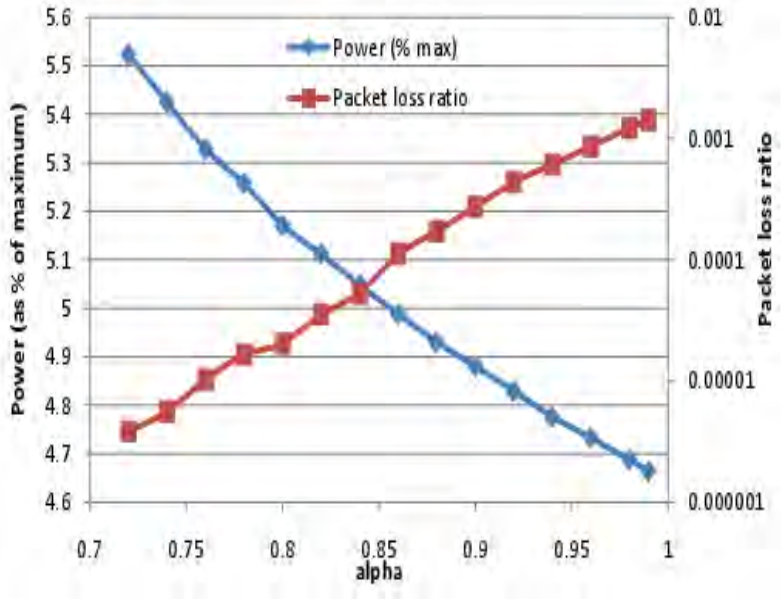


Figure 5.7: Power-savings versus loss trade-off

For the range $\alpha \in (0.7, 1)$, the performance of the algorithm is depicted in Fig. 5.7. Taking the point corresponding to $\alpha = 0.8$, we found that our algorithm activated SRAM for only 2.66% of the time (significantly lower than when $\alpha = 0.7$), and the four DRAM rows for 0.83%, 0.61%, 0.20% and 0.05% of the time, respectively. We found that SRAM toggled between states 62.4 times/sec, while for the four DRAM rows it was in the range 0.1-2.68 times/sec. While the baseline power consumption was 6.66W, our algorithm reduced the average power consumption to 0.344W by virtue of putting off-chip memory to sleep whenever not required, which is only about 5.17% of the baseline power. However, this comes at the cost of increased packet loss due to sudden bursts for which buffer memory is not active yet: for $\alpha = 0.8$, there is loss for about 2×10^{-5} packets (which is within the tolerance of 10^{-3} typical of many SLAs, e.g. [79]). The figure shows that as α increases to 1, power consumption (left axis) falls, while loss (right axis) increases. The operating point on this trade-off curve can be chosen by the network operator, and will depend on the memory availability and configuration on the router, as well as traffic characteristics, cost of power, and criticality of traffic.

5.5.2 On-Line ns2 Simulations with TCP

Though the previous study demonstrated tangible benefits when dynamic buffer adaptation was applied off-line to traces from operational networks, it did not tell us how losses would affect TCP performance, which carries over 90% of Internet traffic today. We therefore incorporated our algorithm into the ns2 simulator, and ran several tens of simulations with different topologies, link capacities, number of flows, and mixes of long and short-lived TCP connections. Here we present results for a small subset of scenarios that illustrate the efficacy of dynamic buffer adaptation with a realistic traffic mix under different loading conditions.

The topology we used comprised of a three-level hierarchy: one core *bottleneck link* fed by 50 edge links, with each edge link aggregating traffic from 20 access links. There were thus 1000 source hosts generating TCP traffic. The core and edge links had capacity 1 Gbps, while the access links had capacity uniformly distributed in $[100, 300]$ Mbps. The mean round-trip-time (RTT) for the flows was 250 ms. The total number of TCP (Reno) flows was varied from 1000 to 5000 (by varying the number of flows per user) to simulate different loading conditions. Our simulations comprised of both short- and long-lived TCP flows. The former models HTTP transfers with Pareto distributed file-sizes (mean 100 KB and shape parameter 1.2) and exponentially distributed think-times of mean 1 second, while the latter represents persistent FTP transfers. The number of long-lived flows (50) accounted for only a small fraction of the total number of flows. These parameter settings are consistent with prior literature and based on measurement studies of Internet traffic. The maximum window size was set to a very large value so that transfers are never limited by the window size. Our simulations were run for over 180 seconds, and all links were equipped with delay-bandwidth buffers. The simulation settings (link speeds and number of flows) are at the limit of the memory and CPU constraints available on our ns2 simulation environment.

Workload	Load	AFCT	Power saved
Low	21.5%	2.233 sec	97.4%
Medium	41.1%	2.244 sec	97.2%
High	59.8%	2.250 sec	83.4%
Heavy	78.6%	2.295 sec	52.9%
Very heavy	90.9%	2.757 sec	11.6%

Table 5.1: Power savings and average flow completion times (AFCT) from ns2 simulations

A sample trace of queue occupancy obtained via simulation from two of the above representative scenarios have already been shown in Fig. 5.4. We repeated the simulations with our algorithm for dynamic buffer adaptation (implemented at the core link), and tried parameter settings α in $[0.75, 0.95]$. The results are summarised in Table 5.1. At low to medium workloads (up to 41%) the off-chip SRAM and DRAM buffers were used very sparingly ($\approx 0.25\%$), thus saving over 97% of the off-chip buffering energy. There were no packet losses induced by buffers turning active/asleep, and so the average flow completion time (AFCT) for HTTP flows was identical to the case when all buffers were always-on. In addition, all values of α in $[0.75, 0.95]$ gave identical results.

Next, when the load went relatively high (i.e. 59.8%), we observed that the off-chip buffers were used about 12.3% of the time. In spite of this, our algorithm resulted in over 83% energy savings. It however induced a very small fraction of packet loss (of the order of 10^{-7}), which barely increased AFCT by a few milliseconds. Even under heavy workload regime (corresponding to 78.7% load), we found that our algorithm could save over 50% of the off-chip buffering energy as the SRAM/DRAM buffers were used for only 40% of the time. Increase in packet loss (of 10^{-6}) and AFCT (< 4 ms) was also negligible.

Finally, under very heavy load ($> 90\%$), off-chip buffers were (unsurprisingly) used nearly 82% of the time, and power savings are limited to about 11%. Even for this scenario, our algorithm induced very small loss (of $< 10^{-6}$, which is again within the tolerance of typical SLAs [79]), and AFCT was barely affected (by no more than 6 ms). These results clearly demonstrate that our algorithm adapts quite well across a wide range of workloads at a core bottleneck link, with minimal impact on TCP traffic performance.

5.5.3 Real-Time Implementation in a Router

The aim of this section is to demonstrate the feasibility of deploying our scheme in hardware, and to quantify the power savings in the presence of real TCP traffic. To do so, we consider the programmable NetFPGA platform in conjunction with hardware-based traffic generators and delay emulators, as described next.

5.5.3.1 Buffer Size Control of the NetFPGA Gigabit Router

A logical structure of the output queue module in the NetFPGA reference design is shown in Fig. 5.8:

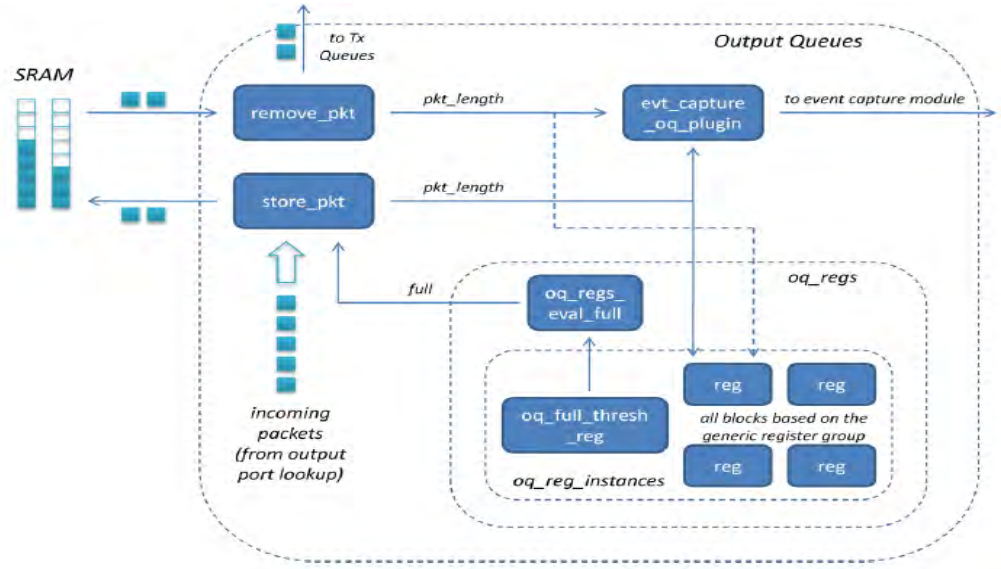


Figure 5.8: A logical structure of the output queue module in the NetFPGA

- Store_pkt module puts incoming packets into destination output queues in the SRAM and updates statistical information (e.g., length of the packet or destination output queue) to on-chip registers. This module will make a decision of saving packets in queues or dropping incoming packets by enquiring buffer size from oq_regs_eval_full module. Remove_pkt module reads packets from the output queues, sends them to corresponding Tx queues in I/O module and also updates statistical data to internal registers.

- Oq_regs_eval_full module fetches buffer size settings from a register named oq_full_thresh_reg and notifies the store_pkt module in the case that the queue is full. Oq_full_thresh_reg is the register saving the buffer size settings and is configurable by users through command line software applications.

- Evt_capture_oq_plugin module calculates the real time buffer occupancy so that it is a perfect module to implement our algorithm. This module also has interfaces to event capture module that outputs all packet-storing, packet-removing and packet-dropping events happening in one output queue to the host by packing these events into a special event packet.

When a packet arrives at the output queue module in User Data Path (see the section 3.2.2), store_pkt parses the packet header to get destination output queue number, informs oq_regs_eval_full module to enquire the buffer size setting of this queue from oq_full_thresh_reg register and updates packet length in a series

of statistic registers. In case that the queue is full, oq_regs_eval_full will generate "full" signal to store_pkt to drop the packet. No matter what happens to an incoming packet (stored, read or dropped), the events can be captured by event capture module, which will package the captured events into an event packet and send it to the software application through PCI bus. Finally we parse all event packets with an application written in C and evaluate the performance of our algorithm.

5.5.3.2 Implementation and Set-Up

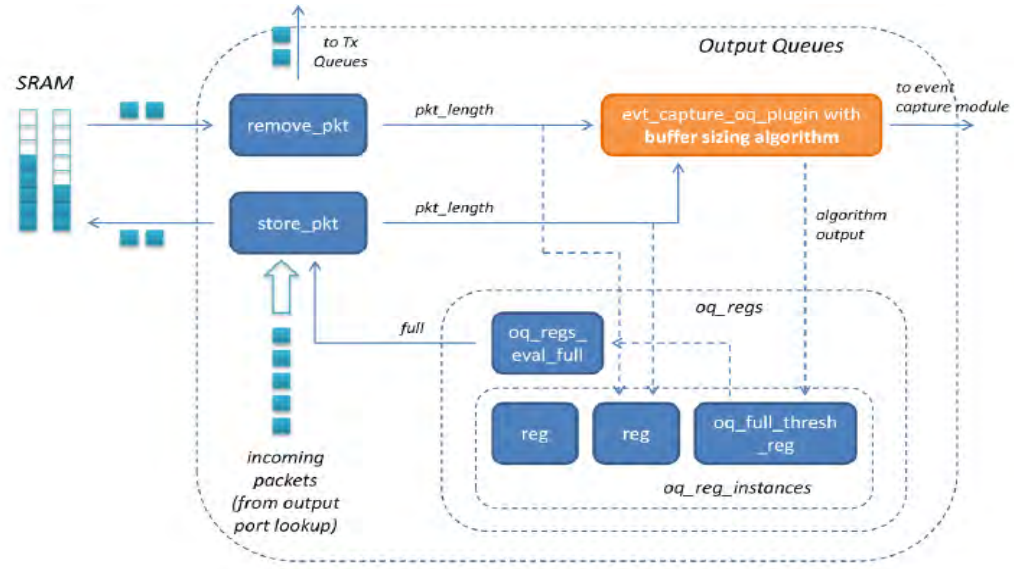


Figure 5.9: The output queue module with the algorithm incorporated

Our algorithm is implemented in the hardware data-plane using Verilog and extends the reference design available at the NetFPGA website for router buffer sizing studies [4]. Specifically, we incorporated our algorithm in the *evt_capture_oq_plugin.v* file located in the output queues module as shown in Fig.5.9. Since the reference NetFPGA gateware provides 512 KB of output queue capacity (internally implemented on off-chip SRAM), we evaluated our algorithm by partitioning this buffer capacity (virtually) into 16 KB of on-chip buffers, 48 KB of SRAM buffers, and 448 KB of DRAM buffers organised as four rows (as shown in Fig. 5.5). The algorithm is executed at every packet arrival or departure instant to determine the size of buffers that should be active. The queue size register *oq_queue_full_thresh* is then updated by the algorithm accordingly, which takes effect after a few clock cycles. Note that our implementation does not explicitly put memory elements to sleep nor does it artificially introduce delays to model memory state transition latencies. The

objective of our experimental work is to demonstrate the feasibility of implementing our scheme in hardware and quantifying the potential power gains. We tried various settings of α, and the results described here are for $\alpha = 0.8$, which was found to yield a good balance between power and loss performance.

The evt_capture_oq_plugin module packages several (typically 340) events at the output queue into a special "event packet" that is sent to the host software for display on the Graphical User Interface (GUI) . We extended the software to extract the queue size information and also log all queuing and dequeuing events, indicating the queue size and queue capacity, so we can plot and analyse them. For our tests the focus was on a single output link at the NetFPGA router. Since the NetFPGA router has only four ports, to emulate a large fan-in we rate-limited the output port – this also lets us make that port a bottleneck link for some tests.

5.5.3.3 Validation with UDP Traffic Burst

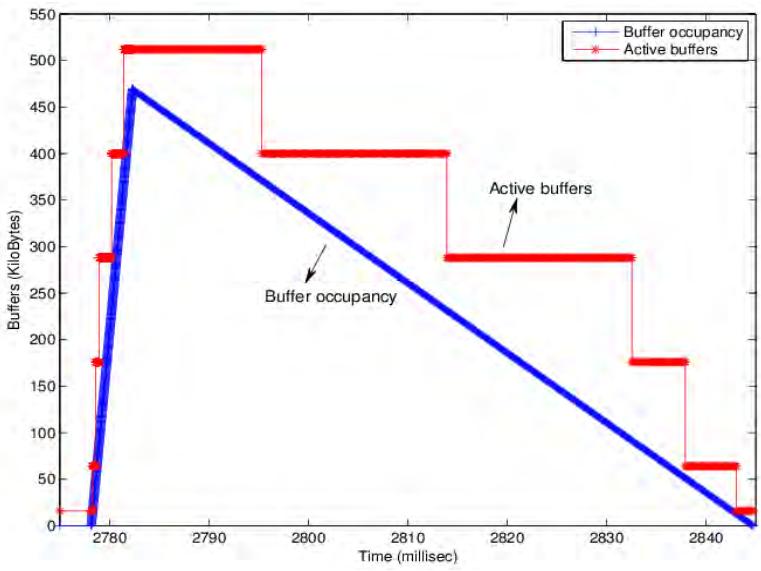


Figure 5.10: Buffer adjustment for UDP burst

The first objective is to validate our hardware implementation. To this end we rate-limited the output port to 62 Mbps and fed in a burst of UDP traffic at four times the output link rate. The burst was generated using a high-precision hardware-based IXIA [36] traffic generator. Fig. 5.10 shows (lower blue curve) how the queue occupancy rises from being empty to nearly full (about 500 KB) in just a couple of ms. Our algorithm is able to detect and respond to these changes in real-time. The figure denotes (upper red curve) the active buffer size triggered by the

algorithm during various stages of the UDP traffic burst. At its peak, the active buffers hits the maximum available buffer size of 512 KB. Once the input burst stops at 2782 ms, the output queue begins to drain and goes empty after roughly 62 ms. The algorithm keeps track of the changes in queue occupancy and adjusts the active buffer size accordingly (the red step lines), finally setting it to 16 KB (the on- chip buffer size) when the queue occupancy hits zero.

5.5.3.4 Power Savings with TCP Flows

In this experiment we had 150 concurrent TCP flows (generated using Iperf) sharing the 123 Mbps link under observation for 180 sec. The RTT was set, using a hardware-based delay emulator from Anue Systems [80], to 35 ms so that the 512 KB of available buffers corresponds to the delay-bandwidth product. To emulate network conditions where this link may or may not be the bottleneck at all times, we introduced on-off UDP traffic (using IXIA) in another downstream link so that the link under observation toggles between being and not being a bottleneck every few seconds.

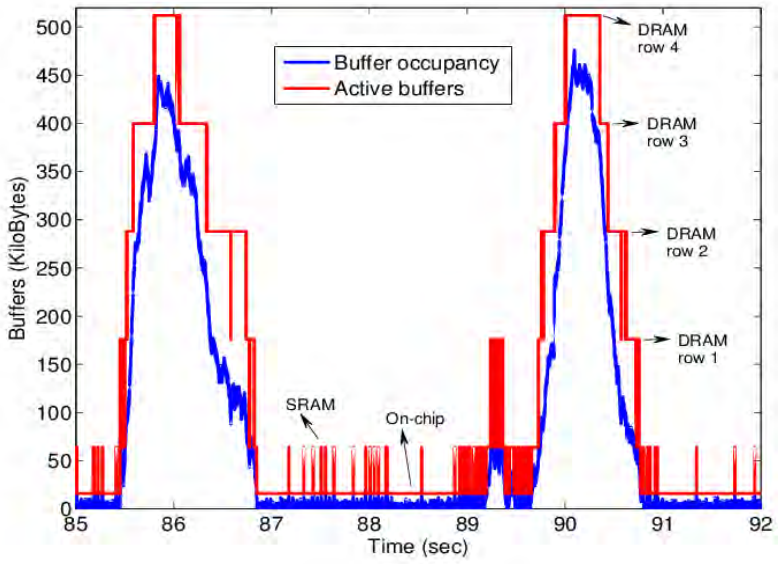


Figure 5.11: Buffer adjustment with 150 TCP flows

With our algorithm running on the NetFPGA, the output queue occupancy trace and the dynamically adjusted buffer size over a 7-sec interval for a chosen run is shown in Fig. 5.11. The link is a bottleneck at around 86 and 90.1 sec, when the queue occupancy rises above 400 KB (all 4 DRAM rows are active), while the link is clearly not a bottleneck between 87-89 sec, when the queue occupancy is just a few

KB (only SRAM gets activated). It is seen that our algorithm adapts dynamically to buffer occupancy.

The algorithm used SRAM buffers 44.3% of the time, while DRAM rows 1-4 are used 38.7%, 27.6%, 17.5% and 7.6%, respectively. This corresponds to a saving of nearly 40% of the off-chip buffering energy. The reader may note that the 40% energy savings for this scenario (88.7% load) are larger than the 11% savings for the similar simulation scenario (90.9% load, see Table 5.1) of the previous section – this is explained by the fact that in the current scenario the link toggles periodically between being and not being a bottleneck link (unlike the simulation scenario in which the link stays a bottleneck), and this fluctuation in load (for the same mean) presents increased opportunity for energy savings.

5.6 Summary

We believe that much of the off-chip packet buffer energy (typically 10% of the total line-card energy) in backbone routers can be saved by selectively putting to sleep memory components when not needed. While packet buffers are just one component of a router, and the energy savings from our proposal are thus limited to around 10%, this reduction comes at virtually no cost: the hardware/software changes required for dynamic buffer size adaptation are minimal. Since these changes are contained within a router, the scheme can be deployed incrementally today without requiring any new network protocols or architectures. The risk of affecting traffic performance is also minimal, since losses will only happen during transients when the memory components are transitioning from sleep to active state, and these can be mitigated to some extent by adjusting the threshold parameter α in the algorithm. The scheme is therefore almost fully transparent to the operator and the user of the network. We hope our work can persuade router manufacturers and operators to consider dynamic buffer size adjustment as a relatively safe and easy way of reducing power consumption.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

The exponential growth of Internet traffic drives ISPs to deploy high capacity routers that consume tens of KiloWatts of power in core networks. Because power awareness was not involved in the initial design of network hardware, architecture and protocols, ISPs are facing serious problems to cope with the huge power cost and complexity of managing heat dissipation. In recent years there have been a lot of research efforts to improve energy efficiency of Internet equipment. The proposed methods present power saving consideration in different aspects of the networks such as profiling power consumption of Internet equipment, reducing power consumption of router components, switching off idle line cards or ports, modifying existing routing protocols and adapting certain hardware to reduce wasted power. These works usually require great architecture/protocol changes in the existing networks that lead to huge implementation cost and service quality risks to ISPs, thus are less practical to be widely deployed. Most of them also use the simulation results as the only proofs to their arguments, which cannot provide solid support to their research without the hardware implementation. In this thesis we provide a practical proposal for improving energy efficiency of Internet equipment and evaluate it with real time implementation in NetFPGA routing platform. Our main contributions in this thesis are highlighted below:

- First, we proposed a fine-grained power model for the NetFPGA Gigabit router, which is composed of per-packet processing energy, per-byte energy for receipt and storage at the ingress to the router, as well as per-byte energy for queueing and transmission at the egress of the router. We devised a

series of experiments to quantify each items of the power model. Our model provides a precise benchmark for researchers who use the NetFPGA platform to prototype new architectures and protocols to evaluate the performance of their achievements. Our work formed two papers [55; 81] and it was presented in the Computer Communications Workshops of IEEE Conference in 2011.

- Second, we proposed a power model for small switches that support IEEE standard 802.3az —Energy Efficient Ethernet. We measured the power consumption of EEE switches in different scenarios and based on an in-depth analysis of the results presented the power model. Our work provides an accurate reference for other researchers to evaluate the performance of their power saving techniques for EEE equipment. This contribution was delivered in the paper [82] and published in the Proceedings of International Conference on High Performance Computing and Simulation (HPCS) in 2012.

- Third, we proposed a practical algorithm to dynamically adjust buffer size, thus much power consumed by off-chip memories can be saved when the workload is low. We proved that link loads are low for a large portion of time so that there is an opportunity to turn off certain amount of buffers during the low-utilisation period. This motivated us to develop a buffer adapting algorithm that requires minimum changes in existing router design and easy to be deployed. We evaluated the algorithm not only by ns2 simulations, but also by a real-time implementation of buffer adaptation in the NetFPGA Gigabit router platform, which presents the feasibility of deploying the algorithm in a real routing system. We contributed this algorithm in the paper [83] and it was published in the Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies (ACM CoNEXT) in 2011.

6.2 Future Works

The work in this thesis focuses on improving energy efficiency of Internet equipment in three areas: router buffer sizing, power consumption analysis of the NetFPGA routing platform and EEE switches. Based on the achievements we have made there are some interesting directions for the future work, which are listed below:

- We already quantified the fine-grained energy consumption of the NetFPGA Gigabit router based on the reference design provided by Stanford group. Since

the NetFPGA is an open platform for which all source codes are free to be modified, we have many opportunities to implement new power saving techniques in different parts of the hardware like PHY ports, routing tables and buffers. We can accurately estimate the power consumption change generated by the new techniques and thus clearly evaluate the performance of the new mechanisms in terms of power saving. Furthermore, 10G NetFPGA router is available in the market now. We can similarly decompose the power consumption of 10G NetFPGA router and provide another benchmark for NetFPGA researchers in the future.

- There have been some reports for power consumption of EEE NICs. Our work in this thesis proposes a power model for EEE switches. Next step should be an analysis of energy consumption of EEE routers when such equipment is available in the market. Since EEE is a relatively new standard and the studies on how to improve EEE performance are still in initial stage, there will be many opportunities for researchers to develop new methods that can make better use of EEE functionality for energy efficiency in the future.

- Our buffer adaptation algorithm presented in chapter 5 is practical and requires minimum changes in existing routers. Thus router manufacturers and network operators are more likely to accept it. It will be even more attractive to ISPs if we can combine this buffer adapting mechanism with some other power saving solutions to achieve better energy efficiency. Following this direction, our next work is to combine this buffer algorithm with link bundle adaptation that dynamically adapts the numbers of the active links in a bundle.

  The technology called Link bundling or link aggregation is widely used in today's core networks. IEEE issued standard 802.1AX to define the link aggregation operation. By bundling several sublinks into one virtual link, network operators can easily achieve bigger bandwidth with low cost and remove the risk of single-link failure. Based on the traces of link loads from two backbone and two enterprise networks, in section 5.2.2 we present the fact that at most of the time the link loads in one network are very low. This observation leads to our argument that the buffers built into routers to deal with worst-case congestion are rarely used and can be deactivated when not in use. It also applies to link bundle because the link loads for each sublink are much lower

than the total loads when traffic is split into several subflows. Thus we have great opportunities to save power of the bundle by conducting power saving operations on under-utilised sublinks. Furthermore, there are two approaches to be chosen as power saving solutions for Ethernet links: adapting link rate to a lower level or shutting down a link. We've had a review of the literature about the adaptation of link rates and bundle in section 2.2.2. Based on the analysis of the previous works, as well as considering the feasibility and complexity of the implementation of the approach in router hardware, we choose the link deactivation as our power saving method for Ethernet link bundles.

The previous works in [19; 20; 21] have proposed several ideas about deactivating sublinks in one bundle for energy efficiency. Three algorithms proposed in [19] take a long time (half an hour or more) to solve the NP-complete problem that optimise the set of cables to be shut down in one network, require network topology and unpredictable traffic demands as inputs and are not practical to be deployed. The work in [20; 21] analyses the traffic data from Internet2 and operates sublinks in one bundle based on a 5-minute time slot, which is too long to cope with the high burstiness traffic happening in a short time and may lead to big loss. To provide a better solution to these problems, we produced a practical algorithm that can adapt active link numbers in one bundle based on fine-grained link load traces to cope with high burstiness traffic (using 100ms time slot). Our algorithm also takes the buffer size into consideration to adapt it according to the traffic. We set up a simple model for joint buffer & link power consumption. Using a certain loss rate as constraint, and traces of link loads, buffer size and link capacity as inputs, our algorithm can optimise the model to achieve the lowest energy consumption by selectively adapting buffer size and the number of active sublinks in the bundle, which means when the traffic load is high the algorithm should make a decision to either activate a sublink or turn on a certain amount of buffers in order to get the optimised power value of the joint model and vice versa.

In additional to the conventional ns2 simulation, we implemented the above algorithm in the NetFPGA routing platform to provide solid support to its feasibility for ISPs. We managed to realize the link bundle adaptation (two sublinks in the bundle) in the reference design of the NetFPGA by adding the following new functionalities:

– We added two new registers in the register pipeline of the NetFPGA to

receive instructions from user applications running in the host. One is used to control the direction of the traffic flows between two sublinks, the other is used as a controller for deactivation/activation operations of a sublink.

- We set up a mechanism in the routing table lookup module to control the output port of each packet, which enables us to distribute all traffic between two sublinks in any way we want. For example, the implementation we have now is to split traffic evenly between two sublinks. This mechanism also enables us to switch all traffic from one sublink to the other before we deactivate it, thus we avoid any induced loss by the link deactivation.
- We successfully found a way to deactivate or reset any port in the NetFPGA board. By inputting certain digit sequences into MDIO pin of the PHY chip, we can put any port into power-down mode or reset it when necessary.

With the link bundle adaptation running in the NetFGPA hardware, we developed a C program running in the host to implement buffer adaptation and our algorithm. This arrangement ensures the flexibility to deploy any new algorithms in the future, just some modifications in C are needed. The C program collects real-time link load information as inputs to the algorithm, makes decisions to operate a sublink or a certain amount of buffers and communicates with the registers in the hardware to conduct these operations.

Up to now we have completed the theory analysis and software/hardware implementation of this joint buffer & link bundle adaptation algorithm. Next we need to conduct a series experiments and simulations to evaluate the performance of the algorithm in terms of power saving. All the above work will form a new paper [84] in the near future, which is out of the scope of this thesis and still under preparation.

The above discussion shows significant potential of our research directions in saving energy. We hope to conduct more detailed investigations in these areas and make more contributions in improving energy efficiency of Internet equipment in the future.

Bibliography

[1] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008, pp. 323–336. ix, 11, 12, 62

[2] P. Reviriego, J. Maestro, J. Hernax0301 andndez, and D. Larrabeiti, "Burst transmission for Energy-Efficient Ethernet," *Internet Computing, IEEE*, vol. 14, no. 4, pp. 50 –57, july-aug. 2010. ix, 16, 17

[3] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. Maestro, "IEEE 802.3az: the road to energy efficient ethernet," *Communications Magazine, IEEE*, vol. 48, no. 11, pp. 50 –56, november 2010. ix, 2, 3, 14, 17, 18, 42

[4] NetFPGA Gigabit router. [Online]. Available: www.netfpga.org ix, 25, 26, 27, 28, 75

[5] Cisco-Systems-Inc., "Approaching the Zettabyte era," *Cisco Systems, San Jose, CA*, 2008. [Online]. Available: www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481374.pdf 1, 23

[6] ——, "Carrier routing system (CRS-1)." [Online]. Available: www.cisco.com/en/US/prod/collateral/routers/ps5763/prodbrochure0900aecd800f8118.pdf 1, 23

[7] R. S. Tucker, J. Baliga, R. Ayre, K. Hinton, and W. V. Sorin, "Energy consumption in IP networks," in *Optical Communication, 2008. ECOC 2008. 34th European Conference on*, sept. 2008, p. 1. 1, 23, 53

[8] M. Gupta and S. Singh, "Greening of the Internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '03. New York, NY, USA: ACM, 2003, pp. 19–26. [Online]. Available: http://doi.acm.org/10.1145/863955.863959 2, 12, 24, 42

[9] M. Yamada, T. Yazaki, N. Matsuyama, and T. Hayashi, "Power efficient approach and performance control for routers," in *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, june 2009, pp. 1 –5. 2, 9

[10] T. Ye, L. Benini, and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," in *Design Automation Conference, 2002. Proceedings. 39th*, 2002, pp. 524 – 529. 2, 8

[11] M. Gupta and S. Singh, "Dynamic Ethernet link shutdown for energy conservation on Ethernet links," in *Communications, 2007. ICC '07. IEEE International Conference on*, june 2007, pp. 6156 –6161. 2, 12, 42, 44, 50

[12] S. Aleksic, "Analysis of power consumption in future high-capacity network nodes," *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 1, no. 3, pp. 245 –258, august 2009. 3, 8, 56

[13] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, april 2008, pp. 457 – 465. 3, 9, 24, 33, 42, 44, 50, 62

[14] K.-H. Ho and C.-C. Cheung, "Green distributed routing protocol for sleep coordination in wired core networks," in *Networked Computing (INC), 2010 6th International Conference on*, may 2010, pp. 1 –6. 9

[15] 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," *Int. J. Netw. Manag.*, vol. 15, no. 5, pp. 297–310, Sep. 2005. [Online]. Available: http://dx.doi.org/10.1002/nem.56510

[16] C. Gunaratne, K. Christensen, and S. Suen, "NGL02-2: Ethernet adaptive link rate (ALR): Analysis of a buffer threshold policy," in *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, 27 2006-dec. 1 2006, pp. 1 –6. 10

[17] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the energy consumption of Ethernet with adaptive link rate (ALR)," *Computers, IEEE Transactions on*, vol. 57, no. 4, pp. 448 –461, april 2008. 11

[18] M. Gupta and S. Singh, "Using low-power modes for energy conservation in Ethernet LANs," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, may 2007, pp. 2451 –2455. 12

[19] W. Fisher, M. Suchara, and J. Rexford, "Greening backbone networks: reducing energy consumption by shutting off cables in bundled links," in *Proceedings of the first ACM SIGCOMM workshop on Green networking*, ser. Green Networking '10. New York, NY, USA: ACM, 2010, pp. 29–34. [Online]. Available: http://doi.acm.org/10.1145/1851290.1851297 13, 82

[20] L. Liu and B. Ramamurthy, "Rightsizing bundle link capacities for energy savings in the core network," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, dec. 2011, pp. 1 –6. 13, 82

[21] ——, "A dynamic local method for bandwidth adaptation in bundle links to conserve energy in core networks," in *Advanced Networks and Telecommunication Systems (ANTS), 2011 IEEE 5th International Conference on*, dec. 2011, pp. 1 –6. 13, 82

[22] "IEEE Std 802.3az: Energy Efficient Ethernet-2010." 14, 42

[23] P. Reviriego, J. Hernandez, D. Larrabeiti, and J. Maestro, "Performance evaluation of Energy Efficient Ethernet," *Communications Letters, IEEE*, vol. 13, no. 9, pp. 697 –699, sept. 2009. 15, 16, 17, 19, 42, 50

[24] P. Reviriego, K. Christensen, J. Rabanillo, and J. Maestro, "An initial evaluation of Energy Efficient Ethernet," *Communications Letters, IEEE*, vol. 15, no. 5, pp. 578 –580, may 2011. 15, 19, 45, 47

[25] C. Villamizar and C. Song, "High performance TCP in ANSNET," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, pp. 45–60, Oct. 1994. [Online]. Available: http://doi.acm.org/10.1145/205511.205520 20

[26] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '04. New York, NY, USA: ACM, 2004, pp. 281–292. [Online]. Available: http://doi.acm.org/10.1145/1015467.1015499 20, 62

[27] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with very small buffers," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, april 2006, pp. 1 –11. 20, 62

[28] A. Dhamdhere, H. Jiang, and C. Dovrolis, "Buffer sizing for congested Internet links," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 2, march 2005, pp. 1072 – 1083 vol. 2. 20

[29] R. Morris, "TCP behavior with many flows," in *Network Protocols, 1997. Proceedings., 1997 International Conference on*, oct 1997, pp. 205 –211. 21

[30] ——, "Scalable TCP congestion control," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, mar 2000, pp. 1176 –1183 vol.3. 21

[31] C. Kellett, R. Shorten, and D. Leith, "Sizing Internet router buffers, active queue management, and the Lur'e problem," in *Decision and Control, 2006 45th IEEE Conference on*, dec. 2006, pp. 650 –654. 21

[32] Y. Zhang and D. Loguinov, "ABS: Adaptive buffer sizing for heterogeneous networks," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, june 2008, pp. 90 –99. 21, 63

[33] GreenTouch. [Online]. Available: www.greentouch.org 24, 54

[34] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via rate-adaptation and sleeping," in *Proceedings Of NSDI*, 2008. 24

[35] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," *NETWORKING 2009*, pp. 795–808, 2009. 24

[36] Ixia traffic generator. [Online]. Available: www.ixiacom.com 30, 76

[37] Ultraview PCI smart extenders. [Online]. Available: www.ultraviewcorp.com/displayproduct.php?part_id=4&sub_id=2 31

[38] Cleverscope oscilloscope. [Online]. Available: www.cleverscope.com/products/cs328a.php 31

[39] Y. Chen, T. Wang, and R. Katz, "Energy efficient Ethernet encodings," in *Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on*, oct. 2008, pp. 122 –129. 37, 38

[40] M. Ellis, N. Jollands, I. E. Agency, O. for Economic Co-operation, and Development, *Gadgets and Gigawatts: Policies for Energy Efficient Electronics*. OECD/IEA, 2009. 41

[41] L. Barroso and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis Lectures on Computer Architecture*, vol. 4, no. 1, pp. 1–108, 2009. 41

[42] "Smart 2020 report," *The Climate Group*, 2008. [Online]. Available: www.smart2020.org/_assets/files/02_Smart2020Report.pdf 41

[43] L. Barroso and U. Holzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33 –37, dec. 2007. 42, 51

[44] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future Internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *Communications Surveys Tutorials, IEEE*, vol. 13, no. 2, pp. 223 –244, quarter 2011. 42

[45] D-link, "DGS-1100 easysmart switches 16/24 port Gigabit switches," *Datasheet*, Sep. 2011. 42, 44

[46] Trendnet, "8-port Gigabit GREENnet switch," *Datasheet*, Jul. 2011. 42

[47] Level-One, "GEU-0820 8-port Gigabit switch," *Datasheet*, Sep. 2011. 42, 44

[48] Hewlett-Packard, "HP E8200 zl v2 switch series," *Technical specifications*, Sep. 2011. 42

[49] R. Seifert and J. Edwards, *The All-New Switch Book: The Complete Guide to LAN Switching Technology.* Wiley-India, 2008. 43

[50] M. Mostowfi and K. Christensen, "Saving energy in LAN switches: New methods of packet coalescing for Energy Efficient Ethernet," in *Green Computing Conference and Workshops (IGCC), 2011 International*, july 2011, pp. 1 –8. 43, 44, 50

[51] Broadcom, "8-GE port switch with integrated SerDes," *Datasheet*, 2006. 43

[52] Vitesse, "24-port layer-2 Gigabit Ethernet switch with 12 integrated copper PHYs and embedded 416 MHz CPU," *Product Brief*, 2010. 43

[53] M. Lau, S. Shieh, P.-F. Wang, B. Smith, D. Lee, J. Chao, B. Shung, and C.-C. Shih, "Gigabit Ethernet switches using a shared buffer architecture," *Communications Magazine, IEEE*, vol. 41, no. 12, pp. 76 – 84, dec. 2003. 43

[54] S. Ricciardi, D. Careglio, U. Fiore, F. Palmieri, G. Santos-Boada, and J. Solé-Pareta, "Analyzing local strategies for energy-efficient networking," in *NETWORKING 2011 Workshops.* Springer, 2011, pp. 291–300. 43, 44

[55] V. Sivaraman, A. Vishwanath, Z. Zhao, and C. Russell, "Profiling per-packet and per-byte energy consumption in the netfpga Gigabit router," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, april 2011, pp. 331 –336. 43, 44, 80

[56] D. Kharitonov, "Time-domain approach to energy efficiency: High-performance network element design," in *GLOBECOM Workshops, 2009 IEEE*, 30 2009-dec. 4 2009, pp. 1 –5. 43

[57] A. Odlyzko, "Data networks are lightly utilized, and will stay that way," *Review of Network Economics*, vol. 2, no. 3, pp. 210–237, 2003. 44

[58] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the energy consumption of Ethernet with adaptive link rate (ALR)," *Computers, IEEE Transactions on*, vol. 57, no. 4, pp. 448 –461, april 2008. 45

[59] D.Dove, "Energy Efficient Ethernet:switching perspective," IEEE 802.3 meeting, Tech. Rep., 2008. 50

[60] M. Pickavet and R. Tucker, "Network solutions to reduce the energy footprint of ICT," in *European Conference on Optical Communication (ECOC) Symposium, Belgium*, 2008. 53

[61] S. Iyer, R. Kompella, and N. McKeown, "Designing packet buffers for router linecards," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 3, pp. 705 –717, june 2008. 55, 56, 63

[62] Micron-Technology-Inc. System power calculators. [Online]. Available: www.micron.com/support/dram/power_calc.html 56, 65

[63] Cypress-Semiconductor. QDR-ii SRAM CY7C1515KV18. [Online]. Available: www.cypress.com/?docID=24145 56, 65

[64] R. Tucker, R. Parthiban, J. Baliga, K. Hinton, R. Ayre, and W. Sorin, "Evolution of WDM optical IP networks: A cost and energy perspective," *Lightwave Technology, Journal of*, vol. 27, no. 3, pp. 243 –252, feb.1, 2009. 56

[65] O. Tamm, "Scaling and energy efficiency in next generation core networks and switches," *ECOC, Vienna*, 2009. 56

[66] G. Epps, D. Tsiang, and T. Boures, "System power challenges," *Technical report, Cisco Routing Research Seminar*, Aug. 2006. [Online]. Available: www.cisco.com/web/about/ac50/ac207/proceedings/POWER_GEPPS_rev3.ppt 56

[67] Advanced-Micro-Devices-Inc. AMD opteron processors. [Online]. Available: www.amd.com/acp 56

[68] EZchip-Technologies-Ltd. EZchip 50 Gbps NP-4 network processor. [Online]. Available: www.ezchip.com/Images/pdf/NP-4_Short_Brief_online.pdf 56

[69] Internet2-Network. [Online]. Available: www.internet2.edu/network/ 57

[70] G. Shen and R. Tucker, "Energy-minimized design for IP over WDM networks," *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 1, no. 1, pp. 176 –186, june 2009. 62

[71] B. Arnaud, "CANARIE: Research networks to help reduce global warming," in *OFC/NFOEC*, vol. 9, 2009. 62

[72] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, and E. Vittorini, "An energy saving routing algorithm for a green OSPF protocol," in *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, march 2010, pp. 1 –5. 62

[73] Y. Zhang, P. Chowdhury, M. Tornatore, and B. Mukherjee, "Energy efficiency in telecom optical networks," *Communications Surveys Tutorials, IEEE*, vol. 12, no. 4, pp. 441 –458, quarter 2010. 62

[74] A. Vishwanath, V. Sivaraman, and M. Thottan, "Perspectives on router buffer sizing: recent results and open problems," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, pp. 34–39, Mar. 2009. [Online]. Available: http://doi.acm.org/10.1145/1517480.1517487 62

[75] X. Wu, J. Li, L. Zhang, E. Speight, and Y. Xie, "Power and performance of read-write aware hybrid caches with non-volatile memories," in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, april 2009, pp. 737 –742. 65

[76] I. Norros, "On the use of fractional Brownian motion in the theory of connectionless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 6, pp. 953 –962, aug 1995. 69

[77] D. Ostry, "Synthesis of accurate fractional Gaussian noise by filtering," *Information Theory, IEEE Transactions on*, vol. 52, no. 4, pp. 1609 –1623, april 2006. 69

[78] CAIDA. packet length distributions. [Online]. Available: http://www.caida.org/analysis/AIX/plen_hist/ 69

[79] Voxel SLAs. [Online]. Available: https://www.voxel.net/sla 71, 73

[80] Anue systems Ethernet network emulator. [Online]. Available: http://www.anuesystems.com/Products_NetworkEmulator_landing.shtml 77

[81] A. Vishwanath, Z. Zhao, V. Sivaraman, and C. Russell, "An empirical model of power consumption in the NetFPGA Gigabit router," in *Advanced Networks and Telecommunication Systems (ANTS), 2010 IEEE 4th International Symposium on*, dec. 2010, pp. 16 –18. 80

[82] P. Reviriego, V. Sivaraman, Z. Zhao, J. A. Maestro, A. Vishwanath, A. Sanchez-Macian, and C. Russell, "An energy consumption model for energy efficient ethernet switches," in *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, july 2012, pp. 98 –104. 80

[83] A. Vishwanath, V. Sivaraman, Z. Zhao, C. Russell, and M. Thottan, "Adapting router buffers for energy efficiency," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, ser. CoNEXT '11. New York, NY, USA: ACM, 2011, pp. 19:1–19:12. [Online]. Available: http://doi.acm.org/10.1145/2079296.2079315 80

[84] Z. Zhao, V. Sivaraman, H. Habibi, and C. Russell, "Joint buffer & link bundle adaptation for energy efficiency," 2012. 83

Index