Green Computing in IEEE 802.3az-enabled Clusters

Dimitar Pavlov dimitar.pavlov@os3.nl
Joris Soeurt joris.soeurt@os3.nl

Supervisors

Dr. Paola Grosso p.grosso@uva.nl
Dr. Zhiming Zhao z.zhao@uva.nl

Universiteit van Amsterdam
System & Network Engineering

July 12, 2012

Abstract

Although major efforts have been put into increasing the energy efficiency of computing environments, energy-efficient networking has not been actively researched until 2006. The most prominent result of research in that area is the IEEE 802.3az standard, which allows active copper Ethernet links to save power by transitioning into low-power mode when no traffic is expected on the link. In this research, we examine 802.3az and other energy-saving Ethernet technologies with the goal of applying them in high performance computing environments. Apart from studying these techniques from a theoretical point of view, we also examine their operation in a test environment, using different implementing devices. Using the results of our experiments, we also introduce an energy model, which aims to optimize the energy usage of applications in a high performance computing environment by taking energy-efficient Ethernet into account. Furthermore, we show a practical example of how to apply the proposed model to an actual application.

Contents

1 Introduction **1**

1.1 Green Networking 1

1.2 Research Question & Approach 2

1.3 Thesis Outline 3

2 Theoretical Study on Green Networking Techniques **4**

2.1 Limiting Transmission Power Based on Cable Length 4

2.2 Sleeping When the Line Protocol is Down 4

2.3 Sleeping When Idle 4

2.4 Summary 7

3 Experimental Study on Green Ethernet and 802.3az **8**

3.1 Experiment 1: Green Ethernet Energy Behavior 8

3.2 Experiment 2: 802.3az Baseline Energy Consumption 10

3.3 Experiment 3: 802.3az Maximum Energy Savings 13

3.4 Experiment 4: 802.3az Energy Consumption vs. Throughput 15

3.5 Experiment 5: 802.3az Energy Consumption vs. Inter-frame delay 17

3.6 Discussion on Experiments 18

4 Applying 802.3az to Cluster Computing **20**

4.1 Time Distribution Estimation 20

4.2 Power Usage Estimation 21

4.3 Prototype Implementation & Results 22

4.4 Discussion 25

5 Research Discussion **27**

5.1 Achievements and Weak Points 27

5.2 Maturity of 802.3az and Vendor Adoption 27

5.3 802.3az and Other Green IT Efforts 28

6 Conclusion **29**

7 Future Research **30**

A Acronyms **33**

B Power Usage Estimation Calculator Code **34**

C Testing Scripts **38**

D Additonal Plots **44**

D.1 802.3az Baseline Power Consumption 44

D.2 802.3az Maximum Energy Savings 44

D.3 802.3az Energy Consumption vs. Throughput 46

D.4 802.3az Power Consumption as Function of Inter-frame delay 47

Acknowledgements

We would like to express our gratitude to all who supported this project. First of all, we would like to thank our supervisors, Dr. Grosso and Dr. Zhao, for their support and feedback. Second, we would like to thank Huawei, Cisco and Extreme for providing us the hardware needed to conduct our experiments with. Also, we would like to thank Ralph Koning for his helpful input and for providing us with access and connectivity to the facilities where we built our different test setups. Finally, we would like to thank Karel van der Veldt and Cosmin Dumitru for their interest in this project and their support.

1 Introduction

Energy efficiency [1] is an increasingly important requirement for computing and communication systems – especially with their increasing pervasiveness. Recent studies have shown that the energy consumption of data cent res accounts for over 150 metric tons of CO2 per year, with the volume still growing [2]. Furthermore, CO2 emissions account for 80% of the contribution to global warming [3]. It is therefore important to work towards "green" computing strategies – techniques that aim to reduce the energy consumption of computing environments and advance environmentally sustainable IT [4][5]. Different strategies for achieving environmentally sustainable IT exist. Examples include, improving computational effiency by optimizing algorithms [6], consolidation by means of virtualization [7], processing of data based on location and energy efficiency [], and green networking.

1.1 Green Networking

While energy efficiency in IT has been an intensely discussed topic in recent years, the subject of green networking has only started attracting attention as of 2006 [8]. In many non bandwidth-intensive environments (i.e. outside the core networking infrastructure of a computational facility), networking capacity is rarely utilized fully. However, it has been known that switches consume a constant amount of power regardless of their bandwidth utilization [9]. The pursuit of solutions that optimize the efficiency of networking equipment, along with advances in energy efficiency concepts, have led to the development of several techniques, which contribute to energy savings. These techniques include powering down switch ports when unused, adjusting transmission power based on cable length, and powering down active switch ports when no traffic is expected. The first two techniques, implemented as *energy detect* and *short reach* respectively, originate from the industry, with equipment manufacturers leading their development [10]. The last technique, used in the 802.3az Energy Efficient Ethernet (EEE) standard, is the product of the Institute of Electrical and Electronics Engineers (IEEE) standardization body [11]. All of these techniques have been implemented partially or fully by networking equipment vendors. In this project we examine all three techniques; however, we focus on the 802.3az standard because of its flexibility and the capability of saving energy based on the specific traffic requirements of links.

The foundation work for the 802.3az standard was done by the IEEE 802.3az working group, with the standard itself being finalized in September 2010. The operation of the 802.3az protocol, has been evaluated from different aspects [12] [8]. It has been shown that the state transitions, which are part of the 802.3az operation, consume a significant amount of power. The result is that rather than linear growth of consumed power (as a function of link load), the protocol exhibits a logarithmic growth [13]. Related to that, it has also been shown that grouping packets can improve the power consumption profile of 802.3az-enabled devices, as by buffering packets for a short time, one can avoid multiple state transitions, which would be the case if multiple packets arrive at an interface with certain gaps between them [14].

However, we have not found a work evaluating the operation of 802.3az in the context of a High Performance Computing (HPC) environment. HPC is a generic term for using supercomputers to solve complicated scientific problems spread over different research areas such as quantum physics, climate

research and planetary movements. HPC is characterized by large amounts of computational power, achieved by using distribution, and high-bandwidth, low-latency networking [15]. Applications in HPC environments are often developed and executed using scientific *workflow management systems*, which govern and guide their execution [16].

In the world of energy-efficient High Performance Computing, the focus lies on optimizing the power usage of the compute systems themselves. However, we were interested in examining the possible advantages of decreasing the energy consumption of switches and network interface cards – namely, by employing the 802.3az standard in such environments. In this project we examined the 802.3az protocol and the practical implications of its operation in a testing environment. We ran experiments that aimed to determine the efficiency of the protocol and to compare it with the efficiency of regular Ethernet switches (without 802.3az support). These experiments were performed using different workloads. Finally, we considered the typical operation and networking traffic patterns within clusters from a theoretical point of view and propose a way to estimate the number of worker nodes needed for a task, so that one can make use of the behavior of the 802.3az protocol.

1.2 Research Question & Approach

In this project, we state our research problem as follows:

How can an application optimize its energy effiency using the 802.3az protocol in cluster environments?

More specifically, we focus on the following problems:

- What is the background of the protocol and how does the protocol achieve its energy savings?
- How can we model the energy characteristics of 802.3az compliant devices?
- How can we apply this model in cluster computing?

In this project we focused on the protocol itself and how it operates in general. Therefore, it is not our main focus to compare hardware from different vendors. Moreover, one of our goals was to propose guidelines for considering 802.3az in HPC scheduling rules.

To be able to fulfil these tasks, we performed our research in three phases. First, we conducted a literature study on the 802.3az protocol. In this phase we studied existing publications to obtain a better insight into 802.3az standard and its operation. The gained knowledge was instrumental in conducting the next phase of our project – an experimental study on energy-efficient networking techniques with a focus on 802.3az. In this phase we created several test environments, consisting of switches, which support the mentioned energy efficiency techniques, and a number of servers. We measured the energy consumption of the switches within the test environments under different scenarios and using different traffic patterns. Besides using throughput as the main variable to determine the amount of energy saved, we also aimed at analyzing the behavior of 802.3az when

confronted with bursty traffic and specific delay intervals, as this type of traffic is more likely to be observed in an HPC environment.

Having analyzed and verified the 802.3az operation, we integrated the results of the first two phases and created an approach for optimizing energy usage within a computing environment. We also created a prototype tool, which estimates the power usage of switches in a cluster environment, based on the time and number of nodes needed for the completion of a computational task. This tool takes the 802.3az operation into account and produces as an output a recommendation about the number of nodes to use for a particular task, so that the networking equipment energy usage can be minimized.

1.3 Thesis Outline

The rest of this paper is organized as follows. In the next chapter we present the literature study we performed on the 802.3az protocol. The obtained knowledge was employed in Chapter 3, where we describe our test setups, the experiments we performed, and the results we obtained. In Chapter 4 we show how we combine the obtained knowledge and results, and how we apply them to create an energy estimation model centered around the 802.3az protocol. In the final chapters – Chapter 5 and Chapter 6, we discuss the achievements and shortcomings of this project, and conclude with the most important lessons learned.

2 Theoretical Study on Green Networking Techniques

In Chapter 1 we introduced three different techniques towards optimizing the energy efficiency of networking equipment have been developed. In this chapter we give a detailed explanation on how these techniques work and achieve energy savings.

2.1 Limiting Transmission Power Based on Cable Length

Using this technique, the switch attempts to determine the length of connected cables, and adjusts the transmission power of the interfaces accordingly. The implementation of this technique is called "Short Reach" and is part of the so-called "Green Ethernet" family of energy-saving networking techniques [10]. Without this functionality interfaces transmit with the maximum necessary power for the maximum segment length possible – 100 meters for 100BASE-TX and 1000BASE-T. However, when a shorter cable is used, such high levels of transmission power are not needed and by transmitting a weaker signal, the transmitter saves energy. This feature can be implemented per interface on the switch only and does not need support on both sides of the link.

2.2 Sleeping When the Line Protocol is Down

Using this technique, the switch puts switch ports and related circuitry in a power saving mode when it detects that the line protocol on these ports is down. The implementation of this technique is called "Energy Detect" and is also part of the so-called "Green Ethernet" family of energy-saving networking techniques. Similarly to short reach, this feature can also be implemented per interface on the switch and does not need support on both sides of the link.

2.3 Sleeping When Idle

The fundamental idea behind this technique is that a switch port should consume power only when actual data is being sent, rather than consume a constant amount of power regardless of the link usage. This technique is implemented by the IEEE 802.3az protocol [11]. 802.3az is designed as a signaling protocol, which allows the transmitter to notify the receiver that a gap in data transmission is imminent and, therefore, the switch port can go into sleep mode.

This is achieved by sending a special type of Ethernet control code – a Low Power Idle (LPI) code, instead of normal IDLE control code. After a predefined period $T_s = time\ to\ sleep$ expires, the link becomes completely quiescent for a period $T_q = time\ quiescent$ until a refresh is sent during the period $T_r = time\ refresh$. This refresh is sent with a different frequency, depending on the linkspeed, and allows the detection of unplugged or failed links. Such detection would not have been possible if the link stayed quiescent for an indefinite period of time. To resume data transmission, the transmitter sends a normal IDLE control code, after which the receiver wakes up and can receive data. The time to wake is defined as $T_w = time\ to\ wake$, which is generally equal to the time it takes the send one maximum length packet for the linkspeed used on the link. The transmitter waits

for a period of time T_w before it starts transmitting. Any data ready to be sent is buffered so that no data is lost. An illustration of these state transitions is shown in Figure 1.

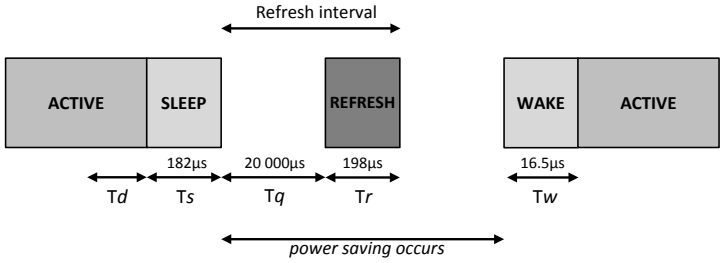


Figure 1: Illustration of the 802.3az state transitions (timing for 1000BASE-T linkspeed)

These state transitions of 802.3az can be visualized in a state diagram, as shown in Figure 2. In this diagram, *states* are shown by vertical lines, and *transitions* are shown as arrows from source to destination state, with the reason for the transition shown as a note.

The five states are defined as follows:

1) ACTIVE – Normal, fully operational state. *2) QUIET* – Quiescent state in lower power mode. *3) WAKEUP* – Link is powering on. *4) SLEEP* – Link is powering down. *5) REFRESH* – Link is being refreshed.

Each PHY that supports 802.3az advertises this capability during the autonegotiation phase of the Ethernet link establishment. Both sides need to support 802.3az in order to use it. If one or both sides do not advertise their support, the link operates in legacy mode and no energy is saved. This means that both the switch and the end-device's NIC have to support the protocol.

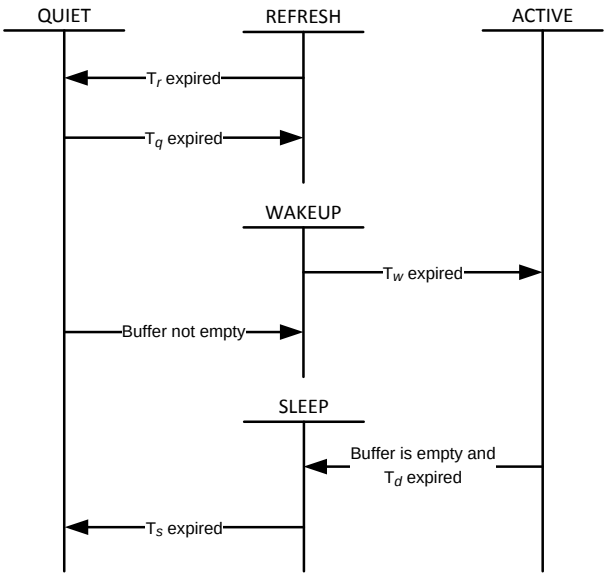


Figure 2: State diagram of 802.3az.

2.3.1 Protocol Overhead

Ideally, the power consumption of the switch would be linearly proportional to the load of the switch. Unfortunately this is not the case. Besides the baseline power usage of the switch itself, 802.3az introduces an additional overhead – the time and power needed for state transitions.

State transitions occur when the switch needs to go from active into sleep mode, or into any other state. Since state transitions need some time and power to be performed, this decreases the energy efficiency of the switch when compared to the

ideal situation. This was first studied and explained by Reviriego et al in 2009 [14]. Their simulated results for 1Gbps are shown in Figure 3. The most inefficient case is when a single frame, surrounded by transmission gaps, has to be sent when the corresponding switch port is in sleep mode. In this case, the port needs to wake up (T_w), send the frame (T_{frame}) and then go into sleep mode again (T_s). The efficiency of the protocol can be calculated for different transmission speeds by using Formula (1). The calculated values can be examined in Table 1. In this calculation, the time to send a maximum size frame (1500 bytes) has been used as value for T_{frame}.

$$FrameEfficiency = \frac{T_{frame}}{T_w + T_{frame} + T_s} \tag{1}$$

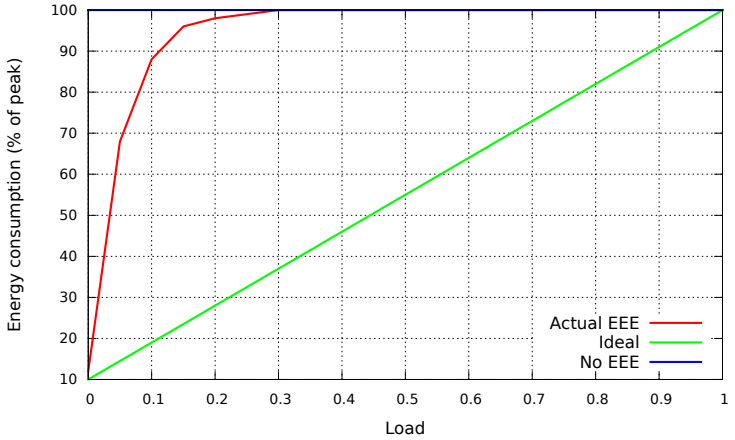


Figure 3: Energy consumption versus load for 1Gbps (Simulated in Matlab) [17].

Protocol	Min T_s (μs)	Min T_q (μs)	Min T_r (μs)	Min T_w (μs)	Min T_{frame} (μs)	Frame efficiency
100BASE-TX	200	20 000	200	30	12	34.28%
1000BASE-T	182	20 000	198	16.5	1.2	5.70%
10GBASE-T	2.88	39.68	1.28	4.48	0.12	14.01%

Table 1: Table of different timing values per Ethernet standard [11].

2.3.2 Timer Parameters

The 802.3az standard defines the signaling between connected network devices, but the protocol does not specify a power-saving policy; i.e. when an interface should go into power saving mode. This policy is up to the implementation of each vendor. Example policies [12] are:

1. *Simple*, in which LPI is communicated after a short time period of empty transmit buffer and the link is reactivated immediately after a packet is to be transmitted;
2. *Buffer-and-burst*, in which LPI is communicated after a short time period of empty transmit buffer and the link is reactivated after a large enough burst arrives or until a timer expires. This correlates to the grouping of packets in [14];
3. *Synchronized bursts*, in which the link is de-activated and activated in predefined intervals to which the data is synchronized.

2.3.3 Deeper Sleep Negotiation via LLDP

The 802.3az standard specifies an additional feature, which enables sender and receiver to renegotiate the wake time T_w after the link is established. This is done using the IEEE 802.1ab Link Layer Discovery Protocol (LLDP). The purpose of this feature is to enable devices to go into a deeper sleep mode. The deeper sleep mode saves even more energy, but takes longer to wake up from.

The result of this negotiation is the balance between the desired time to spend in deep sleep mode, the needed time to wake up and the maximum possible delay time the transmitter can tolerate. This delay depends on the buffer size of the transmitter, as, otherwise, dropping of data may occur while the transmitter waits for the receiver to wake up. The negotiation of T_w does not have to be symmetrical and can result in different times for egress and ingress traffic.

2.4 Summary

In this chapter we introduced the most prominent green networking techniques. While the first two techniques can be implemented per interface on the switch only, the last technique relies on signalling and, therefore, has to be supported by both sides of the Ethernet link. While these techniques can be used in combination, only the energy efficiency of the third technique is based on the amount of traffic flowing through the interfaces, and thus, has the greatest potential for energy savings during active switch operation. The next chapter evaluates the actual energy savings of all three techniques using implementing equipment within a test environment.

3 Experimental Study on Green Ethernet and 802.3az

In the experimental phase of our research, we performed different experiments, using both Green Ethernet- and 802.3az-enabled devices. The goal of this phase was to observe the energy behavior of Green Ethernet- and 802.3az-enabled switches under different conditions, and to use the obtained information to construct energy profiles of the used devices. The variable parameters in our experiments were, among others, the device, the linkspeed, the transmission rate, the transport protocol and the inter-frame delay.

Although we conducted the experiments using several devices of different vendors, it was certainly not our goal to evaluate or compare these devices in any way. The reason to use multiple devices was to establishing whether they would react in a similar manner to similar traffic patterns, and, thus, to better understand the behavior of the different energy-saving techniques. We used the obtained energy consumption data as input for the next phase of our research (see Section 4). In this chapter we discuss only the most relevant to our research test results, while the remaining results can be found in Appendix D.

3.1 Experiment 1: Green Ethernet Energy Behavior

This experiment analyzes the energy usage of Green Ethernet-enabled switches. Its purpose is to examine and analyze the behavior of Green Ethernet in general and, also, of the individual functions of Green Ethernet. The experiment also allowed us to determine quantatively the energy saving capabilities of the technology.

3.1.1 Testbed Description

The testbed (see Figure 4) consisted of one Cisco SG-300-28 switch [18] (hardware version 1, software version 1.1.2.0) connected via Ethernet (port 1) and serial console (RS-232) to a monitoring server. Furthermore, between 1 and 9 servers were connected using 1000BASE-T Ethernet NICs to ports 2-10 of the switch. The remaining Ethernet ports of the switch were connected to a regular non power-saving switch (implementing neither Green Ethernet, nor 802.3az). All cable lengths were below 5 meter. We used a Schleifenbauer Power Distribution Unit (PDU) [19] to measure power in this setup as well as SNMP polling of the switch to obtain the reported power usage. All used Ethernet cables had a length of below 5 meters.

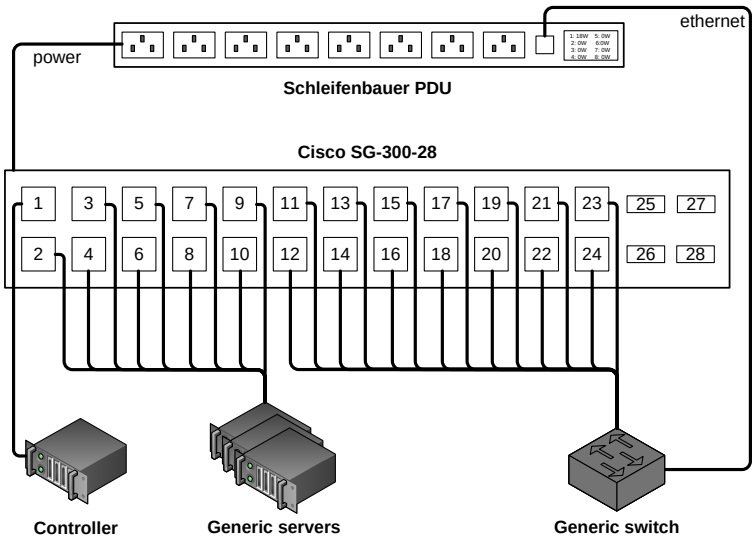


Figure 4: Setup for Experiment 1

3.1.2 Method

The experiment was performed multiple times – with the *Short Reach* and *Energy Detect* features both enabled and disabled. The experiment was conducted with 0 to 28 connected switch ports at 1Gbps. Some of the ports were brought into the UP state by connecting servers, while others were brought into the UP state by connecting them to a generic switch. Due to our preliminary testing showing that throughput did not influence the energy usage, we omitted transmission speed as a variable in this experiment.

3.1.3 Results & Analysis

The results of the experiment are given in Figure 5. This figure clearly shows that the "Energy Detect" feature saves most energy with no systems connected to the switch. It also shows that this technique does not save any power when all switch ports are connected to systems. This result is expected, as the energy detect technique saves energy by powering down unused circuitry when the line protocol is down – i.e., when either no host is connected, or the connected host has been powered down.

A reverse effect can be seen in the energy consumption profile when the "Short Reach" feature is enabled. This technique affects energy consumption the most when all links are used, as power saving resulting from limiting the transmission power of switch ports may only take effect when the switch ports are active and transmitting.

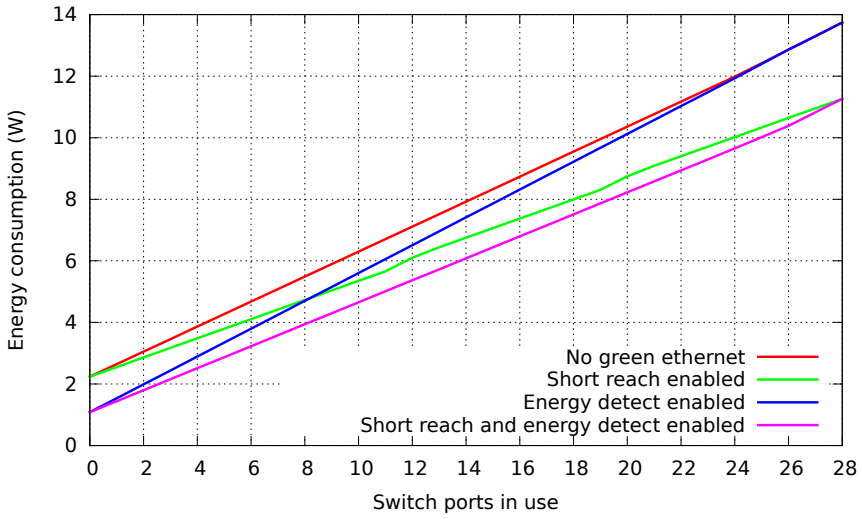


Figure 5: Energy consumption profile of Cisco SG-300 (Green Ethernet)

3.1.4 Discussion

By examining our measurements, we found that the results provided by the Schleifenbauer PDU were unreliable, as our devices consumed less power than the minimum needed for accurate results. The power usage reported by the switch via SNMP under certain conditions was always exactly the same when the test was performed multiple times under the same conditions. We therefore assume that these results are not obtained by direct measurements within the switch, but are synthesized by the switch software based on certain parameters. Unfortunately, due to the unreliability of the PDU results, we were not able to verify this observation. As a result, we would advise approaching the obtained results with scepticism. Furthermore, since these test results indicate that the energy consumption is not influenced by the traffic load in the context of Green Ethernet, we did not use the obtained measurements further in our research.

3.2 Experiment 2: 802.3az Baseline Energy Consumption

This experiment measures the energy consumption of an 802.3az-enabled switch with multiple switch ports connected, but with no traffic traversing the switch. The purpose of this experiment is to determine the power consumption of our test equipment in an idle state. The outcome of the experiment allowed us both to create a reference power consumption baseline for our next experiments, and to determine the power savings 802.3az yields on a completely idle switch.

3.2.1 Testbed Description

The testbed (see Figure 6) for this experiment consisted of one 802.3az-enabled switch connected to two test servers, which were used as source and destination respectively for the generated traffic.

The servers were each equipped with a dual port Intel I350-T2 NIC [20], which supports the final 802.3az-2010 standard. The source server was connected to port 1 and the destination server – to port 22 of the switch. All other adjacent ports were looped in a daisy chain fashion, so that traffic entering port 1 is switched to port 2, which is externally linked to port 3 and so on. Every group of two adjacent ports was assigned a single unique VLAN in order to force the traffic to use the external path, instead of the switch backplane. This ensured that all ports see the same traffic and the same load generated at the source server. This, in turn, made every switch port undergo the same amount of state transitions, and in the same order, as each of its neighboring ports. The short time intervals at the beginning and end of the datastream, where not all ports are actively receiving/transmitting data, were omitted from the measurement results. Furthermore, a monitoring/control server was connected to port 23, which was used for configuration and measurement. The monitoring/ control server was also connected to the console port of the switch using a serial (RS-232) cable. All used ethernet cables had a length shorter than 5 meters.

The power inlet of the switch was connected to a Racktivity ES6024-16 PDU [21], which was used for measuring the power consumption of the switch. The PDU was also connected to the monitoring/ control server, using a separate management network, for reading out the power consumption using the Simple Network Management Protocol (SNMP). The PDU provided energy consumption measurements once per second.

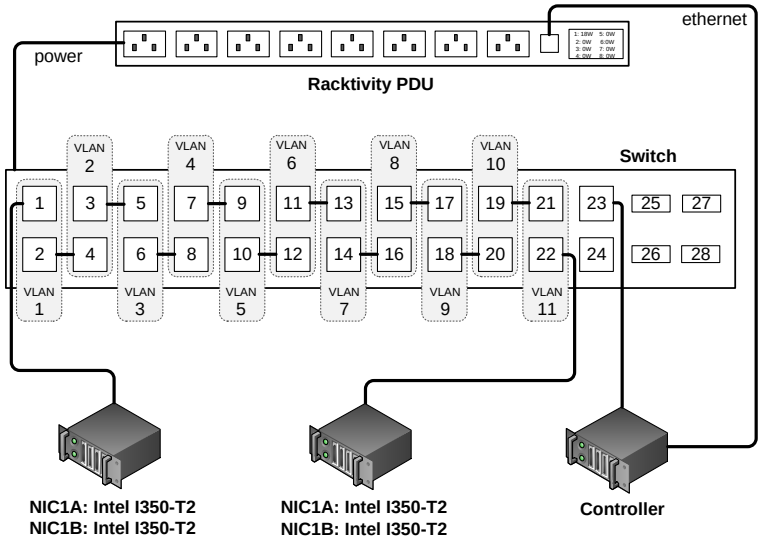


Figure 6: Test Setup for Remaining Experiments

3.2.2 Method

The experiment was conducted with three different 802.3az-enabled switches (Cisco SG300-28 with hardware version 2 and software version 1.1.2.0, Extreme x440-24p [22] and Huawei S1728GWR-4P [23]) for a duration of 5 minutes per run. Furthermore, for each switch the experiment was run 4 times, each run with a unique combination of linkspeed (100BASE-TX or 1000BASE-T) and 802.3az

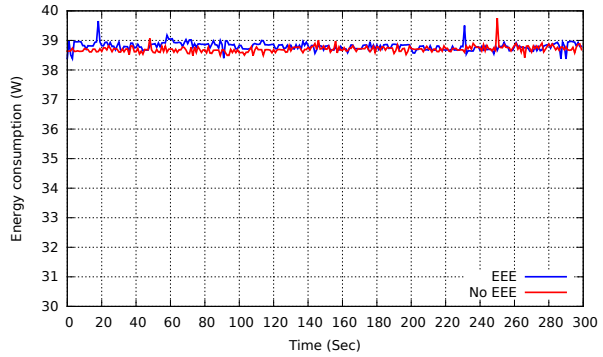
state (enabled/ disabled).

3.2.3 Results & Analysis

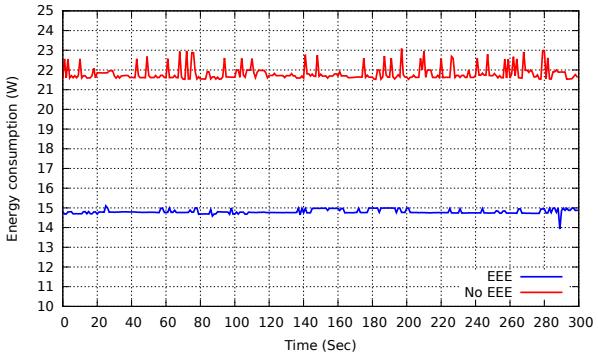
The results of this experiment when using 1000BASE-T linkspeed are given in Figure 7.

The results of the Huawei and the Cisco switches do show a clear decrease in power consumption when the 802.3az protocol is enabled and the linkspeed of all ports is 1Gbps. Furthermore, the power consumption of these devices when using 802.3az is more steady than without 802.3az.

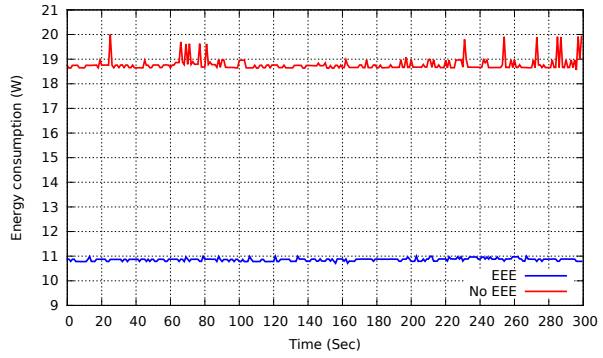
The Extreme switch, on the other hand, consumed the same amount of power with 802.3az both enabled and disabled. Since that switch did not take advantage of 802.3az, we did not perform any further experiments with it.



(a) Extreme x440-24p



(b) Huawei S1728GWR-4P



(c) Cisco SG300-28

Figure 7: Baseline (1000BASE-T)

When performing the experiment at 100Mbps linkspeed with 802.3az enabled, we measured a decrease in energy consumption of only 1 Watt with the Huawei and the Cisco devices – a decrease of about 9% for both. Since most modern HPC environments are presumed to use 1Gbps connectivity

or faster, we determined that results with 100Mbps linkspeed are not relevant in the context of this research, and omitted them from this section. However, we believe that this test case needs to be examined as part of future research. The result data is given in Appendix D, Figures 14(a) and 14(b).

3.2.4 Discussion

When examining the results obtained with the Extreme switch, we did not observe a similar energy behavior as the one exhibited by the other two devices. However, we attribute that to the fact that, although the device itself *is* equipped with an 802.3az-enabled Ethernet controller (the BCM54380), 802.3az is not advertised as a feature for this switch. Also, 802.3az can only be enabled on the device by using a low level debug console – one that would be inaccessible to regular users. Although the debug console shows a number of 802.3az state transitions when using 802.3az enabled NICs, the results data shows no difference between using 802.3az and not using it. One explanation could be that although an 802.3az Ethernet chip is integrated into the switch, the Printed Circuit Board (PCB) design does not take this feature into account and, therefore, is unable to save energy. Another explanation would be that the firmware of the device is not capable of controlling it in such a way, as to save energy with 802.3az. For these reasons we determined the Extreme switch to be inapplicable to this research, and did not use it in further experiments.

We cannot explain why the energy consumption seems to be more steady when using 802.3az; however, when interpreting the plotted results, one needs to take into account the fact that the measurement precision of the PDU was one sample per second, and thus, averaging may already have occurred without our knowledge.

3.3 Experiment 3: 802.3az Maximum Energy Savings

This experiment estimates the maximum energy savings achievable with the 802.3az protocol by contrasting the maximum and minimum energy consumption of a switch. The purpose of this experiment is to aid in the creation of a power profile for a switch. By estimating the maximum and minimum energy consumption of a switch, one can give the energy savings in other circumstances as a percentage of what is possible with that switch.

For this experiment we used the same testbed as in the previous experiment (see above).

3.3.1 Method

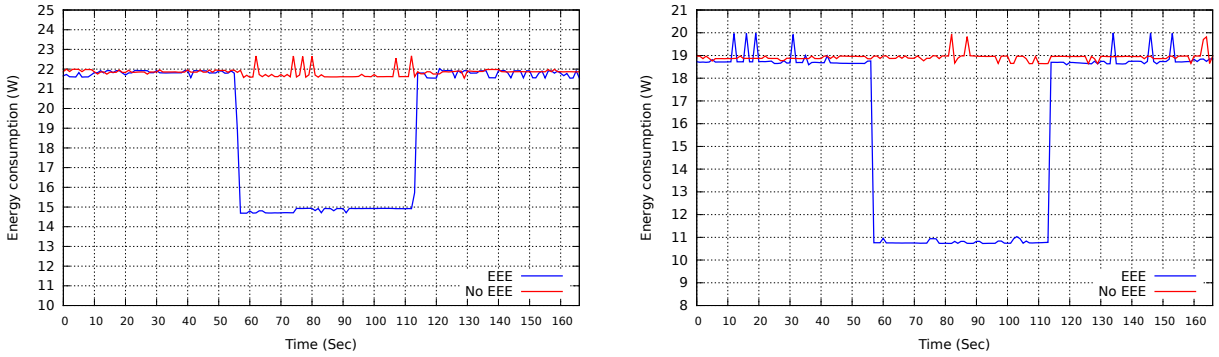
This experiment was conducted by transmitting data at maximum link speed for two consecutive timeperiods of 60 seconds, which were separated by a period of no transmission (60 seconds). Traffic was generated using Iperf [24] and without using any traffic shaping. The experiment was conducted with both the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP), and with 100BASE-TX and 1000BASE-T linkspeeds.

3.3.2 Results & Analysis

The results of this experiment when using 1000BASE-T and TCP traffic are given in Figure 8.

When using UDP traffic in combination with the same linkspeed, the plotted results (given in Appendix D, Figure 17) have the same shape as when using TCP traffic. The results from the tests using 100Mbps linkspeed can be found in Appendix D, Figures 15 and 16.

As can be seen from the TCP plots, this experiment shows a difference of about 8W (about 32%) for the Huawei switch and 7W (about 42%) for the Cisco switch when using maximum link load and no link load. As comparison, the energy usage of the switches is also given when the same experiment was performed with 802.3az disabled. In that case, we see that the switches use a constant amount of power regardless of the link load.



(a) Huawei S1728GWR-4P

(b) Cisco SG300-28).

Figure 8: Maximum energy savings (TCP, 1000BASE-T)

3.3.3 Discussion

Although the plotted results clearly show a difference in energy consumption related to load when using 802.3az, they do not show the exact relationship between power usage and transmission rates. Since this information is significant for the application of 802.3az to cluster computing, this was determined in the next experiment.

Further, our experiments with the Huawei switch produced odd results. Our test data (given in Appendix D, Figure 15(a)) shows that the switch does not take advantage of 802.3az at that linkspeed and, therefore, saves no energy. However, since most HPC environments are assumed to be using 1Gbps connectivity, or faster, we did not look into this issue further, although we do believe it should be examined as part of future research.

3.4 Experiment 4: 802.3az Energy Consumption vs. Throughput

This experiment determines the energy consumption of the 802.3az switch using different transmission rates. The goal of this experiment is to determine the relationship between energy consumption of 802.3az-enabled switches and throughput.

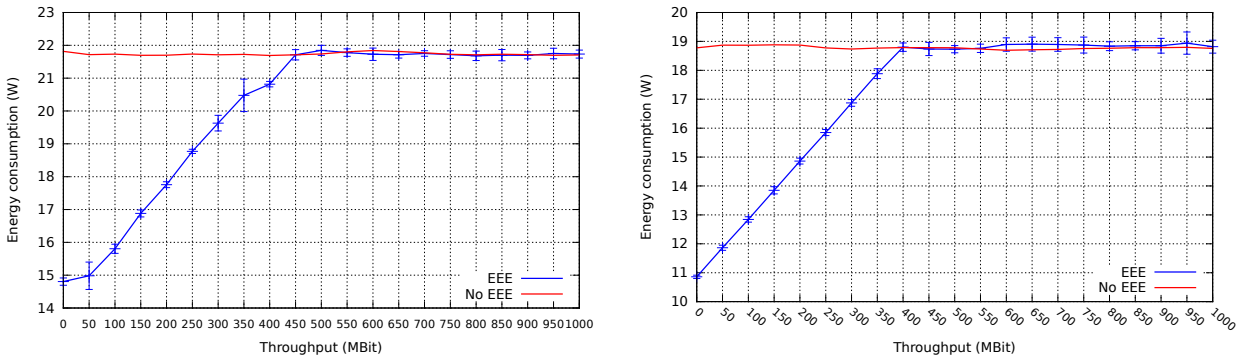
For this experiment we used the same testbed as in Experiment 1 (see Section 3.2.1).

3.4.1 Method

In this experiment we generated traffic from the source to the destination server in our test setup. We performed multiple test runs for this experiment, setting the desired transmission rate at each run. In parallel, we measured the energy consumption of the switch and related that to the different throughput levels. Since our power measurement equipment (Racktivity Racktivity ES6024-16 PDU) only reported power once per second, we ran each test for a duration of 5 minutes and averaged the consumption over that time period. For limiting the transmission rate, we used both the built-in functionality of Iperf, as well as the `tc` Linux traffic control tool [25]. Two methods were used since Iperf only supports traffic rate limiting of UDP traffic. However, since the measurements with both methods did not differ, we only show the data gathered when using `tc` as a rate-limiting tool.

3.4.2 Results & Analysis

The results of this experiment when using 1Gbit linkspeed and TCP traffic are given in Figure 9. The results from the same test performed with UDP traffic are given in Figure 10. The results from the tests performed at 100Mbps can be found in appendix D, Figure 18 and 19. Please note that the given plots show averages of energy consumption over a 5-minute time interval and, thus, the standard deviation is also given in the plots.

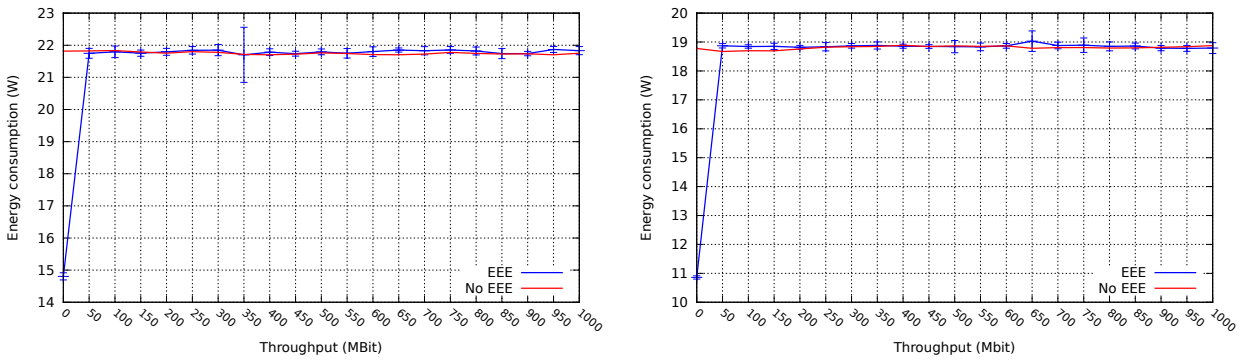


(a) Huawei S1728GWR-4P

(b) Cisco SG300-28

Figure 9: Energy consumption vs. throughput (TCP, 1000BASE-T)

When using TCP traffic, both switches show a linear increase in power consumption running up from the baseline power consumption to maximum power consumption at around 450Mbps for the Huawei switch and 400Mbps for the Cisco switch. However, when using UDP traffic, both switches consume maximum energy – even at traffic rates as low as 50Mbps. Although we performed the experiments multiple times and used other throughput testing tools (BWCTL [26], Netcat [27]), we achieved the same results each test run.



(a) Huawei S1728GWR-4P

(b) Cisco SG-300-28

Figure 10: Energy consumption vs. throughput (UDP, 1000BASE-T)

3.4.3 Discussion

We could not find an explanation for the maximal energy consumption consumption when using UDP traffic. The difference between TCP and UDP is unexpected, as the switch only forwards frames on layers 1 and 2, and is not layer 4 aware. Furthermore, when using 50Mbps out of the available 1Gbps bandwidth, there should be enough inter-frame delay for switch ports to go into sleep mode. However, due to the time frame of this project, we did not look into this matter further and believe this issue should be examined as part of future research.

Overall, both devices confirm the simulations performed by earlier research (see Section 1). The throughput at which the energy consumption reaches the maximum value matches the simulated results as shown in Figure 3. The difference in the "tipping point" between the two devices – the point at which the switches start consuming maximum energy, might be explained by a difference in the implemented T_d value for both switches. A different T_d value would influence the respective switch's power consumption with the same traffic pattern.

Also, when observing the pattern of increase in the power consumption, we see a linear increase instead of the logarithmic increase expected. Although we believe this is related to the specific vendor implementation, we were unable to verify this, and believe this should be examined further as part of future research.

3.5 Experiment 5: 802.3az Energy Consumption vs. Inter-frame delay

This experiment measures the energy consumption of an 802.3az-enabled switch and relates it to the inter-frame delay of a stream of consecutive frames. The purpose of this experiment is twofold. As stated in Section 2.3.2, the T_d parameter is not formally defined in the protocol specification; rather, it is vendor- and implementation-specific. The first goal of this experiment is to determine whether it is possible to experimentally retrieve timing information using our test setups. The second goal is to try and find explanation for the lack of decrease in energy consumption when using UDP traffic and 802.3az-enabled switches.

For this experiment we used the same testbed as in Experiment 1 (see Section 3.2).

3.5.1 Method

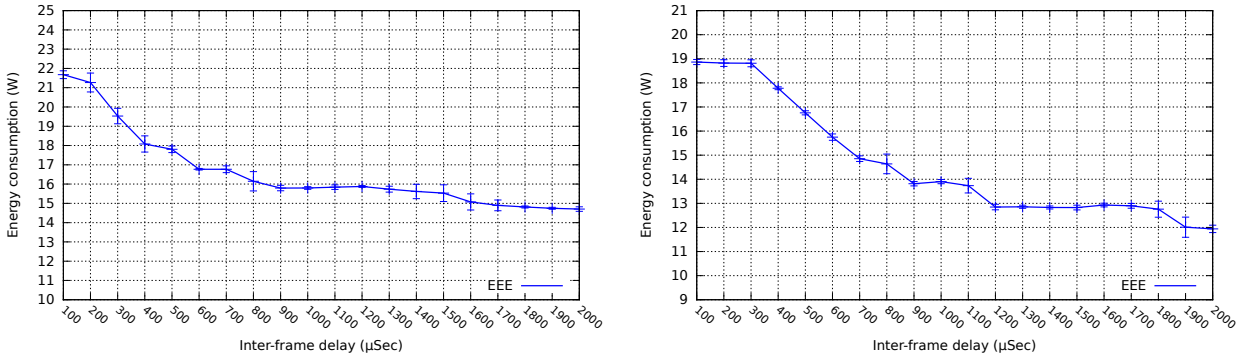
This experiment uses the `mz` packet generation tool [28] to send empty frames (consisting of only the frame header) at specific time intervals. The experiment was performed multiple times, where frames were sent with different inter-frame delays. Each test was run for one minute and energy consumption was averaged over that period.

3.5.2 Results & Analysis

The results of this experiment are given in Figure 11. The plots show the average energy consumption of the switch per delay interval, and, thus, the standard deviation is given as well.

The results of this experiment show a clear relation between the inter-frame delay and the switch energy consumption at 1Gbps linkspeed (see Figures 11(b) and 11(a)). We can note a difference between the Huawei and the Cisco switches' results in the point where they start saving energy – the Huawei device starts saving energy at smaller inter-frame delay values than the Cisco device. We can relate this to the fact that the Huawei switch saves energy until 450Mbps of throughput is processed, while the Cisco switch saves energy only until 400Mbps (see Figure 9).

When using a 100Mbps linkspeed, the power consumption of the switches does not exceed the baseline consumption. This can be explained with `mz` not being capable of generating frames fast enough to get the interfaces out of idle mode most of the time. The result for this linkspeed using the Cisco switch are shown in Appendix D, Figure 20.



(a) Huawei S1728GWR-4P (b) Cisco SG-300-28

Figure 11: Energy consumption vs. frame pattern (1000BASE-T)

3.5.3 Discussion

Although a clear relation between the frame pattern and the energy consumption can be derived from these testresults, the reported timing values do not seem to match the values defined in the 802.3az standard. To be able to save power in sleep mode, the interface has to be idle for a period equal to at least $min(\mathrm{T}_s) + min(\mathrm{T}_q) + min(\mathrm{T}_w)$ (see Figure 1), which equals 20230μ for 1000BASE-T ($200\mu + 20\ 000\mu + 30\mu$) (see Tabel 1). This does not explain our test results, which show a decrease in power consumption for an inter-frame delay of higher than 300μ. One reason for this could be that generic hardware does not allow for precise control of timing at the microsecond level. Another reason would be vendors implementing 802.3az in a manner deviating from the standard. However, both of these are out of the scope of this research and were not looked into further.

3.6 Discussion on Experiments

The first experiment using Green Ethernet (see Section 3.1) shows that the features of this technique can decrease the energy consumption of switches. However, since energy savings are not related to throughput. Thus, these techniques offer limited capabilities for an application to optimize its energy usage.

The results of the subsequent 802.3az experiments generally confirm the theoretical analysis of the IEEE 802.3az protocol, performed by previous research [8][14]. The most significant difference we observed is the fact that the amount of consumed power increases linearly with the increase in transmission speed, while previous research suggests that this amount should increase logarithmically. Furthermore, instead of increasing steadily until it reaches the maximum transmission speed, we observed that power usage increases until it hits a tipping point, after which the consumption stabilizes (see Figure 9). We believe that this is related to the protocol's state transition timers, which prevent the networking controller from sleeping after a certain boundary. As these timers are also dependent on the specific vendor implementation, we could not look further into this matter.

By examining switches from different vendors, we came to the conclusion that different implementations of the protocol behave differently. However, the differences are quantitative, rather than qualitative – the general behavior of the different examined implementations is similar, but the amount of energy saved, and the threshold of effectiveness of the protocol differ. This can clearly be seen in Figures 9(a) and 9(b).

Some of our experiments yielded unexpected results. Firstly, the energy savings achieved by enabling 802.3az on 100BASE-TX links seems marginal – about 9% decrease in both the Cisco and the Huawei switch. The experimental results from these tests are given in Appendix D, Figures 14, 15 and 16.

Also, the Extreme switch we had did not save any power with 802.3az enabled. We believe this is due to the fact that the switch's hardware design does not take the 802.3az operation into account. However, although this switch uses an 802.3az-capable Ethernet controller, the device itself is not marketed as 802.3az-capable.

Further, when using UDP traffic, we could not establish a relationship between throughput and device energy consumption. This could be explained by the nature of the UDP protocol. In comparison, the TCP protocol has to wait for acknowledgements of the packets sent, while UDP does not, since it is a stateless protocol, and transmits as fast as the rate limiting allows it. While this could lead to less time for the interface going into sleep mode, we believe that this behavior is odd and should be examined further by future research.

Finally, when trying to experimentally gain knowledge about the 802.3az sleep timers by varying the inter-frame delay (see Section 3.5), our experiments seemed to yield unreliable results. Although the results (see Figure 11) show a decrease in energy consumption when increasing the inter-frame delay, the relation is neither linear (as shown in Section 3.4), nor exponential (as simulated in previous research [17]). Furthermore, the timings do not correspond to the timing values as specified in the 802.3az standard (see Table 1). We therefore assume that the reported timing values do not match the exact timing values as seen on the wire and specialized hardware is needed to achieve better accuracy.

The results of our experiments so far showed that the 802.3az protocol may have a potential application within an HPC environment. In the next chapter we discuss the practical applicability of the protocol in such environments.

4 Applying 802.3az to Cluster Computing

Having analyzed the capabilities of IEEE 802.3az protocol, and having examined closely the operation of two switches in the context of the protocol, we constructed a methodology for estimating the power usage of 802.3az-enabled switches. Our methodology allows one to explicitly estimate the energy consumption in the following cases:

- The switch or switches power usage estimation at a fixed throughput as a function of the active switch ports,
- The switch or switches power usage for the transfer of a fixed amount of data with a fixed number of active switch ports as a function of the throughput,
- The time distribution for a task as a function of the active compute nodes.

Combining these items, two different estimates can be calculated. First, one can estimate the total power usage of the switch as a function of the compute nodes and the time needed for a task's completion, and second the total power consumption of the switch, if the amount of data is fixed, as a function of the switch throughput.

We created a prototype implementation of our methodology using BASH [29] shell scripting. This prototype calculates the estimates described above based on several input parameters.

This chapter describes the details of our methodology, provides details about our prototype implementation and gives a discussion on the results.

4.1 Time Distribution Estimation

To be able to estimate the time distribution for a task as a function of the active compute nodes, we examine the way the time for completion of a task decreases as the number of compute nodes working simultaneously on that task increases. This estimation is needed in the cases when one desires to find the optimal number of compute nodes for a task. For the calculation we use Formula (2).

$$T_t = \frac{T_s}{n} + (n-1) * C \tag{2}$$

Here, the total needed to complete the task on *one node* is denoted as T_s. Then, the *total time* (T_t) needed to complete the task is equal to T_s divided by the number of compute nodes (n). (T_t) is simply divided by the number of nodes because each node entails a certain communication overhead. This overhead is denoted by C which is the time spent coordinating and communicating with other nodes. The increase of the communication overhead is linearly increasing related to the number of nodes, as we assume each node to be communicating with only one other node at a time for the same task.

4.2 Power Usage Estimation

To be able to estimate the switch/switches energy consumption, this methodology requires a knowledge base of energy profiles for different switches. We calculate the energy usage of a single port based on the available energy profile for a particular switch at a particular transmission speed (throughput).

Then, the estimation of the power usage of the whole switch is obtained by considering the baseline switch energy usage and adding the power required for the number of active switch ports. This method allows us to extrapolate the energy consumption of multiple switches of the same or different types by using data obtained from a single device.

For the estimation of the energy required for a single switch port we use Formula (3), where the *estimated power per port* is denoted as P_{pp} and the *average power for a specific throughput level*, the *average baseline power*, and the *number of ports* used for the creation of the profile – as P_{tp}, P_b, and N_p respectively.

$$P_{pp} = \frac{P_{tp} - P_b}{N_p} \tag{3}$$

For calculating the total instantaneous switch energy usage as a function of the used switch ports, we use Formula (4), where the *total energy usage* of the switch and the *number of used switch ports* are denoted as P_{total} and N_u respectively.

$$P_{total} = P_{pp} * N_u \tag{4}$$

Using the calculated instantaneous switch energy usage, we provide this at input for Formula (5), which estimates the total energy usage for a given task (P_{task}). For the *time*, denoted as T, one may use either the calculations for the time distribution of a task (above), or the total transfer time for a given amount of data (below).

$$P_{task} = P_{total} * T \tag{5}$$

For estimating the total energy usage of the switch for transferring a given amount of data, we first calculate the time needed to transfer the data at different transmission speeds (throughput). The calculation is performed by using Formula 6, where the *total transfer time* is denoted as T_{tr}, and the *amount of data* (in MiB) and the *transmission speed* (in Mbps) are denoted as A_d and S_t respectively.

$$T_{tr} = (A_d * 8)/S_t \tag{6}$$

Then, the obtained value for T_{tr} is supplied together with P_{total} as input for Formula (5), which estimates the total energy consumption of the switch to transfer the provided amount of data between the given number of nodes.

4.3 Prototype Implementation & Results

With the gained knowledge and by utilizing the data gathered in the previous research phases, we implemented our estimation methodology and created a prototype *Power Budget Calculator*. The prototype implementation consists of a set of BASH scripts, which estimate the power usage of switches by analyzing their existing power profiles. The created scripts, along with the constructed power profiles in the correct format, are provided in Appendix B. They can also be found on https://github.com/zupper/cluster-efficiency.

We implemented the task-based power usage estimation, where a task's power budget depends on the time needed for the task, separately from the fixed-time estimation of bandwidth. These two implementations operate differently, and as a result, take different parameters as input.

The task-based estimation tool operates in the context of parallel computing, and its output is a recommendation about how to parallelize a certain task, so that power is used optimally.

The power calculator scripts take the following parameters as input:

- *Nodes* – the number of available nodes that *could* be used for this task,
- *Switch* – which switch's profile to use for energy consumption values,
- *Ports per switch* – usable ports per switch,
- *Total time* – the total time in minutes needed to complete the task on one node,
- *Communication overhead* – communication overhead in minutes for adding a node,
- *Speeds* – the list of network transmission speeds to use,
- *Node Power* – the average power a node uses while computing.

Regarding the power consumption of the compute nodes, rather than relying on a model, we use an average power consumption value as input and perform calculations with that. We did not delve deeper into simulating the power profile of compute nodes, as this is out of the scope of the project; however, putting their power usage into the perspective of the power usage of the networking equipment helps show the differences in scale.

By analyzing the input parameters and performing time distribution estimation, we estimate the total energy usage of the switch for a particular task as a function of the number of used ports on a switch. We also take into consideration the power usage of compute nodes and integrate that into the final script output.

The bandwidth estimation tool, on the other hand, estimates the power usage of switches for transferring a fixed amount of data between nodes.

The bandwidth estimation tool takes the following parameters as input:

- *Nodes* – the fixed number of nodes that *will* be used for this task,

- *Data* – the fixed amount of data that needs to be transferred during the execution of this task,
- *Switch* – which switch's profile to use for energy consumption values,
- *Ports per switch* – usable ports per switch,
- *Total time* – the total maximum time in minutes available for the execution of this task.

By analyzing the existing power profile of the switch, this tool estimates the power usage of switches at different transmission rates and, thus, finds the optimal transmission rate for the achieving of the lowest power consumption.

The output of the calculator scripts consists of two parts – a visualization of the power usage as a function of the number of nodes, and a visualization of the power usage of switches, based on the transmission rate they use to transfer a fixed amount of data.

In our testing of the prototype, we used various parameter values for both parts of the calculator – the task-based and the fixed datasize-based power usage estimation tools.

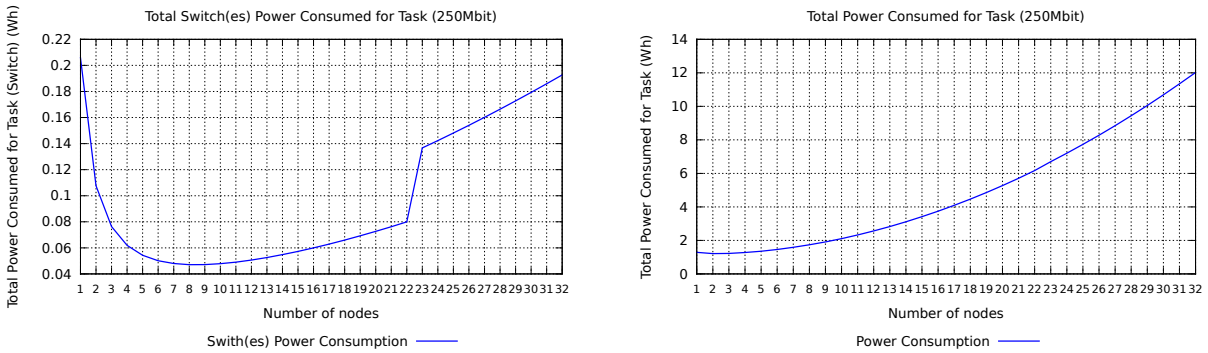
Figures 12(a) and 12(b) show example output of the task-based estimation tool when using the following parameter values:

- *Nodes* – 32
- *Switch* – Huawei
- *Ports per switch* – 23
- *Total time* – 1
- *Communication overhead* – 0.05
- *Speeds* – 250
- *Node Power* – 65

As can be seen in the figures, the power usage of the switches follows a hyperbolic curve, as the energy consumption of the devices is heavily dependent on the time needed for the execution of a task. Since we are using the well-known model for estimating time distribution for parallel tasks, the time needed as a function of the number of compute nodes follows a hyperbolic curve, which in turn influences the shape of the curve of the power consumption function of the switch. Note the jump in power consumption, which corresponds to adding a second switch to handle the needed number of nodes. The total power consumption (shown in Figure 12(b)) follows a similar shape. However, since this estimation takes into account also the compute nodes' power consumption, the energy consumption grows quite fast in comparison to the consumption of the switches.

Since the input parameter are the only influencing factors when using the prototype calculator, we can judge from these results, that using 8 compute nodes would be best for this concrete task, if one

is concerned with optimizing energy usage of the networking infrastructure alone. However, if one is to also consider the power usage of the compute nodes along with the switch's power consumption, then the results suggest that using 2 to 4 nodes would be optimal for this specific task. Although, in this specific example, we show results, based on a transmission rate of 250Mbps, one may decide to estimate for other transmission rates, if that is deemed necessary for the task at hand.

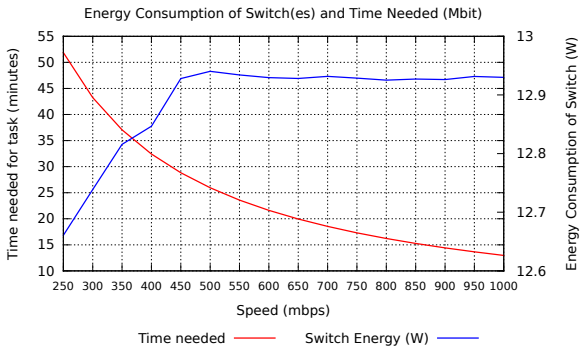


(a) Total Switch(es) Power Consumed for Task (250Mbit) (b) Total Power Consumed for Task (250Mbit)

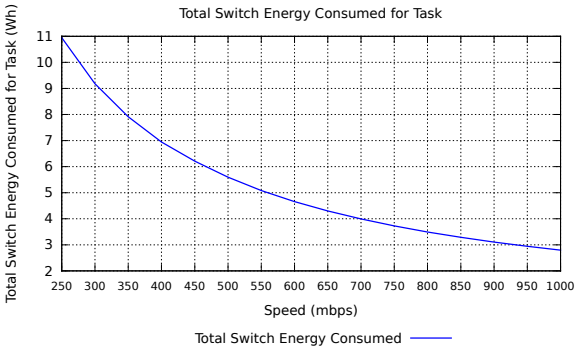
Figure 12: Power usage estimation using 32 nodes and a Huawei S1728GWR-4P

The output of the bandwidth estimation tool is given in Figures 13(a) and 13(b). This result was achieved by using the following input values:

- *Nodes* – 2,
- *Data* – 97280,
- *Switch* – Huawei,
- *Ports per switch* – 23,
- *Total time* – 55.



(a) Power distribution vs. time needed for transfer



(b) Total power consumption as a function of the transmission rate

Figure 13: Power usage estimation using 2 nodes, 97GiB of data and a Huawei S1728GWR-4P

As can be expected, with faster transmission rates, the switch need less time to transfer the fixed amount of data. Since the time needed for transferring the data decreases in a hyperbolic manner as the transmission rate increases, while the power consumption increases linearly and then stabilizes, we observe an overall hyperbolic decrease in the consumption of power as the transmission rate increases. This means that it will always be more energy-efficient to transmit at faster rates when the task time is flexible and dependent on the time needed for the data transfer. Note how the plot does not include speeds lower than 250 Mbps, as at such low speeds the transfer of the given data for the maximum amount of time would not be possible.

4.4 Discussion

Although we have implemented our methodology in the form of shell scripts, it can easily be re-implemented using a more advanced programming language and possibly be integrated into the workings of task scheduling applications, so that these can work in a more power-efficient manner. The produced results can also be used as a reference model for power consumption; however, certain limitations of the methodology and of the created scripts need to be taken into consideration.

Firstly, regarding the estimation of the power usage of the switch/switches, for every run of the script, we assume that only one task will be running on the nodes connected to the switch(es). Thus, we always add the full baseline power usage of the switch(es) to the estimation. While it could be that there is only one task running within a cluster, in most practical cases, nodes connected to a single switch handle multiple tasks separately from one another. Furthermore, regarding the switch make and the network architecture, the calculator tool makes the assumption that the cluster network uses multiple switches of the same type. Also, the architecture is assumed to be a simple one and in the cases where multiple switches are needed, the calculator simply multiplies the baseline power consumption value by the number of switches needed to allow the communication between the nodes. As both assumptions may be false, the accuracy of the estimation may be inaccurate; however, further improvement on this aspect of the estimation tool is out of the scope of this research.

Further, regarding the time distribution model, this model is a simplification of an actual model that truly represents the way time decreases when compute nodes get added. The differences between the current model and the actual model are mostly in the variability of the communication overhead, where we take the overhead C to be constant, while it actually changes and is relative to the number of nodes. However, the difference within a commodity cluster is negligible in comparison to the time needed for the completion of a task.

Lastly, regarding the amount of tasks a compute node can work on simultaneously, we assume that a single node, connected to a single switch port, is busy with only one task at any point in time. Considering the fact that it is likely that modern clusters are comprised of compute nodes with multi-core CPUs, this assumption will likely be false in a practical setting. However, the incorporation of this factor into the calculating script is out of the scope of this project, as it was not our intention to fully implement the estimation tool.

5 Research Discussion

The previous chapters of this paper provided a general overview, as well as details about the different phases of this project. In this section we discuss the project as a whole, its achievements and its shortcomings.

5.1 Achievements and Weak Points

In this research we accomplished several goals – we explored the 802.3az protocol from a theoretical point of view, we explored several actual implementations of the protocol and created power profiles of the implementing devices. Furthermore, we used the obtained knowledge and data to construct an energy model for estimating the power consumption of switches by extrapolating on existing power profiles, and creating predictions about the optimal conditions under which a task should run. However, several shortcomings of this project need to be mentioned.

While other research was performed from an analytical point of view – by simulating the behavior of 802.3az [8][14], we undertook a more practical approach, attempting to measure the reduction in power consumption purely experimentally. However, we discovered that the generally available measuring equipment is not well tailored to the precision, granularity and high resolution of measurement that is required when working with 802.3az. Firstly, the switches we had access to consumed very little power, while the measuring devices were targeted at high-end datacenter environments, where the amount of consumed power is enormous in comparison. Thus, the sensitivity threshold for measuring was at times above the consumption of our devices, which made measuring difficult. Secondly, the 802.3az protocol operates on the microsecond level, while our measuring devices had a measurement resolution of once per second. As a result of these considerations, we modified our approach from a purely experimental one, into a combination of experimentation, measurement and theoretical extrapolation of the measurements.

Furthermore, we did not have access to specialized measuring or traffic generating hardware, which could operate on the microsecond level. As a result of this, we needed to perform extended test runs and rely on averaged data from our experiments. We could not verify some of our findings practically – something that would have been possible with specialized measuring or network capture hardware (see Section 3.4).

Lastly, due to the time frame of this project, we were unable to further investigate certain odd results from our experiments (see Sections 3.2.3 and 3.4.3). We do recommend that these be investigated as part of future research.

5.2 Maturity of 802.3az and Vendor Adoption

Regarding the maturity of the protocol, we determined that adoption of this protocol is still in its infancy, considering the fact that only a few switches and Network Interface Card (NIC)s fully support it. As a result of this, it was difficult to initially obtain the proper hardware for us to run our

experiments. We believe that this factor will become less significant in the future, as more vendors and devices support the protocol.

Also, regarding the vendor adoptance of the protocol, we take into consideration the fact that the implementations we tested are still in their first generation. This was most visible with the Extreme device we tested, which did not produce any energy savings, although the networking controller on the device supported the protocol. We believe, that, as with other technology, these implementations will improve in the future, as we presume that energy savings with 802.3az are not purely related to the networking controller, but to the overall hardware design of the implementing devices.

5.3 802.3az and Other Green IT Efforts

Lastly, we would like to mention that within a cluster environment, energy-efficient networking is only a contributing factor to green computing overall. Much effort is put into researching and deploying other aspects of green computing in various environments, as the issues of energy consumption become increasingly visible in the information technology field [5][30][4]. Only by employing various research and industry findings from the green computing filed of research can one achieve significant power savings within an HPC environment.

6 Conclusion

This project aimed to assess the applicability of the IEEE 802.3az EEE standard and other energy-efficient Ethernet techniques to High Performance Computing environments. Apart from analyzing the operation of the protocol and other techniques, we aimed at proposing a way to optimize the networking-related energy usage of applications within such environments.

After studying the 802.3az protocol both from a theoretical and from an experimental perspective, we determined that this protocol has a real potential to optimize the energy efficiency of networked environments. It is capable of achieving energy savings with a fully operational Ethernet link by putting the link into low-power mode when no traffic is expected on the link. Thus, its operation and effectiveness is possible within an always-on computing environment. Furthermore, we believe the technology, can contribute to other green IT efforts when applied within a distributed high performance computing environment.

After analyzing the operation of 802.3az, we determined that saving energy with this protocol is possible with the vast majority of traffic patterns. This is related to the fact that the protocol operates on the microsecond level, and, as such, saves energy with even small gaps in data transmission. Since traffic in HPC environments is associated with bursts when transmitting data needed for computation, and quiet periods, when computations are carried out, the 802.3az protocol is applicable in such environments even when no further changes are made to the environment. Related to this, we determined that achieving further energy savings with 802.3az is possible by performing low-level software changes in the firmware and device drivers of networking equipment.

With regards to optimizing the networking-related energy usage of applications within HPC environments, we created an energy model, which takes the 802.3az operation into account, and estimates the execution conditions under which specific applications will use the least amount of power. The proposed energy model can be implemented in any programming language, and can be incorporated into the operation of cluster or grid scheduling and workflow systems.

7 Future Research

Using the testbed and methods described in chapter 3 none or only minimal energy savings could be measured when using UDP as transport layer protocol. The lack of energy savings when using UDP seems counter-intuitive, as the 802.3az protocol operates on layers 1 and 2 of the well-known OSI stack, and is not layer 4 aware. While UDP operation differs from TCP in being stateless and not requiring datagram acknowledgements, the interface should be able to go into energy saving mode when, for example, a transmission rate of 50Mbps is used on a 1Gbps link. Also, when using 100Mbps linkspeed, the expected relationship between throughput and energy consumption should be similar to links using 1Gbps linkspeed. However, this could not be confirmed in our test setup. Both issues require future research to explain and understand. We believe that using a hardware based loadgenerator, which gives very accurate control over low level timing parameters, is the key to explaining this behavior. This presumption is strengthened by the results of our last experiment (see Section 3.5), which confirms that generic hardware and software do not give accurate control over low level timing parameters. One of the future directions of research will be investigating the energy behavior of 802.3az-enabled devices when using transport protocols different from TCP and UDP.

Regarding the constructed energy model, although we believe it gives an accurate estimation of power consumption in the cases that we take into consideration, it is limited in its scope. As a starting point, we focused on application in parallel computing environments. However, we believe that future research can build on our work and extend the model to encompass also other distributed computing environments. Also, in modern HPC environments it is common to use multi-core processor architectures, which can work on several tasks simultaneously, while using a single networking interface. As our model makes the assumption that a single node (and a single switch port) works on a single task, multi-core architectures need to be incorporated into the model. Further, as we did not have access to specialized traffic capture or measurement hardware, we could not verify our model's applicability to clusters by analyzing the low level traffic patterns of an actual cluster. Lastly, to be able to achieve truly optimal energy efficiency, one needs to take into account the energy efficiency of compute nodes and their performance. Extending the model in one or several of the mentioned directions can possibly be done along with research into other of the aforementioned areas (UDP, 100BASE-TX) when access to specialized hardware is available.

References

[1] Stephen L. Herman. *Delmar's Standard Textbook of Electricity*. 1999.

[2] Kenneth G. Brill. *Data Center Energy Efficiency and Productivity*. 2007.

[3] Daniel A. Lashof and Dilip R. Ahuja. *Relative contributions of greenhouse gas emissions to global warming*. 1990.

[4] D. Wang. *Meeting Green Computing Challenges* . 2008.

[5] Dennis Mocigemba. *Sustainable Computing*. 2006.

[6] V. Zyuban and P. Kogge. *Optimization of high-performance superscalar architectures for energy efficiency*. 2000.

[7] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis. *Energy-Efficient Cloud Computing*. 2009.

[8] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J.A. Maestro. *IEEE 802.3az: The Road to Energy Efficient Ethernet*. 2010.

[9] Prachi Patel-Predd. *Energy-Efficient Ethernet*. 2008. http://spectrum.ieee.org/computing/networks/energyefficient-ethernet.

[10] Green Ethernet Website, 2009. http://web.archive.org/web/20090618035001/http://greenethernet.com/.

[11] LAN/MAN Standards Committee of the IEEE Computer Society. *IEEE Std 802.3az-2010 (Amendment to IEEE Std 802.3-2008)*. 2010.

[12] Cisco / Intel. *IEEE 802.3az Energy Efficient Ethernet: Build Greener Networks*. 2011.

[13] Mehrgan Mostowfi. *Improving the Energy Efficiency of IEEE 802.3az EEE and Periodically Paused Switched Ethernet*. 2010. http://scholarcommons.usf.edu/etd/3623.

[14] P. Reviriego, J. Hernadez, D. Larrabeiti, and J. Maestro. *Burst Transmission in Energy Efficient Ethernet*. 2010.

[15] Maarten van Steen and Andrew Tannenbaum. *Distributed Systems: Principles and Paradigms*. 2007.

[16] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. *Workflows and e-Science: An overview of workflow system features and capabilities*. 2008.

[17] P. Reviriego, J.A. Hernandez, D. Larrabeiti, and J.A. Maestro. *Performance Evaluation of Energy Efficient Ethernet*. 2009.

[18] Specifications of Cisco SG300-28 Switch. http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps10898/brochure_c02-610054.pdf.

[19] Specifications of Schleifenbauer PDU. http://www.schleifenbauer.eu/nl/in-rack-pdu_technische-specificaties.htm.

[20] Specifications of Intel I350-T2 Nic. http://intelethernet-dell.com/wp-content/uploads/2012/04/Dell_Intel-Ethernet-I350-T2_T4-PB_adkit.pdf.

[21] Specifications of Racktivity ES6024-16 PDU. http://www.racktivity.com/media/pdf/Racktivity_EnergySwitch_ES2000-ES6000_Series-Datasheet.pdf.

[22] Specifications of Extreme X440-24p Switch. http://www.extremenetworks.com/libraries/products/DSSumX440_1831.pdf.

[23] Specifications of Huawei S1728GWR-4P Switch. http://market.huawei.com/hwgg/enterprise/u-channel/pdf/S1700.pdf.

[24] Iperf Website. http://sourceforge.net/projects/iperf/.

[25] TC man page. http://linux.die.net/man/8/tc.

[26] BWCTL Website. http://www.internet2.edu/performance/bwctl/.

[27] Netcat Website. http://netcat.sourceforge.net/.

[28] Mausezahn Website. http://www.perihel.at/sec/mz/.

[29] GNU Bash Website. http://www.gnu.org/software/bash/.

[30] C. Qingwen, P. Grosso, K. van der Veldt, C. de Laat, and R. Hofman. *Profiling Energy Consumption of VMs for Green Cloud Computing*. 2011.

Appendices

A Acronyms

API Application Programming Interface

BASH Bourne Again Shell

CPU Central Processing Unit

EEE Energy Efficient Ethernet

HPC High Performance Computing

ICMP Internet Control Message Protocol

IEEE Institute of Electrical and Electronics Engineers

LLDP Link Layer Discovery Protocol

LPI Low Power Idle

MPI Message Passing Interface

NIC Network Interface Card

PCB Printed Circuit Board

PDU Power Distribution Unit

PHY Physical Layer of the OSI model

RAM Random Access Memory

SNMP Simple Network Management Protocol

TCP Transmission Control Protocol

UDP User Datagram Protocol

B Power Usage Estimation Calculator Code

```
#!/bin/bash

# This script estimates the total energy usage of EEE-enabled switches for
# the performing of a specific task in a cluster environment. The output of
# this script is intended to aid the decision-making process of a workflow
# scheduler in a cluster environment.
#
# The total energy is dependend on multiple parameters (explained below)

# ========== START OF USER-SUPPLIED DATA ==========

# The number of available nodes
# * User-supplied
NODES=16

# The average pwoer consumption of a node when busy (in watts)
# * User-supplied
NODEPOWER=65

# Which profile to use for energy consumption values
# * User-supplied
SWITCH="huawei"

# Usable ports per switch
# * User-supplied
PORTSPERSWITCH=23

# The total time needed to complete the task on one node (in minutes)
# * User-supplied
TOTALTIME=1

# Communication overhead for adding a node (in minutes)
# * User-supplied
COMMOVERHEAD=0.01

# Which speeds to consider (possible speeds: 50, 100, 150, ..., 1000)
# * User-supplied
SPEEDS="250"

# Filename of the plots file
# * User-supplied
PLOTSFILE="out.pdf"

# ========== END OF USER-SUPPLIED DATA ==========

# ========== START OF KNOWLEDGE BASE DATA ==========
# Baseline file -- containing the power consumption (in W) of the idle switch over
# time
# * Knowledge base
BASELINE="$SWITCH/baseline.gbit.eee.dat"
```

```
# Directory containing the power profile of the switch at different transmission
# speeds
# * Knowledge base
PROFILEDIR=$SWITCH

# The number of ports connected to establish the baseline
# * Knowledge base
BASELINEPORTS=22

# ========================= END OF KNOWLEDGE BASE DATA =========================

RESULTSDIR="results"
RECOMMENDFILE="$RESULTSDIR/recommendation"
AVGBASE=`awk '{ sum+=$2 } END { print sum/NR }' < $BASELINE`

(rm -rf $RESULTSDIR && mkdir $RESULTSDIR) || \
    (echo "Could not create results dir!" && exit 1)
touch $RECOMMENDFILE

for SPEED in $SPEEDS; do
  echo Working with ${SPEED}M

  TMPRECOMMEND="/tmp/recommend-$SPEED"
  rm -f $TMPRECOMMEND
  touch $TMPRECOMMEND

  FILE="$PROFILEDIR/iperf-link_util-${SPEED}M"
  AVG=`awk '{sum+=$2} END { print sum/NR}' < $FILE`

  PERPORT=$(echo - | awk "{print ($AVG - $AVGBASE)/$BASELINEPORTS}")

  RESFILE=$RESULTSDIR/power-vs-time-at-${SPEED}M.dat
  echo "Ports Energy Time TimePower" > $RESFILE

  nodescount=1
  # Use the variables below to estimate for 2^x number of nodes
  #counter=0
  #nodescount=$((2**counter))
  while [ $nodescount -le $NODES ]; do
    # How many switches are needed to get $counter usable ports?
    SWCOUNT=$((nodescount/PORTSPERSWITCH + 1))

    # We estimate what the switch power usage will be for $counter ports
    # at this $SPEED (in W)
    POWER=$(echo - | awk "BEGIN {print $nodescount * \
                                $PERPORT + $AVGBASE * $SWCOUNT}")
    TIMETASK=$(echo - | awk "{print $TOTALTIME / $nodescount + \
                                ($COMMOVERHEAD * ($nodescount - 1))}")

    # Time-power product (gives energy consumed in Wh)
    TPPRODUCT=$(echo - | awk "{print (($TIMETASK / 60) * $POWER)}")

    COMPUTEPOWER=$((nodescount * NODEPOWER))
```

```
    # Time-(switch/server)-power product
    TPCPRODUCT=$(echo - | awk "{print ((\$TIMETASK / 60) * \
                                        (\$POWER + \$COMPUTEPOWER))}")

    echo $nodescount $POWER $TIMETASK $TPPRODUCT $BANDWIDTH \
                                $COMPUTEPOWER $TPCPRODUCT >> $RESFILE
    echo $TPCPRODUCT $nodescount >> $TMPRECOMMEND

    nodescount=$((nodescount + 1))
    # Use the variables below to estimate for 2^x number of nodes
    #counter=$((counter+1))
    #nodescount=$((2**counter))
  done

  sort -n $TMPRECOMMEND > /tmp/tempfile && mv /tmp/tempfile $TMPRECOMMEND

  echo "Speed: ${SPEED}M" >> $RECOMMENDFILE

  MINPOWER=`sed -n 1p $TMPRECOMMEND | sed -e 's/ .*$//'`
  MINNODES=`sed -n 1p $TMPRECOMMEND | sed -e 's/^.* //'`
  echo "     Nodes:" $MINNODES, Power: $MINPOWER >> $RECOMMENDFILE

  RECOMENDLINES=`cat $TMPRECOMMEND | wc -l`

  if [ $RECOMENDLINES -ge 3 ]; then
    counter=2
    while [ $counter -le 3 ]; do
      THISPOWER=`sed -n ${counter}p $TMPRECOMMEND | \
                                        sed -e 's/ .*$//'`
      THISNODES=`sed -n ${counter}p $TMPRECOMMEND | \
                                        sed -e 's/^.* //'`

      if [ $THISNODES -gt $MINNODES ]; then
        echo "     Nodes:" $THISNODES, \
                        Power: $THISPOWER >> $RECOMMENDFILE
        counter=$((counter + 1))
      fi
    done
  fi

  # cleanup
  rm -f $TMPRECOMMEND /tmp/tempfile
done

# ================ CALCULATION FINISHED. PLOTTING FOLLOWS ================

PLOTCMDS="reset;\
set output '${PLOTSFILE}';\
set terminal pdf;\
set grid;\
set xlabel 'Number of nodes';\
set ylabel 'Time needed for task (minutes)';\
set style fill solid border rgb 'black';\
set y2tics 2;\
```

```
set xtics 2;\
set xtics font '0,5';\
set auto x;\
set auto y;\
set key outside below center horizontal;\
set style data lines;\
"

for BANDWIDTH in $SPEEDS; do
  DATA="results/power-vs-time-at-${BANDWIDTH}M.dat"
  THISPLOT="set title 'Energy Consumption of Switch(es) and \
                                        Time Distribution (${BANDWIDTH}Mbit)';\
    set y2label 'Energy Consumption of Switch (W)';\
    plot '${DATA}' using 3 title 'Time' linewidth 3 axes x1y1, \
    '' using 2 title 'Switches Power (Watt)' linewidth 3 \
                                                linecolor 3 axes x1y2;\
    set y2tics 100;\
    set title 'Compute Nodes Power Consumed (${BANDWIDTH}Mbit)';\
    set y2label 'Energy Consumption of Compute Nodes (W)';\
    plot '${DATA}' using 3 title 'Time' linewidth 3 axes x1y1, \
    '' using 5 title 'Compute Nodes Power (Watt)' linewidth 3 \
                                                linecolor 3 axes x1y2;\
    \
    set grid;\
    set title 'Total Switch(es) Power Consumed for \
                                               Task (${BANDWIDTH}Mbit)';\
    set ylabel 'Total Power Consumed for Task (Switch) (Wh)';\
    set auto y;\
    set y2label '';\
    set y2tics 0;\
    plot '${DATA}' using 4:xtic(1) title 'Swith(es) Power Consumption' \
                                        linestyle 3 linewidth 3 axes x1y1;\
    set title 'Total Power Consumed for Task (${BANDWIDTH}Mbit)';\
    set ylabel 'Total Power Consumed for Task (Wh)';\
    plot '${DATA}' using 6:xtic(1) title 'Power Consumption' \
                                        linestyle 3 linewidth 3 axes x1y1;\
"
  PLOTCMDS=$PLOTCMDS$THISPLOT
done

# DEBUG
#echo $PLOTCMDS
(gnuplot -e "$PLOTCMDS" && echo Created plots) || \
  (echo Failed to create plots && exit 1)

echo
echo Recommendations:
cat $RECOMMENDFILE

exit 0
```

C Testing Scripts

Cisco SNMP Polling

```
#!/bin/bash

if [ -z $1 ]; then
  time=0.5
  echo
  echo "No sleep interval specified, using 0.5s"
  echo
else
  time=$1
  echo
  echo "Sleep interval is $1 seconds"
  echo
fi

while [ : ]; do
  TPOWER=`snmpget -v1 -c public  192.168.10.3 \
                  CISCOSB-GREEN-MIB::rlGreenEthCurrentMaxEnergyConsumption.0 | \
                  sed -e 's/^.*Gauge32: //' -e 's/ mWatt//'`
  RPOWER=`snmpget -v1 -c public  192.168.10.3 \
                  CISCOSB-GREEN-MIB::rlGreenEthCurrentEnergyConsumption.0 | \
                  sed -e 's/^.*Gauge32: //' -e 's/ mWatt//'`
  RATIO=$(echo âĂŞ | awk "{print $RPOWER/$TPOWER}")

  echo Actual/Max/Ratio: ${RPOWER} / ${TPOWER} / $RATIO
  sleep $time
done
```

Racktivity SNMP Polling

```
#!/bin/bash

if [ -z $1 ]; then
  time=1
else
  time=$1
fi

while [ : ]; do
  CURRENT=`snmpwalk -v2c -c public -m ES-RACKTIVITY-MIB 192.168.250.250 \
        pCurrent.1 | sed -n 1p | sed -e 's/^.*Gauge32: //' -e 's/ A//'` #huawei
# CURRENT=`snmpwalk -v2c -c public -m ES-RACKTIVITY-MIB 192.168.250.250 |
              pCurrent.1 | sed -n 4p | sed -e 's/^.*Gauge32: //' -e 's/ A//'` #cisco
  VOLTAGE=`snmpget -v2c -c public -m ES-RACKTIVITY-MIB 192.168.250.250 \
        pVoltage.1 | sed -e 's/^.*Gauge32: //' -e 's/ V//'`
  PFACTOR=`snmpwalk -v2c -c public -m ES-RACKTIVITY-MIB 192.168.250.250 \
        pPowerFactor.1 | sed -n 1p | sed -e 's/^.*Gauge32: //' -e 's/ %//'` #hw
```

```
# PFACTOR=`snmpwalk -v2c -c public -m ES-RACKTIVITY-MIB 192.168.250.250 |
                pPowerFactor.1 | sed -n 4p | sed -e 's/^.*Gauge32: //' -e 's/ %//'` #cs

  #DEBUG
  #echo $CURRENT $VOLTAGE $PFACTOR; exit
  RPOWER=$(echo âĂŞ | awk "{print $CURRENT*$VOLTAGE*($PFACTOR/100)}")

  echo `date +%s` $RPOWER
  sleep $time
done
```

Test Runner Iperf TCP

```
#!/bin/bash

if [ -z $1 ]; then
  TESTNAME="NA"
else
  TESTNAME=$1
fi

EEE1PIP="145.100.135.231"
EEE2PIP="145.100.135.248"

E1INT1="eth3"
E1INT2="eth2"
E2INT1="eth3"
E2INT2="eth4"

# Poller script
POLLER="racktivity_poller.sh"
DIR=`pwd`
# MZ script
MZSCRIPT="run-mz.sh"

# Runtime in seconds
RUNTIME="305"

RESULTDIR="test-tcp-`date +%s`-$TESTNAME"
mkdir $RESULTDIR
cd $RESULTDIR

mkdir iperf-tcp && cd iperf-tcp

runIperf() {
  LINKUTIL=$1
  TESTNUM=$2

  POWERFILE="iperf$TESTNUM-link_util-${LINKUTIL}M"

  ssh root@$EEE1PIP "killall iperf"
  ssh root@$EEE2PIP "killall iperf"
```

```
  sleep 5
  ssh root@$EEE1PIP "killall iperf"
  ssh root@$EEE2PIP "killall iperf"

        ssh root@$EEE1PIP "tc.bash stop"
        ssh root@$EEE1PIP "tc.bash start $LINKUTIL"
        ssh root@$EEE2PIP "tc.bash stop"
        ssh root@$EEE2PIP "tc.bash start $LINKUTIL"

  # Start iperf servers
  ssh root@$EEE1PIP "iperf -s -B $E1IP1" &

  # wait until the iperf servers are running
  sleep 2

  # Start iperf clients
  ssh root@$EEE2PIP "iperf -c $E1IP1 -t 728356 -i 5" &

  sleep 2
  sh $DIR/$POLLER > $POWERFILE &
  PID=`pidof sh`

  # Run for $RUNTIME seconds
  sleep $RUNTIME

  # Stop measuring
  kill -9 $PID

  ssh root@$EEE1PIP "killall iperf"
  ssh root@$EEE2PIP "killall iperf"
}

# Test 1 - iperf
# E1INT1 (server) -- E2INT2 (client)
# E1INT2 (client) -- E2INT2 (server)

ssh root@$EEE1PIP "killall iperf"
ssh root@$EEE2PIP "killall iperf"
sleep 5
ssh root@$EEE1PIP "killall iperf"
ssh root@$EEE2PIP "killall iperf"

E1IP1="192.168.141.1"
E2IP1="192.168.141.2"

i=1
for LOAD in 50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 \
   900 950 1000; do

  echo
  echo Running iperf TCP test at $LOAD Mbit/s
  echo
  runIperf ${LOAD} ${i}
  i=$(expr $i + 1)
```

```
done

echo
echo Done with Test
echo
# END Test 1

#exit
```

Test Runner Iperf UDP

```
#!/bin/bash

if [ -z $1 ]; then
  TESTNAME="NA"
else
  TESTNAME=$1
fi

EEE1PIP="145.100.135.231"
EEE2PIP="145.100.135.248"

E1INT1="eth3"
E1INT2="eth2"
E2INT1="eth3"
E2INT2="eth4"

# Poller script
POLLER="racktivity_poller.sh"
DIR=`pwd`
# MZ script
MZSCRIPT="run-mz.sh"

# Runtime in seconds
RUNTIME="305"

RESULTDIR="test-udp-`date +%s`-$TESTNAME"
mkdir $RESULTDIR
cd $RESULTDIR

mkdir iperf-udp && cd iperf-udp

runIperf() {
  LINKUTIL=$1
  TESTNUM=$2

  POWERFILE="iperf$TESTNUM-link_util-${LINKUTIL}M"

  ssh root@$EEE1PIP "killall iperf"
  ssh root@$EEE2PIP "killall iperf"
  sleep 5
  ssh root@$EEE1PIP "killall iperf"
```

```
  ssh root@$EEE2PIP "killall iperf"

        ssh root@$EEE1PIP "tc.bash stop"
        ssh root@$EEE1PIP "tc.bash start $LINKUTIL"
        ssh root@$EEE2PIP "tc.bash stop"
        ssh root@$EEE2PIP "tc.bash start $LINKUTIL"

  # Start iperf servers
  ssh root@$EEE1PIP "iperf -s -u -B $E1IP1" &

  # wait until the iperf servers are running
  sleep 2

  # Start iperf clients
  ssh root@$EEE2PIP "iperf -c $E1IP1 -t 728356 -i 5 -u -b 1125M" &

  sleep 2
  sh $DIR/$POLLER > $POWERFILE &
  PID=`pidof sh`

  # Run for $RUNTIME seconds
  sleep $RUNTIME

  # Stop measuring
  kill -9 $PID

  ssh root@$EEE1PIP "killall iperf"
  ssh root@$EEE2PIP "killall iperf"
}

# Test 1 - iperf
# E1INT1 (server) -- E2INT2 (client)
# E1INT2 (client) -- E2INT2 (server)

ssh root@$EEE1PIP "killall iperf"
ssh root@$EEE2PIP "killall iperf"
sleep 5
ssh root@$EEE1PIP "killall iperf"
ssh root@$EEE2PIP "killall iperf"

E1IP1="192.168.141.1"
E2IP1="192.168.141.2"

i=1
for LOAD in 50 100 150 200 250 300 350 400 450 500 550 600 650 700 750 800 850 \
    900 950 1000; do

  echo
  echo Running iperf UDP test at $LOAD Mbit/s
  echo
  runIperf ${LOAD} ${i}
  i=$(expr $i + 1)
done
```

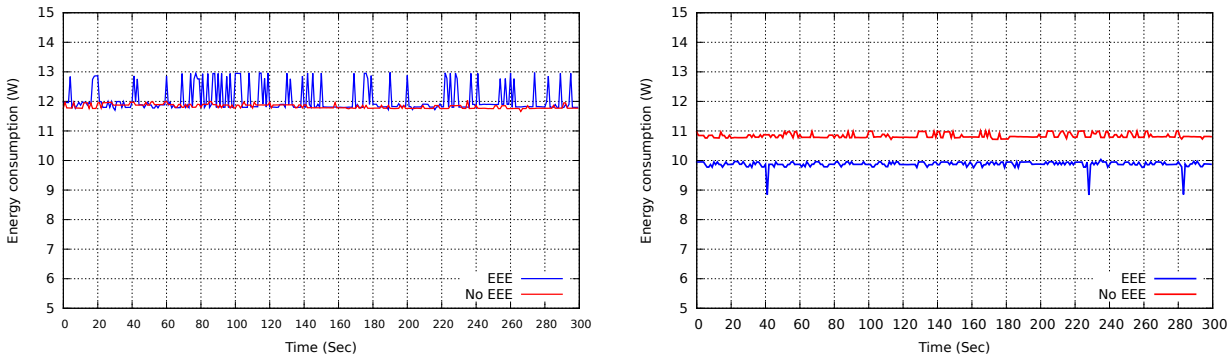
```
echo
echo Done with Test
echo
# END Test 1

#exit
```

Test Runner MZ

```
for i in 1 50 100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 \
          1600 1700 1800 1900 2000 2100 2200 2300 2400 2500 2600 2700 2800 2900 3000;
do
# ssh root@eee1 "mz eth2 -c 10000000 -d $i -b a0:36:9f:09:25:5a" & #cisco
  ssh root@eee1 "mz eth3 -c 10000000 -d $i -b a0:36:9f:09:25:5b" & #huawei
  sleep 2
  sh racktivity_poller.sh > mz-${i}.dat &
  sleep 60
  killall sh
  ssh root@eee1 "killall mz"
done
```

D Additonal Plots

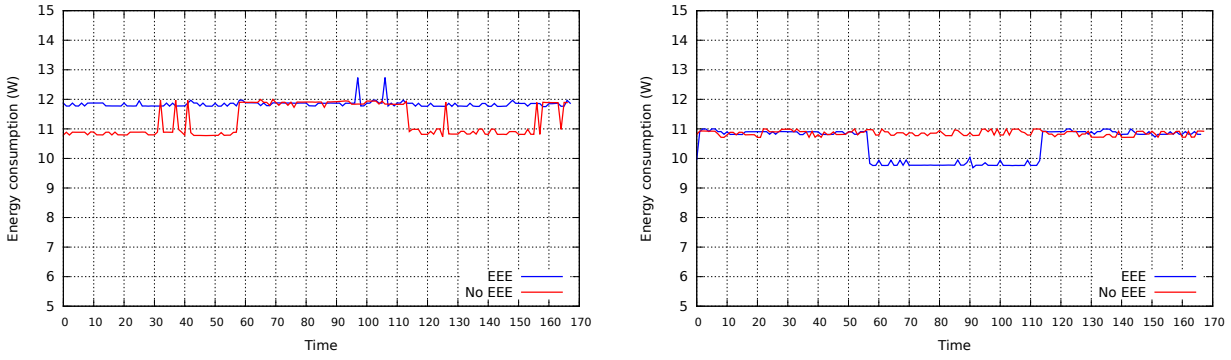
D.1 802.3az Baseline Power Consumption



(a) Huawei S1728GWR-4P (b) Cisco SG-300-28

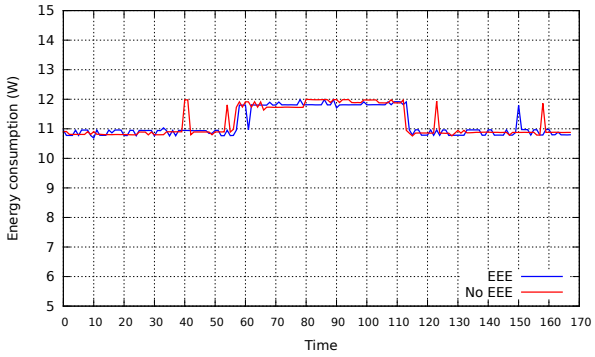
Figure 14: Baseline (100BASE-TX)

D.2 802.3az Maximum Energy Savings

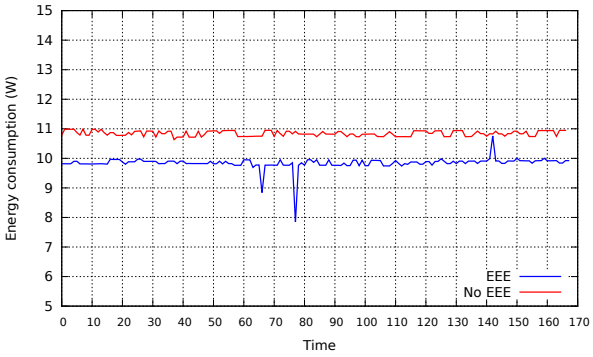


(a) Huawei S1728GWR-4P (b) Cisco SG-300-28

Figure 15: Interval (TCP, 100BASE-TX)

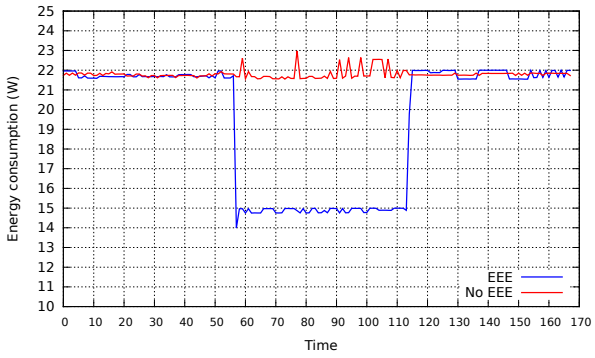


(a) Huawei S1728GWR-4P

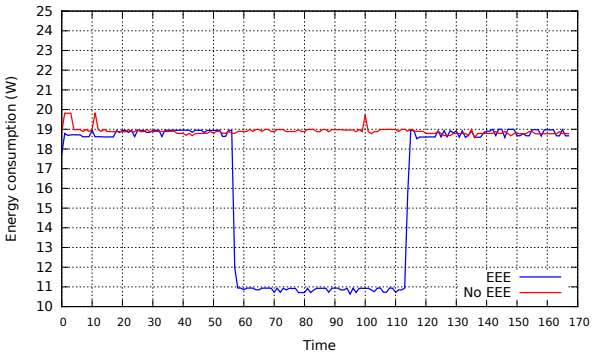


(b) Cisco SG-300-28

Figure 16: Interval (UDP, 100BASE-TX)



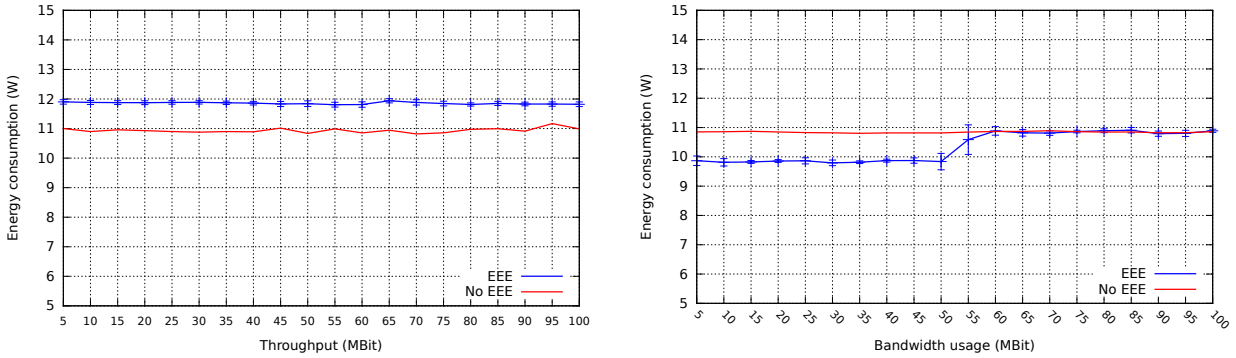
(a) Huawei S1728GWR-4P



(b) Cisco SG-300-28

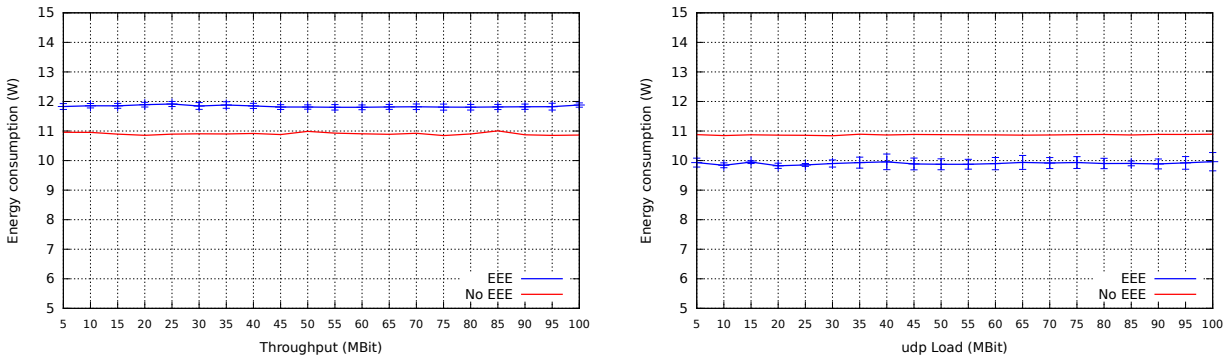
Figure 17: Interval (UDP, 1000BASE-T)

D.3 802.3az Energy Consumption vs. Throughput



(a) Huawei S1728GWR-4P (b) Cisco SG-300-28

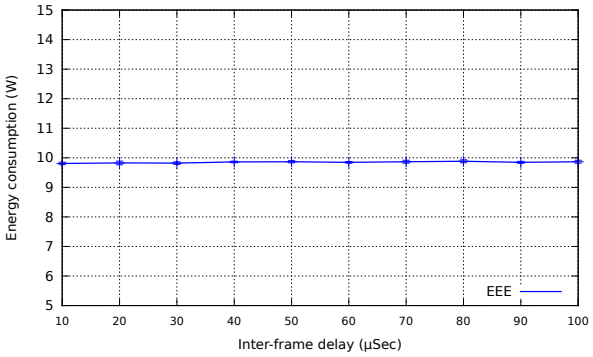
Figure 18: Energy consumption vs. throughput (TCP, 100BASE-TX)



(a) Huawei S1728GWR-4P (b) Cisco SG-300-28

Figure 19: Energy consumption vs. throughput (UDP, 100BASE-TX)

D.4 802.3az Power Consumption as Function of Inter-frame delay



(a) Cisco SG300-28

Figure 20: Power Consumption as Function of Inter-frame Delay (100BASE-TX)