Real-Time Performance Analysis of Adaptive Link Rate

Baoke Zhang, Karthikeyan Sabhanatarajan, Ann Gordon-Ross*, Alan George*
HCS Research Lab, ECE Department, University of Florida
{zhang,sabhanatarajan,ann,george}@hcs.ufl.edu
** Also with the NSF Center For High-Performance Reconfigurable Computing at the University of Florida*

Abstract—**High speed links are widely deployed in modern day computer networks to meet the ever growing needs for increasing data bandwidth. However, with the increase in the link rate, the power consumption of the network interfaces increases exponentially, compounding growing concerns about network power consumption. Fortunately, network traffic characteristics show that rapid link rates are not always required. During times of reduced network traffic, the Adaptive Link Rate (ALR) mechanism allows link rates to be reduced with little impact on network performance. Current research has focused on policies to control when and how to change link rates, and have shown promising energy savings. However, these works have been largely simulative, and have not addressed many of the challenges involved in implementation. In this paper, we develop a hardware prototype ALR system and address real-time challenges involved in realizing such an implementation. We also identify new considerations for control policy development given current technology capabilities as well as future projections.**

Keywords-Adaptive link rate (ALR), local area networks, energy efficient Ethernet, Ethernet, hardware prototyping

I. INTRODUCTION

The aggregate power consumption of computer networks has been increasing at a rapid rate [4] due to the growing number of connected devices such as PCs, switches, and routers. To support the network traffic introduced by these devices, link rates of the connecting infrastructure have also been increasing. We find that as the link rate increases, the power consumption of the links increases exponentially. (Figure 4)

To address this rapid increase in energy use, recent research has focused on reducing power consumption of both network devices and their infrastructure. One potential method for reducing power consumption is through exploiting the characteristics of network traffic, or the link utilization. Research shows that network packets are typically transmitted in bursts [16], with variable length periods of idleness or reduced link utilization between bursts. During these periods, energy is wasted by operating the network interfaces and links at the highest rates available. Devices may adaptively shut down and/or vary link rates in response to link utilization to reduce power consumption [1][4][9]. The ability to vary the link rate is a technique known as adaptive link rate (ALR) [9].

Determining when to change link rates is a challenging problem due to the unpredictability and bursty nature of network traffic. Control policies determine the appropriate time to change link rates and the proper link rate to change to, taking into consideration several factors such as mean packet delay and packet loss. Mean packet delay is the total transit time for a packet, and if increased by too much, could result in human perceivable delay and reduced quality of service. Since the process of switching the link rate can introduce non-negligible delay, control policies should consider this link rate switching time. Furthermore, these topics are currently a focus of the IEEE 802.3az study group [15].

A large switching time is one of the biggest challenges in realizing ALR, and can increase mean packet delay very rapidly. Additionally, significant switching times can lead to buffer overflows and subsequent packet loss. However, in order to achieve fast link switching times, the physical implementation currently faces several challenges at multiple levels. These challenges are device synchronization at the MAC and PHY layers.

Most previous research focused heavily on addressing control policy challenges through simulation models [1][8][9][10]. Due to the absence of a real-time ALR capable system, these efforts did not consider the implementation challenges for MAC and PHY synchronization. In this paper, we develop a hardware prototype ALR system to address these challenges. To the best of our knowledge, we are the first to prototype ALR in an FPGA system. We measure power consumption and link rate switching times using our hardware prototype, and discuss challenges and solutions involved in a direct implementation. Our measurements indicate that the switching time between link rates is on the order of milliseconds, which is at least 70 times larger than that assumed in previous works. Using our real-time measurements of link switching times, power, and energy consumption, we identify new considerations for future control policy development.

II. BACKGROUND AND MOTIVATION

A. Background

Reducing energy consumption of network devices is a major research focus. Gupta et al. [12] proposed a method to reduce energy consumption in Ethernet switches by dynamically shutting down the transceiver completely depending on traffic arrivals, buffer occupancy, and a bounded maximum packet delay. Hays [13] proposed the active/idle

Preamble (7-bytes)	SFD (2-bytes)	Destination MAC Address (6-bytes)	Source MAC Address (6-bytes)	Length/Type (2-bytes) Control (88-08)	Flag (1-byte) (00toff)	Op-Code 02 03 04	Parameters (2-bytes) (New Link Rate)	Reserved (42-bytes) All zeros	Frame Check Sequence (4-bytes)

Figure 1: ALR control Frame

method which transmits data as fast as possible when it arrives and then dynamically switches the transceiver circuits to a low power idle state. ALR, first proposed by Nordman et al. [17], adaptively changes Ethernet link rate to any rate available based on link utilization to reduce energy consumption of the network interfaces of a system.

However, both the ALR and active/idle mechanisms introduce several challenges, such as control policy development, which has been the focus of much research. A transceiver buffer occupancy-based dual threshold policy was first proposed by Gunaratne et al. [8]. In this policy, if the link was operating at a high link rate and the output buffer occupancy fell below a low threshold value, the link rate was decreased. Alternatively, if the link was operating at a low link rate and the output buffer occupancy rose above a high threshold value, the link rate was increased. This method achieved a fast response to traffic variations, but was overly sensitive to minute variations in traffic and caused unnecessary link oscillations, which resulted in increased mean packet delay.

To solve the problem of link oscillation and avoid the resulting increase in mean packet delay, Gunaratne et al. [9] proposed a link utilization based threshold policy. The utilization threshold policy monitors link utilization by counting bytes transmitted during a defined time interval. However, counting these bytes requires additional hardware support such as accumulators and registers, which increases the complexity of ALR. In addition, choosing a representative time interval can be difficult. To reduce the complexity of the link utilization threshold policy, Gunaratne et al. [10] investigated a heuristic policy, called the ALR timeout threshold policy, to hold the link at the high rate for a fixed period of time following a switch from a lower rate.

B. Motivation for Real-Time ALR Hardware

Control policy performance greatly depends on how quickly the link rate can be switched. The lack of real-time ALR systems have led researchers to evaluate control policies through simulation, assuming very fast switching times on the order of 1 ms were achievable with current PHY and IEEE standards. Current research [3][20] estimates that the achievable link rate switching time between 1 Gbps and 100 Mbps link rates ranges from 10 ms to more than 100 ms. This is contrary to the future subset PHY [6] technology proposed by Broadcom which estimates these times in microseconds. In this paper, we measure exact link rate switching times obtained from a real-time hardware prototype to more effectively evaluate ALR based on technology capabilities.

III. LINK RATE SWITCHING MECHANISMS

ALR is a three step process. The first step utilizes the control policy to determine when to switch the link rate. The second step is device synchronization through MAC handshaking. The third step is the physical link rate switching accomplished by configuring the PHY. Since control policy development is beyond the scope of this paper, in this section, we focus on the other two challenges.

A. Device Synchronization

Device synchronization is the mechanism which enables a network device to request a switch in the communicating link rate with another device. To ensure proper synchronization, we begin with an existing two-way MAC layer handshaking protocol [8]. This protocol begins with a request from the initiating device to switch the link rate (ALR request) and ends with a positive acknowledgment (ACK) or negative acknowledgement (NACK) from the recipient device to indicate whether or not the link rate can be switched.

To implement this protocol, we augment the MAC control frames. The IEEE 802.3 standard [14] defines MAC control frames called PAUSE frames [14] to implement link level congestion control. We utilize reserved opcodes in the MAC control opcode field to implement our handshaking mechanism. Figure 1 shows the ALR control frame format, which we adapt from the PAUSE frame. The difference between the ALR frame and the PAUSE frame is the opcode field. The opcode values used for the ALR request, ACK, and NACK frames are 02, 03, and 04, respectively.

To account for situations where two communicating devices do not both support ALR, it is necessary to advertise ALR capabilities during link establishment. This is accomplished using the reserved bits in the unformatted link code word (LCW) pages used in existing auto negotiation [14] (see Section III.B for further details on auto negotiation).

Figure 2 depicts a block diagram of the ALR handshake mechanism. In this example, the two communicating devices are a network interface card (NIC) and a switch, denoted as node A and node B, respectively. Without loss of generality, we assume the control policy is the simple dual buffer threshold policy proposed in [8]. The algorithm for switching from a higher link rate to a lower link rate using the MAC handshake protocol is as follows:

1) During link initialization, ALR capability is advertised during auto negotiation.
2) After auto negotiation completes, the nodes exchange data at the determined link rate.
3) If the control policy at node A determines a link rate switch is appropriate, node A sends an ALR request frame (Figure 2 (1a)) with the desired new (low) rate in the parameter field using the old (high) link rate.

4) Node A starts a timer (Figure 2 (1b)) which expires if an ACK/NACK frame from node B is not received within a specified time. If the timer expires, node A sends the ALR request frame again (Figure 2 (1c)).

5) Once node B receives the ALR request frame, it stops sending data, inspects its buffer occupancy (Figure 2 (2)), and determines whether it can switch to the new (low) link rate

6) If node B accepts the link rate switch, it sends an ACK frame to node A using the old (high) link rate (Figure 2 (3a)). After sending the ACK, node B changes its link rate to the desired value and resynchronizes its clocks (Figure 2 (3b)). If node B rejects the link rate switch, it sends a NACK using the old (high) link rate and remains in the high link rate.

7) After receiving an ACK, node A switches its link rate to the new (low) link rate, resynchronizes its clock (Figure 2 (4)), and starts data transmission at the new (low) link rate (Figure 2 (5)). If node A receives a NACK frame, it will continue to transmit data at the old (high) rate. Node A may re-request a link rate switch in the future.

A similar mechanism can be used to switch from a low link rate to a high link rate. However, the only difference in this case is the recipient devices should always positively acknowledge the request, as a negative acknowledgement may result in buffer over flow and packet loss.

B. PHY Resynchronization

After the need to switch rates has been agreed upon, the PHY must physically switch the link rate. Link switching can be achieved by utilizing the existing auto negotiation feature defined in IEEE 802.3 [14] or by directly configuring the PHY control registers.

Through auto negotiation, the connected PHY transceivers advertise their capabilities and mutually agree to communicate at the highest transmittable link rate. This sequence commences during initial device connection, or when a device is reset or reinitialized. Even though this method is simple and widely supported, auto negotiation is a lengthy process. For example, it requires several seconds for a 1000BASE-T

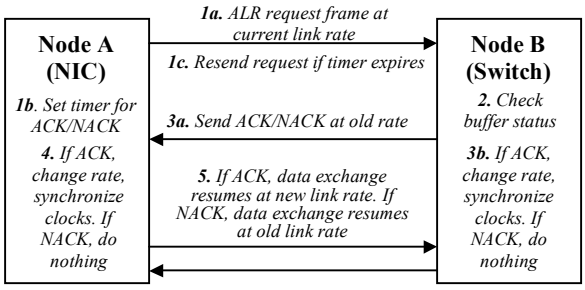

Figure 2: ALR MAC handshake protocol

implementation. For gigabit link rate, a limited sized buffer would fill up in milliseconds, and buffer overflow would be imminent.

To decrease the link switching time, we utilize PHY register configuration, a mechanism to directly configure the PHY control registers allowing a device to change to a desired link speed. Soon after receiving an ACK frame, the MAC sends the appropriate management data input/output (MDIO) command frame to force the PHY transceivers to resynchronize to the desired link rate. Our experimental findings in section VI.A show that this method is orders of magnitude faster than the auto negotiation mechanism.

IV. MATHEMATICAL MODEL

In addition to imposing non-negligible delay, switching the link rate also expends energy during the switching time when no data packets are transferred. In this case, power is consumed, but no work is being done. To achieve overall energy savings, the wasted energy must be amortized by spending a minimum amount of time in the low link rate (MTSLR – Minimum Time to Stay in Low Rate).

We denote the MTSLR as T_L. Let P_H and P_L denote the power consumed by the MAC and PHY layers at the high and low link rates respectively. T_{1MAC} and T_{1PHY} are the times taken by the MAC and PHY layers to switch the link rate down and T_{2MAC} and T_{2PHY} are the times required for the MAC and PHY layers to switch the link rate up. We define the *switching cycle* (T_{TOTAL}) as the total time to switch from the high link rate to the low link rate and then back to the high link rate.

$$T_{TOTAL} = T_{1MAC} + T_{1PHY} + T_L + T_{2MAC} + T_{2PHY}$$

Thus, the total power consumption (P_{ALR}) for the switching cycle is:

$$P_{ALR} = \frac{P_H * T_{1MAC} + P_H * T_{1PHY} + P_L * T_L + P_L * T_{2MAC} + P_L * T_{2PHY}}{T_{TOTAL}}$$

To obtain the minimum value for T_L, we differentiate the above equation with respect to T_{TOTAL} and equate it to zero.

$$P_H * T_{1MAC} + P_H * T_{1PHY} + P_L * T_{2MAC} + P_L * T_{2PHY} = - P_L \times T_L$$

For the net energy consumption to break even, the sum of the switching energy should be equal to the energy in the low link rate. Thus, the MTSLR is:

$$T_L \geq \frac{P_H}{P_L} * (T_{1MAC} + T_{1PHY}) + (T_{2MAC} + T_{2PHY})$$

Given the link switching times and power consumed by the device at both the high and low link rates, these equations can assist in control policy development to ensure overall energy savings.

V. EXPERIMENTAL METHODOLOGY

A. Prototype Hardware Architecture for an ALR-enabled NIC

We developed a hardware prototype of an ALR-enabled NIC in order to analyze and evaluate the performance of ALR in terms of link rate switching time and power consumption.

We utilized the Xilinx Virtex-II Pro development kit (XC2VP20-FF896) manufactured by Avnet Inc [2]. The Avnet board includes a Xilinx Virtex-II Pro FPGA, a Spartan-IIE FPGA, flash memory, SRAM 256MB DDR SDRAM, a PCI interface, a 10/100/1000 Ethernet PHY, and RJ-45 Ethernet connector. We built our NIC architecture based on the RICE programmable NIC [21]. We removed the DMA Unit, DDR controller, and Spartan bridge, since the communication between the NIC and host is not our research focus and resources are limited. The Virtex-II Pro FPGA contained most of ALR-enabled NIC logic including a PowerPC processor, on-chip memory, the clock control module, the MAC control unit, and the ALR MAC core. A high speed, 64-bit memory-mapped processor local bus (PLB) operating at 100 MHz connected these components.

The ALR MAC core is the focal point of ALR MAC handshake protocol implementation and generates the ALR control frames. We developed the ALR MAC core based on the tri-mode Ethernet MAC core available from Opencores [19]. We removed some internal modules and added extra components to existing modules to incorporate the ALR functionality.

Figure 3 depicts the architecture of the ALR MAC core consisting of the transmitting and receiving engines and the PHY interface, as well as critical components outside of the MAC core. The transmitting engine includes the following modules: the ALR control frame generator (MAC_tx_Ctrl), destination and source MAC address generator (MAC_tx_addr_add), cyclic redundancy check (CRC) generator (CRC_gen), and the packet transmit buffer (MAC_tx_fifo). The receiving engine includes the ALR control frame response module (MAC_rx_Ctrl), the packet receive buffer (MAC_rx_fifo), and the CRC (CRC_check) module. The transmitting and receiving engines communicate and synchronize with each other via the flow control module (Flow_ctrl). The GMII and MII modules enable two way communications of ALR and data frames between the MAC core and the PHY. The MAC control unit is located outside the MAC core, and provides data frames via the MAC_tx_fifo buffer.

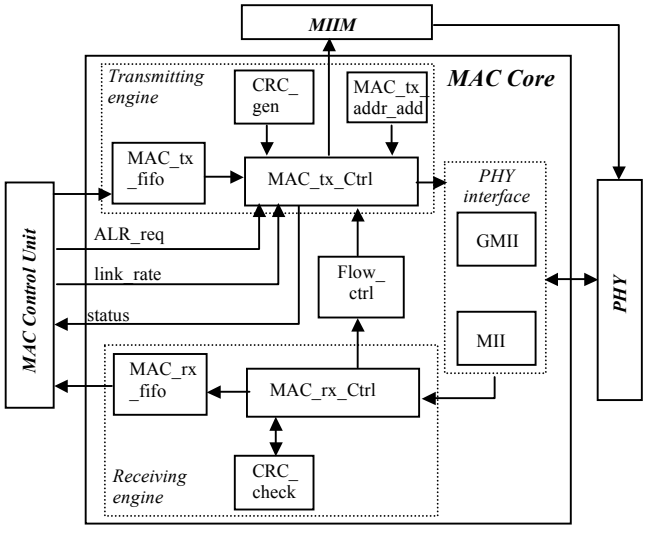

Figure 3: ALR MAC core block diagram

The MAC control unit also contains the control policy. To initiate a link rate switch, the control policy sets the ALR request signal (ALR_req) to trigger the MAC core to begin the ALR MAC handshake protocol. The MAC control unit also specifies the desired link rate on the link_rate bus. The status signal indicates to the MAC control unit that link rate switching is in progress, so that no data packets are sent to the MAC core. Once ALR_req is set, the transmitting engine of the MAC core generates the ALR request frame and refrains from sending data frames until it receives an indication from the receiving engine that it has received an ACK/NACK frame.

We added ALR capabilities to the receiving engine by adding ALR control frame processing to a custom version of the standard MAC data reception module. The receiving engine stores the data frames received from the PHY into a small FIFO until the MAC control unit transfers the frames to its own internal buffer. If the receiving engine receives an ALR control frame indicating the link rate should switch, the receiving engine informs the transmitting engine to configure the PHY via the media independent interface management (MIIM) [14] module.

B. Experimental Setup and Measurement Methodology

Our experimental setup consisted of two desktop workstations both running Linux OS to execute the firmware compiler and Xilinx FPGA tools, including Xilinx ISE [26], Xilinx EDK [25], and Xilinx Chipscope [24]. Each workstation was connected to one of the standalone ALR-enabled NIC via the JTAG port and RS-232 serial port. The JTAG port was used for downloading compiled bitstreams to the FPGA and updating firmware. The serial port was used for viewing console output from the firmware. We connected the RJ-45 Ethernet connectors of the two NICs using two meters of CAT 5E crossover cable. We used Xilinx Chipscope to debug hardware, view the timing diagram, and monitor the link rate switching.

We measure the total switching time as three separate components to identify the bottlenecks. These components are the MAC handshaking time, the PHY register configuration time, and the PHY resynchronization time. We force our system to switch link rates by periodically sending ALR request signals. Once the ALR request is initiated, we set a timer, and record the sequence number for the ALR request frame. After receiving the corresponding ACK, the timer value reports MAC handshaking time. The PHY register configuration time can be viewed directly through Chipscope. After completion of the PHY register configuration, another timer records the time for PHY resynchronization, which can also be viewed from Chipscope.

To measure power consumption during a link rate switch, we force our system to switch link rates as quickly as possible so that the sustained power consumption will reflect only the power consumed during the actual switching. This switching period is the measured link switching time. We measure the average power consumption during constant switching using Xpower [27] and a Watts Up Pro power meter [22].

We note that, without loss of accuracy, we do not need to simulate Internet traffic traces to gather both link rate

TABLE 1. LINK RATE SWITCHING TIMES

Link Rate Switching	MAC Handshake Time (us)	PHY Register Configuration Time (us)	PHY Resynchronize Time (ms)	Total Switching Time (ms)
1Gbps/100Mbps	2.0	13.84	72.320	72.3
100Mbps/1Gbps	8.4	13.84	68.572	68.6
100Mbps/10Mbps	2.0	13.84	575.813	575.8
10Mbps/100Mbps	88.6	14.00	72.320	72.4
1Gbps/10Mbps	8.4	13.84	575.813	575.8
10Mbps/1Gbps	88.6	14.00	68.572	68.7

switching time and power consumption. These two measurements are largely independent of both traffic and control policy. The link switching time has only a small dependency on the particular moment that link switching is determined, because transmission of the current data frame must be completed before link switching occurs.

VI. PERFORMANCE ANALYSIS OF ALR

In this section, we present our measured link rate switching times and power consumption. In addition, we evaluate two control policies based on MTSLR, power, and energy consumption given our measurements.

A. *Link Rate Switching Time*

We measure and report MAC handshaking, PHY register configuration, and PHY resynchronization times separately to show that PHY resynchronization is the dominating component. MAC handshaking time depends on the length of the cable and minimum inter-frame gap, which is the minimum idle period between transmissions of Ethernet frames. In our experiments, we use a two meter cable. The longest possible cable for 1000BASE-T Ethernet is 100 meters which would only increase the MAC handshaking time on the order of microseconds, but can largely increase the PHY resynchronization time due to increases in noise and phase lock time, the time spent acquiring signal frequency [3].

Table 1 summarizes the MAC handshaking, PHY register configuration, and PHY resynchronization times. As expected, the MAC handshaking time depends greatly on the initial link rate. Switching from an initial link rate of 10 Mbps is 10 to 44 times longer than switching from an initial link rate of either 100 Mbps or 1 Gbps.

The PHY register configuration time is independent of link rate, since configuration control data are sent at a constant frequency of 2.5 MHz.

In contrast to the MAC handshaking time, the PHY resynchronization time depends on the target link rate. When the link rate switches to either 1 Gbps or 100 Mbps, the PHY resynchronization time is approximately 70 ms, which is well within the range proposed by [3][20]. However, when the link rate switches to 10 Mbps, the PHY resynchronization time increases dramatically to 576 ms. This is due to the increased time for the devices to negotiate with each other at the PHY layer with lower transmitting and receiving frequencies.

Not only does the switching time vary based on initial and target link rates, we point out that switching times are asymmetric. For instance, switching from 1 Gbps to 10 Mbps takes 8.4 times longer than switching from 10 Mbps to 1 Gbps.

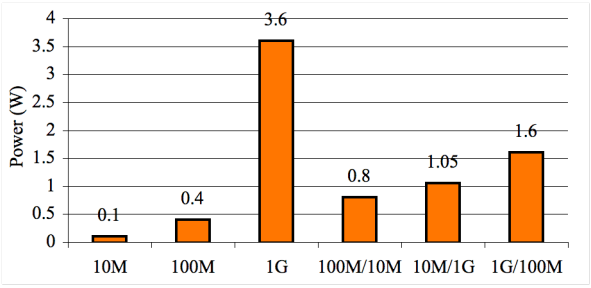

Figure 4: Power consumption for constant link rate and link rate switching.

Both the drastic differences in switching times and the asymmetry reveal three new considerations. First, for control policies to be most effective, they should consider all available link rates, and not focus on switching between only two link rates. Furthermore, since these switching times can be dependent on both architecture and environment, we propose that control policies poll switching times at startup and incorporate these values when making switching decisions.

Secondly, current control policy development has largely revolved around millisecond or even microsecond switching times. In those studies, extremely fast switching times relaxed buffer pressure and had little impact on maximum packet delay. However, large switching times exacerbate buffer pressure, increasing metrics such as mean and maximum packet delay and buffer overflows. We evaluated both the dual threshold and timeout threshold control policies and observed that effects of increased switching time appear to additively affect these metrics, illustrating similar trends for metric verses link utilization.

Furthermore, we acknowledge that our measured switching times are much longer than 1 ms, which is assumed in previous research [8][9][10], but our switching time between 1 Gbps and 100 Mbps is well within the time projected by [3][20]. Unfortunately, this lengthy time is due to limitations of the current PHY technology and IEEE standards. Currently, the fastest available switching mechanism is MAC handshaking with PHY register configuration to manually force the PHY to change rates. Manual PHY register configuration is used in PHY resynchronization to skip unnecessary auto negotiation steps that occur during initial link setup. This switching mechanism reduces the switching time from seconds to milliseconds, but is still much longer than 1 ms. Despite this discrepancy, the asymmetry and variation in switching times that we measure reveal important results needed for further control policy development which is beyond the scope of this paper.

B. *Link Rate Switching Power Consumption*

Figure 4 shows the power consumption for constant link rates and link rate switching. The first three bars show the power consumption for constant link rates of 10 Mbps, 100 Mbps, and 1 Gbps. The measurements show a near exponential increase in link rate power consumption [5]. The last three bars show the link rate switching power when the link switches between two different rates. Link rate switching

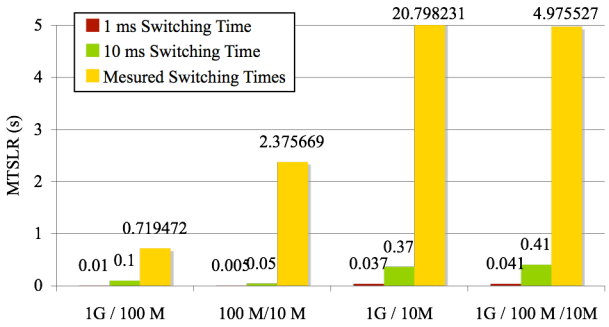

Figure 5: Minimum time to stay in the low link rate (MTSLR) for assumed switching times of 1 ms and 10 ms and our measured switching times. Each bar grouping shows MTSLR for switching from an initial rate to a target rate (initial/target) or an initial rate to a target rate via an intermediate rate (initial/intermediate/target)

power, on the other hand, shows a different trend. During switching, a certain amount of time is spent in both link rates. The link rate switching power cannot simply be assumed to be the average of the initial and target link rates constant power consumption because the time spent in each rate is different. This is evident given the results we present in Table 1. A worst-case scenario could always assume that the power consumed during switching is simply fixed at the power consumption of the highest link rate, but results would be pessimistic, and could compound to very large discrepancies when calculating energy savings.

C. *MTSLR*

The first three bar groups of Figure 5 show the MTSLR for switching between different high initial rates and low target rates (initial/target) for assumed switching times of 1 ms and 10 ms, and our measured switching times. For very low switching times, the MTSLR is largely negligible and, given the bursty nature of network traffic, MTSLR is likely not a concern. However, we notice that for our measured switching times, the MTSLR is quite different. For 1G/100M, the MTSLR is low but hardly negligible. However, we see a very large increase in MTSLR when the target link rate is 10 Mbps due to the long switching time.

Even though the switching time for 100M/10M is nearly the same as 1G/10M, the MTSLR is more pronounced because the long switching time is spent in a higher power consuming state for 1G/10M. We observed that the MTSLR for 1G/10M is not simply the sum of 1G/100M and 100M/10M. This is due to the highest power consuming switch (1G/100M) being a fast switch and the lowest power consuming switch (100M/10M) being a slow switch. Given this, we propose a step-down technique, which switches the link to an intermediate rate before switching down to the ultimate target rate. The last bar group of Figure 5 shows the MTSLR using the step-down technique to switch from 1G/10M by switching from 1G/100M then switching from 100M/10M. The MTSLR is still quite large but is 75% less than switching directly from 1G/10M.

D. *Control Policy Power Analysis*

We reanalyze the dual threshold [8] and timeout threshold [10] control policies for switching between 1 Gbps and 100 Mbps using our measured switching times. To simulate network traffic, we use a Poisson process to model the arrival of packets of a constant maximum packet length of 1518 bytes, which reflects that used in previous work [9][10]. To represent different network traffics ranging from light to heavy, we vary the average link utilization of input traces from 1% to 15%. High and low buffer threshold values are specified as 0 KB and 32 KB respectively [8][9][10].

Figure 6 shows power consumption for the dual threshold policy (a), the timeout threshold policy with a 10 ms holding time (b), and the timeout threshold policy with a 100 ms holding time (c) for both an assumed link switching time of 1 ms and our measured link switching time of 70 ms for varying link utilization. The dual threshold policy results in similar power consumption regardless of link switching time due to frequent link oscillation. In comparison, the timeout threshold policy with a 100 ms holding time reveals less power consumption with a 70 ms switching time than with a 1 ms switching time when link utilization is between 1% and 10%. This reduction is a result of the 70 ms switching time forcing the link to spend more time switching than staying in the high rate. For the timeout threshold policy with a 10 ms holding time, the 70 ms switching time causes the device to experience a sharper increase than it does with a 1 ms switching time. This is because the timeout threshold policy with a long switching time is less sensitive to variation of traffic utilization than with a short switching time.

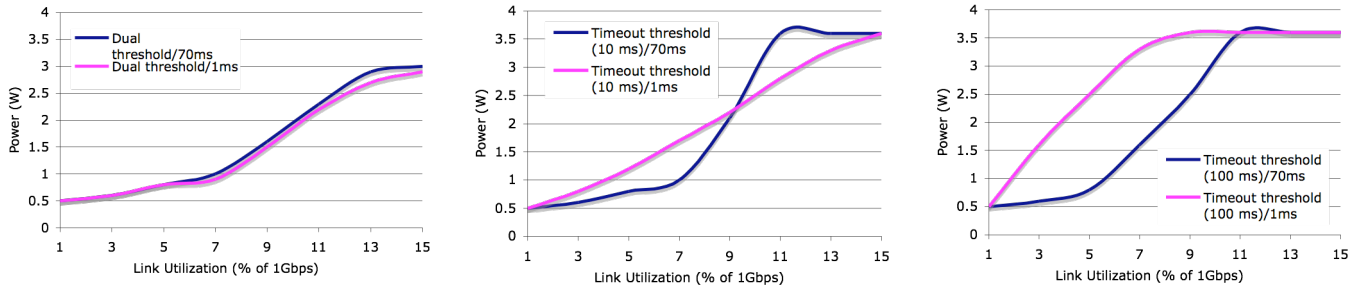

Figure 6: Power consumption for the (a) dual threshold policy, (b) timeout threshold policy with a 10 ms holding time, and (c) timeout threshold policy with a 100 ms holding time for an assumed link switching time of 1 ms and our measured link switching time (70 ms).

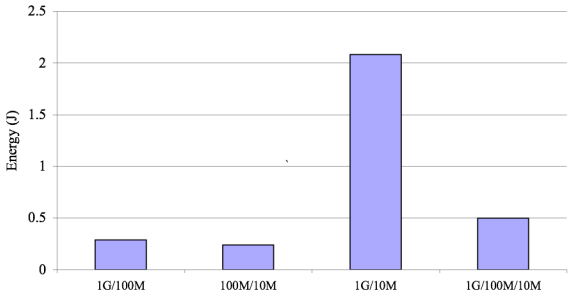

Figure 7: Energy consumption for various link rate switches. Each bar shows energy consumed when switching from an initial rate to a target rate (initial/target) or an initial rate to a target rate via an intermediate rate (initial/intermediate/target)

E. *Energy Analysis*

Given the drastically different switching times depending on the target link rate, we analyze the energy consumed during the link switching process. The first three bars in Figure 7 show energy consumed while switching from a high initial rate to a low target rate (initial/target). The energy consumed by 1G/10M is 4 times that of the energy consumed by 1G/100M or 100M/10M due to the reasons we discussed in sections IV and 0.C. Using our step-down technique (bar four in Figure 7), the energy consumption is 75% less than a direct 1G/10M switch.

VII. Conclusions and Future Work

This paper presents three major contributions. First, we are the first to build an ALR-enabled NIC on an FPGA platform, and disseminate the real-time switching overheads, which are important for evaluating ALR performance. Second, we developed an ALR-capable NIC, which we make available for the research community [23] to test and perform experiments with ALR for further study. Third, we utilize our measured link switching times and power consumption to re-analyze control policies and identify new considerations in future research. Our future work includes developing a complete hardware prototype system with control policy mechanism capabilities, which can act on real time traces to assist control policy development and evaluation.

VIII. Acknowledgments

This material is based on work supported by the National Science Foundation under Grant No. 0520081

IX. References

[1] H. Anand, C. Reardon, R. Subramaniyan, and A. George, "Ethernet Adaptive Link Rate (ALR): Analysis of a MAC Handshake Protocol", Proceedings of the IEEE Conference on Local Computer Networks, pp. 533-534, November 2006

[2] Avnet Inc. http://www.avnet.com/

[3] M Chadha, J. Barnette, amd W. Lertniphonphun "Feasibility of 1000-Base-T RPS Restart" Presentation for IEEE 802.3az Task Force. April, 2007. http://grouper.ieee.org/groups/802/3/eee_study/public/apr07/chadha_1_0407.pdf

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

[5] K. Christensen and B. Nordman, "Improving the Energy Efficiency of Networks: A Focus on Ethernet and End Devices," presentation to Cisco, San Jose, October 20, 2006.

[6] W. Diab and S. Powell, Broadcom "Subset PHY: Cost and Power Analysis" Presetation for IEEE 802.3az Task Force. Sept 2007. http://grouper.ieee.org/groups/802/3/eee_study/public/sep07/diab_2_0907.pdf

[7] C. Gunaratne, K. Christensen, and B. Nordman, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed," International Journal of Network Management, Vol. 15, No. 5, pp. 297-310, September/October 2005.

[8] C. Gunaratne, K. Christensen, and S. Suen, "Ethernet Adaptive Link Rate (ALR): Analysis of a buffer threshold policy," Proceedings of IEEE GLOBECOM 2006, November 2006.

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

[10] C. Gunaratne, K. Christensen, S. Suen, and B. Nordman, "Reducing the Energy Consumption of Ethernet with an Adaptive Link Rate (ALR)," IEEE Transactions on Computers, April 2008

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

[12] M. Gupta and S. Singh, "Dynamic Ethernet Link Shutdown for Energy Conservation on Ethernet Links", Proceedings of the IEEE International Conference on Communications 2007, June 2007.

[13] R. Hays, Intel Corporation "Active/Idle Toggling with Low-Power Idle" Presentation for IEEE 802.3az Task Force. Jan 2008. http://grouper.ieee.org/groups/802/3/az/public/jan08/hays_01_0108.pdf

[14] IEEE 802.3 Standard Part 3: Carrier sense multiple access with collision detection (SMA\CD) access method and physical layer specifications.

[15] IEEE 802.3 az study group http://grouper.ieee.org/groups/802/3/eee_study/index.html

[16] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, " On the self-similar Nature of Ethernet Traffic," In IEEE/ACM Transactions on Networking, vol.2 , No.1 , Feb. 1994.

[17] B. Nordman and K. Christensen, "Reducing the Energy Consumption of Network Devices," Tutorial for the July 2005 IEEE 802 LAN/MAN Standards Committee Plenary Session, San Francisco, July 19,2005

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

[19] Opencores. http://www.opencores.org

[20] S. Powell and H. Frazier, "Technical Considerations and Possible Solution Sets for EEE" Presentation for IEEE 802.3az Task Force http://grouper.ieee.org/groups/802/3/eee_study/public/may07/powell_2_0507.pdf

[21] J. Shafer and S.Rixner, "A Reconfigurable and Programmable Gigabit Ethernet Network Interface Card", Technical report TREE0611, Department of Electrical and Computer Engineering, Rice University, December 2006

[22] Watt Up Pro Power meter, http://www.wattsupmeter.com.htm

[23] http://www.csee.usf.edu/~christen/energy/main.html

[24] Xilinx Chipscope, http://www.xilinx.com/ise/optional_prod/cspro.htm

[25] Xilinx EDK, http://www.xilinx.com/ise/embedded/edk_docs.htm

[26] Xilinx ISE, http://www.xilinx.com/ise_eval/index.htm

[27] Xilinx Xpower, http://www.xilinx.com/products/design_tools/logic_design/verification/xpower.htm