Smart-NICs: Power Proxying for Reduced Power Consumption in Network Edge Devices

Karthikeyan Sabhanatarajan, Ann Gordon-Ross*, Mark Oden, Mukund Navada, Alan George*

HCS Research Lab, ECE Department, University of Florida.

{sabhanatarajan,ann,oden,navada,george}@hcs.ufl.edu

** Also with the NSF Center For High-Performance Reconfigurable Computing at the University of Florida*

Abstract

The number of edge devices connected to the Internet is increasing at a rapid rate. To maintain network connectivity, the majority of these devices remain completely powered on when idle, wasting unnecessary energy. A novel idea to conserve energy while maintaining network connectivity is to place the computer in standby mode during idle periods and delegate the packet-handling functions to its network interface card (NIC). The NIC, acting as a liaison for the host, can proxy a variety of network protocols, increasing the standby time of the host without compromising its active connections. In this paper, we analyze the requirements of such a packet classifier and design a low-power hardware-based packet classification technique, which, compared to a software-based packet classification technique, consumes 59% less energy with a 9x speedup.

1. Introduction

With the rapid increase in the number of edge devices connected to the Internet, the aggregate power consumption of these devices will become a major concern in the near future [2]. The most prevalent of these edge devices are consumer desktop computer systems, consuming on average 60 – 95 watts of power and up to 195 watts in high-end systems [22].

Research estimates that these systems are on average left idle for 75% of the time when powered on [18]. During these idle periods, systems could be powered down to a standby mode to reduce power consumption by 80% [2]. However, standby mode currently disrupts the system's network connectivity. Popular Internet applications such as peer to peer (P2P) clients and instant messengers demand continuous network connectivity in order to respond to incoming file queries and to announce a user's presence. In order to ensure this connectivity, users typically disable the power management features, inhibiting the transition to standby mode, and thereby increasing the energy consumption of otherwise idle systems. However, given existing system architectures, disabling standby mode is the only option to retain two-way network connectivity for user applications.

A novel approach to address this problem is to augment the network interface card (NIC) to act as a proxy (or liaison) for the system during standby mode, and maintain network connectivity by handling a subset of certain application network protocol semantics [8][18]. This subset has the unique characteristic that responses do not require a complex decision process, thus the NIC can proxy automated responses, allowing the system to remain in standby mode – a technique known as *power proxying*. Network protocols that are amenable to proxying are called *proxiable protocols*. Purushothamom et al. [18] demonstrated that the NIC can successfully proxy portions of P2P application protocol semantics, increasing the amount of time a system can be in standby mode by 85%. Similarly, several other applications such as instant messengers, initiating sessions of Internet telephony, and new mail notification of e-mail clients are suitable for power proxying.

For a NIC to provide the capability of power proxying, *power proxying rules* are required to enable the NIC to identify packets that may be appropriately responded to using proxiable protocols. The system provides these rules to the NIC immediately prior to entering standby mode. Such a 'smart'-NIC (SNIC) would, upon receiving a packet, identify the packet and either respond appropriately or wake up the system.

To provide this functionality, the SNIC must have a *packet classifier*, a method of examining incoming packets to determine the appropriate action. Thus, *packet classification* is the process of determining which rule an inbound packet satisfies. This packet classification methodology is similar to that performed in traditional routers. However, router techniques for packet classification are not directly applicable to a desktop NIC due to high resource requirements such as power, energy, and computational intensity. Even though resources to implement packet classification exist on modern NICs with built-in embedded RISC processors [3][9][14][21], these processors are significantly less powerful than those available on routers, and are unable to meet necessary classification speeds. Alternatively, router processors are too costly and demanding for implementation on consumer NICs. Thus, a low-power, small area, and efficient packet classification scheme is required.

In this paper, we present, to the best of our knowledge, the first hardware-based, low power packet classification scheme for reduced power consumption in NICs. We first analyze the characteristics of power proxying rules and develop a packet classifier that suits the proxying requirements. We initially prototype our packet classifier in software intended to run on existing NIC processors and show that these processors lack the ability to classify packets quickly enough to keep up with link speeds greater than 100 Mbps. To keep pace with rapidly advancing technology, we develop and analyze a hardware-based packet classifier to achieve packet classification throughput nearly reaching that needed for 10 Gbps link speeds.

2. Background and Related Work

Power proxying on a NIC requires three key elements. First, a methodology is needed to control the state switching of the system. Second, protocol classes amenable to power proxying must exist. And lastly, the NIC must have the necessary computing resources to implement the power proxying

technique. In this section, we elaborate on these key requirements.

2.1 Power Proxying with the SNIC

We partition our system into the two components responsible for responding to network traffic at particular times: the operating system (OS) and the SNIC. When the system is in full power mode, the OS responds to network traffic, while the SNIC responds when the system is in standby mode.

We define the delegation of network control between the SNIC and the OS. Before the system transitions to the standby state due to system idleness, the PC offloads power proxying rules to the SNIC for all active networking applications and delegates the network control to the SNIC. Thus, the PC enters the low-power state without disrupting network connectivity. At this time, the packet classifier on the SNIC applies the power proxying rules to incoming packets. If the packet classifier identifies a packet that can be handled by the SNIC, the proxy module identifies the target application using the application ID field within the power proxying rule (we discuss further details in section 3.1). The proxy module invokes a corresponding application handler to respond to the packet. For every proxiable application that runs in the system, there is a corresponding application handler running on an embedded processor inside the SNIC to handle protocol semantics. The application handler determines an appropriate response by inspecting the payload of the packet. Additionally, if the packet classifier recognizes the packet as network chatter that is not intended for the system, the packet is discarded.

Two situations exist where the SNIC is unable to proxy a response to an incoming packet, requiring the PC to be woken up out of standby mode using a Wake on LAN (WOL) [4] interrupt. The first is when the SNIC receives a packet that does not match any power proxying rule and is not network chatter. The second is when certain proxied applications such as Internet telephony, on reception of certain types of packets, demand the PC to be woken up.

2.2 Protocol Semantics and Applications

Popular networking applications such as P2P file sharing programs, instant messengers, Internet telephony, and diagnostic applications such as 'ping' have proxiable features, making them the most promising candidates for power proxying. These applications are grouped under one of four protocol classes: Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP).

ARP request packets do not require the PC to be powered on and can be easily delegated to the SNIC. For example, IP conflicts can be avoided when the PC is in a standby mode by allowing the SNIC to respond to gratuitous ARP packets.

Ping, the popular network diagnostic application, uses ICMP to request and respond to messages to detect the presence of a particular system and is amenable for proxying.

Many popular applications such as P2P file sharing programs and session initiations of Internet telephony applications use TCP for network communications. Additionally, several instant messaging implementations using the TCP class constantly send out user status packets (or presence packets) and are amenable to power proxying. While in standby mode, the SNIC can send out these packets at a constant time interval.

The fourth protocol class is UDP. A fitting application example for this category is a new e-mail notification sent as a UDP packet to the corresponding e-mail client [19]. Upon receiving this packet, the PC can be awoken by the SNIC to download the new message.

2.3 NIC-Based Computing Resources

Modern NICs contain embedded processors that are largely underutilized, and much current research focuses on exploiting these resources. Friedman et al. [3] conceived and implemented a NIC-based distributed firewall system called iNIC for Ethernet platforms using a 100 MHz Intel i960 RISC processor with 128 MB of RAM. Otey et al. [16] proposed and empirically evaluated a novel architecture for network intrusion detection systems using NICs for commercial Myrinet platforms featuring a 66 MHz LANai with 4 processors and 1 MB of memory. In [20], a gigabit Ethernet adapter built on a dual RISC processor architecture supported TCP offload implementation. Broadcom's family of Convergent-NICs (C-NIC) is another example of intelligent NICs. Killer NIC is a gaming NIC that utilized a 400 MHz Freescale RISC processor for accelerating game data [9]. All of these implementations function only during the powered-on mode of a system and none propose offloading processing to the NIC so that the system can be placed in a standby mode.

3. Packet Classifier

For a packet classifier to function successfully on the SNIC, we identify the characteristics of the power proxying rules and impose operating requirements.

3.1 Characteristics of Power Proxying Rules

Power proxying rules can uniquely identify application network traffic based on header fields such as port and/or source address. This means packet classification for power proxying is a 6-dimensional problem, the dimensions being the link-layer protocol, network-layer source and destination addresses, network-layer protocol, and the transport-layer source and destination port numbers. For example, all TCP application traffic flows can be uniquely recognized using the source and destination address and port header fields. For UDP applications, only the destination address and port fields identify the traffic flow. In addition to the header fields, link-layer protocol and network-layer protocol fields are required to distinguish between the four classes.

Conventional packet classifier rules are specified as address/mask and operator/number(s) pairs [6]. However, power proxy classifier rules are specified only in operator/number(s) format because the end points of a connection in the network are clearly defined. Therefore, the use of the address/mask representation is avoided. Since this paper primarily targets applications running on specific ports, we limit the scope of the operator to equality. If the packet classifier were to be extended for firewall and security applications, range operators (such as greater than and less than) could easily be implemented.

Upon matching a packet with a rule, the application handler utilizes the information provided by the packet classifier to selectively respond to incoming flows. Given certain situations,

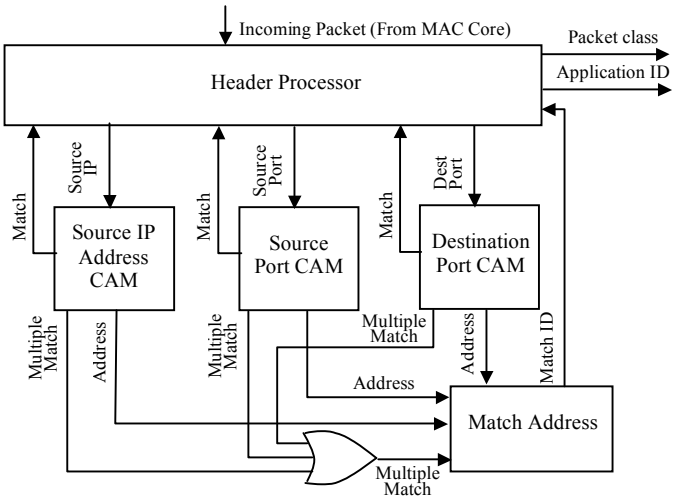

Figure 1: Architecture of the CAM-based hardware packet classifier.

the application handler may choose not to respond, electing to wake up the PC.

3.2 SNIC Packet Classifier Characteristics and Requirements

The SNIC packet classifier is similar to a router-based packet classifier, but the operating environments and goals differ. For example, the SNIC packet classifier operates only during periods of system inactivity and will only deal with packets addressed to the particular destination PC, unlike a router, which must deal with packets addressed to many destinations.

Additionally, the SNIC packet classifier operates under limited processing resources. A typical NIC processor's clock frequency ranges from 66 MHz to 400 MHz. In contrast, routers operate with dedicated network processors at GHz clock frequencies. However, even with limited resources, the packet classifier should be able to sustain link rates of 10/100/1000/10000 Mbps and the latency of the packet classification should avoid any packet loss.

Fundamentally, SNIC packet classification is similar to routing for delay sensitive applications. The primary difference is the nature and number of rules for both cases. Typically, router rules are more complex and are large in terms of quantity and size of rules. The number of rules a SNIC packet classifier searches is directly proportional to the number of running applications suitable for proxying, thus, there are significantly fewer rules. Additionally, SNIC rules are disjoint so that a packet obeys only one rule, in contrast to traditional router-based packet classifiers that have forward or backward redundancy [7].

4. Packet Classification Methods

We designed a software-based packet classification methodology to quantify the packet classification capabilities available on existing unaugmented NICs, and to serve as a comparison for our hardware-based classification methodology. The simplest software classification algorithm utilizes a binary search algorithm, while the simplest hardware classification implementation utilizes Content Addressable Memories (CAMs) [6][7].

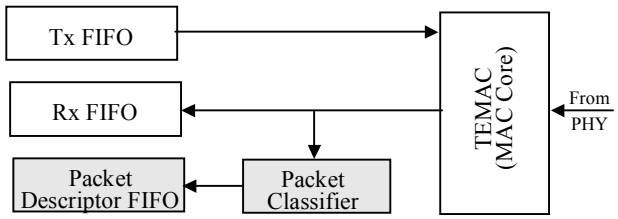

Figure 2: Architectural placement of the packet classifier. New components are shaded.

4.1 Software Packet Classification

We used existing embedded processors available on commercial NICs to implement a software packet classifier. We implemented the software packet classifier using a binary search algorithm with a complexity of O (log N) to find a matching rule.

The software packet classifier functions as follows. A receiver FIFO buffers incoming packets until they are transferred to the NIC's memory. After the packets are moved, the MAC control unit notifies the embedded processor and packet classification begins. The software extracts the required header fields from each packet and passes the fields to the packet classifier implemented in firmware. Finally, the embedded processor performs a binary search on these rules and determines an appropriate action. The process of header extraction and basic classification functionality is similar to our hardware implementation, which we elaborate on in the next section.

4.2 Hardware Packet Classification

We implemented the hardware packet classifier using CAMs. Traditionally, routers use ternary CAMs (TCAMs) for packet classification. Since our packet classifier does not demand a longest prefix match, we can implement the classification using basic CAMs, which require less power than TCAMs.

Figure 1 shows the architecture of our hardware packet classifier. The header processing unit acts as the primary control module and is responsible for extracting the necessary data from the header of the packets, supplying the CAMs with source IP, source port, and destination port. Additionally, the header processor maintains classifier state. The packet classifier receives the input packets from the MAC core.

Figure 2 shows the placement of the packet classifier with respect to the MAC core. The MAC core is attached to two FIFOs, one for transmitting (Tx FIFO) and the other for receiving (Rx FIFO). When a new packet arrives, the MAC core buffers the packet in the Rx FIFO at the rate of one byte per clock cycle [25]. The packet descriptor FIFO is a data structure where the packet classifier writes all the information regarding the classification of a packet, including the packet's class and its matching address, if any. The application handler running on the SNIC processor uses this information to determine the appropriate action.

A packet's critical path is between the MAC core and the receiver FIFO. Because the packet classifier lies outside this path, it does not increase the critical path latency of the packet.

We implemented the header processing unit as a finite state machine, which is triggered when a packet arrives from the MAC core. The Ethernet protocol field specifies if a packet is an ARP or an IP packet. An ARP packet has the quickest classification time, as it requires only a single comparison of the Ethernet protocol field. In the case of an IP packet, the header processor checks whether it is an ICMP, TCP, or a UDP packet. For each packet, the layer three destination address field is checked to see if it matches the PC's address, as we are only interested in packets destined for the host.

Next, the packet classifier compares TCP and UDP packets against the power proxying rules stored in the CAMs. We partitioned the source address, source port, and destination port, and store them in separate CAMs. For a TCP packet, the packet classifier extracts the layer three source address information from the incoming packet data and searches the source address CAM. Only upon a match will the packet classifier continue with packet classification. If the packet classifier finds no match in the source address CAM, the packet classifier interrupts the processing element on the NIC to wake up the PC. Alternatively, if a match in the source address CAM occurs, the packet classifier extracts the source port from the header and searches the source port CAM. If a match occurs in the source port CAM, the packet classifier checks the destination port CAM. Since the CAMs are sequentially searched, unnecessary switching activities are avoided if the header processing unit detects a mismatch in the earlier phases, saving power when compared to a single CAM implementation.

A rule for TCP matches if and only if all three CAMs return the same matching address. In the case of UDP packets, only the destination port CAM needs to match and since the destination address is a single value, the address can be stored in a register and the packet classifier performs an equality comparison.

For TCP traffic, cases arise where multiple TCP flows will map to a single TCP application, giving rise to multiple matches. The match address unit addresses this issue using the multiple match flags and representing the CAM addresses in bit vector format [26]. In such a case, the unencoded CAM address forms a bit vector where each bit indicates a matching address. A match occurs for a TCP application by intersecting the bit vectors of all three CAMs.

5. Experimental Results

We performed experiments to compare the software and hardware classifiers in terms of classification speed and dynamic power dissipation.

5.1 Experimental Setup

We implemented the software packet classifier using the embedded PowerPC 405 on the RiceNIC platform [21]. The RiceNIC is a programmable network interface card that incorporates an FPGA and two embedded PowerPCs. The RiceNIC implementation clocks the PowerPC at 300 MHz and the processor bus at 100 MHz. We also modified the PowerPC to operate at 100 MHz in order to observe the performance of the packet classifier at lower clock frequencies, representing low-end NICs.

Our experimental setup for the software packet classifier consisted of two PCs, one emulating a network switch and the other equipped with the RiceNIC board. Using the packet generation tool NPG [13], the PC emulating the switch injected minimum sized packets to the RiceNIC equipped PC. We instrumented the RiceNIC to record packet classification time statistics.

We prototyped the hardware packet classifier on the Xilinx Virtex-II Pro FPGA XC2VP20 and used Verilog HDL and Xilinx IP cores to generate the CAMs with block memory. We developed and simulated the system, which implemented the Xilinx TEMAC core [25], using Xilinx ISE 9.1 [27] and ModelSim XE [12]. We designed system operation supporting the three link rates of 10/100/1000 Mbps, with corresponding clock frequencies of 1.25, 12.5, and 125 MHz, respectively. Next, we synthesized the hardware system and performed time-constraint based placement and routing with Xilinx XST. We subjected the system to heavy timing simulations (post-place and route timing simulation) using both the ISE simulator and Modelsim XE. We utilized Xilinx XPower [30] for power estimation [1].

Worst case power dissipation occurs when the hardware prototype continuously receives minimum sized Ethernet packets. Hence, we generated the test benches using minimum sized Ethernet packets (64 bytes) and created four types of test benches, each corresponding to one of the four protocol classes.

5.2 Packet Classifier Speed

The primary goal of the packet classifier is to meet the standard minimum-sized Ethernet packet throughput of 1.48 millions of packets per second (MPPS) at a 1 Gbps link rate. For the software packet classifier, worst-case packet classification time occurs when the matching rule is the last rule checked. For the hardware packet classifier, worst-case packet classification time occurs when all dimensions (CAMs) match. For a successful match, the worst-case classification time for the software classifier is O (log n) and O (1) for the hardware classifier.

Figure 3 shows the worst-case packet classification time for successful matches for both classifiers using a power proxy rule set containing 100 rules. In the hardware design, TCP packets take slightly more processing time than the UDP packets due to three sequential CAM lookups for TCP compared to a single lookup for UDP. As expected the hardware-based classification is much faster than software-based classification for both 100 MHz and 300 MHz processors.

Figure 4 examines the variation of the worst-case packet classification time for the TCP/UDP packets with varying rule set sizes. The hardware packet classification time is constant for any number of rules while the software packet classification time increases logarithmically.

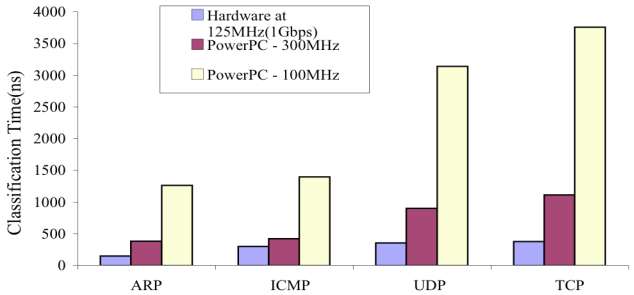

Figure 3: Worst-case packet classification time for each protocol class with a power proxy rule set of 100 rules

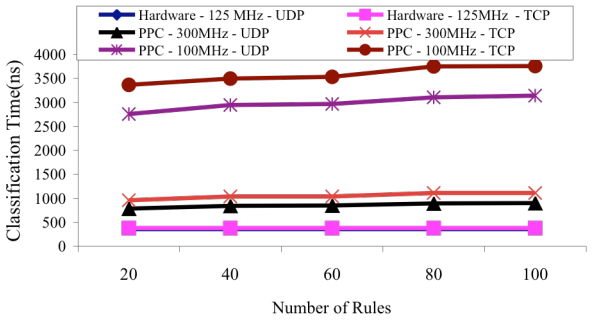

Figure 4: Worst-case packet classification time for TCP and UDP traffic vs. number of power proxying rules. (Both hardware classification times overlap on the bottom line.)

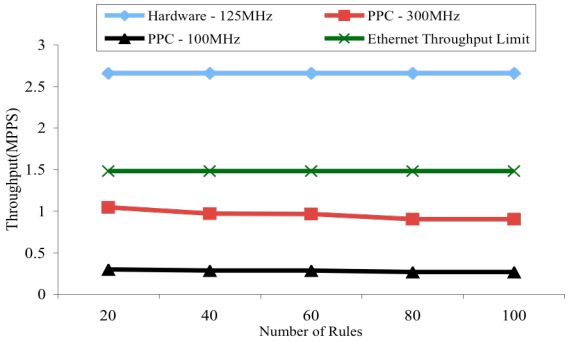

Figure 5: Obtainable throughput in MPPS for hardware and software packet classifiers vs. number of rules for TCP traffic.

We define the throughput of our hardware and software packet classifiers in terms of number of MPPS that the system is able to process. This metric reveals the maximum link rate sustainable by each technique. Figure 5 shows the obtainable worst-case throughput for both packet classification techniques for TCP packets. The software implementation operating at 300 MHz can only process at most 1 MPPS and fails to meet the gigabit Ethernet throughput requirement which may lead to unnecessary dropping of packets. We also estimate that the embedded processing element's clock rate should be at least 500 MHz to meet the gigabit Ethernet throughput requirement. The hardware implementation comfortably meets the throughput requirement and supports up to 2.5 MPPS operating at 125 MHz. At this packet classification speed, the classifier can support one link of 1 Gbps and up to 7 links of 100 Mbps speed giving a total link rate of 1.7 Gbps.

During idle times the system may not be subjected to a huge influx of packets, thus the software implementation may be fast enough to support classification. However, 1 Gpbs link rates are becoming commonplace and 10 Gpbs link rates will soon follow. Significantly more powerful embedded processors are required to speedup software packet classification to meet future link speeds, and these embedded processors are likely too power hungry to be included on a desktop NIC. Not only is our hardware classification technique much closer to meeting 10 Gpbs link rates (and in some rule cases, does meet the requirements), we project that we can optimize the hardware to maintain a link rate of 10 Gpbs with minimal added power overhead.

Figure 6 shows the speedup obtained with hardware classification versus software classification. We assume the hardware and software are continuously supplied with packets to classify. This figure denotes the lower bound on the achievable speed up. Speedup times range from 2.5x to 9x depending on traffic type and available NIC processing speeds.

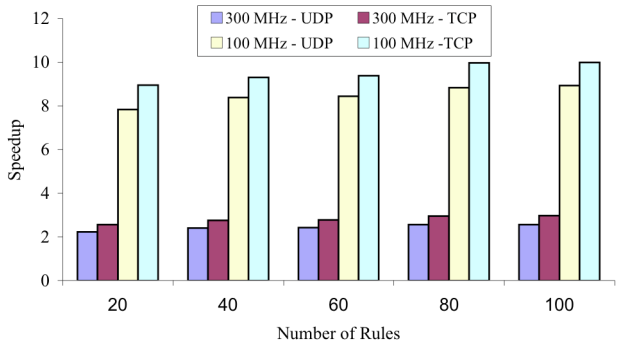

Figure 6: Speedup obtained by using a hardware classifier compared to a software classifier for varying number of rules.

5.3 Power Consumption

We estimated the power consumption of our hardware design using Xilinx XPower. The embedded PowerPC core in the Virtex-II Pro consumes 0.9mW/MHz at an ambient temperature of 25^{o}C [29]. We obtained the power consumption of the PowerPC system operating at 100 MHz and 300 MHz with the bus interface clocked at 100 MHz using the online power estimation tool [28], and found the power consumption to be 100 mW and 280 mW respectively. However, [17] reveals a more realistic power estimation that also accounts for the bus power dissipation. Thus, the PowerPC consumes 259.5 mW and 441 mW of power when clocked at 100 MHz and 300 MHz respectively. These numbers are in close agreement with [15], which also estimates the idle power of the PowerPC to be 50 mW. We obtain all power estimations at an ambient temperature of 25^{o}C.

The highest measured power consumption of our hardware packet classifier is 180 mW when it processes a TCP packet with 100 rules. The software packet classifier consumes between 2.4x and 2.9x more power than the hardware packet classifier. We project that in order for the software classifier to meet the 1 Gbps throughput requirements, the processor must operate at 500 MHz requiring an additional 294 mW over the 300 MHz processor – 4x more power than the hardware packet classifier.

Figure 7 shows the variation of the average hardware power consumption for various packet classes across different link rates for 100 rules. We can trace an exponential increase in power consumption with increasing link rate speed due to the system clock frequency, which is a function of the exponentially increasing link rate. As seen in the figure, processing a TCP packet involves slightly more power than processing other packets. This increase results from switching activity in the source address and source port CAMs, which only occurs in TCP packets.

5.4 Hardware Operating Frequency and Scalability

We obtained a maximum frequency of 177.17 MHz for an implementation with 20 rules and a minimum frequency of 138.9 MHz for an implementation with 100 rules. The standard frequency requirement for a 10 Gbps link rate is 156.25 MHz,

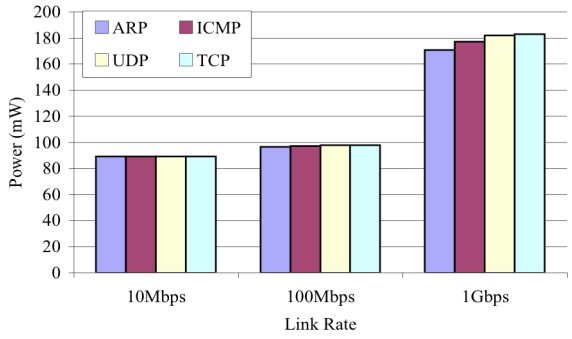

Figure 7: Hardware power consumption vs. link rate for 100 rules.

which transmits data in units of 64 bits. Our prototype meets this requirement for 20 rules and we project that with larger FPGAs, our prototype can easily meet the 10 Gbps frequency requirement for an implementation with 100 rules.

6. Conclusions and Future Work

Power proxying is a key element in realizing energy savings in network devices and allows them to be placed in standby mode without losing network connectivity. In this paper, we analyzed power proxying rules and developed packet classifiers intended to operate on a NIC to enable power proxying. We developed a low power hardware-based packet classification technique and analyzed it in terms of classification speed, packet throughput, and power consumption compared to a software-based implementation. Our hardware packet classifier was able to meet the Gigabit link rate requirements comfortably, with only minor optimizations needed to satisfy 10 Gbps link rates. An equivalent software-based packet classifier would consume 4x more power than the hardware-based packet classifier. Additionally, the hardware packet classifier was up to 9x faster than the software packet classifier, allowing the PC to be awoken sooner, thus reducing the possibility of packet loss.

Future directions of work include development of an energy efficient content inspection module to extend the functionality of SNICs.

7. Acknowledgements

This work is supported by the National Science Foundation under Grant No. 0520081. The authors would like to thank Casey B. Reardon for his insightful comments and review of the paper.

8. References

[1] J. Becker, M. Huebner, and M. Ullmann, "Power estimation and power measurement of Xilinx Virtex FPGAs: trade-offs and limitations", *Proc. of 16th Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2003.

[2] K. Christensen, P. Gunaratne, B. Nordman, and A. George "The next frontier for communications networks: power management," *Computer Communications*, vol. 27, no. 18, pp. 1758-1770, December 2004.

[3] D. Friedman and D. Nagle, "Building Firewalls with Intelligent Network Interface Cards", *CMU SCS Technical Report*, CMU-CS-00-173, May 2001

[4] J.A. Gil-Martinez-Abarca, F. Macia-Perez, D. Marcos-Jorquera,V. Gilart-Iglesias, "Wake on LAN over Internet as Web Service," *IEEE Conference on Emerging Technologies and Factory Automation, 2006. ETFA '06,* Sept. 2006

[5] P. Gupta, S. Lin and N. McKeown, "Routing Lookups in Hardware at Memory Access Speeds", *Proc. Infocom*, April 98, San Francisco.

[6] P. Gupta and N. McKeown, "Packet Classification on Multiple Fields", *Proc. Sigcomm, Computer Communication Review*, vol. 29, no. 4, pp 147-60, September 1999.

[7] P. Gupta and N. McKeown, "Algorithms for Packet Classification", *IEEE Network Special Issue*, vol. 15, no. 2, pp 24-32, March/April 2001.

[8] M. Jimeno, K. Christensen, and A. Roginsky, "A Power Management Proxy with a New Best-of-N Bloom Filter Design to Reduce False Positives," *Proc. of the IEEE International Performance Computing and Communications Conference*, pp. 125-133, April 2007

[9] Killer Network Interface Card, http://www.killernic.com/

[10] H. Kim, S. Rixner, and V. Pai, "Network Interface Data Caching", *IEEE Transactions on Computers*, Volume 54, No. 11, pp. 1394-1408, November, 2005.

[11] Microsoft Corporation, "Scalable Networking: Network Protocol Offload -Introducing TCP Chimney", WinHEC Version, Apr. 2004.

[12] ModelSim, http://www.modelsim.com

[13] Network Packet Generator, http://www.wikistc.org/wiki/Network_packet_generator.

[14] NetXen 10 Gigabit Ethernet Controller, http://www.netxen.com.

[15] J. Noguera, R.M. Badia, "Power performance trade off for reconfigurable computing", *Proc. of International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS)*, September 2004, Stockhom.

[16] M. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula, and D. Panda, "Towards NIC-based intrusion detection" *Proc. of the ACM International Conference on Knowledge Discovery and DataMining*, 2003.

[17] D. Petrick, "Analyzing the Xilinx Virtex-II Pro PowerPC with the Dhrystone Benchmark Application", *Technical report*, NASA – Goddard Space Flight Center, August 2007.

[18] P. Purushothaman, M. Navada, R. Subramaniyan, C. Reardon, and A. George, "Power-Proxying on the NIC: A Case Study with the Gnutella File-Sharing Protocol," *Proc. of 31st IEEE Conference on Local Computer Networks (LCN)*, Nov, 2006, Tampa.

[19] RFC 4146, Simple New Mail Notification.

[20] RN2/RN4/RN6 Datasheet, Raptor Networks Technology Inc.

[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] R. Stedman , "Reducing Desktop PC Power Consumption Idle and Sleep modes", *Technical presentation*, Dell Computer Corporation, June 2005

[23] "The Gnutella protocol specification 0.6", http://rfcgnutella.sourceforge.net.

[24] T. Mohsenin, "Design and Evaluation of FPGA-Based Gigabit-Ethernet/PCI Network Interface Card", *Masters Thesis*, Rice University, 2003

[25] Tri-Mode Ethernet MAC v2.1 user guide, April 2005.

[26] Xilinx IP core – Content Addressable Memory data sheet.

[27] Xilinx ISE, http://www.xilinx.com

[28] Xilinx Online Power Estimator, http://www.xilinx.com/cgi-bin/power_tool/power_Virtex2p.

[29] Xilinx, Virtex-II Pro Platform Data Sheet, March 2005.

[30] Xilinx XPower Tutorial, July 2002.