SMART-NICS: POWER PROXYING FOR REDUCED POWER CONSUMPTION IN NETWORK EDGE DEVICES

By

KARTHIKEYAN SABHANATARAJAN

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2008

To my mom, Umarani Sabhanatarajan; and my dad,
Sabhanatarajan Muthiah

ACKNOWLEDGEMENT

I express my sincere gratitude to Dr. Ann Gordon-Ross for guiding and advising this thesis. She played a greatly inspirational role in my graduate studies by providing constant motivation and shaping my graduate research. I would like to thank Dr. Alan D. George for introducing me to this project, providing necessary resources and being my thesis committee member. I would also like to thank Dr. Greg Stitt for being in my thesis committee , and Dr. Ken Christensen for supervising this work. I extend my gratitude to the National Science Foundation (NSF) for supporting this Energy Efficient Internet project.

I thank my parents, Sabhanatarajan Muthiah and Umarani Sabhanatarajan, for their eternal love, sacrifice, affection and support. I thank my grandfather, Thillaisabapathy, for ingraining priceless values of life in me; and my uncle Manivasagam and my aunt Manimozhi for their affection and solace during difficult times. I am grateful to Karthik and Priya for their invaluable support, care and advise during my graduate studies. I also thank Laya for bringing great joy out of me and the rest of my cousins, Monisha, Madhu, Saranya and Vaishnavi.

I express my sincere gratitude to Mark Oden for being an amazing friend, providing timely guidance, helping me in my research and course work and making me feel ever confident. I thank Joe Antoon for reviewing several of my papers, and accompanying me during occasional workouts; Baoke for all his uplifting spirit and constant supply of amusement; Casey for helping me write my first publication; Arjun for getting me food late night; Max for his patience whenever our testbed was intruded; and Mukund for introducing me to VI and getting me started in this project. I convey my special thanks to Ninad Ghodke and Shrinand Javadekar at VMware Inc. for helping me mature my coding abilities far beyond what they were prior to their guidance.

TABLE OF CONTENTS

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

SMART-NICS: POWER PROXYING FOR REDUCED POWER CONSUMPTION IN NETWORK EDGE DEVICES

By

Karthikeyan Sabhanatarajan

December 2008

Chair: Ann Gordon-Ross
Major: Electrical and Computer Engineering

Recent years have seen tremendous Internet growth. This growth is accompanied by increasing power consumption of the Internet as a whole, due in part to the rapid increase in the number of connected edge devices such as desktop PCs. Despite being left idle a considerable amount of time, most of these PCs have their power management features disabled. Consequently, much recent research has focused on reducing power consumption of Internet edge devices. One such method for reducing PC power consumption is by augmenting the Network Interface Card (NIC) with enhanced processing capabilities. These capabilities pave the way for green computing by allowing the PC to transition to a low-power sleep state while the NIC responds to network traffic on behalf of the PC–a technique known as power proxying. However, such a Smart-NIC (SNIC) requires specialized low-power, resource-constrained processing, and architectural features in order to realize such capabilities.

Challenges in realizing a power proxying capable SNIC are analyzed in this work. Packet header and content inspection have been identified as primary requirements in a SNIC to support power proxying. This thesis presents an energy efficient header and content inspection technique suitable for power proxying. The requirements of a header classifier are analyzed and a low-power hardware-based packet classification technique is designed, which, compared to a

software-based packet classification technique, consumes 59% less energy with a 9x speedup. A novel partitioned Ternary CAM (TCAM) based content inspection system is then presented. The proposed technique results in 87% energy savings and a 62% lower energy-delay product than existing non-partitioned router-based techniques, thus making it highly suitable for SNIC-based deployment.

CHAPTER 1
INTRODUCTION

With the rapid increase in the number of edge devices connected to the Internet, the aggregate power consumption of these devices will likely become a major concern in the near future [9]. The most prevalent of these edge devices are consumer desktop computer systems, consuming on average 60 to 95 watts of power and as much as 195 watts in high-end systems [44].

Research estimates that these systems are on average left idle for 75% of the time when powered on [35]. During these idle periods, systems could be powered down to a standby mode to reduce power consumption by up to 80% [9]. 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 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 maintain two-way network connectivity for user applications.

Network Interface Cards (NICs), an important element in any computer design, can be augmented to act as a proxy (or liaison) for the system during standby mode, and still maintain network connectivity by handling a subset of certain application network protocol semantics [22, 35]. 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.

Research indicates that future NICs will have increased network responsibility in order to reduce the processing burden of the CPUs [6, 24, 40, 31]. Complex next generation NICs, also called smart NICs (SNICs), provide an essential platform to support various network related services including power proxying. For a SNIC to provide the capability of power proxying, power proxying rules are required to identify packets that may be appropriately responded to using proxiable protocols. The host system provides these rules to the SNIC immediately prior to entering standby mode. Such a SNIC, upon receiving a packet, uses these rules to identify the packet and either responds appropriately or wakes up the host system.

Packet inspection is a key step in power proxying implementation. Packet inspection is a two stage process. The first stage, known as header inspection, involves inspecting the header of the packet against the power proxying rules. The second stage is known as content inspection and involves the inspection of the contents of the packet for the occurrence of predefined signatures (patterns).

Our study presents a low power packet header classification scheme for reduced power consumption in NICs. An energy efficient content inspection system for SNICs using Ternary CAMs (TCAMs) is also developed and presented. In addition, the suitability and application of this content inspection system to NIC-based intrusion detection systems is explored.

CHAPTER 2
BACKGROUND

This chapter presents the background and research necessary for developing a SNIC capable of power proxying. The trends in the next generation network interfaces are first reviewed. The suitability of network protocols supporting power proxying is then discussed and relevant research is presented. Available SNIC computation and memory resource are then analyzed. Finally, related work with respect to header and content inspection, which form the central theme of this thesis, is reviewed.

2.1 Next Generation SNICs

Next generation SNICs will be delegated more network responsibility in order to reduce CPU processing burdens. Much research has focused on techniques such as offloading TCP protocol processing (TOEs) [19], power proxying [9, 35], and NIC-based data caching [24]. Such reduced CPU processing burden will enable extended CPU sleep opportunities, reduced operating system overhead, increased network throughput and speed, and thus lower overall system power consumption.

Additionally, next generation SNICs offer attractive solutions for distributed network intrusion detection systems (DNIDS), providing potentially greater network security than router-based network intrusion detection systems (NIDS) [1, 40, 31]. Router-based NIDS are rendered ineffective when nodes inside their local network are compromised, such as the case of internal attacks. However, SNIC-based DNIDS can scan both inbound and outbound packets, thereby effectively isolating malicious nodes. Furthermore, SNIC-based DNIDS can exploit node characteristics such as operating system specifics, resulting in more effective, highly optimized malicious packet detection rules. Due to these large potential benefits, NIC-based DNIDS have been the focus of recent research.

According to [35], the system is partitioned into 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.

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 network control to the SNIC. Thus, the PC enters a low-power standby state without disrupting network connectivity. When a packet arrives, the SNIC applies the power proxying rules to the packet. Each power proxying rule consists of two components, i.e, the header and the payload component. Each rule has an action associated with it. When an inbound packet matches any of the rules, the corresponding action is taken and a response is sent accordingly. Rules can be specifically formulated to drop packets, send a predefined response packet, or wake up the system [39].

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) [17] 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, a 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 such as SIP [36] 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 [37]. Upon receiving this packet, the PC can be awoken by the SNIC to download the new message.

2.3 Computational Resources in Network Interfaces

Implementing a power proxying module can be a resource intensive task. In this section, the computational resource limitations of SNICs are explored. Modern NICs contain embedded processors that are largely underutilized, and much current research focuses on exploiting these resources. Friedman et al. [14] 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. [32] proposed and empirically evaluated a novel architecture for NIDS using NICs for commercial Myrinet platforms featuring a 66 MHz LANai with 4 processors and 1 MB of memory. In [38], 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 [23] is a gaming NIC that utilized a 400 MHz

Freescale RISC processor for accelerating game data. 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.

2.4 Packet Classification

Packet classification is one of the key steps in realizing the functionality of power proxying. According to [20, 21], packet classification is defined as the process of comparing packet components against a known rule set. The two major packet components are the header and payload. Thus, packet classification comprises of header and content inspection. Any rule can involve inspecting several header fields or packet contents or both. In this section relevant research in the fields of packet classification is presented.

2.4.1 Packet Classification through Header inspection

Gupta et al. [20, 21] published the first known research that comprehensively dealt with header-based packet classification in routers. According to them, a rule consisted of several dimensions, where each dimension represented a separate header field. Thus, a header rule was an n-dimensional (n-tuple) rule. After examining the nature of the router based rules [20, 21], they discuss different packet classification algorithms. Packet classification algorithms are broadly classified into four classes. The four classes are data structure-, geometry-, heuristic-, and hardware-based algorithms. Each class has distinct advantages and disadvantages and the reader is directed to [20, 21] for further details.

Packet classification complexity is further compounded by matching ranges of numbers in each dimension (TCP source port <1024 and > 256), resulting in header classification being a longest-prefix matching problem. Thus, hardware classification algorithms are preferred over software classification algorithms for speed and throughput reasons. TCAMs are widely used in header classification due to their effectiveness in representing the “don’t care” (*) state and obtaining very high throughput [25, 43, 52]. However, TCAMs do not scale well with increasing number of rules. Several algorithmic [3, 4] and hardware alternatives, such as bloom filters [12] have been proposed. Relevant background and necessary references are also provided in Chapter 3 as the design and implementation of header classification for power proxying is discussed.

2.4.2 Packet Classification through Content Inspection

Content inspection is a pattern matching technique wherein a packet's payload is matched against a set of pre-defined signatures (signature set) to identify malicious packets (for NIDS) or packets of interest (for power proxying). Whereas popular signature sets include the SNORT [42] and ClamAV [10] virus databases for NIDS, to the best of our knowledge there exists no power proxying signature set, and is thus an ongoing research topic.

A content inspection system that can efficiently process packets fast enough to keep up with high link speeds is essential to enable NIDS and power proxying in next generation SNICs. Content inspection is a well researched topic in the context of routers [13, 15, 53] Router-based content inspection can be implemented using either software- or hardware-based techniques. Software techniques employ string matching algorithms such as Boyer-Moore, Aho Corasick, Wu Manber [45], etc. However, due to inherent software inefficiencies when processing large signature sets, software techniques cannot support high link speeds [12].

To increase data processing throughput, specialized hardware-based techniques exploit parallelism using FPGAs [2], TCAMs [15, 53], and specialized data structures such as Bloom Filters [12]. Whereas these techniques are highly suitable for high-end routers with sufficient processing resources, they are not practical enough in terms of price, power consumption, or area for wide-scale deployment in SNICs [53]. However, key processing techniques may be gleaned from router-based content inspection and adapted for SNIC-based techniques.

TCAMs are one of the critical hardware structures that enable fast content inspection, as recognized by Lakshman et al. [53]. Due to the fully associative search ability, TCAMs are populated with signature sets and are capable of performing pattern matching on the order of constant time O(1). For details on TCAM-based pattern matching, the reader is referred to [53].

Whereas this router-based content inspection technique is attractive in terms of high throughput and complete independence from further payload inspection (bloom filter based methods suffer from false positives [12]), this technique suffers from several drawbacks for SNIC-based content inspection. First, TCAMs have large resource requirements, such as power (approximately 10x as compared to a similar speed SRAM [33]) and cost (4x that of SRAM [33]). Secondly, due to necessary signature partitioning, large auxiliary SRAM data structures, on the order of $O(N^2)$, where N is the number of TCAM entries, are necessary for final signature matching. Whereas larger TCAM widths reduce the auxiliary data structure storage requirements, larger widths result in increased “don’t care” bit padding, and thus reduced TCAM resource use and increased TCAM area and power consumption.

Several techniques have been developed to optimize final signature matching. In order to reduce auxiliary data structure storage requirements without reducing TCAM resource use, Gao et al. [15] proposed an alternative architecture, which reduced the auxiliary data structure space complexity to O(N log N). The auxiliary data structure consisted of a secondary TCAM (in addition to the primary TCAM storing the signature set) populated with valid signature address permutations. Valid signature address permutations are the concatenation of the TCAM addresses for each signature in the primary TCAM. Thus, as a payload is searched in the primary TCAM, the hit addresses are concatenated together to form a candidate signature address permutation. Final signature matching extracts candidate signature address permutations from the aggregated TCAM hit addresses and compares those with the valid signature address permutations in the secondary TCAM.

Even though this optimization reduces the area requirement of the auxiliary data structure, the secondary TCAM structure is still very power hungry. An alternative technique [29] implemented a variable width TCAM to improve resource use over a fixed width TCAM. However, this approach suffered from reduced scalability and could only be implemented using FPGAs, which may not provide throughput to sustain high link rates or enough storage capacity for large signature sets.

Dharmapurikar et al. [12] proposed a low power bloom filter-based technique as an alternative to the TCAM-based final signature matching. This method used a separate bloom filter for each unique signature length. While being very energy efficient, this method was able to achieve a throughput of 2.4 Gbps. However, this technique suffered from limited parallelism in the presence of fixed length patterns. Furthermore, inherent false positives placed an additional burden on the already limited processing resources available on NICs.

From the above discussion, there is a clear indication that a more energy efficient content inspection methodology is needed. The contents presented in this section will be expanded further in subsequent chapters as the header and content inspection techniques are developed.

CHAPTER 3
PACKET HEADER INSPECTION UNIT

Inspecting an inbound packet is an important aspect of SNICs. This chapter describes the header inspection process. After providing an architectural overview of a SNIC's header inspection, the chapter then focuses on the nature of the power proxying rules. The software and hardware header inspection techniques are then analyzed.

3.1 Packet Inspection Process

The packet inspection process is defined as a sequence of stages a packet undergoes in order to provide the SNIC with information to determine an appropriate action. A SNIC can send an appropriate response, drop the packet, or wake up the PC depending on the nature of the inbound packet. Figure 3-1 shows the sequence of steps involved to make one of these decisions. When a packet arrives in the SNIC, the packet header is first segregated and serves as input to the header inspection unit. The header inspection unit is primarily responsible for determining if the packet matches with one of the power proxying rules. If there is a successful match with one or more rules, the packet is then forwarded to the next stage. Depending on the nature of the matched rule, the packet is then reassembled and subjected to payload inspection. The content inspection unit is primarily responsible for this payload inspection. If there is no match in the header inspection unit, then the PC is awoken. If there is a header match, the content inspection unit is the next major unit which checks for the occurrence of certain predefined patterns, signatures, inside the packet.

3.2 Packet Header Classifier

For a header inspection unit to function successfully, power proxying rule characteristics must first be identified and operating requirements must be imposed.

3.2.1 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 header 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 header classification rules are specified as address/mask and operator/number(s) pairs [21]. However, power proxying header classifier rules are specified only as 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 the design primarily targets applications running on specific ports, the scope of the operator is limited to equality. If the header 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 packet is forwarded accordingly to the next stage for reassembly and content inspection. Given certain situations, the SNIC may choose not to respond, electing to wake up the PC.

3.2.2 Packet Header Classifier Characteristics and Requirements

The SNIC packet header classifier is similar to a router-based 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 header classifier should be able to sustain link rates of 10/100/1000/10000 Mbps and the latency of the header classification should avoid packet loss.

Fundamentally, SNIC header 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 header 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 header classifiers that have forward or backward redundancy [20].

3.3 Header Inspection Methods

A software-based header classification methodology was designed to quantify the header classification capabilities available on existing unaugmented NICs, and to serve as a comparison for the hardware-based header classification methodology. The simplest software classification algorithm utilizes a binary search algorithm, while the simplest hardware classification implementation utilizes Content Addressable Memories (CAMs) [20, 21].

3.3.1 Software Packet Classification

Existing embedded processors available on commercial NICs were used to implement the software header classifier. The software header classifier was implemented using a binary search algorithm with a complexity of O (log N) to find a matching rule.

The software header 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 header classification begins. The software extracts the required header fields from each packet and passes the fields to the header 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 the hardware implementation, which is elaborated on in the next section.

3.3.2 Hardware Packet Classification

The hardware header classifier is implemented using CAMs. Traditionally, routers use TCAMs for packet classification. Since the header classifier does not demand a longest prefix match, one can implement the classification using basic CAMs, which require less power than TCAMs.

Figure 3-2 shows the architecture of the hardware header classifier. The header processing unit acts as the primary control module and is responsible for extracting the necessary data from the packet headers, supplying the CAMs with the source IP, source port, and destination port. Additionally, the header processing unit maintains classifier state. The header classifier receives the input packets from the MAC core.

The placement of the header classifier with respect to the MAC core is shown in Figure 3-3. 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 [46]. The packet descriptor FIFO is a data structure where the header classifier writes all the information regarding packet classification, including the packet's class and matching signature address, if any. The firmware running on the SNIC processor, also know as power proxy handler, uses this information to determine if the packet has to be conditionally forwarded to the content inspection stage.

The Physical interface (PHY), MAC core and the receiver FIFO constitute the packet's critical path. Because the header classifier lies outside this path, the header classifier does not increase the packet's critical path latency.

The header processing unit is implemented 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 processing unit 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 packets destined for the host are of primary interest.

Next, the header classifier compares TCP and UDP packets against the power proxying rules stored in the CAMs. The source address, source port, and destination port, are partitioned and stored in separate CAMs. For a TCP packet, the header 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 header classifier continue with packet classification. If the header classifier finds no match in the source address CAM, the header classifier interrupts the processing element on the NIC to wake up the PC. Alternatively, if a match in the source address CAM occurs, the header classifier extracts the source port from the header and searches the source port CAM. If a match occurs in the source port CAM, the header 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 header 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 [47]. 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. Results of experiments on packet header classifier are discussed in Chapter 5.

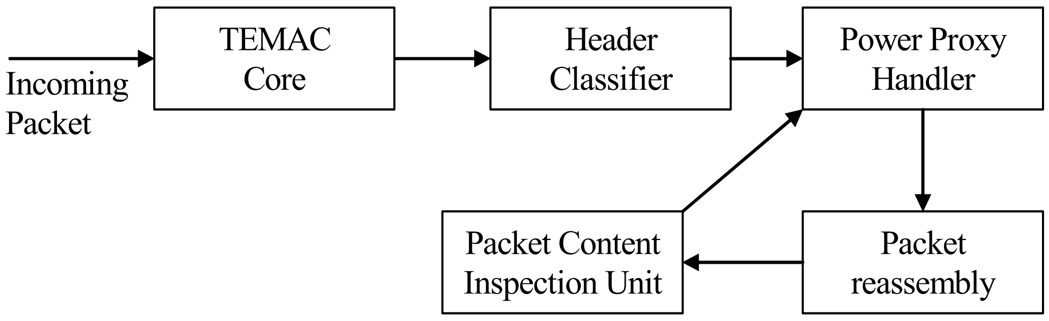


Figure 3-1. Sequence of packet processing steps in a SNIC.

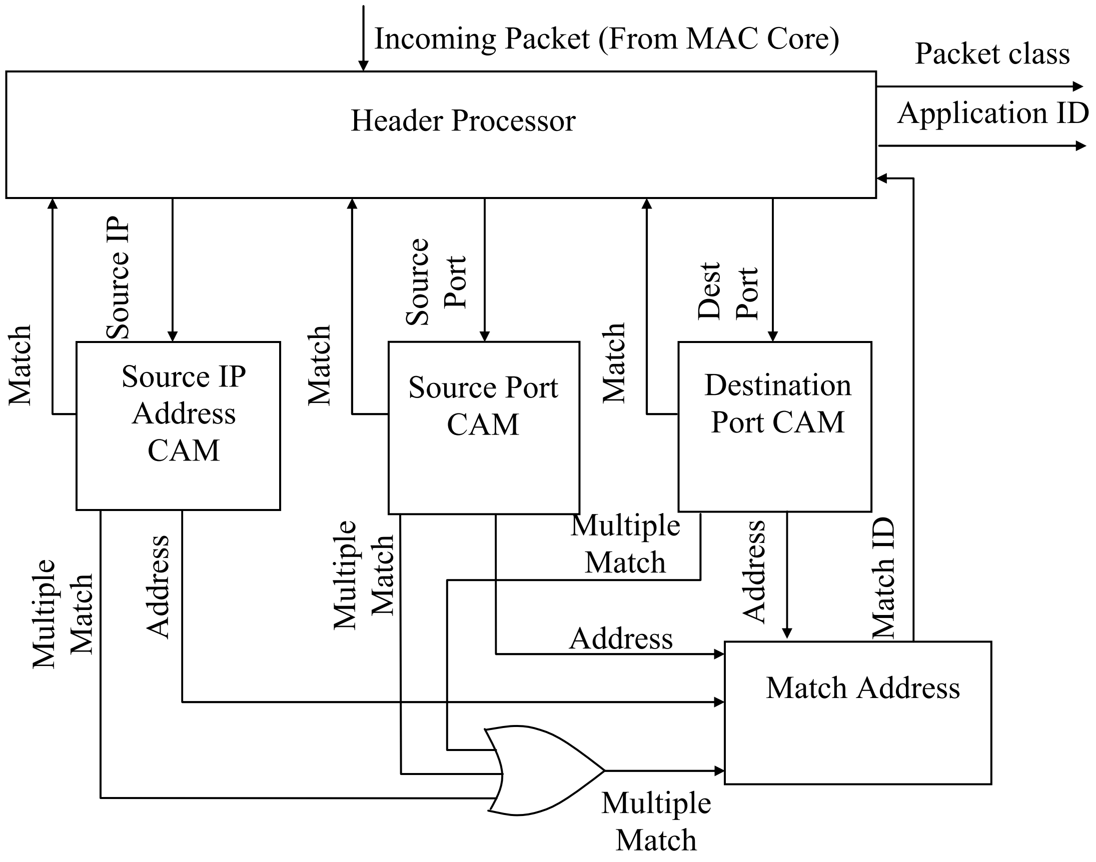


Figure 3-2. Architecture of the CAM-based hardware header classifier.

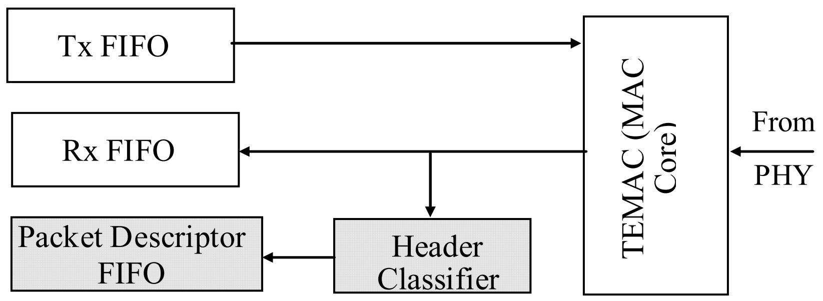


Figure 3-3. Architectural placement of the header classifier. New components are shaded.

CHAPTER 4
CONTENT INSPECTION SYSTEM

We designed and implemented a resource efficient content inspection system for SNICs. The content inspection system searches a packet's payload for the occurrence of known signatures (patterns). Deployment of current content inspection systems discussed in Chapter 2 is impractical due to high resource and energy requirements. This chapter presents a resource efficient content inspection system using TCAMs, which addresses issues specific to deployment in SNICs. Additionally, the content inspection unit is extended to perform DNIDS.

4.1 Traditional TCAM Based Content Inspection Techniques

TCAMs are a popular choice for content inspection [15, 53] due to increased throughput and improved efficiency over other techniques. TCAMs are populated with signature sets and are capable of performing pattern matching on the order of constant time *O(1)*. However, when using TCAMs for content inspection, careful system design considerations must be made. Since signatures are of variable length l (in bytes), the TCAM width w (in bytes) must be equal to the largest signature length L. Thus, all signatures $l < w$ must be padded with $(w-l)*8$ "don't care" bits in order to fill the entire TCAM entry. This method leads to extremely inefficient resource use since signature lengths tend to be highly variable [53].

To improve resource use, TCAM widths are chosen such that $w << L$, and all signatures $l > w$ are partitioned across multiple TCAM entries (*signature partitioning*). Choosing an appropriate TCAM width w is very important, as it affects not only the resource use, but the total number of TCAM entries (depth d) as well. Short patterns are signatures of length $l \leq w$ bytes and these patterns must be padded with $(w-l)*8$ "don't care" bits. Thus, the effective TCAM resource use is reduced for short patterns. Long patterns are signatures of length $l > w$ bytes and these patterns must be partitioned into l/w short patterns. The first $(l/w)-1$ patterns provide full resource use, as only the final partition requires $w-(l \bmod w)*8$ "don't care" bits.

Since every TCAM entry is unique, choosing a smaller width TCAM provides area reduction opportunity in the form of *natural compression* of repetitive patterns. Smaller TCAMs provide more opportunity for pattern repetition in that the probability of repeated patterns increases. However, smaller TCAM widths increases complexity of pattern matching, as additional data structures are required to decode shared entries.

When partitioning long patterns, the first partition is denoted as the prefix pattern and the remaining partitions are denoted as suffix patterns. Figure 4-1 shows the prefix and suffix patterns for a sample long pattern signature given a TCAM width of 4 bytes (each character represents an arbitrary byte).

The long and short patterns are stored as entries in a single TCAM and the TCAM entries are compared to incoming payloads. *Payload examination* occurs by streaming the payload contents through a *w*-byte inspection window. Initially this inspection window contains the first *w* bytes of the payload. For each subsequent clock cycle, the payload contents are left-shifted by one byte in order to inspect the next *w*-byte inspection window. Thus a payload of *X*-bytes contains *X* inspection windows, and the TCAM is searched for each of these windows. Furthermore, since a signature is scattered across $\left| l/w \right|$ TCAM locations, a TCAM match implies that the payload only matches with a portion of a signature. A *final signature matching* step is required to ensure that a payload matches with a complete signature. To assist in final signature matching, an auxiliary SRAM data structure aggregates TCAM hit address information during payload examination [53].

As noted previously, this traditional router-based TCAM technique suffers from problems such as high power, cost, and large auxiliary data structure requirements. A content inspection technique that is more amenable to limited resource SNICs is architected in this chapter by extending TCAM-based techniques [15, 53], reducing both energy consumption and the energy delay product (EDP). A method by which the single TCAM is partitioned into a prefix TCAM and a suffix TCAM is developed. This partitioned technique reduces TCAM switching activity, with little to no area increase, and thereby reduces system energy consumption. A caching technique to further reduce energy consumption is introduced, motivated by a signature caching technique that exploits network traffic locality [24]. This technique assumes the NIC architecture proposed in [40], which includes low resource mechanisms for packet reassembly and check summing.

4.2 SNIC-Based Content Inspection System

4.2.1 Definitions

The distinguishing features of the proposed architecture include: (1) the segregation of the prefix and suffix patterns into two separate TCAMs, the Prefix TCAM (P_TCAM) and the Suffix TCAM (S_TCAM), respectively; (2) the introduction of a suffix cache, which stores a subset of the S_TCAM entries; and (3) the usage of bloom filters for the auxiliary data structure. Previous methods used one large TCAM to store both prefix and suffix patterns. Storing all patterns in a single TCAM has the disadvantage of triggering unnecessary TCAM switching activity. For long patterns ($w<l$), suffixes are of interest only after a prefix match. Thus prefix and suffix segregation isolates prefix pattern matching to a smaller P_TCAM, and the larger S_TCAM is selectively enabled after an associated P_TCAM match. Additionally, identical prefix and suffix patterns are defined as *alias addresses.*

Every signature is expressed as a valid signature address permutation representing the addresses at which each signature's partitions are stored. This permutation may be the concatenation of a P_TCAM address and several S_TCAM addresses (in the case of a long pattern with no alias addresses), an arbitrary number of P_TCAM and S_TCAM addresses (in the case of a long pattern with alias addresses, wherein the first address will always be a P_TCAM address), or just a single P_TCAM address (in the case of a short pattern).

Given a signature partitioned in $\left\lceil {}^{l}\!/_{w} \right\rceil$ patterns, a *concluding* pattern is defined as the final partition $\left\lceil {}^{l}\!/_{w} \right\rceil$ (which may be a prefix pattern for a short pattern or an alias address or a suffix pattern for a long pattern). This pattern marks the final address of a valid signature address permutation. Accordingly, all partitions $1 \le p < \left\lceil {}^{l}\!/_{w} \right\rceil$ are defined as *intermediate* patterns.

4.2.2 Architecture

Figure 4-2 shows the proposed content inspection architecture, consisting of three signature storage units: the P_TCAM, suffix cache, and the S_TCAM. The inspection window size of 4 bytes is assumed and the signature storage units are populated using the sample signature from Figure 4-1. The suffix cache is a small TCAM that stores the most recently used subset of the S_TCAM entries. Since valid signature address permutations only contain P_TCAM and S_TCAM addresses, each suffix cache entry also stores the corresponding S_TCAM address. From Figure 4-1 one can see that a match of EFG* implies a match of EFGH but the converse does not hold true. This property is defined as *mutual inclusion* [15] and must be considered during caching. To avoid inconsistencies due to mutual inclusion, only S_TCAM entries that are exactly *w* bytes are cached (entries without any "don't care" padding bits).

The presence of a payload reconstruction unit is assumed (not shown in Figure 4-2) and this unit aggregates incoming network packets to reconstruct complete payloads. The complete payload is provided to the content inspection architecture. On each clock cycle, the payload is byte-wise left-shifted through a w-byte inspection window. The current w-byte inspection window contents are provided as input to the signature storage units. However, whereas the P_TCAM is searched each cycle by default, the suffix cache and the S_TCAM are selectively searched. The suffix cache is enabled after an intermediate P_TCAM hit and the S_TCAM is enabled after a suffix cache miss.

Since the payload is byte-shifted, but the addresses in the valid signature address permutations represent *w*-byte windows, the suffix cache and S_TCAM only need to be activated *w* clock cycles after a prefix or intermediate pattern hit (in any signature storage unit). The *activator* monitors all signature storage units and upon a prefix or intermediate pattern hit, sets the 0th bit of the *enable buffer* to '1', otherwise '0'. The enable buffer is a *w*-bit wide structure and is right-shifted each clock cycle. The shifted out bit serves as input to the *enabler*, thus signaling a suffix search *w* clock cycles after an intermediate pattern hit.

When the enabler receives a '1' bit input from the enable buffer, the suffix cache is enabled. Upon a suffix cache hit, the payload stream is left-shifted, and the next *w*-byte inspection window is processed. However, on a suffix cache miss, the S_TCAM must be searched on the next clock cycle for the same *w*-byte window. In order to reprocess the current inspection window, the enabler asserts a *pause* signal which effectively halts payload window and enable buffer shifting so that the same window can be reexamined. During this time, the cache controller (*$ Ctr*) orchestrates the suffix cache replacement policy. Since the least recently used (LRU) replacement policy overhead can be prohibitive for large associativities, a random replacement policy is used, which is shown to have similar performance as LRU for large associativities [16]. It should be noted that the introduction of caching stalls the system by one cycle during the cache miss and thus leads to reduced throughput. However, results shown in Chapter 5 indicate the overhead is negligible.

The *retirement buffer* stores candidate signature address permutations, and serves as input to the final signature matching step (the technique proposed by [15] is extended to address partitioning specifics). Each of the entries record information about TCAM hit status for each clock cycle, in the form of a TCAM hit address and associated descriptor bits. The descriptor bit designates if the entry is a P_TCAM ("11") address, an S_TCAM ("01") address, or if there was no hit ("00").

On each clock cycle, the retirement buffer is left-shifted and the *contention resolution* module pushes a new entry onto the right side of the buffer. If there is no hit in any TCAM, the new entrie's hit address is set to NULL (Ø) and the descriptor bits to "00". If there is a concluding P_TCAM hit (and a suffix miss), the prefix represents a short pattern, and thus this single hit indicates a complete signature match and there is no final signature match checking required, thus Ø is pushed onto the retirement buffer. In the case of an intermediate prefix or suffix hit, the associated hit address is pushed onto the retirement buffer, and the descriptor bits are set to "11" or "01", respectively. If there is both a prefix and a suffix hit (in the case of alias addresses) and both hits are intermediate patterns, the contention resolution module ensures that the P_TCAM address is pushed on to the retirement buffer, and the descriptor bits are set to "11". This alias address resolution technique is necessary since the intermediate pattern may indicate the beginning of a signature match.

Since the retirement buffer space is bounded, retirement logic (not shown in Figure 4-2) monitors the left most retirement buffer entry, the sentry position. When the sentry position's descriptor bits are '11' (indicating the start of a potential signature match), the retirement logic extracts all candidate signature match permutations (all the entries that are separated *w* bytes (*w* buffer entries) from each other), terminating on a Ø position. The candidate signature match permutations are dispatched to the final signature matching unit. The final signature matching unit uses hashing structures such as parallel bloom filters [7] to compare candidate and valid signature match permutations. Software methods can also be used as a substitute for final signature matching. Elaborations of such techniques are beyond the scope of this thesis, and optimization of this step is the left as future work.

4.3 Mathematical Model

In this section, the resource requirements for the proposed architecture are analyzed and a model for energy expenditure is developed. To describe the total TCAM (both prefix and suffix) and retirement buffer resource requirements, *w* is defined as the width of the TCAM in bytes, *P* as the depth of the P_TCAM, *S* as the depth of the S_TCAM, and *L* as the maximum signature length. Both *P* and *S* are highly dependent on the natural compression present in a signature set, but in the worst case (no natural compression):

$$P = T;\; S = \sum_{i=1}^{T} \left\lceil \frac{l_i}{w} \right\rceil \tag{4-1}$$

where *T* is the signature set size. The total TCAM resource requirements is *w*N* bytes where *N=P+S*. Additionally, two bits are required to identify each TCAM entry as either a concluding or intermediate pattern or both, requiring additional *2*N* bits. The retirement buffer resource requirements are similar to [15]:

$$\left(1+w\times\left(\frac{L}{w}-1\right)\right)\times\left(\log_2(Max(P,S))+2\right)\text{ bits} \quad (4\text{-}2)$$

The assumption is that the size of the cache *C* contributes very little to the total resource requirements as *C*<<*N*. Since a random replacement policy is used, there is no additional area overhead other than a small cache controller. All TCAM expenditures can be aggregated into the total energy expended:

Total_Energy = Num_P_TCAM_Accesses * P_TCAM_EPA
+Num_Intermediate_Accesses * Cache_EPA
+ Num_Cache_Misses * S_TCAM_EPA
+ Num_Cache_Misses * Cache_Write_EPA
+ Num_S_TCAM_Accesses * S_TCAM_EPA (4-3)

Thus, average energy per access (EPA) is defined as the energy expended for a single *w*-byte window search:

EPA = Total_Energy / Total_Accesses (4-4)

$$Total\_Accesses=\sum_{i=1}^{X}P_i \quad (4\text{-}5)$$

where *X* is the total number of packets processed and P_i is the payload length of packet *i*.

Sample Signature:	

Prefix Pattern:
`A B C D`
	Suffix Patterns:
	`E F G H`
	`A B C D`
	`J K L M`
	`E F G *`

Figure 4-1. Prefix and suffix patterns for a sample signature for a TCAM width w=4. (* = don't care) Each character represents an arbitrary byte.

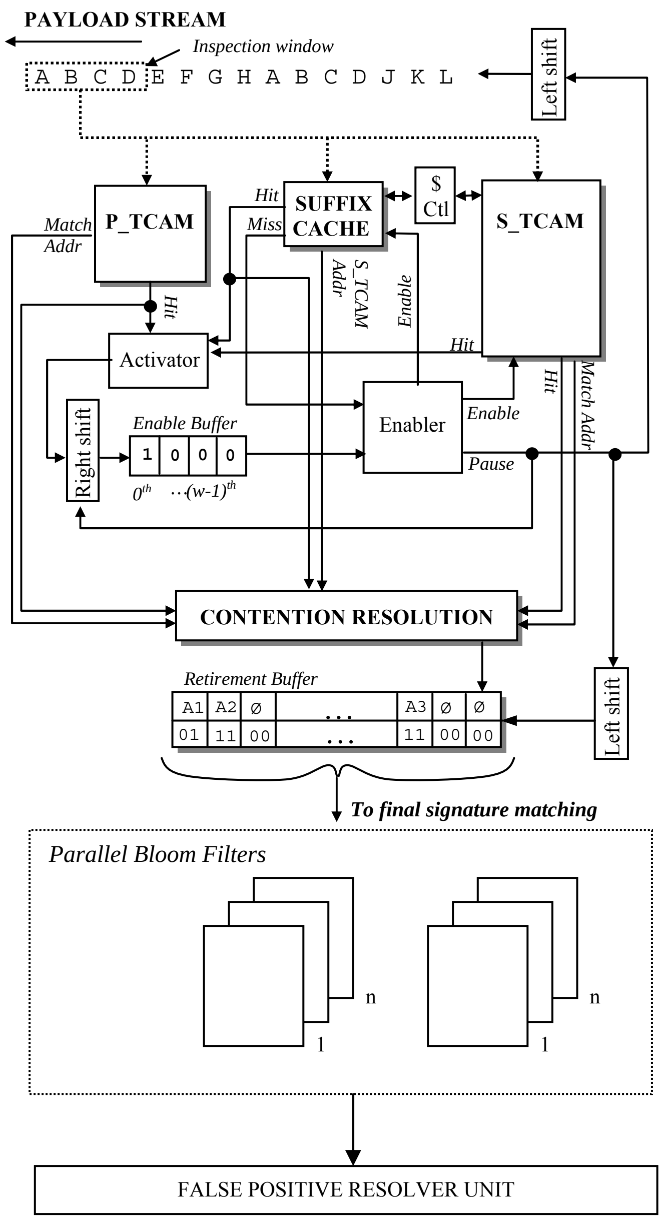


Figure 4-2. Partitioned TCAM system for SNIC-based content inspection.

CHAPTER 5
RESULTS AND EVALUATION

Header and content inspection techniques for deployment in SNICs have been proposed in this thesis. This chapter evaluates the suitability of these techniques for wide scale deployment in the SNICs. Experiments on the proposed techniques are discussed and their results analyzed.

5.1 Header Inspection System Evaluation

In Chapter 2, hardware and software based techniques for header inspection were proposed. This subsection details several experiments performed to compare the hardware and software classification techniques in terms of classification speed and dynamic power dissipation.

5.1.1 Experimental Setup

A software header classifier using binary search is implemented in the firmware of anembedded PowerPC 405 processor on the RiceNIC platform [41]. The RiceNIC is a programmable NIC that incorporates an FPGA and two embedded PowerPCs. The RiceNIC implementation clocks the PowerPC at 300 MHz and the processor bus at 100 MHz. In addition, the PowerPC implementation is modified to operate at 100 MHz in order to observe the performance of the packet classifier at lower clock frequencies, representing low-end NICs.

The experimental setup for the software header classifier consists of two PCs, one emulating a network switch and the other equipped with the RiceNIC board. Using the packet generation tool NPG [28], the PC emulating the switch injects minimum sized packets to the RiceNIC equipped PC. The RiceNIC is instrumented to record header classification time statistics.

The hardware header classifier is prototyped on the Xilinx Virtex-II Pro FPGA XC2VP20 and used Verilog HDL and Xilinx IP cores to generate the CAMs with block memory. The system is developed and simulated, implementing the Xilinx TEMAC core [46], using Xilinx ISE 9.1 [48] and ModelSim XE [27]. The system is designed to support the three link rates of 10/100/1000 Mbps, with corresponding clock frequencies of 1.25, 12.5, and 125 MHz, respectively. Next, the hardware system is synthesized and time-constraint based placement and routing with Xilinx XST is performed. The system is subjected to heavy timing simulations (post-place and route timing simulation) using both the ISE simulator and Modelsim XE. Xilinx XPower [51] is used for power estimation [5].

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

5.1.2 Header Classifier Speed

The primary goal of the header 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 header classifier, worst-case header classification time occurs when the matching rule is the last rule checked. For the hardware header classifier, worst-case header 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 5-1 shows the worst-case header 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

The variation of the worst-case header classification time for the TCP/UDP packets with varying rule set sizes is shown in Figure 5-2. The hardware header classification time is constant for any number of rules while the software packet classification time increases logarithmically.

The throughput of hardware and software header classifiers are defined 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 5-3 shows the obtainable worst-case throughput for both header 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 dropped packets. The embedded processing element's clock rate is estimated to 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 header 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 header 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 the hardware classification technique much closer to meeting 10 Gpbs link rates (and in some rule cases, does meet the requirements), it is projected that one can optimize the hardware to maintain a link rate of 10 Gpbs with minimal added power overhead.

The speedup obtained with hardware classification versus software classification is shown in Figure 5-4. 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.

5.1.3 Power Consumption

The power consumption of the hardware design is estimated using Xilinx XPower. The embedded PowerPC core in the Virtex-II Pro consumes 0.9mW/MHz at an ambient temperature of 25^{o}C [50]. The online power estimation tool [49] is used to obtain the power consumption of the PowerPC system operating at 100 MHz and 300 MHz with the bus interface clocked at 100 MHz and is found to be 100 mW and 280 mW respectively. However, [34] reveals a more realistic power estimation that also accounts for the bus power dissipation. Thus, the PowerPC consumes 259.5 mW and 441 mW when clocked at 100 MHz and 300 MHz respectively. These numbers are in close agreement with [30], which also estimates the idle power of the PowerPC to be 50 mW. All power estimations are obtained at an ambient temperature of 25^{o}C.

The highest measured power consumption of the hardware header classifier is 180 mW when it processes a TCP packet with 100 rules. The software header classifier consumes between 2.4x and 2.9x more power than the hardware header classifier. 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 header classifier.

Figure 5-5 shows the variation of the average hardware power consumption for various packet classes across different link rates for 100 rules. An exponential increase in power consumption can be traced 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.

5.1.4 Hardware Operating Frequency and Scalability

A maximum frequency of 177.17 MHz is obtained 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, which transmits data in units of 64 bits. The prototype meets this requirement for 20 rules and with larger FPGAs the hardware can easily meet the 10 Gbps frequency requirement for an implementation with 100 rules.

5.2 Content Inspection System Evaluation

Having evaluated the header inspection system, this section focuses on the analysis of the TCAM-based content inspection system. The TCAMs first need to be populated with signature sets. The signature length distribution of two popular signature sets, i.e SNORT [42] and ClamAV [10], are analyzed. The impact of partitioning the TCAM (without suffix caching) is then analyzed with respect to area, energy consumption, and the energy-delay product (EDP) [18]. Next, popular NIDS trace benchmarks are simulated to determine average energy savings and are compared with the unpartitioned TCAM approach modeled using the same environment. A suffix cache is finally introduced and its effects are analyzed.

5.2.1 Experimental Setup

A custom C-based simulator is used to model the content inspection system. For a given TCAM width *w,* the SNORT [42] and ClamAV [10] signature sets are populated in the TCAM structures accordingly. For the simulation, popular NIDS benchmark traces from the MIT Lincoln Laboratory (MIT-LL) [26] and the “capture-the-flag” contest for the DEFCON festival [8] are used.

During a trace pre-analysis step, incoming fragmented packets are reassembled and the payload of the reassembled packets are extracted and passed to the simulator. The simulator behaviorally simulates the proposed architecture (excepting the auxiliary structures), recording several statistics such as total number of accesses to each TCAM and total number of intermediate and concluding prefix and suffix hits for postmortem analysis. To analyze the effects of the suffix cache, the S_TCAM access trace is saved to a trace file for future analysis by a cache simulator.

TCAM energy consumption is obtained using the TCAM modeling tool developed by Agarwal et al. [1]. This tool provides search time and energy per access verses width, number of entries, and the fabrication technology, which is assumed to be 130 nm. These measurements are combined with the mathematical models (Chapter 4) to obtain the resource usage and energy consumption.

5.2.2 Signature Length Distribution Analysis

To assist in appropriate TCAM width *w* determination and avoid reduced resource use due to excessive “don’t care” bit padding, signature length distribution is first analyzed. Figure 5-6 shows the cumulative signature length distribution for SNORT v2.4 and v2.8, and the ClamAV signature sets. Primarily, SNORT signatures are short patterns, with 70% of the signatures less than 4 bytes long, and 99.8% of the signatures less than 100 bytes long. ClamAV shows a different distribution, with 72% of the signatures between 30 bytes and 100 bytes long. This suggests that smaller TCAM widths are more suitable for SNORT signature patterns compared to ClamAV patterns. The graphs conform to the findings in [53] showing that future SNORT pattern lengths are becoming increasingly smaller and are more complex as these smaller patterns are distributed across the packet.

Since SNORT v2.4 and v2.8 show similar trends (and these same trends are observed for all experimental results), only SNORT v2.8 experimental results are presented.

5.2.3 Effects of TCAM Partitioning on Size, Energy, and EDP

Partitioning circumvents natural compression and results in an increase in the cumulative TCAM space. For example, given *w=4* the signature "ABCDEFGHABCD" can be represented in a single TCAM using only two entries: ABCD and EFGH. However, partitioning the signature across a P_TCAM and an S_TCAM requires three total entries: ABCD in the P_TCAM and EFGH and ABCD in the S_TCAM. Thus, the impact on total area due to TCAM partitioning is first analyzed.

Partitioning effects on TCAM size in KBytes for the SNORT v2.8 and ClamAV signature sets verses varying TCAM widths are shown in Figure 5-7 (A) and Figure 5-7 (B), respectively. These figures show P_TCAM and S_TCAM sizes, as well as the total combined size of these two TCAMs (combined TCAMs) compared to the non-partitioned TCAM system. The results show negligible natural compression loss, with the largest area overhead increase due to partitioning being only 4% for the smallest width.

Energy per access normalized to the non-partitioned TCAM system for the P_TCAM and S_TCAM individually and both TCAMs combined (combined TCAMs) for the SNORT v2.8 and ClamAV signature sets are shown in Figure 5-8 (A) and Figure 5-8 (B), respectively. For SNORT, Figure 5-8 (A) shows that for the best case scenario (all P_TCAM accesses miss), energy consumption can be reduced by 74% to 40% compared to a non-partitioned TCAM system for TCAM widths ranging from 4 to 16 bytes, respectively. For ClamAV, Figure 5-8 (B) shows that for the best case scenario, energy consumption can be reduced by 93% to 78% compared to a non-partitioned TCAM system for TCAM widths ranging from 4 to 16 bytes, respectively. In the worst case scenario (full activity in both the P_TCAM and S_TCAM), the energy consumption per access is nearly identical to the non-partitioned TCAM system, except for a TCAM width of 4 bytes, where energy is increased by 5% and 1% for SNORT and ClamAV, respectively. However, simulations using popular benchmark traces in next section show that the worst case scenario rarely occurs.

Even though the partitioned TCAM system performs similar to that of a non-partitioned TCAM system in terms of total size and worst case energy per access, the largest advantage of the partitioned system is the reduction in the EDP. Figure 5-9 shows the percentage reduction in the EDP verses TCAM width for the SNORT v2.8 and ClamAV signature sets. The results reveal EDP reduction as high as 62% for both signature sets. This reinforces the fact that our partitioned TCAM system is both energy and throughput aware compared to a non-partitioned TCAM system, which is predominantly throughput aware.

5.2.4 Energy Savings from Partitioning with Real-Time Network Traces

Figure 5-10 shows the energy reduction for a partitioned TCAM system compared to a non-partitioned TCAM system for two MIT-LL and DEFCON traces for both signature sets. Energy savings range from 6% to 69% and 6% to 87% for SNORT and ClamAV, respectively. Both signature sets reveal similar energy reduction trends with smaller TCAM widths revealing larger energy reductions compared to larger TCAMs widths, as larger widths result in much more expensive TCAM accesses and an increase in “don’t care” bits. Furthermore, ClamAV patterns exhibit more energy savings for a TCAM width 8 due to a drastic reduction in S_TCAM accesses, suggesting that the traces contain predominantly short patterns.

5.2.5 Network Trace Locality and Caching

First, network trace locality is analyzed in order to motivate caching benefits. Figure 5-11 shows matching SNORT signature identification (ID) number verses ordered incoming malicious packets for the MIT-LL traces. As the figure shows, only a few unique signatures match, and matched signatures exhibit significant temporal locality.

Next, the distribution of TCAM accesses between the P_TCAM and the S_TCAM is analyzed to reveal further caching potential. Figure 5-12 shows the percentage of S_TCAM accesses for the partitioned TCAM system verses varying TCAM widths for SNORT and ClamAV signature sets using the MIT-LL and DEFCON traces. The figure shows that smaller TCAM widths generate more suffix accesses and hence provide better opportunity for caching. This is promising given that Figure 5-10 shows the greatest energy reduction for small TCAM widths. For all cases except SNORT v2.8 with the DEFCON input trace, S_TCAM access percentage drops below 2% for widths greater than 8 bytes. It should be pointed out that the percentage is largely dependent on the nature of traces and the signature sets used.

Impacts of caching for a TCAM width of 4 bytes is analyzed, as this width provides the greatest number of S_TCAM accesses. Figure 5-13 shows the variations in cache hit rate verses cache size in number of entries. Hit rates range from 28% to 88% with a cache size of only 40 to 60 entries, with very little increased benefit for larger cache sizes. A cache containing 40 to 60 entries represents only 0.002% to 0.004%, respectively, of the S_TCAM entries.

Figure 5-14 shows energy reduction for a partitioned TCAM system with a suffix cache compared to a partitioned TCAM system with no suffix cache. Figure 5-15 analyzes the throughput reduction due to cache misses.

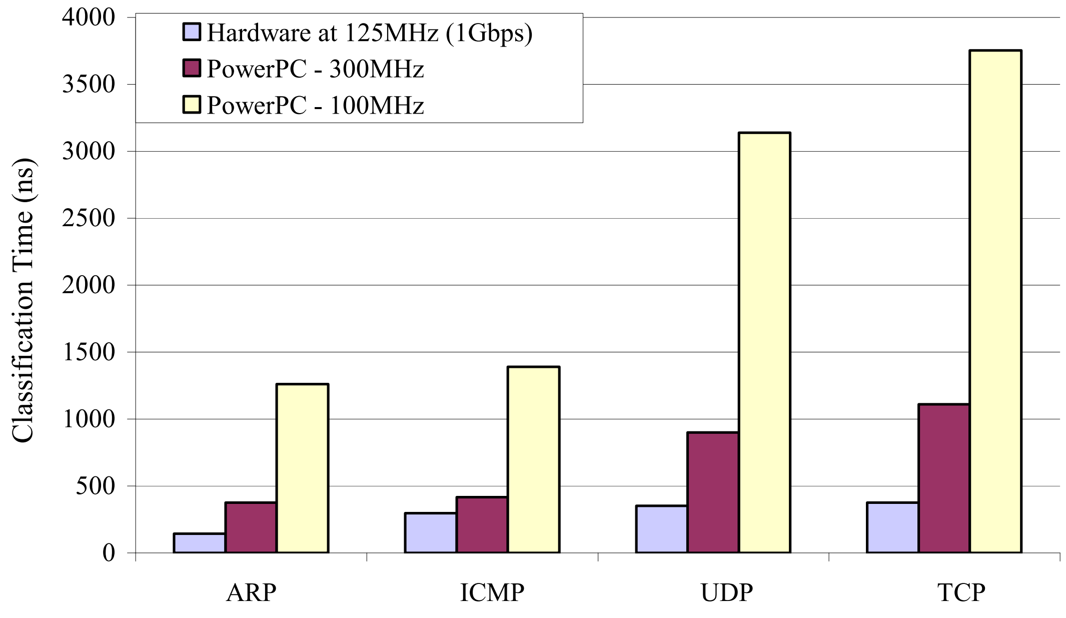


Figure 5-1. Worst-case header classification time for each protocol class with a power proxy rule set of 100 rules.

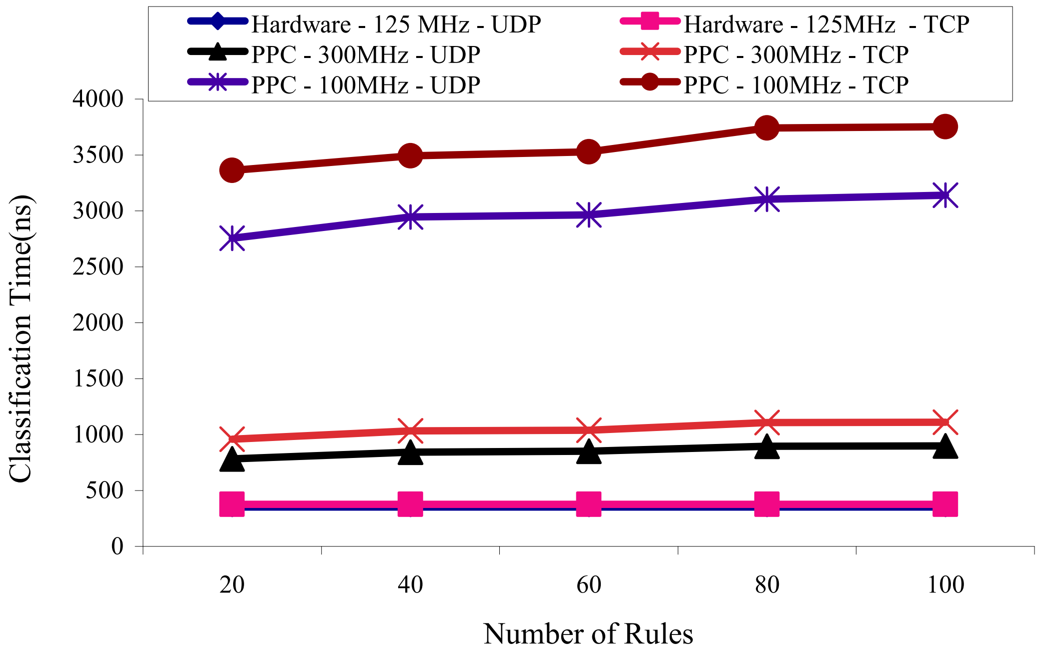


Figure 5-2. Worst-case header classification time for TCP and UDP traffic vs. number of power proxying rules. (Both hardware classification times overlap on the bottom line).

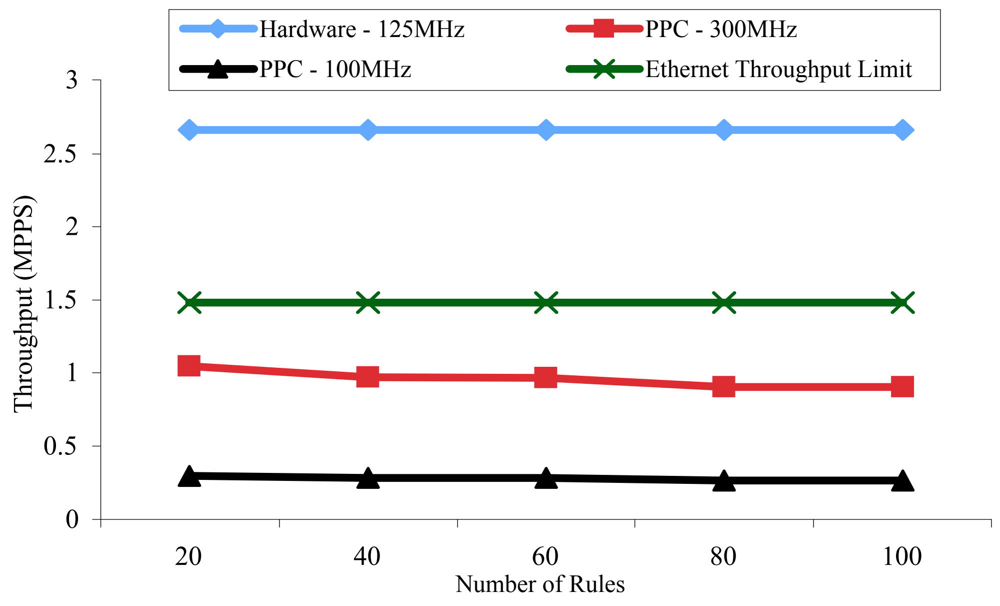


Figure 5-3. Obtainable throughput in MPPS for hardware and software packet classifiers vs. number of rules for TCP traffic.

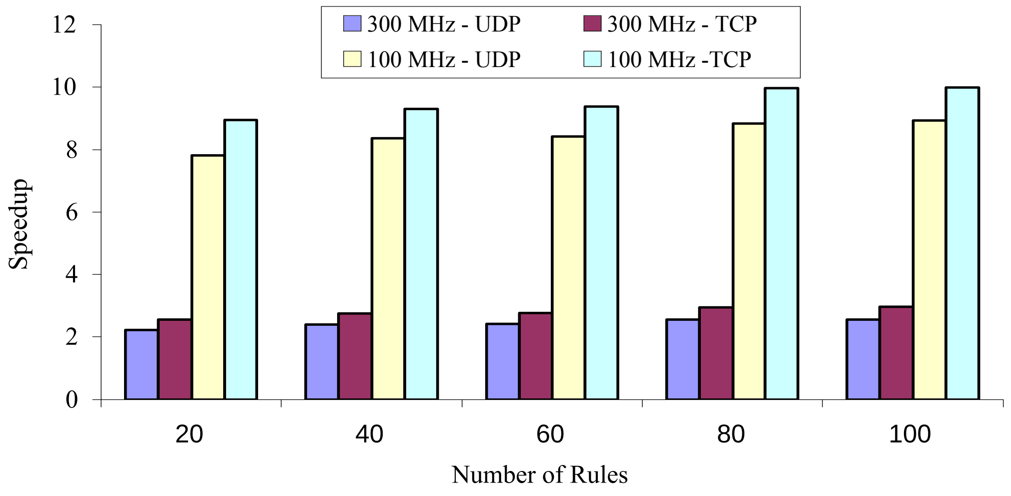


Figure 5-4. Speedup obtained by using a hardware classifier compared to a software classifier for varying number of rules.

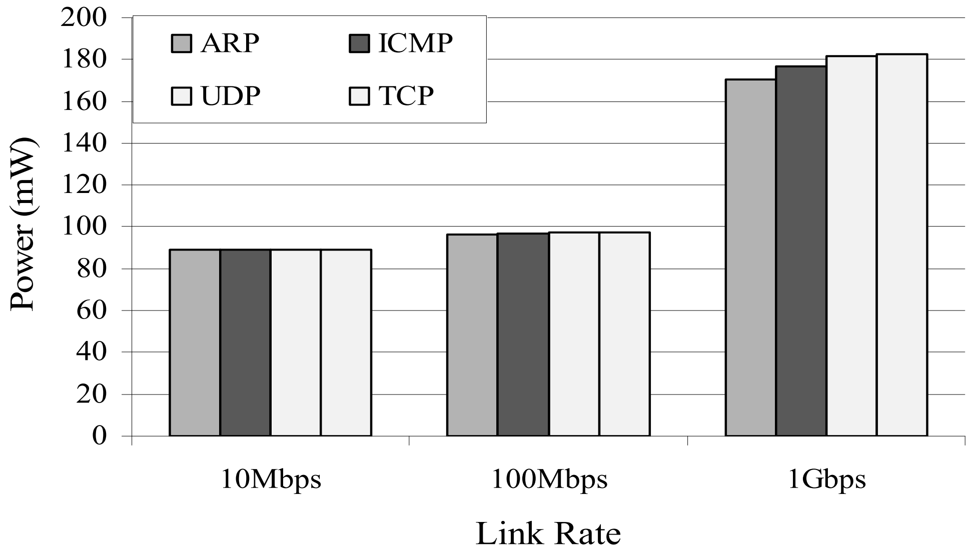


Figure 5-5. Hardware power consumption vs. link rate for 100 rules.

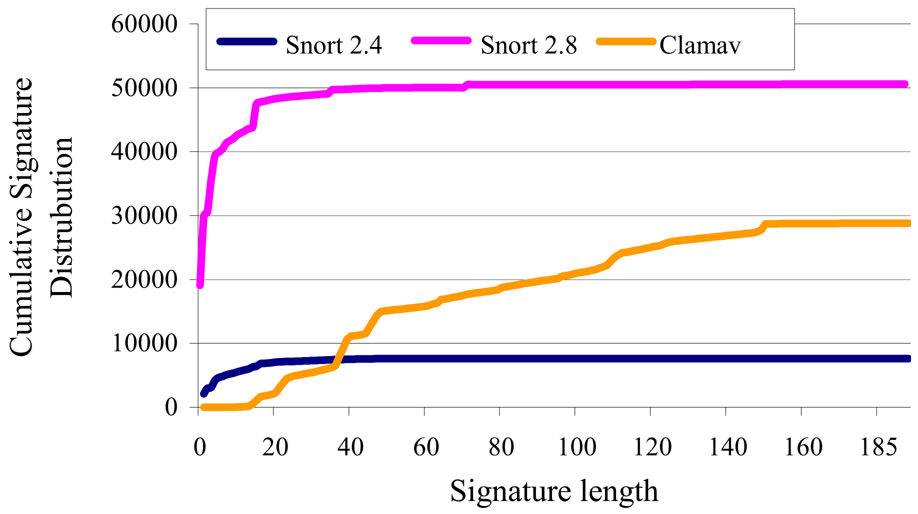
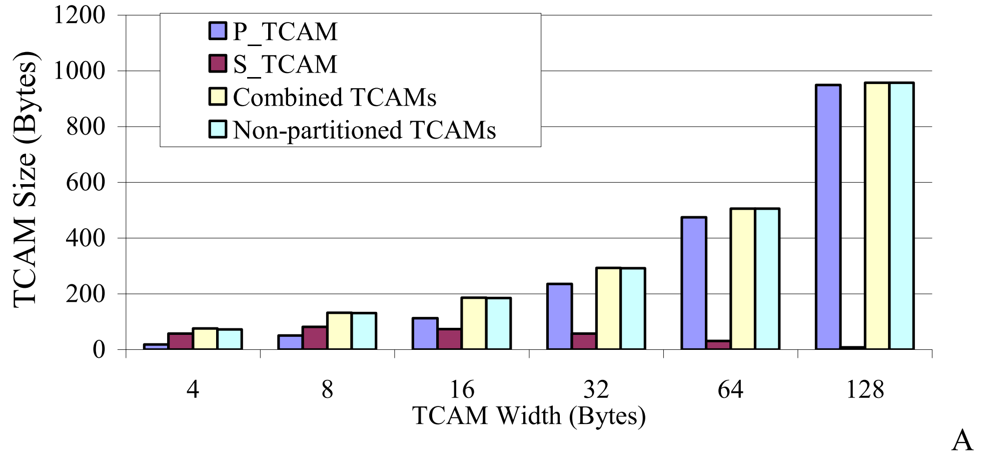


Figure 5-6. Cumulative number of rules (distribution) for increasing signature lengths for Snort and ClamAV signature sets.



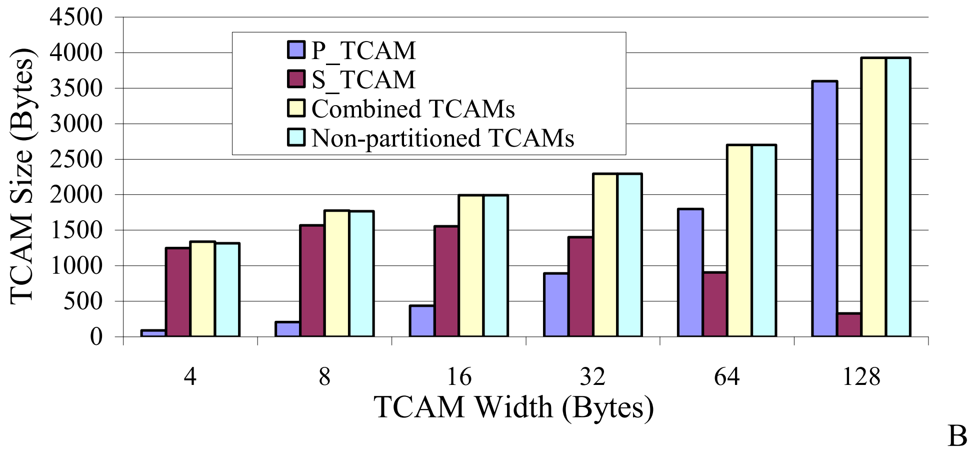
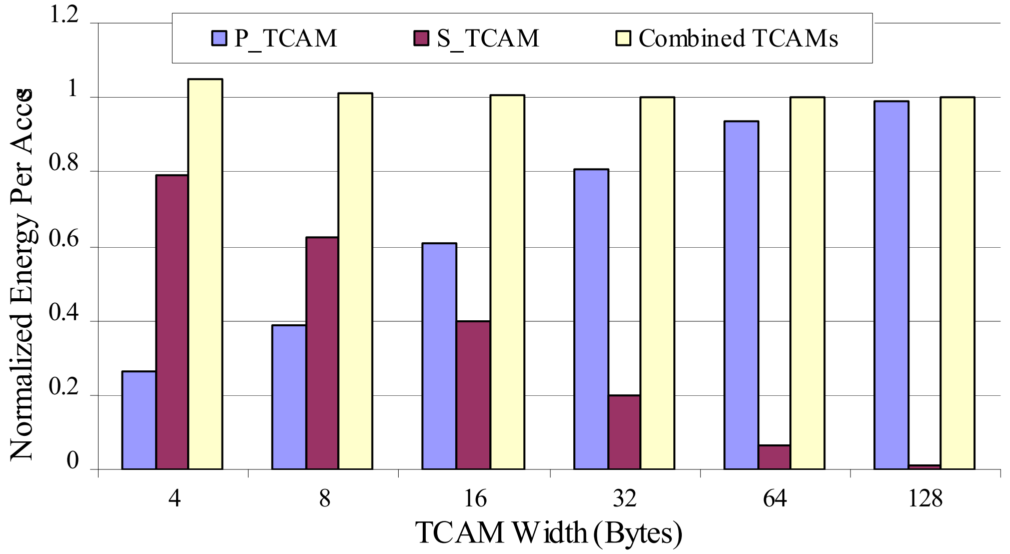
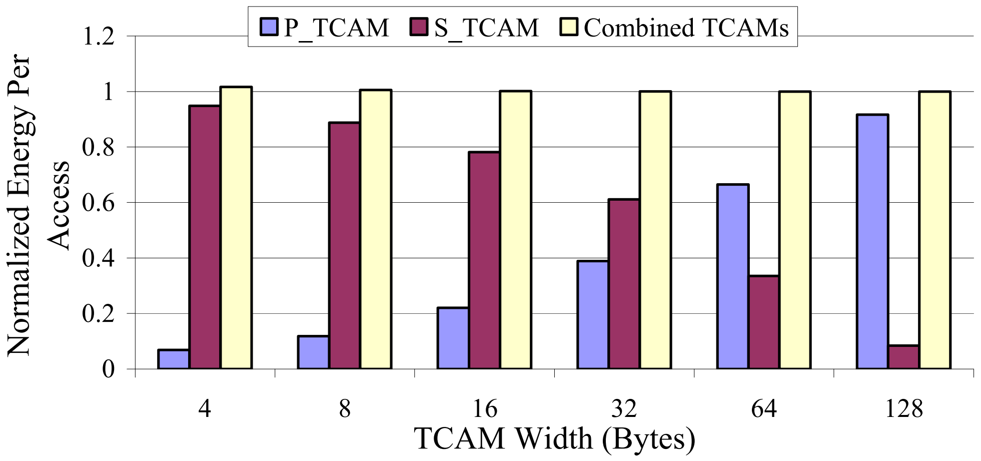


Figure 5-7. Variation of TCAM size verses TCAM width for A) SNORT v2.8 and B) ClamAV signature sets for the P_TCAM and S_TCAM individually, the P_TCAM and S_TCAM combined (Combined TCAMs), and the non-partitioned TCAM system.



A



B

Figure 5-8. Energy per access normalized to a non-partitioned TCAM system verses TCAM width for the A) Snort v2.8 and B) ClamAV signature sets for the P_TCAM and S_TCAM individually as well as the P_TCAM and S_TCAM combined (Combined TCAMs).

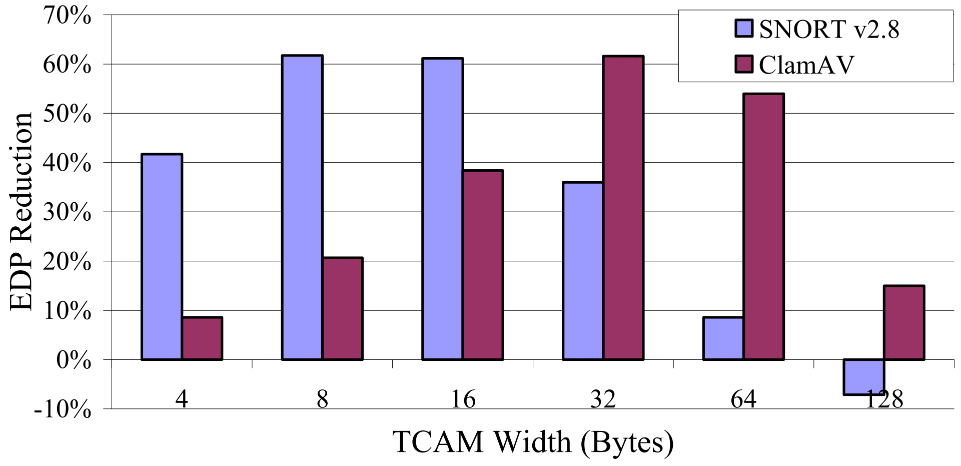


Figure 5-9. Percentage reduction in the energy-delay product (EDP) for a partitioned TCAM system compared to a non-partitioned TCAM system verses TCAM width.

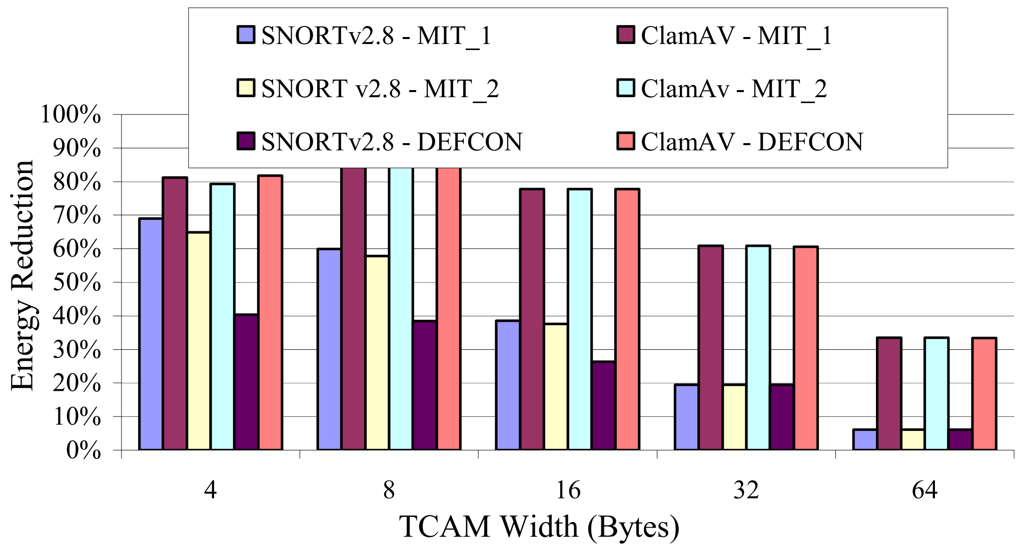


Figure 5-10. Energy reduction for a partitioned system compared to a non-partitioned system verses TCAM width for real-time traffic traces.

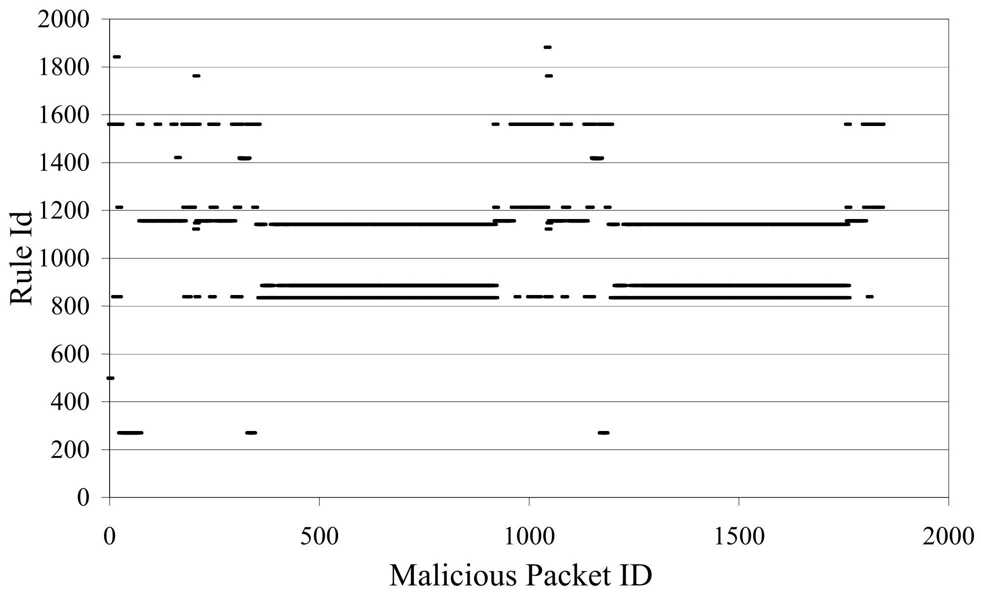


Figure 5-11. Signature access locality (SNORT rule ID verses time represented by the malicious packet ID) as observed by an edge node under attack.

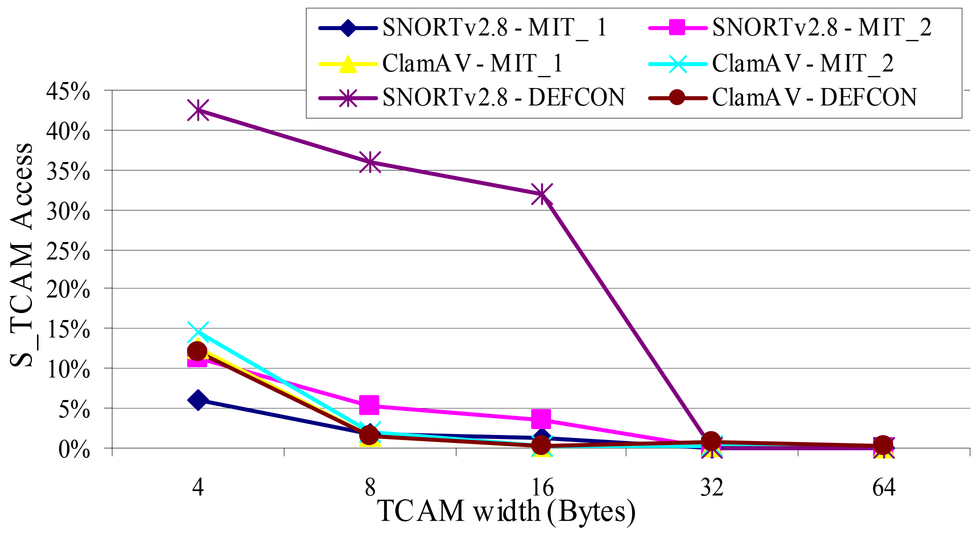


Figure 5-12. Percentage of S_TCAM accesses for various TCAM widths populated by SNORT v2.8 and ClamAV signature sets.

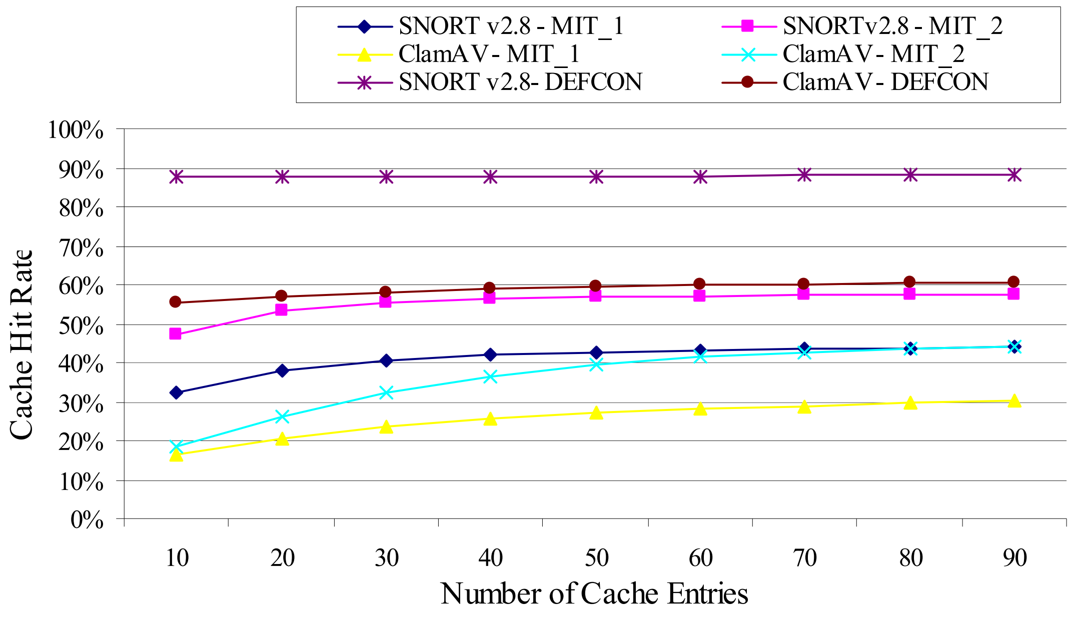


Figure 5-13. Cache hit rates for varying number of cache entries for a TCAM width.

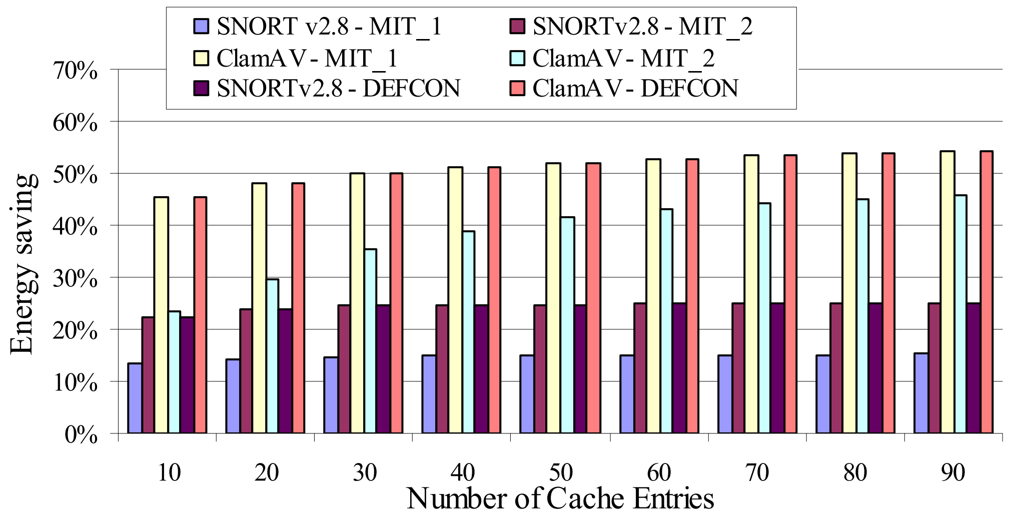
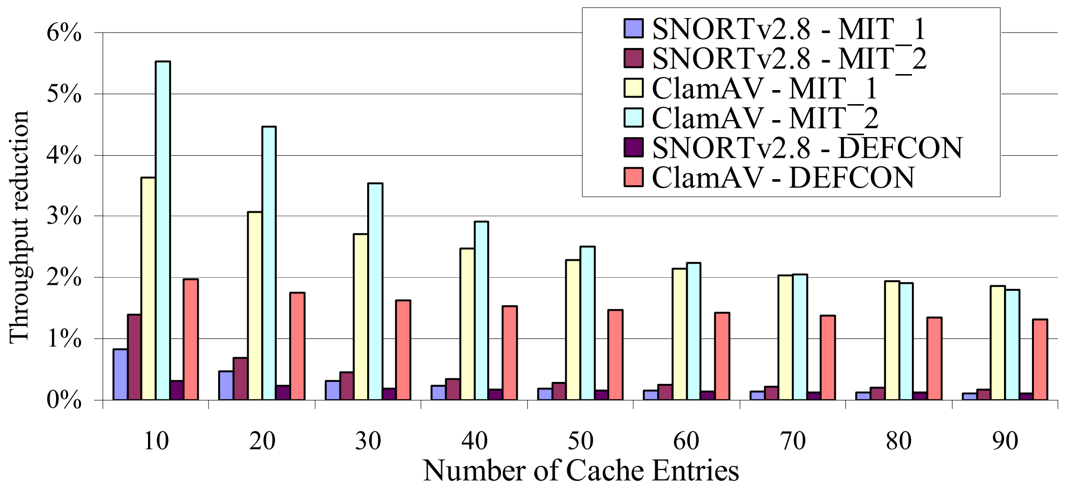


Figure 5-14. Energy savings for a partitioned TCAM system (w=4) with a suffix cache compared to a partitioned TCAM system with no suffix cache for varying number of cache entries.



.

Figure 5-15. Percentage reduction in throughput verses number of cache entries for SNORT and ClamAV signature sets.

CHAPTER 6
SUMMARY

6.1 Conclusion

Next generation NICs will become increasingly complex with increased network responsibilities such as power proxying, network intrusion detection, data caching, etc. 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. This thesis focuses on the challenges of designing a SNIC capable of supporting power proxying.

As the architecture of the next generation SNICs continues to evolve [32, 40], it is identified that header and content inspection systems would be the key architectural elements posing challenges for wide scale deployment. These key architectural elements demand increased processing resources, memory resources, and consume more energy than the other elements in a SNIC's architecture. This work presents an energy and resource efficient design of these key architectural elements suitable for wide scale deployment in next generation SNICs.

This thesis first analyzed the nature of the power proxying rules. As a result of this analysis it became evident that router-based header inspection units needed to be modified for SNICs deployment. Thus, a header inspection system suitable for operation in SNICs to enable power proxying was designed. A low power hardware-based header classification technique is prototyped and analyzed in terms of classification speed, packet throughput, and power consumption compared to a software-based implementation.

The designed hardware header classifier comfortably meets 1 Gbps link rate requirements, with only minor optimizations needed to satisfy 10 Gbps link rates. An equivalent software-based header classifier consumes 4x more power than the hardware-based header classifier. Additionally, the hardware-based header classifier operates up to 9x faster than the software-based header classifier. This increased speed not only supports faster link rates but also enables the PC to be awoken sooner, thus reducing the possibility of packet loss.

Next, this work focused on content inspection unit design. Content inspection is even more challenging problem than header inspection due to high computational and memory requirements. Existing content inspection techniques are carefully evaluated and a TCAM-based content inspection technique is chosen due to its simplicity in pattern matching, high throughput, and scalability with respect to the number and size of the patterns. However, existing TCAM techniques are not suitable for SNICs due to high power consumption and large auxiliary data structure memory requirements. This thesis architected an energy efficient partitioned TCAM-based content inspection system suitable for deployment in next generation SNICs. The proposed system is energy, resource, and throughput aware, with energy delay product improvements of up to 62% compared to previous non-partitioned TCAM systems. Discussion and evaluation of the auxiliary data structure is beyond the scope of this work. Evaluation of the partitioned TCAM system using popular NIDS benchmarks reveals up to 87% energy savings on average compared to a non-partitioned TCAM system. As a further enhancement to the system, a small suffix cache is added to leverage the signature access locality present in network traces. A simple cache with a random replacement policy provides hit rates ranging from 28% to 88%, further reducing the energy consumption of the partitioned TCAM system by 64% compared to a partitioned TCAM system with no cache, with at most a 5.5% throughput reduction.

6.2 Future Work

The work presented in this thesis opens up larger avenues for future work in several directions. Works studying power proxying applications and their behavior are needed to architect a complete SNIC architecture capable of power proxying. The proposed content inspection system was evaluated completely using NIDS signatures sets. Power proxying signature sets need to be developed to enable analysis of signature distributions and architectural optimizations.

This work studied the locality in network traces and proved that caching can improve the energy efficiency of the content inspection system. However, studying and analyzing different caching techniques to further improve energy efficiency is suggested as a future work. Developing a pipelined architecture to circumvent the impact of cache misses on throughput can also extend the proposed work.

In addition, new techniques are needed to address the attack robustness of the content inspection system by developing a methodology to overcome maliciously engineered packets to purposefully defeat energy savings by exploiting system behavior. Also, substantial work exists in evaluating the proposed improvements in auxiliary data structures and final signature matching techniques. Currently, the proposed options use hashing, bloom filters, or software methods in order to further enhance content inspection for wide scale SNIC deployment. However, the suitability of these options needs to be evaluated experimentally to complement the proposed work.

LIST OF REFERENCES

1. B. Agrawal and T. Sherwood, “Modeling TCAM power for next generation network devices,” in *IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, Texas, 2006, pp. 120-129.

2. Z. K.Baker and V. K. Prasanna, “Time and area efficient pattern matching on FPGAs,” in *Proc. ACM/SIGDA International Symposium on Field Programmable Gate Arrays FPGA*, Monterey, CA, 2004, pp. 223 -232.

3. F. Baboescu, S. Singh and G. Varghese, “Packet classification for core routers: Is there an alternative to CAMs?,” in *IEEE Infocom*, San Francisco, CA, 2003, pp. 53-63.

4. F. Baboescu , G. Varghese, “Scalable packet classification, ” in *Proc. Conference on Applications, technologies, architectures, and protocols for computer communications*, San Diego, CA, 2001, pp.199-210.

5. J. Becker, M. Huebner, and M. Ullmann, “Power estimation and power measurement of Xilinx Virtex FPGAs: trade-offs and limitations,” in *Proc. 16th Symp. Integrated Circuits and Systems Design (SBCCI)*, Sao Paulo, Brazil, 2003, pp. 283-288.

6. N. L. Binkert, L. R. Hsu, A. G. Saidi, R. G. Dreslinski, A. L. Schultz, and S. K. Reinhardt. “Analyzing NIC Overheads in Network-Intensive Workloads,” in *Proc 8th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*, Austin, Texas, 2005.

7. B. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *ACM*, May 1970, pp. 422–426.

8. The Shmoo group, “Capture the Capture the Flag Data set,” *http://cctf.shmoo.com/*, Sterling, VA, Nov. 2008.

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

10. Tomasz Kojm, “The Clam AntiVirus Software”, *http://www.clamav.net*, Columbia, MD, Nov. 2008.

11. C. Clark, W. Lee, D. Schimmel, D. Contis, M. Kone, and A. Thomas, “A Hardware Platform for Network Intrusion Detection and Prevention,” *in Proc 3rd Workshop on Network Processors and Applications (NP3)*, Madrid, Spain, 2004,

12. S. Dharmapurikar, P. Krishnamurthy, T.S. Sproull and J.W. Lockwood, “Deep packet inspection using parallel bloom filters,” *IEEE Micro*, 2004, pp. 52-61.

13. S. Dharmapurikar, H. Song, J. Turner and J. Lockwood, “Fast packet classification using bloom filters,” in *Proc.ACM/IEEE Symposium on Architecture for Networking and Communications Systems (ANCS)*,San Jose, CA, 2006, pp. 61-70.

14. D. Friedman and D. Nagle, “Building Firewalls with Intelligent Network Interface Cards”, *CMU SCS Technical Report*, CMU-CS-00-173, May 2001.

15. M. Gao, K. Zhang, J. Lu, “Efficient packet matching for gigabit network intrusion detection using TCAMs,” *in Proc. of Advanced Information Networking and Applications (AINA)*, Vienna, Austria, 2006, pp 18-20.

16. H. Ghasemzadeh, S. Mazrouee, H. G. Moghaddam, H. Shojaei, and M. R. Kakoee, “Hardware Implementation of Stack-Based Replacement Algorithms,” in *Proc. of world academy of science,engineering and technology*, 2006.

17. J.A. Gil-Martinez-Abarca, F. Macia-Perez, D. Marcos-Jorquera, V. Gilart-Iglesias, "Wake on LAN over Internet as Web Service," in *IEEE Conference on Emerging Technologies and Factory Automation (ETFA),* 2006.

18. R. Gonzalez and M. Horowitz, "Energy Dissipation in General Purpose Microprocessors," *IEEE Journal on Solid-State Circuits*, 1996.

19. P. Gupta, A. Light, I. Hameroff, “Boosting Data Transfer with TCP Offload Engine Technology”, *Dell Power Solutions*, 2006.

20. P. Gupta and N. McKeown, “Algorithms for Packet Classification”, *IEEE Network Special Issue*, 2001, vol. 15, no. 2, pp 24-32.

21. P. Gupta and N. McKeown, “Packet Classification on Multiple Fields”, in *Proc. Sigcomm*, Computer Communication Review, 1999, vol. 29, no. 4, pp 147-60.

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

23. Bigfoot Networks, “Killer Network Interface Card”, *http://www.killernic.com/*, Austin, TX, Aug. 2008.

24. H. Kim, S. Rixner, and V. Pai, “Network Interface Data Caching,” *IEEE Transactions on Computers*, 2005, pp. 1394-1408.

25. K. Lakshminarayanan, A. Rangarajan, S. Venkatachary, “Algorithms for advanced packet classification with ternary CAMs,” *ACM SIGCOMM Computer Communication Review*, October 2005.

26. MIT Lincoln laboratory, “MIT-DARPA Intrusion Detection Data Sets,” *http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html*, Lexington, MA, 2008.

27. Mentor Graphics, "ModelSim,"http://www.modelsim.com, Wilsonville, OR, 2008.

28. Jason Todd, "Network Packet Generator (NPG) - Packet injection software," *http://www.wikistc.org/wiki/Network_packet_generator*, Scottsdale, AZ, Oct. 2008.

29. G. Nilsen, J. Torresen and O. Sorasen, "A variable word-width content addressable memory for fast string matching," *in Proc. of Norchip Conference*, Oslo, Norway, 2004, pp. 214-217.

30. J. Noguera, R.M. Badia, "Power performance trade off for reconfigurable computing", in *Proc. International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS)*, San Cugat Del Valles, Spain , 2004, pp 116-121.

31. M.Otey, R. Noronha, G.Li, S. Parthasarathy, and D. Panda, "NIC-based Intrusion Detection: A feasibility study," in *Proc. IEEE ICDM Workshop on Data Mining for Cyber Threat Analysis*, 2002.

32. M. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula, and D. Panda, "Towards NIC-based intrusion detection," in *Proc. ACM International Conference on Knowledge Discovery and DataMining*, Washington, DC, 2003.

33. D. Pao, Y. K. Li and P. Zhou, "Efficient packet classification using TCAMs," *International Journal of Computer and Telecommunications Networking*, 2006, pp. 3523-3535.

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

35. 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," in *Proc. 31st IEEE Conference on Local Computer Networks (LCN)*, Tampa, Fl, 2006.

36. J.Rosenberg , H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, "RFC 3261 - Session Initiation Protocol (SIP)," in *Internet Official Protocol Standards*, East Hanover, NJ, June 2002.

37. R.Gellens, "RFC 4146 - Simple New Mail Notification," in *Internet Official Protocol Standards*, San Diego, CA, 2005.

38. Raptor Networks Technology Inc, "RN2/RN4/RN6 Datasheet," *www.raptor-networks.com*, Santa Ana, CA, 2008.

39. K. Sabhanatarajan, A. Gordon-Ross, M. Oden, M. Navada, and A. George, "Smart-NICs: Power Proxying for Reduced Power Consumption in Network Edge Devices," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Montpellier, France, 2008.

40. D. Schuff, V. Pai, P. Willmann and S. Rixner, "Parallel Programmable Ethernet Controllers: Performance and Security," *IEEE Network*, 2007.

41. 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, 2006.

42. Martin Roesch, “SNORT intrusion detection system”, *www.snort.org*, Columbia, MD, 2008.

43. E. Spitznagel, D. Taylor, and J. Turner, “Packet Classification Using Extended TCAMs,” in *Proc.11th IEEE international Conference on Network Protocols (ICNP)*, 2003.

44. R. Stedman, “Reducing Desktop PC Power Consumption Idle and Sleep modes”, *Technical presentation*, Dell Computer Corporation, June 2005.

45. G. A. Stephen, “String Searching Algorithms,” *Lectures Notes Series on Computing*, 1994, Vol. 3.

46. Xilinx Inc., “Tri-Mode Ethernet MAC v2.1,” *Xilinx product manual*, San Jose, CA, 2008.

47. Xilinx Inc., “Xilinx IP core – Content Addressable Memory”, *Xilinx IP core data sheet*, San Jose, CA, 2008.

48. Xilinx Inc., “Xilinx ISE Simulator,” *http://www.xilinx.com* , San Jose, CA, 2008.

49. Xilinx Inc., “Xilinx Online Power Estimator,” *http://www.xilinx.com/cgi-bin/power_tool/power_Virtex2p* , San Jose, CA , 2008

50. Xilinx Inc., “Virtex-II Pro Platform Data Sheet”, *Xilinx Virtex product manual*, San Jose, CA, March 2005.

51. Xilinx Inc., “Xilinx XPower Tutorial”, *Xilinx Xpower product manual*, San Jose, CA, July 2002.

52. F. Yu, R. H. Katz and T. V. Lakshman, “Efficient Multimatch Packet Classification and Lookup with TCAM”, *IEEE Micro*, 2005, v.25 n.1, p.50-59.

53. F. Yu, R. H. Katz and T. V. Lakshman, “Gigabit rate packet pattern-matching using TCAM,” *IEEE Int’l Conf on Network Protocols*, Berlin, Germany, 2004, pp. 174-183.

BIOGRAPHICAL SKETCH

Karthikeyan Sabhanatarajan received his Bachelor of Engineering (B.E.) degree in the field of electronics and communications from Anna University, Chennai, India in 2006. He then pursued his graduate studies in the Department of Electrical and Computer Engineering at University of Florida, Gainesville in the same year. He joined the High Performance Computing and Simulation (HCS) research laboratory in 2007 focusing his research on the NSF sponsored Energy Efficient Internet project. His primary interests are in the fields of computer networking, computer architecture, embedded systems and virtualization with strong emphasis on green computing. Under the guidance of Dr. Ann Gordon-Ross he furthered his interests in the above fields. He has strong interests in open source computing. He joined VMware Inc. (Palo Alto, California) in 2008 to enhance his knowledge on virtualization technologies. He then worked for Cisco Systems Inc (San Jose, California), as an intern, gaining exposure to the state of art switching and networking technologies. In December 2008, he completed his graduate studies and took up employment with Cisco Systems Inc.