A Resource Efficient Content Inspection System for Next Generation Smart NICs

Karthikeyan Sabhanatarajan and Ann Gordon-Ross
HCS Research Lab, ECE Department, University of Florida
*sabhanatarajan@hcs.ufl.edu , ann@ece.ufl.edu**
** Also with the NSF Center for High-Performance Reconfigurable Computing (CHREC) at the University of Florida*

***Abstract*— The aggregate power consumption of the Internet is increasing at an alarming rate, due in part to the rapid increase in the number of connected edge devices such as desktop PCs. Despite being left idle 75% of the time, 90% of 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. In this paper, we present a NIC-based packet content inspection system for power proxying and network intrusion detection. We use a novel partitioned TCAM technique that results in 87% energy savings and a 62% lower energy-delay product than existing non-partitioned router-based techniques, thus making our technique highly suitable for SNIC-based deployment.**

I. Introduction

Network Interface Cards (NICs) constitute an increasingly important element in modern computer design. Next generation NICs, or smart NICs (SNICs), will be delegated more network responsibility in order to reduce the processing burden on a computer system's CPU [3][14][17][20]. This enables new opportunities for reduced power consumption and increased network security.

One example of a power saving opportunity made possible by increased SNIC network responsibility is power proxying [8][19]. Research shows that 90% of network edge devices (PCs) have their power management systems disabled in order to maintain network connectivity [19], even though these PCs are otherwise idle 75% of the time [19]. Power proxying is a technique that maintains network connectivity while the PC is in a low power sleep state by delegating responsibility to the SNIC. The SNIC responds to incoming network traffic in one of three ways: (1) responds with an automated response (PC remains asleep); (2) ignores packets that are not destined for the PC (PC remains asleep); or (3) wakes up the PC if no automated response exists. Research shows that power proxying can increase sleep time by as much as 85% [19].

One method for increased security made possible by increased network responsibility is a Distributed Network Intrusion Detection System (DNIDS). In a DNIDS, the SNIC's network responsibility includes scanning both inbound and outbound packets for malicious content. The DNIDS delivers increased network security as it can effectively isolate compromised nodes, even those internal to the network, as opposed to router-based centralized NIDS. DNIDS can also increase the overall effectiveness of network security because DNIDS can identify malicious packets based on operating system specifics.

To enable both power proxying and DNIDS, SNICs require packet processing capabilities in the form of *content inspection*. During content inspection, the SNIC extracts packet payloads and performs pattern matching in order to identify packets and respond accordingly. For example, the SNIC can identify a malicious packet if the payload contains any predefined malicious signature patterns.

Modern routers include both software and/or hardware-based content inspection functionalities. However, since routers have much larger computing resources than NICs, these techniques are not immediately suitable for SNIC implementation. Whereas routers utilize processors in the GHz range, NICs include 66 MHz to 400 MHz processors [21], making software-based content inspection infeasible as these processors fail to meet the throughput requirements for 1 Gbps and future 10 Gbps link speeds [21]. Thus hardware based techniques are required.

However, current hardware-based router content inspection techniques are also unfavorable for SNIC implementation. FPGAs [2] allow for fast reprogrammable content inspection, but are too costly and consume more energy than is suitable for wide-scale, low-power SNIC deployment. TCAMs [10][25] provide extremely fast content inspection, but are too power hungry and incur additional resource overhead. Bloom filters [9] provide a low-power alternative solution for content inspection, but suffer from scalability due to the large number of parallel structures required and the overhead of false-positive resolution.

Thus, for feasible wide-scale low-cost SNIC-based content inspection, an energy, power, and area efficient technique is required. While the SNIC constitutes a small percentage of total PC power consumption, even a small reduction in power per PC will aggregate to tremendous total power savings, as the number of PCs is expected to reach 1.3 billion worldwide by 2010 [24].

In this paper, we develop an energy efficient content inspection system for SNICs. The proposed architecture uses a partitioned TCAM-based methodology and achieves up to 87% energy savings and a 62% reduction in the energy delay product compared to existing non-partitioned TCAM techniques. The introduction of a small cache further improves the average energy savings by 64% while reducing the throughput by at most 5.5%.

II. Background and Related Work

In this section, we present related work on next generation SNICs and NIC-based DNIDS, comparing the advantages of NIC-based DNIDS to router-based NIDS. Additionally, we review relevant work on content inspection and evaluate the suitability of these techniques for SNICs.

A. 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) [10], power-proxying [8][19], and NIC-based data caching [14]. 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 DNIDS, providing potentially greater network security than router-based NIDS [7][17][20]. 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. Otey et al. [17] analyzed the feasibility of NIC-based DNIDS and verified that such a system would offer increased coverage, reliability, and performance. However, the authors also recognize that realization of such systems would be challenging given limited NIC processing resources [17][21]. Schuff et al. [20] proposed a NIC-based intrusion detection architecture harnessing the processing resources available in future multi-core RISC processors coupled with specialized content inspection hardware [25]. However, since this technique was based on router content inspection techniques, this technique was too power and resource hungry for SNICs.

B. 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 [23] and ClamAV [6] 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 intrusion detection and power proxying in next generation SNICs. This is a well researched topic in the context of routers [9][10][25]. 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 [22], etc. However, due to inherent software inefficiencies when processing large signature sets, software techniques cannot support high link speeds [9].

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

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

To increase data processing throughput, specialized hardware-based techniques exploit parallelism using FPGAs [2], TCAMs [10][25], and specialized data structures such as Bloom Filters [9]. 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 [25]. 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. [25]. 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, we refer the reader to [25].

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 utilization since signature lengths tend to be highly variable [25].

To improve resource utilization, 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 utilization, 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 utilization 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 utilization, 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*. Fig 1 shows the prefix and suffix patterns for a sample long pattern signature given a TCAM width $w = 4$ (each character represents an arbitrary byte).

The long and short patterns are stored 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 $\lceil l/w \rceil$ 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 [25].

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 [9]), 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 [18]) and cost (4x that of SRAM [18]). 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 utilization 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 utilization, Gao et al. [10] 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 prefix and suffix signatures) populated with valid signature address permutations. *Valid signature address permutations* are the concatenation of the prefix and suffix 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 [16] implemented a variable width TCAM to improve resource utilization 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. [9] 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.

In this paper, we architect a content inspection technique that is more amenable to limited resource SNICs by extending TCAM-based techniques [10][25], reducing both energy consumption and the energy delay product. We propose a method by which the single TCAM is partitioned into a prefix TCAM and a suffix TCAM. This partitioned technique reduces TCAM switching activity, without increasing area, and thereby reduces system energy consumption. Finally, we also introduce a caching technique to further reduce energy consumption, motivated by a NIC packet caching technique that exploits network traffic locality [14]. Our technique assumes the NIC architecture proposed in [20], which includes low resource mechanisms for packet reassembly and check summing.

III. SNIC-Based Content Inspection System

In this section, we present an energy efficient content inspection architecture for SNIC-based systems to aid in power-proxying and DNIDS.

A. Definitions

The distinguishing features of our 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; and (2) the introduction of a suffix cache, which stores a subset of the S_TCAM entries. 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, we define identical prefix and suffix patterns 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 $\lceil l/w \rceil$ patterns, we define a *concluding* pattern as the final partition $\lceil l/w \rceil$ (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, we define all partitions $1 \le p < \lceil l/w \rceil$ as *intermediate* patterns.

B. Architecture

Fig 2 depicts our proposed content inspection architecture, consisting of three signature storage units: the P_TCAM, suffix cache, and the S_TCAM. We assume the inspection window size is 4 bytes and the signature storage units are populated using the sample signature from Fig 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 Fig 1 we 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* [10] and must be considered during caching. To avoid inconsistencies due to mutual inclusion, we only cache S_TCAM entries that are exactly *w* bytes (entries without any "don't care" padding bits).

We assume that payload reconstruction (not shown in Fig 2) aggregates incoming network packets to reconstruct complete payloads, and this 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 an intermediate pattern hit (in any signature storage unit). The *activator* monitors all signature storage units and upon an intermediate pattern hit, sets the 0^{th} 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, we use a random replacement policy, which is shown to have similar performance as LRU for large associativities [11]. It should be noted that the introduction of caching stalls the system by a cycle during the cache miss and thus leads to reduced throughput. In section V.E, we show that this overhead is minimal.

The *retirement buffer* stores candidate signature address permutations, and serves as input to the final signature matching step (we extend the technique proposed by [10] 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 entries 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,

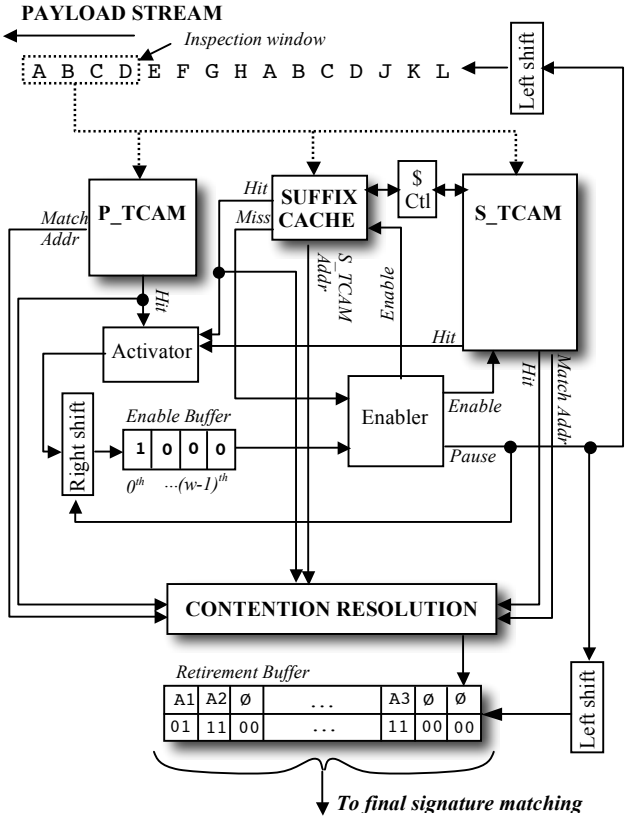

Fig 2: Partitioned TCAM system for SNIC-based content inspection.

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 Fig 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 from each other), terminating on an Ø position. The candidate signature match permutations are dispatched to the final signature matching unit (not shown in Fig 2). The final signature matching unit can use hashing structures such as bloom filters [4] or software methods to compare candidate and valid signature match permutations. Elaborations of such techniques are beyond the scope of this paper, and optimization of this step is the focus of our future work.

IV. MATHEMATICAL MODEL

In this section, we analyze the resource requirements for our proposed architecture and develop a model for energy expenditure analysis.

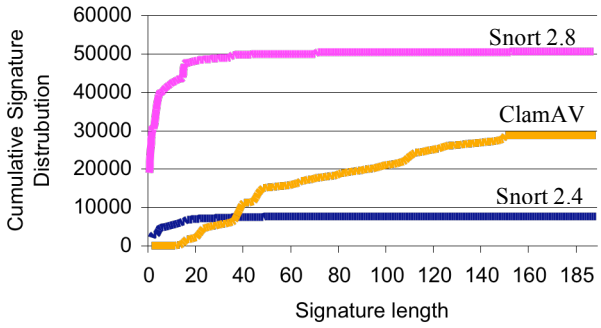

Fig 3: Cumulative number of rules (distribution) for increasing signature lengths for Snort and ClamAV signature sets.

To describe the total TCAM (both prefix and suffix) and retirement buffer resource requirements, we define w as the width of the TCAMs 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):

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

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 [10]:

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

We assume 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.

All TCAM expenditures can be aggregated into the total energy expended:

$$\begin{aligned} Total\_Energy = {} & Num\_P\_TCAM\_Accesses * P\_TCAM\_EPA \\ & + Num\_Intermediate\_Accesses * Cache\_EPA \\ & + Num\_Cache\_Misses * S\_TCAM\_EPA \\ & + Num\_Cache\_Misses * Cache\_Write\_EPA \\ & + Num\_S\_TCAM\_Accesses * S\_TCAM\_EPA \end{aligned}$$

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

$$EPA = Total\_Energy / Total\_Accesses$$

$$Total\_Accesses = \sum_{i=1}^{X} P_i$$

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

It should also be noted that best case energy consumption occurs when all lookups miss in the P_TCAM (no inspection windows match any signatures) and the worst case energy consumption occurs when there is a hit in the P_TCAM and a subsequent cache miss.

V. Analysis and Experimental Results

In this section, we provide experimental analysis of our proposed intrusion detection system. We first analyze signature length distribution of two popular signature sets. We then analyze the impact of TCAM partitioning (without suffix caching) with respect to area, energy consumption, and the energy-delay product (EDP) [12]. Next, we simulate popular NIDS trace benchmarks to determine average energy savings and compare this to the unpartitioned TCAM approach modeled using the same environment. Finally, we introduce the suffix cache into our system and analyze its effects.

A. Experimental Setup

For our experiments, we modeled our intrusion detection system using a custom C-based simulator. For a given TCAM width w, the SNORT [23] and ClamAV [6] signature sets are populated in the TCAM structures accordingly. We use popular NIDS benchmark traces from the MIT Lincoln Laboratory (MIT-LL) [15] and the "capture-the-flag" contest for the DEFCON festival [5].

During a trace pre-analysis step, incoming fragmented packets are reassembled and the payload of the reassembled packets are extracted and passed to our intrusion detection simulator. The simulator behaviorally simulates our proposed architecture, 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.

We obtain TCAM energy consumption 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. We combine this with the our mathematical models (section IV) to obtain the resource usage and energy consumption.

B. Signature Length Distribution Analysis

To assist in appropriate TCAM width w determination and avoid reduced resource utilization due to excessive "don't care" bit padding, we first analyze signature length distribution. Fig 3 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. Our graphs conform to the findings in [25] 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 we observed these same trends for all experimental results), we only present experimental results for SNORT v2.8.

C. Effects of TCAM Partitioning on Size, Energy, and the Energy Delay Product

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, we first analyze the impact on total area due to TCAM partitioning.

Fig 4 depicts partitioning effects on TCAM size in KBytes for the SNORT v2.8 (a) and ClamAV (b) signature sets verses varying TCAM widths. 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.

Fig 5 depicts 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 (a) and ClamAV (b) signature sets. For SNORT, Fig 5 (a) shows that 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, Fig 5 (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, our simulations using popular benchmark traces in section V.D shows that the worst case scenario rarely occurs.

Even though our 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. Fig 6 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.

D. *Energy Savings from Partitioning with Real-Time Network Traces*

Fig 7 depicts 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

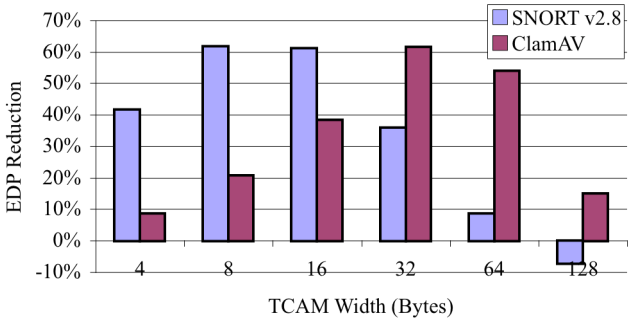

Fig 6: Percentage reduction in the energy-delay product (EDP) for a partitioned TCAM system compared to a non-partitioned TCAM system verses TCAM width.

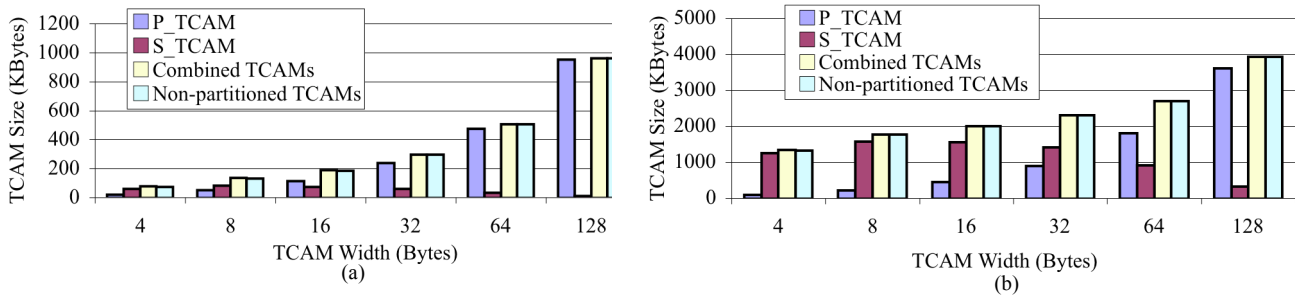

Fig 4: TCAM size variation verses TCAM width for the (a) SNORT v2.8 and (b) ClamAV signature sets for the P_TCAM and S_TCAM individually, the P_TCAM and S_TCAM combined (Combined TCAMs), and the non-partitioned TCAM system.

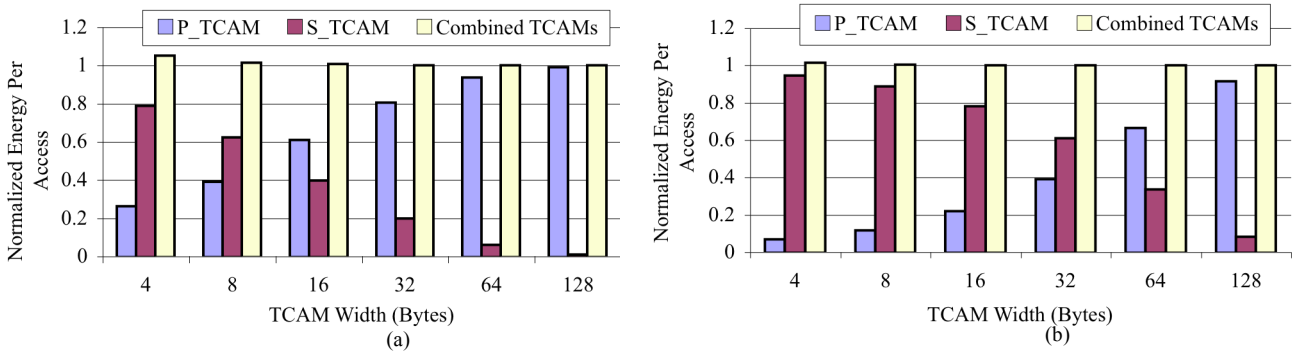

Fig 5: Energy per access normalized to a non-partitioned TCAM system verses TCAM width for the (a) Snort v2.8 and (b) ClamAV signature sets for the P_TCAM and S_TCAM individually as well as the P_TCAM and S_TCAM combined (Combined TCAMs).

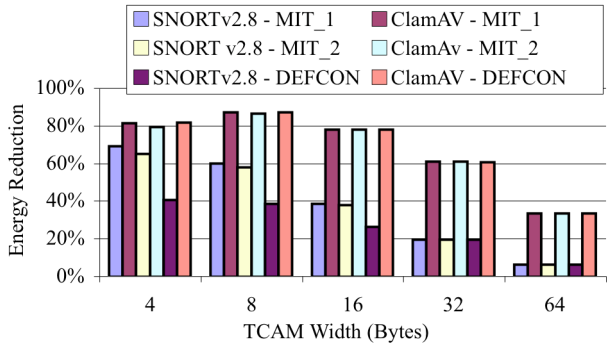

Fig 7: Energy reduction for a partitioned system compared to a non-partitioned system verses TCAM width for real-time traffic traces.

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.

E. Network Trace Locality and Caching

First, we analyze network trace locality in order to motivate caching benefits. Fig 8 is a plot of the matching SNORT signature identification (ID) number verses ordered incoming malicious packets for the MIT-LL traces. As the figure shows, only a very few unique signatures match, and those matched exhibit significant temporal locality.

Next, we analyze the distribution of TCAM accesses between the P_TCAM and the S_TCAM to reveal further caching potential. Fig 9 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 Fig 7 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. We point out that the percentage is largely dependent on the nature of traces and the signature sets used.

We analyze caching impacts for a TCAM width of 4

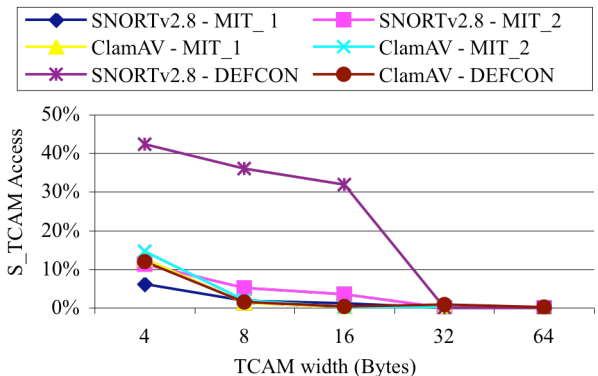

Fig 9: Percentage of S_TCAM accesses for various TCAM widths populated by SNORT v2.8 and ClamAV signature sets

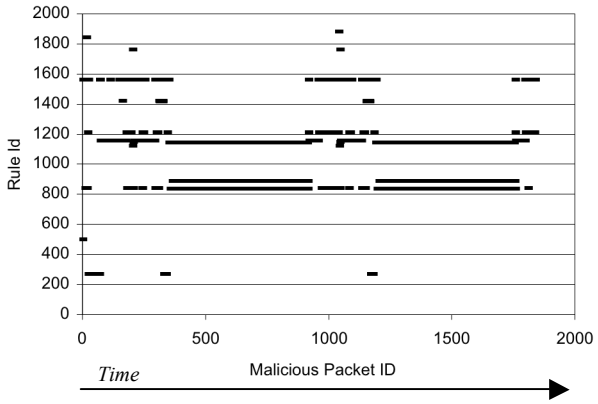

Fig 8: Signature access locality (SNORT rule ID verses time represented by the malicious packet ID) as observed by an edge node under attack

bytes, as this width provides the greatest number of S_TCAM accesses. Fig 10 depicts 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.

Fig 11 shows energy reduction for a partitioned TCAM system with a suffix cache compared to a partitioned TCAM system with no suffix cache. The inclusion of a small cache revealed 13% to 64% additional energy savings compared to a partitioned TCAM system with no suffix cache.

Fig 12 analyzes the throughput reduction due to cache misses. Whereas in the worst case (all P_TCAM accesses hit and all suffix cache accesses miss) throughput would be reduced by 100%, Fig 12 shows that actual throughput reduction is minimal and ranges from 0.001% to 5.5%.

VI. CONCLUSION

In this paper, we architected an energy efficient partitioned TCAM-based content inspection system suitable for deployment in next generation SNICs. The proposed system is both energy and throughput aware, with energy delay product improvements of up to 62% compared to previous non-partitioned TCAM systems. Evaluation of our partitioned TCAM system using popular NIDS benchmarks revealed up to 87% energy savings on average compared to a non-partitioned TCAM system. We further enhanced our system by adding a small suffix cache to leverage the

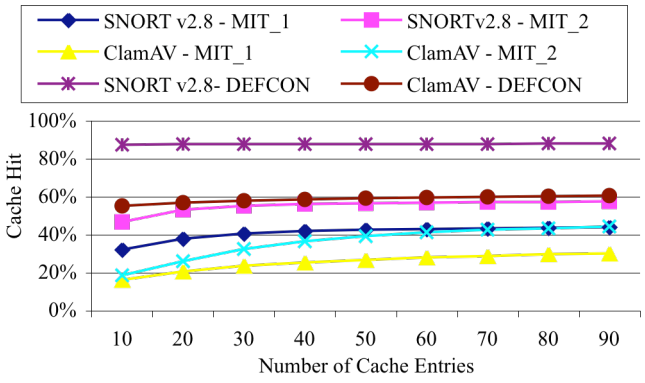

Fig 10: Cache hit rates for varying number of cache entries for a TCAM width of 4 bytes.

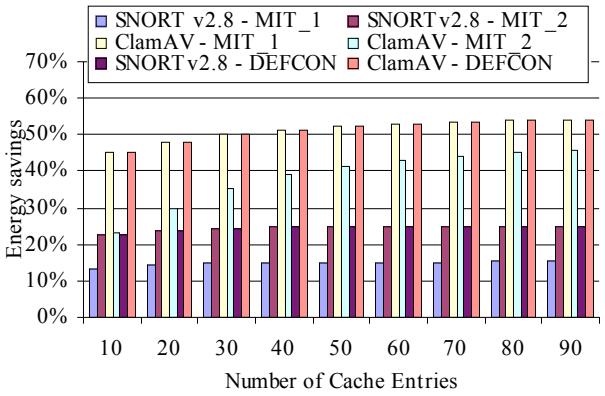

Fig 11: Energy savings for a partitioned TCAM system (w=4) with a suffix cache compared to a partitioned TCAM system with no suffix cache for varying number of cache entries.

signature access locality present in network traces. A simple cache with a random replacement policy provided 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.

Future work includes studying improved caching techniques with respect to energy consumption and development of a pipelined architecture to circumvent the impact of cache misses on throughput. We also plan to address the attack robustness of our system by developing a methodology to overcome maliciously engineered packets to purposefully defeat energy savings by exploiting system behavior. Finally, we will develop improved auxiliary data structures and final signature matching techniques using hashing, bloom filters and other software methods in order to further enhance content inspection for wide scale SNIC deployment.

VII. ACKNOWLEDGMENT

This work is supported by the National Science Foundation under Grant No. 0520081. The authors would like to thank Dr. Ken Christensen for his insightful review of the work.

VIII. REFERENCES

[1] B. Agrawal and T. Sherwood, "Modeling TCAM power for next generation network devices," IEEE International Symposium on Performance Analysis of Systems and Software, 2006,pp. 120-129.

[2] Z. K.Baker and V. K. Prasanna, "Time and area efficient pattern matching on FPGAs," Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays FPGA '04, 2004.

[3] 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.," Eigth Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW), 2005

[4] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *ACM*, May 1970, pp. 422–426.

[5] Capture the Capture the Flag Data set, http://cctf.shmoo.com/

[6] Clam AntiVirus, www.clamav.net

[7] C. Clark, W. Lee, D. Schimmel, D. Contis, M. Kone, and A. Thomas, "A Hardware Platform for Network Intrusion Detection and Prevention," Proceedings of The 3rd Workshop on Network Processors and Applications (NP3), February 2004

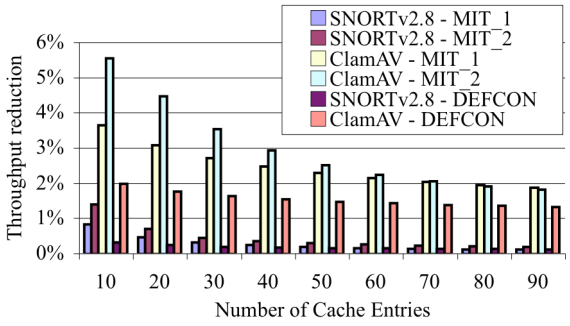

Fig 12: Percentage reduction in throughput verses number of cache entries for SNORT and ClamAV signature sets.

[8] K. Christensen, P. Gunaratne, B. Nordman, and A. George, "The next frontier for communications networks: power management," Computer Communications, 2004,pp. 1758-1770.

[9] S. Dharmapurikar, P. Krishnamurthy, T.S. Sproull and J.W. Lockwood, "Deep packet inspection using parallel bloom filters," Micro, IEEE, 2004, pp. 52-61.

[10] M. Gao, K. Zhang, J. Lu, "Efficient packet matching for gigabit network intrusion detection using TCAMs," in proceedings of Advanced Information Networking and Applications, 2006. AINA

[11] H. Ghasemzadeh, S. Mazrouee, H. G. Moghaddam, H. Shojaei, and M. R. Kakoee, "Hardware Implementation of Stack-Based Replacement Algorithms," Proceedings of world academy of science,engineering and technology, 2006

[12] R. Gonzalez and M. Horowitz, "Energy Dissipation in General Purpose Microprocessors," IEEE Journal on Solid-State Circuits, September 1996.

[13] P. Gupta, A. Light, I. Hameroff, "Boosting Data Transfer with TCP Offload Engine Technology ", Dell Power Solutions, August 2006

[14] H. Kim, S. Rixner, and V. Pai, "Network Interface Data Caching," *IEEE Transactions on Computers*, 2005, pp. 1394-1408.

[15] MIT-DARPA Intrusion Detection Data Sets, http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html.

[16] G. Nilsen, J. Torresen and O. Sorasen, "A variable word-width content addressable memory for fast string matching," Proceedings of Norchip Conference, 2004, pp. 214-217,

[17] M.Otey, R. Noronha, G.Li, S. Parthasarathy, and D. Panda, "NIC-based Intrusion Detection: A feasibility study," Proceedings of the IEEE ICDM Workshop on Data Mining for Cyber Threat Analysis, December 2002

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

[19] 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," Proceedings of 31st IEEE Conference on Local Computer Networks (LCN), 2006.

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

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

[22] G. A. Stephen, "String Searching Algorithms," Lectures Notes Series on Computing, 1994, Vol. 3

[23] SNORT intrusion detection system, www.snort.org

[24] S. Yates, "Sizing the Emerging-Nation PC Market", Forrester research.

[25] F. Yu, R. H. Katz and T. V. Lakshman, "Gigabit rate packet pattern-matching using TCAM," IEEE Int'l Conf on Network Protocols, Oct. 2004, pp. 174-183.