Partition Filter Set for Power-Efficient Packet Classification

Haibin Lu, Mian Pan
luhaibin@missouri.edu, mpry6@mizzou.edu
Department of Computer Science
University of Missouri, Columbia, MO 65211

***Abstract*— Ternary Content-Addressable Memory (TCAM) has been widely used for high-performance multi-dimensional packet classification. High power consumption limits the use of TCAM for large filter sets. TCAM power consumption is proportional to the number of TCAM entries enabled for search. Dividing TCAM into many blocks and enabling only a few blocks for search has been proposed to reduce the power consumption dramatically. However, it is quite challenging to design efficient algorithms to partition a set of multi-dimensional filters into many subsets (a subset is placed in one TCAM block). The efficiency of the algorithm is evaluated in three aspects: the maximum number of TCAM blocks that need to be enabled for a single search, the storage utilization of TCAM blocks, and the time and space complexity of the partition algorithm. In this paper, we developed a simple but efficient partition algorithm based on the Hilbert curve. The algorithm reduces TCAM power consumption by a factor of ten on average. The TCAM storage utilization is over 99%. The algorithm takes $O(n \log n)$ time and $O(n)$ space.**

Keywords: packet classification, TCAM, power efficient, Hilbert curve, firewall.

I. Introduction

Packet classification is an essential module in firewall and access control. For each incoming packet, the packet classification module takes several fields from IP header and TCP/UDP header and then matches against the filter set. The packet header fields used for classification include source IP address, destination IP address, source port number, destination port number, protocol, etc. Each filter is a pair of the form (rule, action). The action part specifies what to do if the rule part matches the incoming packet, for example, forward the packet to a specific output port, drop the packet, mark the packet for express forwarding, etc. The rule part specifies the possible values for each header field in the format of range or prefix. Figure 1 shows a set of six filters. It assumes that the length of IP address is three bits, that is, IP address is between 0 and 7 (between 000 and 111 when specified in binary format). The possible values of source/destination IP addresses are specified in prefix format. "x" is "don't care" bit. It matches both bit "0" and bit "1". Hence, prefix "0xx" matches the following IP addresses: 000, 001, 010, and 011 (in binary format). The possible values of source/destination port numbers are specified in range format. For example, [0, 5] matches port numbers between 0 and 5. A dash matches any port number. The protocol part is either fully specified or a dash which matches any protocol. For instance, the first filter can only match packets carrying TCP segments. The sixth filter can match any protocol. As a convention we use binary number for prefixes and IP addresses, and decimal number for ranges and port numbers. If the header fields chosen from the incoming packet are (011, 110, TCP, 2, 5), the second filter is the matching filter. When there are multiple matching filters, the commonly used tie breaker is the first matching rule in which the filter set is considered as a linear list and the first filter matching the incoming packet will be used.

	source IP address	dest IP address	protocol	source port	dest port	action
1.	0xx	101	TCP	[0, 1]	[0, 5]	fwd0
2.	01x	11x	TCP	[1, 2]	[1, 7]	fwd3
3.	0xx	111	UDP	[3, 4]	[2, 4]	fwd2
4.	01x	10x	UDP	-	[4, 6]	fwd2
5.	100	1xx	UDP	[1, 3]	-	fwd4
6.	11x	01x	-	-	-	drop

Fig. 1. A set of six filters. The rule part consists of possible values for source IP address, destination IP address, protocol, source port number, and destination port number. "x" is "don't care" bit. A dash is a wildcard which matches any value.

Packet classification is a hard problem due to its multi-dimensional nature and high performance requirement. High-end systems need to perform tens of millions of classification per second on a large filter set. Currently, the size of the largest filter set is about a few thousands of filters, but is expected to grow substantially.

Ternary Content-Addressible Memories (TCAM) use parallelism to finish classification in a single cycle. Each memory cell of a TCAM may be set to one of three states 0, 1, and "don't care". Each TCAM entry consists of a fixed number of memory cells. When classification is performed, the header fields chosen from the incoming packet are compared, in parallel, against every TCAM entry. The matching entries are sent to a specific logic module to decide which one to use according to the tie breaker. However, native TCAM does not support range matching. A simple way to deal with it is to split a single range into a set of prefixes. For example, range [0, 2] is split into ranges [0, 1] and [2, 2], each of which can be represented by a single prefix (i.e., [0, 1] by prefix 00x and [2, 2] by prefix 010). This approach expands one filter into many filters and results in low TCAM storage utilization. Spitznagel et al. [15] propose extended TCAM that can handle range matching naturally.

Data structures for non-TCAM-based multi-dimensional packet classification are developed in [1], [3]–[6], [9], [12], [16], [17]. These data structures have the advantage of lower power consumption and often lower cost. However, the per-

This work was supported, in part, by the National Science Foundation under grant CNS-0423386.

1-4244-0357-X/06/$20.00 ©2006 IEEE

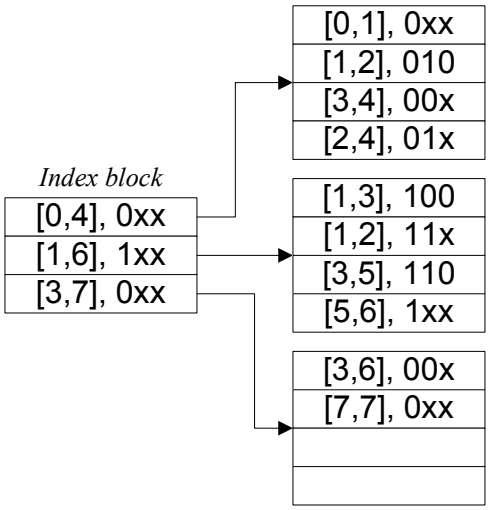

Fig. 2. A set of 10 filters is partitioned into 3 subsets on the right side. Each subset is represented by an index filter stored at the index block on the left side. For simplicity, the action part is omitted and the rule part consists of only two fields: destination port range and source IP address prefix.

formances of these non-TCAM-based data structure are often inferior to that of TCAM-based schemes. The use of TCAM is limited by its high power consumption, which is proportional to the number of TCAM entries enabled for classification. To reduce the power consumption, it is necessary to reduce the number of TCAM entries enabled for classification, which can be achieved because there are only a very small number of filters that match an incoming packet. If these filters are placed together, we only need to enable the blocks containing these filters. This idea has been used for TCAM-based packet forwarding (one-dimensional classification) [14]. Several partitioning algorithms [11], [14], [19] for one-dimensional prefix filters have been proposed. Spitznagel et al. [15] apply this architecture to TCAM-based multi-dimensional packet classification. Figure 2 gives an example. Ten filters on the right side are partitioned into three groups. The first two groups have four filters each and the third group has two filters. Each group is stored at one TCAM block on the right side. Each group is represented by an *index filter* at the index TCAM block on the left side. For example, ([0, 4], 0xx) is the index filter for the first group. [0, 4] is *the minimum bounding range* for ranges [0, 1], [1, 2], [3, 4] and [2, 4]. 0xx is *the minimum bounding prefix* for prefixes 0xx, 010, 00x and 01x. Suppose the chosen header fields of an incoming packet are (1, 011). The index block is searched first. There is only one matching filter, which is the first index filter. So the first TCAM block is enabled for search. Suppose the chosen header fields of an incoming packet is (4, 011). Searching index block returned two matching index filters, the first and the third index filters. So the first and the third TCAM blocks are enabled for search. To complete the classification, the output of TCAM blocks that were enabled for search is fed into a tie breaker module, which is omitted in Figure 2.

Spitznagel et al. [15] propose a heuristic recursive cut algorithm for partitioning a filter set. In this paper, we propose a much simpler algorithm to partition a filter set. Our algorithm

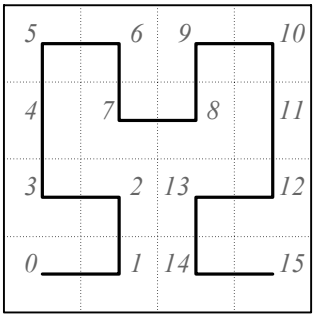

Fig. 3. Hilbert Curve in two-dimensional space (4×4 grid)

achieves similar power reduction as that of [15], guarantees near 100% TCAM block utilization (versus 95% in [15]), and takes only $O(n \log n)$ time to run.

In Section II, we describe a heuristic filter grouping algorithm based on the Hilbert curve. Experimental results are presented in Section III. Our results are summarized in the conclusion, Section IV.

II. Filter Grouping Based on Hilbert Curve

Definition 1: (region). Each filter defines a *region* in a d-dimensional space. For example, filter ([0, 1], 0xx) defines a rectangle ([0, 1], [0, 3]) in two-dimensional space.

Definition 2: (cover). Filter f_1 *covers* filter f_2 if f_1's region completely contains f_2's region. Similarly, let p be any point in a d-dimensional space. We say that filter f_1 covers p if p is within f_1's region.

Definition 3: (overlap). Filter f_1 *overlaps* filter f_2 if their regions share at least one point.

Definition 4: (overlap number). Let F be a set of n filters. We define the overlap number of F as the maximum number of filters that covers any common point.

Our goal is to divide a set of n filters into m subsets, where $m = \lceil n/k \rceil$ and k is the size of TCAM block, such that the overlap number of the index filters of these subsets is as small as possible.

Intuitively, the filters whose regions are close to each other should be grouped together. However, unlike one-dimensional case, there is no total ordering that preserves spatial locality in multi-dimensional space. Commonly used ordering functions are based on the z curve [13], the Gray-coded curve [2], or the Hilbert curve. These functions map a multi-dimensional point to a single integer value. Jagadish [7] shows that the mapping based on the Hilbert curve outperforms the others in preserving spatial locality under most circumstances. Figure 3 shows an example of Hilbert Curve for a 4×4 grid in two-dimensional space. Hilbert Curve traverses every grid point exactly once in some particular order. Correspondingly, each grid point is labelled with a sequence number according to the order traversed by the Hilbert curve.

Due to its superior clustering property, we use Hilbert Curve to group the filters. We sort filters according to their Hilbert values, and then group these filters in order by their Hilbert values. However, the Hilbert mapping function takes a point as the input, but our filters are regions in

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

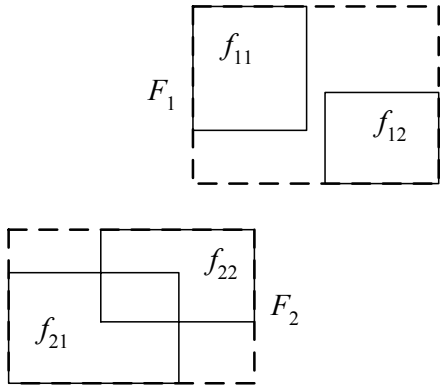

Fig. 4. Minimum bounding boxes are shown in dash lines. The original filters are in sold lines

d-dimensional space. To solve this problem, we simply take the smallest coordinates of the filter region in each dimension to form a d-dimensional point. For example, if a filter $f = ((sip_l, sip_h), (dip_l, dip_h), (sport_l, sport_h), (dport_l, dport_h))$, we use $(sip_l, dip_l, sport_l, dport_l)$ as the input to the Hilbert mapping function. Below is our filter grouping algorithm.

1) Use Hilbert Curve to map each d-dimensional filter to a single integer value (Hilbert value).
2) Sort these filters according to their Hilbert values.
3) Group every k filters together, where k is the size of a TCAM block. Each group is represented by the minimum bounding box of the filters in this group.
4) For each minimum bounding box, replace the minimum bounding ranges of source/destination IP addresses with the corresponding minimum bounding prefixes. The resulting bounding box is used as the index filter for each group.

Note that Step 4 is unnecessary if the range matching circuits proposed in [15] are used for source/destination IP address range matching. Step 4 may increase overlap number of the index filters since it may expand the minimum bounding boxes. Figure 4 shows two minimum bounding boxes, F_1 and F_2. The filters are f_{11}, f_{12}, f_{21}, and f_{22}. Suppose $F_2 = ([0, 5], [0, 2])$. If we replace the second range with the minimum bounding prefix, which is 0xx, the resulting bounding box covers more area than the minimum bounding box.

Mapping a d-dimensional point to its Hilbert value takes $O(d)$ time. Sorting takes $O(n \log n)$ time, where n is the number of filters. Computing the minimum bounding box of k filters takes $O(kd)$ time. Therefore, the time complexity of this algorithm is $O(n \log n + nd)$.

Note that we can also use the Hilbert values of eight-dimensional points that incorporate the size of filter region, say, $(sip_l, sip_h, dip_l, dip_h, sport_l, sport_h, dport_l, dport_h)$, or $(sip_l, dip_l, sport_l, dport_l, sip_h - sip_l, dip_h - dip_l, sport_h - sport_l, dport_h - dport_l)$. Our experiment shows that the overlap number when these eight-dimensional points are used is no better than that when $(sip_l, dip_l, sport_l, dport_l)$ is used and in some cases it is slightly worse. This result is consistent with the result in [8], in which Hilbert curve is used to pack R-tree on two-dimensional rectangles.

The algorithm above fills all the TCAM blocks except the last one. Therefore, high TCAM storage utilization is guaranteed.

III. Experiment

We used the code of [10] to compute the Hilbert value of each filter. The minimum bounding ranges of source and destination IP addresses are expanded to the minimum bounding prefixes. The real-world classifiers that are available for experiment are often very small (less than 1,000). We use ClassBench [18] to generate large classifiers. ClassBench uses the statistical properties of real-world classifiers to generate random filters. We use two seed files, which contains statistical properties of two real-world classifiers, $acl1$ and $acl5$. n is the target size of the filter set generated. ClassBench also takes a few parameters to adjust the property of the filter set to be generated. The *smoothness* parameter adjusts the distribution of new prefix lengths around the existing prefix lengths. A bigger smoothness parameter introduces wider distribution of new prefix lengths. The *scope* parameter adjusts the specificness of prefixes and port range. Positive scope values favor less specific address prefixes and port ranges. That is, prefix or range generated matches a larger number of values. For example, range [0, 1] or prefix 00x covers two values, 0 and 1, while range [0, 3] or prefix 0xx covers four values, 0, 1, 2, and 3. Negative scope values favor more specific address prefixes and port ranges. That is, prefix or range generated matches a smaller number of values.

The ***power fraction*** is equal to $(\lambda k + n/k)/n$, where λ is the overlap number of the index filters (this is the maximum number of TCAM blocks that need to be enabled for search), k is the size of TCAM block (i.e., the number of TCAM entries per TCAM block), and n/k is the number of index filters (index block is always enabled for search). When extended TCAM [15] is used, one TCAM entry holds a filter. Thus, k is also the number of filters per TCAM block. The smaller power faction, the bigger the reduction of power consumption.

Since our algorithm always fills all TCAM blocks except the last block, the storage efficiency is over 99%. Due to its simplicity, the algorithm finishes in less than 100 millisecond on a P4 2G processor. The low computational complexity is important as periodical reconstruction becomes more frequent when the number of flow-specific filters increases.

Table I shows the overlap number with and without step 4 of the algorithm. It is clear that step 4 increases the overlap number. The overlap number with step 4 of the algorithm is plotted in Figure 5. We can see that power fraction highly depends on the size of TCAM block. On the one hand, reducing k by half does not double the overlap number, thus has the potential of achieving better power fraction. On the other hand, reducing k increases the size of index block, which is always enabled for lookup. The experiment shows that $k = 64, 128$ result in the best power reduction. In the rest experiment results, step 4 of the algorithm is applied.

Figure 6 shows the filter set size versus power fraction. $k = 128$ for the sets with 50K and 32K filters, and $k = 32$

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

k		16	32	64	128	256	512
index block size		3200	1600	800	400	200	100
acl1	without step 4	10	9	9	8	7	6
	with step 4	12	12	11	9	8	7
acl5	without step 4	12	10	9	7	7	5
	with step 4	16	13	11	10	9	7

TABLE I

THE OVERLAP NUMBER WITH AND WITHOUT STEP 4 OF THE ALGORITHM. $n = 50K$, SMOOTHNESS = 2, ADDRESS SCOPE PARAMETER = 0.5, APPLICATION SCOPE PARAMETER = -0.1

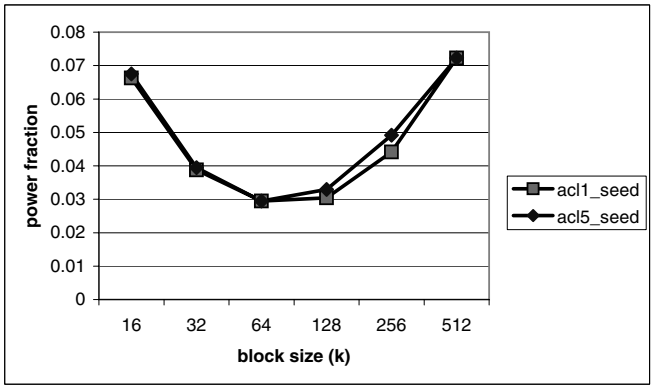

Fig. 5. TCAM block size versus power fraction. $n = 50K$, smoothness = 2, address scope parameter = 0.5, application scope parameter = -0.1

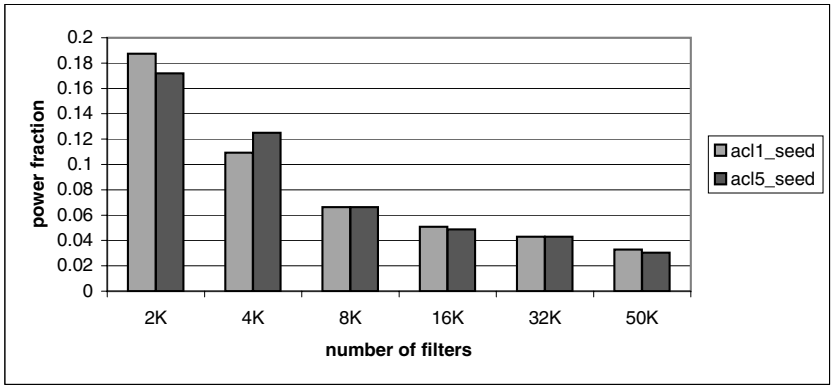

Fig. 6. Filter set size versus power fraction. Smoothness = 2, address scope parameter = 0.5, application scope parameter = -0.1

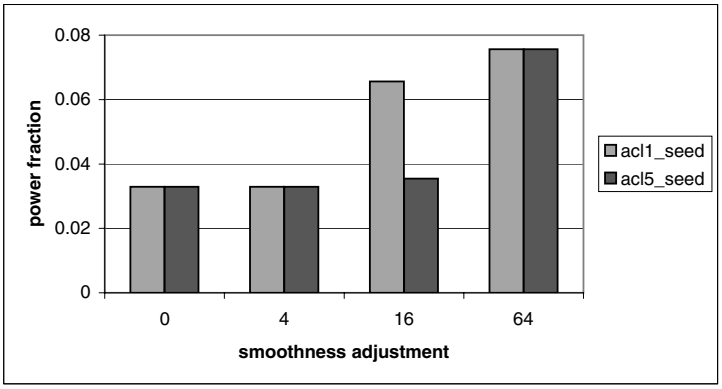

Fig. 7. Smoothness adjustment versus power fraction. $n = 50K$, address scope parameter = 0.5, application scope parameter = -0.1

for the rest sets. Power reduction is less effective when filter set size gets smaller.

Figure 7 shows the smoothness adjustment versus power fraction. A bigger smoothness parameter introduces wider distribution of new prefix lengths, thus increases the possibility of filter overlapping. Thus, power reduction is less effective

1-4244-0357-X/06/$20.00 ©2006 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.

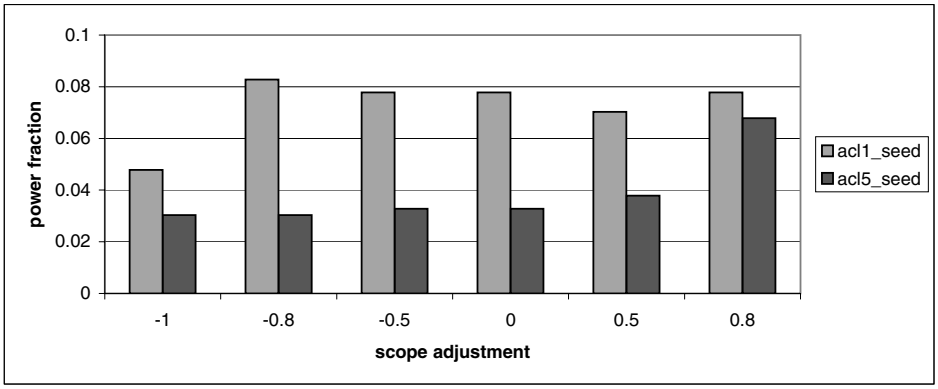

Fig. 8. Scope adjustment versus power fraction. $n = 50K$, $k = 128$, smoothness = 16.

for large smoothness parameters.

Figure 8 shows the scope adjustment versus power fraction. We set address scope parameter and application scope parameter to the same value in the range of [-1.0, 1.0). Figure 15 in [15] uses scope parameters between -64 and 16. The parameters we use here are normalized (the released ClassBench software only supports normalized scope parameters). Positive scope values favor less specific address prefixes and wider port ranges, thus increase the possibility of filter overlapping. As expected, larger scope parameter generally results in higher power fraction (i.e., less power reduction). However, most power fractions are below 8%.

Compared with the heuristic algorithm in [15], our algorithm achieves comparable power reduction, higher TCAM storage efficiency (99% vs 95% for [15]), and is much simpler.

IV. Conclusion

To reduce the power consumption of TCAM-based packet classification, partition-based architecture divides the filter set into many subsets. Each subset is stored at a TCAM block and is represented by an index filter in the index TCAM block. The index TCAM block is always searched first. The output of the index block is used to enable the corresponding TCAM blocks for search. Since only a fraction of TCAM blocks are enabled per classification, the power consumption is greatly reduced. However, how to design an efficient filter grouping algorithm is quite challenging. The overlap number of the index filters needs to be as small as possible, TCAM storage utilization needs to be high, and the algorithm should have low time and space complexities. We have evaluated the filter grouping algorithm based on Hilbert curve. Our algorithm has over 99% TCAM storage efficiency, reduces power consumption by a factor of ten on average, and runs in $O(n \log n)$ time with $O(n)$ space complexity.

References

[1] M. Buddhikot, S. Suri and M. Waldvogel, Space Decomposition Techniques for Fast Layer-4 switching, *Conference on High Speed Networks*, 1998.

[2] C. Faloutsos, Multiattribute Hashing Using Gray Codes, *Proc. ACM SIGMOD*, May 1986.

[3] A. Feldman and S. Muthukrishnan, Tradeoffs for Packet Classification, *INFOCOM*, 2000.

[4] F. Baboescu and G. Varghese, Fast and Scalable Conflict Detection for Packet Classifiers, *10th IEEE International Conference on Network Protocols (ICNP'02)*, 2002.

[5] F. Baboescu, S. Singh and G. Varghese, Packet Classification for Core Routers: Is there an alternative to CAMs? *IEEE INFOCOM*, 2003.

[6] P. Gupta and N. McKeown, Packet Classification Using Hierarchical Intelligent Cuts, *ACM SIGCOMM*, 1999.

[7] H.V. Jagadish, Linear Clustering of Objects with Multiple Attributes, *ACM SIGMOD*, 1990.

[8] I. Kamel and C. Faloutsos, On Packing R-trees, *Second Int. Conference on Information and Knowledge Management (CIKM)*, 1993.

[9] T. Lakshman and D. Stidialis, High Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching, *ACM SIGCOMM*, 1998.

[10] J.K. Lawder, Calculation of Mappings Between One and n-dimensional Values Using the Hilbert Space-filling Curve, Research Report JL1/00, School of Computer Science and Information Systems, Birkbeck College, University of London, 2000.

[11] H. Lu, Improved Trie Partitioning for Cooler TCAMs. *IASTED International Conference on Advances in Computer Science and Technology (ACST)*, 2004.

[12] H. Lu and S. Sahni, $O(\log W)$ Multidimensional Packet Classification. *IEEE/ACM Transactions on Networking*, to appear.

[13] J. Orenstein, Spatial Query Processing in an Object-Oriented Database System, *Proc. ACM SIGMOD*, 1986.

[14] R. Panigrahy and S. Sharma, Reducing TCAM Power Consumption and Increasing Throughput. *10th Symposium on High Performance Interconnects HOT Interconnects (HotI'02)*, 2002.

[15] E. Spitznagel, D. Taylor, and J. Turner, Packet Classification Using Extended TCAMs, *Proceedings of ICNP*, 2003.

[16] V. Srinivasan, A Packet Classification and Filter Management System, *IEEE INFOCOM*, 2001.

[17] X. Sun, S. Sahni, and Y. Zhao, Packet Classification Consuming Small Amount of Memory. *IEEE/ACM Transactions on Networking*, to appear.

[18] D. Taylor and J. Turner, ClassBench: A Packet Classification Benchmark, *IEEE INFOCOM*, 2005.

[19] F. Zane, G. Narlikar, and A. Basu. CoolCAMs: Power-efficient TCAMs for Forwarding Engines. *IEEE INFOCOM*, 2003.

1-4244-0357-X/06/$20.00 ©2006 IEEE

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2006 proceedings.