

# Dynamic Multi-Field Packet Classification

Authors: Jan van Lunteren, Ton Engbersen

IBM Zurich Research Laboratory  
Säumerstrasse 4  
CH-8803 Rüschlikon, Switzerland  
[jvl@zurich.ibm.com](mailto:jvl@zurich.ibm.com), [apj@zurich.ibm.com](mailto:apj@zurich.ibm.com)

Published in: Proceedings of the IEEE Global Telecommunications Conference  
GLOBECOM'02, vol. 3, pp. 2215 -2219, Taipei, Taiwan,  
November 2002.

© 2002 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

# Dynamic Multi-Field Packet Classification

Jan van Lunteren and Ton Engbersen  
IBM Research, Zurich Research Laboratory  
CH-8803 Rüschlikon, Switzerland

*Abstract*— Emerging Internet applications create the need for advanced packet classifiers. This paper investigates the mechanisms that determine the performance of state-of-the-art multi-field classification methods, and proposes a novel scheme called  $P^2C$  for packet classification at OC-192 and OC-768 speeds.  $P^2C$  combines the strengths of embedded memory and ternary CAM technologies to achieve very high storage efficiency while maintaining fast incremental updates. Key feature of the new scheme is its ability to adapt to the complexity of a classification rule set, while providing effective control over the update dynamics and storage requirements at the granularity of individual rules. This makes  $P^2C$  suitable for a broad range of applications.

## I. INTRODUCTION

Advanced packet classifiers that are capable of examining packets at full wire-speed against large and dynamic sets of complex classification rules are essential building blocks for realizing important emerging Internet applications such as quality of service, web-server load balancing, and firewalls. Whereas the problem of searching single packet fields (e.g., routing table lookup) is considered to be solved, classification on multiple fields remains a difficult problem [1]. This is due to the multi-dimensional nature of the searches in combination with the large number of bits, often on the order of hundreds, that have to be inspected for each packet.

Meeting the wire-speed challenge in a cost-efficient manner requires classifiers that are also highly storage-efficient. This is necessary because SDRAM performance cannot keep pace with the rapidly growing link speeds, which forces classifiers to use faster memory technologies, such as SRAM, embedded DRAM and ternary CAM (TCAM), which are substantially more expensive and have significantly smaller storage capacities. To make things even more challenging, the dynamic nature of several new applications also requires improved update rates, which typically is a conflicting goal [2].

This paper presents a novel scheme for multi-field packet classification at wire speed. The Parallel Packet Classification ( $P^2C$ ) scheme employs a new encoding concept which exploits the strengths of embedded memory and TCAM technologies to achieve improved storage efficiency over conventional methods, in combination with fast incremental updates.

## II. CLASSIFICATION RULES

Multi-field classification rules specify conditions for a selected set of packet fields, conformal with a given application. A rule is said to be matching when all conditions are met by the packet being classified. If multiple rules are matching, then the highest-priority rule will be selected, and packet processing will be based on information associated with that rule. An overview of the fields and operators used in actual classification rules is given in [3].

## III. CONVENTIONAL CLASSIFICATION SCHEMES

This section examines the mechanisms that affect the performance of state-of-the-art multi-field classification schemes. It will focus specifically on the multi-dimensional aspect by investigating the relations between the searches on the individual fields. Three cases are distinguished, and discussed in the following subsections.

### A. Conversion into Single-Field Search

The multi-dimensional nature of the search operation can be removed by combining the various fields into one search key and treating the problem as a single-field search. This approach is commonly used by hashing-based classifiers for flow identification [4] and by TCAM-based classifiers [5]. Its main advantage is that existing algorithms for one-dimensional searches can be used, several of which support fast incremental updates. Its main disadvantage is that it can only be efficiently applied to rule sets with exact-match conditions (hashing) or with exact- and prefix-match conditions (TCAMs). For example, the tuple space search scheme [6] needs multiple hash tables to support prefix-match conditions, while arbitrary range-match conditions require multiple TCAM entries. Other disadvantages are the large size of the composite search key, and the inability to optimize based on the occurrence of identical field conditions in multiple rules (as will be discussed later) because the individual fields are no longer visible. Examples of other schemes applying this concept are described in [7]-[9].

### B. Dependent Field Searches

Hierarchical tries and set-pruning tries are examples of classifiers employing dependent field searches, i.e., the results of fields that have already been searched influence the way subsequent fields will be searched. The main advantage of this approach is that well-known and relatively simple tree-search algorithms can be used. Its main disadvantage is that fast search times can only be achieved by replicating parts of the tree structure (e.g., set-pruning tries [1]) or by inserting additional linking information (e.g., grid-of-tries [10]), either of which results in larger storage requirements and a more complex update operation. The HiCuts scheme, presented in [11], applies heuristics to achieve fast update rates, but cannot provide any worst-case bounds. Examples of other schemes applying dependent field searches can be found in [12]-[14].

### C. Independent Field Searches

Fig. 1 shows the concept of a classifier in which the packet fields are searched independently, using conventional one-

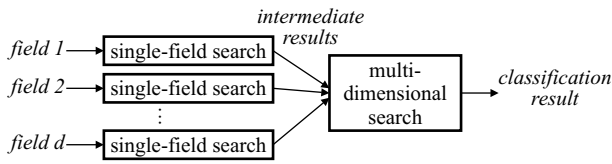


Fig. 1. Independent field searches.

dimensional searches, and the classification result is determined by a multi-dimensional search on the intermediate search results. The rationale behind this approach is that the multi-dimensional search in Fig. 1 can be simplified through proper encoding of the intermediate search results, as compared with the multi-dimensional searches discussed in the previous subsections. The independent field searches can, in this respect, be regarded as a “preprocessing” step.

An important advantage of this approach is that each unique field condition needs to be inserted only once in the corresponding (single-field) search structure, resulting in improved storage efficiency and related performance gains for rule sets with many identical field conditions. The overall performance, however, strongly depends on the encoding of the intermediate search results and the (related) multi-dimensional search. This will be illustrated using the two-dimensional classifier shown in Fig. 2. This is a variation on an example presented in [15], and shows four rules represented as rectangles in a two-dimensional space, where the ranges covered by the rules in each dimension result from the exact-, prefix-, or range-match conditions on the corresponding field. The sets of nonoverlapping intervals  $X_0$  to  $X_8$  and  $Y_0$  to  $Y_6$  are obtained by projecting the rectangle boundaries onto the corresponding axes.

Fig. 2 also shows the intermediate result vectors for a classification scheme based on bitmap intersections as presented in [15]. Each vector includes one bit for each rule, which is set if the corresponding interval is part of the range related to that rule. The classification result is determined by performing a logical AND operation on the independent search results in both dimensions. Matching rules will show up as set bits in the AND product. In the case of multiple matches, the highest-priority rule is selected based on the bit order.

The recursive flow classification (RFC) scheme, presented in [3], also employs the concept of independent field searches, although the boundaries between the single-field and multi-field searches as shown in Fig. 1 are less obvious because all searches are implemented as several stages of parallel table lookups. According to the RFC approach, result vectors simply are interval identifiers (called “equivalence class IDs” in [3]). For example, intervals  $X_0$ ,  $X_1$ , and  $X_2$  in Fig. 2 could be assigned the binary numbers 000b, 001b, 010b as intermediate result vectors, and so on. Based on the interval identifiers found by the independent field searches, the multi-dimensional search will determine the classification result by accessing a data structure containing information on the highest-priority matching rule for each possible combination of interval identifiers. For example, this data structure would contain information that rule 3 is the highest-priority rule matching the interval

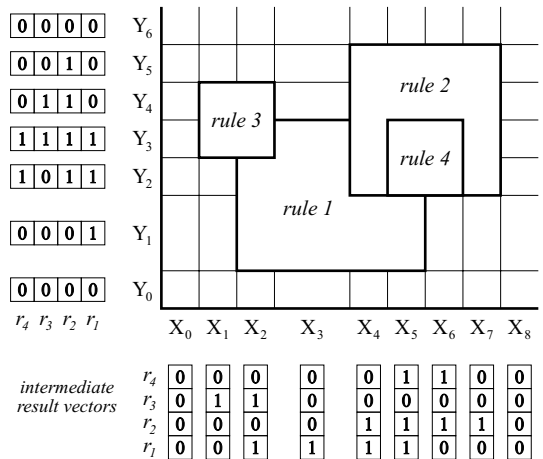


Fig. 2. Example of a two-dimensional classifier.

combinations  $\{X_1, Y_3\}$ ,  $\{X_1, Y_4\}$ ,  $\{X_2, Y_3\}$ , and  $\{X_2, Y_4\}$  for the classifier shown in Fig. 2.

The bitmap-intersection and RFC schemes are, in a sense, extremes. The bitmap-intersection scheme encodes all rule information (i.e., match conditions and rule priorities) in the intermediate result vectors, and the multi-dimensional search, which basically consists of a logical AND operation, does not need to retrieve any additional rule information from a separate data structure. The RFC scheme, on the other hand, encodes no rule information whatsoever in the intermediate result vectors. Instead, this information needs to be retrieved from the multi-dimensional search structure.

A main disadvantage of both encoding styles is that the storage requirements grow exponentially with the number of rules, while fast incremental updates are not supported, limiting their application to more static and relatively small rule sets only. The cross-production method presented in [10] and a modified version of the bitmap-intersection scheme presented in [16] have the same drawbacks.

#### IV. PARALLEL PACKET CLASSIFICATION ( $P^2C$ )

The  $P^2C$  scheme applies the concept of independent field searches as shown in Fig. 1 using a TCAM for the multi-dimensional search. The name *Parallel Packet Classification* indicates its focus on a hardware implementation supporting parallel field searches; however, other implementations are also feasible. The  $P^2C$  scheme uses an approach intermediate between the extremes of the bitmap-intersection and RFC schemes: Instead of encoding *all* rule information in the intermediate result vectors or storing it in the multi-dimensional search structure, the  $P^2C$  scheme encodes *part* of the rule information in the intermediate result vectors and stores the remaining *part*, including rule priority information, in the TCAM.

##### A. Encoding of Intermediate Search Results

Rule information is encoded in the intermediate search results based on the concept of primitive ranges, which are col-

lections of adjacent intervals. The encoding is done such that each primitive range can be identified by one ternary match condition on the intermediate result vectors. This will now be explained using several encoding examples for the result vectors in the X-dimension of the classifier shown in Fig. 2.

The first example will illustrate how this concept can be used to generate the result vectors for the bitmap-intersection scheme, which are also shown in Fig. 2. Fig. 3(a) shows four primitive ranges, which equal the actual ranges covered in the X-dimension by the four rules in Fig. 2. The primitive ranges are organized in a hierarchical structure, comprised of four layers that each correspond to one bit position in the result vector (this bit position is shown in brackets to the right of each layer). Each primitive range is assigned an identifier equal to 1 (binary). The remaining part of each layer, which is not part of a primitive range, is assigned a zero identifier (not shown in Fig. 3). The intermediate result vectors are now obtained by determining the set of primitive ranges in which each interval is located, and by substituting the primitive range identifiers at the bit positions associated with the corresponding layers. For example, the result vector corresponding to interval  $X_2$  equals 0101 because  $X_2$  is part of the primitive range at layer 1 (resulting in the set bit at the bit position 0) as well as of the primitive range at layer 3 (the set bit at the bit position 2). Table I (first row) lists all result vectors that are derived in this way and which equal those shown in Fig. 2 (note that the zero result vectors corresponding to intervals  $X_0$  and  $X_8$  are omitted). Table II lists the corresponding ternary match conditions for all primitive ranges related to each rule. For example, the ternary match condition  $xx1x$  for the primitive range related to rule 2 (also at layer 2 in Fig. 3(a)) will only match the result vectors corresponding to the intervals that are part of this primitive range:  $X_4$ ,  $X_5$ ,  $X_6$ , and  $X_7$ .

Fig. 3(b) shows the first type of  $P^2C$  encoding, which is based on grouping as many nonoverlapping primitive ranges as possible at the same layer, and assigning those unique (nonzero) identifiers. The number of result-vector bits associated with each layer is increased to match the identifier sizes. The corresponding result and ternary vectors are also listed in Tables I and II. This encoding style yields shorter intermediate result vectors and ternary match conditions for rule sets with many nonoverlapping ranges, because the number of layers will decrease linearly with the number of ‘grouped’ primitive ranges whereas the number of result-vector bits per layer will only grow logarithmically. In this case, the result-vector size remained four bits, owing to the small number of rules in this example. However, as can be seen, there would be room for a fifth (nonoverlapping) primitive range at layer 3, which could be assigned identifier 11, without increasing the result-vector size. A fifth rule would always require an additional layer and a result-vector bit in Fig. 3(a).

The second type of  $P^2C$  encoding reduces the result-vector sizes by exploiting relations between primitive ranges: Two primitive ranges at the same layer can be assigned a common identifier, if both ranges are subsets of two disjoint primitive ranges at other layers, and the (nonidentical) identifiers of the

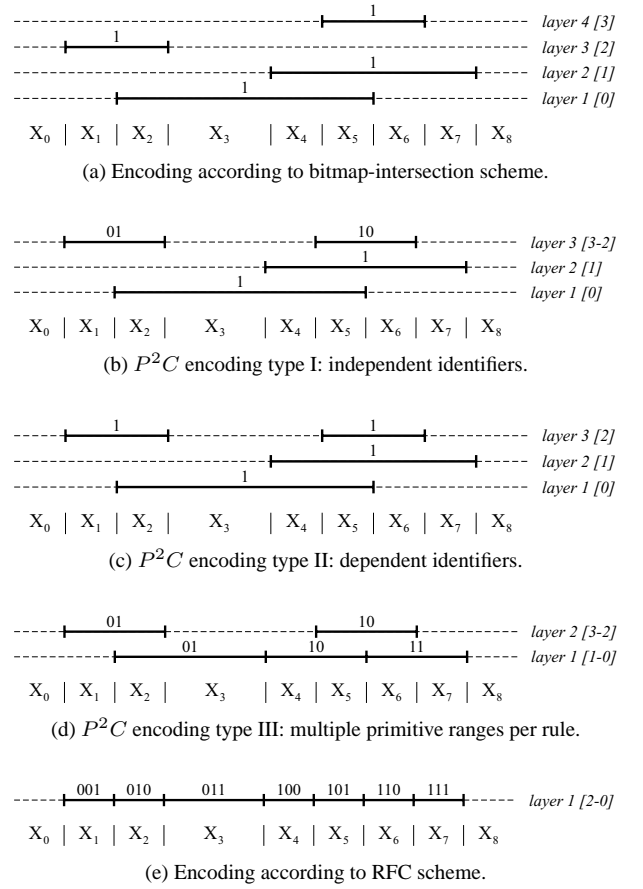


Fig. 3. Primitive Range Hierarchies.

TABLE I  
INTERMEDIATE RESULT VECTORS.

Fig.	$X_1$	$X_2$	$X_3$	intervals			
				$X_4$	$X_5$	$X_6$	$X_7$
3(a)	0100	0101	0001	0011	1011	1010	0010
3(b)	0100	0101	0001	0011	1011	1010	0010
3(c)	100	101	001	011	111	110	010
3(d)	0100	0101	0001	0010	1010	1011	0011
3(e)	001	010	011	100	101	110	111

TABLE II  
TERNARY MATCH CONDITIONS.

Fig.	rule 1	rule 2	rule 3	rule 4
3(a)	xxx1	xx1x	x1xx	1xxx
3(b)	xxx1	xx1x	01xx	10xx
3(c)	xx1	x1x	10x	11x
3(d)	xx01, xx10	xx10, xx11	01xx	10x
3(e)	010 .. 101	100 .. 111	001, 010	101, 110

latter ranges are inserted in the appropriate ternary match conditions in addition to the common identifier. This is illustrated in Fig. 3(c), in which both primitive ranges at layer 3 are assigned a common identifier 1. The primitive range at the right-hand side of layer 3 is a subset of the primitive range at layer 2, with an identifier 1, resulting in a ternary match condition 11x. The other range at layer 3 is a subset of the ‘‘empty’’ portion of layer 2 which will be treated as a primitive range

with identifier 0, resulting in a ternary match condition 10x.

The third type of  $P^2C$  encoding reduces the number of layers by converting overlapping primitive ranges at different layers into a larger number of nonoverlapping primitive ranges at the same layer. This is illustrated in Fig. 3(d). The consequence of this encoding style is that multiple primitive ranges and corresponding ternary match conditions will relate to single rules (rules 1 and 2 in Table II). Repeated application of this approach will ultimately lead to a primitive range hierarchy consisting of a single layer as shown in Fig. 3(e), which corresponds to the RFC-scheme type of encoding. This clearly shows the position of the  $P^2C$  encoding types between the extremes of the bitmap-intersection and RFC schemes.

### B. Independent Field Searches

The individual fields can be searched based on two different approaches. The first involves a range search to determine the interval in which the field value is located and then outputting the corresponding intermediate result vector, which is precomputed and stored separately for each interval. This approach is used by the bitmap-intersection [15] and RFC schemes [3]. A major disadvantage is that information related to a single rule can be distributed over many result vectors, resulting in substantial storage requirements and poor update performance.

The second, and preferred, approach involves a search to determine the actual primitive ranges in which the field value is located and an *on-the-fly* construction of the result vector based on the matching primitive range identifiers and associated bit-position information. This improves both storage efficiency and update performance because the information related to a single rule is stored only once (assuming one primitive range per rule). The  $P^2C$  scheme uses the BART algorithm [17] (slightly modified to find all matching ranges) for searching the individual fields. This scheme achieves very low storage requirements by exploiting the wide data buses available in embedded memory technology (SRAM or DRAM).

### C. Ternary CAM Search

The TCAM contents are obtained by concatenating all combinations of ternary match conditions for all fields that relate to the same rule, and storing those in the TCAM ordered according to the rule priorities. The input for the TCAM search consists of the concatenated result vectors of the independent field searches. The rule corresponding to the matching TCAM entry with the highest priority is the classification result.

### D. Incremental Updates

Classifier updates (e.g., rule inserts and deletes) require modification of the primitive range hierarchies, the field search structures, and the TCAM contents. The latter two types of data structures can be modified efficiently using the incremental update algorithms presented in [17] and [18]. The overall update performance, therefore, depends especially on an efficient modification of the primitive range hierarchies. This can be realized more easily with the first type of  $P^2C$  encoding

which creates fewer dependencies in the data structures than the other two encoding types. The third encoding type renders modifications even more complex because multiple entries in the field search structures and the TCAM can relate to the same rule. The second and third encoding types, however, achieve smaller storage requirements by compromising on update performance. The above observation in combination with the fact that the encoding type can be selected separately for each rule comprise a key feature of the  $P^2C$  scheme: update dynamics and storage efficiency can be balanced at the granularity of individual rules. This could, for example, be applied very well for firewalls that involve both static rules as well as dynamic rules that only exist during the lifetime of a connection.

## V. PERFORMANCE EVALUATION

### A. Classification and Update Performance

The field searches (using BART) and the TCAM search can be performed in a pipelined fashion, resulting in a classification rate that is mainly determined by the largest memory cycle time. Current memory technologies enable wire-speed classification for OC-192 and OC-768 links, which require maximum cycle times of about 38 and 10 ns, respectively. Performance evaluations have shown that the update rates feasible with BART and with current TCAM technology enable overall update rates of several hundred thousand per second for the first type of  $P^2C$  encoding. However, a detailed discussion and performance analysis of the update algorithms have to be postponed to a future paper because of space limitations.

### B. Storage Requirements

The existence of efficient single-field search algorithms in combination with an on-the-fly construction of the intermediate result vectors will render the overall storage complexity mainly dependent on the TCAM search. Worst-case figures will now be derived for the first type of  $P^2C$  encoding, which has the largest storage requirements of all three encoding types. This encoding type implies that the number of TCAM entries equals the number of rules, while the TCAM-entry width equals the sum of all intermediate result-vector sizes.

The result-vector size is determined by the total number of bits associated with the layers in the corresponding primitive range hierarchy. If there are  $n$  primitive ranges at a given layer, then a total of  $\lceil \log(n+1) \rceil$  bits will be associated with that layer. The largest result-vector size will be obtained when all ranges are distributed equally over all layers. Taking into account that the maximum number of primitive ranges equals the number of rules  $N$  and assuming a total of  $k$  layers, we obtain a maximum intermediate result-vector size of

$$I = k \left\lceil \log \left( \frac{N}{k} + 1 \right) \right\rceil \text{ bits.} \quad (1)$$

As each TCAM entry corresponds to  $d$  result vectors for a classification on  $d$  fields, and the total number of TCAM entries equals the number of rules  $N$ , the total TCAM storage requirements will be  $O(NdI)$  in the worst case.

The number of layers,  $k$ , depends directly on the *complexity* of a rule set, namely, the maximum number of (different) match conditions that *all* overlap each other.

## VI. SIMULATIONS

Several simulations have been performed with (proprietary) rule sets of industrial firewall applications. Details of these rule sets have to be omitted for privacy reasons (which is a problem shared by most papers on classification schemes). The largest rule set, consisting of more than 1,500 rules, required 2 KB of either SRAM or DRAM storage to implement all field searches using BART, and 5 KB of TCAM storage to implement the multi-dimensional search. These small numbers, which are achieved by applying the first type of  $P^2C$  encoding, correspond to averages of 1.3 bytes of SRAM/DRAM storage and 3.1 bytes of TCAM storage per rule. This is less than the accumulated sizes of all fields involved in the classification (e.g., source and destination addresses, port numbers, protocol number, TCP acknowledgement), which exceed 13 bytes. Note that a conventional TCAM-based classifier requires at least 20 KB (1,500 times 13 bytes) of TCAM storage (which can be significantly more owing to arbitrary range-match conditions).

## VII. COMPARISON

Table III shows the worst-case storage complexity of various state-of-the-art classification schemes for rule sets with exact-, prefix- and arbitrary range-match conditions on  $d$  packet fields (unless indicated otherwise) with an average size of  $W$  bits. It also indicates whether these schemes exploit the occurrence of identical field conditions to optimize the storage requirements and whether fast incremental updates are supported.

The  $P^2C$  scheme is the only scheme that is able to optimize based on identical field conditions, while also supporting fast incremental updates. Furthermore, it is also the only scheme that can adapt to the complexity of a rule set, as represented by the parameter  $k$ , which equals the maximum number of match conditions that all overlap each other. Analysis of several rule sets has revealed that  $k$  is usually very small (typically less than ten), which is also consistent with other publications. For example, [16] reports a maximum of four overlapping rules matching the same packet.

TABLE III  
PERFORMANCE COMPARISON.

method	w.c. storage complexity	optimizes ident. fields	fast incr. updates
$P^2C$	$dNk \log(\frac{N}{k} + 1)$	yes	yes
RFC [3]	$N^d$	yes	no
bitmap-intersect. [15]	$dN^2$	yes	no
cross-producing [10]	$N^d$	yes	no
TCAM <sup>a</sup>	$NdW$	no	yes
tuple space <sup>ab</sup> [6]	$NdW$	no	yes
set-pruning tries [1]	$N^d dW$	no	no
grid-of-tries [10]	$NdW$	no	no
HiCuts [11]	$N^d$	no	yes

<sup>a</sup> storage-complexity shown for exact- and prefix-match conditions only

<sup>b</sup> requires searching  $O(W^d)$  hash tables in the worst case

Only three schemes in Table III have a storage complexity that grows linearly with the number of rules. However, the inability to optimize on identical field conditions can result in significant storage requirements for the TCAM (see Section VI) as well as for the tuple space search scheme (e.g., [6] reports 130 KB for only 278 firewall rules). The necessity to insert additional linking information (switch pointers) for realizing fast search times causes the same problem for the grid-of-tries approach (e.g., [10] reports 240 KB for 1000 rules).

## VIII. CONCLUSIONS

$P^2C$  is a novel multi-field classification scheme which exploits the strengths of embedded memory and TCAM technologies to achieve very high storage efficiency while supporting fast incremental updates. The  $P^2C$  scheme has the unique feature of being able to adapt to the complexity of a classification rule set while allowing to tune the storage requirements and update dynamics at the granularity of individual rules.

## REFERENCES

- [1] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE Network*, vol. 15, no. 2, pp. 24-32, March/April 2001.
- [2] C. Macián and R. Finthammer, "An evaluation of the key design criteria to achieve high update rates in packet classifiers," *IEEE Network*, vol. 15, no. 6, pp. 24-29, November/December 2001.
- [3] P. Gupta and N. McKeown, "Packet classification on multiple fields," *Computer Commun. Rev.*, vol. 29, no. 4, pp. 147-160, October 1999.
- [4] Z. Cao, Z. Wang, and E. Zegura, "Performance of hashing-based schemes for internet load balancing," *Proc. IEEE Infocom*, vol. 1, pp. 332-341, March 2000.
- [5] SiberCore Technologies, SiberCAM application note, [http://www.sibercore.com/pdf/an\\_scan001\\_1.pdf](http://www.sibercore.com/pdf/an_scan001_1.pdf).
- [6] V. Srinivasan, S. Suri, and G. Varghese, "Packet classification using tuple space search," *Computer Commun. Rev.*, vol. 29, no. 4, pp. 135-146, October 1999.
- [7] V. Srinivasan, "A packet classification and filter management system," *Proc. IEEE Infocom*, vol. 3, pp. 1464-1473, April 2001.
- [8] T.Y.C. Woo, "A modular approach to packet classification: algorithms and result," *Proc. IEEE Infocom*, vol. 3, pp. 1213-1222, March 2000.
- [9] J. Xu, M. Singhal, and J. Degroat, "A novel cache architecture to support layer four packet classification at memory access speeds," *Proc. IEEE Infocom*, vol. 3, pp. 1445-1454, March 2000.
- [10] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and scalable layer four switching," *Computer Commun. Rev.*, vol. 28, no. 4, pp. 191-202, October 1998.
- [11] P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," *Proc. Hot Interconnects 7*, August 1999.
- [12] A. Feldmann and S. Muthukrishnan, "Tradeoffs for packet classification," *Proc. IEEE Infocom*, vol. 3, pp. 1193-1202, March 2000.
- [13] P. Gupta and N. McKeown, "Dynamic algorithms with worst-case performance for packet classification," *Proc. IFIP Networking*, pp. 528-539, May 2000.
- [14] Ching-Fong Su, "High speed packet classification using segment tree," *Proc. IEEE Globecom*, vol. 1, pp. 582-586, November 2000.
- [15] T. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," *Computer Commun. Rev.*, vol. 28, no. 4, pp. 203-214, October 1998.
- [16] F. Baboescu and G. Varghese, "Scalable packet classification," *Proc. ACM SIGCOMM*, pp. 199-210, August 2001.
- [17] J. van Lunteren, "Searching very large routing tables in wide embedded memory," *Proc. IEEE Globecom*, vol. 3, pp. 1615-1619, November 2001.
- [18] D. Shah and P. Gupta, "Fast incremental updates on ternary-CAMs for routing lookups and packet classification," *Proc. Hot Interconnects 8*, pp. 145-153, August 2000.