This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2008 proceedings.

All-Match Based Complete Redundancy Removal for Packet Classifiers in TCAMs

Alex X. Liu Chad R. Meiners Yun Zhou
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48823, U.S.A.
{alexliu, meinersc, zhouyun}@cse.msu.edu

***Abstract*—Packet classification is the core mechanism that enables many networking services on the Internet such as firewall packet filtering and traffic accounting. Using Ternary Content Addressable Memories (TCAMs) to perform high-speed packet classification has become the de facto standard in industry. TCAMs classify packets in constant time by comparing a packet with all classification rules of ternary encoding in parallel.**

Despite their high speed, TCAMs suffer from the well-known interval expansion problem. As packet classification rules usually have fields specified as intervals, converting such rules to TCAM-compatible rules may result in an explosive increase in the number of rules. This is not a problem if TCAMs have large capacities. Unfortunately, TCAMs have very limited capacity, and more rules means more power consumption and more heat generation for TCAMs. Even worse, the number of rules in packet classifiers have been increasing rapidly with the growing number of services deployed on the internet.

The interval expansion problem of TCAMs can be addressed by removing redundant rules in packet classifiers. This equivalent transformation can significantly reduce the number of TCAM entries needed by a packet classifier. Our experiments on real-life packet classifiers show an average reduction of 58.2% in the number of TCAM entries by removing redundant rules.

In this paper, we propose an all-match based complete redundancy removal algorithm. This is the first algorithm that attempts to solve first-match problems from an all-match perspective. We formally prove that our redundancy removal algorithm guarantees no redundant rules in resulting packet classifiers. We conducted extensive experiments on both real-life and synthetic packet classifiers. These experimental results show that our redundancy removal algorithm is both effective in terms of reducing TCAM entries and efficient in terms of running time.

I. Introduction

Packet classification, which has been widely deployed on the Internet, is the core mechanism that enables routers to perform many networking services such as firewall packet filtering, virtual private networks (VPNs), network address translation (NAT), quality of service (QoS), load balancing, traffic accounting and monitoring, differentiated services (Diff-serv), etc. As more services are deployed on the Internet, packet classification grows in demand and importance.

The function of a packet classification system is to map each packet to a decision (*i.e.*, action) according to a sequence (*i.e.*, ordered list) of rules, which is called a *packet classifier*. Each rule in a packet classifier has a predicate over some packet header fields and a decision to be performed upon the packets that match the predicate. To resolve possible conflicts among rules in a classifier, the decision for each packet is the decision of the first (i.e., highest priority) rule that the packet matches. Table I shows an example packet classifier of three rules. The format of these rules is based upon the format used in Access Control Lists on Cisco routers.

The work of Alex X. Liu is supported in part by the National Science Foundation under Grant No. CNS-0716407.

A. Motivation

Using Ternary Content Addressable Memories (TCAMs) to perform high-speed packet classification has become the de facto standard in industry. A TCAM is a memory chip where each entry can store a packet classification rule that is encoded in ternary format. Given a packet, the TCAM hardware can compare the packet with all stored rules in parallel and then return the decision of the first rule that the packet matches. Thus, it takes $O(1)$ time to find the decision for any given packet. Current TCAMs can support up to 133 million searches per second for 144-bit wide keys [16]. Because of their high speed, TCAMs have become the de facto industrial standard for high speed packet classification [1]–[3], [16]. In 2003, most packet classification devices shipped were TCAM-based [4]. More than 6 million TCAM devices were deployed worldwide in 2004 [4].

Despite their high speed, TCAMs have their own limitations with respect to packet classification.

a) Interval expansion: TCAMs can only store rules that are encoded in ternary format. In a typical packet classification rule, source IP address, destination IP address, and protocol type are specified in prefix format, which can be directly stored in TCAMs, but source and destination port numbers are specified in intervals (*i.e.*, ranges), which need to be converted to one or more prefixes before being stored in TCAMs. This can lead to a significant increase in the number of TCAM entries needed to encode a rule. For example, 30 prefixes are needed to represent the single interval $[1, 65534]$, so $30 \times 30 = 900$ TCAM entries are required to represent the single rule r_2 in Table I.

b) Low capacity: TCAMs have limited capacity. The largest TCAM chip available on the market has 18Mb while 2Mb and 1Mb chips are most popular [4]. Given that each TCAM entry has 144 bits and a packet classification rule may have a worst expansion factor of 900, it is possible that an 18Mb TCAM chip cannot store all the required entries for a modest packet classifier of only 139 rules. While the worst

Rule	Source IP	Destination IP	Source Port	Destination Port	Protocol	Action
r_1	*	192.168.0.1	*	*	*	discard
r_2	1.2.3.0/24	192.168.0.1	[1,65534]	[1,65534]	TCP	accept
r_3	*	*	*	*	*	accept

TABLE I
AN EXAMPLE PACKET CLASSIFIER

case may not happen in reality, this is certainly an alarming issue. Furthermore, TCAM capacity is not expected to increase dramatically in the near future due to other limitations that we will discuss next.

c) High power consumption and heat generation: TCAM chips consume large amounts of power and generate large amounts of heat. For example, a 1Mb TCAM chip consumes 15-30 watts of power. Power consumption together with the consequent heat generation is a serious problem for core routers and other networking devices.

d) Large board space occupation: TCAMs occupy much more board space than SRAMs. For networking devices such as routers, area efficiency of the circuit board is a critical issue.

e) High hardware cost: TCAMs are expensive. For example, a 1Mb TCAM chip costs about $200 \sim 250$ U.S. dollars. TCAM cost is a significant fraction of router cost.

All these limitations of TCAMs can be addressed by reducing the number of TCAM entries that a packet classifier requires. As we reduce the number of TCAM entries required, we can use TCAMs of smaller capacities, which results in less board space and lower hardware cost. Furthermore, reducing the number of rules in a TCAM directly reduces power consumption and heat generation because the energy consumed by a TCAM grows linearly with the number of ternary rules it stores.

An effective way to reduce the number of TCAM entries that a packet classifier requires is to remove the redundant rules in the packet classifier. A rule in a packet classifier is *redundant* if and only if removing the rule does not change the semantics of the packet classifier. For example, rule r_2 in the packet classifier in Table I is redundant because there is no packet whose first matching rule is r_2. Through this equivalent transformation of removing redundant rules, the number of TCAM entries needed by a packet classifier can be significantly reduced. Using the example of the packet classifier in Table I, removing redundant rules reduces the number of TCAM entries needed from 902 down to 2. Our experiments on real-life packet classifiers show an average reduction of 58.2% on the number of TCAM entries by removing redundant rules.

A key advantage of reducing TCAM entries by removing redundant rules is that it can be easily deployed because it does not require any modification of existing packet classification systems. In comparison, a number of previous interval expansion solutions require hardware and architecture modifications to existing packet classification systems, making their adoption by networking manufacturers and ISPs much less likely [16], [18], [21], [25].

B. Redundancy Removal

A rule that examines d-dimensional fields can be viewed as a d-dimensional object. Real-life packet classifiers are typically 4-dimensional or 5-dimensional. While identifying redundant rules in 1-dimension packet classifiers is simple, identifying redundant rules in multi-dimensional packet classifiers is by no means easy.

In this paper, we present an all-match based complete redundancy removal algorithm. This is the first algorithm that attempts to solve first-match problems from an all-match perspective. We formally prove that the resulting packet classifiers have no redundant rules after running our redundancy removal algorithm. We conducted extensive experiments on both real-life and synthetic packet classifiers. The experimental results show that our redundancy removal algorithm achieves an average compression ratio of 41.8% for TCAM entries.

In our previous workshop paper [17], we presented a first-match based redundancy removal algorithm. We have improved upon that previous work in several ways. First, the redundancy theorem becomes simpler. The redundancy theorem in [17] distinguishes upward and downward redundant rules, and detects them separately. In contrast, the redundancy theorem presented here gives a single criterion that can detect both upward and downward redundant rules. Second, the new redundancy removal algorithm is more efficient. The algorithm in [17] scans a packet classifier twice and build packet decision trees twice in order to remove the two types of redundant rules. In comparison, the new algorithm only scans a packet classifier once and build one all-match tree with similar cost of building a packet decision tree. The new algorithm is about twice as efficient as the algorithm in [17]. Third, in this paper, we extensively measured the effectiveness of redundancy removal on reducing TCAM entries, whereas no such empirical results were presented in [17].

Redundancy detection and removal have benefits beyond minimizing TCAM entries. One exemplary use of redundancy detection is in analyzing packet classifiers for potential errors. For instance, when a rule is shadowed by rules above it, the rule becomes redundant; however, this is typically not the intent of the router or firewall administrator. Therefore, redundancy could be an indicator of errors in packet classifiers.

The rest of this paper proceeds as follows. We start by reviewing previous work in Section II. Then, we formally define the terms and concepts related to redundancy removal in Section III. We introduce all-match trees and the redundancy theorem based on them in Section IV. In Section V, we present the algorithm for constructing all-match trees and the algorithm for removing redundant rules based on the all-match tree. In Section VI, we show the experimental results on both real-life and synthetic packet classifiers. Finally, we give concluding remarks in Section VII.

II. Related Work

Many software solutions have been proposed for finding the decision of the first rule that a packet matches in a given packet classifier (*e.g.*, [6], [7], [10], [13], [15], [19], [20], [22], [26], [27]). A comprehensive survey of this work is given in [24].

Little previous work has been done on redundancy removal other than our preliminary work in [17]. In [12], Gupta defined two types of redundant rules: backward redundant rules and forward redundant rules. A rule r in a packet classifier is backward redundant if and only if there exists another rule r' listed above r such that all packets that match r also match r'. A rule r in a packet classifier is forward redundant if and only if there exists another rule r' listed below r such that the following three conditions hold: (1) all packets that match r also match r', (2) r and r' have the same decision, (3) for each rule r'' listed between r and r', either r and r'' have the same decision, or no packet matches both r and r''. The backward and forward redundant rules defined in [12] are two special types of redundant rules. For example, in the example packet classifier in Figure 1, rules r_2 and r_3 are redundant, but they are neither backward redundant rules nor forward redundant rules.

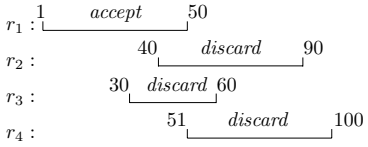

Fig. 1. A simple packet classifier

A significant amount of work explores ways to cope with the well-known interval expansion problem. These solutions fall into three broad categories: (1) *TCAM modification*, which requires changing TCAM hardware circuits, and (2) *range encoding*, which does not require changing TCAM hardware circuits, but does require preprocessing for every packet. and (3) *classifier minimization*, which does not require changing TCAM hardware circuits nor preprocessing for any packet. Next, we review previous work in these three categories.

TCAM Modification: The basic idea is to modify TCAM circuits for packet classification purposes. For example, Spitznagel *et al.* proposed adding comparators at each entry level to better accommodate range matching [21]. This is an important research direction. However, any solutions from this research line will not be deployed for many years due to issues of cost and development [16]. Furthermore, changing the ternary nature of TCAMs makes such TCAMs less generally applicable to applications other than packet classification.

Range Encoding: The basic idea is to re-encode intervals that appear in a packet classifier and then store the re-encoded rules in the TCAM. When a packet comes, the packet needs to be preprocessed according to the re-encoding scheme such that the packet, after preprocessing, can be used as a search key for the TCAM. Several range encoding schemes have been proposed [16], [18], [25]. While the TCAM circuit does not need to be modified to implement range encoding, the system hardware does need to be reconfigured to allow for preprocessing of packets, and the delay caused by packet preprocessing could be problematic.

Classifier Minimization: The basic idea is to convert a given packet classifier to another semantically equivalent packet classifier that requires fewer TCAM entries. These solutions are the most likely to be deployed by networking vendors and ISPs because they require no changes to TCAM hardware or existing packet classification systems and incur no preprocessing overhead for packets. Our work, along with [5], [8], [9], [23], falls into this category.

Three papers focus on one-dimensional and two dimensional packet classifiers. Draves *et al.* proposed an optimal solution for one-dimensional packet classifiers in the context of minimizing routing tables in [9]. Subsequently, in the same context of minimizing routing tables, Suri *et al.* proposed an optimal dynamic programming solution for one-dimensional packet classifiers. They also observed that a generalization of the dynamic program was optimal for two-dimensional packet classifiers in which two rules either are non-overlapping or one contains the other geometrically [23]. Suri *et al.* noted that their dynamic program would not be optimal for packet classifiers with more than 2 dimensions. In our studies, we have extended and implemented Suri *et al.*'s algorithm to minimize 5-dimensional packet classifiers. Unfortunately, the extended algorithm is prohibitively slow even for a packet classifier with just a few rules. Recently, Applegate *et al.* proposed an optimal solution for packet classifiers with two dimensions in which each rule must have one field specified as the whole domain of the field and there are only 2 decisions [5].

In [8], Dong *et al.* proposed a method for minimizing packet classifiers in TCAMs. In Dong's algorithm, the redundancy removal algorithm in our previous work [17] is used as a core routine that is repeatedly called.

III. Formal Definitions

We now formally define the concepts of fields, packets, rules, packet classifiers, and redundant rules. A *field* F_i is a variable of finite length (*i.e.*, of a finite number of bits). The domain of field F_i of w bits, denoted $D(F_i)$, is $[0, 2^w - 1]$. A *packet* over the d fields $F_1, \cdots, F_d$ is a d-tuple $(p_1, \cdots, p_d)$ where each p_i $(1 \leq i \leq d)$ is an element of $D(F_i)$. Packet classifiers usually check the following five fields: source IP address, destination IP address, source port number, destination port number, and protocol type. The length of these packet fields are 32, 32, 16, 16, and 8 respectively. We use Σ to denote the set of all packets over fields $F_1, \cdots, F_d$. It follows that Σ is a finite set and $|\Sigma| = |D(F_1)| \times \cdots \times |D(F_d)|$, where $|\Sigma|$ denotes the number of elements in set Σ and $|D(F_i)|$ denotes the number of elements in set $D(F_i)$.

A *rule* has the form $\langle predicate \rangle \rightarrow \langle decision \rangle$. A $\langle predicate \rangle$ defines a set of packets over the fields F_1 through F_d, and is specified as $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d$ where each S_i is a subset of $D(F_i)$ and is specified as either a prefix or a nonempty nonnegative integer interval. A *prefix* $\{0,1\}^k\{*\}^{w-k}$ with k leading 0s or 1s for a packet field of length w denotes the integer interval

$[\{0,1\}^k\{0\}^{w-k}, \{0,1\}^k\{1\}^{w-k}]$. For example, prefix 01** denotes the interval $[0100, 0111]$. A rule $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d \rightarrow \langle decision \rangle$ is a *prefix rule* if and only if each S_i is represented as a prefix.

When using a TCAM to implement a packet classifier, we typically require that all rules be prefix rules. However, in a typical packet classifier rule, some fields such as source and destination port numbers are represented as integer intervals rather than a prefix. This leads to *interval expansion* (also called *range expansion*), the process of converting a rule that may have fields represented as integer intervals into one or more prefix rules. In interval expansion, each field of a rule is first expanded separately. The goal is to find a minimum set of prefixes such that the union of the prefixes corresponds to the integer interval. For example, if one 3-bit field of a rule is the integer interval $[1, 6]$, a corresponding minimum set of prefixes would be 001, 01$*$, 10$*$, 110. The worst-case interval expansion of a $w-$bit integer interval results in a set containing $2w-2$ prefixes [14]. The next step is to compute the cross product of each set of prefixes for each field, resulting in a potentially large number of prefix rules. In Section I, the interval expansion of rule r_2 in Table I resulted in $30 \times 30 = 900$ prefix rules.

A packet $(p_1, \cdots, p_d)$ *matches* a predicate $F_1 \in S_1 \wedge \cdots \wedge F_d \in S_d$ and the corresponding rule if and only if the condition $p_1 \in S_1 \wedge \cdots \wedge p_d \in S_d$ holds. We use α to denote the set of possible values that $\langle decision \rangle$ can be. For firewalls, typical elements of α include accept, discard, accept with logging, and discard with logging.

A sequence of rules $\langle r_1, \cdots, r_n \rangle$ is *complete* if and only if for any packet p, there is at least one rule in the sequence that p matches. To ensure that a sequence of rules is complete and thus is a packet classifier, the predicate of the last rule is usually specified as $F_1 \in D(F_1) \wedge \cdots F_d \in \wedge D(F_d)$, which every packet matches. A *packet classifier* f is a sequence of rules that is complete. A packet classifier f is a *prefix packet classifier* if and only if every rule in f is a prefix rule.

Two rules in a packet classifier may overlap; that is, there exists at least one packet that matches both rules. Furthermore, two rules in a packet classifier may conflict; that is, the two rules not only overlap but also have different decisions. Packet classifiers typically resolve conflicts by employing a first-match resolution strategy where the decision for a packet p is the decision of the first (i.e., highest priority) rule that p matches in f. The decision that packet classifier f makes for packet p is denoted $f(p)$.

We can think of a packet classifier f as defining a many-to-one mapping function from Σ to α, where Σ denotes the set of all possible packets and α denotes the set of all possible decisions. Two packet classifiers f_1 and f_2 are *equivalent*, denoted $f_1 \equiv f_2$, if and only if they define the same mapping function from Σ to α; that is, for any packet $p \in \Sigma$, we have $f_1(p) = f_2(p)$. Using the concept of equivalent packet classifiers, we define redundant rules as follows.

Definition 1 (Redundant Rule): A rule r is redundant in a packet classifier f if and only if the resulting packet classifier f' after removing rule r is equivalent to f.

IV. All-Match Based Redundancy Theorem

In this section, we introduce the concept of all-match trees and the all-match based redundancy theorem.

A. All-Match Trees

Definition 4.1 (All-Match Tree): An *all-match tree* t for a packet classifier $f : \langle r_1, r_2, \cdots, r_n \rangle$ over fields $F_1, \cdots, F_d$ is a tree that has the following five properties:

1) Each node v is labeled with a packet field denoted $F(v)$. If v is a nonterminal node, then $F(v)$ is a packet field. If v is a terminal node, then $F(v)$ is a list of integer values $\langle i_1, i_2, \cdots, i_k \rangle$ where $1 \leq i_1 < i_2 \cdots < i_k \leq n$.
2) Each edge $e{:}u \rightarrow v$ is labeled with a nonempty set of integers, denoted $I(e)$, where $I(e)$ is a subset of the domain of u's label (*i.e.*, $I(e) \subseteq D(F(u))$).
3) The set of all outgoing edges of a node v in t, denoted $E(v)$, satisfies the following two conditions:
   a) *Consistency*: $I(e) \cap I(e') = \emptyset$ for any two distinct edges e and e' in $E(v)$.
   b) *Completeness*: $\bigcup_{e \in E(v)} I(e) = D(F(v))$.
4) A directed path from the root to a terminal node is called a *decision path*. No two nodes on a decision path have the same label. Given a decision path $\mathcal{P} : (v_1 e_1 v_2 e_2 \cdots v_m e_m v_{m+1})$, the matching set of $\mathcal{P}$ is defined as the set of all packets that satisfy $(F(v_1) \in I(e_1)) \wedge (F(v_2) \in I(e_2)) \wedge \cdots \wedge (F(v_m) \in I(e_m))$. We use $M(\mathcal{P})$ to denote the matching set of $\mathcal{P}$.
5) For any decision path $\mathcal{P} : (v_1 e_1 v_2 e_2 \cdots v_m e_m v_{m+1})$ where $F(v_{m+1}) = \langle i_1, i_2, ..., i_k \rangle$ and for any rule $r_j (1 \leq j \leq n)$, if $M(\mathcal{P}) \cap M(r_j) \neq \phi$, then $M(\mathcal{P}) \subseteq M(r_j)$ and $j \in \{i_1, i_2, \cdots, i_k\}$. □

Fig 3 shows an all-match tree for the simple packet classifier in Fig 2. In this example, we assume every packet has only two fields F_1 and F_2, and the domain of each field is $[1, 10]$.

$$
\begin{array}{lll}
r_1: & F_1 \in [1, \ \ 5] \wedge F_2 \in [1, 10] & \rightarrow accept \\
r_2: & F_1 \in [1, \ \ 5] \wedge F_2 \in [5, 10] & \rightarrow accept \\
r_3: & F_1 \in [6, \ 10] \wedge F_2 \in [1, 3] & \rightarrow discard \\
r_4: & F_1 \in [1, \ 10] \wedge F_2 \in [1, 10] & \rightarrow discard
\end{array}
$$

Fig. 2. A simple packet classifier

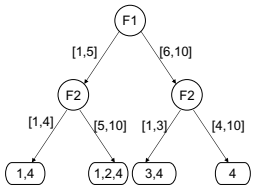

Fig. 3. An all-match tree for the packet classifier in Fig 2

In an all-match tree for a packet classifier f, for any decision path $\mathcal{P} : (v_1 e_1 v_2 e_2 \cdots v_m e_m v_{m+1})$ where $F(v_{m+1}) = \langle i_1, i_2, ..., i_k \rangle$, if a packet p satisfies this path $\mathcal{P}$, then $\{r_{i_1}, r_{i_2}, ..., r_{i_k}\}$ are exactly *all the rules in f that p matches*. This is why we call such a tree an "all-match tree".

The structure of an all-match tree is somewhat similar to that of a firewall decision diagram (FDD) [11]. However, they have two major differences. First, they are semantically different. Each all-match tree is associated with a particular packet classifier that is specified by a sequence of rules and it encodes the quantitative relationship among the rules in the packet classifier. In contrast, a firewall decision diagram describes the semantics of a packet classifier, just as a sequence of rules does. Second, they are syntactically different. Unlike all-match trees, a firewall decision diagram may not be a tree. Furthermore, in an all-match tree, a terminal node is labeled by a sequence of integer values; in contrast, in a firewall decision diagram, a terminal node is labeled by a decision for the packets that match the decision path from the root to the terminal node.

B. The All-Match Based Redundancy Theorem

Before we present the All-Match Based Redundancy Theorem, we first prove the following lemma.

Lemma 4.1: Let t be an all-match tree for packet classifier $f : \langle r_1, r_2, \cdots, r_n \rangle$. For any rule r_i in f, let $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_h$ be all the decision paths whose terminal node contains r_i, then the following condition holds: $M(r_i) = \bigcup_{j=1}^{h} M(\mathcal{P}_j)$. □

Proof:

(1) According to property 5 in the definition of all-match trees, we have $M(\mathcal{P}_j) \subseteq M(r_i)$ for every j $(1 \leq j \leq h)$. Thus, we have $\bigcup_{j=1}^{h} M(\mathcal{P}_j) \subseteq M(r_i)$.

(2) Consider a packet p in $M(r_i)$. According to the consistency and completeness properties of all-match trees, there exists one and only one decision path that p matches. Let $\mathcal{P}$ be this decision path. Thus, we have $p \in M(r_i) \cap M(\mathcal{P})$. According to property 5 in the definition of all-match trees, i is in the label of $\mathcal{P}$'s terminal node. Thus, we have $\mathcal{P} \in \{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_h\}$. Therefore, we have $p \in \bigcup_{j=1}^{h} M(\mathcal{P}_j)$. Thus we get $M(r_i) \subseteq \bigcup_{j=1}^{h} M(\mathcal{P}_j)$. □

Theorem 4.1 (All-Match Based Redundancy Theorem): Let t be an all-match tree for packet classifier $f : \langle r_1, r_2, \cdots, r_n \rangle$. Rule r_i is redundant in f if and only if in all terminal nodes of t that have i as their first value, i is immediately followed by another integer j such that r_i and r_j have the same decision. □

Proof:

(1) Suppose in all terminal nodes of t that have i as their first value i is immediately followed by another integer j such that r_i and r_j have the same decision. We next prove that r_i is redundant in f.

We observe that removing a rule r_i only possibly affects the decisions for the packets in $M(r_i)$. Let $\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_h$ be all the decision paths in t whose terminal node contains i. According to Lemma 4.1, we have $M(r_i) = \bigcup_{j=1}^{h} M(\mathcal{P}_j)$. Consider an arbitrary packet p in $M(r_i)$. Suppose we have $p \in M(\mathcal{P}_j)$. Let f' be the resulting packet classifier after removing r_i from f. To prove that r_i is redundant in f, we only need to prove $f(p) = f'(p)$. Let the label of the terminal node of $\mathcal{P}_j$ be $\langle i_1, i_2, \cdots, i_k \rangle$. Because $i \in \{i_1, i_2, \cdots, i_k\}$, there are two cases:

Case 1: $i_1 \neq i$. In this case, r_{i_1} is the first rule in f that p matches. Thus, removing r_i does not affect the decision for p. In this case, we have $f(p) = f'(p)$.

Case 2: $i_1 = i$, and r_i has the same decision as r_{i_2}. In f, r_i is the first rule that p matches. In f', r_{i_2} is the first rule that p matches. Because r_i and r_{i_2} has the same decision, we have $f(p) = f'(p)$ in this case.

Therefore, r_i is redundant in f.

(2) Suppose rule r_i is redundant in f and there exists a terminal node in t whose first two values are i followed by j. and r_i and r_j have different decisions. Let $\mathcal{P}$ denote the decision path from the root to this terminal node. Consider a packet $p \in M(\mathcal{P})$. Thus, r_i is the first rule that p matches in f and r_j is the first rule that p matches in f'. Because r_i and r_j have different decisions, we have $f(p) \neq f'(p)$. This conflicts with the assumption that r_i is redundant. Therefore, if r_i is redundant in f, then in all terminal nodes of t that have i as their first value i is immediately followed by another integer j such that r_i and r_j have the same decision. □

V. All-Match Based Redundancy Removal

In this section, we first present an algorithm for constructing all-match trees from packet classifiers. Second, we present a redundancy removal algorithm based on Theorem 4.1. Third, we prove that the resulting packet classifier does not have any redundant rules.

A. The All-Match Tree Construction Algorithm

According to Theorem 4.1, in order to detect and remove redundant rules in a packet classifier, we first need to construct an all-match tree for that packet classifier. The pseudocode for the all-match tree construction algorithm is shown in Figure 4.

All − Match Tree Construction Algorithm
Input : A packet classifier $f : \langle r_1, r_2, \cdots, r_n \rangle$.
Output : A all-match tree t for packet classifier f.
Steps:
1.Build a path from rule r_1. Let v denote the root.
The label of the terminal node is $\langle 1 \rangle$.
2.**for** $i := 2$ **to** n **do** APPEND(v, r_i, 1, i);
End

APPEND(v, $(F_1 \in S_1) \wedge \cdots \wedge (F_d \in S_d) \rightarrow \langle dec \rangle$, m, i)
/*$F(v) = F_m$ and $E(v) = \{e_1, \cdots, e_k\}$*/
1. **if** $(m = d + 1)$ **then**
Add i to the end of v's label;
Return;
2. **if** $(\ S_m - (\ I(e_1) \cup \cdots \cup I(e_k)\ )\ ) \neq \emptyset$ **then**
(a) Add an outgoing edge e_{k+1} with label $S_m - (I(e_1) \cup \cdots \cup I(e_k))$ to v;
(b) Build a decision path from $(F_{m+1} \in S_{m+1}) \wedge \cdots \wedge (F_d \in S_d) \rightarrow \langle dec \rangle$, and make e_{k+1} point to the first node in this path;
(c) Add i to the end of the label of the terminal node of this decision path;
3. **for** $j := 1$ **to** k **do**
if $I(e_j) \subseteq S_m$ **then**
APPEND(e_j's target,$(F_1 \in S_1) \wedge \cdots \wedge (F_d \in S_d) \rightarrow \langle dec \rangle$,m+1,i);
else if $I(e_j) \cap S_m \neq \emptyset$ **then**
(a) Add one outgoing edge e to v, and label e with $I(e_j) \cap S_m$;
(b) Make a copy of the subgraph rooted at the target node of e_j, and make e points to the root of the copy;
(a) Replace the label of e_j by $I(e_j) - S_m$;
(d) APPEND(e's target,$(F_1 \in S_1) \wedge \cdots \wedge (F_d \in S_d) \rightarrow \langle dec \rangle$,m+1,i);

Fig. 4. All-Match Tree Construction Algorithm

Consider the packet classifier in Figure 2. The process of constructing the corresponding all-match tree is shown in Figure 5.

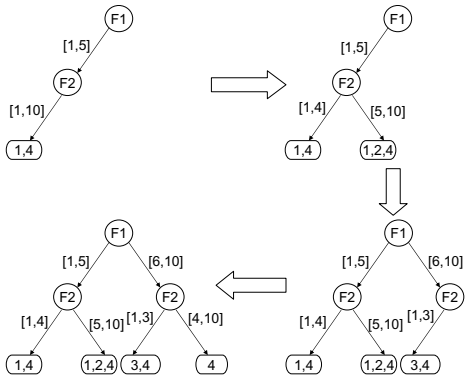

Fig. 5. Constructing an all-match tree

B. The All-Match Based Redundancy Removal Algorithm

We first introduce two auxiliary lists that are used in the all-match based redundancy removal algorithm: containment list and residency list. Given an all-match tree that has m terminal nodes, we assign a unique sequence number in $[1, m]$ to each terminal node. In the containment list, each entry consists of a terminal node sequence number and the rule sequence numbers contained in the terminal node. In the residency list, each entry consists of a rule sequence number and the set of terminal nodes which contains this rule. The all-match list and the residency list for the all-match tree in Figure 3 are in Figure 6.

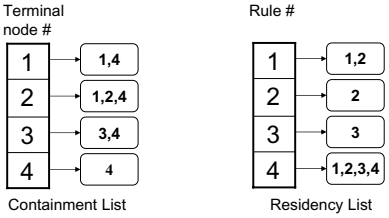

Fig. 6. The containment list and the residency List for the all-match tree in Figure 3

The all-match based redundancy removal algorithm works as follows. Given a packet classifier $f : \langle r_1, r_2, \cdots, r_n \rangle$, this algorithm scans f from r_n to r_1, and checks whether each rule is redundant using Theorem 4.1. Whenever a rule is detected as redundant, the rule is removed from f. The pseudocode of the algorithm is shown in Figure 7.

Consider the packet classifier in Figure 2 and its all-match tree in Figure 3. The all-match list and the residency list are in Figure 6. We next demonstrate the process of determining whether r_4 is redundant using the all-match based redundancy removal algorithm shown in Figure 7. From the residency list, we know that rule r_4 is contained in terminal nodes $1, 2, 3$ and 4. In terminal node 4, r_4 is the only value, and thus is not redundant. Next, we check whether r_3 is redundant. From the residency list, we know that r_3 is contained in terminal node 3. In terminal node 3, the first value is 3, and is immediately followed by a 4, and r_3 and r_4 have the same decision. According to the Theorem 4.1, r_3 is redundant. Subsequently,

All − Match Based Redundancy Removal Algorithm
Input : (1) A packet classifier $f : \langle r_1, r_2, \cdots, r_n \rangle$.
(2) A all-match tree t for f.
Output : A packet classifier f' where $f \equiv f'$ and there is no redundant rule in f'.
Steps:
1.Build the containment list $ConList[1..m]$ from t.
Build the residency list $ResList[1..n]$ from t.
2.**for** $i := n$ **to** 1 **do**
(1) redundant := true
(2) **for** each terminal node sequence number tn in $ResList[i]$ **do**
if (i is the only value in $ConList[tn]$) **or**
(i is the first value in $ConList[tn]$ and the second value in $ConList[tn]$, say j, satisfies the condition that r_i and r_j have different decisions)
then
redundant := false;
break;
(3) **if** redundant **then**
remove r_i from f;
for each terminal node sequence number tn in $ResList[i]$ **do**
delete i from $ConList[tn]$;

Fig. 7. All-Match Based Redundancy Removal Algorithm

we remove r_3 from the packet classifier and delete 3 from the third entry of the all-match list. In a similar fashion, we can further detect that r_2 is redundant and r_1 is not redundant. The resulting packet classifier is shown in Figure 8.

$$F_1 \in [1, \quad 5] \wedge F_2 \in [1, 10] \rightarrow accept$$
$$F_1 \in [1, \quad 10] \wedge F_2 \in [1, 10] \rightarrow discard$$

Fig. 8. The resulting packet classifier after removing redundant rules from the packet classifier in Figure 2

C. Proof of Complete Redundancy Removal

A packet classifier redundancy removal algorithm is a *complete redundancy removal algorithm* if and only if for any packet classifier the algorithm produces a semantically equivalent packet classifier in which no rule is redundant.

Theorem 5.1: The All-Match Based Redundancy Removal Algorithm is a complete redundancy removal algorithm. □

Proof:

Let f be a given packet classifier and Let t be an all-match tree for f. Let f'' be the resulting packet classifier after running the all-match based redundancy removal algorithm. Suppose f'' has a rule r_i that is redundant in f''. Let f' be the resulting packet classifier after the algorithm has examined all the rules from r_{i+1} to r_n and the redundant rules from r_{i+1} to r_n has been removed. Because the algorithm does not remove r_i, r_i is not redundant in f'. According to Theorem 4.1, there is at least a terminal node v that satisfies one of the following conditions:

1) this terminal node only contains i,
2) this terminal node has i as its first value and i is immediately followed by another value j such that r_i and r_j have different decisions.

If v satisfies one of the two conditions, then v still satisfies that condition after the algorithm removes all the redundant rules above r_i, because i will never be deleted from v according to the algorithm. Therefore, r_i is not redundant in f'' according to Theorem 4.1. □

It is worth noting that the order from n to 1 in detecting redundant rules is critical. If we choose another order, the algorithm may not be able to guarantee complete redundancy removal. Take the order from 1 to n as an example. When we check whether r_i is redundant, suppose r_i is not redundant because there is one and only one terminal node in the all-match tree that has i as its first value and i is immediately followed by another value j such that r_i and r_j have different decisions. We further suppose j is immediately followed by another value k where r_i and r_k have the same decision. After moving all the redundant rules after r_i, j is possibly removed from the terminal node and consequently r_i and r_k become the first two values in the terminal node and they have the same decision. Thus, r_i becomes redundant.

VI. Experimental Results

In this section, we evaluate the effectiveness and efficiency of the redundancy removal algorithm on both real-life and synthetic packet classifiers.

A. Effectiveness on Real-life Packet Classifiers

We first define the metrics that we used to measure the effectiveness of the redundancy removal algorithm. In this paragraph, f denotes a packet classifier, S denotes a set of packet classifiers, and RR denotes the redundancy removal algorithm. We then let $RR(f)$ denote the classifier produced by applying the redundancy removal algorithm on f, $Direct(f)$ denote the prefix classifier produced by applying direct interval expansion on f, and $|f|$ denote the number of rules in f. We define the following six metrics for assessing the performance of RR on a set of classifiers S.

- The *average compression ratio* of RR over $S = \frac{\Sigma_{f \in S} \frac{|Direct(RR(f))|}{|Direct(f)|}}{|S|}$.
- The *total compression ratio* of RR over $S = \frac{\Sigma_{f \in S} |Direct(RR(f))|}{\Sigma_{f \in S} |Direct(f)|}$.
- The *average expansion ratio* of RR over $S = \frac{\Sigma_{f \in S} \frac{|Direct(RR(f))|}{|f|}}{|S|}$.
- The *total expansion ratio* of RR over $S = \frac{\Sigma_{f \in S} |Direct(RR(f))|}{\Sigma_{f \in S} |f|}$.
- The *average expansion ratio* of *direct expansion* over $S = \frac{\Sigma_{f \in S} \frac{|Direct(f)|}{|f|}}{|S|}$.
- The *total expansion ratio* of *direct expansion* over $S = \frac{\Sigma_{f \in S} |Direct(f)|}{\Sigma_{f \in S} |f|}$.

We next define a set RL of 17 real-life packet classifiers that we performed experiments on. We actually obtained 42 real-life packet classifiers from distinct network service providers that range in size from dozens to hundreds of rules. Although this collection of classifiers is diverse, some classifiers from the same network service provider have similar structure and exhibited similar results under the redundancy removal algorithm. To prevent this repetition from skewing the performance data, we divided the 42 packet classifiers into 17 structurally distinct groups, and we randomly chose one from each of the 17 groups to form the set RL.

Compression Ratio of Real-life Packet classifiers. The average compression ratio of the redundancy removal algorithm over the packet classifiers in RL is 41.8%. The total compression ratio of the redundancy removal algorithm over the packet classifiers in RL is 35.0%. In Figure 9, we show the compression ratios of the redundancy removal algorithm for each packet classifier in RL in increasing order. Figure 10 shows the distribution of compression ratios achieved by the redundancy removal algorithm over the packet classifiers in RL. From these experimental results, we can see that our redundancy removal algorithm can significantly reduce the number of TCAM entries for a packet classifier.

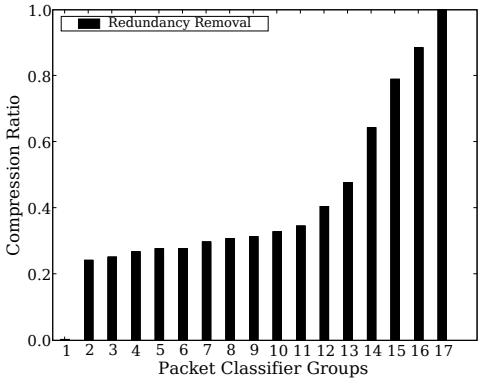

Fig. 9. Compression ratios of real-life packet classifier groups

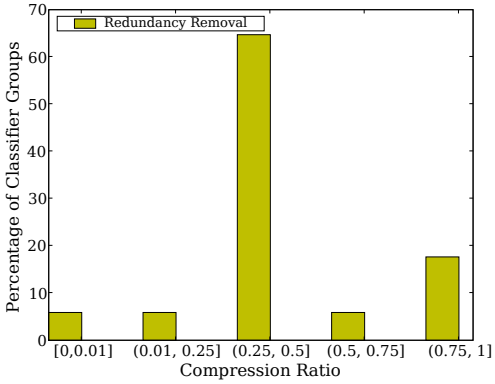

Fig. 10. Distribution of real-life packet classifiers by compression ratio

Expansion Ratio of Real-life Packet classifiers. The average expansion ratios of the redundancy removal algorithm and the direct expansion algorithm over the packet classifiers in RL are 19.9 and 69.9 respectively. The total expansion ratios of the redundancy removal algorithm and the direct expansion algorithm over the packet classifiers in RL are 7.1 and 20.4 respectively. In Figure 11, we show the expansion ratios of the redundancy removal algorithm and the direct expansion algorithm for each packet classifier in RL. Figure 12 shows the distribution of expansion ratios achieved by the redundancy removal algorithm and the direct expansion algorithm on the packet classifiers in RL. Interval expansion is indeed a real

issue as over 60% of our packet classifiers have an expansion ratio of over 50 if we use direct interval expansion. From these experimental results, we can see that redundancy removal can significantly reduce the expansion ratio of packet classifiers.

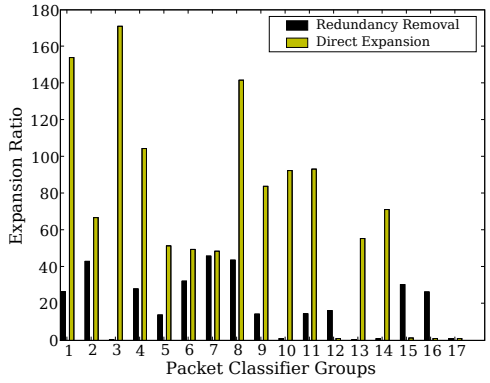

Fig. 11. Expansion ratios of real-life packet classifier groups

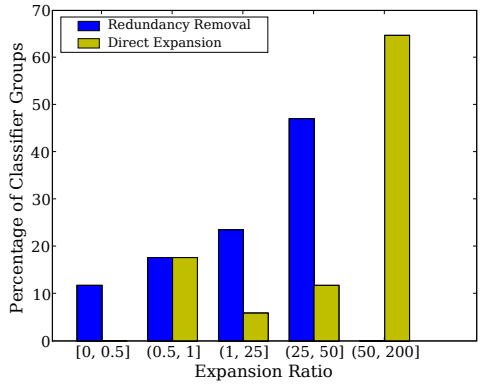

Fig. 12. Distribution of real-life packet classifiers by expansion ratio

B. Effectiveness on Synthetic Packet Classifiers

Packet classifier rules are considered confidential due to security concerns. Thus, it is difficult to get many real-life packet classifiers to experiment with. To address this issue and further evaluate the effectiveness of our redundancy removal algorithm, we generated a set of synthetic packet classifiers, denoted SYN, in the following fashion. Every predicate of a rule in our synthetic packet classifiers has five fields: source IP address, destination IP address, source port number, destination port number, and protocol type. We first randomly generated a list of values for each field. For IP addresses, we generated a random class C address; for ports we generated a random interval; for protocols, we generated a random protocol number. Given these lists, we generated a list of predicates by taking the cross product of all these lists. We added a final default predicate to our list. Finally, we randomly assigned one of two decisions, accept or deny, to each predicate to make a complete rule.

Compression Ratio of Synthetic Packet classifiers. The average compression ratios of the redundancy removal algorithm over the packet classifiers in SYN is 35.2%. The total compression ratios of the redundancy removal algorithm over the packet classifiers in SYN is 24.4%. Figure 13 shows the distribution of compression ratios achieved by the redundancy removal algorithm over the packet classifiers in SYN.

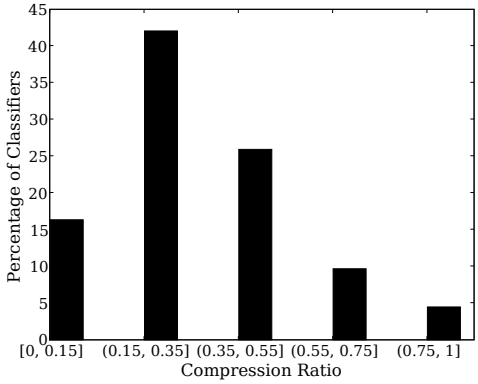

Fig. 13. Distribution of synthetic packet classifiers by compression ratio

Expansion Ratio of Synthetic Packet classifiers. The average expansion ratios of the redundancy removal algorithm and the direct expansion algorithm over the packet classifiers in SYN are 60.1 and 178.0 respectively. The total expansion ratios of the redundancy removal algorithm and the direct expansion algorithm over the packet classifiers in SYN are 45.8 and 196.1 respectively. Figure 14 shows the distribution of expansion ratios achieved by the redundancy removal algorithm and the direct expansion algorithm on the packet classifiers in SYN.

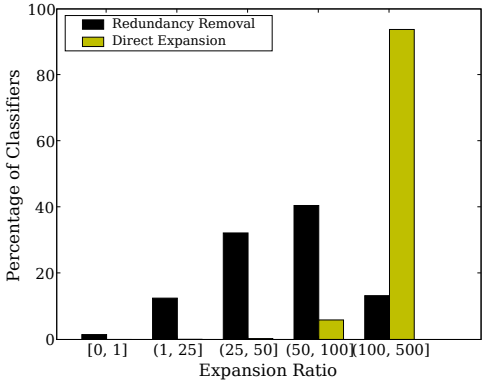

Fig. 14. Distribution of synthetic packet classifiers by expansion ratio

C. Efficiency on Real-life Packet Classifiers

We implemented our algorithm using SUN Java JDK 1.4. Our experiments were carried out on a desktop PC running Windows XP with 1G memory and a single 2.2 GHz AMD Opteron 148 processor. Table II shows the running time of the first-match based redundancy removal algorithm and

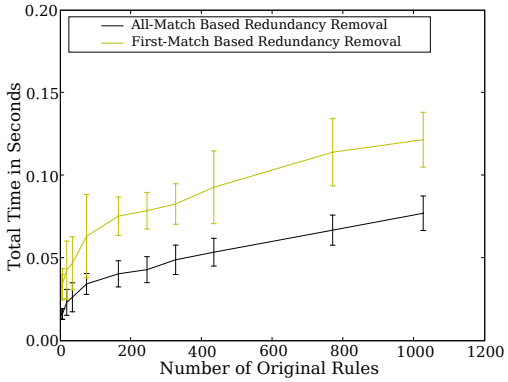

Fig. 15. Running time of both all-match based and first-match based redundancy removal algorithms vs. number of original rules

the all-match based redundancy removal algorithm for three representative packet classifiers.

Number of Rules	First-Match (sec)	All-Match (sec)
42	0.491	0.171
87	0.179	0.047
661	1.105	0.750

TABLE II
SAMPLE RUNNING TIME DATA FOR REAL-LIFE PACKET CLASSIFIERS

D. Efficiency on Synthetic Packet Classifiers

Figure 15 shows the average running time of the all-match based redundancy removal algorithm in comparison with the first-match based redundancy removal algorithm on the synthetic packet classifiers. We can see that the all-match based redundancy removal algorithm is about twice as efficient as the first-match based redundancy removal algorithm.

VII. Concluding Remarks

Our contributions are three-fold. First, we present a new concept called all-match trees, and develop a new redundancy theorem based on such trees. Second, we propose an all-match based redundancy removal algorithm, which guarantees that the resulting packet classifier has no redundant rules. This algorithm shows the promise of solving first-match problems from an all-match perspective. Last, we conducted extensive experiments on both real-life and synthetic packet classifiers. The experimental results show that our redundancy removal algorithm can effectively reduce TCAM entries by an average of 58.2%.

The results in this paper can be extended for use in many systems where a system can be represented by a sequence of rules. Examples of such systems are rule-based systems in the area of artificial intelligence and access control in the area of databases. In these systems, we can extend the results in this paper to remove redundant rules and thereby make the systems more efficient.

References

[1] Cypress semiconductor corp. content addressable memory. http://www.cypress.com/.
[2] Integrated device technology, inc. content addressable memory. http://www.idt.com/.
[3] Netlogic microsystems. content addressable memory. http://www.netlogicmicro.com/.
[4] A guide to search engines and networking memory. *http://www.linleygroup.com/pdf/NMv4.pdf*, November 2006.
[5] D. A. Applegate, G. Calinescu, D. S. Johnson, H. Karloff, K. Ligett, and J. Wang. Compressing rectilinear pictures and minimizing access control lists. In *Proceedings of the Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, January 2007.
[6] F. Baboescu, S. Singh, and G. Varghese. Packet classification for core routers: Is there an alternative to CAMs? In *Proceedings of IEEE INFOCOM*, 2003.
[7] F. Baboescu and G. Varghese. Scalable packet classification. In *Proceedings of ACM SIGCOMM*, pages 199–210, 2001.
[8] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, and A. Shukla. Packet classifiers in ternary CAMs can be smaller. In *Proceedings of SIGMETRICS*, pages 311–322, 2006.
[9] R. Draves, C. King, S. Venkatachary, and B. Zill. Constructing optimal IP routing tables. In *Proceedings of IEEE INFOCOM*, pages 88–97, 1999.
[10] A. Feldmann and S. Muthukrishnan. Tradeoffs for packet classification. In *Proceedings of 19th IEEE INFOCOM*, Mar. 2000.
[11] M. G. Gouda and A. X. Liu. Structured firewall design. *Computer Networks Journal*, 51(4):1106–1120, March 2007.
[12] P. Gupta. *Algorithms for Routing Lookups and Packet Classification*. PhD thesis, Stanford University, 2000.
[13] P. Gupta and N. McKeown. Packet classification on multiple fields. In *Proceedings of ACM SIGCOMM*, pages 147–160, 1999.
[14] P. Gupta and N. McKeown. Algorithms for packet classification. *IEEE Network*, 15(2):24–32, 2001.
[15] T. V. Lakshman and D. Stiliadis. High-speed policy-based packet forwarding using efficient multi-dimensional range matching. In *Proceedings of ACM SIGCOMM*, pages 203–214, 1998.
[16] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary. Algorithms for advanced packet classification with ternary cams. In *Proceedings of the ACM SIGCOMM*, pages 193 – 204, August 2005.
[17] A. X. Liu and M. G. Gouda. Complete redundancy detection in firewalls. In *Proceedings of 19th Annual IFIP Conference on Data and Applications Security, LNCS 3654, S. Jajodia and D. Wijesekera Ed., Springer-Verlag*, pages 196–209, August 2005.
[18] H. Liu. Efficient mapping of range classifier into ternary-cam. In *Proceedings of the Hot Interconnects*, pages 95– 100, 2002.
[19] L. Qiu, G. Varghese, and S. Suri. Fast firewall implementations for software-based and hardware-based routers. In *Proceedings the 9th International Conference on Network Protocols (ICNP)*, 2001.
[20] S. Singh, F. Baboescu, G. Varghese, and J. Wang. Packet classification using multidimensional cutting. In *Proceedings of ACM SIGCOMM*, 2003.
[21] E. Spitznagel, D. Taylor, and J. Turner. Packet classification using extended tcams. In *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP)*, pages 120– 131.
[22] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and scalable layer four switching. In *Proceedings of ACM SIGCOMM*, pages 191–202, 1998.
[23] S. Suri, T. Sandholm, and P. Warkhede. Compressing two-dimensional routing tables. *Algorithmica*, 35:287–300, 2003.
[24] D. E. Taylor. Survey & taxonomy of packet classification techniques. *ACM Computing Surveys*, 37(3):238–275, 2005.
[25] J. van Lunteren and T. Engbersen. Fast and scalable packet classification. *IEEE Journals on Selected Areas in Communications*, 21(4):560– 571, 2003.
[26] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner. Scalable high speed IP routing lookups. In *Proceedings of ACM SIGCOMM*, pages 25–36, September 1997.
[27] T. Y. C. Woo. A modular approach to packet classification: Algorithms and results. In *Proceedings of IEEE INFOCOM*, pages 1213–1222, 2000.