Directions in Packet Classification for Network Processors

Michael E. Kounavis, Alok Kumar, Harrick Vin, Raj Yavatkar and Andrew T. Campbell

***Abstract*--To classify a packet as belonging to a flow often requires network systems—such as routers and firewalls—to maintain large data structures and perform several memory accesses. Network processors, on the other hand, are generally configured with only a small amount of memory with limited access bandwidth. Hence, a key challenge is to design packet classification algorithms that can be implemented efficiently on network processor platforms. We conjecture that the design of such algorithms will need to exploit the structure and characteristics of packet classification rules. In this paper, we analyze several databases of classification rules found in firewalls and derive their statistical properties. Our analysis yields three main conclusions. (1) The rules found in classification databases contain two types of fields—source-destination IP address pairs that identify network paths and transport-level fields that characterize network applications; further, the databases contain many more network paths than applications. (2) IP address pairs identify regions in a two-dimensional space that overlap with each other; however, the number of overlaps is significantly smaller than the theoretical upper-bound. (3) Only a small number of transport-level fields are sufficient to characterize databases of different sizes. We justify our findings based on several standard practices employed by network administrators, and thereby argue that although our findings are for specific databases, the properties are likely to hold for most databases. Based on these findings, we suggest a classification architecture that a can be implemented efficiently on network processors.**

I. INTRODUCTION

PACKET classification involves identifying flows from among a stream of packets that arrive at routers. It is a fundamental building block that enables routers to support access control, Quality of Service differentiation, virtual private networks, and other value added services. To be classified as belonging to a flow, each packet arriving at a router is compared against a set of rules. Each rule contains one or more fields and their associated values, a priority, and an action. The fields generally correspond to specific portions of the TCP/IP header—such as the source and destination IP addresses, port numbers, and protocol identifier. A packet is said to match a rule if it matches every field in that rule. On identifying the matching rules, actions associated with the rules are executed.

Michael E. Kounavis and Andrew T. Campbell ({mk, campbell}@comet.columbia.edu) are affiliated with the COMET Group, Columbia University. Harrick Vin (vin@cs.utexas.edu) is affiliated with the University of Texas at Austin. Alok Kumar and Raj Yavatkar ({alok.kumar, raj.yavatkar}@intel.com) are affiliated with Intel Corporation.

Packet classification is often the first packet processing step in routers. It requires network systems to maintain and to navigate through search data structures. Since flows can be identified only after the classification step, to prevent performance interference across flows, network systems must ensure that classification operates at line speeds. Unfortunately, the overhead of navigating through search data structures can often exceed the time budget enforced by the line-speed processing requirement. Thus, a key challenge is to design packet classification algorithms that impose low memory space and access overhead and hence can scale to high bandwidth networks and large databases of classification rules.

In this paper, we take a step in the direction of designing such efficient classification algorithms. In particular, we study the properties of packet classification rules; our intent is to expose characteristics that can be exploited to design packet classifiers that can scale well with link bandwidths and the sizes of classification rule databases. Since access control is the most common application of packet classification today, we study four databases of classification rules collected from firewalls supported by large ISPs and corporate intranets. Our analysis yields the following key observations:

1. The fields contained in each rule in firewall databases can be partitioned into two logical entities: (1) source and destination IP address pairs that characterize distinct network paths, and (2) a set of transport-level fields (e.g., port numbers, protocol identifier, etc.) that characterize network applications. In most cases, the number of distinct network paths far exceeds the number of network applications.
2. The IP address pairs define regions in the two-dimensional space that can overlap with each other. However, the number of overlaps is significantly smaller than the theoretical upper-bound.
3. Many source-destination IP address pairs share the same set of transport-level fields. Hence, only a small number of transport-level fields are sufficient to characterize databases of different sizes.

We justify these observations based on standard network administration practices; and thereby argue that these findings, although derived from a small number of databases, are likely to hold for most firewall databases. Based on these findings, we provide the following guidelines for designing efficient

classification algorithms.

1. The multi-dimensional classification problem should be split into two sub-problems (or two stages): (1) finding a 2-dimensional match based on source and destination IP addresses contained in the packet; and (2) finding a (n-2) dimensional match based on transport-level fields. Whereas the first stage only involves prefix matching, the second stage involves the more general range matching.
2. Because of the overlap between IP address filters maintained in a database, each packet may match multiple filters. Identifying all the matching filters is complex. Since the total number of overlaps observed in firewall databases is significantly smaller than the theoretical upper-bound, a design that maintains all of the intersection filters and returns exactly a single filter is both feasible an desirable.
3. Since each IP address filter is associated with multiple transport-level fields, identifying the highest priority rule that matches a packet requires searching through all the transport-level fields associated with the matching IP filter. Since the number of transport-level fields associated with most databases is rather small, it is possible to rely upon a small, special-purpose hardware unit (e.g., a TCAM unit) to perform the (n-2) dimensional searches in parallel.

The paper is structured as follows. In Section 2, we formulate the classification problem and discuss our methodology for studying ACLs. We discuss our findings in Sections 3 and 4, and expose the implications of our findings in Section 5. Finally, Section 6 summarizes our contributions.

TABLE I
EXAMPLES OF CLASSIFICATION RULES

src. IP address	*dest. IP address*	*src. Port*	*dest port*	*action*	*priority*
128.59.67.100	128.*	*	15	drop	2
128.*	128.2.3.1	*	24	DSCP 2	1

II. PROBLEM FORMULATION

Since access control is the most common application of packet classification today, we focus on the problem of packet classification in firewalls. In a firewall rule database, each rule contains one or more fields and their associated values, a priority, and an action. The fields generally correspond to specific portions of the TCP/IP header—such as the source and destination IP addresses, port numbers, and protocol identifier. Because of the hierarchical nature of IP address allocation, source and destination IP addresses are often specified as prefixes. To accommodate a collection of user or network management applications, port numbers are often specified as ranges. Finally, other protocol attributes, such as the protocol identifier, are specified as exact values. Table I shows some examples of classification rules.

The first rule indicates that packets originating from the IP address 128.59.67.100, and destined to any host within the IP address domain beginning with 128 and port number 15 should be dropped. The priority level for this rule is 2. The second rule states that packets originating from any host in the domain beginning with 128, and destined to the host 128.2.3.1 and port number 24 should be forwarded with the Differentiated Services Code Point (DSCP) set to 2. This rule has priority level of 1.

In this context, the packet classification problem can be stated as follows: Given a set—often referred to as an Access Control List (ACL)—of access control rules, determine the action *A* associated with the highest priority rule that matches packet *p*. To reduce the overhead of identifying rules that may match each packet, most packet classification algorithms employ search data structures for organizing classification rules. These data structures occupy memory space. Furthermore, navigating on these data structures incurs several memory accesses. In what follows, we first discuss several existing packet classification algorithms and argue that they do not scale well with increase in network bandwidth or ACL sizes. We then argue that understanding the structure and properties of ACLs is crucial in designing efficient, scalable algorithms. Finally, we describe our methodology for studying the properties of ACLs.

A. State-of-the-art

Existing packet classification algorithms can be grouped into four classes: trie-based algorithms, hash-based algorithms, parallel search algorithms, and heuristic algorithms. Throughout this discussion, we use *n* to denote the number of rules in a classification database, *k* to denote the number of fields (i.e., dimensions), and w to denote the maximum length of the fields (in bits).

1. **Trie-based Algorithms**: Trie-based algorithms [2, 3] build hierarchical radix tree structures where once a match is found in one dimension a search is performed in a separate tree linked into the node representing the match. Examples of such algorithms are the Grid-of-tries [3] and Area-based Quad Tree (AQT) [5] algorithms. Trie-based algorithms require, in worst case, as many memory accesses as the number of bits in the fields used for classification. Multi-bit trie data structures are more efficient from the perspective of the number of memory accesses required. However, these data structures incur significantly higher memory space overhead. In general, trie-based schemes work well for single-dimensional searches. However, the memory requirement of these schemes increases significantly with increase in the number of search dimensions.
2. **Hash-based Algorithms**: Hash-based algorithms [9] group rules according to the lengths of the prefixes specified in different fields. The groups formed in this manner are called ‘tuples’. Hash-based algorithms perform a series of hash lookups one for each tuple to identify the highest priority matching rule. Tuple space search has $O(n)$ storage and time complexity. Hash-based

algorithms, in the worst case, require as many memory accesses as the number of hash tables, and the number of hash tables can be as large as the number of rules in a database. As a result, hash-based techniques do not scale well with the number of rules. An optimized hashing technique, referred to as rectangle search [9], reduces the lookup time complexity from $O(n)$ to $O(w)$ in two dimensions. However, to support lookups in more than two dimensions, the algorithm still requires a significant number of memory accesses[1].

3. **Parallel Search Algorithms**: These algorithms formulate the classification problem as an n-dimensional matching problem and search each dimension separately. In some algorithms [4], when a match is found in a dimension, a bit vector is returned identifying the matches. The logical AND of the bit vectors returned from all dimensions identifies the matching rules. Such bit-vector techniques are associated with $O(n)$ memory accesses in the lookup process. Fetching a single bit vector or an aggregate bit vector (as described in [13]) can be memory access intensive, especially in cases where the ACL contains more than a few thousand rules. Another parallel search technique called Cross-Producting Table [3] reduces the lookup time complexity to $(O(kw))$ where k is the number of fields and w is maximum length of the fields. However, this technique increases the worst case storage complexity to $(O(n^k))$ making it impractical.
4. **Heuristic Algorithms**: A fourth category of algorithms includes heuristic algorithms that exploit the structure and redundancy in the rule set [7, 8]. The algorithms proposed to-date are associated with very low lookup time complexity $(O(k))$; however, they impose significant memory space requirements $(O(n^k))$. Hence, these algorithms are suitable for single- or two-dimensional searches, but their space requirement makes them unsuited for the more common five-dimensional searches.

From the above discussion, it is apparent that exploiting the structure and properties of ACLs is a promising direction for designing packet classification algorithms that can scale well with link bandwidth and ACL sizes. Unfortunately, the literature contains no detailed studies of ACL properties. This is in-part because ISPs and enterprises, for privacy and security reasons, protect access to their rule databases. Recently, we have obtained access to four firewall databases from ISPs and corporate intranets. Hence, in this paper, we conduct a careful study to expose the structure and properties of these ACLs, and postulate how these properties can be used to design efficient classification algorithms. The design of specific packet classification algorithms, however, is beyond the scope of this paper.

[1] A lower bound on the complexity of rectangle search is discussed in [9]. It is proven than tuple probes can be at least $w(k\text{-}1)/k$!

B. Experimental Methodology

We analyze four firewall databases; three of these databases are from large ISPs, whereas one is from a corporate intranet. Table II summarizes the basic statistics of these ACLs.

As Table II indicates, the ISP ACLs are generally much larger than those of the enterprise intranets. Further, it shows that the fields specified in ACLs can be partitioned into two logical entities: (1) source and destination IP address pairs that characterize distinct network paths represented in ACLs, and (2) a set of transport level fields (e.g., port numbers, protocol identifier, etc.) that characterize network applications. In most cases, the number of distinct network paths far exceeds the number of network applications represented in the ACLs.

In what follows, we first analyze IP address pairs and then study the characteristics of transport-level fields. We justify our findings based on standard practices for creating ACLs used by network administrators. Hence, we argue that although our observations are derived from a small number of rule databases, our conclusions are likely to be valid across a large number of such rule databases.

TABLE II
SUMMARY OF ACLS

	type	*number of rules*	*number of unique source/destination IP address fields*	*protocol types*	*unique port number fields*
ACL1	ISP	754	426	4	140
ACL2	ISP	607	527	5	30
ACL3	ISP	2399	1588	5	192
ACL4	Intranet	157	98	4	36

III. IP PREFIX PAIR ANALYSIS

Each rule in an ACL contains a specification of source and destination IP address pairs (also referred to as IP address filters). These addresses are specified as wildcards, prefixes, or exact values. Based on these specifications, the filters represent rectangles, lines or points in the two-dimensional IP address space. Further, the filters may overlap with each other. In what follows, we first conduct a structural analysis of the filters; this allows us to characterize ACLs as a composition of different types of filters (i.e., filters that represent a different shape in the two-dimensional space). We find that only a small number of filters contain wildcards in the source or the destination dimensions in the ISP ACLs. Further, for most filters that do not contain any wildcards, the destination field contains complete IP addresses (representing individual hosts), while the source field contains prefixes (representing IP address domains). Second, we analyze the overlaps among the filters. This allows us to characterize the number of filters that may match a packet, as well as the overhead of maintaining in the ACL a unique filter representing each of the overlaps such that the maximally matching filter can be uniquely identified for each packet. We find that overlaps are created mostly by filters that contain a wildcard in their source or destination fields. Since only a small number of filters contain wildcards, the actual number of overlaps observed in ACLs is

significantly smaller than the theoretical upper bound.

A. Structural Analysis

The source-destination IP address pairs can be classified into two types: Partially-specified filters and fully-specified filters. Partially-specified filters contain at least one wildcard (*) in the source or in the destination IP address dimension; these filters capture traffic sent to/from designated servers or subnets of ISP networks. Fully-specified filters, on the other hand, contain an IP address prefix in both the source and destination IP address dimensions. These filters identify the traffic exchanged between specific IP address domains of ISP networks. In most cases, the traffic handled by fully-specified filters is exchanged between important servers (e.g., web, e-mail, NTP, or streaming servers) and clients.

Each IP address filter can be represented geometrically as a point, a line, or a rectangle in a two dimensional IP address space. Whereas partially-specified filters of the form (*,*) cover the entire two dimensional address space, filters of the form (x, *) and (*, y) can be represented either as a line or a rectangle in the 2-D space depending on the values of x and y. If x and y represent IP address domains (i.e., IP prefixes of length smaller than 32), then these filters are represented as rectangles; on the other hand, if x and y denote hosts (i.e., full 32-bit IP addresses), then the corresponding filters are represented as lines. Similarly, depending the lengths of x and y, fully-specified IP address filters of the form (x, y) represent lines, points, or rectangles in the two dimensional space.

TABLE III
PARTIALLY- AND FULLY- SPECIFIED FILTERS

	partially-specified filters	*fully-specified filters*	*total number of filters*
ACL1	4 (1%)	421 (99%)	426
ACL2	68 (13%)	458 (87%)	527
ACL3	160 (10%)	1427 (90%)	1588
ACL4	83 (86%)	14 (14%)	98

TABLE IV
BREAKDOWN OF PARTIALLY-SPECIFIED FILTERS

	wildcard in source address	*wildcard in destination address*
ACL1	2	2
ACL2	36	32
ACL3	112	48
ACL4	12	71

Table III shows the breakdown of partially- and fully-specified filters in our firewall ACLs. It illustrates that, whereas partially-specified filters represent a small percentage of the total number of filters in large ISP databases; they constitute a significant percentage of the relatively small-size enterprise intranet firewall ACL. This is because large ISPs often describe administrative policies between specific IP address domains within their network. Examples of such policies include the admission of all HTTP traffic between a server and a client subnet, or the blocking of all RTSP traffic between two specific IP address domains. In intranets, on the other hand, administrators do not specify cross-domain traffic management policies, since such policies are often enforced by their ISP. Instead, most of the rules in intranet firewalls refer to specific sources or destinations, but not both.

We further analyze the partially-specified filters to determine the relative occurrence of the wildcard in the source or the destination IP address fields, as well as the lengths of specified IP addresses. We find that in the intranet ACL, which is the smallest in size, filters with the wildcard in the destination address are the majority. In the first two ACLs, which are of medium size, there is a balance between the filters that have the wildcard in the source and destination address fields. In the third ACL, which has the largest size, most filters have the wildcard in the source address field.

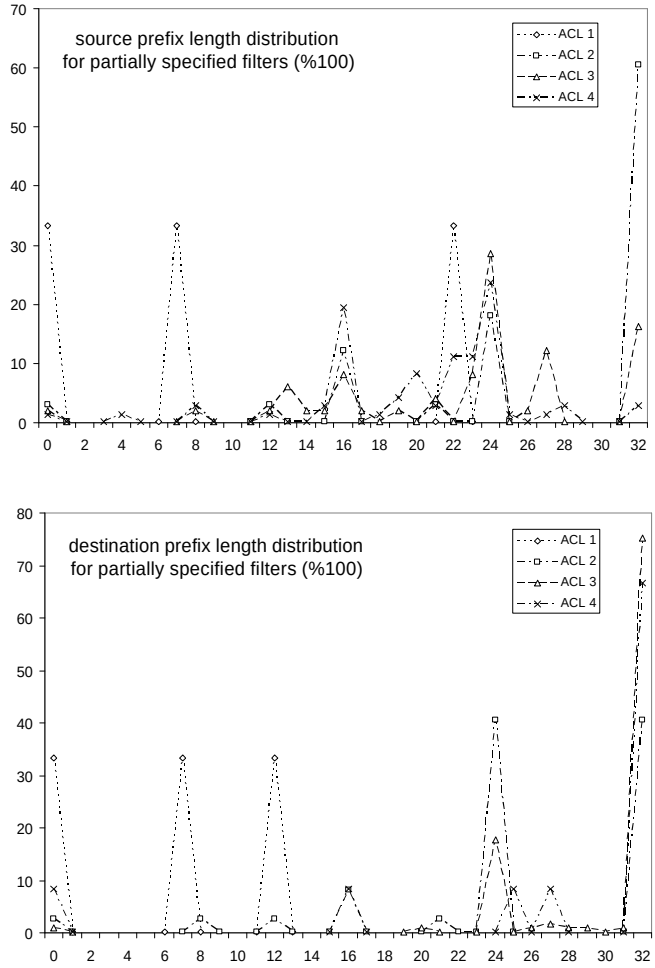

Fig. 1. Distribution of prefix lengths for partially-specified filters

From the results of table IV it appears as if there is a dependency between the size of an ACL and the numbers of filters that have the wildcard in the source or destination IP address fields. Typically, the smaller an ACL is the closer to client networks the firewall is located. The intranet ACL in our example describes policies that block the traffic form many specific client subnets of the intranet and thus contains many rules having the wildcard in the destination dimension. Larger ACLs on the other hand are closer to the Internet core and describe higher-level policies for connecting to important

servers or networks. Such policies are typically expressed as rules having the wildcard in the source dimension.

Figure 1 shows the distribution of prefix lengths for partially-specified filters. It shows that the source and destination IP address specifications are spread across the entire range of prefix lengths, with 8-bit, 16-bit, 24-bit and 32-bit prefixes constituting the majority. Geometrically, this indicates that most partially-specified filters represent lines or rectangles characterized by a few standard width values in the two-dimensional space.

TABLE V
BREAKDOWN OF FULLY-SPECIFIED FILTERS

	domain-domain filters	*host-domain filters*	*domain-host filters*	*host-host filters*
ACL1	30	31	37	323
ACL2	124	99	154	81
ACL3	165	18	755	489
ACL4	9	0	2	3

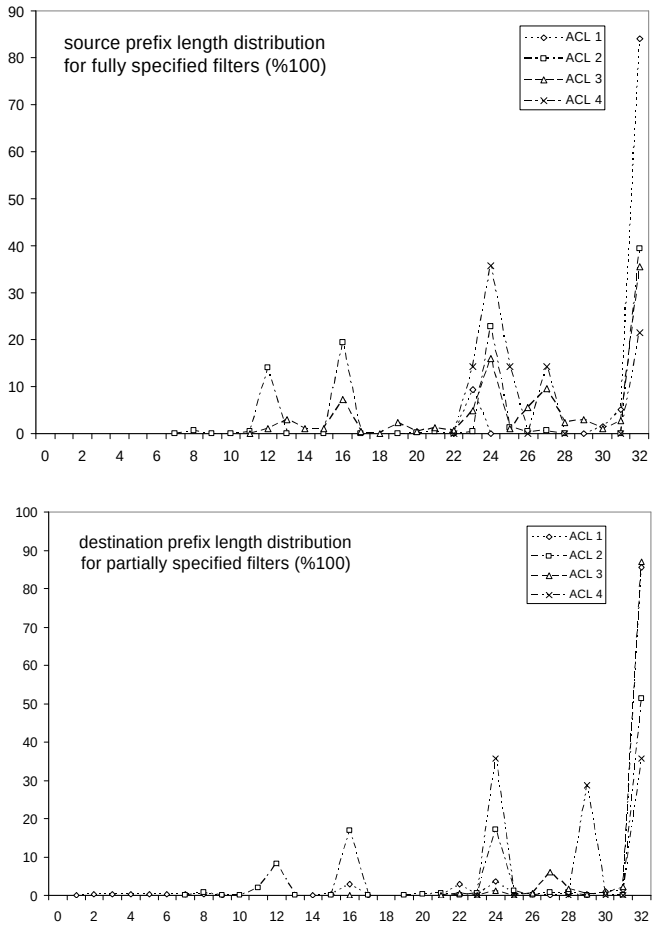

Fig. 2. Distribution of source and destination prefix lengths for fully-specified filters

There are four types of fully-specified filters: (1) filters that characterize traffic exchanged between two domains, (2) filters that characterize traffic originating within a domain but destined to a host, (3) filters that characterize traffic originating a host destined for an IP domain, and (4) filters that characterize traffic exchanged between a specific pair of hosts. In these filters, a host is represented using a 32 bit address (IPv4 address) while a domain is represented by a shorter prefix. Table V shows the breakdown of these four types of filters in our ACLs. It shows that majority of the fully-specified filters represent communication where either the sender or the receiver is a host. In many cases these hosts are servers representing important resources of large networks. On the other hand, in the intranet ACL the majority of fully-specified filters represent Domain-Domain filters.

TABLE VI
TRIE BLOCK ANALYSIS

	number of unique source prefixes	*observed number of source trie blocks*	*theoretical bound on the number of source trie blocks*
ACL1	97	29	759
ACL2	182	231	1439
ACL3	431	496	3256
ACL4	79	127	615
	number of unique destination prefixes	*observed number of destination trie blocks*	*theoretical bound on the number of destination trie blocks*
ACL1	205	383	1623
ACL2	207	243	1639
ACL3	516	620	3855
ACL4	20	60	155

Figure 2 shows the distribution of source and destination prefix lengths for fully-specified filters. Geometrically, this indicates that most fully-specified filters represent lines or points in the two-dimensional space. The spatial distribution of IP prefixes is a very important property, especially to analyze requirements to store the IP prefixes. We have created a 4-bit trie data structure for both source and destination IP addresses, measured the number of trie blocks required to store IP prefixes, and compared this number with theoretical maximum for number of trie blocks. The results are shown in Table VI. We find that the total number of trie blocks needed to represent source and destination prefixes is much less than the theoretical upper bound in real world data bases.

From the above analyses, we derive the following general conclusions:

1. Filters in real world ACLs are either fully-specified or partially-specified. Partially-specified filters represent a small percentage of the total number of filters in medium and large size ACLs.
2. The breakdown of partially-specified filters between filters having the wildcard in source and destination IP addresses may depend on the size of the ACL. Careful study of more ACLs would help investigating the existence of such dependency.
3. Most fully specified filters are segments of straight lines

or points in medium and large size ACLs.

4. Trie data structures representing source and destination prefixes require much fewer blocks than the theoretical upper bound.

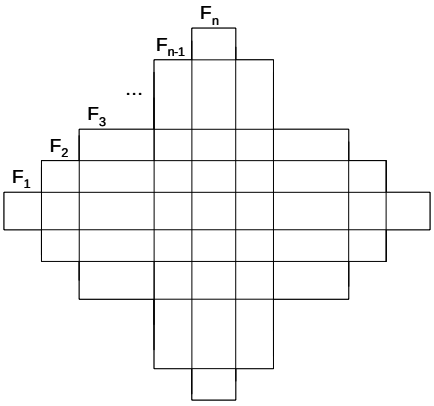

Fig. 3. Worst-case filter structure

B. Overlap Analysis

The geometrical objects representing filters may overlay in the two dimensional space. Since each packet represents a point in the two dimensional space, it may be contained within the geometrical space defined by one or more filters in the ACL. In such an event, a packet may match multiple filters within the ACL; hence, identifying the highest priority rule requires comparing transport-level fields associated with all the matching filters with the appropriate fields contained in the packet. Clearly, the larger the number of filters that a packet may match with, the greater is the complexity of identifying the highest priority rule that matches the packet. In the worst-case, if all filters within an ACL overlap with each other (as shown in Figure 3), then identifying the highest priority rule for a packet that represents a point in the intersection of these filters may require a search on all filters. Thus, the complexity of packet classification depends on the amount of overlap between filters (which in turn determines the number of filters that may match a packet).

TABLE VII
NUMBER OF FILTERS THAT MAY MATCH A PACKET

	average	*standard deviation*	*maximum*
ACL1	4.00	0.36	5.00
ACL2	3.96	0.73	7.00
ACL3	3.75	0.84	7.00
ACL4	3.71	0.90	7.00

In what follows, we analyze our ACLs for their overlap properties. Table VII and Figure 4 show the distribution of the number of filters that may match a packet. Figure 4 illustrates that for all the ACLs, on an average, about 4 filters match every packet. Although this is not a very large number, identifying these filters imposes significant overhead. The navigation on the data structures that store two-dimensional filters (e.g., hierarchical singe-bit or multi-bit tries) typically requires significantly more memory accesses than the number of filters matching a packet. For instance, the Extended Grid of Tries (EGT) algorithm reported in [12] requires 137 accesses to classify packets from a database of 2799 rules.

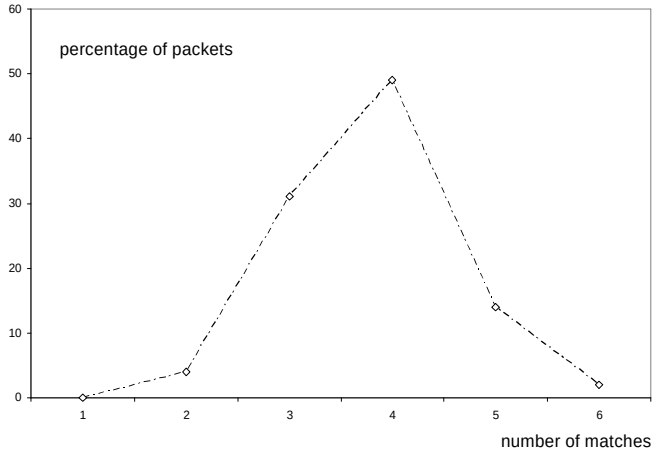

Fig. 4. Distribution of the number of filters that may match a packet

TABLE VIII
OBSERVED FILTER OVERLAPS

	number of rules	*number of filters*	*observed number of partial overlaps*	*upper bound on the number of partial overlaps*
ACL1	754	426	4	90,525
ACL2	607	527	2,249	138,601
ACL3	2,399	1,588	6,138	1,260,078
ACL4	157	98	852	4,753

An alternative architecture involves maintaining a filter that represents each overlap in the ACL. We observe that overlaps between filters can be complete or partial. In the event that one filter is completely contained in another, the overlap between the filters is represented exactly by the contained filter. In such a case, no additional filter needs to be stored. On the other hand, if filters overlap partially, then the overlap can be identified uniquely by a filter that represents the intersection region between the two filters; hence, each partial overlap introduces a new filter in the ACL. If all such filters are maintained, then the classifier can determine the most refined filter for each packet. In the worst case, if each filter overlaps with all the other filters in the ACL, then maintaining all the intersection filters would incur an $O(n^2)$ overhead[2], where n is the number of distinct IP prefix pairs in the ACL. However, as we have illustrated earlier, most ACLs contain filters that can be represented as points, lines or small rectangles. Hence, we can expect the number of additional filters required for real ACLs to be much smaller than the theoretical worst-case.

To validate this hypothesis, we determine the number of such overlap observed in our ACLs. The results are shown in

[2] The worst-case scenario is one where each filter in the ACL overlaps with all the other filters. In such a case, the number of filters that represents the overlaps can be bounded by $(n\text{-}1) + (n\text{-}2) + \ldots + 1 = n(n\text{-}1)/2$.

Table VIII. Table VIII indicates that the number of filters representing intersections that may need to be stored is, in fact, several orders of magnitude smaller than the theoretical upper bound.

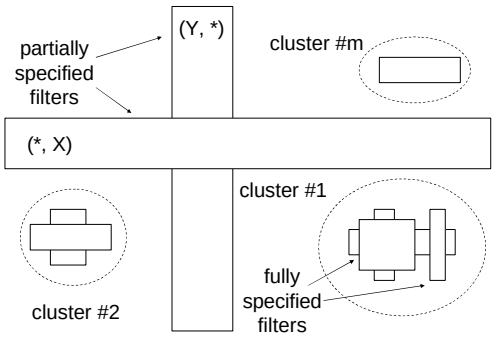

Fig. 5. A realistic structure of filters in ACLs.

Our analyses of the ACLs show that the organization of filters in real-world ACLs is significantly different from the worst-case structure shown in Figure 3. A more realistic structure of filters is shown in Figure 5. The filters in the structure of Figure 5 are either-fully specified or partially-specified as explained in the previous section. Some fully-specified filters form 'clusters' as shown in Figure 5. A cluster is a set of filters where every filter overlaps either partially or completely with at least one other filter in the cluster. A closer analysis of the ACLs reveals that there are three cases that create partial overlap between filters.

1. Overlaps between partially-specified filters. Each filter having the wildcard in the source IP address dimension creates a unique partial overlap with all the filters having a wildcard in the destination IP address dimension. Since IP addresses are specified as prefixes, filters with a wildcard in the same dimension do not create partial overlaps between each other; such filters are either disjoint or completely overlapping. The number of partial overlaps created only by partially-specified filters is equal to the product of the number of partially-specified filters in each of the two dimensions.
2. Overlaps between fully-specified filters. Fully-specified filters may overlap with each other either fully or partially.
3. Overlaps between fully- and partially-specified filters.

Table IX shows the breakdown of the number of partial overlaps created in each of the four ACLs. It shows that the overlaps created by partially-specified filters represent the majority in all ACLs, ranging from 51% in ACL 2 up to 100% in ACL 1 and ACL 4. We also observe that the overlaps created between partially and fully-specified filters represent a significant percentage (45%) of the total number of overlaps in ACL 2. In all the ACLs, fully-specified filters create an insignificant number of overlaps (it turns out that most clusters have size equal to one). These results indicate that partially-specified filters are the main source of overlaps in all ACLs. Further, as we had demonstrated earlier, partially-specified filters generally represent only a small percentage of the total number of filters in large databases. These two observations together justify why the total number of partial overlaps is significantly less than the theoretical upper bound. In Appendix A, we derive a tighter upper bound on the number of partial overlaps.

TABLE IX
BREAKDOWN OF OVERLAPS

	number of overlaps	*overlaps formed by partially specified filters only*	*overlaps formed by fully specified filters only*	*overlaps formed by between fully ad partially specified filters*
ACL 1	4	100%	0%	0%
ACL 2	2249	45%	4%	51%
ACL 3	6138	88%	1%	11%
ACL 4	852	100%	0%	0%

IV. TRANSPORT LEVEL FIELD ANALYSIS

The Internet supports thousands of routes but relatively only a few, commonly used applications. Hence, as indicated in Table II, only a small number of unique transport-level fields (consisting of port numbers and protocol types) are usually present in ACLs. Further, many source-destination pairs share the same transport-level fields. In what follows, we first analyze the transport-level fields associated with individual source-destination pairs (or IP address filters) and then expose the sharing of these transport-level fields across multiple IP filters.

A. Analysis of Transport-level Fields for Individual IP Filters

ACLs generally contain several rules with the same IP address filter (i.e., source-destination IP address pair) but with different combination of transport-level fields. To understand this phenomenon carefully, we analyzed the sets of such transport-level fields associated with the same IP filters.

Figure 7 depicts the distribution of the set sizes observed in the four ACLs under consideration. It shows that for all the ACLs, most (about 90%) transport-level field sets are small (1-4 entries); the remaining 10% of the sets have sizes between 5 and 40. This is mainly because most ACLs contain rules that identify explicitly only a small number of the most popular applications; in today's Internet the number of these applications is very small.

We observe that the highest percentage of transport-level fields in our ACLs specify TCP and UDP protocols. This is because most data traffic in today's Internet uses TCP and a smaller percentage of traffic uses UDP. Further, most transport-level fields specify a destination port or port range. The source port field is generally unspecified (i.e., a wildcard specification). This is because most classification rules apply to packets that request the establishment of TCP connections. These packets are sent to servers that are listening to well-known non-ephemeral or ephemeral ports. Table X depicts the

distributions of the source and destination port numbers observed for the four ACLs.

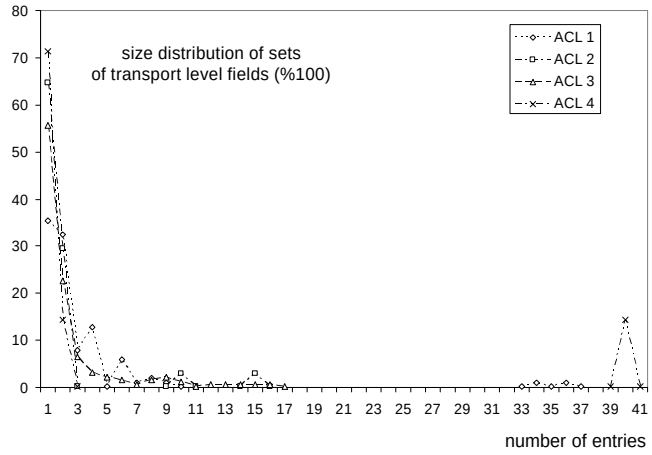

Fig. 6. Distribution of sizes of transport level field sets

TABLE X
DISTRIBUTION OF SOURCE AND DESTINATION PORT NUMBERS IN TRANSPORT-LEVEL FIELDS

	number of unique transport-level fields	*source port number*			*destination port number*		
		wildcard	*range*	*exact value*	*wildcard*	*range*	*exact value*
ACL1	146	146	0	0	4	74	68
ACL2	40	40	0	0	8	29	3
ACL3	202	200	2	0	5	157	40
ACL4	43	42	1	0	8	32	3

B. Sharing Transport-level Fields Across Multiple IP Filters

To analyze the sharing of transport-level fields across multiple IP filters, we derive the total number of transport-level field entries with and without any sharing across filters. Table XI summarizes our findings. It shows that for all ACLs, many source-destination IP prefix pairs share the same sets of transport-level fields. The relative priority and corresponding actions of fields are the same in different occurrences of each set. In addition the number of unique entries characterizing the shared sets of transport level fields is also small. This number is much smaller than the total number of entries in the unique sets.

TABLE XI
SHARING TRANSPORT-LEVEL FIELDS AMONG IP FILTERS

	number of transport-level fields	*number of transport-level fields in unique sets*	*number of unique transport-level fields*
ACL1	754	316	146
ACL2	607	67	40
ACL3	2399	442	202
ACL4	157	48	43

V. IMPLICATIONS

Our evaluation of ACLs leads us to the following main conclusions.

1. The fields contained in each rule in ACLs can be partitioned into two logical entities: (1) source and destination IP address pairs that characterize distinct network paths represented in ACLs, and (2) a set of transport level fields (e.g., port numbers, protocol identifier, etc.) that characterize network applications. In most cases, the number of distinct network paths far exceeds the number of network applications.
2. The IP address filters are either partially-specified or fully-specified. Partially-specified filters represent a small percentage of the total number of filters in databases. Furthermore, most of the overlap between filters is caused by partially-specified filters. Fully-specified filters create only a few partial overlaps with each other. Thus, the total number of overlaps is significantly smaller than the theoretical bound.
3. Many source-destination IP address pairs share the same set of transport-level fields. Hence, only a small number of transport-level fields are sufficient to characterize databases of different sizes.

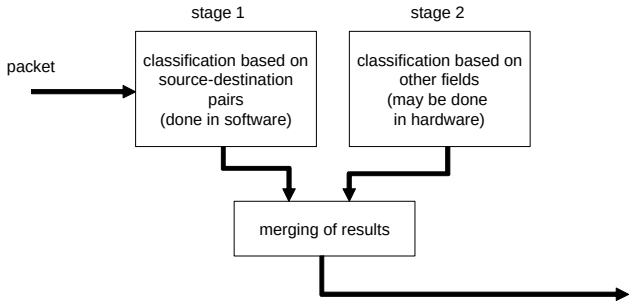

Fig. 7. Two stage classification architecture

Based on these findings, we provide the following guidelines for designing efficient classification algorithms.

1. The multi-dimensional classification problem should be split into two sub-problems (or two stages): (1) finding a 2-dimensional match based on source and destination IP addresses contained in the packet, and (2) finding a (n-2) dimension match based on transport-level fields (see Figure 8). Whereas the first stage only involves prefix matching, the second stage involves the more general range matching.
2. Because of the overlap between IP address filters maintained in an ACL, each packet may match multiple filters in stage 1. Identifying all the matching filters is complex. Since the total number of overlaps observed for ACLs is significantly smaller than the theoretical upper-bound, a design that maintains all of the intersection filters and returns exactly a single match from stage 1 is both feasible an desirable.

3. Since each IP address filter is associated with multiple transport-level fields, identifying the highest priority rule that matches a packet requires searching through all the transport-level fields associated with the matching IP filter. Since the number of transport-level fields associated with most ACLs is rather small, it is possible to rely upon a small, special-purpose hardware unit (e.g., a TCAM unit) to perform the (n-2) dimensional search in parallel.

The combination of a fast software algorithm for finding a 2-dimensional match in stage 1 and a specialized hardware acceleration unit for performing (n-2) dimensional match in stage 2 can result in a classification system capable of meeting the stringent space time constraints of network processors.

VI. Conclusion

To classify a packet as belonging to a flow often requires network systems—such as routers and firewalls—to maintain large data structures and perform several memory accesses. Network processors, on the other hand, are generally configured with only a small amount of memory with limited access bandwidth. Hence, a key challenge is to design packet classification algorithms that can be implemented efficiently on network processor platforms. We argue that the design of such algorithms will need to exploit the structure and characteristics of packet classification rules.

In this paper, we analyze several databases of classification rules found in firewalls and derive their statistical properties. Our analysis yields three main conclusions: (1) the rules found in ACLs contain two types of fields—source-destination IP address pairs that identify network paths and transport-level fields that characterize network applications; further, these rules refer to many more network paths than applications. (2) IP address pairs identify regions that overlap with each other; however, the number of overlaps is significantly smaller than the theoretical upper-bound. (3) Only a small number of transport-level fields are sufficient to characterize ACLs of different sizes. We justify our findings based on several standard practices employed by network administrators, and thereby argue that although our findings are for specific databases, the properties are likely to hold for most databases. Based on these findings, we suggest that a hybrid, two-stage classification architecture that combines a software scheme for matching in 2-dimensions (IP address pairs) with a hardware unit that performs efficient (n-2) dimensional searches has the potential of scaling well with link speeds and ACL sizes.

. References

[1] P. Gupta and N. McKeown “Algorithms for Packet Classification”, IEEE Network Magazine, 2001

[2] P. Tsuchiya. “A search algorithm for table entries with non-contiguous wildcarding,” unpublished report, Bellcore.

[3] V. Srinivasan, S. Suri, G. Varghese, and M. Waldvogel. “Fast and Scalable Layer four Switching,” Proceedings of ACM Sigcomm, pages 203-14, September 1998.

[4] T.V. Lakshman and D. Stiliadis. “High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching”, Proceedings of ACM Sigcomm, pages 191-202, September 1998.

[5] M.M. Buddhikot, S. Suri, and M. Waldvogel. “Space decomposition techniques for fast layer-4 switching,” Proceedings of Conference on Protocols for High Speed Networks, pages 25-41, August 1999.

[6] A. Feldman and S. Muthukrishnan. “Tradeoffs for packet classification,” Proceedings of Infocom, vol. 3, pages 1193-202, March 2000.

[7] P. Gupta and N. McKeown, Packet Classification on Multiple Fields, Proc. Sigcomm, Computer Communication Review, vol. 29, no. 4, pp 147-60, September 1999, Harvard University.

[8] P. Gupta and N. McKeown, Packet Classification using Hierarchical Intelligent Cuttings , Proc. Hot Interconnects VII, August 99, Stanford. This paper is also available in IEEE Micro, pp 34-41, vol. 20, no. 1, January/February 2000.

[9] V. Srinivasan, S. Suri, and G. Varghese. “Packet Classification using Tuple Space Search”, Proceedings of ACM Sigcomm, pages 135-46, September 1999.

[10] F. Shafai, K.J. Schultz, G.F. R. Gibson, A.G. Bluschke and D.E. Somppi. “Fully parallel 30-Mhz, 2.5 Mb CAM,” IEEE Journal of Solid-State Circuits, vol. 33, no. 11, November 1998.

[11] A. Prakash, and A. Aziz, “OC-3072 Packet Classification Using BDDs and Pipelined SRAMs”, Hot Interconnects, 2001

[12] F. Baboescu, S. Singh, and G. Varghese, “Packet Classification for Core Routers: Is there an alternative to CAMs?”, Technical Report, University of California, San Diego, 2003.

[13] F. Baboescu, G. Varghese, “Scalable Packet Classification”, Proceedings of ACM Sigcomm, pages 199-210, August, 2001.

[14] M. Degermark, A. Brodnik, S. Carlsson, and St. Pink, "Small forwarding tables for fast routing lookups," in Proc. ACM SIGCOMM, September 1997, pp. 3—14

Appendix A: A Tighter Bound on the Number of Partial Overlaps

From the analyses of ACLs, we have shown that the number of overlaps between IP filters is significantly smaller than the theoretical upper-bound of $n \cdot (n-1)/2$. In this appendix, we derive a tighter upper bound on the number of partial filter overlaps. The derivation of the upper bound is based on properties that characterize medium size and large ISP ACLs. Therefore the analysis presented in this appendix applies to the first three of our ACLs only.

There are three factors that produce intersections between IP filters in ACLs. First, partially-specified filters create intersections with each other. The number of such overlaps O_1 is exactly equal to $S \cdot D$ where S is the number of partially-specified filters that specify the source IP address dimension and D is the number of partially-specified filters that specify the destination IP address dimension. Since partially-specified

filters represent a small percentage of the total number of filters in all databases (1%- 13%) we expect their overlaps to be bounded by the square of the number of filters divided by a large constant. In fact, the majority of partial overlaps (51%-100%) are created by partially-specified filters in databases.

Second, partial overlaps result from the intersections between fully-specified filters in the same cluster cluster. However, clusters with more than one element are only but a few in our ACLs. Fully specified filters form an insignificant amount of overlaps between each other. This happens in part because server and client subnets are characterized by disjoint IP address domains in rules. As a result the number of partial overlaps O_2, created by fully-specified filters, is also much less than the theoretical upper bound.

Third, partial overlaps are created between fully and partially-specified filters. Each fully-specified filter may create partial overlaps with one or more partially-specified filters. In most medium size and large ACLs the total number of servers representing the prefixes of partially specified filters is bounded per IP address domain. As a result the total number of overlaps formed between fully-and partially-specified filters O_3 is bounded by the product of the number of filters times a constant. The detailed derivation of an upper bound is given below:

$$O = O_1 + O_2 + O_3 \qquad \text{Eq. 2}$$

$$O_1 = S \cdot D \qquad \text{Eq. 3}$$

$$O_2 \le \sum_{i=1}^{q} \frac{C_i \cdot (C_i - 1)}{2} \qquad \text{Eq. 4}$$

$$O_3 = \sum_{j=1}^{f} F_j \qquad \text{Eq. 5}$$

Equation 5-8 result in:

$$O \le S \cdot D + \sum_{i}^{q} \frac{C_i \cdot (C_i - 1)}{2} + \sum_{j=1}^{f} F_j \qquad \text{Eq. 6}$$

S and D are the number of partially-specified filters that specify the source and destination IP address dimensions respectively, q is the number of clusters that contain more than one filter, C_i is the number of filters in cluster i, f is the number of fully-specified filters that create partial overlaps with partially-specified filters, and F_j is the number of partially-specified filters that overlap with filter j. To complete the derivation of an upper bound we need to understand the relation between the parameters S, D, q, f, C_i, F_j and the number of filters in a database, n.

Let r_1, be the ratio of the total number of filters in a database over the number of partially-specified filters. Then $S+D = n/r_1$. The ration r_1 is expected to be greater than one with very high probability in many different databases. The number of overlaps formed by partially-specified filters O_1, is equal to the product $S \times D$. This product is maximized when $S = D = n/(2 \cdot r_1)$. Therefore:

$$O_1 \le \frac{n^2}{4 \cdot r_1^2} \qquad \text{Eq. 7}$$

Let r_2, be the ratio between the number of fully-specified filters in a database and the number of fully-specified filters that create partial overlaps with each other. Such filters participate in clusters having more than one element. The ratio r_2 is also expected to be greater than one with very high probability. The number of fully-specified filters that create partial overlaps with each other is equal to $(r_1 - 1) \cdot n/(r_1 \cdot r_2)$.

As a result:

$$O_2 \le \frac{(r_1 - 1) \cdot n}{2 \cdot r_1 \cdot r_2} \left(\frac{(r_1 - 1) \cdot n}{r_1 \cdot r_2} - 1 \right) \qquad \text{Eq. 8}$$

The values *of* F_j refer to overlaps between partially and fully-specified filters. The number of such overlaps per fully-specified filter is independent of n as a property of a pair of IP address domains. As a result we can consider that each F_j is bounded by some value F. Therefore:

$$O_3 \le n \cdot F \qquad \text{Eq. 9}$$

Eq. 2, 7-9 result in:

$$O \le \frac{n^2}{4 \cdot r_1^2} + \frac{(r_1 - 1) \cdot n}{2 \cdot r_1 \cdot r_2} \left(\frac{(r_1 - 1) \cdot n}{r_1 \cdot r_2} - 1 \right) + n \cdot F \qquad \text{Eq. 10}$$

Even though the result of Eq. 10 is also $O(n^2)$ this bound is much tighter than the worst case. This happens because the number of filters n is divided by the parameters r_1 and r_2 in Eq. 10. The parameters r_1 and r_2 are expected to be greater than one. Another difference is that the worst case bound is a deterministic bound whereas the bound of Eq. 10 is a stochastic bound, since the parameters r_1, r_2 and F are random variables.

The random variables r_1, r_2 and F have unknown distributions. However, we expect with very high probability that r_1 and r_2 are greater than one and F is a small number. Estimations on the upper bound of Eq. 10 can be derived by selecting the values with the highest frequency from the limited number of databases we experimented with, for r_1, r_2 and F. The values for r_1, r_2 and F used in our calculations are $r_1 = 8.75$, $r_2 = 4.3$ and $F = 15$. More accurate results would require the parameters to be estimated from a greater number of samples. Our results are shown in Table XII. The worst case estimations derived from Eq. 10 are compared against the worst case estimations in this table.

TABLE XII
UPPER BOUNDS ON THE NUMBER OF PARTIAL FILTER OVERLAPS

database number	*number of filters*	*observed number of partial overlaps*	*upper bound from Eq. 10*	*worst case upper bound*
1	426	4	10,790	90,525
2	527	2,249	14,651	138,601
3	1,588	6,138	85,393	1,260,078