Towards a Packet Classification Benchmark

David E. Taylor and Jonathan S. Turner

Technical Report WUCSE-2003-42

Applied Research Laboratory
Department of Computer Science and Engineering
Washington University in Saint Louis
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899
{det3,jst}@arl.wustl.edu

May 19, 2003

Abstract

Packet classification is the enabling technology for next generation network services and often the primary bottleneck in high-performance routers. Due to the importance and complexity of the problem, a myriad of algorithms and resulting implementations exist. The performance and capacity of many algorithms and classification devices, including TCAMs, depend upon properties of the filter set and query patterns. Unlike microprocessors in the field of computer architecture, there are no standard performance evaluation tools or techniques available to evaluate packet classification algorithms and products. Network service providers are reluctant to distribute copies of real filter databases for security and confidentiality reasons, hence realistic test vectors are a scarce commodity. The small subset of the research community who obtain real databases either limit performance evaluation to the small sample space or employ ad hoc methods of modifying those databases. We present a tool for creating synthetic filter databases that retain characteristics of a seed database and provide systematic mechanisms for varying the number and composition of the filters. We propose a benchmarking methodology based on this tool that provides a mechanism for evaluating packet classification performance on a uniform scale. We seek to initiate a broader discussion within the community that will result in a standard packet classification benchmark.

1 Introduction

Deployment of next generation network services hinges on the ability of Internet infrastructure to provide flow identification at physical link speeds. A packet classifier must compare header fields of every incoming packet against a database of filters in order to identify a flow. The resulting flow identifier is used to apply security policies, application processing, and quality-of-service guarantees to packets belonging to the specified flow. Typical packet classification filter databases have fewer than a thousand filters and reside in enterprise firewalls or edge routers. As networks services and packet classifiers continue to migrate into the network core, it is anticipated that classification filter databases could swell to tens of thousands of filters or more.

Packet classification is a difficult multi-dimensional searching problem on the packet 5-tuple. Specifically, the search consists of longest prefix matches on the source and destination IP addresses, range matches

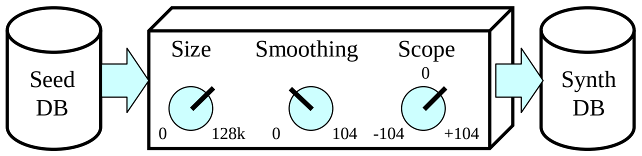

Figure 1: Conceptual view of the synthetic database generator. The size, smoothing, and scope adjustments provide a high-level, systematic mechanism for altering the composition of synthetic databases.

on the source and destination transport port numbers, and an exact match on the protocol field. More complex packet inspection is performed for some applications, but 5-tuple classification is the most prominent case. Commonly cited performance bounds for searching databases with no restriction on the composition of filters are taken from computational geometry which specifies that a point may be located in multi-dimensional space in $O(\log n)$ time and $O(n^k$ space, or $O(\log^{k-1} n)$ time and $O(n)$ space, where n is the number of k-dimensional regions. For packet classification, k is typically five and n depends on how the filter database is organized for searching. It has been observed that "real" packet classification databases exhibit a considerable amount of structure. In response, several algorithmic techniques have been developed which exploit database structure to accelerate search time or reduce storage requirements [1][2][3][4]. Consequently, the performance of these approaches are subject to the structure or statistical characteristics of the database. Ternary Content Addressable Memories (TCAMs) provide a mechanism for searching all filters in parallel, reducing search time to a single memory access. TCAM entries represent filter fields as a prefix; hence, fields specifying arbitrary ranges may require as many as $2(w-1)$ entries, where w is the number of bits required to specify a point in the range. This implies that filters specifying arbitrary ranges for both source and destination ports require up to 900 TCAM entries per filter. Clearly, the capacity of TCAMs are also subject to the statistical characteristics of the filter database.

Despite the influence of filter database composition on the performance of packet classification search techniques and devices, no benchmark suite of databases or formal methodology exists for standardized performance evaluation. Due to security and confidentiality issues, access to large "real" databases for statistical study and performance measurements of new classification techniques has been limited to a small subset of the research community. Some researchers in academia have gained access to databases through confidentiality agreements, but are unable to distribute those databases. Performance evaluations using real databases are restricted by the size and structure of the sample databases. Some researchers have proposed ad hoc methods, such as independently selecting filter fields from a one-dimensional distribution, to modify the composition and number of filters in the databases. In order to facilitate future research and provide a foundation for a meaningful benchmark, we present a technique for generating large synthetic databases which model the statistical structure of a seed database. Along with scaling database size, the tool provides mechanisms for systematically altering the number and composition of filters as depicted in Figure 1. Two adjustments, *smoothing* and *scope*, provide high-level adjustments for database generation and an abstraction from the low-level statistical characteristics. Based on extensive study of real databases, we apply approximations to true joint distributions to preserve the multi-dimensional structure of the seed database. Given a small set of seed databases, our tool for synthetic database generation can be used to generate a set of benchmark databases for a variety of packet classification applications such as firewalls, edge-routers, and core-routers.

Finally, we explore the major issues concerning a methodology for packet classifier benchmarking. It is our hope that this initiates a broader discussion which will lead to the construction of a meaningful benchmark. Its value will depend on its perceived clarity and usefulness to the interested community. In the case of packet classification, this community is comprised of at least the following groups:

- *Researchers* seeking to evaluate new classification algorithms relative to alternative approaches and commercial products.

- *Classification product vendors* seeking to market their products with convincing performance claims over competing products.

- *Classification product customers* seeking to verify and compare classification product performance on a uniform scale. This group can be sub-divided into two major sub-groups: router vendors seeking to compare competing classification products during the design process and prior to selecting components, and router customers seeking to independently verify performance claims of router vendors based on the components used in the router.

We highlight previous performance evaluation efforts by the research community as well as related benchmarking activity of the IETF in Section 2. Results from our study of five firewall databases and discussion of the characteristics relevant to building synthetic models are presented in Section 3. In Section **??** we provide a detailed description of our synthetic database generator and examine the effects of the high-level control parameters. The purpose this tool is to explore the effect of various properties of filter databases on the performance of packet classification algorithms in devices. Section 5 identifies the major issues related to benchmarking packet classifiers and outlines future refinements to the database generator. We seek to initiate a discussion within the community that will guide the remainder of this work and promote the development of valuable performance evaluation tool.

2 Related Work

Extensive work has been done in developing benchmarks for many applications and data processing devices. Benchmarks are used extensively in the field of computer architecture to evaluate microprocessor performance. The effectiveness of these benchmarks to accurately distinguish the effects of architectural improvements, fabrication advances, and compiler optimizations is debatable; yet, there exists inherent value in providing a scale for comparison.

In the field of networking, the IETF (Internet Engineering Task Force) has several working groups exploring network performance measurement. Specifically, the IP Performance Metrics (IPPM) working group was formed with the purpose of developing standard metrics for Internet data delivery [5]. The Benchmarking Methodology Working Group (BMWG) seeks to make measurement recommendations for various internetworking technologies [6][7]. These recommendations concern metrics and performance characteristics as well as collection methodologies.

The BMWG specifically attacked the problem of measuring the performance of Forwarding Information Base (FIB) routers [8][9]. Realizing that router throughput, latency, and frame loss rate depend on the structure of the Forwarding Information Base (FIB) or route table, the BMWG prescribes a testing methodology with accompanying terminology. The recommendations describe testing at the router level, compounding the effects of system interfaces, control, and switching fabric. While the suggested tests take into consideration table size and prefix distribution, they lack specificity in how prefix distributions should be varied. The testing methodology also purposely omits dynamic table update effects, which is a realistic concern in a dynamic routing environment where route updates arrive at millisecond intervals. The recommendations do introduce a methodology for determining the maximum FIB size and evaluating throughput relative to the table size.

To our knowledge, there are no current efforts to provide a benchmark for multi-dimensional filter matching. In the absence of publicly available packet filter databases and backbone packet traces, researchers have exerted much effort in order to generate realistic performance tests for new algorithms. Several research groups obtained access to real filter databases through confidentiality agreements [10]. Gupta and

McKeown obtained access to 40 real filter databases and extracted a number of useful statistics which have been widely cited [1]. In [2], Gupta and McKeown generated synthetic two-dimensional databases consisting of source-destination address prefix pairs by randomly selecting destination address prefixes from publicly available route tables. Baboescu and Varghese also generated synthetic two-dimensional databases by randomly selecting prefixes from publicly available route tables, but added refinements for controlling the number of zero-length prefixes (wildcard) and prefix nesting [3]. A simple technique for appending port ranges and protocols from real databases in order to generate synthetic five-dimensional databases is also described in [3]. In [4], Baboescu and Varghese introduce a simple scheme for using a sample database to generate a larger synthetic five-dimensional database. This technique replicates filters by changing the IP prefixes while keeping the other fields unchanged. While these techniques address some aspects of scaling databases in size, they lack high-level mechanisms for adjusting the generation of new filter types and the general specificity of filters. Our technique for generating synthetic five-dimensional databases addresses these issues and provides a more flexible foundation for a packet classification benchmark.

As with previous approaches, our technique uses a real database to seed the synthetic generator; hence, there still exists a need for a larger sample space to accurately characterize "real" databases as they scale. Consolidating efforts to gather database samples this enables contributions from a larger body of researchers and promotes the development of a meaningful benchmark. Generation of this benchmark could eliminate the significant access barrier to realistic test vectors for researchers and designers.

3 Characteristics of Seed Databases

Previous work reported many statistical characteristics of filter databases useful for constructing fast search algorithms. In a similar fashion, we examine seed databases and extract statistics essential for constructing larger synthetic models with similar structure and characteristics. Selection of the relevant statistics is based upon experience garnered from a detailed study of five firewall databases of modest size. This study paid particular attention to the five-dimensional structure of the databases and the "correlation" among fields. In the context of our discussion, "correlation" is generally defined as the probability that two fields share the same value or initial bits, and will be more precisely defined in the following subsections.

3.1 Tuple Distribution

From a geometric perspective, a filter defines a region in 5-dimensional space. The volume of that region is the product of the one-dimensional "lengths" specified by the filter. For example, length in the source address dimension corresponds to the number of addresses covered by the source address prefix of the filter. Points in the 5-dimensional space correspond to packet headers; hence, the geometric volume translates to the number of possible packet headers that match the filter. Instead of geometric lengths and volumes, we often refer to filter properties in terms of a *tuple* specification. To be specific, we define the filter 5-tuple as vector containing the following fields.

- $t[0]$, source address prefix length, $[0...32]$
- $t[1]$, destination address prefix length, $[0...32]$
- $t[2]$, source port range width, the number of port numbers covered by the range, $[0...2^{16}]$
- $t[3]$, destination port range width, the number of port numbers covered by the range, $[0...2^{16}]$
- $t[4]$, protocol specification, Boolean value denoting whether or not a protocol is specified, $[0, 1]$

Table 1: 5-tuple scope measurements for firewall databases.

Database	# filters	μ_{scope}	σ_{scope}
rules1	279	50.7	15.7
rules2	183	54.1	14.5
rules3	68	56.6	23.1
rules4	158	55.6	16.7
rules5	264	48.1	27.9
Average	190	53.0	19.6

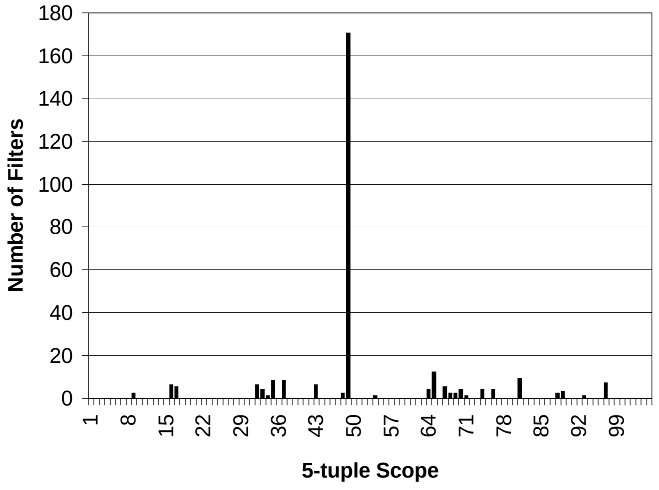

Figure 2: 5-tuple scope distribution for filters in firewall database *rules1* consisting of 279 filters. $\mu = 46.4$, $\sigma = 15.3$

The tuple defines the structure of the filter without specifying the actual values for address prefixes, port ranges, and protocol fields. To facilitate database measurements and synthetic database generation, we define a new metric, *scope*, to be lg*(# of possible packet headers covered by the filter)*. Specifically,

$$\begin{aligned} scope &= \lg\{(2^{32-t[0]}) \times (2^{32-t[1]}) \times t[2] \times t[3] \\ &\quad \times (2^{8(1-t[4])})\} \\ &= (32 - t[0]) + (32 - t[1]) + (\lg t[2]) + (\lg t[2]) \\ &\quad +8(1 - t[4]) \end{aligned} \tag{1}$$

Scope translates the filter tuple into a measure of filter specificity and is essentially just the logarithm of the geometric volume. Results from scope measurements for five firewall databases is given in Table 1. While the average scope of the databases does not vary drastically, the distributions of filter scope can exhibit drastically different characteristics. Figure 2 shows the 5-tuple scope distribution of database *rules1*. Note that most filters share a scope of 48. Figure 3 shows the 5-tuple scope distribution of database *rules5*. Note that there are several spikes in the distribution including a spike of 56 filters, or 21% of the total, that are fully specified and have a scope of zero.

Scope provides a high-level measure for the specificity of filters in the database and is one of the tunable parameters for the synthetic database generator which will be discussed later. While it is a meaningful measure, scope by itself does not characterize the five-dimensional distribution of tuples. In order to help visualize the tuple distribution, we plot the joint source/destination address prefix distributions for databases

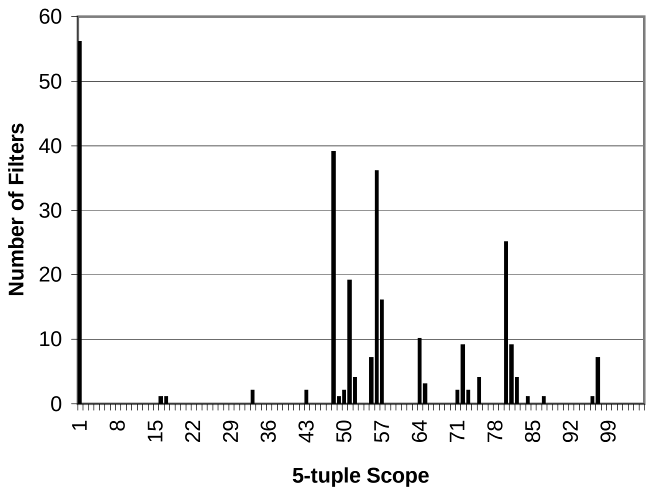

Figure 3: 5-tuple scope distribution for filters in firewall database *rules5* consisting of 279 filters. $\mu = 48.1$, $\sigma = 27.9$

rules1 and *rules5* in Figure 4 and Figure 5, respectively. Note that the most common source/destination prefix combination for *rules1* is a fully specified destination address and a wildcard source address. Unlike prefix distributions in backbone route tables, *rules1* has very few length 24 prefixes. The distribution for *rules5* exhibits significantly different characteristics. The most common source/destination prefix combinations are fully specified for both prefixes, length 24 destination address and wildcard source address. These figures provide strong motivation for accurately modeling the five-dimensional distribution of tuples, as the structure of these distributions can vary significantly.

A database generated via weighted, independent selection of source and destination address prefix lengths and random selection of port range widths does not produce the joint distributions observed in real databases. In order to accurately capture the relationships among prefix lengths, port range widths, and protocol specification, we extract the tuple distribution from the seed database. The distribution consists of unique tuple specifications and relative weights corresponding to the frequency of their occurrence in the seed database. Relative tuple weights are used during the synthetic database generation process to select tuples specifications for filters. Creation of new tuple specifications is enabled and controlled by a *smoothing* parameter which will be discussed in Section 4.1.

3.2 Address Prefix Skew

While the tuple specifies prefix lengths, it does not specify the composition of the prefixes themselves. Ideally, we would like to completely characterize the structure of address prefixes for the purpose of retaining this structure during the synthetic database generation process. Consider a binary tree constructed from the IP source address prefixes of all filters in the seed database. From this tree, we could completely characterize the data structure by determining a branching probability for each node. For example, assume that an address prefix is generated by traversing the tree starting at the root node. At each node, the decision to take to the 0 path or the 1 path exiting the node depends upon the branching probability at the node. For a complete characterization of the tree, the branching probability at each node is unique. As shown in Figure 6, $p10*$ is the probability that the 1 path is chosen at level 2 given that the 1 path was chosen at level 0 and the 0 path was chosen at level 1.

Such a characterization is infeasible, hence we employ a suitable approximation to capture the important characteristics such as prefix containment. Using the source and destination address prefixes for the filters

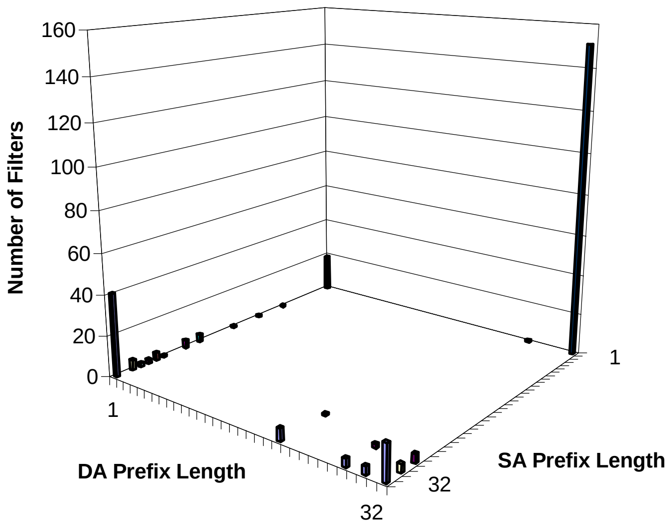

Figure 4: Joint source/destination prefix distribution for firewall database *rules1*.

in the seed database, we build two binary trees. For each level in the tree, we compute the probability that a node has zero children, one child, and two children. For nodes with two children, we compute *skew*, which is the ratio of the weights of the left and right subtrees of the node. Subtree weight is defined to be the number of filters specifying prefixes in the subtree, not the number of prefixes in the subtree. Since our goal is to eventually generate branching probabilities for the filter generation process, this definition of weight accounts for "popular" prefixes that occur in many filters. For example, assume that 10 filters specify prefixes in the left subtree of node k and 25 filters specify prefixes in the right subtree of node k, then the skew at node k is 2.5. For each level (depth) of the binary tree, we compute the average skew for nodes containing two children. The result of this analysis is two sets of distributions, one for source address and one for destination address, that characterize the expected number of children and average skew for nodes at each level of the address trees. Table 2 provides a child probability and skew distribution from the destination addresses in database *rules1*. Note that the probability that a node has no children is not reported in Table 2; hence, the child probabilities provided in columns two and three may not sum to one.

3.3 Address Prefix "Correlation"

Address prefix skew characterizes the structure of the individual source and destination address prefix trees; however, it does not capture their interdependence. It is possible that some filters in a databases match flows contained within a single subnet, while others match flows between different subnets. In order to capture this characteristic of a seed database, we measure the "correlation" of source and destination prefixes. In this context, we define correlation to be the probability that the source and destination address prefixes are equal up to a given number of bits. For example, if a database contained filters that all specified flows contained within class B networks, then the correlation for levels 0 through 15 would be 1, then fall off for levels 16 through 32 as the source and destination address prefixes diverge. From the seed database, we simply generate a probability distribution over the range of possible levels in the tree, $[0...32]$. For the five firewall databases we studied, the address prefix "correlation" varies widely. One database exhibits no correlation past level five while all address prefixes of another database are correlated up to level 17.

Table 2: Child probability and skew statistics for the destination addresses in *rules1*

Level	P{1 child}	P{2 children}	μ_{skew}	σ_{skew}
0	0.0000	1.0000	193.0000	0.0000
1	0.5000	0.5000	31.1667	0.0035
2	1.0000	0.0000	0.0000	0.0000
3	1.0000	0.0000	0.0000	0.0000
4	0.6667	0.3333	92.5000	0.0000
5	1.0000	0.0000	0.0000	0.0000
6	0.7500	0.2500	45.2500	0.0000
7	0.8000	0.2000	1.0000	0.0000
8	0.8333	0.1667	180.0000	0.0000
9	1.0000	0.0000	0.0000	0.0000
10	1.0000	0.0000	0.0000	0.0000
11	1.0000	0.0000	0.0000	0.0000
12	1.0000	0.0000	0.0000	0.0000
13	1.0000	0.0000	0.0000	0.0000
14	1.0000	0.0000	0.0000	0.0000
15	1.0000	0.0000	0.0000	0.0000
16	1.0000	0.0000	0.0000	0.0000
17	1.0000	0.0000	0.0000	0.0000
18	0.8571	0.1429	2.0000	0.0000
19	1.0000	0.0000	0.0000	0.0000
20	1.0000	0.0000	0.0000	0.0000
21	0.8750	0.1250	1.7903	0.0000
22	0.8889	0.1111	21.2000	0.0008
23	0.8000	0.2000	4.0357	2.5357
24	0.8333	0.1667	2.5282	1.3139
25	0.7857	0.2143	5.9935	3.4276
26	0.6471	0.3529	8.5836	11.0334
27	0.8696	0.1304	4.4722	2.1477
28	0.7308	0.2308	4.0833	2.3169
29	0.6774	0.3226	1.9503	1.0682
30	0.7073	0.2683	2.3955	2.0204
31	0.8431	0.1569	1.4167	0.8079
32	0.0000	0.0000	0.0000	0.0000

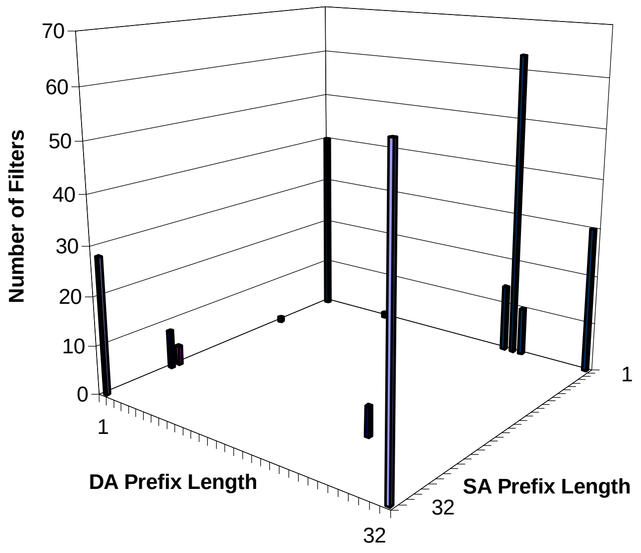

Figure 5: Joint source/destination prefix distribution for firewall database *rules5*.

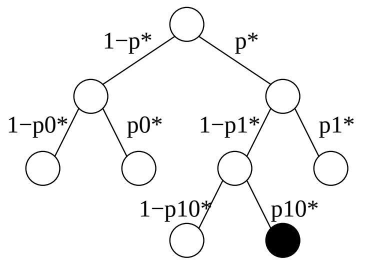

Figure 6: Example of complete statistical characterization of address prefixes.

3.4 Port Range Distributions

While the tuple distribution captures the inter-dependence of port range widths and other tuple fields, it does not characterize the choice of range bounds given a range width. Analysis of the specified port ranges in the firewall databases yielded an interesting result. All port range widths other than one (fully specified) correspond to a single port range. For example, all port range widths of 64512 specify port range [1024:65535]. The port numbers for fully specified port ranges approximate a uniform random distribution. Based on these results, it seems plausible to associate a specific port range for port range widths greater than one. Should a seed database specify several ranges for a given port range width, then a probability distribution for the set of possible ranges could be associated with the given range width. For fully specified port ranges, selection of a random port number in the range [0:65535] suffices.

3.5 Port Range “Correlation”

In the context of port ranges, we define “correlation” to be the probability that the source and destination port ranges are the same. Given the direct relationship between port range widths greater than one and the specified bounds of the range, the correlation for port ranges greater than one is maintained explicitly. For filters that fully specify both source and destination ports, we must determine the probability that the specified port numbers are the same. This is easily measured given a seed database and later used during the synthetic database generation process.

Table 3: Protocol selection distributions based on port range specifications for *rules1*. Note that GRE is General Routing Encapsulation.

Protocol	p_{ww}	p_{ws}	p_{ss}
ICMP	0.20	0.00	0.00
TCP	0.52	0.65	0.31
UDP	0.00	0.35	0.69
GRE	0.28	0.01	0.00

3.6 Protocol Specification

Finally, we must characterize protocol specification for filters that specify a protocol. Note that the tuple distribution dictates whether or not a filter specifies a protocol. We note that some protocols such as ICMP do not employ ports, while others such as UDP are free to specify any port number. After some preliminary analysis, we choose to characterize the relationship between protocol specification and port ranges using three distributions. The first distribution is the probability p_{ww} that a given protocol is specified when both port ranges are wildcarded. The second distribution is the probability p_{ws} that a given protocol is specified when one port range is specified and one port range is wildcarded. The third distribution is the probability p_{ss} that a given protocol is specified when both port ranges are specified.

Table 3 contains the three distributions for protocol selection based on port range specifications for *rules1*. Note that the set of specified protocols is dominated by ICMP, TCP, and UDP. Also note that, as expected, ICMP only occurs when both port ranges are unspecified. We performed additional analyses to confirm that specification of UDP or TCP is not strongly correlated to any particular port range specification or combination of specifications. The only other protocol specified in *rules1* was GRE (General Routing Encapsulation).

4 Synthetic Database Generation

A technique for generating synthetic databases is fundamental to building a packet classification benchmark. As previously discussed, our method of synthetic database generation uses a seed database which provides a realistic starting point for the process. While providing the ability to scale the total number of filters, we also would like to provide control over the variability or deviation from “real” database structure without manually editing probability distributions and other input parameters. In essence, we want to impose high-level controls over the composition of filters in the synthetic database. In the following subsections, we define two parameters which allow randomness to be injected into the filter generation process in a controlled fashion, as well as adjustment of the average 5-tuple scope of the synthetic database.

Prior to generating a synthetic database, the seed database is fed to an analysis tool which extracts the statistics and distributions described in the previous section and generates a seed file of parameters in a standard format. Specifically, the file contains a list of unique tuples with associated weights, child probability and skew distributions for both source and destination addresses, the address prefix correlation distribution, port range specifications for given port range widths, port range correlation for fully specified port combinations, and protocol distributions. We anticipate that a future benchmark will consist of a standard set of seed files of parameters from which a variety of metrics may be taken by varying the high-level controls for scale, smoothing, and scope.

The following list of steps provides an overview of the synthetic database generation process. Details of the important and non-obvious steps are discussed in the subsequent sub-sections. For each filter in the synthetic database:

1. Select a tuple specification from tuple distribution
2. Adjust the tuple specification based on the smoothing parameter
3. Adjust the tuple specification based on the scope parameter
4. Select source address prefix based on length from tuple and source address tree distributions
5. Select destination address prefix based on length from tuple, address "correlation", and destination address tree distributions
6. Select source port based on width from tuple; if exact, choose random port number; if not exact, select range based on distribution associated with width
7. Select destination port based on width from tuple; if destination is exact and source is exact, check for correlation; if destination is exact and (not correlated or source port is not exact), choose random port number; if not exact, select range based on distribution associated with width
8. If specified by tuple, select protocol using the distribution associated with the port range specifications
9. Verify that the generated filter is unique; if so, add it to the synthetic database; if not, generate a new filter

4.1 Smoothing Adjustment

As databases scale in size, we anticipate that new tuple specifications will emerge in the set of unique tuples. In order to provide for this possibility, we provide for the creation of new tuples in a structured manner. Injecting purely random tuples during the generation process neglects the structure of the seed database. Using scope as a measure of distance, we would like new tuples to emerge "near" an existing tuple. Referring back to the plot of the joint source/destination address prefix length in Figure 4, we would like new tuples to be clustered around the existing spikes in the distribution. This structured approach translates spikes in the distribution into smoother "hills"; hence, we refer to the process as smoothing.

In order to control smoothing, we define a smoothing parameter *r* which limits the maximum radius of deviation from a target tuple. Radius is measured in units of scope, or the $\lg($*5–d distance*$)$. For example, if the target tuple is [30,24,1,65536,0], then [32,26,1,32768,0] is a tuple at radius five from the target. A database generated with r set to zero would have a tuple distribution whose set of unique tuples and relative weights would be identical to the seed database. A database generated with r set to 104, the maximum value of scope, would have a tuple distribution whose set of unique tuples and relative weights would resemble a uniform random distribution with slight "bumps" at the sites of peaks in the seed tuple distribution.

The first step of generating a filter, regardless of the value of r, is selecting a *target tuple* from the tuple distribution of the seed database. If r is set to zero, then the target tuple is the tuple specification used for the filter. If r is greater than zero, then a smoothing adjustment is made to the target tuple; the scope of the tuple is shifted by some distance less than r. We would like the probability that we specify a tuple equal or near to the target to be greater than the probability that we choose a tuple farther away from the target. Hence, we define a geometric probability distribution for selecting the radius of the smoothing adjustment. Specifically, the probability that we select a radius i is:

$$p_i = (1 - \frac{1}{r})p_{i-1} \tag{2}$$

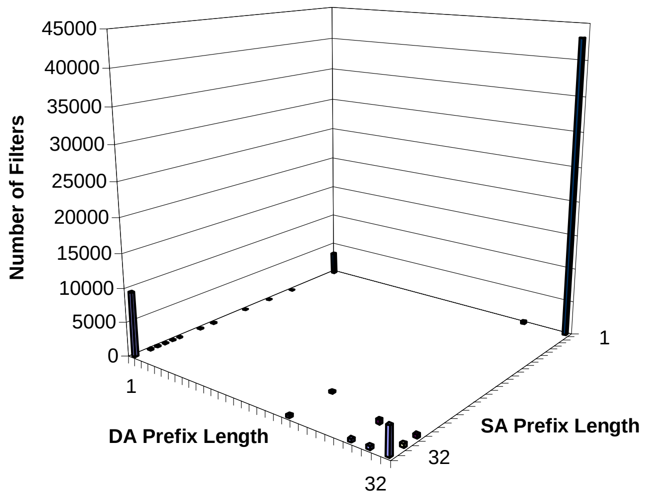

Figure 7: Joint source/destination prefix distribution for a synthetic database of 64000 filters seeded with firewall database *rules1* and generated with smoothing parameter $r = 0$.

Given that we are limiting the maximum radius, the distribution must be constructed subject to the constraint:

$$\sum_{i=0}^{r} p_i = 1 \tag{3}$$

Applying the definition of p_i to the constraint results in the following expression.

$$\begin{aligned} p_0 + (1 - \frac{1}{r})p_0 + (1 - \frac{1}{r})^2 p_0 + ... + (1 - \frac{1}{r})^r p_0 &= 1 \\ p_0 \sum_{i=1}^{r+1} (1 - \frac{1}{r})^{i-1} &= 1 \end{aligned}$$

The values of p_i for $i \epsilon [1...r]$ are readily available once p_0 is known. Solving for p_0 and using the closed form solution for the sum of the first $(r + 1)$ terms of a geometric series yields the following expression.

$$p_0 = \frac{1}{r[1 - (1 - \frac{1}{r})^{r+1}]} \tag{4}$$

From this closed form, we can generate a discrete distribution with r terms that sums to one.

After generating the distribution based on the smoothing parameter r, a radius i is selected for the smoothing adjustment. We now specify a process which provides for uniform random selection among the tuples at radius i from the target tuple. For each dimension, we randomly select a direction of scope adjustment. For example, increase scope of source address prefix, decrease scope of destination address, etc. This direction of adjustment remains static during the smoothing adjustment to the tuple of the current filter, but a new selection of directions is made for each filter. Dimensions are then selected at random for scope adjustment.

In order to demonstrate the effect of smoothing, Figure 7 shows the joint source/destination prefix distribution for a 64000 filter databases generated using *rules1* as a seed database with the smoothing parameter r set to zero. Figure 8 and Figure 9 show the same distribution for smoothing parameters of 16 and 64, respectively. As expected, increasing the value of r injects more unique tuple distributions into the synthetic database while maintaining the structure present in the seed database.

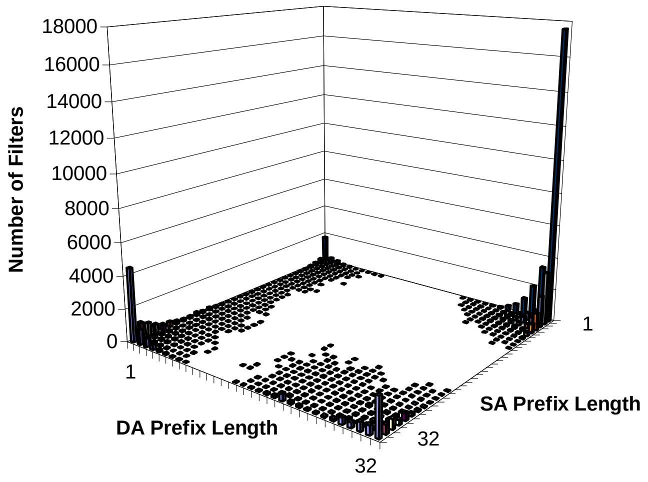

Figure 8: Joint source/destination prefix distribution for a synthetic database of 64000 filters seeded with firewall database *rules1* and generated with smoothing parameter $r = 16$.

Given that the domain of possible tuple specifications is finite, smoothing adjustments may have an effect on the average scope of the database. Note that many of the common tuples reside at the edge of the tuple domain, such as fully specified in one dimension and unspecified in another. Even though we are randomly selecting the direction of smoothness adjustments, target tuples at the edge of the tuple domain limit the possible directions of adjustment. In order to evaluate the significance of this effect, we plot the average scope of synthetic databases generated with *rules1* and *rules5* as seed databases and various values of r in Figure 10. Note that as r is increased, the average scope of the databases approaches the median value of 52 and the standard deviation slightly decreases. This is logical behavior considering that a uniform random distribution of tuples should have an average scope equal to the median value.

4.2 Scope Adjustment

As filter databases scale, it is logical to assume that the average scope of the database will change. As the number of flow-specific filters in a databases increases, the specificity of the database increases and the average scope decreases. If the number of explicitly blocked ports for all packets in a firewall database increases, then the specificity of the database may decrease and the average scope may increase.

In order to explore the effect of average database scope on the performance of algorithms and packet classification devices, we provide for adjustment of the average scope of the synthetic database. An input scope parameter s specifies the relative adjustment of average database scope and may assume values in the range $[-104...104]$. For example, if the synthetic database would naturally have an average scope of 50, then specifying a scope parameter s equal to -16 would attempt to shift average scope of the database to 34. The reason why the average scope cannot be shifted by exactly the amount specified by s will become apparent in a moment.

Following the selection of a target tuple from the seed tuple distribution and smoothing adjustment, scope adjustment is performed when s is non-zero. The process of applying the scope adjustment to a tuple is very similar to the smoothing adjustment with a few important exceptions. Instead of randomly selecting a direction of adjustment for each field, the direction of adjustment is static and specified for all fields by the sign of the scope parameter. Instead of randomly selecting fields for adjustment, a random starting field is selected. The fields are then adjusted in a regular cycle until s adjustments have been made or a tuple domain boundary has been reached. For example, if s is set to -4 and the source port is selected as the starting field,

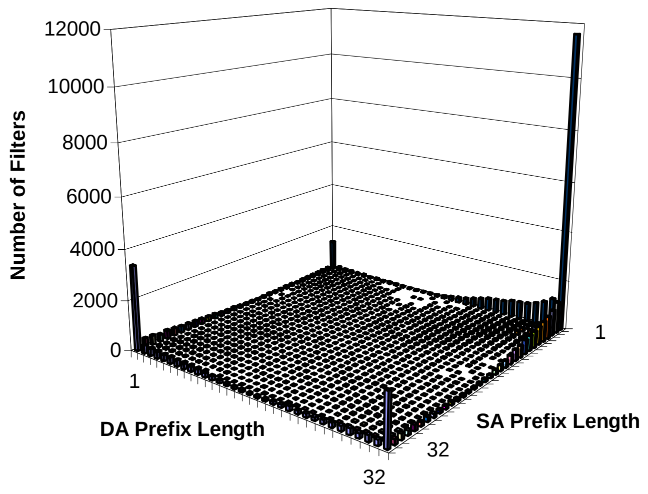

Figure 9: Joint source/destination prefix distribution for a synthetic database of 64000 filters seeded with firewall database *rules1* and generated with smoothing parameter $r = 64$.

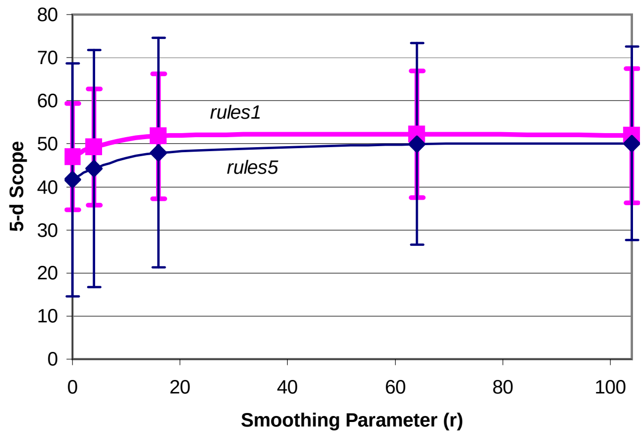

Figure 10: Five-dimensional scope of synthetic databases consisting of 64000 filters seeded with firewall databases *rules1* and *rules5*, generated with scope parameter $s = 0$ and various values of smoothing parameter r. Lines indicate average 5-d filter scope and error bars indicate standard deviation.

the scope adjustments would be the following: divide source port width by 2, divide destination port by 2, set the protocol field to specified, and decrement the source address prefix by 1. This process uniformly distributes the scope adjustment across tuple fields. In cases where a tuple domain boundary has been reached, no further scope adjustment can be made. It is these cases that prevent an explicit shift of the average scope of the database by the given scope parameter s. A plot of the five-dimensional scope of synthetic databases of size 64000 filters generated with *rules1* and *rules5* as a seed databases, smoothing parameter $r = 16$ and various values of s is shown in Figure 11.

Recall that the average five-dimensional scope of seed databases *rules1* and *rules5* is 50.7 and 48.1, respectively. For values of s in the range $[-16...16]$, the average scope of the generated databases scales linearly according to s. Generating large databases with values of s greater than 16 presents a challenge, as the number of unique filters from which to choose decreases with increasing scope. In this situation, the synthetic database can become saturated with filters of large scope making it extremely unlikely that the tool will be able to generate a unique filter. While this property limits the range of our study of the effects

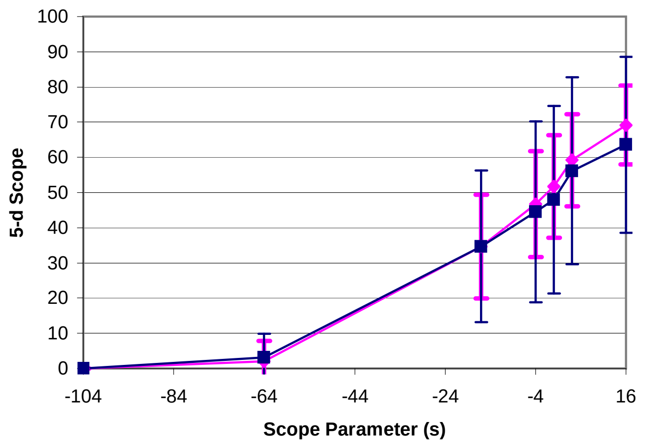

Figure 11: Five-dimensional scope of synthetic databases consisting of 64000 filters seeded with firewall databases *rules1* and *rules5*, generated with smoothing parameter $r = 16$ and various values of the scope parameter s. Lines indicate average 5-d filter scope and error bars indicate standard deviation.

of tuning the s parameter, we argue that it has negligible implications on the practical use of the tool. We anticipate that s will be tuned to values within the "linear region" or set the increasingly larger negative values as database size is scaled up.

4.3 Source & Destination Address Prefix Generation

As described in the previous section, a child probability and average skew distribution is extracted from the seed database for both source and destination address prefixes. We use these distributions to dynamically build source and destination address trees during the filter generation process. For each filter, we begin by constructing a source address prefix by traversing the source address tree according to the length specified by the tuple. Each tree node contains a flag indicating whether or not it has been assigned a branching probability. If the tuple specifies a prefix length greater than the level (tree depth) of the current node, then a branching decision must be made. If the node's flag is set, then the right (one) or left (zero) child is selected as the next node using the assigned branching probability. If the node's flag is not set, then a branching probability must be assigned prior to making a branching decision.

The branching probability for a node is affected by three parameters associated with the node's level: the probability that the node has one child, the probability that the node has two children, and the skew. Note that the probability that a node has no children is dictated by the prefix length in the tuple. When assigning a branching probability to a node, we know that the node must have one or two children; therefore, the child distribution must be adjusted accordingly. Once the number of children is selected according to the adjusted child distribution, we choose a random "heavy path" specifying right or left (one or zero). If the node has one child, then the child is assigned to the "heavy path". If the node has two children, then a branching probability is assigned according to the skew, where the more probable path is the "heavy path". The source address prefix is constructed by recording the branching decisions made at each level during the tree traversal.

Generation of the destination address prefix is similar to the process for the source address prefix. The difference lies in accounting for address prefix "correlation". As previously described, correlation is defined to be the probability that the address prefixes are identical up to the given level. We begin by determining the maximum possible depth of correlation, *max_depth* = *min*$\{$*(source address length),(destination address length)*$\}$, for the address prefixes in the filter. When the level of the current node is greater than *max_depth* or after the address prefixes have diverged, the destination address tree is traversed and branching probabilities are assigned exactly as described for the source address tree. If the level of the current node is less than or equal to *max_depth* and the address prefixes have not yet diverged, then the correlation probability for that level is used to determine the branching direction. If a branching probability has not been assigned for the node, then the value of the next source address prefix bit is used as the "heavy path". After computing the branching probability, the more probable direction is associated with the "heavy path" and the process continues at the child along that path. If a branching probability has been assigned for the node, then it is simply ignored in the case of correlated address prefixes. Note that this creates a conflict in the case when a node has a single child on the path opposite the path specified by the correlation. In this case we adjust the branching probability of the node to provide for two children. We observe that these conflicts do not cause significant deviation from the original destination address tree structure of the seed database.

4.4 Port & Protocol Generation

The port range distributions extracted from seed databases specify the range bounds for given port range widths of size greater than one. Should a port range width greater than one specify more than one set of range bounds, then a probability distribution for the set of range bounds is associated with the range widths. When the tuple specifies a port range width represented in the distribution, the associated range bounds are used for the port range in the filter. When the tuple dictates a fully specified port number, values are selected from a uniform random distribution over the possible range of port numbers. When the tuple dictates that both port numbers are fully specified, a source port value is selected from a uniform random distribution over the possible range of port numbers. Based on the correlation present in the seed database, the destination port value is either set to the same value as the source port or randomly selected.

When a synthetic database is generated with non-zero smoothing and/or scope parameters, port range widths not present in the seed database are likely to emerge in the synthetic filter tuples. In these cases, port range bounds are randomly selected from the set of admissible bounds for the given port range width. This logical and seemingly innocuous process can significantly affect the performance of some algorithms. Many researchers have observed that in the small sample space of available filter databases, there are no overlapping port range specifications. Port ranges $[x_i, x_j]$ and $[y_i, y_j]$ are defined to be overlapping if

$$(x_i < y_i) \cdot (x_j < y_j) \cdot (x_j \leq y_i)$$

or

$$(x_i > y_i) \cdot (x_j > y_j) \cdot (x_i \geq y_j)$$

Note that this definition allows for nested ports. While we also do not observe overlapping port ranges in the five firewall databases, there is no evidence to suggest that this property should be maintained as databases scale and change in composition. Adjustment of smoothing and scope parameters can help asses the effect of overlapping port ranges on protocol performance.

As previously described, protocol specifications depend upon the port range width specifications in the tuple. The seed database specifies the set of possible protocols and three distributions for protocol selection. Given that the set of unique protocols is relatively static, we do not provide a mechanism for injecting new protocol specifications. If the tuple requires a protocol to be specified, the port range widths of the tuple determine the distribution used for the selection process. If both port ranges in the tuple are unspecified (wildcarded), then the associated probability distribution for p_{ww} is used. If one port ranges is unspecified and one port range is specified, then the associated probability distribution for p_{ws} is used. If both port ranges are specified, then the associated probability distribution for p_{ss} is used.

4.5 Filter Priority

Traditionally, filter priority is inferred by placement in an ordered list. In the case of a linear search, the first matching filter is the best matching filter. This arrangement could obviate a filter f_i if a less specific filter $f_j \supset f_i$ occupies a higher position in the list. To prevent this, filters in the synthetic database can be ordered according to scope, where filters with minimum scope occur first.

5 Towards a Benchmark

Packet classification enables a wide variety of network applications such as security firewalls, per-flow queuing for guaranteed bandwidth allocation, per-flow packet processing, network monitoring and measurement, and usage-based accounting. In order to provide value to the interested community, a packet classification benchmark must provide meaningful measurements that cover the broad spectrum of application environments. It is with this in mind that we designed the synthetic database generation tool to be flexible while hiding the low-level details of database structure. While it is unclear if real databases will vary as specified by the smoothing and scope parameters, we assert that the tool provides a useful mechanism for measuring the effects of database composition on classifier performance. We now propose a framework for a packet classification benchmark utilizing this tool. It is our intention to initiate and frame a broader discussion within the community that results in a larger sample space of seed databases as well as the formulation of a standard benchmark. While we have focused on accurate modeling of five dimensional databases, general packet classification may range from one-dimensional longest prefix matches (LPM) on the destination IP address to more complex filter matches on header fields in layer four and above.

Packet classification algorithms and devices range from purely conceptual, to software implementations targeted to a variety of platforms, to state-of-the-art ASICs (Application Specific Integrated Circuits). For the purpose of a robust benchmark, we present a generic packet classifier model as shown in Figure 12. In this model, the classifier consists of a search engine connected to memory which stores the database and any other data structures required for the search. For each packet header passed to the classifier, the search engine queries the database and returns an associated flow identifier. Note that the set of possible flow identifiers is application dependent. Firewalls may only specify two types of flows, admissible and inadmissible, whereas routers implementing per-flow queuing may specify thousands of unique flow labels. The type of search (or dimensionality of the search) to be performed is specified by the configuration control. In order to model application environments where per-flow filters are dynamically created and deleted, the generic classifier model provides a mechanism for dynamic database updates.

There are three primary metrics of interest for packet classification algorithms and devices: throughput, memory requirement, and power consumption. For packet classification devices or fixed implementations of algorithms, throughput can be directly measured throughout the course of an experiment. Throughput measurement for algorithms is not as straight-forward. In this case, the metric most directly influencing throughput is the required number of *sequential* memory accesses. Using parallel and pipelined design techniques, non-sequential memory accesses can be masked. The benchmark should also provide the total required memory accesses in terms of average, worst observed, and best observed. The second metric of vital interest is the amount of memory required to store the database and supplemental data structures. For classification techniques employing random access memory, garnering memory usage metrics is straight-forward. For TCAM-based devices, memory usage can be measured in terms of storage efficiency, which is defined to be the ratio of the number of required TCAM slots and the number of filters in the database. Utilizing the synthetic database generator the effect of database size, average scope, and smoothness on throughput and memory usage can be measured.

In the past, power consumption has not been a primary concern for those developing new packet classification techniques. Recently, TCAM-based classifiers have become the most popular solution for high

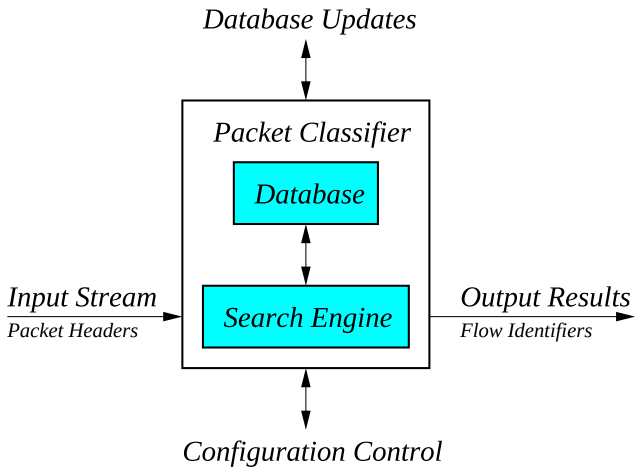

Figure 12: Generic model of a packet classifier.

performance routers, but they suffer from high power consumption. A typical TCAM consumes more than 100 times the power of state-of-the-art SRAMs operating 3 times faster and can account for a large fraction of the power budget on a router interface card. Recent developments in TCAM technology provide for partitioning the device such that only a subset of the available slots are activated at one time. IP lookup and packet classification techniques can take advantage of this capability to lower power consumption [11]. The effect of database size, scope, and smoothness on partitioning the route tables and databases in order to lower power consumption can be measured using the synthetic database generator.

Generating query patterns for the databases used for testing remains a large challenge. Worst-case performance is often the primary concern of designers, but for most packet classification techniques is deterministic. Creating a general technique to identify the pathological query pattern for a classifier under test is infeasible. Instead, the benchmark should provide throughput measurements for a variety of arrival patterns that fully exercise the database. We anticipate that these metrics will be useful for evaluating competing techniques or products under realistic operating conditions. Due to the use of pipelined implementations and caching, we propose phased query patterns with varying locality and frequency of reference. By frequency of reference, we mean the average period between packet headers matching the same filter. By locality of reference, we mean the size of the geometric space probed by the packet headers during a give phase. Filter nesting and overlap complicates the process of generating phased query patterns that fully exercise the database. Generating a packet header via random completion of the unspecified bits for a given filter does not guarantee that the filter is the best matching filter in the database for the packet header. Generation of update patterns and checking for correctness is also complicated by filter nesting and overlap.

Given that the synthetic database generator utilizes a seed database, there remains a need to assemble a standard set of seed databases representative of the wide variety of applications for packet classification. We believe that a consolidated effort with the purpose of developing a benchmark will be far more effective at gaining access to real databases than individual efforts. Mechanisms exist to preserve confidentiality by scrambling the addresses in filters without distorting the statistical characteristics of interest. We also believe that a larger sample space of databases is essential for refining the synthetic databases generator.

6 Conclusions

With the desire to refine synthetic database generation methods and formalize a benchmarking methodology, we seek to initiate a broader discussion and solicit input from the community to help guide the remainder of this work. To facilitate this discussion, we have created an online forum. A link to this forum can be found at: `http://www.arl.wustl.edu/~det3/`. Input garnered from the community will be used to refine the synthetic database generator, assemble a standard set of seed databases, and formally specify a benchmarking methodology. While we have already found the synthetic database generator to be very useful, it is our hope to promote its broader use as a standard tool for evaluating packet classifier performance.

Acknowledgments

We would like to thank Venkatachary Srinivasan for making five firewall databases available for study.

References

[1] P. Gupta and N. McKeown, "Packet classification on multiple fields," in *ACM Sigcomm*, August 1999.

[2] P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," in *Hot Interconnects VII*, August 1999.

[3] F. Baboescu and G. Varghese, "Scalable packet classification," in *ACM Sigcomm*, August 2001.

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

[5] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for ip performance metrics." RFC 2330, May 1998.

[6] S. Bradner, "Benchmarking Terminology for Network Interconnect Devices." RFC 1242, July 1991.

[7] S. Bradner and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices." RFC 2544, March 1999.

[8] G. Trotter, "Terminology for Forwarding Information Base (FIB) based Router Performance." RFC 3222, December 2001.

[9] G. Trotter, "Methodology for Forwarding Information Base (FIB) based Router Performance." Internet Draft, January 2002.

[10] V. Srinivasan, S. Suri, and G. Varghese, "Packet classification using tuple space search," in *SIGCOMM 99*, pp. 135–146, 1999.

[11] G. Narlikar, A. Basu, and F. Zane, "CoolCAMs: Power-Efficient TCAMs for Forwarding Engines," in *Proc. of Infocom*, 2003.