Optimized Packet Classification for Software-Defined Networking

Optimized Packet Classification for Software-Defined Networking

K. Guerra Pérez, X. Yang, S. Scott-Hayward, S. Sezer
The Institute of Electronics, Communications and Information Technology (ECIT)
Queen's University of Belfast, UK

***Abstract*—Recent trends, such as Software-Defined Networking (SDN), introduce programmability to the network with the opportunity to dynamically route traffic based on flow descriptions. Packet header lookup is the first phase in this process. In this paper, we illustrate improved header lookup and flow rule update speeds over conventional lookup algorithms. This is achieved by performing individual packet header field searches and combining the search results. We propose that individual algorithms should be selected for packet classification based on the application requirements. Improving the network processing performance with our configurable solution will directly support the proposed capability of programmability in SDN.**

Keywords—Software-Defined Networking (SDN); Packet Classification; multi-dimensional algorithms; one-dimensional algorithms; Access Control List (ACL) rules.

I. INTRODUCTION

In order to support the explosion of cloud services and convergence of data centers based on virtualization technologies, network operators, services and product providers are driving a revolution in networking, which is known as Software-Defined Networking (SDN). The key differentiator between traditional networking and SDN is flow-based management of the network elements (e.g. switches, routers, etc.) leading to improved programmability, reduced latency and higher performance within the network [1].

Packet Classification is the first step in network processing for identifying different applications and protocols that exist on a network system. Layer 3-4 (in OSI model) processing is of interest in this research work, composed mainly by five individual fields (also called tuples), *Source and Destination Port fields*, *Source and Destination Internet Protocol Address fields and Protocol field*. Routers process each input packet according to a pre-defined rule that the packet matches. The rule information is defined by the data and the mask for each dimension or field; where the mask represents the possible wildcard configuration. An action is associated to each rule. If the rule set is an Access Control List (ACL), the action can be "to reject" or "to accept" the packet. Fig. 1 shows an example of an ACL rule set and an input packet, which matches against rule number two (R2).

There are two different methods of matching depending on the header field for Packet Classification. The first type is *Exact Matching* (EM), in which input data is compared with a given data set. A data set entry is only selected if it matches on every bit. The second matching type is *Wildcard Matching* (WM) in which there is possible masking of part of the input data. Consequently, the input data comparison does not have to form a complete match. *Longest-Prefix Matching* (LPM) is a special case of WM, which refers to algorithms that select the entry in a table of defined prefixes with the most matching bits. An example is shown in Fig.1, where the input packet matches with R2 and R3. The action is determined by R2 because it obeys LPM based on the destination IP address. The most commonly used approaches for LPM are Tree (or Trie) algorithms. *Range Matching* (RM) is another WM type, which searches the entry between different ranges of defined values in the structure. This type of matching is well suited to port field lookup [2] [3], which can also be seen in Fig. 1 (destination and source port range).

ACL Rules	Source IP Address	Destination IP Address	Source Port	Destination Port	Protocol	Action
R1	175.77.88.248/32	195.156.56.0/22	0 : 65535	1489 : 1489	0x06/0xFF	Deny
R2	175.77.88.254/32	195.156.61.102/32	0 : 65535	1700 : 1710	0x06/0xFF	Permit
R3	175.77.88.254/32	195.156.61.0/24	0 : 65535	0 : 65535	0x00/0x00	Deny

Packet :	175.77.88.254	195.156.61.102	1200	1708	0x06	➡ The Matching Rule: R2

Fig. 1. Example of a packet header match against an ACL rule set.

There are three main performance requirements regarding packet processing. These are fast lookup, low density and incremental update. There are two forms of Packet Classification: Stateless and Stateful Classification. In the Stateless Classification case, each packet is analyzed and its matching rule is applied independently. This type of classification requires a high lookup performance to enable packet processing at the network speed.

In the case of Stateful Classification, routing and security policy related information are cached as part of state information for each Flow Look-Up Table (*Flow LUT*). It is not therefore necessary for every packet header belonging to the same flow to be analyzed in terms of routing lookup and access permission [4]. This reduces the criticality of the lookup speed. However, with Stateful Classification, the system bottleneck is the *Flow LUT* where the flow patterns are stored. The *Flow LUT* contains a large number of entries. It must also be possible to change the table entries quickly in an updating rule. Consequently, any

algorithm that needs a re-construction of its structure for update is not suitable for Stateful Classification.

Packet Classification can be performed either by programming on the Central Processing Unit (software approach), by using Application-Specific hardware, or a combination. In order to address the needs of emerging technologies, such as SDN, only hybrid solutions (software and hardware together) comprised of multiple algorithms are expected to address corner cases, whilst maintaining performance for given memory technology and memory size. Several Packet Classification solutions have been proposed in the literature. Those algorithms are based on Stateless Classification whereby the lookup speed outweighs the importance of other parameters such as memory space or update time. In this work, we consider all requirements for stateful classification.

With the objective of optimizing the lookup performance for the highly dynamic SDN environment, a study of existing lookup methods has been performed. We then analyse the performance of individual methods based on a set of ACL rule filters. We determine that the optimum approach is a combination of parallel one-dimensional methods selected by application type. Our implementation based on the label method is well-suited to current hardware devices and to the programmable platform of SDN, providing greater flexibility than previous algorithms.

II. LOOKUP APPROACHES AND IMPLEMENTATIONS

It is well-known that Ternary Content Addressable Memory (TCAM) is often used in routers for very high lookup speed and wildcard support. However, it is a device with high cost and high power consumption. Furthermore TCAMs are not suitable for accommodating large numbers of rule filters due to high cost of manufacturing. The memory size of a TCAM chip is limited to 1M entries using 144-bit words for IPv4 [5].

In order to support the programmable SDN platform, this work focuses on optimal algorithms whose lookup performance is comparable to TCAM. Thus, several lookup algorithms based on multi-dimensional and one-dimensional approaches are explored and discussed. We evaluate their performance with regards to memory accesses for lookup and update processes and memory requirements.

A. *Multi-dimensional Lookup*

Algorithms belonging to this methodology use five tuples of the packet header, either separately or independently. These algorithms can be classified into three main groups in terms of the different structural approaches: Geometric, Decomposition and Hashing.

Optimized tree-based lookup approaches are proposed as alternative solutions for TCAM. HyperCuts [6] is a heuristic tree algorithm based on multi-dimensional space division. In this method, each node defines a hyper cube and each packet header defines a point in this space. When the corresponding leaf node is found, the highest priority matching rule is searched linearly, resulting in a lookup dependent on the number of rules. Several algorithms, such as [7], are focused on improving HyperCuts performance.

The classic decomposition-based algorithm is Recursive Flow Classification (RFC) [8], which works with the packet header partitioned in parallel. The decomposition algorithms present high-speed lookup but require large memory storage. For example RFC requires 1.62 Gbits memory to perform lookup in four phases. However, due to the high lookup performance of RFC, it is still of great interest in recent research [9].

Tuple Space Search (TSS) [10] is an example of using hashing algorithm. It is based on storing rule groups called tuples into a hash table, according to the length of prefixes of all dimensions. The input packet header must be processed by an additional algorithm, such as binary tree, in order to determine the corresponding tuple location. TSS performs a linear search to find the highest priority rule.

The lookup performance of HyperCuts, RFC and TSS is compared in Fig. 2. To produce these results, the algorithms from [11] were modified. The evaluation in terms of number of memory accesses and memory bit consumption uses 10K ACL rules (filter set *acl1_10K* [11]).

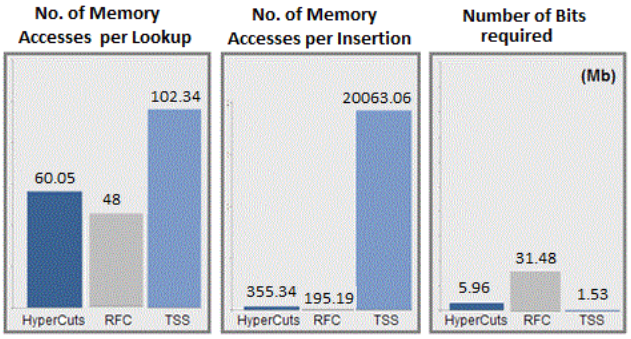

Fig. 2. Lookup performance of multi-dimensional algorithms

It can be seen that TSS consumes less memory bits since the rules are stored only once in the tables. However, this method requires more memory accesses in both the lookup stage and the rule insertion stage, making this method less suitable for high lookup speed. The algorithms that use parallel search, such as RFC, present better results for insertion and lookup processes, as shown Fig 2.

B. *Combination of One-dimensional Lookups*

Methods based on single-header-field search are insufficient for performing classification of an input packet. However, handling fields independently presents advantages regarding lookup speed and/or update complexity. The individual results from each header field are combined to search for the matching rule. In this section, we present several algorithms that focus on header field splitting, characterized for one-dimensional lookup applications, such as LPM, RM, etc.

Asymmetrical Multi-bit Trie (AM-T) [2] is a multi- bit structure, where every child position is calculated using a redundant expression, assuming that all header fields can be represented by prefixes. AM-T uses parallelism with one trie for each dimension. According to the experimental study of each one-dimensional trie, better results are achieved using tries with four levels for IP address fields. The rest of the fields use three-level tries.

Taylor et al. proposed Distributed Cross-producting Field Labels (DCFL) [12] to solve the Packet Classification problem. DCFL is a parallel lookup scheme that performs one-dimensional lookup for each packet header field. The results are combined and linearly searched in several filters in a phased process. It presents inefficient memory utilization and slow lookup speed in the filters.

Controlled Cross-Producting (CC-P) [13] improves the cross-producting algorithm transforming the range filters into prefixes. The algorithm constructs a prefix trie where each node is associated to a filter list. The final result is found in a final cross-product table. Table I summarizes the average number of memory accesses for the lookup process in the worst case, as well as the memory space required using ten thousand ACL rules.

TABLE I. PERFORMANCE EVALUATION OF COMBINATION ONE-DIMENSIONAL LOOKUP ALGORITHMS

Algorithm	Avg. Memory Accesses per Lookup (worst case)	Memory bits required
AM-T	44.99	186.11 Mb
DCFL [12]	23.1	22.54 Mb
CC-P [13]	30	20 Mb

AM-T lookup process is determined by the worst case trie, which corresponds to the Destination IP address trie. As shown in Table I, large memory consumption of AM-T is a result of the replication of rules in different tries. DCFL presents a faster lookup process but requires greater memory space than CC-P.

III. CONFIGURABLE ONE-DIMENSIONAL LOOKUP FOR SDN

The combination approaches discussed in the previous section are based on determined algorithms on each header field for lookup. In this section we analyze several algorithms with a view to selecting different algorithms for each header field in circuit run time. This supports re-configurability for the SDN framework. These algorithms were implemented in C++ and evaluated with different ACL rule sets.

A. IP Address Dimension Lookup

Algorithms for IP address lookup handle two fields from the packet header, i.e. source and destination IP address. As each header field is composed of a large number of bits (32 bits per packet for IPv4 (IPv4) or 128 bits per packet for IPv6), algorithms based on IP lookup have limitations regarding prefix length. Moreover, the rule for these IP address fields is formed by 32-bit or 128-bit data *plus* wildcard bits mask. For this reason, algorithms that support LPM are suitable for this field.

There are different tree/trie structures for IP lookup. One type of Tree/trie structures is based on LPM and the analysis is performed in data segments, according to the tree levels. Multi-bit search tree algorithms [14] or Binary search tree algorithms use this methodology and they are performed for IP Address dimension lookup. Binary search tree structures require a large number of memory accesses for update and lookup. These processes depend on the depth of the trie and, consequently, the length of the prefixes. Other types of Tree/trie structures are focused on range search where the comparison is performed according to given intervals. Segment tree and Range tree are included in this group.

It is possible to apply AM-T only to IP dimension lookup. As shown in Table I, it requires a lower number of memory accesses than multiple header fields as shown in Fig 2. IP address lookup comparison has been performed between Multi-bit trie algorithm and AM-T adapted to one-dimensional lookup.

Different scenarios were studied for IPv4 using two 16-bit tries per dimension in order to acquire the optimal parameter values. From our analysis, Multi-bit trie and AM-T present better results in terms of a tradeoff between the memory space and the number of memory accesses, when four (MultiT-4 and AM-T-4) or five levels (MultiT-5 and AM-T-5) are selected.

B. Port Address Dimension Lookup

The main challenge in a Port Lookup is to search intervals when a rule is being updated and perform a point search for the packet lookup. Two structures based on binary lookup using RM are studied in this section.

There are different variations of Range tree. Range Search tree proposed for this application is constructed while a new rule interval is inserted. Despite each node representing an interval, one main characteristic is that the tree node information is dynamically updated. It presents a fast lookup but is not suitable for long filters because updating requires a re-organization of the tree, resulting in slower insertion and deletion. We propose three possible range trees according to the interval split. The first case (Range-1) works with the complete 32 bits of the input data and, consequently, there is only one tree in each dimension. The second case (Range-2) implies that there are two trees for source port field and another two for the destination port field. Each tree analyzes 8-bit start-point interval and 8-bit end-point interval. The final case (Range-4) applies the same idea using 4 trees per dimension.

For a further comparison, we implement Segment tree, which is a fixed balanced data structure where each node defines a specific range. The child node contains part of its father interval. This static structure requires a build time and presents a straightforward update. Segment tree was

studied with division trees as well. The first case (Segm-4) is composed of 4 trees for each header field, handling 8 bits from the input Port interval. There are two other cases (Segm-5 and Segm-6) provided in our test.

C. Rule Filters Analysis and Performance Optimization

Different algorithms have been presented and analyzed for an optimal update and lookup performance. For a further optimized implementation, several parameters of different rule types are analyzed in this section.

Rule syntax has been widely researched. Trie-based algorithms for IP lookup are an example. This kind of method uses the presence of wildcards in the rule in order to differentiate the trie structure by defining parameters such as the trie depth or number of trie nodes. Likewise, TSS performs the grouping of tuples according to the number of non-wildcard bits in the rule fields.

In order to determine the importance of a method that supports LPM, a study of the wildcards was performed for three different ACL rule-sets (*acl1_1K*, *acl1_5K*, *acl1_10K*) composed by 917, 4415 and 9604 rules [11]. From this survey we can conclude that more than 50% of the rules are formed mainly by wildcards and, particularly, the source IP address field presents a greater index of wildcards. Consequently, the destination IP address field determines the worst case for the LPM methodology.

A further study was performed using the ACL rule set to analyze how the wildcards are distributed. Table II expresses the number of rules with a certain number of wildcard bits. It can be identified that most wildcards are concentrated in the last eight bits, followed by 16 bits. From the analysis, we conclude that the partition of 8-bit segments presents a potential optimization for lookup performance.

TABLE II. NUMBER OF WILDCARDS/RULE FOR IP FIELD

No. of wildcards/rule	acl1_1K		acl1_5K		acl1_10K	
	Src	*Dst*	*Src*	*Dst*	*Src*	*Dst*
Total	**610**	**534**	**2819**	**2428**	**5602**	**4862**
8 bits	273	256	1153	1146	2425	2301
16 bits	50	83	210	316	431	681

Another method which derived from analysis rule construction is DCFL. DCFL is focused on the repetition of rule fields. In our study, a significant number of unique rules for each dimension was extracted from a set of rule filters, as shown in Table III.

TABLE III. NUMBER OF UNIQUE RULES

Algorithm	acl1_1K	acl1_5K	acl1_10K
Source IP Address	103	805	4784
Destination IP Address	297	640	733
Source Port	1	1	1
Destination Port	99	108	108
Protocol	3	3	3

As previously mentioned, IP address is the largest field. The IP address field can be split into two or more partitions in order to make it more manageable. Similarly the decomposition methodology presents performance time advantages and thus, each IP address partition can be performed in parallel. With this consideration, Table IV summarizes the number of different partitions when the IP address for source and destination fields are divided into two 16-bit sets; high part and low part. Comparing Table III and Table IV, we can conclude that the partition of a field presents advantages with a decrease in the number of unique rule fields. For example, for acl1_1K filter there are 90 unique rules when the source IP address is split into 16-bits compared with 103 unique rules without partitions.

TABLE IV. NUMBER OF UNIQUE RULES WITH 2 IP ADDRESS PARTITIONS

IP address Field	acl1_1K		acl1_5K		acl1_10K	
	High	*Low*	*High*	*Low*	*High*	*Low*
Source	4	90	33	740	142	4393
Destination	130	257	236	496	380	609

Taking into account the above observations, a label method, which was introduced by DCFL, is presented for our algorithm combination, where each unique rule field is labeled. The label method is an efficient technique for algorithms with fixed structure such as Segment tree. In this method, a list of labels is stored in trie nodes instead of rule list. The lookup process traverses the tree in the same manner as before in order to find the corresponding label list. However, the update process of algorithms with dynamic structures such as Range Search tree require a re-configuration time for the node labels such that the label method is not applicable.

Table V shows the evaluation results of Segment Tree using the proposed label method. Using this approach, better Highest Priority Matching Rule (HPMR) lookup performance can be achieved. As most rules consist of the same header field, rule insertion is not always necessary for rule updates. A new rule is only inserted when its unique label is not in the tree, resulting in a faster lookup/insertion operation. Additionally, memory storage is reduced by avoiding replicated rules stored in the tree nodes. The same approach is also applied to IP address field lookup.

TABLE V. LABEL METHOD EVALUATION ON SEGMENT TREE

Method	Number of Memory Accesses				Memory Bits required (Mb)	
	HPMR lookup/packet		Insertion/rule			
	Label List	Rule List	Label List	Rule List	Label List	Rule List
Segm-4	49.3	>10K	5.21	10.02	1.249	11.40
Segm-5	31.33	8927	4.165	7.93	1.25	13.21
Segm-6	56.55	9063	4.17	7.92	1.29	16.56

D. Discussion

Table VI summarizes the results in terms of the average memory access per packet lookup, which is evaluated in

terms of HPMR and the list stored in nodes, node list (NL), per rule insertion and memory required for each algorithm. The results are presented after applying the label method to the suitable algorithms. Range Search tree does not use the label method, as previously described.

TABLE VI. PERFORMANCE EVALUATION OF ONE-DIMENSIONAL ALGORITHMS

Algorithm	Memory Accesses of Lookup per packet		Memory Accesses of Insert/rule	Memory bits required
	NL	*HPMR*		
IP Address field				
MultiT-4	2.55	28.8	5.39	5.11 Mb
MultiT-5	3.53	29.05	5.78	4.32 Mb
AM-T-4	2.8	40.8	10.8	8.86 Mb
AM-T-5	3.96	41.3	8.58	6.01 Mb
Port field				
Segm-4	5	49.3	5.21	1.249 Mb
Segm-5	5	31.33	4.165	1.25 Mb
Segm-6	4	56.55	4.17	1.29 Mb
Range-1	6.92		137.48	54.31 Mb
Range-2	3.48	>10K	18.22	13.38 Mb
Range-4	3.48	1440	18.22	25.34 Mb

Comparing the results shown in Fig. 2, Tables I and VI, it is found that by performing individual searches on each header field with algorithm combinations, better results in terms of lookup and update speed can be achieved. For example, AM-T-5 for IP lookup requires on average 41.3 memory accesses in the worst case. However, RFC requires 48 memory accesses for multi-dimensional search (Fig. 2). The presented algorithms in Table I suffer from a memory blowup. However, it can be seen that the results regarding lookup speed are considerably improved. In the worst case lookup, AM-T requires 44.99 memory accesses (Table I), exceeding the multi-dimensional lookup algorithms. The challenge is to reduce the total storage required and determine an efficient combination of different algorithms avoiding multiple rule copies.

Considering the IP address dimension lookup results from Table VI, we can conclude that, although MultiT-4 presents faster search of list in nodes and MultiT-5 requires the least amount of storage, Multi-bit search trie with 4 levels is optimal because it requires the lowest number of accesses. In addition, the Multi-bit algorithm presents a straightforward lookup and update process.

Considering the port dimension results, Segment tree surpasses Range Search tree in all evaluation parameters apart from NL lookup process. When Range Search tree is split, the lookup speed is increased due to the reduction in trie depth and the replicated rules in the nodes. In particular, Segment-5 presents a tradeoff between number of bits and the number of memory accesses. The label filter lookup is taken into consideration in the number of memory accesses of HPMR lookup.

Comparing the lookup approaches, the one-dimensional algorithms perform the lookup and insertion processes using fewer memory accesses per packet. The space requirement for one-dimensional algorithms is considerably lower than that for multi-dimensional algorithms that use parallel lookup. Nevertheless, the individual header field lookup algorithms share the same rule data. Therefore, an efficient architecture combining several one-dimensional algorithms with a shared rule filter is required.

According to the basic idea of Stateful Classification, where only the first packet of each flow is analyzed, the need for very high speed lookup is used only for flow set-up. Therefore, the lookup for Packet Classification is not time constrained as it is performed at the establishment of the flow. From a hardware perspective, a lookup latency of several cycles can be tolerated. This lookup throughput is sufficient to provide bandwidth for new flows.

IV. SYSTEM INTEGRATION

As discussed in the previous section, we determine that managing individual fields with efficient dedicated algorithms results in high performance Packet Classification. The challenge is how to lookup in parallel each header field in order to achieve higher search speed and an efficient memory distribution. According to this argument, each algorithm studied in this work presents advantages and disadvantages across three main evaluation parameters: lookup time, update speed and memory space. There is no unique algorithm which can handle five fields efficiently in the above three aspects. As a result, we propose a programmable system to select the optimal combination as shown in Fig. 3.

This hardware architecture is ideally suited for SDN with the separate control and data plane. In an example implementation, the host Packet Classifier selects the optimal algorithms combination according to the network application. This means that the appropriate algorithm per lookup dimension (header field) is selected to optimize the packet classification. The Configuration/Lookup controller is configured with the rules and algorithms specified by the host Packet Classifier.

For each packet, *Flow LUT* examines the packet header and classifies the packet into a flow. If this packet is the first packet of a flow, a flow ID is created in the *Flow LUT* and extracted packet header fields are passed to the Configuration/Lookup Controller for Packet Classification.

At this point the combined lookup algorithm method presented in this work is deployed to efficiently classify the packet. Following the lookup process, the Configuration/ Lookup Controller block returns an action determined by the matching rule. This action is stored in the Flow State block at the location indexed by the flow ID. For the following packets belonging to the same flow, a matched flow ID is output to the Flow State and the same action is retrieved directly from the Flow State block.

In an illustration of the performance of the combined algorithm approach, Table VII shows examples of algorithm configurations focused on improving one of the three key parameters for the Packet Classifier block. The optimal combination of algorithms for the Port field and the IP address field is presented for the worst case assuming a

parallel lookup. It should be noted that the protocol field is not included in Table VII. This is due to the fact that the protocol field only contains three values; TCP, UDP or *don't care,* as determined in the rule analysis of Section III.C. As such, the protocol field is easily implementable with a simple lookup table.

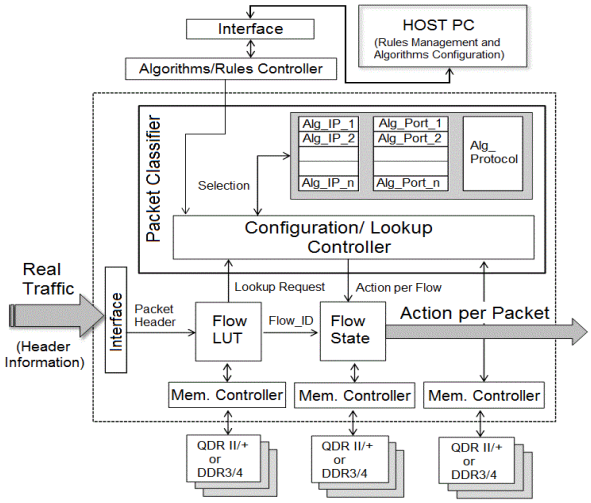

Fig. 3. Example implementation of the proposed HW architecture

TABLE VII. EVALUATION OF ALGORITHM CONFIGURATIONS

Main Parameters	Configuration		Performance			
	IP Field Method	Port Field Method	Memory required (Mbits)	No. of Memory Accesses for Update	Number of Memory Accesses for Lookup	
					NL	HPMR
Low Density	MultiT-5	Segm-4	5.57	5.78	5	49.3
Fast Lookup NL	AM-T-4	Range-4	34.2	18.22	3.48	1440
Fast Lookup HPMR	MultiT-4	Segm-5	6.36	5.39	5	31.33
Fast Update	MultiT-4	Segm-5	6.4	5.39	5	56.55
Tradeoff Lookup and Update	MultiT-4	Segm-5	6.36	5.39	5	31.33

The trade-off in system parameters is clear from the results in Table VII. For example, both fast lookup and update can be achieved with the MultiT-4 method applied to the IP field and Segm-5 applied to the Port field lookup. In comparison, if low density is required, in order to reduce the memory area on the chip, for example, then the MultiT-5 method is best applied to the IP field with Segm-4 applied to the Port field lookup.

Table VII shows results using 10K rules. In current network at least one million entries are essential for Packet Classification. Using the configuration for a tradeoff lookup and update, in the worst case, the memory requirement should be 10 times the current memory space required for 10K rules. It is possible to implement the system in current devices. As an example, Stratix V GX FPGA platform provides 8 Gbits DDR3-SDRAM and 36 Mbits QDR II SRAM memory devices and a maximum embedded memory capacity of 65 Mbits. TCAM supports 1M entries, whilst our proposed system can store 76M entries using this platform.

V. CONCLUSION

Packet classification requires lookup on multiple fields of the packet header. With increasing volumes of network traffic, the ability to perform fast, efficient Packet Classification is the key to meeting the security and performance requirements of carrier-grade networks. In this work, a range of lookup approaches have been tested against the criteria of memory access requirements for lookup and update and number of stored bits. An analysis of rule-sets has also been performed.

Based on our analysis, this work has identified that using a combination of different lookup approaches and performing lookups in parallel provides a distinct advantage over the multi-dimensional lookup method based on a unique algorithm such as RFC or HyperCuts. The proposed architecture for our implementation of label method is well-suited to the programmable platform of SDN, providing greater flexibility than the combination of one-dimensional lookup algorithms. Our future work will focus on optimizing memory accesses to maximize network traffic throughput including IPv6 and other packet header fields.

REFERENCES

[1] S. Sezer, S. Scott-Hayward, P. Kaur Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, N. Rao. "Are we ready for SDN?-Implementation Challenges for Software- Defined Networks."*IEEE Communications Magazine,* Vol 51, July 2013.

[2] B. Zheng, C. Lin and X. Peng, "AM-Trie: An OC-192 Parallel Multidimensional Packet Classification Algorithm for Network Processor". *IMSCCS'06.* Vol.1 pp.:377-384, 2006.

[3] I. Sourdis, G. Stefabakis, R. de Semt, G. N. Gaydadjiev, "Range Trie for Scalable Address Lookup", *5th ACM/IEEE ANCS'09*, pp. 143-152, 2009.

[4] "Stateful and Stateless Data Processing " [Online]. Available: *http://www.juniper.net.* Accessed April 2013.

[5] K. Kannan, S. Baneerje, "Compact TCAM: Flow entry compaction in TCAM for Power Aware SDN", *ICDCN,* volume 7730, pp. 439-444, 2013.

[6] S. Singh, F. Baboescu, G. Varghese, J. Wang "Packet Classification Using Multidimensional Cutting". *SIGCOMM,* pp. 213-224, 2003.

[7] B. Vamanan, G. Voskuilen, T. N. Vijaykumar, "EffiCuts: optimizing Packet Classification for Memory and Throughput", *SIGCOMM,* pp. 207-218, 2010.

[8] P. Gupta and N. Mckeown, "Packet classification on Multiple Fields", *SIGCOMM'99,* 1999.

[9] M. Dixit, A. Kale, M. Narote, S. Talwalkar, B.V. Barbadekar, "Fast Packet Classification Algorithms", *International Journal of Computer Theory and Engeneering,* Vol 4, no. 6, 2012.

[10] V. Srinivasan , S. Suri , G. Varghese. "Packet Classification using Tuple Space Search". *SIGCOMM*, 1999.

[11] H. Song *http://www.arl.wustl.edu/~hs1/project/filterset/.* Accessed on July 2013.

[12] D. E. Taylor and J.S. Turner, "Scalable Packet Classification using Distributed Crossproducting of Field labels", *IEEE INFOCOM 2005*, Vol. 1, pp.269-280, March 2005.

[13] P. C. Wang, " Scalable packet classification with controlled cross-producting", *Journal Computer Network* vol 53, pp. 821-834, 2009.

[14] X. Qui, L. Xu, B. Yang, Y. Xue , J. Li, "Packet Classification Algorithms: From Theory to Practice", *IEEE INFOCOM*, pp. 648-656, 2009.