International Journal of Engineering and Technology Volume 4 No. 3, March, 2014

HNPC: Hardware Network Packet Classifier for High Speed Intrusion Detection Systems

Nekoo Rafiei Karkvandi, Hassan Asgharian, Amir Kusedghi, Ahmad Akbari
Department of Computer Engineering, Iran University of Science and Technology

ABSTRACT

Increasing speed and bandwidth of network traffic and existence of related attacks require the intrusion detection system work in real time. Packet classification is an integrated part of a full featured Network Intrusion Detection System. Software solutions are available for the packet classification problem but their performance is not sufficient for wire speed processing in the high speed networks. In this paper a hardware solution is proposed to extract the packet features and process the packets for high throughput intrusion detection systems. Matching packets with some predefined rules in packet classification as the number of rules increase, causes extreme performance degradation. We propose two heuristic algorithms for fast search of a port number in range and searching IP address with mask values. According to the achieved results, the designed system is able to work with 10,000 different rules in 100 MHz clock which illustrates that our proposed system can work with link rate of 32 Gbps. The proposed algorithm has been tested by standard benchmarks and simulated in Modelsim 6.5 and further implemented on the FPGA device xc5vlx110t-3ff1136 using XillinxISE13.2 software.

Keywords: *intrusion detection, packet classification, hardware implementation, filter sets, port number*

1. INTRODUCTION

Computer networks are ubiquitous with reasonable prices. However, they deal with challenges such as unauthorized access, tampering the data and etc. The content of a packet implies its mission in the network. In intrusion detection system, a malicious activity can be detected by classification techniques such as searching packet headers, pattern matching methods for payload or calculating the traffic volume. Increase of network speed and number of attacks lead researchers to use some real-time algorithms with the capability of being updated easily. In these methods not only high speed processing is important, but also efficient implementation should be considered as well. There are various methods for intrusion detection. Packet header classification is an integrated part of a full featured Network Intrusion Detection System [NIDS] in which headers of packet matches some predefined rules [17]. Unfortunately in packet classification, searching rules for each packet could take lots of processing time of CPU [16]. Number of rules can be higher than thousands, so hundred megabytes of memory is required for the process. Analyzing such a huge category of rules may cause a bottleneck in high speed networks, due to the required memory for each packet [15]. Some attacks may generate many packets to confuse and disable the detection engine. Traffic analysis mechanism also uses matching rules algorithms such as pattern matching but final detection is performed by considering the traffic volume. On the other hand, when traffic volume exceeds the timing threshold, alarms would be asserted. If incoming packet matches with some rules, related counter is increased. If the counter is more than the threshold, an alarm should be sent to the CPU [1].

In this paper a structure for intrusion detection system is proposed that matches some rules with the packet. This structure is able to detect more attacks such as Attacks that increase the traffic volume intentionally by flowing a huge number of packets in the network. In order to extract the packet features, storage and classification methods are needed. Software solutions cannot be efficient in future high speed networks[23,24] because they are not fast enough for wire speed processing in the high speed networks, so in this paper a hardware solution is proposed to extract the packet features and process the packets for high throughput intrusion detection systems. Fast rule matching, efficient utilization, less memory consumption, less number of packet header fields needed are some contributions for packet classification method proposed in this paper.

In the next section some of the previous related works are described. In Section 3, the target problem is explained and our solution is presented in detail. Results of the implemented algorithm are explained in section 4. Performance comparison of the proposed solution with other published solutions is depicted in Section 5. Finally, conclusion and some potential future works are described in section 6.

2. RELATED WORKS

The goal of intrusion detection system is to monitor the computer networks in order to detect malicious activities infringing the security policies. This technology has been growing over 20 years and follows an intelligent monitoring strategy of the networks. Intrusion detection systems that are designed for working in high speed networks should have high throughput and flexibility against new threats. FPGA technologies are suitable for this purpose because of their high performance and reconfigurable architecture. Packet classification is done based on the header information specifically the main five fields concluding source IP address, destination IP address, source port number, destination port number and protocol. In packet classification, high performance algorithms have a complex structure such as Distributed Crossproducting of Field Labels (DCFL) [4]. Utilization and power consumption are two problems of proposed methods like Ternary Content Addressable Memory (TCAM). On the other hand, there are some algorithms like Hierarchical Intelligent Cuttings (Hicuts) [20] or Multidimensional Cutting (Hypercuts) [21] that are scalable but their throughput is not acceptable. There are some other proposed algorithms that may not be suitable for hardware implementation because of their special computing procedure [2]. The main performance metrics that should be considered in design of any packet classification system are as follows [2, 16]:

- **Resource limitations**: there is a tradeoff between processing time and memory usage per look up for packet classification solutions.
- **Number of supported rules**: number of classification rules depends on the place that the system will be installed on the network.
- **Used packet header fields**: the classification systems work based on some predefined features that are extracted explicitly or derived from network packet headers. The number of used header fields depends on the application. This may suffice For the routing IP address but in intrusion detection or firewalls more fields are required.
- **Nature of rules**: selected header fields and nature of rules should be affected by classification algorithms. For example port numbers can accept ranges that should be handled in classification algorithm.
- **Updating rule set**: since the number of attacks increase rapidly, the classification algorithms should support probable future updates.

According to the classification features and tradeoffs described above and considering the fact that packet classification is the heart of all intrusion detection and firewall systems, it is important to have a fast and scalable classification algorithm. Therefore, much hardware implemented classification systems are proposed for this purpose.

Different algorithms and architectures were proposed for the packet classification problem. Rather than categorizing the techniques based on their performance, memory requirements, or scaling properties, a different taxonomy is presented in [3] that breaks the design space into four regions based on the high-level approach to the aforementioned problem. It might be useful to present a comparison of this work with other methods from the design perspective. Similar methods with common characteristics are placed next to each other. In other words adjacent techniques are related and hybrid techniques have overlapped quadrant boundaries.

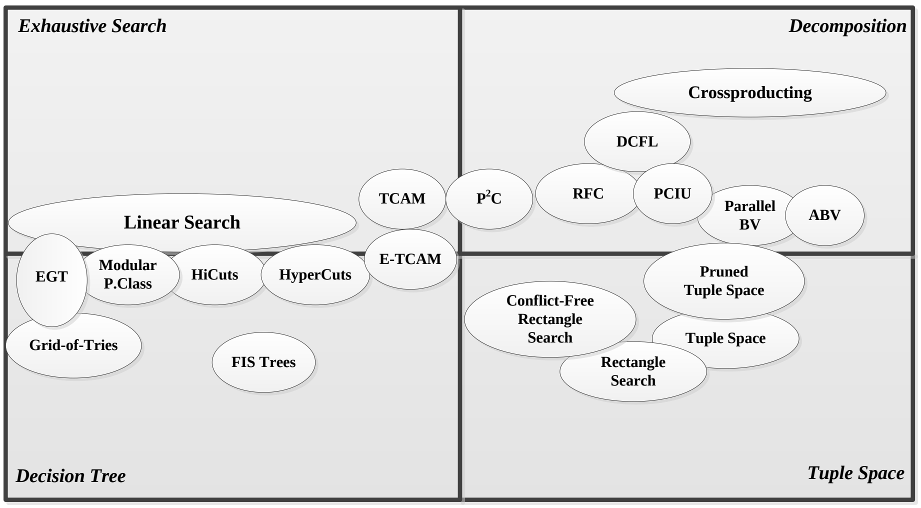


Fig. 1. Taxonomy of multiple field search techniques for packet classification [19]

In this taxonomy all methods break into four regions as follows.

1. **Exhaustive Search**: Examines all the entries in the filter set.
2. **Decision Tree**: Constructs a decision tree from the filters in the filter set and uses the packet fields to traverse the decision tree.
3. **Decomposition**: Decomposes the multiple-field search into instances of single field search and performs independent searches on each packet field and combines the achieved results.
4. **Tuple Space**: Partitions the filter set according to the number of specified bits in the filters and probes the partitions or a subset of the partitions using simple exact-match search.

Figure 1 shows an illustration of classification techniques scattered in the four regions. Some methods are combinations of multiple methods that are shown by being placed next to each other. The expected overlaps present the similarities between different methods.

Crossproducting is a fast searching method but is prone to the scalability problem [5]. BitVector and Recursive Flow Classification (RFC) are suitable for hardware implementation but their performance is not desirable. The Problem of slow searching can be improved by combining them with some algorithmic methods [6, 8]. TCAM is a high speed method which has the ability to check the input with multiple conditions simultaneously in one clock cycle [4] but it suffers from resource and power consumption problems. Parallelization techniques and combining them with algorithmic methods could be useful for reduction of resource utilization. In addition, TCAM is not efficient in some cases like to be applied for the port number which needs to search ranges. So Extended TCAM (ETCAM) is proposed in [7] to deal with the range searching issue but it confronts the scalability problem too.

A scalable high throughput firewall using an FPGA was proposed in [4]. The Distributed CrossProducing of Field Labels (DCFL) is used with the firewall application and an improvement has been achieved over DCFL by extending DCFL [4]. Fast IP Lookup (FIPL) method for searching IP is based on Tree Bitmap algorithm, performed in [22]. Scalable lookup engine design is able to achieve high-performance with the use of a small portion of the reconfigurable logic device [9]. The other well-known proposed algorithms are Hicuts and Hypercuts which work based on decision trees. In these algorithms, a few number of rules are stored and mathematic operations would perform a linear search. Long preprocessing time, no additional updates and having no fixed delay are problems to be mentioned for these methods [10, 11]. Timing and area complexity of these algorithms are presented by *"O"* in Table 1. In this table *"N"* represents number of rules, *"W"* is number of prefix bits or memory width, *"d"* is number of fields and t_{RL} is delay of the searching ranges for classification algorithm.

Table 1. Time and area complexity of previous proposed algorithm

Algorithm	Time Complexity	Area Complexity
Linear Search	$O(N)$	$O(N)$
Hicuts	$O(d)$	$O(Nd)$
RFC	$O(d)$	$O(Nd)$
Crossproduct	$O(dw)$	$O(Nd)$
TCAM	$O(1)$	$O(N)$
Bit Vector	$O(logN)$	$O(N^2)$
Bitmap	$O\left(dtRl+\frac{dN}{w}\right)$	$O(N^2)$
BitMapIntersection	$O\left(dw+\frac{N}{MemWidth}\right)$	$O(N)$
Perfect Hash	$O(wd)$	$O(N)$
PCIU	$O(d)$	$O(Nd)$
Proposed Algorithm	$O(1)$	$O(N)$

Decision tree-based packet classification algorithms are easy to implement but allow the tradeoff between storage and throughput. The memory consumption of these algorithms is quite high so many researches has been done to keep this feature acceptable while the throughput is high but like [18] they have not achieved enough improvement to compete TCAM[4] and ETCAM[7].

Recently [25] exploits parallelism and other features of field-programmable gate arrays to introduce a 2-D pipeline architecture for next generation packet classification which requires combination of numerous packet header fields for example, OpenFlow Switching using up to 12-tuple packet header field. Packet classification is an essential function for Traffic management, access control, intrusion prevention, intrusion detection and many other network services. This architecture and complexity and power hunger is not necessary for intrusion detection systems with specific attacks that is the aim of this paper. In [26] an efficient GPU-based pattern matching is proposed that is not concluded in traditional intrusion detection systems to be compared with the proposed one.

Our proposed algorithm is based on the ideas of BitVector, hardware implementations of an efficient Packet Classification algorithm with an Incremental Update capability (PCIU) [19] and Extended TCAM (ETCAM) [4] methods. Our proposed algorithm has a delay of one clock cycle for matching all rules of filter set. Therefore, time complexity is $O(1)$ and the area complexity is related to number of rules that is $O(N)$. In the next section, intrusion detection system with the proposed classification algorithm is described in details.

3. HNPC ALGORITHM

Although today's Internet presents best-effort services, future networks will provide users more quality-aware services. These services may be such as Virtual Private Network (VPN), Distributed Firewalls, Security IP Gateways, Traffic based billing and etc. All these services need packet classification to determine which incoming packet belongs to which data flow by just checking the header fields. Packets are classified by matching the header fields: source and destination IP addresses, source and destination port numbers and also other protocol fields of each packet. Rules are set by determining thresholds for each header fields. Nominating the most fitting rule for each packet is the packet classification problem [2]. In intrusion detection systems by extracting data flow of packets, related decisions are made which can accept or drop the predetermined packet or send alerts or provide other related statistics for additional analysis to user interfaces. Shown in Figure 2, the network packet is saved in a First in First out (FIFO) manner and by considering the result of the decision unit, the final decision about the current packet is made. Five fields of the IP packet header fields in 104 bits are used in our proposed system in classification module.

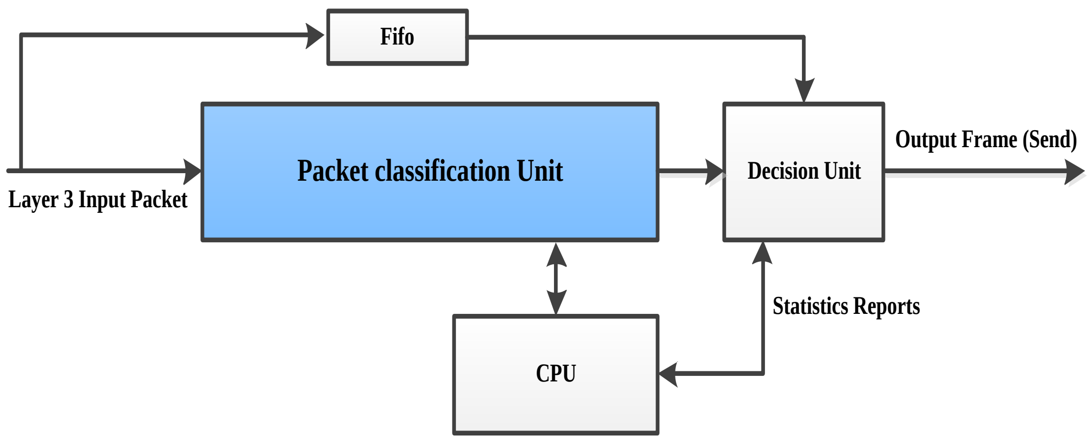


Fig. 2. General block diagram of typical intrusion detection system

In addition to dropping or passing the processed packets, more statistical details or alarms can be reported by CPU. Also CPU can help to identify some type of multistage attacks that need more historical information. The aim of this paper is to represent the structure, based on traffic volume analysis which can detect and prevent intrusion attacks. For feature extraction, saving and classification of packets in the memories are required. Conventional software methods are not suitable for high-speed networks. Consequently, we extract the appropriate features in a hardware approach. We used a 5-tuple (source and destination IP address, source and destination port number and protocol) as basic and explicit utilized features. The method used for rule set matching is derived from PCIU [19] and ETCAM [4] designs. We considered the high frequency rule matching requirement with more simple design simultaneously. The main modules of the proposed system are shown in Figure 3.

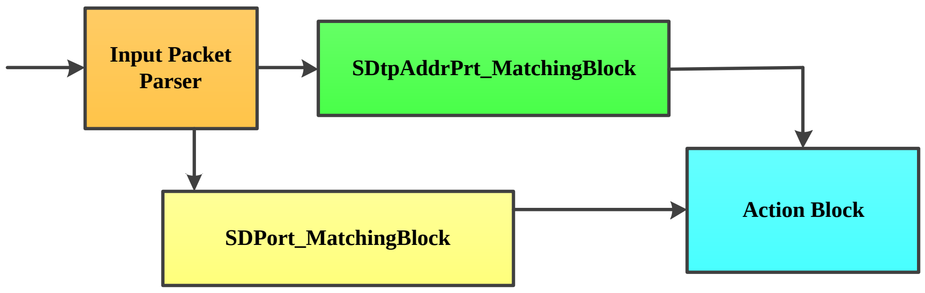
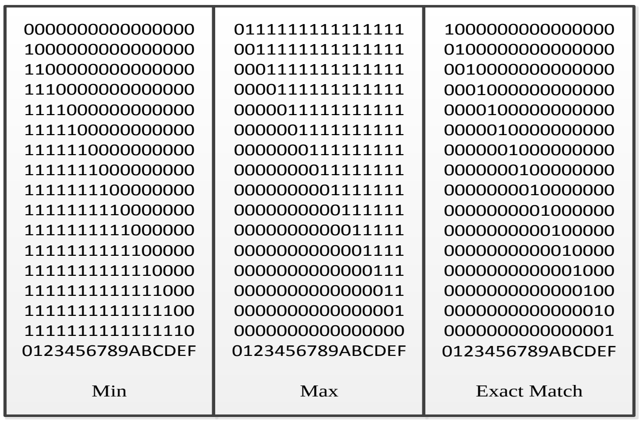


Fig. 3. Packet classification block

In this procedure, the extracted header information of the packet is sent to the IP address-protocol matching block and port number matching block simultaneously. Regarding the mask of IP Address-protocol and ranges of port numbers, two different methods are applied. Each input data is broken into 4 nibbles and are entered into the matching blocks. These blocks are initialized by an algorithm which will be introduced in the following section. All blocks work in parallel search for the input data types within the saved tables in a clock cycle and output vectors will be sent to an AND operator together to detect all the matching rules. Each bit in the bit vector represents a rule. When a rule matches an input packet its related bit will be set in the output bit vector. Finally the Decision module decides whether the predetermined packet should be sent or dropped and statistics of special rules are stored for advance analysis in CPU. The rule matching algorithms for IP address and port number are discussed in the next section in detail.

3.1 Proposed Port Matching Algorithm

Destination and source port numbers are depicted as 16-bit fields in the header of transport and the rules may contain minimum and maximum thresholds for port numbers that should be considered in rule matching algorithms. We break the port numbers into four nibbles. So each nibble that contains 4 bits can be between 0 to F. Since filtering rules can have exact value or a range of values for port numbers, we implement the exact match as a range with the same maximum and minimum. To fill out the tables, each nibble of rule is selected from one of the 3 tables in Figure 4. Related column is chosen from the table and is placed in the index of related rule in lookup table. This procedure is continued to fill out all the columns of the lookup tables. Note that lookup tables have 16 rows and the number of columns is equal to the number of rules.



Min	Max	Exact Match
0000000000000000	0111111111111111	1000000000000000
1000000000000000	0011111111111111	0100000000000000
1100000000000000	0001111111111111	0010000000000000
1110000000000000	0000111111111111	0001000000000000
1111000000000000	0000011111111111	0000100000000000
1111100000000000	0000001111111111	0000010000000000
1111110000000000	0000000111111111	0000001000000000
1111111000000000	0000000011111111	0000000100000000
1111111100000000	0000000001111111	0000000010000000
1111111110000000	0000000000111111	0000000001000000
1111111111000000	0000000000011111	0000000000100000
1111111111100000	0000000000001111	0000000000010000
1111111111110000	0000000000000111	0000000000001000
1111111111111000	0000000000000011	0000000000000100
1111111111111100	0000000000000001	0000000000000010
1111111111111110	0000000000000000	0000000000000001
0123456789ABCDEF	0123456789ABCDEF	0123456789ABCDEF

Fig. 4. Tables for generating Main Look Up tables

We used three kinds of tables in this paper. MIN and MAX tables are used for comparison to find the proper range and the exact match table is used for the border numbers. Therefore 32 tables are generated for detecting source and destination port number matching rules. As it can be seen in the proposed algorithm, these tables are used as lookup tables and should be filled before execution time, based on the defined patterns. Figure 5 shows an example usage of this table for finding appropriate range.

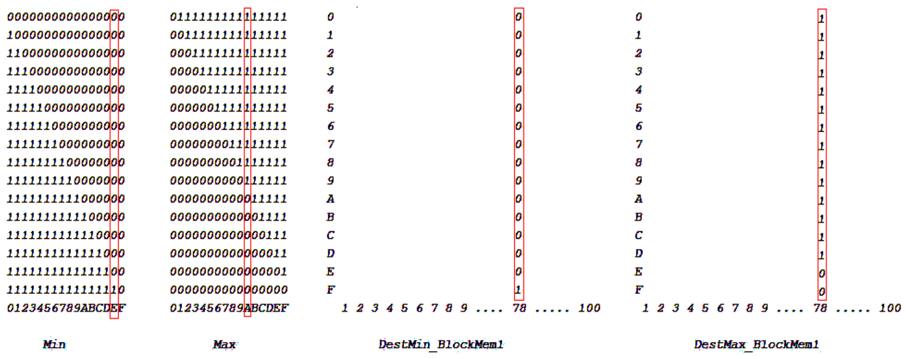


Fig. 5. Port Number Lookup Tables

After filling the lookup tables with appropriate patterns, the search process is done simultaneously in all four nibbles. The proposed algorithm is shown in Figure 6.

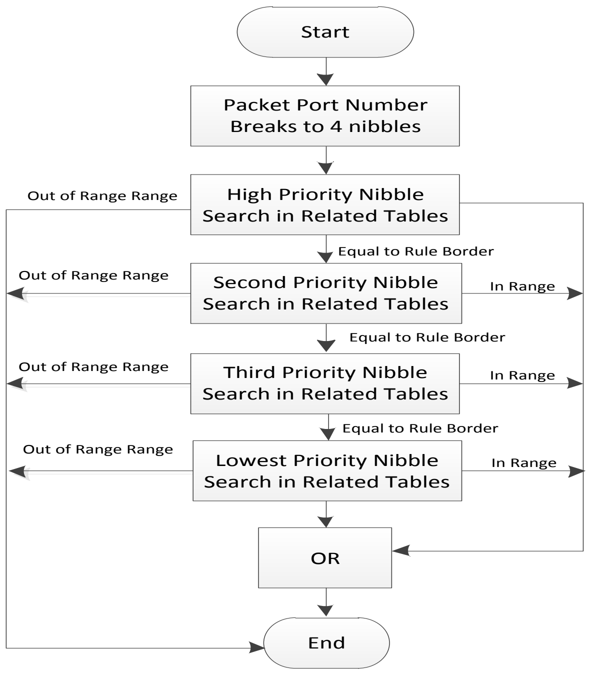

Start
Packet Port Number Breaks to 4 nibbles
High Priority Nibble Search in Related Tables
Out of Range Range
Equal to Rule Border
Second Priority Nibble Search in Related Tables
Out of Range Range
In Range
Equal to Rule Border
Third Priority Nibble Search in Related Tables
Out of Range Range
In Range
Equal to Rule Border
Lowest Priority Nibble Search in Related Tables
Out of Range Range
In Range
OR
End

Fig. 6. Proposed Port Number Search Algorithm

The following figure shows a sample execution of this algorithm.

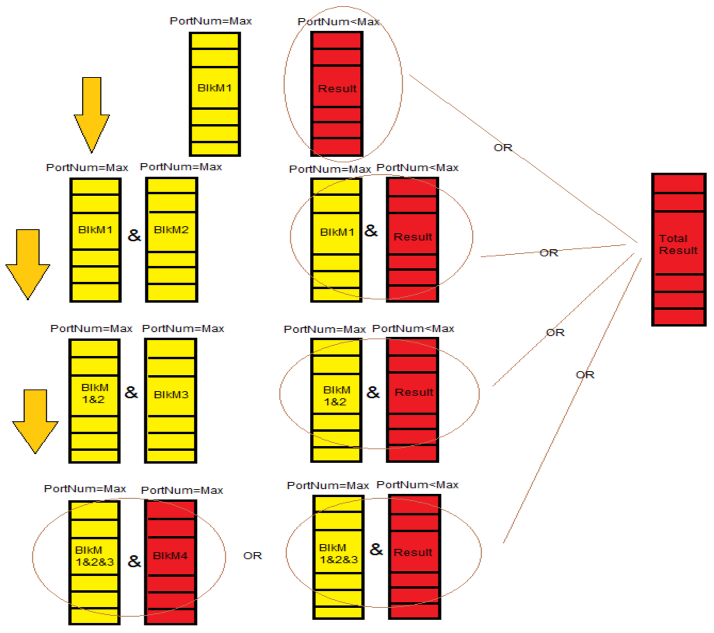
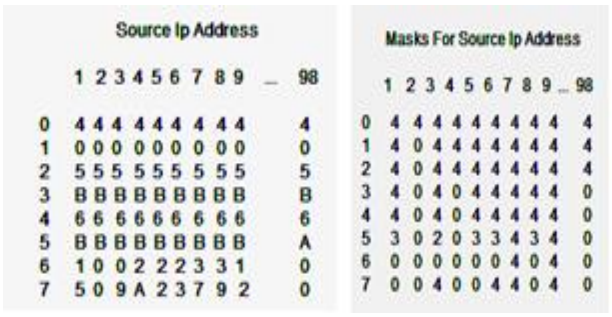


Fig. 7. Example Usage of Port Number Search Algorithm

3.2 Matching Algorithm for IP Address and Protocol

Proposed matching algorithm for IP address and protocol is implemented with two different tables. The IP addresses and masks are saved in these tables. Each nibble of source IP address fills out the columns of the table for the corresponding rule. In this figure, an instance for the 98 rules is illustrated.



Source Ip Address

	1	2	3	4	5	6	7	8	9	_	98
0	4	4	4	4	4	4	4	4	4		4
1	0	0	0	0	0	0	0	0	0		0
2	5	5	5	5	5	5	5	5	5		5
3	B	B	B	B	B	B	B	B	B		B
4	6	6	6	6	6	6	6	6	6		6
5	B	B	B	B	B	B	B	B	B		A
6	1	0	0	2	2	2	3	3	1		0
7	5	0	9	A	2	3	7	9	2		0

Masks For Source Ip Address

	1	2	3	4	5	6	7	8	9	_	98
0	4	4	4	4	4	4	4	4	4		4
1	4	0	4	4	4	4	4	4	4		4
2	4	0	4	4	4	4	4	4	4		4
3	4	0	4	0	4	4	4	4	4		0
4	4	0	4	0	4	4	4	4	4		0
5	3	0	2	0	3	3	4	3	4		0
6	0	0	0	0	0	0	4	0	4		0
7	0	0	4	0	0	4	4	0	4		0

Fig. 8. Tables of Source IP Address and Related Masks

In the mask table, each value can be from 0 to 4 corresponding to the IP address. A detailed description of the mask values is given in the following table.

Table 2. Used the Mask Values in IP Address Matching

Meaning Of Masks in Related Tables	Numbers in mask Tables
All 4 bits are don't care	0
One bit is important that is 8 state from 16 state and others are 0	1
Two bits is important that is 4 State from 16 state	2
Three Bits is important that is 2 State from 16 state	3
All 4 bits is important and there is an exact match	4

The procedure of preparing values for the lookup tables is done offline with some high level tools. We implemented a high level C# application for this purpose. After filling these tables, the search process has no complexity. Source and destination IP Addresses and also protocol header fields are extracted from each packet and passed to the related matching tables. These fields are searched in the generated look up tables simultaneously for each nibble and the result bit vectors from all tables are considered (OR operator is performed on the output bit vectors). In this phase we can apply the probable priority on the matched rules.

4. EXPERIMENTAL RESULTS

The most important part of any intrusion detection system is the packet classification module. In this section we try to setup a test bench for evaluating our proposed solution for packet classification. We implemented two different designs which call them primary and extended. The primary design contains 400 rules. All the rules are obtained from ACL,IPC and FW Filter Sets [12]. The deployed algorithm for IP address matching in this design is based on the TCAM. Our proposed algorithm for port matching that is explained in part 3.1 is applied from default range of numbers determined in rules. The implementation results of this design which is done on Xilinx ISE are shown in Table 3. Because of the problem of TCAMs that use lots of Block memories when the number of rules are increased and also to reach more clock frequency with maximum rules, we decided to omit TCAM. So in order to decrease the utilized area and increase the number of supported rules on the same device, we extended our proposed algorithm by using the same idea of port number matching for IP address and protocol instead of using TCAM. The results of this optimization are summarized in the Table 3. As it can be seen in the same device we can have more rules and also higher clock rate. When TCAM replaced with our algorithm we could omit all block memories and instead we modeled TCAMs by distributed memories that works like TCAMs but uses logics. So not only Number of LUTs and Slices decreased because of our optimization in area, but also the clock frequency increased. To understand how much rules could be matched with our design we tried to implement some more rules to fit our FPGA. Our Implemented device xc5vlx110t-3ff1136 (search in Virtex-5 Family Overview) has 17,280 slices and we use 99% of all slice of virtex5 with our method. This achievement is valuable hence despite of the high frequency rate and least clock cycles, most slices of the FPGA are occupied without any problem in place and rout or problems of Lack of resources.

For validating our implementation and ensuring the true functionality of the proposed algorithm, we used some standard benchmarks which their detail can be found on [13] [12]. These benchmarks contain some real (and also synthetic) filter sets with different number of rules that present some labeled network packets which can be used for evaluation of any intrusion detection system. We used this benchmark for validating and evaluating our proposed solution. All of our simulations are done after successful completion of "*place and route*" phase of implementation by Modelsim.

Table 3. Synthesis results of the two different implemented designs (All results are summarized after successful place and route on the Xilinx-Virtex5xc5vlx110t-3ff1136)

Implementation Results	*Used filter set*	*Number Of Rules*	*Memories (KB)*	*LUT*	*SLICE*	*Frequency (MHZ)*
Primary Design	Filter Sets ACL1_100	100	864	4,165 (6%)	1,223 (7%)	111.68
Primary Design	Filter Sets ACL1+FW1+IPC1	400	0	15,971	4,262	117.64
Extended Design	Filter Sets ACL1_100	100	0	2,143 (3%)	862 (4%)	197.16
Extended Design	FilterSetsACL1+FW1+IPC1	500	0	2,600	1,253	145.01
Extended Design	ACL1_1K	1K	0	5,431	2,256	115.47
Extended Design	ACL_5K	5K	0	26,692	10,654	100.34
Extended Design	ACL_10K	10K	0	53,089	17,233	100.24

The Implementation results of the proposed architecture and algorithms that are summarized in the Table above show that:

In first row of above table we have primary design with TCAM structure for Ip address matching and proposed algorithm for port matching with 100 rules that are taken from [12]. In this case 6% of LUTs and 7% of slices have been used. We tried to use maximum slices possible for our classification block.

So our primary design lead us to match 400 rule with 15,971 LUT that is 92% of our FPGA device.

As explained before we extended our work with using the same idea for matching Ip address as proposed idea for port matching algorithm. With the same rule number shown in table3 (100 rule) our resource utilization decreased to 3% of LUTs and 4% of slices. Use of block memory is decreased to zero because we used distributed memories instead of TCAM. We tried to increase number of rules to 10000 and 99% device utilization. The proposed extended design works with 10K rules in 100 MHz (post place and route simulation confirms this claim). With respect to the parallel nature of the proposed algorithm, the decision making process of the implemented system needs only one clock cycle that leads to have a throughput more than 10 Gbps (just by considering minimum Ethernet packet size).

5. EVALUATION AND COMPARISON OF THE PROPOSED SYSTEM

In section 2, short introduction of some related works are presented based on the previous categorization of classification systems which is presented on [19]. As mentioned in previous sections, our proposed algorithm is presented based on the benefits of some well-known systems. The proposed system merges the benefits of decomposition and tuple space based techniques by decomposing the search space into different parts and searches all of them simultaneously and it also has all functionality of exhaustive search-based techniques by deploying the TCAM and ETCAM in its proposed algorithm (Figure 9). More specifically we used the benefits of the TCAM, ETCAM, PCIU, DCFL and Parallel BV methods in our proposed system.

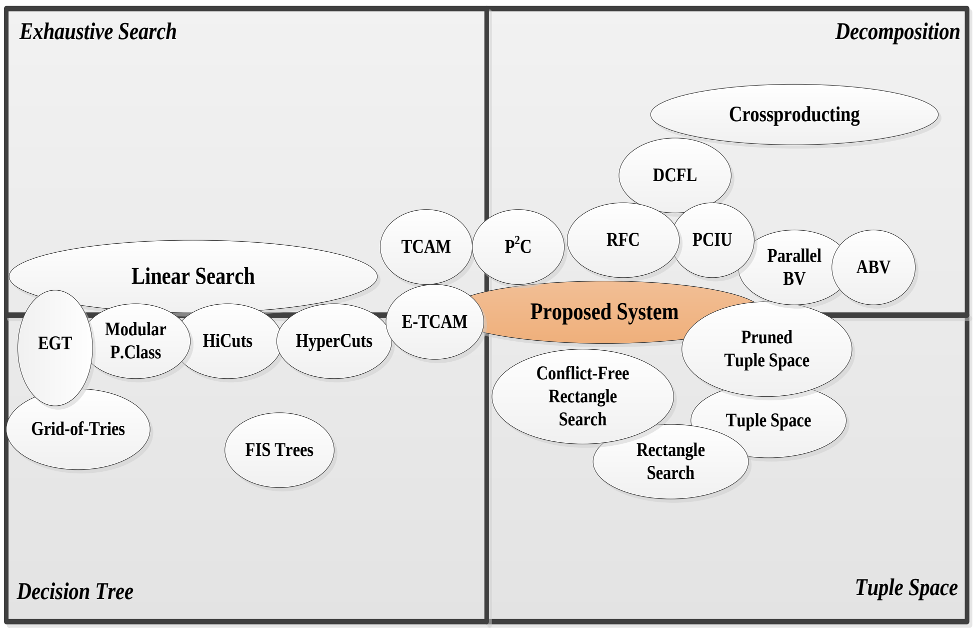


Fig. 9. Status of proposed system in comparison with other related works

We try to compare the HNPC with other proposed systems by some other design metrics in the following table. We used the clock cycle duration, frequency, link rate and throughput as our basic comparison metrics. Since we do not have the HDL codes of the previous proposed works to synthesize them on the same device, we referred to their documented synthesis results of their papers and summarized them in this table. This table shows that the proposed design has an acceptable value in all design metrics in comparison with other methods.

Table 4. Proposed algorithm between Other Methods

Year	*Device*	*Block Ram*	*Throughput [1]Mpps*	*Rule Number*	*Link Rate Gbps*	*Frequency (MHZ)*	*Clk Cycle*	*Method*
2011	Spartan3E XCS3S500E	360 KB +32MB of external DDR + 16MB of external Flash memory	5.5, 6.4, 6.5	1K, 5K, 10K	6.76	~143	9 Cycle	PCIU [19]
2008	Virtex5 SX95T	134 (54%)	7.6	1K, 1.5K, 2K	7	77	2 cycle, 4 cycle	Hicuts, [29], Hypercuts
2010	Virtex-II ProXC2VP30	49 (36%)	50	128	16	50	1 Cycle	DCFLE [4]

2005	Virtex^{TM}-EXCV2000E	128KB	1, 7.5	512	2.4, 10	33, 100	13 Cycle	BV+ TCAM [6]
2012	Virtex5_XC5LX110t-3ff1136	0	117	400	12.23	117.647	1 Cycle	Primary Design
2012	Virtex5_XC5LX110t-3ff1136	0	100	10K, 5K, 1000, 500, 100	32	100.241	1 Cycle	Extended Design

The detail comparison of our proposed algorithm with other related systems are shown in the following figures regarding to the design metrics which are introduced in preceding sections.

[1]Million Packet Per Second

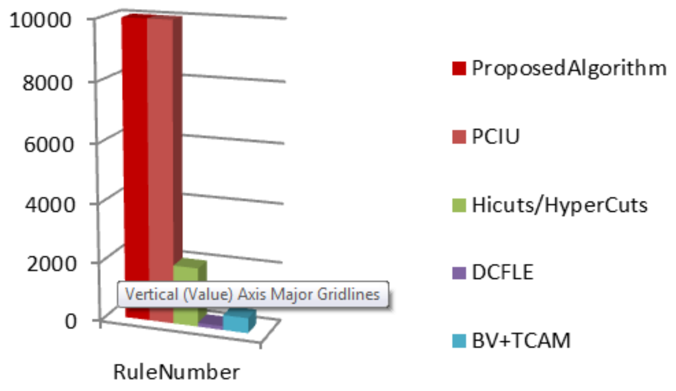


Fig. 10. Number of supported rules

The number of supported rules has direct impact on the performance of the classification engine. Figure 10 show that only PCIU can have as many as rules as our proposed system.

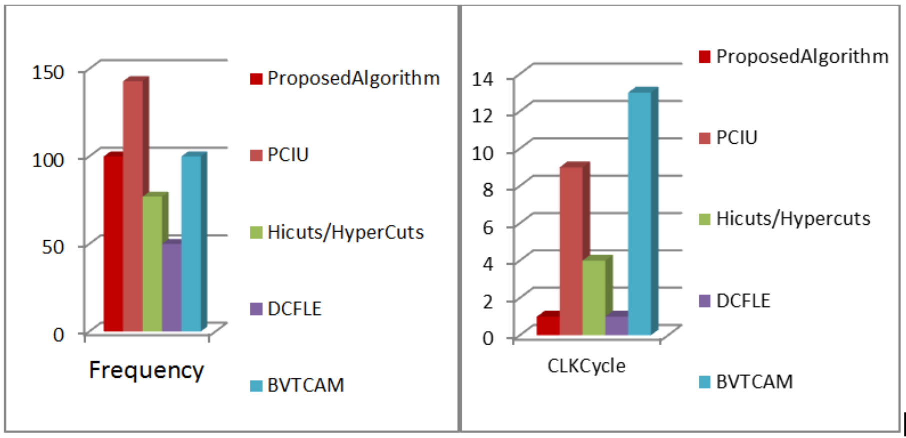


Fig. 11. Timing comparison of proposed system with other related works

Figure 11 shows the timing comparison between the related works. As it can be seen, the proposed algorithm has higher frequency than other systems (except PCIU) and also its output will be ready in one clock which makes the throughput more than other proposed approaches (Figure 12).

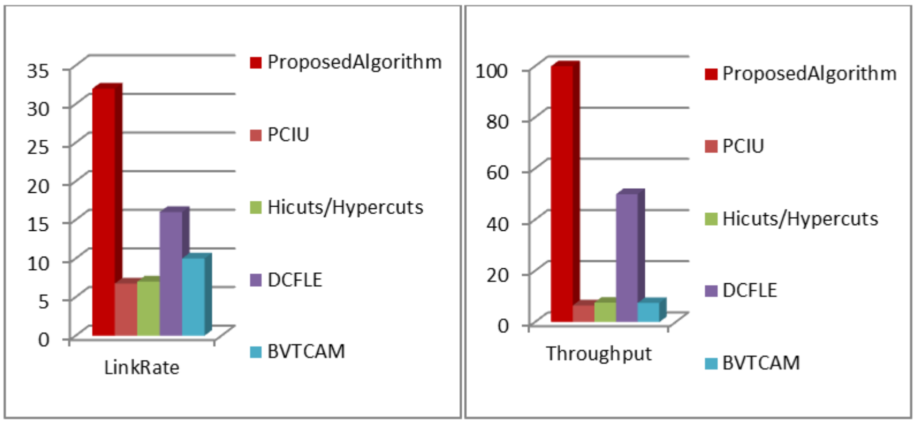


Fig. 12. Comparison of related systems throughput

6. CONCLUSION AND FUTURE WORKS

Packets are classified by matching some header fields like source and destination IP addresses, source and destination port numbers and protocol type. In this paper we utilized these header fields as features for performing packet classification tasks but we didn't take into consideration the flow of packets. We plan to use the Microblaze as a high level processor of our system to identify some advanced malicious activities. We also plan to add other outputs like sending alerts and collecting statistics in addition to accept or drop the current packet to our proposed system. Our two different proposed implementations can be worked with 100 MHz and 10,000 rules which makes the throughput more than 10 Gbps. The proposed algorithm has been simulated (post place and route) in Modelsim with some standard benchmarks. To extend the current design, we want to use the capability of runtime configurability of FPGA devices to add new rules during its execution. Also we think that using compression techniques like hash functions will be useful for reducing the number of block rams that may lead to better performance and reduction of the resource utilization.

REFERENCES

[1]. Kim,B; Jang, J; Sohn, S.w; Chung T.M, "*High-Performance Intrusion Detection in FPGA-based Reconfiguring Hardware*", Security Gateway System Team Electronics and Telecommunications Research Institute, 2005.

[2]. Feldman, A.; Muthukrishnan, S., "*Tradeoffs for packet classification*", IEEE INFOCOM, 2000.

[3]. Taylor D., "*Survey and Taxonomy of Packet Classification Techniques*", ACM Computing Surveys, vol. 37, 2005.

[4]. Jedhe G.S., Ramamoorthy A., Varghese K., "*A Scalable High Throughput Firewall in FPGA*", 16th International Symposium on Field-Programmable Custom Computing Machines, 2008.

[5]. Taylor D. E., Turner J. S., "*Scalable Packet Classification using Distributed Crossproducting of Field Labels*", IEEE INFOCOM, 2005.

[6]. Song H., Lockwood J. W., "*Efficient Packet Classification for Network Intrusion Detection using FPGA*", IEEE FPGA, 2005.

[7]. Spitznagel E., Taylor D., Turner J., "*Packet Classification Using Extended TCAMs*", IEEE ICNP, 2003.

[8]. Kannaiya Raja N., Arulanandam K., Umadevi P., David Raj G., "*Routers Packet Classification using Configuration Machine*", International Journal of Computer Science and Technology, 2012.

[9]. Taylor D.E., Lockwood J.W, Sproull T., Turner J.S., Parlour D. B., "*Scalable IP Lookup for Programmable Routers*", IEEE INFOCOM 2002.

[10]. Gupta P., McKeown N., "*Packet classification using hierarchical intelligent cuttings*" IEEE Micro, 2000.

[11]. Singh S., Baboescu F., Varghese G., Wang J., "*Packet Classification Using Multidimensional Cutting*", ACM SIGCOMM, 2003.

[12]. http://www.arl.wustl.edu/~hs1/PClassEval. html#3._Filter_Sets, Last Visited: 15-jan-2013.

[13]. Taylor D. E., Turner J. S., "*Classbench: a packet classification benchmark*", INFOCOM, 2005.

[14]. Yoshioka A., Shaikot S.H., Kim M. S, "*Rule Hashing for Efficient Packet Classification in Network Intrusion Detection*", IEEE ICCCN, 2008.

[15]. Ganegedara T., Prasanna V., Brebner G., "*optimizing packet lookup in time and space on FPGA*", International Conference on Field Programmable Logic and Applications (FPL), 2012.

[16]. Ahmed O., Areibi S., Grewal G., "*Hardware accelerators targeting a novel group based packet classification algorithm*", International Journal of Reconfigurable Computing, 2013.

[17]. Pati S., Narayanan R., Memik G., Choudhary A., Zambreno J., "*Design and Implementation of an FPGA Architecture for High-Speed Network Feature Extraction*", IEEE ICFPT, 2007.

[18]. ,Song H., Turner J.S" *ABC: Adaptive Binary Cuttings for Multidimensional packet classification*", ACM Transactions on Networking, 2012.

[19]. Ahmed O., Areibi S., Chattha K., Kelly B., "*PCIU: Hardware Implementations of an Efficient Packet Classification Algorithm with an Incremental Update Capability*", International Journal of Reconfigurable Computing, 2011.

[20]. Gupta P., McKeown N., "*Packet classification using hierarchical intelligent cuttings*" IEEE Micro, 2000.

[21]. Singh S., Baboescu F., Varghese G., Wang J., "*Packet Classification Using Multidimensional Cutting*", ACM SIGCOMM, 2003.

[22]. Gupta, P., McKewon, N., "*Algorithms for Packet Classification*", IEEE Network, 2001.

[23]. Netfilter: firewalling, NAT and packet managing for Linux. http://www.netfilter.org/.

[24]. PF: The OpenBSD Packet Filter. http://www.openbsd.org/faq/pf/.

[25]. Jiang W., Prasanna Viktor K., " *Scalable Packet Classification on FPGA*", IEEE Transactions On Very Large Scale Integration (VLSI) systems, VOL. 20, NO. 9, 2012.

[26]. Hung C.L, Chang C.Y, Lin C.Y, "*Efficient Packet Pattern Matching for Gigabit Network Intrusion Detection usingGPUs* ", International Conference on High Performance Computing and Communications, 2012.

[27]. Lee T., Yusuf S., Luk W., Sloman M., Lupu E., Dulay N., "*Irregular Reconfiguration CAM Structures for Firewall Application*", IEEE FPL, 2003.

[28]. Taylor D. E., Turner J. S., "*Scalable Packet Classification using Distributed Crossproducting of Field Labels*", IEEE INFOCOM, 2005.

[29]. Kennedy .A, Wang .X, Liu .B, "Energy Efficient Packet Classification Hardware Accelerator", Proc. IEEE Int’l Symp. Parallel and Distributed Processing (IPDPS ’08), pp. 1-8, Apr. 2008.