Routers Packet Classification Using Configuration Machine

Dr. V. Francis Densil Raj
Assistant Professor - MCA,
Anna University of Technology Madurai,
Tamilnadu, India

Mrs. C. M. Selvarani
Professor - CSE
Pannai Engineering College
Tamilnadu, India

ABSTRACT:

The packet classification is the vital part mechanism that leads to create many networking services in the internet such as firewall packet filtering and traffic accounting. So the packet classification is in need which enables many networking services in the network. There are two types support such as hardware and software support. The hardware support is more expansive compared to software and it also has limited scalability. In this paper we discussed the extreme packet classification by decreasing the optimization rule in turn which improves the incremental rules update efficiently. Our work also improves the source and destination ip prefixes specified in rules which are based on Router Configuration Engine Packet Classification (**RCEPC**), inspects multidimensional sequence based tree algorithm. The firewall rules are built in the form of 24 bit sub rules which reads the router instruction and improved classification process, which in turn finds the classification rule in the network.

Keywords: Packet classification, Packet Filtering, RCEPC, Tree based algorithm, Firewall rules

INTRODUCTION

Packet classification implementation is done at routers by applying some rules for incoming packets and categories for flow of packets. At the arrival packet contains multiple fields in the header. For this search key can be used to identify the best suitable rule to apply. Each header fields contains values. These values are used to create a rules for differentiate packets constituting a filter set. A field value in filter can be an IP prefix, a range or an exact number. Multiple rules can be obtained for real filter data set in order to communicate the pair network one for each application. In multiple filters is the application appears as one for each pair of communications network using the application. Lookups over a filter set with respect to multiple header fields are complex [4] and can easily become router performance bottlenecks.

The different classification mechanism are used to improve the efficiency of the quicken packet classification through the hardware and software approaches. The software approach uses of specific data structures to hold filter data sets for fast search [8]. The software oriented classification is more attractive, less expensive and more flexible. That is provided lookup speed can be quickened by sorting rules in on chip SRAM. The hardware approach is basically implemented by using field programmable gate arrays (FGPA) or ASIC logics [2] [6] plus ternary content addressable memory (TCAM) to hold filters [9] or registers for rule caching[3]. The classification mechanism is based on software and hardware. Unfortunately an approach with hardware support is expansive and has limited scalability, where as one with optimization fails to incremental rule updates effectively and TCAM give to hold a filter set would dictate the maximal set size allowable. The Recursive Flow Classification [4] or inserted tuple space search rules (rectangle TSS [7],binary TSS [10] diagonal TSS [5] are used to increasing optimization via pre-processing speedup lookups often fail to deal with incremental rule updates effectively.

For rapid packet classification we use router configuration engine packet classification (RCEPC), here an IP prefix with £ bits is rounded down to include its first £ bits only for £≤ ι , £ є DPL, “designated prefix length”. Router Configuration engine Packet Classification (RCEPC) acquires high classification throughput and superior memory efficiency means of one with two staged search. In this method it round down the prefixes to a small number of DPL denoted by m, i.e., m is DPL. Here each DPL corresponds to one unit.

For fewer than 32 under IPV4, when every prefix length is permitted without rounding down. For set association we take the hash accesses per packet classification and collapse those hash units to one lumped hash table (LuHa) for better performance. Router Configuration Engine Packet Classification (RCEPC) has a fast classification than Lumped Hash (LuHa) table keyed by the source and destination IP. Since it has small number of hash access to router.

Manuscript received December 5, 2011
Manuscript revised December 20, 2011

Table1. An example of rules configuration in classifier

Destination IP (addr/mask)	Source IP (addr/mask)	Destination port	Protocol
152.163.190.69/32	152.163.80.11/24	*	*
152.168.3.0/24	152.163.200.157/32	80	UDP
152.168.3.0/24	152.163.200.157/32	20,21	UDP
152.168.3.0/24	152.163.200.157/32	80	TCP
152.163.198.4/32	152.163.160.0/22	>1023	TCP
152.163.198.4/32	152.163.36.0/24	>1023	TCP

The Lumped Hash Table (LuHa) table gives high storage utilization by identifying multiple candidate sets for each rule. The LuHa table has three major differences infrastructure. As a fundamental element, packet classification is the most important process in TCS, which categories packets into different traffic flows using a set of filters or rules. With rapid increasing Internet services, traffic control systems need more accurate packet classification algorithms and high performance processing.

Most previous packet classifications support the prefix match, exact match and range match. In IP network, for example, many applications use five classic fields (protocol, source IP, source port, destination IP and destination port) of packet header to determine the flow which packet belongs to. Table 1 shows a rule configuration from ISPs and enterprise networks [11].compared from other d-left hashing [1]. They are: First the LuHa table needs only one hash function whereas d function needs d-left hashing. Second the hash function is given by 2m for different prefixes for each pair of the source and destination IP address. Third it gives higher storage usage by combining different LuHa table was designed by Traffic Control System (TCS).

Traffic control system (TCS) is becoming an indispensable and widespread-deployed device of network Table 2 shows an example of traffic distribution with inaccurate classification. We collected applications traffic proportion from 31 different network nodes. A large amount of applications transmit packets through not only the traditional transport layer port but random ports. Therefore, approximate 30% traffic cannot be categorized into a flow which complies with a rule. However, new network applications and services demand efficient and accurate packet classification. Thus, it is necessary that traditional packet classifications evolve to a scalable traffic classification in IP network. Especially, traffic control systems need classifying, shaping network traffic and detecting malicious packet.

An example, we can change HTTP port from80 to a random port. Furthermore, port 80 is being used by a variety of non-web applications to circumvent TCS and firewalls which do not filter 80 port traffic and some new applications hardly use fixed transport layer port to communicate, such as P2P flows, multimedia stream. So, accurate and efficient packet classification is an urgent problem in TCS.

Table2. An example of traffic distribution

Protocol And Application	Percentage Of Traffic
HTTP	10%
P2P	15.7%
FTP	1.9%
Others	39.1%
Unclassified	33.3%
Total	100%

To obtain high performance and flexibility, we use a promising hardware solution Intel IXP2800 network processor. The IXP2800 is the second-generation network processor, which enables fast deployment of intelligent network services by providing flexible programming and high performance. It is suitable for complex packet processing in a wide variety of network applications. Each hardware thread in network processor can run independently and parallel to process packets. The IXP2800 supports a broad range of speed from OC48 to OC192.

This paper describes an efficient hybrid packet classification in gigabits traffic control system using second-generation programmable network processor. We address the problem of inaccurate packet classification and analyze the payload of new applications. In particular, we focus on the packet classification on not only packet header but the first 64-bit payload and other flags of packet router property. Then, we present the software pipeline architecture, hardware implementation

consideration, and report measurements. We show a classifier that can exploit network processors, and make suggestions about hardware features that can significantly improve accuracy and performance of traffic control system.

Related Work

Classification lookup mechanisms may be categorized, in accordance with their implementation approaches, as being hardware-centric and software-oriented, depending upon if dedicated hardware logics or specific storage components (like TCAM or registers) are used. Different hardware centric classification mechanisms exist. In particular, a mechanism with additional registers to cache evolving rules and dedicated logics to match incoming packets with the cached rules was pursued [3]. Meanwhile, packet classification using FPGA was considered [6] by using the BV (Bit Vector) algorithm [17] to look up the source and destination ports and employing a TCAM to hold other header fields, with search functionality realized by FPGA logic gates. Recently, packet classification hardware accelerator design based on the HiCuts and HyperCuts (HC) algorithms [13], [20] has been presented [16]. Separately effective methods for dynamic pattern search were introduced [2], realized by reusing redundant logics for optimization and by fitting the whole filter device in a single Xilinx FPGA unit, taking advantage of built-in memory and XOR-based comparators in FPGA.

Hardware approaches based on TCAM are considered attractive due to the ability for TCAM to hold the Don't care state and to search the header fields of an incoming packet against all TCAM entries in a rule set simultaneously [18], [9]. Widely employed storage components in support of fast lookups, TCAM has such noticeable shortcomings (listed in [8]) as lower density, higher power consumption, and being pricier and unsuitable for dynamic rules, since incremental updates usually require many TCAM entries to be shifted (unless provision like those given earlier [19], [9] is made). As a result, software-oriented classification is more attractive, provided that its lookup speed can be quickened by storing rules in on-chip SRAM.

Software-Oriented Classification

Software-oriented mechanisms are less expensive and more flexible in filter lookups when compared with their hardware-centric counterparts.

Such mechanisms are abundant, commonly involving efficient algorithms for quick packet classification with an aid of caching or hashing. Their classification speeds rely on efficiency in search over the rule set using the keys constituted by corresponding header fields. Several representative software classification techniques are reviewed in sequence.

Recursive Flow Classification

Recursive flow classification (RFC) carries out multistage reduction from a lookup key (composed of packet header fields) to a final class ID, which specifies the classification rule to apply [4]. Given a rule set, pre-processing is required to decide memory contents so that the sequence of RFC lookups according to a lookup key yields the appropriate class ID [4]. Based on a pre-computed decision tree, Hierarchical Intelligent Cuts (Hi Cuts) [15] holds classification rules merely in leaf nodes and each classification operation needs to traverse the tree to a leaf node, where multiple rules are stored and searched sequentially. During tree search, Hi Cuts relies on local optimization decisions at each node to choose the next field to test. Hyper Cuts (HC) is an improvement over Hi Cuts by allowing cuts to be made over multiple dimensions at a node [20], as opposed to just a single dimension each in Hi Cuts. At every node, there can be totally $\pi^{D}_{i=1}$ nc(i)child nodes, where nc(i) is the number of splits made in the ith dimension. Like Hi Cuts, HC is also a decision tree-based classification mechanism, but each of its tree nodes splits associated rules possibly based on multiple fields. It is shown to enjoy substantial memory reduction while considerably quickening the worst-case search time under core router rule sets [20], when compared with Hi Cuts and other earlier classification solutions.

Decision Tree Based Method

Given that decision tree-based methods (like HiCuts and HC) in general are notorious for the tree size explosion problem (with the decision tree size highly depending on the data sets and possibly to grow exponentially), refinement techniques are introduced to reduce their storage requirements. The trade-off between storage and lookup performance

can be measured by the space factor (SF), with a larger SF value yielding a wider and shallower decision tree. A larger SF is expected to consume more storage but support faster lookups.

Meanwhile, storage-saving for decision tree-based classifiers can be achieved by pushing common rules upwards, aiming to keep a common set of rules at the parent node if the rules hold true for all of its child nodes. Although this way lets rules be associated with non leaf nodes to save storage by avoiding replicas at the leaves, it can degrade lookup performance, as a lookup then has to examine internal nodes. Additionally, since decision trees are known to involve excessive (child node) pointers, a common fix lets the parent node keep merely the starting base address pointing to its first child

plus an additional n-bit "Extended Path Bitmap" (EPB) to remember existing child nodes [20]. This pointer compression reduces space greatly from n pointers to one plus (n -1) bits. Also, practical decision tree implementation stores all filter rules in a memory array of consecutive locations; the decision tree only keeps indices in the tree nodes. This reduces storage by avoiding replicas of filter rules which hold true for multiple sub trees due to wildcard addresses or port ranges. Adversely, such compression techniques make incremental updates very difficult. The linear memory array representation leaves holes upon rule deletions and is hard to accommodate new rules. The resulting indirect lookup process becomes inefficient because memory, in principle, exhibits maximal bandwidth under continuous bursts of requests. When data objects are accessed in a random (or non burst) manner, memory bandwidth efficiency dwindles rapidly, so is the packet classification rate.

An efficient router packet classification algorithm was introduced [12] by hashing flow IDs held in digest caches (instead of the whole classification key comprising multiple header fields) for reduced memory requirements at the expense of a small amount of packet misclassification. Recently, fast and memory-efficient (2D) packet classification using Bloom filters was studied [14] by dividing a rule set into multiple subsets before building a cross-product table for each subset individually. Each classification search probes only those subsets that contain matching rules (and skips the rest) by means of Bloom filters, for sustained high throughput. The mean memory requirement is claimed to be some 32-45 bytes per rule [14]. As will be demonstrated later, our mechanism achieves faster lookups (involving 8- 16 hash probes plus four more SRAM accesses, which may all take place in parallel, per packet) and consumes fewer bytes per rule (taking 15-25 bytes per rule).

A dynamic packet filter, dubbed Swift [21], comprises a fixed set of instructions executed by an in-kernel interpreter. Unlike packet classifiers, it optimizes filtering performance by means of powerful instructions and a simplified computational model, involving a kernel implementation. Lately, HyperSplit has been pursued for its superior classification speed and memory usage [22]. Based on recursive space decomposition, HyperSplit has the tree size explosion problem and requires 66 MB storage for 10K ACL rules, in sharp contrast to less than 250 KB under router Configuration Engine Packet Classification (RCEPC).

The point of implementation, router packet classification includes two kinds: hardware-based algorithms and software-based algorithms. Typical algorithms show in Table 3. (n denotes the number of rules, d denotes the number of classification fields, w denotes width of classification fields).

A. Novel algorithms

Router Packet Classification algorithm (RPC) splits the set of filter rules into several subsets by the hash-compression index table built based on the first 8-bit prefix of IP and constructs fast search trees for each subset. These search trees with smaller-sized filters can be more quickly constructed, optimized and updated, so the rate of search is correspondingly improved. It can be easily implemented in hardware at line speeds using a pipeline fashion.

From the view of geometry, the best bounds for point location in N rectangular regions and $d > 3$ dimensions are O (log n) time with O (Nd) space; or O ((log n)d-1) time with O (N) space.

A kind of Parallel Router Packet Classification algorithm [24] adopts a parallel decision tree based packet classification scheme via rule set pre-partitioning. The original rule set is pre-partitioned into a series of subsets and then the overlaps among the rules are eliminated on several packet fields. So that decision cuts within each subset (on such packet fields) will not incur rule duplication and the storage efficiency is, therefore, significantly increased as well as $O(\text{Log}(N))$ processing delays.

A new algorithm [25] for router packet classification uses the concept of independent sets. The algorithm has very small memory requirements. The search speed is not sensitive to the size of the rule table or to the percentage of wildcards in the fields. It also scales well from two-dimensional classifiers to high-dimensional ones. In particular, the algorithm is inherently parallel. Hardware tailored to this algorithm can achieve very fast search speed. The update algorithm proposed is also very fast in general.

Type	Algorithm type	Typical algorithm	Time complex-ity	Space complex-ity	Update complex-ity
Software	Basic data structure	Linear search algorithm	n	n	n
		Hierarchical tries algorithm	w^d	ndw	d^2w
	Geometric algorithm	Grid-of-tree algorithm	w^{d-1}	ndw	high
		AQT algorithm	w	nw	-
	Heuristics algorithm	RFC algorithm	d	n^d	high
		Hierarchical cuttings algorithm	d	n^d	low
		Tuple space search algorithm	n	n	High
Hardware	TCAM algorithm		1	n	-
	Bitmap-intersection algorithm		n	dn^2	-

A Novel Router Packet Classification Algorithm [26] makes use of bits distribution features. It is a high level classification method. It always takes the bits from every dimension into account, instead of constraining the search process in some of the dimensions at every stage. It is composed of three lookup procedures: a heuristic hash lookup, a novel multi-way dynamic search tree lookup and a lookup in a small rule set. This architecture is aimed at finding the best matching rule by traversing an optimal search path.

The performance analysis method of algorithms

The actual performance analysis needs many rules, data packets and the relation between them for whichever algorithms. It is a difficult task to simulate the data sets. Class Bench [27], a suite of tools for benchmarking packet classification algorithms and devices, has been widely applied. It is based on a battery of analyses on 12 real filter sets provided by Internet Service Providers (ISPs), a network equipment 563 vendor, and other researchers working in the field. The filter sets range in size from 68 to 4557 entries and utilize one of the following formats: access control list (ACL), firewall (FW), and IP chain (IPC). The general approach of ClassBench is to construct a set of benchmark parameter files that specify the relevant characteristics of real filter sets, generate a synthetic filter set from a chosen parameter file and a small set of high-level inputs, and generate a sequence of packet headers to probe the synthetic filter set using the Trace Generator. Markers-based Space Decomposition Algorithm [28] and O(log W) Multidimensional Packet Classification used this benchmark tools. Taking the level of network protocol and fields extension into account, Class Bench should include MAC and IPv6 headers in the future.

Construction

Router Packet Classification Algorithm Design in Cable Modem Quality of Service System

A. Router Packet Classification application in Quality of Service system

This paper is based on Cable Modem (CM) research with Godson CPU, SDRAM memory chip, Embedded Linux OS and DOCSIS (Data-Over-Cable Service Interface Specification) to allow transparent bidirectional transfer of Internet Protocol (IP) traffic, between the cable system head end and customer locations in hybrid-fiber/coax cable network. CM and CMTS (Cable Modem Termination System) classify packets traversing the RF MAC interface into a Service Flow to provide Quality of Service (QoS) by shaping, policing, and prioritizing traffic according to the Parameter Set defined for the Service Flow. Service Flow and Classifier are two important concepts in DOCSIS V1.1. A Service Flow is a MAC-layer transport service that provides unidirectional transport of packets either to upstream packets transmitted by the CM or to downstream packets transmitted by the CMTS. A Service Flow is characterized by a set of QoS Parameters such as latency, jitter, and throughput assurances. A Classifier is a set of matching criteria applied to each packet entering the cable network. It consists of some packet matching criteria (destination IP address, for example), a classifier priority, and a reference to a service flow. If a packet matches the specified packet matching criteria, it is then delivered on the referenced service flow. Figure 1 illustrates the mappings discussed below.

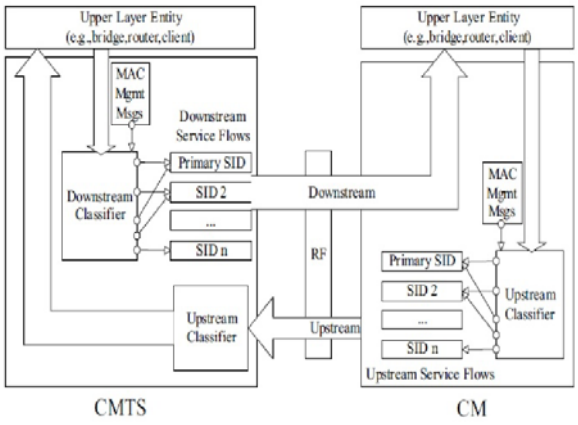

Figure 1. Classification within the MAC Layer

CM and CMTS Packet Classification consist of multiple Classifiers. Several Classifiers may all refer to the same Service Flow. The mapping of classifier and service flow shows in Figure 2. If a Classifier is found in which all parameters match the packet, the Classifier MUST forward the packet to the corresponding Service Flow. If no Classifier is found in which all parameters match the packet then the packet is classified to the Primary Service Flow.

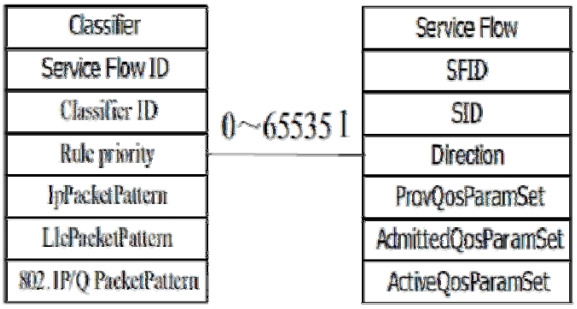

Figure 2. Mapping of classifier and service flow

The packet classification table contains the following fields in DOSCSISV1.1, in Table 4. But DOCSISV3.0 also includes Ipv6 Traffic Class Range and Mask, Flow Label and Next Header Type.

Classifiers can be added to the table either via management operations (configuration file, registration) or via dynamic operations

(dynamic signaling, DOCSIS MAC sub layer service interface). SNMP based operations can view Classifiers that are added via dynamic operations, but cannot modify or delete Classifiers that are created by dynamic operations.

In real network, rules mainly focus on IP Classification Parameters. LLC and 802.1p/q Classification Parameters are seldom.

The BH algorithm is proposed to meet QoS requirement in Cable Modem and limit resource in embedded system. The algorithm is based on B tree structure and non-collision function. The algorithm divides di into two stages.

Fields	Description
Priority	Determines the search order for the table. High priority classifier are searched before lower priority classifier.
IP classification parameter	Zero or more of the IP classification parameters(IP TOS Range/mask, IP destination Address/mask, TCP/UDP Source Port Start, TCP/UDP Source Port End, TCP/UDP Destination Port Start, TCP/UCP Destination Port End)
LLC Classification Parameters	zero or more of the LLC classification parameters (Destination MAC Address, Source MAC Address, Ethertype/SAP)
IEEE 802.1P/Q Parameters	zero or more of the IEEE classification parameters (802.1P Priority Range, 802.1Q VLAN ID)
Service Flow Identifier	identifier of a specific flow to which this packet is to be directed

The first stage reduces the memory by storing the redundant once and uses a non collision function to speed up search time for IP Classification.
Then the number of rules sharply decreases. The match rule is finally obtained by linear search with priority for LLC and 802.1p/q Classification. The algorithm process shows in Figure 3.

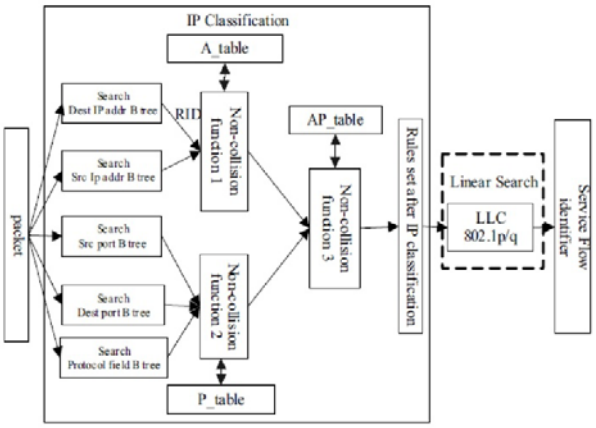

Figure 3. Router Packet classification

Architecture of the Packet Classification Engine

This section explains about the steps involved to design the Packet Classification Engine Architecture. The design is based on the schematic representation explained above. PCE implements basic inspection from 5 header packet field in the Ethernet packet. Those five fields are considered input to the system. The inputs are Source IP Address, Destination IP Address, Source Port, Destination Port, Protocol, START, RESET and Clock Signal whereas the outputs are valid and forward.

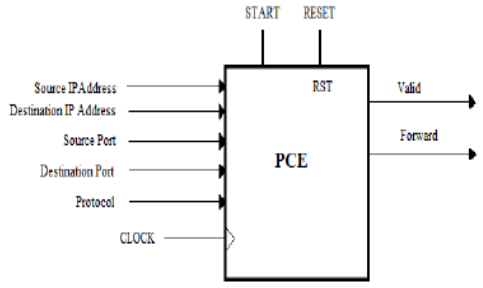

Figure 4. PCE Top Level

START signal is a control signal to start an inspection process by PCE. Every inspection is controlled by this signal to receive field input as system inputs and processes it. RESET signal cleans register value and

output and initialize the system. Output Forward signal shows 1 inspection process is finished and read Valid signal as a result. Valid signal shows fields' inspection result and control
packet buffer to forward or discard the Ethernet packet from the firewall. CLOCK signal is needed to synchronize the system process.

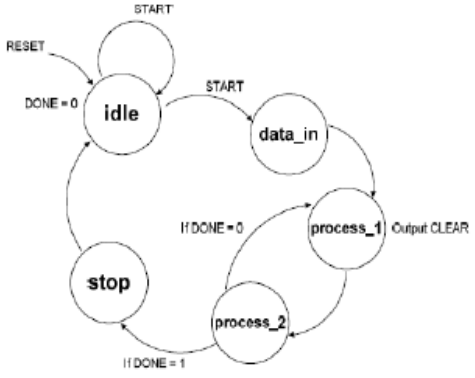

Figure 5.High level state machine of PCE

Process in PCE starts from input and ends with output signal generation, shown by High Level State Machine in Figure 7. When the system is turned on, PCE will be in the idle state. In this state, the system waits for start signal input gives logic „1". When the signal is given, all the field inputs are received and system will move to process_1 state. In this state, all the output is clean. PCE inspects the input fields based on the algorithm. Every one inspection, it moves to process_2 state and checks for inspection status. When the inspection is not finished, it will loop back to process_1. When it is finished, the system moves to stop state. In this state, outputs are generated and the system will move to idle state automatically until the next process.

Based on the algorithm explained above, we need an architecture which can maximized its advantages and works fast and reliable. The architecture must have simple design but also have the capability to facilitate various modifications of firewall rules and specification. The idea of building our PCE architecture comes from the algorithm itself. This PCE takes form of a single cycle processor whose processes are decided by instruction memory.

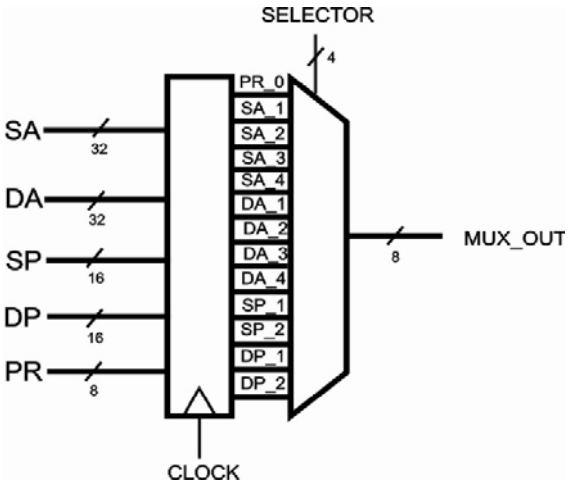

Figure 6. Subfields multiplexer

This architecture is most suitable for PCE in a firewall which implements variable rules and specification. Each of 8 bit sub-fields from tree-based algorithm will produce instructions for the processor. From 5 inspected fields of packet header, we will get 13 subfields, 4 sub-fields of Source IP Address, 4 sub-fields of Destination IP Address, 2 sub-fields of Source Port, 2 subfields of Destination Port, and 1 sub-field of Protocol. With 13 sub-fields inspected by PCE, we need mechanism to decide which sub-field is inspected. The processor will need a multiplexer with 4-bit selector for the data processed. Figure 8 shows the multiplexer used. Before the multiplexer, there is a register to secure the system from unwanted input change.

The most important component in sub-fields inspection by PCE is comparator. It compares sub-field input with sub-fields in the firewall rules. The comparison process has been explained in the tree-algorithm. Comparator is designed to compare 2 input 8 bit data and give result based on the criteria. These criteria are greater, less, or equal. If the comparison of 2 inputs and the criteria matches, this component will generate logic „1" result, and otherwise if it is not. Figure 9 shows comparator used in the PCE. The criteria is 2 bit input at the component.

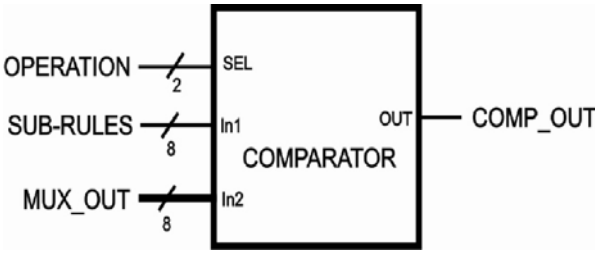

Figure 7. Comparator

The architecture needs a component to control system output at the right time, which is after field inspection process is finished. Logic „1“ from the comparator only shows that the comparison is suitable but the process is not over yet. Final compile unit is designed to generate output if the process is finished. Valid and Forward signal in PCE is generated by this component. Command to generate output will be saved in the last part of a process in the accessed rules by the system. Output generation is shows by the action leaves at the end of inspection tree. Final Compile Unit is shown in Figure 8. 1 bit is needed to control output generation.

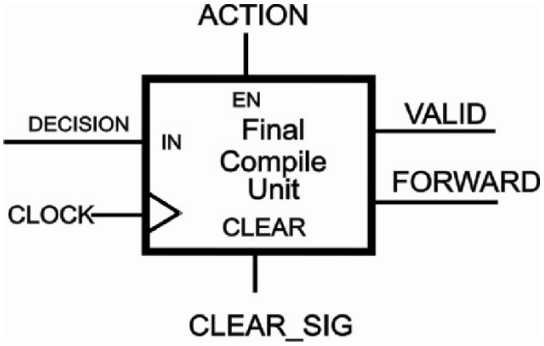

Figure 8. Final Compile Unit

By taking processor form as the architecture, PCE needs instructions to control process undergoing in each cycle. These instructions are saved in instruction memory. In our proposed PCE, the component saving the instructions is called Rules Memory. Each instruction can be accessed by giving address input to the component. Figure 9 shows Rules Memory used in PCE. Because in this research we do not use complex memory and updating mechanism yet, Rules Memory address is consisted only 8 bit. This address will change based on the specification.

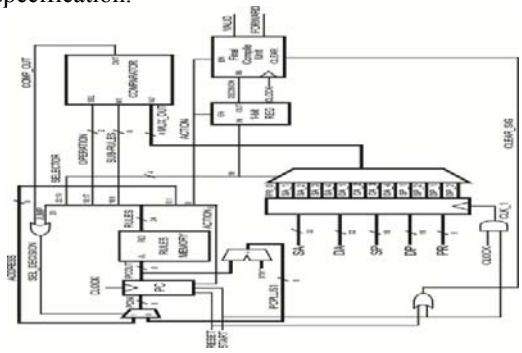

Figure 9 Packet Classification Engine Architecture

The hardware accelerator can save search structures for rule sets containing up to 49,000 rules, when implemented on a Stratix EP3SE260F115C3 FPGA, and rule sets containing up to 24,000 rules when implemented on a Cyclone EP3C120F484C7 FPGA.

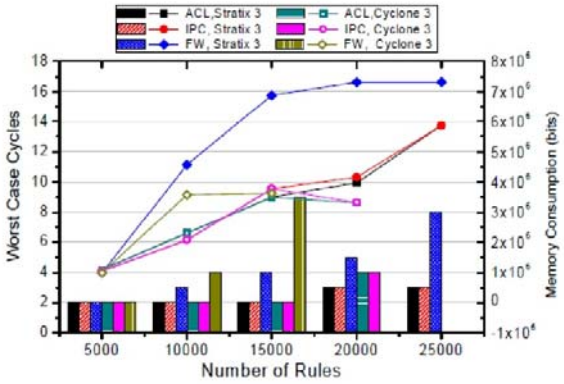

Figure 10. Memory usage(lines) and worst case number of clock cycles(bars).

The search structures built using the ACL1, IPC1 rule sets for both the Cyclone and Stratix implementation show similar performance results. This is because memory consumption is not a major problem for these rule sets as they don’t contain many rules with wildcard fields. For the search structure built using the FW1 rule sets it can be seen that the Stratix implementation shows better performance than the Cyclone implementation. This is because memory consumption is a problem due to the many rules containing wildcard fields. The FW1 rule set for example with 15,000 rules needs, at worst 9 clock cycles to classify a packet when 3,944,448 bits of memory are available for the search structure using the Cyclone. This figure is reduced to 4 clock cycles when the amount of memory available is doubled using a Stratix.

The router classification in the hardware accelerator is shown in Fig. 16 and 17, with the results for the hardware accelerator generated using a search structure requiring at worst 2 clock cycles to classify a packet. Post place and route simulations were carried out using Quartus 2 Power Play Power Analyzer Tool with VCD files generated by ModelSim. These results were compared to the power consumed by the state of the art Cypress Ayama 10000 Network Search Engine [32], which uses similar amounts of memory. The hardware accelerator implemented on the Cyclone 3 FPGA with 3,944,448 bits of memory is compared to the Cypress Ayama 10128 search engine with 4,608,000 bits of TCAM. The Stratix 3 implementation of the hardware accelerator with 7,888,896 bits of memory available is compared to the Cypress Ayama 10256 search engine with 9,216,000 bits of TCAM.

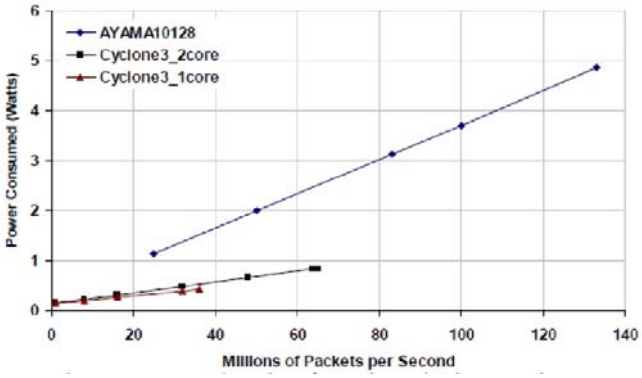

Figure11. Power vs. throughput for Cyclone 3 hardware accelerator.

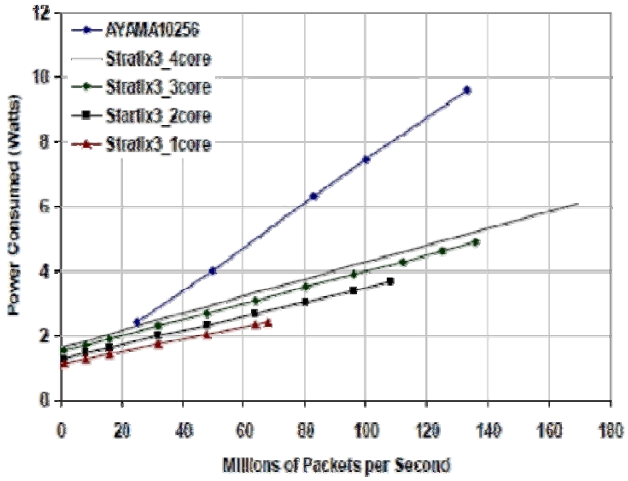

Figure 12. Power vs. throughput for Stratix 3 hardware accelerator.

Looking at Fig. 16 it can be seen that the TCAM has a maximum throughput of 133 mpps, while the hardware accelerator implemented on the Cyclone 3 FPGA with 2 packet classification engines has a maximum throughput of 65 mpps when running at 65 MHz. This throughput decreases to 36 mpps when 1 engine is used while running the hardware accelerator at 36 MHz. These levels of Throughputs are more than enough to cope with line rates up to OC-768. At these speeds the hardware accelerator shows an energy saving of 66.38%.

Conclusion

Packet classification is essential for most network system functionality and services, but it is complex, since it involves comparing multiple fields in a packet header against entries in the filter data set to decide the proper rule to apply for handling the packet [4]. This paper has considered a rapid packet classification mechanism realized by RCEPC able to not only exhibit high scalability in terms of both the classification time and the router size involved, but also effectively handle incremental updates to the filter data sets. Based on a single set-associative LuHa hash table (obtained by lumping a set of hash table units together) to support two- staged search, RCEPC promises to enjoy better classification performance than its known software-oriented counterpart, because the LuHa table narrows the search Scope effectively based on the source and the destination IP addresses of an arrival packet during the first stage, leading to fast search in the second stage. With its required router size lowered considerably, RCEPC makes it possible to hold entire search data structures in the local cache of each core within a contemporary processor, further elevating its classification performance than previous software- based techniques.

Note that theoretically pathological cases may occur despite encouraging pragmatic results by ρ=1:0, as we have witnessed in this study. For example, a large number of (host son the same subnet with) prefixes ρ/ω can differ only in a few bits. Hence, those prefixes can be router configuration engine into the same set after being
rounded down, say from ρ/ω down to ρ| ι_i , for $\iota_i \leq \omega < \iota_{i+1}$, under RCEPC. There are possible ways to deal with such cases and to avoid overwhelming the indexed set. A possible way is to use one and only one entry to keep the round-down prefix ρ| ι_i , as opposed to holding all ρ/ω"s in individual entries under the current design. Subsequently, the (ω- ι_i) round-down bits can form a secondary indexing structure to provide the differentiation (among rules specific to each host) and/or the round-down bits can be mingled with the remaining fields of the filter rules. Thus, each stage narrows the search range by small and manageable structures. These possible options are being explored.

Device		Stratix 3				Cyclone 3			Stratix 3				Cyclone 3			Stratix 3				Cyclone 3	
Speed		2 MHz							8 MHz							32 MHz					
Engines		1	2	3	4	1	2		1	2	3	4	1	2		1	2	3	4	1	2
ACL5000	OC-48	17	17	17	17	17	17	OC-192	28	27	29	29	28	27	OC-768	18	19	20	21	18	19
ACL10000		17	17	17	17	17	17		28	27	29	29	28	27		18	19	20	21	18	19
ACL15000		17	17	17	17	17	17		28	27	29	29	28	27		18	19	20	21	18	19
ACL20000		17	17	17	17	32	32		28	27	29	29	44	44		18	19	20	21	44	43
ACL25000		17	17	17	17				28	27	29	29				18	19	20	21		
FW5000		17	17	17	17	17	17		28	27	29	29	28	27		18	19	20	21	18	19
FW10000		20	20	19	20	24	24		36	36	36	35	42	42		24	25	26	27	35	35
FW15000		20	21	21	21	49	49		40	39	40	39	61	60		33	33	33	33	57	57
FW20000		33	33	33	33				50	49	50	50				45	45	44	44		
FW25000		46	46	46	46				57	57	55	55				53	53	53	53		
IPC5000		17	17	17	17	17	17		28	27	29	29	28	27		18	19	20	21	18	19
IPC10000		17	17	17	17	17	17		28	27	29	29	28	27		18	19	20	21	18	19
IPC15000		17	17	17	17	17	17		28	27	29	29	28	27		18	19	20	21	18	19
IPC20000		17	17	17	17	18	18		28	27	29	29	33	32		18	19	20	21	27	28
IPC25000		17	17	17	17				28	27	29	29				18	19	20	21		

Figure 17.Maximum Buffer usage

[1] A. Broder and M. Mitzenmacher, "Using Multiple Hash Functions to Improve IP Lookups," Proc. 20th Ann. Joint Conf. IEEE Computer and Comm. Soc. (INFOCOM '01), pp. 1454-1463, Apr. 2001.

[2] Y.H. cho and W.h. Magione-Sdmith, "Deep Packet Filter with Dedicated Logic and Read Only Memories," Proc. 12th IEEE Symp. Field-Programmable Custom Computing Machines. pp. 125-134. Apr. 2004.

[3] Q. Dong et al., "Wire Speed Packet Classification without TCAMs: A Few More Registers (and a Bit of Logic) Are Enough," Proc. ACM SIGMETRICS Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS '07), pp. 253-264, June 2007.

[4] P. Gupta and N. McKeown, "Packet Classification on Multiple fields," Proc. ACM Ann. Conf. Special Interest Group on Data Comm. (SIGCOMM '99), pp. 147-160, Aug./Sept. 1999.

[5] F.-Y. Lee and S. Shieh, "Packet Classification Using Diagonal- Based Tuple Space Search," Computer Networks, vol. 50, pp. 1406-1423, 2006.

[6] H. Song and J.W. Lockwood, "Efficient Packet Classification for Network Intrusion Detection Using FPGA," Proc. ACM/SIGDA 13th Int'l Symp. Field Programmable Gate Arrays(FPGA '05), pp. 238-245, Feb. 2005.

[7] V. Srinivasan, S. Suri, and G. Varghese, "Packet Classification Using Tuple Space Search," Proc. ACM SIGCOMM '99, pp. 135-146, Aug./Sept. 1999.

[8] D.E. Taylor, "Survey and Taxonomy of Packet Classification Techniques," ACM Computing Surveys, vol. 37, no. 3, pp. 238-275, Sept. 2005

[9] G. Wang and N.-F. Tzeng "TCAM-Based Forwarding engine with Minimum Independent Prefix Set(MIPS) for Fast Updating," Proc. IEEE Int'l Conf. Comm. (ICC '06), June 2006.

[10] P. Warkhede. S. Suri, and G. Varghese, "Fast Packet Classification for Two-Dimensional Conflict-Free Filters," Proc. 20th IEEE INFOCOM '01. pp. 1434-1443, Apr. 2001

[11] P. Gupta and N. McKeown, "Packet classification on multiple fields", Proceedings of ACM Sigcomm '99. pp.147-160, August 1999.

[12] F. Chang et al., "Efficient Packet Classification with DigestCaches," Proc. Third Workshop Network Processors and Applications(NP-3), Feb. 2004.

[13] W.T. Chen. S.B. Shih, and J.L. Chiang, "A Two-Stage Packet Classification Algorithm," Proc. 17th Int'l Conf. Advanced Information Networking and Applications (AINA '03), pp. 762-767, Mar. 2003.

[14] S. Dharmapurikar et al., "Fast Packet Classification Using Bloom Filters," Proc. IEEE/ACM Symp. Architectures for Networking and Comm. Systems(ANCS '06), pp. 61-70, Dec. 2006.

[15] P. Gupta and N. McKeown, "Classifying Packets with Hierarchical Inteligent Cuttings," IEEE Micro, vol. 20, no. 1, pp. 34-41, Jan./Feb. 2000.

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

[17] T.V. Lakshman and D. stiliadis, "High-Speed Policy-Based Packet Forwarding Using Efficient Multi-Dimensional Range Matching," Proc. ACM SIGCOMM '98, pp. 191-202, Aug./Sept. 1998

[18] J. van Lunteren and T. Engbersen, "Fast and Scalable Packet Classification," IEEE J. Selected Areas in Comm., vol. 21, no. 4, pp. 560-571, May 2003.

[19] D. Shah and P. Gupta, "Fast Incremental Updates on Ternary- CAMs for Routing Lookups and Packet Classification," Proc. Eighth Ann. IEEE Symp. High-Performance Interconnets (Hot Interconnects '08), pp. 145-153, Aug. 2000.

[20] S. Singh et al., "Packet Classification Using Multidimensional Cutting," Proc. ACM SIGCOMM '03. pp. 213-114, Aug. 2003.
[21] Z. Wu, M. Xie, and H. Wang, "Swift: A Fast Dynamic Packet Filter," Proc. Fifth USENIX Symp. Networked Systems Design and Implementation (NSDI '08), pp. 279-292, Apr. 2008.
[22] L. Xu et al., "Packet Classification Algorithms: From Theory to Practice," Proc. 28th IEEE INFOCOM '09, APR. 2009.
[23] Yu Lei, Deng Ya-Ping, Wnag Jiang-Bo, and Jiang Chao-Yong, A Novel IP Packet Classification Algorithm Based on Hierarchical Intelligent Cuttings, The 6th International Conference on ITS Telecommunication Proceedings, 2006:1033-1036.
[24] Zheng Kai, Liang Zhiyong, and Ge Yi, Parallel Packet Classification via Policy Table Pre-Partitioning, IEEE Globecom, 2005:73-78.
[25] Xuehong Sun, Sartaj K. Sahni, and Yiqiang Q Zhao,Packet Classification Consuming Small Amount of Memory, IEEE/ACM TRANSACTIONS ON NETWORKING, 2005, 13(5):1135-1145.