A High-Speed and Memory Efficient Pipeline Architecture for Packet Classification

Yeim-Kuan Chang, Yi-Shang Lin, and Cheng-Chien Su
Department of Computer Science and Information Engineering
National Cheng Kung University, Tainan, 701, Taiwan
{ykchang, p7696104, p7894104}@mail.ncku.edu.tw

Abstract—Multi-field Packet classification is the main function in high-performance routers. The current router design goal of achieving a throughput higher than 40 Gbps and supporting large rule sets simultaneously is difficult to be fulfilled by software approaches. In this paper, a set pruning trie based pipelined architecture called Set Pruning Multi-Bit Trie (SPMT) is proposed for multi-field packet classification. However, the problem of rule duplications in SPMT that may cause a memory blowup must be solved in order to implement SPMT with large rule sets in FPGA devices consisting of limited on-chip memory. We will propose two rule grouping schemes to reduce rule duplications in SPMT. The first scheme called Partition by Wildcards (PW) divides the rules into subgroups based on the positions of their wildcard fields. The second scheme called Partition by Length (PL) rules partitions the rules into subgroups according to their prefix lengths. Based on our performance experiments on Xilinx Virtex-5 FPGA device, the proposed pipeline architecture can achieve a throughput of over 100 Gbps with dual port memory. Also, the rule sets of up to 10k rules can be fit into the on-chip memory of Xilinx Virtex-5 FPGA device.

I. INTRODUCTION

As the Internet becomes widely used, the next generation routers need to support a variety of value-added network services such as QoS differentiation, traffic billing, network security, and others. In order to support these services, packet classification that classifies packets traversing the Internet into flows is applied. It is well known that multi-field packet classification is a difficult problem [11].

Algorithms for packet classification can be divided into two groups: software-based and hardware-based. Software-based algorithms have an advantage of the flexibility but are slow. For example, pipeline and parallel techniques are hard to apply to software-based algorithms and can hurt throughputs. Today, researchers have turned to the hardware-based solutions that use field programmable gate array (FPGA), application specific integrated circuit (ASIC), and ternary content addressable memory (TCAM). ASIC and FPGA based approaches usually store the rule set in static random access memory (SRAM) or TCAM that stores not only 0 and 1 but also “don’t care” value on TCAM cells. At present, the achievable link rate is OC-768, i.e., 40Gbps, which is equivalent to the processing speed of one 40-byte packet every 8 ns.

In order to achieve such high throughput, we develop a data structure called set pruning multi-bit trie (SPMT) which is very suitable for the pipeline architecture. In order to reduce the memory blowout problem caused by the rule duplications in set pruning tries, two rule partitioning schemes are proposed to divide rules into subgroups based on wildcards and prefix lengths. Then, SPMT is built for each subgroup and all the SPMTs are searched in parallel to obtain the best matched rule.

Our experimental results conducted on Xilinx Virtex-5 XC5VFX200T FPGA [13] device show that the proposed pipeline architecture can achieve a throughput of over 100 Gbps with dual port memory. Also, the block RAMs of size 16,416 Kbits available in the Xilinx Virtex-5 XC5VFX200T FPGA are capable of storing the rule sets of up to 10k rules.

The rest of the paper is organized as follows. In section 2, we introduce the background about packet classification problem and related work. The section 3 illustrates the proposed design. Section 4 describes implementation of our proposed design. In section 5, we present our experiment results. The last section is the conclusion.

II. BACKGROUND AND RELATED WORK

Packet classification in the routers classifies packets into flows by searching the rule set which consists of a finite set of rules (or called filters) for obtaining the action, i.e. flow ID. Each filter consists of a set of fields and the associated action. The types of match conditions in the fields are typically prefixes for 32-bit source and destination IP fields (SA/DA), ranges for the 16-bit source and destination ports (SP/DP), and an 8-bit protocol number for transport layer protocol (PROT).

Formally, a filter $F$ with $d$ fields is defined as $F = (f_1, f_2, \ldots, f_d)$. A packet $P$ is said to match a particular filter $F$ if the $i$th header field of packet $P$ is contained in $f_i$ for all $i$. The packet may match multiple filters and the classifier only applies the action with highest priority among all the matching filters [11]. An example rule set is shown in Table I while we assume that the packet header fields consist of 8-bit source/destination IP address prefix, 5-bit source/destination port range, and 8-bit protocol number.

Not so many packet classification proposals have been implemented on FPGAs. Existing FPGA implementations of packet classification algorithms include Improved HyperCuts [3], Power Saved HyperCuts [6], Dual Stage Bloom Filter Classification Engine (2sBFCE) [7], Bloom Based Packet Classification (B2PC) [8], and Nest Level Tuple Merging and Cross-product (NTLMC) [2]. Also, it is worth mentioning that several pipeline architectures have been proposed for IP lookup based on trie data structures, such as [4][5].

An improved data structure called set pruning trie [10] avoids the backtrack operations and thus increases the search speed. However, set pruning trie incurs a memory blowout problem caused by the rule duplications in the tries at the
In this section, we propose to use the multi-bit version of set pruning trie as our basic data structure because it is easy to have a very fast pipeline implementation for the trie-based structure. Due to a lot of rule duplications in the lower level to the lower levels incur a lot of memory usage. Hence, we proceed to seek improvements by analyzing the wildcard field values contained in rules as much as possible. Additionally, we target on the 5-field rule tables.

### Proposed Schemes

In this section, we propose to use the multi-bit version of set pruning trie as our basic data structure because it is easy to have a very fast pipeline implementation for the trie-based structure. Due to a lot of rule duplications in the lower level to the lower levels incur a lot of memory usage. Hence, we proceed to seek improvements by analyzing the wildcard field values contained in rules as much as possible. Additionally, we target on the 5-field rule tables.

#### Set Pruning Multi-Bit Trie

It is well-known that the traditional multi-bit trie is very suitable for the IP address lookup pipelined search engine [5]. Because the 5-field set pruning trie uses too many stages, we develop a multi-bit version of the set pruning trie called set pruning multi-bit trie (SPMT) to minimize the number of pipeline stages. The SPMT building process is very similar to the set pruning trie. The only difference is that when building SPMT we have to avoid the redundant data structures caused by the address expansion onto the multi-bit nodes. In other words, the nodes should be shared and pointed to by as many pointers as possible. For the reason of limited space, Fig. 2 only shows the 2-bit SPMT constructed from the source and destination IP addresses of the rules in Table I. Each node contains four elements with indices of 00, 01, 10, 11 in the 2-bit SPMT. Each element contains a next-field link and a next-level link that are drawn by dotted and solid lines in Fig. 2, respectively. The root node has 2 prefixes “0*” and “0*” in the SA field. The prefix “0*” is expanded to 00 and 01 and the prefix “0*” is expanded to 00, 01, 10 and 11. Therefore, the next-field link of the two elements 00 and 01 of the root node point to the same next-field node X that consists of the sub-rules from R1, R2, R3, and R6. Similarly, the elements 10 and 11 of the root node point to node Y that consists of sub-rules from only R6. When we construct DA field, the nodes W, X, Y could share the same node Z pointed to by the same next-level link. Our experimental results show that the node sharing can reduce 40%-45% of the trie nodes.

The search operation is similar to that of the traditional set pruning trie [10]. The only difference is that we use multiple bits to traverse the SPMT nodes. Although some nodes can be shared, duplications inside the nodes are not removed. Also, the duplications caused by pushing the rules in upper levels to the lower levels incur a lot of memory usage. Hence, we proceed to seek improvements by analyzing the wildcard field values contained in rules as follows.

#### Partitioning by Wildcards (PW)

We can divide 5-field rule set into 32 possible cases (subgroups) such that the wildcard field values appear in some specified fields. However, we ignore the protocol field for dividing rules into subgroups because the protocol field has more values of TCP and UDP than wildcard and thus the impact on rule duplication from protocol field in set pruning trie is much smaller than the other four fields. As shown in the example of Fig. 1, wildcard fields will be duplicated in many places in the lower levels. Hence, we can divide the rules into at most 16 subgroups and an SPMT is built for each subgroup by ignoring the wildcard fields. Thus, a lot of rule duplications can be removed from the resulting SPMT.

As shown in Table II, we analyze three rule sets containing about 10K rules generated by ClassBench [12]. The IPC tables have more non-empty subgroups than ACL and Firewall tables which just have 5-6 subgroups. Although we divide the rule set into subgroups by wildcards, non-wildcard fields are still covered by short prefix of lengths 1 and 2. Based on our analysis on the length distribution of the source and destination IP fields, some of the rule tables contain up to 50 percentages of the rules whose lengths are 1 or 2. Therefore, we can further divide the subgroups partitioned by wildcards into many smaller subgroups and thus the reduction of rule duplications can be reduced further.

#### Partitioning by Lengths (PL)

<table>
<thead>
<tr>
<th>Filter</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>SA</td>
</tr>
<tr>
<td>00</td>
<td>0*</td>
</tr>
<tr>
<td>00</td>
<td>0*</td>
</tr>
<tr>
<td>00</td>
<td>1*</td>
</tr>
<tr>
<td>01</td>
<td>11*</td>
</tr>
<tr>
<td>00</td>
<td>001*</td>
</tr>
<tr>
<td>11</td>
<td>11*</td>
</tr>
<tr>
<td>00</td>
<td>001*</td>
</tr>
</tbody>
</table>

**Table I. A Sample Rule Set**

---

**Figure 1.** A 2-D 1-bit set pruning trie example.

**Figure 2.** The 2-bit SPMT constructed from the rules in Table I.
Consider the general problem of determining the minimum duplicating cost of $R_6$. Totally, rule $R_6$ is duplicated 3 times and thus the further computation is needed in the third field of Figure 1.

Since the second field is already the penultimate field, no of rule $R_6$ can be obtained by summing up all the nodes in the second field trie that are covered by the second field values set to * and 111*, respectively. Rule $R_6$ in the binary trie of the first field. Fig. 1 shows that there should be duplicated in the valid nodes (prefixes) covered by $R_6$ in the binary trie of the first field. The dynamic programming formula is based on the Duplicating Cost (DC) which is defined to be number of duplications. The initial problem is DC $[L, k]$, where $L$ is the maximum prefix length of the first field that is targeted for partition.

### IV. IMPLEMENTATION

A pipelined and parallel design is used to improve the throughput of our proposed SPMT. The trie nodes in the same level are mapped to a single pipeline stage. Each packet traversing the SPMT level by level is equivalent to traversing the pipeline architecture stage by stage. Each pipeline stage must perform the following actions:

1. The memory of the SPMT is accessed to obtain the next level address and next field address.
2. The next field memory address of the longest prefix is recorded from the first field to the last field.
3. If the last field is reached, the matching rule information is obtained and then its priority is compared with matching rule obtained from previous stages.

FPGAs provide massive parallelism and high-speed dual-port Block RAMs distributed across the device. We exploit these features and propose a parallel architecture, as shown Fig. 3. The design is based on the following considerations:

1. A pipeline $P_i$ is constructed for each subgroup ($PRT_i$) of the original rules after applying the proposed PW and PL partitioning schemes. The entire rule set is equal to $\sum PRT_i$ for $i = 1 \ldots n$.
2. Traversing the SPMT level to level and each level is mapped to a stage in pipeline. It is shown as gray color blocks in Fig 3.
3. Assume a field is $X$-bit wide and stride is $Y$-bit. The number of pipeline stages required is $\lceil XY \rceil$.
4. The incoming packets will be input in all pipelines in parallel. Then, each pipeline is executed independently. The packet header field arrangement engine will select a field order to construct the proposed SPMT for each subgroup.

Each pipeline will output a result which will be sent to priority resolver to select the rule with highest priority as the final result.
and dual-port memory. Our hardware cost is lower than Improved HyperCuts and clock rate also is higher than Improved HyperCuts. As we can see, both schemes can achieve the throughput of OC-768 but our proposed schemes are better. For the table of 10k rules, our slice utilization is lower (24% vs. 33%) but our Block RAM utilization is higher (94% vs. 89%) than Improved HyperCuts. Hence, Xilinx Virtex-5 XC5VFXX200T is sufficient to support the proposed SPMT with 10k rules. Table V compares the throughputs of the proposed SPMT and other existing schemes. Our proposed schemes are implemented with Xilinx Virtex-5 XC5VFXX200T and packet size is 40 bytes. We can see that the proposed SPMT outperforms all the schemes.

VI. CONCLUSION

In this paper, we proposed a pipeline packet classification architecture which aims at achieving a very high throughput for large rule table. The Set Pruning Multi-bit Trie based on the set pruning trie is first proposed. The drawback of memory blowout for large rule tables caused by rule duplication is solved by the proposed rule table partitioning schemes PW and PL. The PL partitioning scheme uses dynamic programming for choosing the best lengths to divide rule table into many independent subgroups. Based on our performance experiments on Xilinx Virtex-5 FPGA device, our proposed design can achieve the throughput over 100 Gbps with dual port memory while supporting up to 10k rules.

REFERENCES