Memory Optimization for Packet Classification Algorithms

Juraj Blaho, Jan Kořenek
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno, Czech Republic
xblaho00@stud.fit.vutbr.cz,
korenek@fit.vutbr.cz

Viktor Puš
CESNET z. s. p. o.
Zikova 4, 160 00 Prague, Czech Republic
pus@liberouter.org

ABSTRACT
We propose novel method how to reduce data structure size for the family of packet classification algorithms at the cost of additional pipelined processing with only small amount of logic resources. The reduction significantly decreases overhead given by the crossproduct nature of classification rules. Therefore the data structure can be compressed to 10 % on average. As high compression ratio is achieved, fast on-chip memory can be used to store data structures and hardware architectures can process network traffic at significantly higher speed.

Categories and Subject Descriptors
B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Gate arrays, Algorithms implemented in hardware*;
C.2.0 [**Computer-Communication Networks**]: General—*Security and protection (e.g., firewalls)*

General Terms
Design, Performance, Security, Complexity

Keywords
Packet Classification, FPGA, SRAM, Optimization

1. INTRODUCTION
With the rapid development of computer networks, traffic filtering has become one of the first steps in securing any network or computer. Basic traffic filtering device is the firewall, which performs per-packet decision based on the given set of rules. As network speeds are increasing, the demand for the speed of packet classification algorithms is also growing.

The classification algorithm contains a set of rules ordered by priority. The goal of a packet classification algorithm is to find the matching rule with the highest priority. The output of the algorithm is then the number of the matched rule. Software solutions for the packet classification problem are available, but their performance is not sufficient for wire-speed processing in the highest-speed networks. Existing hardware architectures can achieve multigigabit speeds only at the cost of large data structures and they must deal with great memory overheads imposed by fields crossproduct.

We propose a novel method that significantly reduces memory requirements for classification algorithms by pipelined processing with only small amount of logic resources. As the processing is pipelined, only system latency is increased, but the throughput is not affected. The proposed method utilizes classification rules structure and can be used for any decomposition-based classification algorithm, where the processing is split between the longest prefix match (LPM) and rule matching operations.

2. RELATED WORK
As the packet classification problem is inherently hard from a theoretical standpoint, a large number of hardware and software solutions [1, 3] have been proposed. These solutions are based on exhaustive search, decision tree and grid-of-tries. From the wide choice of packet classification algorithms available, we discuss only those which are related to our work. All of them belong to the family of decomposition-based methods.

First step in these methods is the LPM operation, which is performed independently in each dimension. From the given set of prefixes with various lengths, the LPM algorithm finds the one that best fits the given full-length value. Basic algorithm for the LPM is a trie – the tree algorithm processing one input bit at each tree level and returning the last valid prefix visited.

Result of the LPM stage is a vector of prefixes, each prefix is represented by unique number. After the LPM, all fields of the resulting LPM vector must be combined together to get the resulting rule number. Basic crossproduct algorithm [5] precomputes a crossproduct table, which contains resulting rule numbers for all possible combinations of prefixes. Because of the multiplicative nature of the crossproduct, this table may become extremely large. Instead of crossproduct, term *pseudorules* with slightly different meaning is used in following works [2, 4].

3. MEMORY OPTIMIZATION
Our method is based on the observation that many classification rules often do not specify all the classification fields.

For example, if user wants to block a specific source IP address, the rule does not specify any destination IP address or port number. We use term *ANY* for these field conditions. This means that any destination IP and any port number can match this rule. However, the rule can create many pseudorules because all more specific destination IPs and ports (specified in other rules) have to be covered by pseudorules.

Fig. 1 is an example for two fields, where an ANY value in the rule produces many pseudorules. While the rule is quite *general*, we must deal with all more *specific* pseudorules. The situation is even worse for multiple fields, because pseudorules create crossproduct.

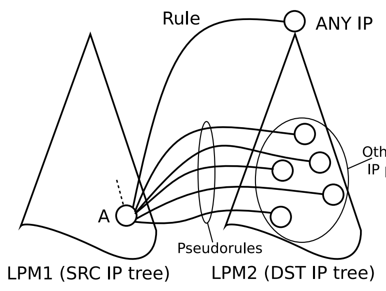

Figure 1: One of the most severe causes of pseudorules: ANY values in the ruleset.

We address this issue and propose a solution to insert a *generalization stage* (GS) into the classification algorithm pipeline after LPM engines. The GS is able to replace LPM results with more general ANY value in certain situations. As a result, the number of output combinations is reduced after GS. This will result in smaller data structures of the following stages of all crossproduct algorithms. GS contains a set of *generalization rules* (GRs). The LPM result is compared to all values stored in the GS, and in the case of match, additional information in matched GR instructs the GS to replace some of the other LPM results with the ANY value (to perform generalization). Thanks to this replacement, the number of possible GS outputs is reduced. We also provide algorithm to find GRs.

The hardware implementation of the Generalization Stage must have enough performance to process packets at the wirespeed. The proposed scheme is a simple pipeline step inserted after LPM stage of the classification algorithm. As all classification stages are in processing pipeline, throughput of the whole system is not affected. The Generalizatio Stage has a small on-chip CAM for each classification field, where Every LPM result is compared to GRs. If any GR is matched, additional information instructs the GS to replace some LPM results with the ANY value.

4. RESULTS

We performed analysis and Generalization Rules search for several real-life firewall sets from the university campus network, as well as synthetic rulesets generated by the ClassBench tool. The compression ratios for all mentioned rulesets are shown in Table 1. Number of pseudorules before and after reduction are compared.

Ruleset	Rules	Pseu. before	Pseu. after	Ratio
real1	68	168 000	27 888	0.166
real2	335	44 153	2 197	0.049
real3	1194	114 826	19 600	0.170
real4	1529	2 584 281	63 546	0.024
synth1	47	42 500	11 570	0.272
synth2	472	4 985 457	263 085	0.052
synth3	962	3 205 517	949 205	0.296

Table 1: Results of generalization rules search.

5. CONCLUSION

In this paper, we propose a novel method how to significantly reduce memory resources for fast packet classification. The presented method can be used for any decomposition-based classification algorithm where the processing is split between the longest prefix match and a rule matching operations. In addition, the proposed method is orthogonal to other existing memory reduction approaches and provides further reduction in memory needs.

The proposed reduction method achieves compression ratio up to 0.024 and 0.1 in average for all available rulesets. It means that the memory size can be reduced 10 times in average and about 50 times in the best case. The results depend on the size of the ruleset and its structure. After the reduction, small rulesets can fit into the faster on-chip memory and the classification algorithm can be speed-up.

Acknowledgment

This research has been partially supported by the Research Plan No. MSM, 6383917201 – Optical National Research Network and its New Applications and Research Plan No. MSM, 0021630528 – Security-Oriented Research in Information Technology.

6. REFERENCES

[1] F. Baboescu, S. Singh, and G. Varghese. Packet classification for core routers: Is there an alternative to CAMs? In *INFOCOM*, 2003.

[2] S. Dharmapurikar, H. Song, J. Turner, and J. Lockwood. Fast packet classification using Bloom filters. In *ANCS '06: Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*, pages 61–70, New York, NY, USA, 2006. ACM.

[3] P. Gupta and N. McKeown. Algorithms for packet classification, 2001.

[4] V. Puš and J. Kořenek. Fast and scalable packet classification using perfect hash functions. In *FPGA '09: Proceedings of the 17th international ACM/SIGDA symposium on Field programmable gate arrays*, New York, NY, USA, 2009. ACM.

[5] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel. Fast and scalable layer four switching. *SIGCOMM Comput. Commun. Rev.*, 28(4):191–202, 1998.