ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Implementation of Boundary Cutting Algorithm Using Packet Classification

Dasari Mallesh
M.Tech Student
Department of CSE
Vignana Bharathi Institute of Technology, Hyderabad.

V Sridhar Reddy
Associate Professor
Department of CSE
Vignana Bharathi Institute of Technology, Hyderabad.

ABSTRACT:

Decision-tree-based packet classification algorithms such as HiCuts, HyperCuts, and EffiCuts show excellent search performance by exploiting the geometrical representation of rules in a classifier and searching for a geometric subspace to which each input packet belongs. However, decision tree algorithms involve complicated heuristics for determining the field and number of cuts. Moreover, fixed interval-based cutting not relating to the actual space that each rule covers is ineffective and results in a huge storage requirement. A new efficient packet classification algorithm using boundary cutting is proposed in this paper. The proposed algorithm finds out the space that each rule covers and performs the cutting according to the space boundary. Hence, the cutting in the proposed algorithm is deterministic rather than involving the complicated heuristics, and it is more effective in providing improved search performance and more efficient in memory requirement. For rule sets with 1000–100 000 rules, simulation results show that the proposed boundary cutting algorithm provides a packet classification through 10–23 on-chip memory accesses and 1–4 off-chip memory accesses in average.

Index Terms — ***Binary search, boundary cutting, decision tree algorithms, HiCuts, packet classification.***

INTRODUCTION:

PACKET classification is an essential function in Internet routers that provides value-added services such as network security and quality of service (QoS) [1]. A packet classifier should comparemultiple header fields of the received packet against a set of predefined rules and return the identity of the highest-priority rule that matches the packet header. The use of application-specific integrated circuits (ASICs) with off-chip ternary content addressable memories (TCAMs) has been the best solution for wire-speed packet forwarding [2], [3]. However, TCAMs have some limitations. TCAMs consume 150 more power per bit than staticrandom access memories (SRAMs). TCAMs consume around 30%–40% of total line card power [4], [5]. When line cards are stacked together, TCAMs impose a high cost on cooling systems. TCAMs also cost about 30 more per bit of storage than double-data-rate SRAMs. Moreover, for an -bit port range field, it may require TCAM entries, making the exploration of algorithmic alternatives necessary. Many algorithms and architectures have been proposed over the years in an effort to identify an effective packet classification solution [6]–[35]. Use of a high bandwidth and a small on-chipmemorywhile the rule database for packet classification resides in the slower and higher capacity off-chip memory by proper partitioning is desirable [36], [37]. Performance metrics for packet classification algorithms primarily include the processing speed, as packet classification should be carried out in wire-speed for every incoming packet. Processing speed is evaluated using the number of off-chip memory accesses required to determine the class of a packet because it is the slowest operation in packet classification. The amount of memory required to store the packet classification table should be also considered. Most traditional applications require the highest priority

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

matching. However, the multimatch classification concept is becoming an important research item because of the increasing need for network security, such as network intrusion detection systems (NIDS) and worm detection, or in new application programs such as load balancing and packet-level accounting [7]. In NIDS, a packet may match multiple rule headers, in which case the related rule options for all of the matching rule headers need to be identified to enable later verification. In accounting, multiple counters may need to be updated for a given packet, making multimatch classification necessary for the identification of the relevant counters for each packet [4].

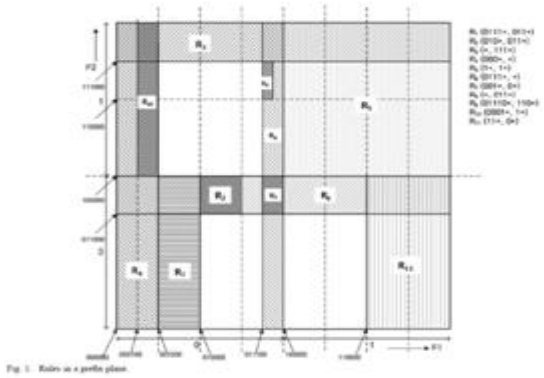
Existing System:

Our study analyzed various decision-tree-based packet classification algorithms. If a decision tree is properly partitioned so that the internal tree nodes are stored in an on-chip memory and a large rule database is stored in an off-chip memory, the decision tree algorithm can provide very high-speed search performance. Moreover, decision tree algorithms naturally enable both the highest-priority match and the multimatch packet classification. Earlier decision tree algorithms such as HiCuts [8] and HyperCuts [9] select the field and number of cuts based on a locally optimized decision, which compromises the search speed and the memory requirement. This process requires a fair amount of preprocessing,which involves complicated heuristics related to each given rule set. The computation required for the preprocessing consumes much memory and construction time, making it difficult for those algorithms to be extended to large rule sets because of memory problems in building the decision trees. Moreover, the cutting is based on a fixed interval, which does not consider the actual space that each rule covers; hence it is ineffective.

Proposed System:

In this paper, we propose a new efficient packet classification algorithm based on boundary cutting. Cutting in the proposed algorithm is based on the disjoint space covered by each rule. Hence, the packet classification table using the proposed algorithm is deterministically built and does not require the complicated heuristics used by earlier decision tree algorithms. The proposed algorithm has two main advantages. First, the boundary cutting of the proposed algorithm is more effective than that of earlier algorithms since it is based on rule boundaries rather than fixed intervals. Hence, the amount of required memory is significantly reduced. Second, although BC loses the indexing ability at internal nodes, the binary search at internal nodes provides good search performance. The organization of the paper is as follows. Section II provides an overview of the earlier decision tree algorithms. Section III describes the proposed BC algorithm. Section IV describes the refined structure of the BC algorithm, termed selective BC. Section V shows the data structure of each decision tree algorithm. Section VI shows the performance evaluation results using three different types of rule sets with 1000–100 000 rules each acquired from a publicly available database.

RELATED WORKS



Packet classification can be formally defined as follows [10]: Packet P matches rule , for , if all the header fields , for , of the packet match the corresponding fields in , where is the number of rules and is the number of fields. If a packet matches multiple rules, the rule with the highest priority is returned for a single-best-match packet classification problem and the list of matching rules is returned for

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

the multimatch packet classification problem. Rule sets are generally composed of five fields. The first two fields are related to source and destination prefixes and require prefix match operation. The next two fields are related to source and destination port ranges (or numbers), which require range match. The last field is related to protocol type and requires an exact match.

HiCuts

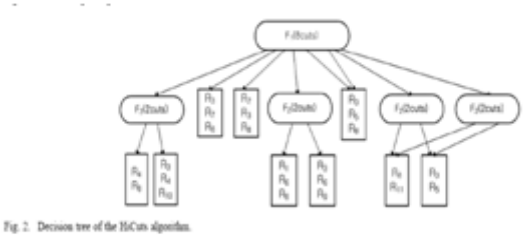


Fig. 2. Decision tree of the HiCuts algorithm.

Each rule defines a five-dimensional hypercube in a five-dimensional space, and each packet header defines a point in the space. The HiCuts algorithm [8] recursively cuts the space into subspaces using one dimension per step. Each subspace ends up with fewer overlapped rule hypercubes that allow for a linear search. In the construction of a decision tree of the HiCuts algorithm, a large number of cuts consumes more storage, and a small number of cuts causes slower search performance. It is challenging to balance the storage requirement and the search speed. The HiCuts algorithm uses two parameters, a space factor (spfac) and a threshold (binth), in tuning the heuristics, which trade off the depth of the decision tree against the memory amount. The field in which a cut may be executed is chosen to minimize the maximum number of rules included in any subspace.

HyperCuts

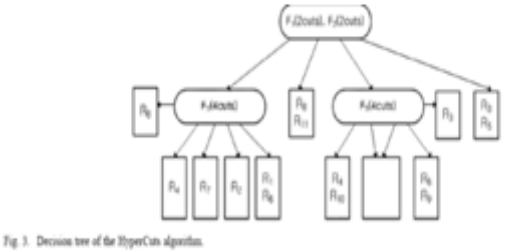


Fig. 3. Decision tree of the HyperCuts algorithm.

While the HiCuts algorithm only considers one field at a time for selecting cut dimension, the HyperCuts algorithm [9] considers multiple fields at a time. For the same example set, the decision tree of the HyperCuts algorithm is shown in Fig. 3. The spfac and binth are set as 1.5 and 3, respectively. As shown at the root node, the and fields are used simultaneously for cutting. Note that each edge of the root node represents the bit combination of 00, 10, 01, and 11, respectively, which is one bit in the first field followed by one bit in the second field.

PROPOSED ALGORITHM

HiCuts and HyperCuts algorithms perform cutting based on a fixed interval, and hence the partitioning is ineffective in reducing the number of rules belonging to a subspace. Moreover, when the number of cuts in a field is being determined, complicated preprocessing should be made to balance the required memory size and the search performance. In this study, we propose a deterministic cutting algorithm based on each rule boundary, termed as boundary cutting (BC) algorithm.

Building a BC Decision Tree

When the cutting of a prefix plane according to rule boundaries is performed, both the starting and the ending boundaries of each rule can be used for cutting, but cutting by either is sufficient since decision tree algorithms generally search for a subspace in which an input packet belongs and the headers of the given input are compared for entire fields to the rules belonging to the subspace (represented by a leaf node of the decision tree).

Searching in the BC Algorithm

The cuts at each internal node of the BC decision tree do not have fixed intervals. Hence, at each internal node of the tree, a binary search is required to determine the proper edge to follow for a given input. However, it will be shown in Section VI that the BC algorithm provides better search performance than the HiCuts algorithm despite of the memory access for the binary search at the internal nodes.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

DATA STRUCTURE

There are two different ways of storing rules in decision tree algorithms. The first way separates a rule table from a decision tree. In this case, each rule is stored only once in the rule table, while each leaf node of a decision tree has pointers to the rule table for the rules included in the leaf. The number of rule pointers that each leaf must hold equals the binth. In searching for the best matching rule for a given packet or the list of all matching rules, after a leaf node in the decision tree is reached and the number of rules included in the leaf is identified, extra memory accesses are required to access the rule table. The other way involves storing rules within leaf nodes. In this case, search performance is better since extra access to the rule table is avoided, but extra memory overhead is caused due to rule replication. In our simulation in this paper, it is assumed that rules are stored in leaf nodes since the search performance is more important than the required memory.

SIMULATION RESULTS

Simulations using C++ have been extensively performed for rule sets created by Classbench [38]. Three different types of rule sets—access control list (ACL), firewall (FW), and Internet protocol chain (IPC)—are generated with sizes of approximately 1000, 5000, 10 000, 50 000, and 100 000 rules each. Rule sets are named using the set type followed by the size such as with ACL1K, which means an ACL type set with about 1000 rules.

Setting binth:

The HiCuts and HyperCuts algorithms do not have a cut-stop condition other than binth. Setting binth as an arbitrary small number will cause an infinite loop of unnecessary cutting if there is no rule boundary inside the subspace covered by an internal node. The optimization of rule overlapping that removes rules except the highest-priority rule in such nodes can solve this problem. However, no rule should be ignored for multimatch packet classification.

Decision Tree Characteristics

In the second step, we constructed decision trees of BC, SBC, HiCuts, and HyperCuts algorithms setting binth as the identified lower bound value. Cutting is recursively performed and stopped if the number of rules included in a subspace is less than or equal to the lower bound binth. The performances of the HiCuts and HyperCuts algorithms depend on spfac as well. In our simulation, spfac was set to 1.5 for 1K sets and 2 for all other sets.

CONCLUSIONS

In this paper, decision tree algorithms have been studied, and a new decision tree algorithm is proposed. Throughout the extensive simulation using Classbench databases [38] for the previous decision tree algorithms, HiCuts and HyperCuts, we discovered that the performance of decision tree algorithms is highly dependent on the rule set characteristics, especially the number of rules with a wildcard or a short-length prefix. For example, the HiCuts algorithm can provide high-speed search performance, but the memory overhead for large sets or sets with many wildcard rules makes its use impractical. We also discovered that the HyperCuts algorithm either does not provide high-speed search performance or requires a huge amount of memory depending on how to implement the pushing upward optimization. While the cutting in the earlier decision tree algorithms is based on a regular interval, the cutting in the proposed algorithm is based on rule boundaries; hence, the cutting in our proposed algorithm is deterministic and very effective. Furthermore, to avoid rule replication caused by unnecessary cutting, a refined structure of the proposed algorithm has been proposed. The proposed algorithms consume a lot less memory space compared to the earlier decision tree algorithms, and it is upto several kilobytes per rule except for FW50K and FW100K. The proposed algorithms achieve a packet classification by 10–23 on-chip memory accesses and 1.0–4.0 off-chip memory accesses in average. New network applications have recently demanded a multimatch packet classification [4] in which all

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

matching results along with the highest-priority matching rule must be returned. It is necessary to explore efficient algorithms to solve both classification problems. The decision tree algorithms including the proposed algorithms in this paper naturally enable both the highest priority match and the multimatch packet classification.

REFERENCES

[1] H. J. Chao, "Next generation routers," Proc. IEEE, vol. 90, no. 9, pp.1518–1588, Sep. 2002.

[2] A. X. Liu, C.R.Meiners, andE.Torng, "TCAMrazor: A systematic approach towards minimizing packet classifiers in TCAMs," IEEE/ACM Trans. Netw., vol. 18, no. 2, pp. 490–500, Apr. 2010.

[3] C. R. Meiners, A. X. Liu, and E. Torng, "Topological transformation approaches to TCAM-based packet classification," IEEE/ACM Trans. Netw., vol. 19, no. 1, pp. 237–250, Feb. 2011.

[4] F. Yu and T. V. Lakshnam, "Efficient multimatch packet classification and lookup with TCAM," IEEE Micro, vol. 25, no. 1, pp. 50–59, Jan. –Feb. 2005.

[5] F. Yu, T. V. Lakshman, M. A. Motoyama, and R. H. Katz, "Efficient multimatch packet classification for network security applications," IEEE J. Sel. Areas Commun., vol. 24, no. 10, pp. 1805–1816, Oct. 2006.

[6] H. Yu and R. Mahapatra, "A memory-efficient hashing by multi-predicate bloom filters for packet classification," in Proc. IEEE INFOCOM, 2008, pp. 2467–2475.

[7] H. Song and J. W. Lockwood, "Efficient packet classification for network intrusion detection using FPGA," in Proc. ACM SIGDA FPGA, 2005, pp. 238–245.

[8] P. Gupta and N. Mckeown, "Classification using hierarchical intelligent cuttings," IEEE Micro, vol. 20, no. 1, pp. 34–41, Jan.–Feb. 2000.

[9] S. Singh, F. Baboescu, G. Varghese, and J. Wang, "Packet classification using multidimensional cutting," in Proc. SIGCOMM, 2003, pp. 213–224.

[10] P. Gupta and N. Mckeown, "Algorithms for packet classification," IEEE Netw., vol. 15, no. 2, pp. 24–32, Mar.–Apr. 2001.

[11] B. Vamanan, G. Voskuilen, and T. N. Vijaykumar, "EffiCuts: Optimizing packet classification for memory and throughput," in Proc. ACM SIGCOMM, 2010, pp. 207–218.

[12] H. Song, M. Kodialam, F. Hao, and T. V. Lakshman, "Efficient trie braiding in scalable virtual routers," IEEE/ACM Trans. Netw., vol. 20, no. 5, pp. 1489–1500, Oct. 2012.

[13] J. Treurniet, "A network activity classification schema and its application to scan detection," IEEE/ACM Trans. Netw., vol. 19, no. 5, pp. 1396–1404, Oct. 2011.

[14] L. Choi, H. Kim, S. Ki, and M. H. Kim, "Scalable packet classification through rulebase partitioning using the maximum entropy hashing," IEEE/ACM Trans. Netw., vol. 17, no. 6, pp. 1926–1935, Dec. 2009.

[15] F. Baboescu and G. Varghese, "Scalable packet classification," IEEE/ACM Trans. Netw., vol. 13, no. 1, pp. 2–14, Feb. 2005.

[16] P. C. Wang, C. L. Lee, C. T. Chan, and H. Y. Chang, "Performance improvement of two-dimensional packet classification by filter rephrasing," IEEE/ACM Trans. Netw., vol. 15, no. 4, pp. 906–917, Aug. 2007.

[17] P. C. Wang, C. L. Lee, C. T. Chan, and H. Y. Chang, "O(log W) multidimensional packet classification," IEEE/ACM Trans. Netw., vol. 15, no. 2, pp. 462–472, Apr. 2007.

[18] X. Sun, S. K. Sahni, and Y. Q. Zhao, "Packet classification consuming small amount of memory,"

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

IEEE/ACM Trans. Netw., vol. 13, no. 5, pp. 1135–1145, Oct. 2005.

[19] W. Lu and S. Sahni, "Succinct representation of static packet classifiers," IEEE/ACM Trans. Netw., vol. 17, no. 3, pp. 803–816, Jun. 2009.