2007 IEEE International Conference on Signal Processing and Communications (ICSPC 2007), 24-27 November 2007, Dubai, United Arab Emirates

POWER ANALYSIS OF PACKET CLASSIFICATION ON PROGRAMMABLE NETWORK PROCESSORS

Alan Kennedy, David Bermingham, Xiaojun Wang

HDL Lab, School of Electronic Engineering,
Dublin City University, Dublin 9, Ireland
alan.kennedy@eeng.dcu.ie

Bin Liu

Department of Computer Science and Technology,
Tsinghua University, Beijing P.R.China
liub@tsinghua.edu.cn

ABSTRACT

Packet classification algorithms are increasingly being used to provide security and Quality of Service guarantees. These algorithms are usually implemented on power hungry programmable network processors, which are used in devices such as core routers and firewalls. This paper compares the energy used by five best-known algorithms Recursive Flow Classification, HiCuts, HyperCuts, Extended Grid-of-Tries with Path Compression and Tuple Space Search with Pruning. It does this by measuring the energy used to build the search structure during preprocessing for each of the five algorithms and the average energy taken to classify a packet. To do this we implemented all five algorithms in C code and used a microarchitectural power simulation tool called Sim-Panalyzer to estimate the power dissipated by the five algorithms while running on a SA1100 StrongARM RISC processor similar to the type found on many of today's programmable network processors.

Index Terms— Packet Classification, Power Analysis

1. INTRODUCTION

Multidimensional packet classification is increasingly being used by Internet Routers to implement QoS Policies such as guaranteeing minimum bandwidth. It is also used for services such as giving priority to Voice over IP and IP-TV packets, traffic billing based on network usage and the blocking of unwanted Internet traffic. Fast packet classification is essential due to the fact that Internet usage is doubling every six months and backbone link speeds have increased to OC-192 (10 Gb/s) [1].

One of the most popular ways to implement fast packet classification is through the use of hardware such as Ternary CAMs. Due to their parallel comparison TCAMs can use up larger amounts of power than good algorithmic approaches during packet classification. TCAMS also use up large amounts of board area due to poor density of storage cells compared with SRAM or DRAM. This means algorithmic approaches are still a good alternative for packet classification.

Increasing Internet traffic also means that the amount energy being used by networking devices such as network processors is growing rapidly. Greening of the Internet by Gupti and Singh [2] shows that in 2000 the amount of energy used by various networking devices in the U.S accumulated to 6.05 Tera-Watt hours, which is nearly the yearly output of an average nuclear reactor unit.

Gupta and Mckeown carried out an extensive study of classifiers [3] and found only 0.7% of the classifiers they examined contained over 1000 rules and none contained more than 2000 rules. With these points in mind we closely examined many packet classification algorithms [1, 3-10] and implemented the five mentioned to discover which algorithms offer the best combination of high speed classification, low memory usage, scalability to large rulesets and low power consumption for five field rulesets. To do this we tested the algorithms with rulesets containing between 60 and 2193 rules. Some of the algorithms tested allow incremental updates to the rulesets. This means rules can be deleted or added to the rulelist without the need to rebuild the search structure.

The layout of the rest of this paper is as follows. Section two will give a brief explanation of the five algorithms being tested. Section 3 gives an explanation of the microarchitectural power simulation tool Sim-Panalyzer used to estimate the power consumption of the algorithms. Section 4 presents the results and section 5 gives a summary of the results and states any conclusions.

2. PACKET CLASSIFICATION ALGORITHMS

2.1 Recursive Flow Classification (RFC)

RFC by Gupta and McKeon [3] is a decomposition-based algorithm, which classifies packets at high lookup speeds, at the cost of long preprocessing time, high memory consumption and the inability to allow incremental ruleset updates. The RFC algorithm works by first breaking the F fields of the packet header into multiple chunks. This usually means breaking the source IP and destination IP addresses into 16-bit chunks.

In the first of P phases each of the header chunks are used as an index to access a direct lookup memory location containing a preprocessed *eqID*, which will represent and be smaller than the index used to access the memory location. The index for performing a direct

lookup on preprocessed tables for the next phase is formed from the *eqIDs* from the previous phases. There will be one lookup in the final phase, with the result corresponding to the matching rule. This is possible because of the way the lookup tables are constructed.

2.2 Hierarchical Intelligent Cuttings (HiCuts)

HiCuts by Gupta and McKeown [4] is a decision based tree algorithm, which allows incremental updates to a ruleset. It takes a geometric view of packet classification by considering each rule in a ruleset as a hypercube in hyperspace defined by the F fields of a packets header. The algorithm constructs the decision tree by recursively cutting the hyperspace one dimension at a time into sub regions guided by heuristics that exploit the structure of the classifier. These sub regions will contain the rules whose hypercube overlap. Each cut along a dimension will increase the number of sub regions with each sub region containing fewer rules.

A predetermined space measure function *(spmf)* limits the amount of cuts allowed to a dimension at each step. Cutting is finished when all sub regions (leaf nodes) contain fewer rules than a pre determined value called *binth*. The lookup algorithm traverses the decision tree based on the values of a packet's header by copying the cutting sequence until it finds a leaf node. It then performs a linear search of the rules contained inside this leaf node.

2.3 HyperCuts

HyperCuts by Singh et al [5] is similar to HiCuts in that it is a decision tree based algorithm, which takes a geometric view of packet classification and allows incremental updates. The main differences from HiCuts are that HyperCuts recursively cuts the hyperspace into sub regions by performing cuts on multiple dimensions at a time. HyperCuts also takes advantage of extra heuristics, which exploit the structure of the classifier such as region compaction and pushing common rule subsets upwards.

Region compaction allows for more efficient cutting of a dimension as it only cuts the region covered by the rules rather than the full region. Pushing common rule subsets upwards will reduce the replicated storage of rules. The lookup algorithm is similar to that of HiCuts. It traverses the decision tree based on values of a packet's header by copying the cutting sequence until it finds a leaf node followed by a small linear search of its rules.

2.4 Extended Grid-of-Tries with Path Compression (EGT-PC)

EGT-PC by Baboescu et al [6] is a decision tree based algorithm, which allows incremental updates. In EGT-PC a path compressed trie is first created from the prefixes in the ruleset's first dimension. Each node in this trie, which represents a valid prefix P in the first dimension, will contain a pointer to another path compressed trie made up of all the prefixes from the second dimension whose first dimension prefixes are equal to P. Each node in the second dimension trie corresponding to a valid prefix in this dimension will contain a list of all the rules, which match the prefixes of the first and second dimension nodes. This means a rule can only occur in one position. In order to avoid back tracking all failure points in the second dimension tries contain a jump pointer, which points to the next, possible second dimension trie, which could contain a matching rule.

The search algorithm works by first performing a longest prefix match (LPM) on the first dimension trie. The resulting pointer is then followed to a second dimension trie. A LPM is then carried out on this trie to find nodes containing matching rules. Each time there is a failure or the end of a second dimension trie is reached a jump pointer is followed. This is continued until a node is reached which contains no jump pointer. All matching rules along the way are recorded and a small linear search of these rules is carried out at the end.

2.5 Tuple Space Search with Tuple Pruning (TSS)

TSS by Srinivasan et al [7] is a hash-based algorithm, which supports incremental updates. All filters are divided into groups called tuples. All rules that map to a particular tuple have the same prefix length for the source and destination IP addresses. The port number fields will either be a wildcard or the same nesting level inside the port range. Protocol values will either be wildcard or a specific value. Each port address inside a tuple will have a *RangeId* depending on its position inside its nesting level. A packet's port number is usually converted to its *RangeId* using a 65KB direct lookup table.

The required number of bits from a filters tuple specification e.g. (24, 24, 0, 0, 8) is used to form a hash key for that filter. All filters belonging to tuple T are stored in *Hashtable(T)*. A probe of a tuple T involves using the required bits from the tuple specification to construct a hash key from a packets header. This means only one memory access is needed for each tuple to determine if it contains a matched filter. The algorithm is motivated by the fact that a linear search through all tuples will be smaller than a search through all rules.

The number of tuples, which need to be searched, is further reduced through tuple pruning, which involves performing a LPM usually on the source and destination IP addresses. The LPM finds the lengths of the matched source and destination IP address prefixes so only a subset of tuple groups need to be searched.

3. POWER ESTIMATION

To estimate the energy used during the building of the search structures and packet classification for the algorithms running on a SA1100 StrongARM, we used Sim-Panalyzer [11]. Sim-Panalyzer is an infrastructure for microarchitectural power simulation implemented on top of "sim-outorder" a component within the SimpleScalar simulator. It simulates the execution at the level of

individual cycles keeping track of power changes across cycles. It consists of several distinct components: cache power models; datapath and execution unit power models; clock tree power models; and I/O power models. The processor was configured to run at 200 Mhz at 1.8 V using 0.18μm technology. ARM binaries were produced using an ARM-Linux cross compiler with optimisation level set at –O2.

4. EXPERIMENTAL RESULTS

4.1 Rulesets

The code we wrote for all five algorithms was tested extensively using the ACL1, FW1 and IPC1 rulesets obtained from [12]. They are a mixture of both real filter sets and synthetic filter sets generated using ClassBench. The results presented were generated using the ACL1 rulesets with similar results obtained using FW1 and IPC1.

4.2 Results

The first results presented are the widely analyzed fields of memory consumption and worst-case lookups (WCLU) [5, 6, 12]. Our memory consumption results match these closely and our WCLU results for HiCuts and HyperCuts are slightly higher. This could be due to the fact we included all memory lookups needed to traverse the decision tree and a memory lookup for each rule in the linear search. We then present our new research, which shows the energy used by the algorithms during the building of the search structure and classification of the packets along with the throughput. All of which measured while running on a SA1100 StrongARM processor similar to the type found on programmable network processors.

It can be seen from looking at figure 1 that RFC is the worst performer in terms of memory consumption needing over 3MB of memory for 2191 rules. HyperCuts performed best over the full range of rules needing only 56 KB of memory for 2191 rules and 1.7 KB for 60 rules. EGT-PC and HiCuts also performed well matching HyperCuts closely. The memory consumption was high for TSS due to the 65 KB direct lookup and hash tables.

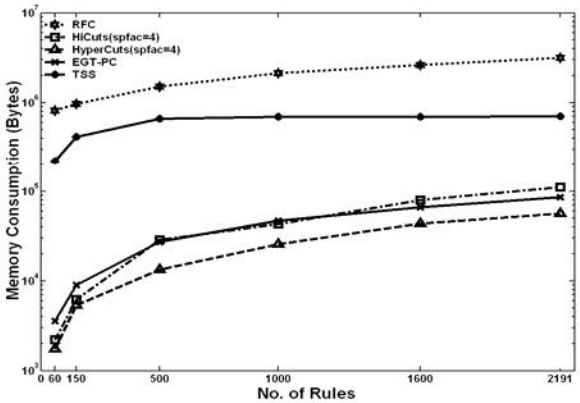

Figure 1. Memory needed for the search structure and Ruleset.

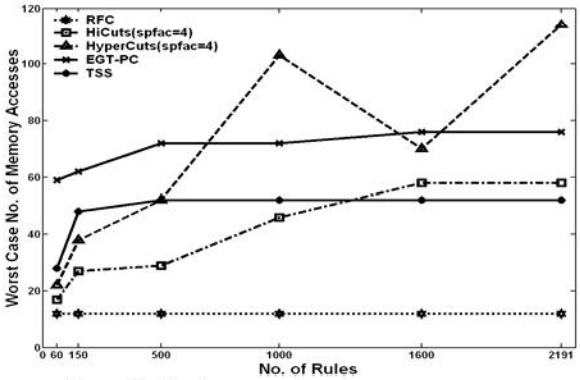

Figure 2. Worst case number of memory accesses.

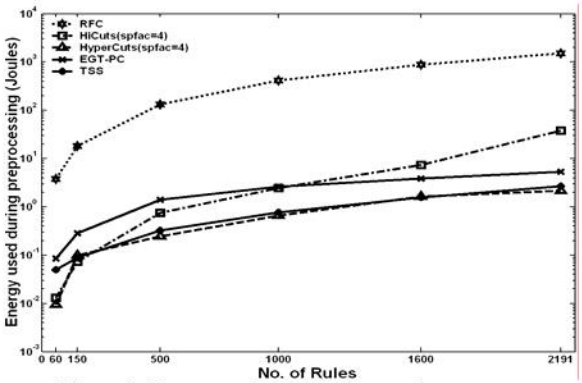

Figure 3. Energy used building the search structure.

Figure 2 shows that for the WCLU metric RFC is by far the best performer needing only 12 memory lookups for all rulesets. The algorithm with the largest WCLU figure for rulesets with more than 1000 rules was HyperCuts. It was outperformed by HiCuts due to the fact it needed to access extra information when traversing the decision tree. This information includes multiple dimensions, which may need to be cut, and minimum and maximum range values for these dimensions when using the Region Compaction heuristic. The TSS levelled out at a WCLU of 52 after 500 rules due to the fact the number of tuples never increased and the LPM trees never got deeper. The chances of hash collisions and a packet being misclassified increased significantly as the ruleset got larger than 2191 rules.

Figure 3 shows the energy used when building the search structure. For algorithms allowing incremental updates this will not be a significant percentage of the energy used during packet classification. It will however be a significant part of the energy used for algorithms such as RFC, which don't allow incremental updates, if the ruleset is updated regularly. The energy used building the search structure is proportional to the memory consumption for these algorithms. This indicates an algorithm with low memory consumption will use low amounts of energy building the search structure.

The RFC algorithm performs very poorly when compared to the other algorithms. When the ruleset contains 2191 rules the RFC algorithm needs 1512 Joules of energy to build the search structure compared with

HyperCuts needing only 2.7 Joules. Along with HyperCuts the EGT-PC and TSS algorithms also scale well when it comes to building the search structure. HiCuts performs slightly poorer using 37.9 Joules of energy to build the search structure for 2191 rules. This should not be a problem however as HiCuts supports fast incremental updates to the ruleset meaning the search structure will not have to be rebuilt regularly.

The important metric of the average energy used to classify a packet can be seen in figure 4. For the algorithms that support incremental updates this graph will represent the majority of the energy used during packet classification. From the graph it is clear to see that RFC performs best in this category needing on average only 1.46 μJ to classify a packet due to the simplicity of its lookup algorithm. The worst performing algorithm is EGT-PC needing on average 76.57 μJ followed by TSS 53.25 μJ, HyperCuts 19.2 μJ and Hicuts 10.89 μJ. All five algorithms scale well across the full range of rules.

Figure 5 shows that the throughput for all five algorithms is directly proportional to the average energy needed to classify a packet. This is good news as it means algorithms with faster classification rates operating on RISC type processors will have low power consumption. The highest average number of packets classified per second across all rulesets was 400937 for RFC followed by HiCuts 57042, HyperCuts 32242, TSS 10700 and EGT-PC with 7491. EGT-PC performed poorly due to the fact the average amount of memory lookups for each packet was close to its worst case. TSS performed poorly as it had to create a hash key for each hash table lookup.

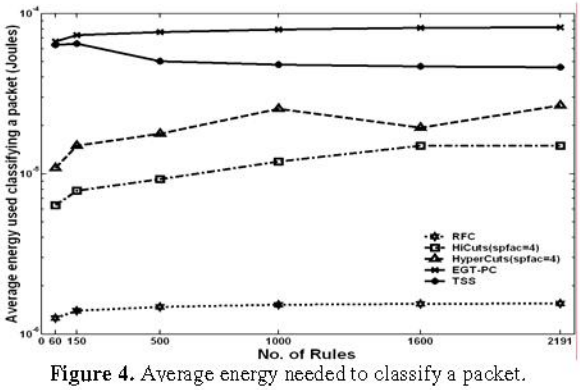

Figure 4. Average energy needed to classify a packet.

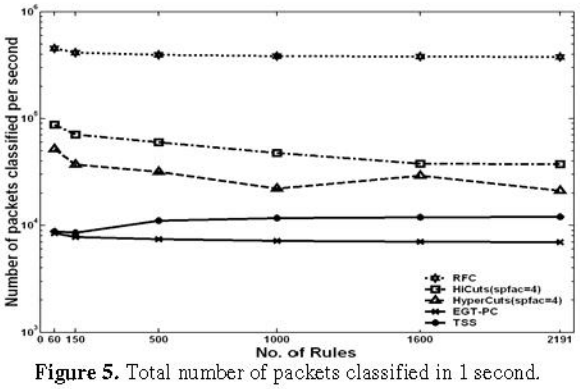

Figure 5. Total number of packets classified in 1 second.

5. CONCLUSIONS

With the results presented we have given as fair a comparison as possible of the five algorithms. Our results show that the best performing algorithm for small rulesets that remain the same for more than several minutes a time is RFC due to its low power consumption when classifying packets and large throughput.

The best performing algorithm for rulesets we examined up to 2191 rules in terms of both energy efficiency and throughput is HiCuts. It out-performs RFC due to the fact it supports incremental updates while RFC needs to spend large amounts of time and energy rebuilding a search structure every time a rule is added or deleted from the ruleset. HiCuts also scales better in terms of memory with its search structure consuming 111 KB for 2191 rules, which could easily fit on the onboard cache of today's processors. RFC on the other hand needs 3.2 MB of memory for the same amount of rules.

HyperCuts is a strong contender for rulesets greater than 2191 rules as it is the algorithm which scaled best in terms of memory consumption needing only 56 KB of memory for 2191 rules. Its throughput and average energy needed to classify a rule also matched HiCuts closely.

6. AKNOWLEDGMENTS

This work was funded by the Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan.

7. REFERENCES

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

[2] M. Gupta and S. Singh, "Greening of the Internet" in *ACM SIGCOM 2003*, pp. 19-26.

[3] P. Gupta and N. McKeown, "Packet classification on multiple fields," in *ACM SIGCOMM 1999*, pp.147-160

[4] P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," *IEEE Micro*, vol.20, no. 1, pp. 34-41, 2000.

[5] S. Singh, F. Baboescu, G. Varghese and J. Wang, "Packet Classification Using Multidimensional Cutting" in *ACM SIGCOMM*, 2003, pp.213-214

[6] F. Baboescu, S. Singh, and G. Varghese, "Packet classification for core routers: Is there an alternative to CAMs?" in *IEEE INFOCOM*, 2003, pp. 53-63.

[7] V. Srinivasan, S. Suri, and G. Varghese, "Packet Classification using Tuple Space Search" in *ACM SIGCOMM 1999*, pp. 135-146.

[8] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE Network Mag.*, vol. 15, no. 2, pp.24-32, 2001

[9] T. Woo, "A modular approach to packet classification: algorithms and results," in IEEE INFOCOM, Mar. 2000, pp. 1213-1222.

[10] P. C. Wang, C. T. Chan, C. L. Lee and H. Y. Chang "Scalable Packet Classification for Enabling Internet Differentiated Services" *IEEE Trans. on Multimedia*, vol. 8, no. 6, pp. 1239-1249, 2006.

[11] Sim-Panalyzer, The SimpleScalar-ARM Power Modeling Project. [Online]. Available: www.eecs.umich.edu/~panalyzer/

[12] ACL1 RuleSets and Packet traces [Online]. Available: www.arl.wustl.edu/~hs1/PClassEval.html