**Proceedings of the 20th IASTED International Conference**
**Parallel and Distributed Computing and Systems (PDCS 2008)**
**November 16 - 18, 2008 Orlando, Florida, USA**

# AHSM: ADAPTIVE PACKET FILTERING WITH NETWORK TRAFFIC STATISTICS

Bo Xu[1, 2], Guangyu Zhou[1], Yibo Xue[2, 3], Jun Li[2, 3]

[1]Department of Automation, Tsinghua University, Beijing, China
[2]Research Institute of Information Technology (RIIT), Tsinghua University, Beijing, China
[3]Tsinghua National Lab for Information Science and Technology, Beijing, China

**ABSTRACT**
Packet filtering plays an important role in network devices such as firewalls, routers, security gateways and intrusion detection systems. Numerous schemes have been proposed to improve packet filtering techniques. Many previous works struggled to utilize the characteristics of filtering rule-sets as optimization heuristics. However, there are rarely efforts excavating network traffic characteristics. This paper focuses on analyzing the statistical characteristics of network traffic and imposing it to optimize packet filtering algorithms. Contribution of the paper includes two aspects: first, the skewness and time correlation in real-life traffic is presented to illustrate network traffic statistics; second, an adaptive packet filtering algorithm, AHSM, is proposed based on HSM algorithm for improving average packet filtering speed. AHSM takes traffic statistics as heuristics and constructs statistical search trees for single field searching. Experimental results show that the optimized algorithm reduces 20%~50% of the single field matching overhead compared with the HSM algorithm and improves the overall performance by 35%~45%, while retaining the same memory usage.

**KEY WORDS**
Packet filtering, traffic statistics, binary trees

## 1. Introduction

Packet filtering plays a critical role in Internet services including access control, policy-based routing, service differentiation and load balancing. With the rapid increasing of network bandwidth, packet filtering is required to cope with the link speed upgrade of Internet backbone devices. However, the packet filtering problem is inherently difficult to solve from a theoretical point of view. It is proved that the computational complexity bounds for searching $N$ rules with $F$ fields is $O(\log N)$ best in time but $O(N^F)$ in space, or $O(N)$ best in space with $O(\log^{F-1} N)$ in time [1]. Thereby, it is easy to perform packet filtering at high speed with large memory occupation or at low speed with small memory occupation. It is rather difficult to achieve high filtering speed with small memory occupation. Researchers have been working hard to find out modest trade-off between time and space. Many previous works have been done to utilize the intrinsic rule-set characteristics as optimization heuristics.

This paper proposes a novel algorithm that improves packet filtering speed with modest memory requirement. The new packet filtering scheme adopts the network traffic statistics as heuristics in building up statistical search trees to reduce rule fields searching time. Besides, the proposed AHSM algorithm intersects the field searching results with recursive space mapping tables based on the ideas of the Hierarchical Space Mapping (HSM) algorithm [2], which was proposed by Xu, Jiang and Li in 2005. Main contributions of this paper include:

- ***Novel Ideas:*** This paper excavates the network traffic statistical heuristics to improve real time packet filtering speed. The authors acquire network traffic from real-life circumstances and carefully analyze the network statistical characteristics including field distribution skewness and time correlation. The statistical characteristics are then employed to construct statistical search trees to improve single filed searching speed.
- ***New Algorithm:*** The proposed AHSM algorithm engages the alphabetic search trees in each packet header field to improve single field searching speed. It then combines the field matching results with recursive intersecting tables, which have better time performance than trees. As a result, AHSM improves filtering speed by optimized binary trees and recursive intersecting tables while preserves memory usage at a modest level.

## 2. Previous Work

In recent years, the packet filtering problem has been studied intensively. The original approach for packet filtering is to search the sequential rules linearly until a match is found. Though this approach consumes very little memory, its performance is quite challenged and not scalable since the search time is proportional to the rule-set size. Many algorithms have been proposed recently and they can be classified into three categories: hardware-based solutions, geometric algorithms and heuristic-based algorithms.

Hardware-based solutions mainly utilize the parallelism of Content Addressable Memory (CAM) to perform rule matching in parallel. Zheng et al. proposed a TCAM-based IP lookup algorithm using prefix match in 2004 and then

further extended the filtering engine to support range match using the TCAM wildcard. [5] It is claimed that both the solutions can reach the filtering requirement of OC-192 line speed. However, the solutions cannot afford middle to large rule-set because of the high cost, power consumption and size limitations of CAMs.

A variety of geometric algorithms have been proposed for packet filtering. Trie based schemes, including Grid-of-tries [6], Hierarchical tries [7] and Set-pruning tries [6], all construct searching tries based on the geometric layout of the search space. Bit vector algorithms such as BV [8] and ABV [8] also use trie lookup on each header field, followed by a combining phase of bit vectors, which indicates the potential matching rules.

Heuristic-based algorithms are introduced by Gupta and McKeown [7] and Woo [9]. Hierarchical Intelligent Cuttings (HiCuts) [10] builds a decision tree based on the structural characteristics of the rule-set and uses optimization decisions to decide the next dimension to cut at each node.

Gupta and Mckeown proposed another heuristic scheme call Recursive Flow Classification (RFC) [11] in 1999. RFC simplifies packet filtering by reducing structural redundancy in the rule matching. RFC needs only 9 memory accesses for one packet filtering and it can be pipelined due to its structure. However, RFC does not scale well with medium or large rule-sets because of exponential increase of memory demand.

HSM improved RFC by using binary trees to replace the first phase indexing tables. As a result, the improved algorithm remarkably reduces the memory requirement with the expense of slightly increased searching time.

Although the previous work contribute significantly to packet filtering research, they mainly design algorithms based on the geometric layout or structural characteristics of the rule-set. Therefore, they have not exploited the network statistical characteristics to improve the average packet filtering speed.

The related work closest to our approach is Dynamic Cuttings (D-Cuts) [3] and the paper by Hamed et al [4]. D-Cuts adopts network statistics into decision trees and thus achieves higher average filtering speed than HiCuts. The work by Hamed builds alphabetic trees on each field and proposes two methods to form the final statistical matching tree. However, both the schemes suffer from the relative long term tree searching, compared with table-based space mapping algorithms such as RFC and HSM. Our work inherits the hierarchical space mapping tables from HSM and thus transfers the tree combinations into fast table indexing.

## 3. Network Traffic Statistics

This section analyzes the statistical characteristics of the traffic passing through firewalls. The traffic analysis is performed on real-life traces captured at the edge firewall of a research institute in Tsinghua University. Furthermore, we tick out the beginning packets of the flows to generate the testing traces, as in today's stateful firewalls, only the first packet of each flow is processed by the packet filtering module, while the subsequent packets are processed in the stateful inspection module, or session filtering module. The traces of Friday and Sunday are taken to represent the realistic network conditions of working day and weekend. Each trace contains the header information of 15M to 20M packets, and correspondingly 3M to 5M SYN packets.

The observed statistical characteristics lie in two aspects: (1) the traffic skewness in each header field; (2) the time correlation of field value. The single field skewness motivates us to employ statistical search trees to accelerate single field searching. Besides, the time correlation gives us the idea that how long the skewness will last and thus choose a proper update frequency. The two statistics are illustrated in the following.

### 3.1. Traffic Skewness in Packet Filtering

Previous studies [4] have shown that the majority of Internet flows have short flow sizes in terms of number of packets, while most of the traffic is contributed by the long flows. In packet filtering problem, we focus on the distribution of the beginning packets of flows, since today's firewalls use sessions to handle the subsequent packets of flows. Based on our observation, we argue that the scattering of the beginning packets also show considerable degree of skewness, which motivates the engagement of statistical heuristics in packet filtering.

The field value frequency distribution is said to be skewed if the distribution is non-uniform. It means part of the values have higher hit frequencies while others have lower. To evaluate the degree of skewness, we use the entropy formulation in information theory [12]. The entropy returns a value between 0 and 1, where 0 corresponds to uniform distribution and 1 corresponds to a completely skewed distribution. Higher entropy refers to higher skewness in the frequency distribution. The skewness factor of a header field $f$ is denoted as $S_f$ defined by Formula 1, where $p_i$ is the probability of field value $v_i$ and $n$ is the number of possible values of field $f$.

$$S_f = 1 - \frac{\sum_{i=1}^{n} p_i \lg p_i}{\lg n} \qquad (1)$$

The skewness of frequency distribution is calculated for all field values at different sampling time intervals. Since only the beginning packets of flows are concerned here, the time intervals are measured in number of packets instead of time in seconds.

Figure 1-(a) and Figure 1-(b) show the skewness of field value frequency distribution of traffic passing through the real-life firewall. Figure 1-(a) gives the skewness of the four header fields on Friday and Figure 1-(b) gives that of Sunday. It is shown in the two figures that the skewness factor of the source address ranges from 0.55 to 0.85, while the skewness factor of the destination address ranges from 0.15 to 0.7. It also shows that the source port has a skewness factor of 0.03-0.5, while the destination port has a skewness factor of 0.15-0.7.

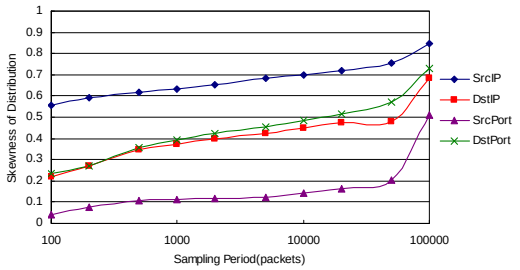


**Figure 1-(a). Skewness of Distribution on Friday**

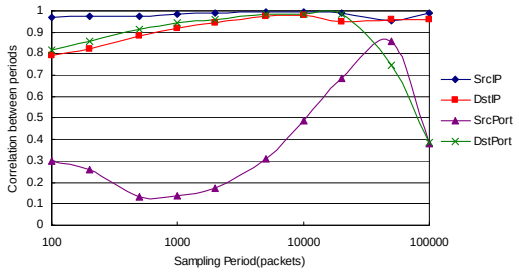


**Figure 2-(a). Time Correlation on Friday**

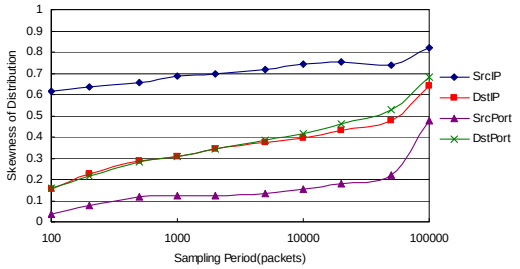


**Figure 1-(b). Skewness of Distribution on Sunday**

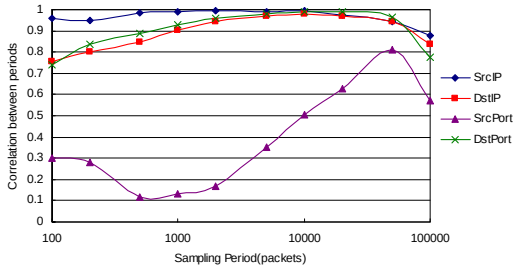


**Figure 2-(b). Time Correlation on Sunday**

It is observed from Figure 1 that the source address have the highest skewness, and the destination address along with the destination port have moderate skewness, while the source port has the lowest skewness. Figure 1 also shows that the skewness of all fields grows remarkably with the increasing of sampling period. Moreover, the skewness on Friday is similar to that of Sunday, which provides compelling evidence that the skewness in network traffic constantly exists, no matter it is working day or weekend.

The four head fields have skewness larger than 0.40 when the sampling period is longer than 10,000 packets, except the source port field. Thus, it will be rather appealing if we can use the skewness to optimize the single field searching tree structures.

### 3.2. Time Correlation of Field Frequencies

To measure how long the skewness will last, we investigate the correlation of the frequency distribution of packet header fields over two consecutive time intervals. If the field frequency distribution is similar between consecutive intervals, then is will be said to be time-correlated over the two intervals. The correlation factor of field $f$ is calculated as Formula 2 [4], where $p_i$ is the probability of $v_i$ in a certain time interval, and $q_i$ is the probability in the following interval. $\mu_p$ and $\mu_q$ represent the mean value of the two probability distributions while $\sigma_p$ and $\sigma_q$ represent their standard deviation. The correlation factor $C_f$ returns a value between 0 and 1, where 0 indicates an uncorrelated distribution and 1 indicates a completely correlated distribution.

$$C_f = \frac{\sum_{i=1}^{n}(p_i - \mu_p)(q_i - \mu_q)}{n \cdot \sigma_p \cdot \sigma_q} \qquad (2)$$

Figure 2-(a) and Figure 2-(b) show the time correlation of the header fields over consecutive time intervals. Figure 2-(a) illustrates the time correlation of the four header fields on Friday and Figure 2-(b) gives that of Sunday. It is shown that the source address field has the highest time correlation factor ranging from 0.9 to 1. The destination address field has moderate time correlation factor ranging from 0.75 to 0.95. The destination port field has a moderate time correlation factor from 0.7 to 0.95 when the sampling period is smaller than 20,000 packets. However, the correlation factor of destination port deceases rapidly if the sampling period exceeds 20,000 packets. Besides, the source port field has the lowest time correlation factor, which grows smoothly when the sampling period is smaller than 50,000 packets but decreases rapidly if the sampling period exceeds this value. This is because that the ports are likely to be reused if the sampling period is too long.

It is also observed in Figure 2 that the correlation factors of the source and destination address fields tend to increase slowly with the increasing of sampling period. But it does not mean that the longer sampling period is always better, since the time correlation will decrease when the sampling period exceeds certain threshold. The correlation reduction with sampling periods larger than 50,000 in Figure 2-(b) demonstrates this affirmation, which is consistent with intuitive opinions. We argue that this kind of decease in correlation factor is due to the inherent rhythm of the network traffic. The characteristics of the traffic determine an intrinsic time correlation frequency, longer or shorter sampling periods will both cause the decline in time correlation. Moreover, Figure 2 shows that the time correlation of the four header fields on Friday is similar to that of Sunday, while it may have different inflexions of time correlation.

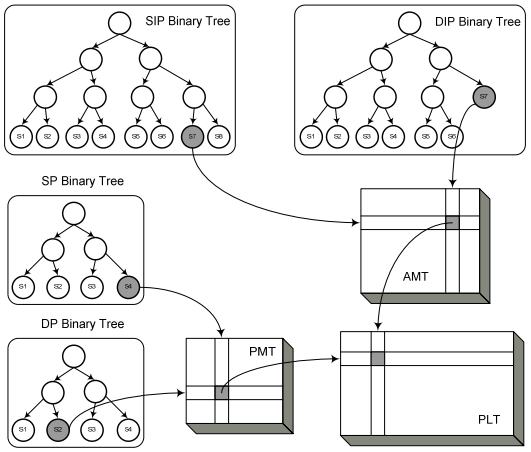


**Figure 3. HSM Data Structure**

**Table 1. Example Statistics of Destination Port Filed**

| Segments | Value | Statistics |
|---|---|---|
| S1 | 0-19 | 0.01 |
| S2 | 20-21 | 0.01 |
| S3 | 22-52 | 0.02 |
| S4 | 53 | 0.10 |
| S5 | 54-79 | 0.12 |
| S6 | 80-88 | 0.60 |
| S7 | 89-65535 | 0.14 |

Based on these observations, we can see that the source address, destination address and destination port fields all have high and stable time correlation factors. Nevertheless, the source port field has a wide time correlation range that varies according to the sampling period. However, the time correlation factor of the source port could reach a high value if we can pick a suitable sampling period.

As a result, according to the network traffic statistics observed from the real-life traces, it can be concluded that the four header fields, including the source address, the destination address, the source port, and the destination port, do have considerable skewness in their frequency distributions and the distributions last for consecutive time intervals with high correlation. These statistical characteristics significantly motivate our work described in the following section.

# 4. Adaptive Filtering Algorithm

This section proposes an adaptive packet filtering algorithm called AHSM, which utilizes the network traffic statistical characteristics for improving the average filtering speed in real-life circumstances. The basic ideas are based on the algorithm HSM.

## 4.1. Basic Ideas

HSM performs binary search at each header field using balanced binary trees, and then searches the hierarchical space mapping tables for possible combinations of results from single field binary searches. Figure 3 shows the data structure of HSM, where the four header fields (source address, destination address, source port and destination port) are searched via balanced binary trees and after that Address Mapping Table (AMT) and Port Mapping Table (PMT) are indexed respectively by the combining the searching results of the two address fields and the two port fields. At last, Policy Lookup Table (PLT) is indexed by combining AMT and PMT lookup results.

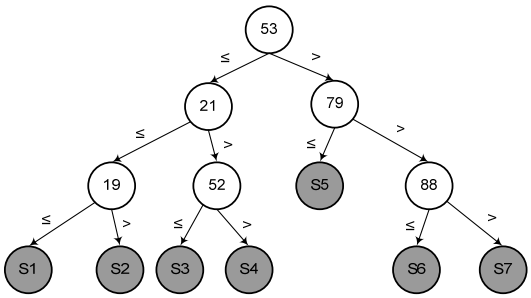


**Figure 4-(a). Binary Search Tree for Table 1**

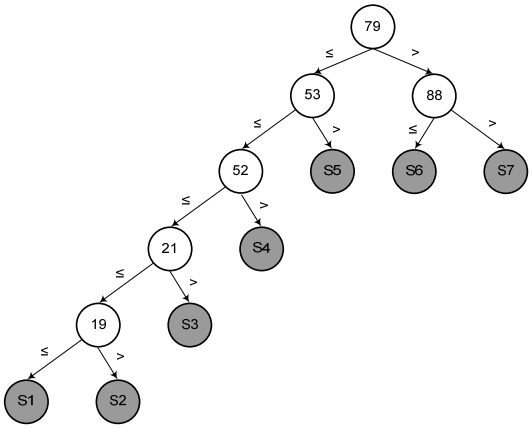


**Figure 4-(b). Alphabetic Search Tree for Table 1**

Although HSM is superior to RFC in space complexity by using binary trees to replace the indexing tables of RFC in the first phase of searching, it slightly degrades filtering speed due to the time-consuming single field binary searches. If the number of rules is $N$, the time for $f$ field binary searches will be $f \log_2 2N$, while the search in the subsequent hierarchical space mapping tables only takes 3 memory accesses [2]. Thus, aiming at improving packet filtering speed of HSM, we must reduce the single filed search time. Fortunately, based on the analysis in Section 3, we believe that the average searching speed can be accelerated by employing statistical heuristics.

## 4.2. AHSM Algorithm

The proposed AHSM algorithm is an optimization of HSM algorithm. Its basic idea is to use the alphabetic search trees to replace the balanced binary trees in HSM to improve the single field search speed, while retaining the subsequent hierarchical space mapping tables. This section analyzes the skewness in header filed segments and constructs the alphabetic trees for single filed searching. At last, it discusses the reconstruction and update of the alphabetic trees.

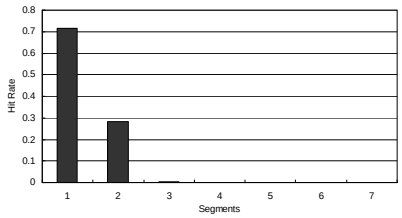


**Figure 6-(a). Source Port Field Hit Rates**

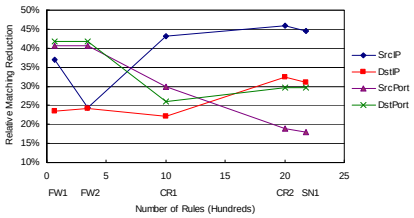


**Figure 7-(a). Relative Reduction Rate (a)**

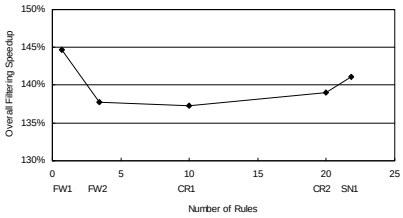


**Figure 8-(a). Overall Filtering Speedup (a)**

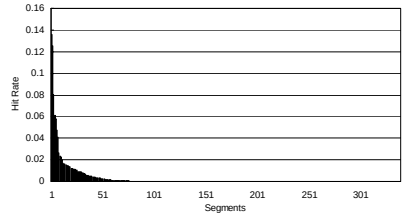


**Figure 6-(a). Destination Port Field Hit Rates**

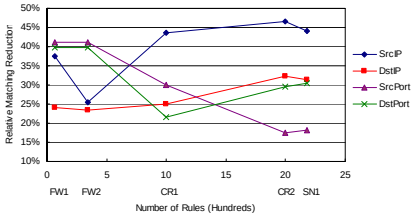


**Figure 7-(b). Relative Reduction Rate (b)**

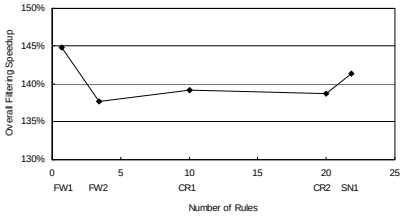


**Figure 8-(b). Overall Filtering Speedup (b)**

### 4.2.1. Skewness in Header Field Segments

It has been observed in Section 3 that the network traffic has certain degree of skewness in the header field value frequencies and the skewness will last for certain period of time. Apparently, this kind of skewness will also cause the skewness in the header field segments generated by the rules. To demonstrate this deduction, we choose one real-life rule-set with 2,180 rules and take 10,000 packets from the firewall traffic to observe the data locality. The rule-set splits the source address sub-space into 840 segments, the destination address sub-space 959 segments, the source port sub-space 7 segments and the destination port sub-space 339 segments. Figure 6-(a) to (b) shows the segment hit rates of the source port and destination port header fields in a decreasing order, where it can be seen that the majority of the hits belong to a small part of the segments. This kind of skewness makes it possible to construct statistical search trees for the header field searches.

### 4.2.2. AHSM Matching Phases

AHSM algorithm engages similar matching phases with HSM algorithm. As shown in Figure 3, HSM takes four balanced binary tree searches in Phase 1, and then perform table indexing in AMT, PMT and PLT. AHSM improves HSM by replacing the balanced binary trees with statistical search trees, while keeping the hierarchical space mapping tables.

*Alphabetic search trees:* There are several types of statistical search trees that can be adopted to minimize the weighted tree depth. AHSM chooses the Alphabetic Search Trees [13] mainly because it has lower tree construction complexity when compared with the Optimal Binary Search Trees and has less searching overhead when compared with Huffman Trees [14]. The alphabetic search tree stores the segments in leaf nodes and keeps the ordering of the parent and child nodes. The best time complexity for constructing alphabetic search trees is $O(n \log n)$, achieved by Hu-Tucker [13].

Assuming we have the segments of destination port field with statistical probabilities as shown in Table 1, the normal balanced binary search tree will be like Figure 4-(a), while the alphabetic search tree can be illustrated as Figure 4-(b). Considering the corresponding statistics of the segments, the average depth of balanced binary tree is 3.88 while the average depth of the alphabetic search tree is 3.2. The average number of matching is reduced by 18% compared with the balanced binary tree.

*Hierarchical space mapping tables:* In phase 2 and phase 3, AHSM adopts the recursive mapping tables inherited from HSM. The space mapping tables are generated with assistance of bitmaps, which records the serial numbers of the potential matching rules. Consequently, after locating the header segments with the alphabetic trees in phase 1, AHSM only needs 3 additional memory accesses to find out the final match.

## 5. Performance Evaluation

To evaluate the performance of the alphabetic tree filtering technique, we use four real-life rule-sets and one synthesized rule-set. *FW1* and *FW2* are access control lists (ACLs) obtained from edge firewalls in Tsinghua University and they contain 69 and 341 rules respectively. *CR1* and *CR2* are filtering sets provided by typical enterprise networks and they contain 1001 and 2000 rules respectively. *SN1* is a synthesized rule-set according to the characteristics of real-life policies, which has 2180 rules. The testing trace is the Sunday trace mentioned in Section 3.

### 5.1 Performance of Statistical Trees

To evaluate the performance of the alphabetic trees, we conduct experiments with the five filtering rule-sets on a real-life trace. Figure 7-(a) shows the relative matching time reduction of the four header fields compared with the balanced binary search trees. The statistical search trees are using update time interval of 10,000 packets and the results are the average testing values of 10 times. From Figure 7-(a), it can be seen that the source address field obtains a relative matching reduction of 25%~45%, and the

**Table 2. Field Skewness and Relative Reduction**

| Fields | SrcIP | DstIP | SrcPort | DstPort |
|---|---|---|---|---|
| Skewness | 74.1% | 39.7% | 15.4% | 41.6% |
| CR2 Reduction | 45.8% | 32.4% | 18.9% | 29.6% |
| SN1 Reduction | 44.4% | 30.9% | 18.1% | 29.7% |

destination address field obtains a relative reduction of 22%~32%. The source port gets a relative matching reduction of 18%~40% and the destination port gets a relative reduction of 25%~40%. Figure 7-(a) also shows that the reduction varies a lot with different rule-sets and it shows that the addresses get higher reduction rates with *CR1* and *CR2* and the ports get higher reduction rates with *FW1* and *FW2*. This should be attributed to the characteristics of the rule-sets, which determine the number of segments in each field. Normally, for the same field, larger number of segments is likely to obtain higher reduction rate. For example, *FW1* and *FW2* have 65 and 275 segments in source address while *CR2* and *SN1* have 2661 and 840 segments. Meanwhile, *FW1* and *FW2* both have 39 segments in source port while *CR2* and *SN1* have 7 and 10 segments.

Moreover, if we pay attention ot the field value skewness factors along with the relative matching reductions with large rule-sets, we will get the data in Table 2, which shows certain degree of correlation between the skewness and reduction rates. However, the small rule-sets such as *FW1*, *FW2*, and *CR1* do not exhibit this kind of correlation.

Figure 7-(b) shows the results with update time interval of 100,000 packets and the curves are just similar with Figure 7-(a).

### 5.2 Overall Filtering Speedup

We further evaluate the overall filtering speedup by introducing statistical trees, compared with the HSM algorithm. Since both HSM and AHSM algorithms only take 3 times of hierarchical table indexing after the first phase tree searches, the overall filtering performance is ultimately determined by the single field searching times. Through experiments on the rule-sets along with the real-life trace, we get the performance speedup illustrated in Figure 8-(a) and (b). Figure 8-(a) uses the statistical tree update interval of 10,000 packets and Figure 8-(b) uses the statistical tree update interval of 100,000 packets. It is shown that the alphabetic trees achieved an overall performance speedup of 35%~45% and the curves with different update intervals are very close. It is believed that this degree of filtering speedup is attractive in practical network circumstances.

## 6. Conclusion and Future Work

In this paper, we focus on the statistical characteristics of network traffic and their impact in performance improvement of packet filtering. The skewness and time correlation in network traffic is observed by analyzing two real-life traces and it is shown that the four-tuple header fields do have considerable skewness in their value frequency distribution and the distribution will last for consecutive time intervals with high correlation. Based on the observations, we proposed the AHSM algorithm which employs the alphabetic search trees in single field searches and inherits hierarchical space mapping tables from the HSM algorithm.

Experimental results show that the optimized statistical trees reduce 20%~50% of the single filed matching time when compared with the binary search trees, and the AHSM algorithm achieves a filtering performance improvement of up to 45% compared with the HSM algorithm.

Future work includes the introduction of statistical search trees into other prevailing packet filtering algorithms, and comparison study of their effect on different schemes. Although the network statistics cannot improve filtering algorithms from time complexity, it is believed that it does help a lot in promoting the average matching speed in practical cases.

## Acknowledgement

This work is supported by the National High-Tech R&D (863) Plan of China (No. 2007AA01Z468)

## References

[1] M. H. Overmars and A. F. van der Stappen, Range searching and point location among fat objects, *Journal of Algorithms*, 21(3), 1996.

[2] B. Xu, D. Y. Jiang and J. Li, HSM: A fast packet classification algorithm, *Proc. 19th Advanced Information Networking and Applications (AINA)*, 2005.

[3] Y. X. Qi and J. Li, Dynamic cuttings: packet classification with network traffic statistics, *Proc. 3rd Trusted Internet Workshop (TIW)*, 2004.

[4] H. Hamed, A. Al-Atawy and E. Al-Shaer, Adaptive statistical optimization techniques for firewall packet filtering, *Proc. IEEE INFOCOM*, 2006.

[5] K. Zheng, C. C. Hu, H. B. Lu and B. Liu, An ultra high throughput and power efficient TCAM-based IP lookup engine, *Proc. IEEE INFOCOM*, 2004.

[6] F. Baboescu and G. Varghese, Packet classification using multidimensional cutting, *Proc. ACM SIGCOMM*, 2003.

[7] P. Gupta and N. McKewon, Algorithms for packet classification, *IEEE Network*, 15(2):24-32, 2001.

[8] F. Baboescu and G. Varghese, Scalable packet classification, *Proc. ACM SIGCOMM*, 2001.

[9] T. Y. C. Woo, A modular approach to packet classification: Algorithms and results, *Proc. IEEE INFOCOM*, 2000.

[10] P. Gupta and N. McKeown, Packet classification using hierarchical intelligent cuttings, *Proc. Hot Interconnects*, 1999.

[11] P. Gupta and N. McKeown, Packet classification on multiple fields, *Proc. ACM SIGCOMM*, 1999.

[12] A. L. Edwards, An introduction to linear regression and correlation, *W. H. Freeman and Co*, San Francisco, California, 1993.

[13] T. C. Hu and A. C. Tucker, Optimal computer search trees and variable length alphabetic codes, *SIAM Journal on Applied Mathematics*, 21:514–532, 1971.

[14] D. Knuth, Sorting and Searching, *The Art of Computer Programming, Addison-Wesley*, Reading, Massachusetts, 2nd edition.