A High Performance IPv6 Flow Table Lookup Algorithm Based on Hash

Huan Guo† Zhengmin Li†‡ Qingyun Liu† Jia Li† Zhou Zhou† Bo Sun‡*

†Institute of Information Engineering, Chinese Academy of Sciences
89 A Minzhuang Rd, Beijing, China
‡National Computer Network Emergency Response Technical Team/Coordination Center of China
A3, Road Yumin, Beijing, China

*Corresponding Author: Email: sunbo@mail.nisac.gov.cn

ABSTRACT
With the rapid increasing IPv6 network traffic, some network process systems like DPI and firewall cannot meet the demand of high network bandwidth. Flow table based on hash is one of the bottlenecks. In this paper, we measure the characteristics of IPv6 address and propose an entropy based revision hash algorithm, which can produce a better distribution within acceptable time. Moreover, we use a hierarchical hash strategy to reduce hash table lookup times further more even in extreme cases.

CCS Concepts
• Security and privacy~Network security

Keywords
IPv6; load balancing; hash.

1. INTRODUCTION

With the rapid development of network, communication has been more efficient. Meanwhile, the spread of pornographic, violence and other information has also been more convenient. In order to prevent the spread of harmful information, Deep Packet Inspection (DPI) technology has been widely used. At the same time, processors and memory speed has not advanced at the same pace. Larger network scale and faster network transmission bring in increasing pressure on DPI[1] systems. This leads to the problem that DPI systems can hardly handle high speed traffic in the circumstances.

One of the main bottlenecks of DPI systems is flow table lookup, as these systems often maintain a flow table to keep tracking the context of TCP/UDP flows. The implementation of flow table is based on hash algorithms using IP address as the input value. Nowadays, IPv6 networks are deployed more and more widely. Since IPv6 has different IP address assignment strategies, as well as some other characteristics, most of the hash algorithms for IPv4 is not available for IPv6 anymore, which may lead to longer computing time or worse distribution.

This paper focuses on the above issue. We first analyze the characteristics of IPv6 address. Then combining with the effect of some bit operators, we propose an Entropy based Revision Hash, namely ERH. Compared with other hash algorithms, the ERH can produce a better result within an acceptable time. However, balancing load in practical cases may not always be perfect due to rapidly varying and unpredictable traffic patterns. So at last, we use a hierarchical hash strategy, which can adapt its own collision solving method to the traffic. It ensures acceptable collisions even in extreme cases.

2. RELATED WORK
Hash has been used thoroughly and widely. The most commonly used hash algorithms are destination address hash, destination address exclusive OR hash, source and destination address exclusive OR hash, network checksum, IPSX[2], Bob[3], CRC32[4-6], MMH[7], XOR_SHIFT and so on. For the first several algorithms, they are very simple, with poor distribution results[8]. For the last several algorithms, which are pretty good and widely used, Guang *et al.* did some test. In the view of computing time, IPSX<Bob<MMH<CRC32, and from the perspective of distribution, CRC32>Bob>MMH>IPSX[9].

XOR and CRC are two well-known hash algorithms. Although neither of them is novel, they can provide high uniformity and low cost[10]. Cao *et al.* simulated performance of several hash algorithms and showed that CRC16 provides the best performance tradeoff [11].

The algorithms above are used for IPv4. IPv6 address is 128 bits long, which is four times long as IPv4, and IPv6 has different IP address assignment strategies. So, if we use the algorithms without any change, we will get a longer computing time and poorer distribution results. There are a few hash algorithms for IPv6 at present, which are listed below.

Von Neumann (VONN) algorithm splits source and destination IP address into 64bits separately, and then take the seven-tuple instead of five-tuple into computation. This method is easy to

WTMC'16, May 30 2016, Xi'an, China

DOI: http://dx.doi.org/10.1145/2903185.2903187

realize by software or hardware, thus it can get a very fast computing speed. Unfortunately, it cannot make a good distribution.

Fowler/Noll/Vo (FNV) uses multiplication and some large constants, so it will run much slower than von Neumann algorithm. But it can get a fairly even distribution.

Linux Kernel Hash uses the last 32 bits of IPv6 address as input directly, and has been used in practical engineering. In most cases, it can achieve a satisfactory effect.

Recent research paper on IPv6, is mainly devoted to provide the methods of looking up table for IPv6 routing. For example, Hu *et al.* studied the realization of IPv6 routing forwarding in multi-core multi-threaded network processors [12]. For the 100G routers, Song *et al.* achieved the routing forwarding by utilizing the distributed and load balancing Bloom Filter algorithm [13]. Wang *et al.* studied the high performance in routing lookup[14]. However, little considerable efforts have been spent on the IPv6 hash algorithms and strategies, and these cannot use on IPv6 hash directly owing to their significant difference between application situations.

3. HIGH PERFORMANCE IPV6 HASH ALGORITHM

Hash algorithm and strategy play a significant role in load balancing. In this section, we first measure the characteristics of IPv6 address. Then we come up with an Entropy based Revision Hash (ERH) algorithm, which can produce a commendable distribution in an acceptable amount of computing time. At last we propose a hierarchical hash strategy to further reduce the hash table lookup time.

3.1 Measurement

One of the reasons for why most hash algorithms in IPv4 are inadequate for IPv6 is that, IPv6 address is 128 bits, which is four times long as IPv4. This leads to a longer computing consumption, which is intolerable for high-speed networks. Another reason is that, IPv6 has a different IP address assignment strategies, which means the randomness in every bit of IPv6 address is also different from IPv4 address.

At present most of the hash algorithms take the four-tuple (SrcIP/DstIP, SrcPort/DstPort) or five-tuple (SrcIP/DstIP, SrcPort/DstPort, protocol) as the input value. Since at least 86.97% of traffic is TCP or UDP [15-17], the protocol field has little information, so we take no account of the protocol field as one of the input values. It's obvious that the distribution of IP address and port influence the hash result directly.

Flow label plays an important role in end-to-end QoS guarantees, security authentication, and load sharing[18]. RFC 6437 recommends a method of calculating hash using the flow label. In some way, flow label also has an influence on hash results. But Li et al. shows that most flow labels are unset, staying in unusable stages [19].

In view of the above points, we mainly measure and analyze the distribution and randomness of IPv6 address.

The most commonly used randomness tests [20] are frequency test, block frequency test, run test, block the longest continuous "1" test, matrix rank test, discrete Fourier transform test, non-overlapping template matching test, overlapping template matching test, the general statistical test, the compression test, linear complexity test, continuity test, approximate entropy test, the part sum test, random walk test, random walk test of variables, etc.. In addition, there is a measurement named Information Entropy [21] in information theory.

As the packets order has no influence on our experiment, and the dataset is a set rather than a sequence, we decide to ignore the factor of order, and make Information Entropy our randomness measurement method.

According to the entropy formula

$$\mathrm{H(x)} = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i) \quad (1)$$

Calculating the bit randomness, we have

$$\mathrm{H(x)} = -\sum_{i=1}^{2} p(x_i) \log_b p(x_i) - p(0) \log_2 p(0)\, p(1) \log_2 p(1) \quad (2)$$

When

$$p(0) = p(1) = \frac{1}{2}$$

H(x) gets a maximum of 1. At this point it gets the best randomness.

We use two datasets. The first one is DS1, which was collected from Mar. 4th to Mar. 11th in 2015 from CSTNET, containing 31,047,034 unique four-tuples. CSTNET is one of the four largest backbones in China, mainly providing non-profit Internet service for science and education with IPv4/IPv6 double stacks access supported. Its bandwidth is about 10Gbps. The second one is DS2, which was collected from Mar. 9th to Mar. 13th in 2015 from China Telecom, containing 54,399,438 unique four-tuples. China Telecom is also a large backbone in China with IPv4/IPv6 double stack access supported. Its bandwidth is about 10Gbps.

The result calculated by information entropy in DS1 and DS2 are shown in Figure 1 and Figure 2.

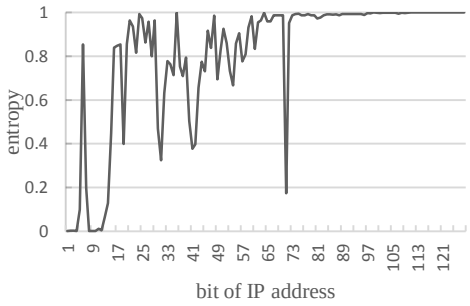

Figure 1. The entropy of every bit of IP address in DS1

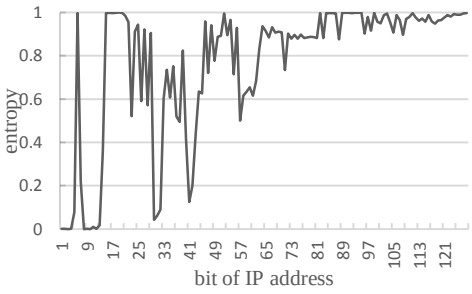

Figure 2. The entropy of every bit of IP address in DS2

From the result data, we can see an obvious difference between ours and the result of Qiao et al. [17]. The reason is Qiao may use a small and single dataset. Besides, they only measured in narrow scope so that there are a lot of common prefixes among the IPv6 addresses, making many restrictions on the result. Qiao gets the conclusion that the 8,12,14,15,16 bytes have high entropies. We

get a conclusion that 3, 7-16 bytes have high entropies, which means the bit range which can be used as the input value of hash increases 1.4 times compared with Qiao.

Although we have much more convincing conclusions using much larger datasets, this is far from enough. With there being different IPv6 address allocation methods, different regions may have different randomness characteristics. So it stands a chance that our conclusion is only suitable for most situations, but there are still some exceptions.

Due to this, we classify the dataset grouping by regions and do the experiment once more. About half thousand regions are detected. We ignore some regions that only consist a small number of unique four-tuples. The regions left are divided into two parts according to the characteristics of IP address distribution. The normal part has common distribution and the rest is the rare part. The two line graphs of entropy are shown in Figure 3 and Figure 4.

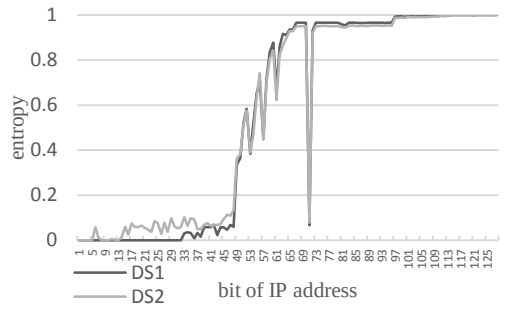

Figure 3. The entropy of every bit of IP address of the normal part in DS1 and DS2

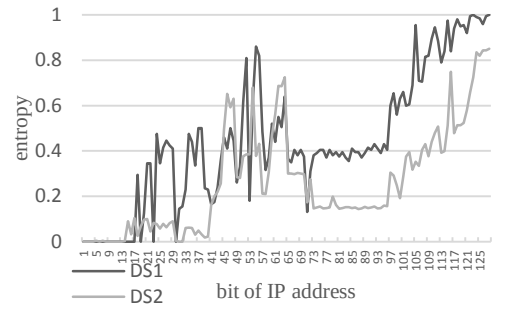

Figure 4. The entropy of every bit of IP address of the rare part in DS1 and DS2

For the normal part, there are little problems with hash algorithms. Simply dealing with the last 32 bits is always effective enough as well as fast enough. But for the rare part, dealing properly with it is just as important as with the normal part. The bytes of 0xFFFE contributed by the modified EUI-64 is a factor, and some regions' allocation method is another. Some of them just allocate simply from 0x00000001 to 0xFFFFFFF. What's more, the number of hosts is far less than the address space can hold. In another word, more than half of the interface ID bits are zero. On this condition, using the same algorithm will lead to a bad distribution.

3.2 Entropy based revision hash (ERH)

Combining with the rare part distribution, we propose a simple, fast and effective way to enhance the entropy of the last 32bits of IPv6 addresses in the rare part. The basic idea is to revise the low entropy bits with the high ones, using XOR or other operations. Generally, we can revise the 97th-112th bits (the 13th byte) using 57th-72th bits (the 8th byte) or front bits. When revising, we should skip the zero byte so that we can make it more effective. After the processing (simply use XOR), the entropy is doubled, meaning 1.23 growth. The code below is our entropy based revision hash algorithm.

```
ERH pseudo-code
  a1 = bytes of srcIP[12..13]
  a2 = bytes of srcIP[14..15]
  b1 = bytes of dstIP[12..13]
  b2 = bytes of dstIP[14..15]
  for x in (a1,a2,b1,b2)
    if x==0x00 or x==0xFF
      then
        tmp = previous byte with highest entropy
        x = x xor tmp
    a = a1<<16 + a2
    b = b1<<16 + b2
    c = src_port xor dst_port
    hash_result = operation_with_xor_shift(a,b,c)
```

3.3 Hierarchical hash strategy

However, balancing load in practical cases may not always be perfect due to rapidly varying and unpredictable traffic patterns [22].

A good hash algorithm may be perfect for one situation, but may perform badly in other circumstances. We have real data to prove it. So, only a hash algorithm for IPv6 is far from enough. An appropriate hash strategy is also absolutely necessary.

A good hash should make little collisions, but no hash can ensure no collisions. In extreme cases, a good hash algorithm can still lead to many collisions. So, a good hash strategy is needed in this case to reduce average lookup times. Hence, we design a hierarchical hash strategy, on the accounts of three points: simple hash table may lead to serious collisions even using a good hash algorithm; when just using source and destination address as hash input, the distribution is relatively even; a hierarchical hash strategy can ensure acceptable collisions even in extreme cases.

Steps of hierarchical hash strategy are described below.

Initial a two-level hash table at first. Using the source and destination IP addresses as the first level hash input, and four-tuple or two-tuple left as the second level hash input. From the second level, every level is treated as below.

Use zippers to solve collisions. Re-sort the zipper linked lists by LRU method. Expand to a next-level hash table when collisions exceed the threshold.

Use two-level hash because of two main reasons: streams can be distributed evenly after source and destination IP hash; Intel series NICs can compute the two-tuple/four-tuple hash itself automatically when it receives a packet, and we can get the result through its APIs, so we can save a large amount of computing time.

There are two main arguments need to be settled: the size of each level in the hash table, and the collision threshold. Different situations should make different arguments to reach their best performance. But in general, the size of the next level should be less than its higher level hash table, and the threshold should not be too large, usually no more than the maximum acceptable lookup times.

In our environment, we set the size of each level to be 1.5 times as large as its next level, and the threshold to be 5.

4. EXPERIMENTAL RESULTS

The first experiment compares ERH with some other algorithms with respect to the computing time and average lookup times. Results are shown below in Figure 5 and Figure 6. DS1 and DS2 were collected from CSTNET and China Telecom separately, all containing 10,000,000 packets, and the size of hash table is set to 20,000,000.

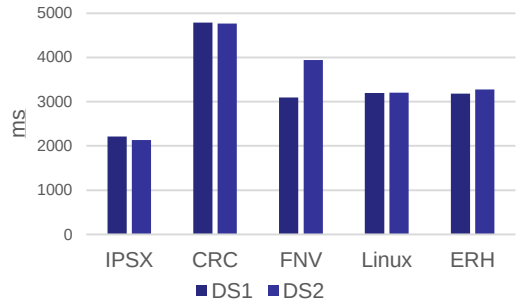

Figure 5. The computing time comparison

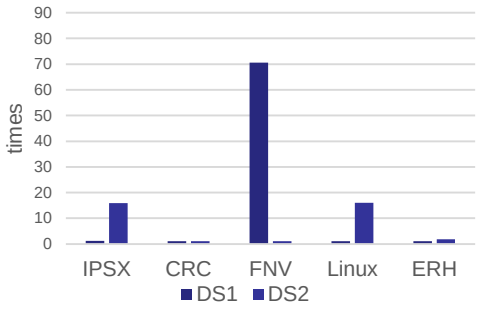

Figure 6. The average lookup times comparison

It can be seen that, CRC has the best distribution result at the cost of more computing time. IPSX uses the least time with a common result. Among the algorithms within acceptable time consumption, Entropy based Revision Hash (ERH) has the best distribution result. Our comparison results are shown in Figure 7 and Figure 8. Our system was deployed on CSTNET and China Telecom, and run for 24 hours from 2016-01-21 16:00 PM to 2016-01-22 16:00 PM.

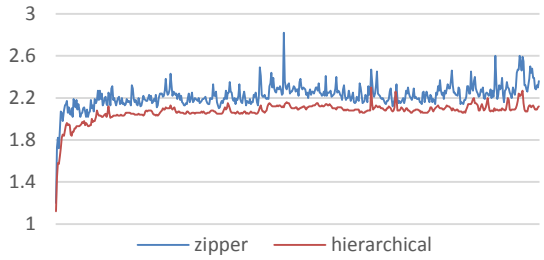

Figure 7. The average lookup times comparison for CSTNET

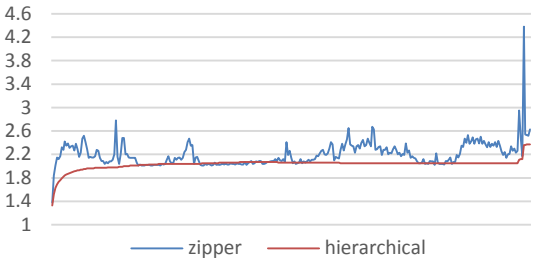

Figure 8. The average lookup times comparison for China Telecom

From the results above, we can see that traditional hash table with zippers to solve collisions is not stable. Our hierarchical hash strategy makes it stable, and reduces the average lookup times at the same time.

5. CONCLUSIONS

In this paper, we first measure the characteristics of IPv6 address, and find that 3, 7-16 bytes of the address have high entropies. Then we propose an entropy based revision hash algorithm for IPv6 address, which can make a better distribution under acceptable computing time. A good hash should make little collisions, but no hash can ensure no collisions. So we come up with a hierarchical hash strategy so that we can ensure an acceptable and stable average lookup times even under extreme circumstances.

In the future work, we will design a method which can dynamically set the arguments to adapt to certain circumstance automatically.

6. ACKNOWLEDGEMENTS

This work was supported by The Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDA06030200); The National Natural Science Foundation of China (No.61402474).

7. REFERENCES

[1] Kim, N. U., Jung, S. M., & Chung, T. M. (2011). An efficient hash-based load balancing scheme to support parallel NIDS. In Computational Science and Its Applications-ICCSA 2011 (pp. 537-549). Springer Berlin Heidelberg.

[2] T. Zseby, Sampling and filtering techniques for IP packet selection.

[3] Jenkins B. Algorithm Alley, Dr. Dobb's Journal September 1997[J /OL].http://burtleburtle.net/bob/hash/doobs.html.

[4] M. Molina, S. Niccolini, N. Duffield, A comparative experimental study of hash functions applied to packet sampling, in: International Teletraffic Congress (ITC-19), Beijing, 2005.

[5] R. Braden, D. Borman, C. Partridge, Computing the internet checksum, ACM SIGCOMM Computer Communication Review 19 (2) (1989) 86–94.

[6] I. O. for Standardization, Information Processing Systems, Data Communication, High-level Data Link Control Procedures, Description of the X.25 LAPBcompatible DTE Data Link Procedures, International standard, International

Organization for Standardization, URL https://books.google.com/books?id=ff8YuQAACAAJ, 1986.

[7] S. Halevi, H. Krawczyk, MMH: Software message authentication in the Gbit/second rates, in: Fast Software Encryption, Springer, 172–189, 1997.

[8] Ying Z. & Hesheng W. (2014). Hash algorithm comparison and analysis of load balancing for multi-process. Computer engineering, 40(9), 71-76.

[9] Guang C., Jian G., Wei D, & Jialing X. (2005). Hash algorithms for IP flow measurement. Journal of software, 16(5), 652-658.

[10] Xiong, B., Yang, K., Li, F., Chen, X., Zhang, J., Tang, Q., & Luo, Y. (2014). The impact of bitwise operators on hash uniformity in network packet processing. International Journal of Communication Systems, 27(11), 3158-3184.

[11] Cao, Z., Wang, Z., & Zegura, E. (2000). Performance of hashing-based schemes for internet load balancing. In INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (Vol. 1, pp. 332-341). IEEE.

[12] Hu, X., Tang, X., & Hua, B. (2006, March). High-performance IPv6 forwarding algorithm for multi-core and multithreaded network processor. In Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming (pp. 168-177). ACM.

[13] Song, H., Hao, F., Kodialam, M., & Lakshman, T. V. (2009, April). Ipv6 lookups using distributed and load balanced bloom filters for 100gbps core router line cards. In INFOCOM 2009, IEEE (pp. 2518-2526). IEEE.

[14] Wang, R., Du, H., & Wang, Y. (2012, August). A Design and Implementation of a High Performance IPv6 Lookup Algorithm Based on Hash and Cam. In Proceedings of the 2012 International Conference on Computer Application and System Modeling. Atlantis Press.

[15] Shen, W., Chen, Y., Zhang, Q., Chen, Y., Deng, B., Li, X., & Lv, G. (2009, August). Observations of IPv6 traffic. In Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium on (Vol. 2, pp. 278-282). IEEE.

[16] Li, F., An, C., Yang, J., Wu, J., & Zhang, H. (2014). A study of traffic from the perspective of a large pure IPv6 ISP. Computer Communications, 37, 40-52.

[17] Qiao, P., & Changxing, P. (2007). Distributed sampling measurement method of network traffic in high-speed IPv6 networks. Systems Engineering and Electronics, Journal of, 18(4), 835-840.

[18] Hu, Q., & Carpenter, B. (2011). Survey of proposed use cases for the IPv6 flow label.

[19] Li F, An C, Yang J, et al. A study of traffic from the perspective of a large pure IPv6 ISP[J]. Computer Communications, 2014, 37(1):40–52.

[20] https://en.wikipedia.org/wiki/Randomness.

[21] https://en.wikipedia.org/wiki/Entropy_(information_theory).

[22] Molina, M., Niccolini, S., & Duffield, N. G. (2005, August). A comparative experimental study of hash functions applied to packet sampling. In International Teletraffic Congress (ITC-19), Beijing.