A Fast Update Scheme for TCAM-Based IPv6 Routing Lookup Architecture

Yi-Mao Hsiao, Ming-Jen Chen, Yu-Jen Hsiao, Hui-Kai Su and Yuan-Sun Chu
SoC Laboratory, Department of Electrical Engineering,
National Chung Cheng University,
Chia-Yi 621 Taiwan, China
93mowmow@vlsi.ee.ccu.edu.tw

Abstract—**With the rapid development of the Internet applications and the explosion of end users, IP address has been exhausted and routing lookup speed is the bottleneck of router design. In addition TCAM is widely used for implementation fast IP forwarding table lookup. However, the need to maintain a sorted list incremental updates may slow the lookup speed in a TCAM. In this paper, a fast TCAM update scheme is proposed for TCAM-Based IPv6 routing lookup architecture. Based on the characteristic of prefix distribution, we design a TCAM-Based router architecture for IPv6 routing lookup. As simulation result shows more than 90% update do not need any memory movement in updating operation. The worse case is (D-2)/2, where D is the length of chain.**

Keywords- routing lookup; TCAM; update; IPv6

I. Introduction

With the growth of Internet users and services, the IPv4 network has two major limitations- routing lookup speed and exhausted IP address. The router is the bottleneck of network because the network speed increases rapidly. There are three key design issues for the next generation IP routers: link speeds, router throughput, and packet forwarding rates. Two issues are available. The packet forwarding design performs poorly improvement at present. The router looks routing table up by destination IP address, and forwards the packet to next router bases on port number. Therefore, designing a fast routing lookup scheme is very essential.

The increasing of internet users causes IPv4 address exhaust. In order to resolve this limitation, the short term solution is Classless Inter-Domain Routing (CIDR)[1][2]. Each routing entry uses a < prefix, prefix length > pair to increase the usage of IP address and longest prefix match (LPM) for routing table lookup [3]. The LPM limitation makes routing table design become more complex. The long term solution for insufficient IP addresses is the IPv6 protocol defined by IETF (Internet Engineering Task Force) [4] [5]. The IPv6 address length is 128 bits so that can avoid the situation of exhausted IP address. However, IPv6 also makes the routing lookup becomes more complex because of 128 bits length address.

Ternary content-addressable memory (TCAM) is widely used as lookup engine for commercial router or network device [6][7][8]. Each cell in TCAM can take three logic states: 0, 1, or do not care bit X. TCAM searches all routing entries in parallel that performs *O(1)* complexity to achieve ten million packet lookup per second. Therefore, TCAM is very suitable for packet classification and packet forwarding. However, TCAM has some disadvantages. It has high power consumption. Zheng decreases the number of TCAM entries triggered in each lookup operation [9]. Panigrahy partitions the set of prefixes in the routing table into small groups, and each route lookup uses only one group [10]. He proposes the concept of paged TCAM to achieve significantly lower power consumption.

Because of longest prefix match issue, routing entries need to be sorted by prefix length in order to ensure lookup priority. The sorting limitation is the main bottleneck of using TCAM for routing lookup. Almost 100~1000 prefixes are updated consequently per second in a router nowadays. The router usually does inserting and deleting operation. The sorting limitation slows down the update speed of the router because packets are buffered by previous operation. Therefore, the update speed is an essential issue for improving router performance. In this paper, we propose a fast update scheme for IPv6 routing lookup based on TCAM architecture. The architecture contains two layers, the first one is a simple compare logic circuit and the second one is TCAM arrays. The proposed fast update scheme can overcome the sorting limitation of TCAM. The rest of the paper is described as follow. Section II is the related work. Section III introduces the system architecture. Section VI is the fast update scheme. The performance analysis and simulation result is in section V. Section VI is the conclusion.

II. Related Work

A. *TCAM updating*

The previous researches sort prefix efficiently and arrange the empty space to speed up TCAM updating. The Centralized Sorting Prefix (CSP) scheme [11] puts empty space at the bottom of TCAM. But, the worse-case time complexity *O(N)* is too slow. The Sorting Prefix Block (SPB) scheme [11] keeps a few empty space locations at all nonempty memory locations. The advantage of SPB is that the system can use the empty space to move the entries. If one empty space is full, the next empty space is also moved for accessing the entries. The worse case of SPB is *O(N)* which is the same as CSP. Another shortage of this solution is waste of TCAM space.

Prefix-length ordering constraint optimal algorithm (PLO_OPT) [11] arranges the prefix by prefix length distribution. In the PLO_OPT scheme, the empty space is in the center of TCAM. The arrangement is such that the set of prefixes of length L, L − 1, ..., L/2 are always above the free space pool, and the set of L/2 − 1, L/2 − 2, ..., 1 prefixes are always below the free space pool. L is the width of the destination address field (L is equals to 32 in IPv4 and 128 in IPv6). The same as CSP, the update scheme uses a neighbor space in the empty space. If the update prefix is under the empty space, the system asks a free space from the bottom of the empty space; If the update prefix is on the empty space, the system asks a free space from the top of the empty space. Therefore, the movement times do not exceed twice in update operation. The worse case is *O(L/2)*.

If all entries on the routing table converts into trie (is an ordered tree data structure that is used to store an associative array where the keys are usually strings.) data structure on the same chain (the path from the root to a leaf node), the entries needs sorting. Therefore, only the overlapping prefixes need sorting [11] [12]. Chain-ancestor ordering constraint optimal algorithm (CAO_OPT) proposes how to arrange the empty space which is in the center of TCAM [11]. It divides the chain into two groups. Assume D is the length of a chain. There are at most ⌈D/2⌉ prefixes between the prefix and a free space entry in the TCAM. Therefore, when a prefix is moved, the movement times are less or equal to D/2. Whether in the deleting or inserting operation, the movement of prefix does not affect the chain-ancestor ordering constraint. The prefixes are moved up or down in the same chain, it does not change the ordering of the chain. The memory movement cost is less or equal to D/2.

B. *routing table compaction*

The Internet developments very fast in nowadays, the number of routing prefixes are growing rapidly. Compacting routing tables can efficiently control the explosion of routing table. The commercial TCAM is expensive than SRAM and DRAM. TCAM is cost more power consumption than them also. If the routing table is compacted, we can not only reduce the routing entries but also reduce the power consumption of TCAM. Pruning technique deletes the redundant prefixes in a routing table [13]. Mask extension base on the characteristic of TCAM that can store and lookup don not care bit x [14]. TCAM stores prefix and mask which is compose by series 1 and 0. The serious 1 is the length of prefix. Ravikumar figure out next hop are not so many that they can compact the prefix and mask in a TCAM. In order to prevent error, the mask extension technique uses the same next hop and length of prefixes.

TABLE I. THE PERCENTAGE OF PREIFX BEGIN WITH 0X2001

	Potaroo	***Tilab***	***USA***	***Japan***
Prefix begin with 0x2001	81%	68%	75%	65%

III. SYSTEM ARCHITECTURE

In order to understand the situation of IPv6 such as [15], we also collect and analyze four backbone routers BGP routing tables in 2008 from the following source: USA in the RouterViews project [16], Japan in the RouterViews project [16], Potaroo [17] and Tilab [18]. Most of the prefixes are begin at 0x2001 shown in table I. The first 16bit prefix distribution of four routers is shown in figure1.The IPv6 global unicast address format is shown as figure 2.

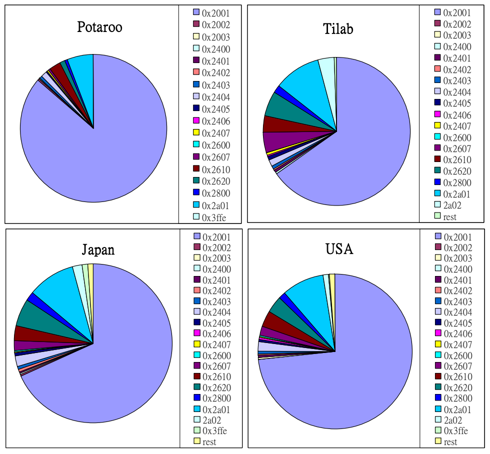

Figure 1. First 16bit prefix distribution of four router(USA 、 Japan 、 Potaroo、Tilab)

3	45 bits	16 bits	64 bits
001	Global routing prefix	Subnet ID	Interface ID
Network ID (1~64)			

Figure 2. IPv6 global unicast address format

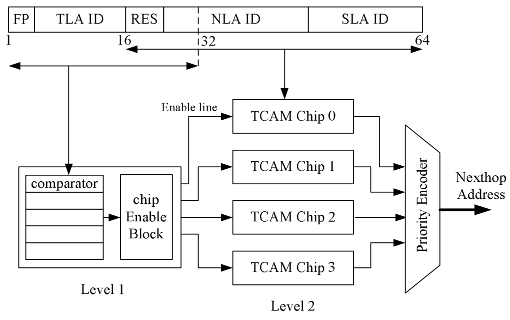

Figure 3. System architecture

In this paper, we design the system as 1-64 bit length distribution base on network ID. We design a TACM based IPv6 routing lookup architecture shown in figure 3. The architecture contains two layers, a simple compare logic circuit with chip enable block and TCAM arrays. The compare logic circuit decides the lookup prefix based on the first 32bit of the prefix entry. Chip enable block is a combinational circuit that connects the driver line of compare logic and generates the enable line to the second layer TCAM. Chip enable block chooses suitable enable line to drive the proper TCAM based on the result of compare circuit shown in Figure 4.

Comparator	
2001:0: ~ 2001:5554	TCAM chip 1
2001:5555~2001:AAA9	TCAM chip 2
2001:AAAA~2001:FFFF	TCAM chip 3
Other prefixes	TCAM chip 4

Figure 4. The comparator diagram

IV. FAST TCAM UPDATE SCHEME

After observing from backbone routers, we figure out the first 16bits of entries are the same. But the $17^{th} \sim 32^{th}$ bits are not the same. We also find that 80% prefixes do not overlap with the other prefix. There should be the first two nodes of one chain when prefixes are converted into trie chains. We propose a fast update scheme for TCAM. Prefixes are converted into trie structure and stored in TCAM based on the ordering constraint of chain-ancestor. The empty spaces locate in two spaces of TCAM.

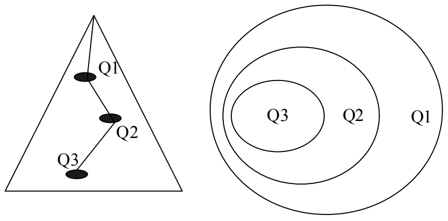

Figure 5. Prefixes set relation.

Assume Q1, Q2 and Q3 are prefixes in the routing table. Figure 5 shows the relation of prefix set and trie structure. If there is an IP address does routing lookup on TCAM then Q1, Q2 and Q3 can match. Therefore, the longest prefix match of D is Q3. Q1 is a superset of Q2, Q1 is a superset of Q3 and Q2 is a superset of Q3. Q2 and Q3 are subsets of Q1; Q3 is a subset of Q2.

After the prefixes are converted into trie structure, we divide the prefixed into three classes:

- Root prefixes: The prefix has no superset.
- Level two prefixes: The prefix has a superset so that the prefix is in the second order.
- Other prefixes: The prefix is neither the root prefix nor the level two prefix.

Figure 6 shows the structure example. In order to decrease memory movement during update operation, we divide the empty space into two spaces and arrange special region for different prefix class.

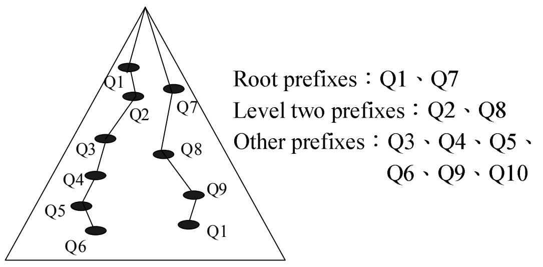

Figure 6. Prefixes classification example.

The TCAM space management architecture is shown in Figure 7. We design four storage space and two empty spaces. Space 1 store root prefix, space 2 store level two prefix and an empty space is between space 1 and space 2. Space 3 and space 4 store other prefix; another empty space is also between them.

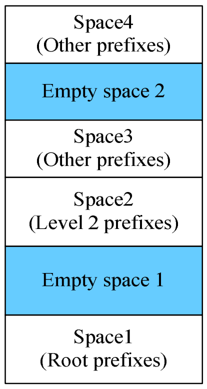

Figure 7. TCAM space management architecture.

- Assume D is the length of a chain and D(p) is the depth from root to prefix of the chain.

Space 1 stores the prefixes which D(p)=1, space 2 stores D(p)=2, space 3 stores the prefix which range of D(p) is from 3 to (D + 2) / 2 and space 4 stores the prefix which range of D(p) is from (D + 2) / 2 + 1 to D. Space 2 and 3 are opposite location so that there is no obvious line between them as long as the chain does not violate chain-ancestor ordering constraint.

When the system updates a prefix, it can use empty space directly and needs no extra movement. The complexity is *O(1)*. Besides, when the system updates the other prefix, we divide the other prefixes into two spaces and there is another empty space between them. Because root prefix and level two prefix are stored in another empty space. The movement times of other prefix are less than (D-2)/2. The propose scheme includes insert and delete operation shown in figure 8 and figure 9. In insert operation: There are no movement in the root prefix without subset and one movement in the root prefix with subset. When inserting a level two prefix, it needs no movement also. When inserting other prefix, the memory movement is less than (D-2)/2. In delete operation, there is no movement in the root prefix without subset and one movement in the root prefix with subset. When delete level two prefix needs one movement. When delete other prefix, the memory movement is less than (D-2)/2.

```
Insert ( prefix)
{
    Find corresponding TCAM ( ) ;
    If ( prefix has superset)
        {
            If(number of superset ==1)
                {  Insert to Space 2  }
            Else
                {
                    If (D(p)>3 and D(p) < ( D+2 )/2 )
                        {Insert to Space 3}
                    Else
                        {Insert to Space4}
                }
        }
    Else
        {  Insert to Space 1}
}
```

Figure 8. Inserting operation

```
Delete ( prefix)
{
    Find corresponding TCAM ( ) ;
    If ( Find in Space1)
        {
            If( prefix has subset)
                {   delete in Space1
                    move Q2 to Space1     }
            Else
                {   delete in Space1 }
        }
    If ( Find in Space2)
        {   delete in Space2 }
    If ( Find in Space3)
        {   delete in Space3 }
    If ( Find in Space4)
        {   delete in Space4 }
}
```

Figure 9. Delete operation.

An example is shown in Figure 10. Q1 belongs to root prefix so that Q1 is stored in space 1, Q2 belongs to the level two prefix so that it is stored in space 2. Space 3 store the prefix which range of D(p) is from 3 to (D + 2) / 2 . Now D is 6, so Q3 and Q4 store in space 3. Space 4 stores the prefix which the range of D(p) from (D + 2) / 2 + 1 to D. So, Q5 and Q6 are stored in space 4.

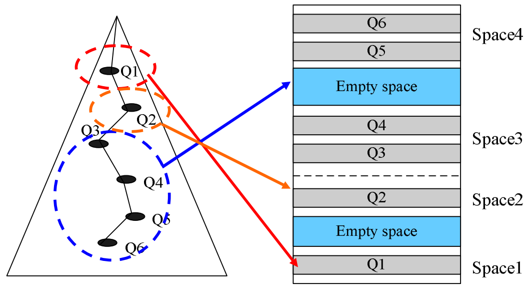

Figure 10. Example of prefixes storage in TCAM.

V. Simulation Result and Analysis

We use two routing table Potaroo and Tilab to simulate lookup operation. Because the existence routing entries are very few that are almost 1000 entries. Based on IPv6 characteristic, V6Gen [19] is development to create IPv6 packets. In order to simulate huge numbers of prefixes, we use V6Gen to generate 10000, 50000 and 100000 entries.

In our fast update scheme for TCAM, prefixes are classified into root prefix, level two prefix and other prefix. Table II shows the prefix class distribution analysis of five routing tables. In our scheme, root prefix without subset and level two prefix need no memory movement. The root prefix with subset are 89.6% and the level two prefixes with subset are 9.3% of total routing table. So 89.6% entries need no movement and 9.3% entries need only two movements.

TABLE II. Prefixes Class Distribution Analysis.

	Root prefixes			Level two prefixes	Other prefixes
Potaroo (933)	716	Has subset	63 (7%)	192 (20%)	25 (3%)
		No subset	653 (70%)		
Tilab (1098)	1007	Has subset	41 (4%)	91 (8%)	0 (0%)
		No subset	966 (88%)		
V6Gen 10000	9227	Has subset	1145 (12%)	484 (5%)	206 (2%)
		No subset	8082 (81%)		
V6Gen 50000	48003	Has subset	4801 (10%)	1319 (2.5%)	349 (0.7%)
		No subset	43202 (86.8%)		
V6Gen 100000	93577	Has subset	13098 (13.5%)	2647 (2.9%)	663 (0.6%)
		No subset	80459 (83%)		

Chain length (D) affects update performance. The chain length statistic of five routing tables is shown in Figure 11. The memory movement of other prefix is (D-2)/2. Therefore, the movement of other prefix is one in Potaroo routing table. There is no prefix in Tilab routing table. There are three movement in V6Gen 10000 table and five movement in both V6Gen 50000 and 100000. But other prefixes are very less almost 1.2% of total entries so that most entries need no memory movement during updating operation. Table III shows the memory movement statics of five routing source that almost 90% routing entries need no memory movement.

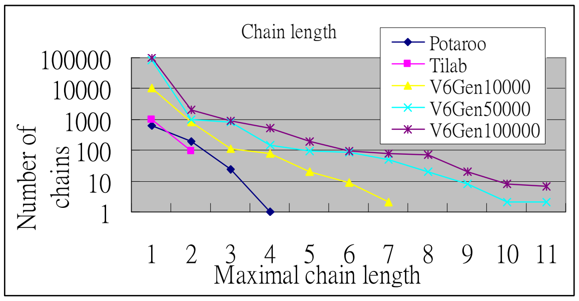

Figure 11. Chain length statistic.

TABLE III. Memory Movement Times Stastics.

	no movement	2	(D-2) / 2
Potaroo	90%	7%	3%
Tilab	96%	4%	0%
V6Gen 10000	86%	12%	2%
V6Gen 50000	89.3%	10%	0.7%
V6Gen 100000	85.9%	13.5%	0.6%

Table IV shows the comparison result. L is maximal prefix length and D is length of maximal chain. Our scheme is better that other scheme, worse case is (D-2)/2 and 90% prefixes need no memory movement.

TABLE IV. COMPARISON WITH OTHER SCHEME.

	Memory movement Worst case	**In IPv6**
L - algorithm	L	Too much memory movement. In IPv6 , L = 48
PLO_OPT algorithm	L / 2	Too much memory movement In IPv6 , L = 48
CAO_OPT algorithm	D / 2	D = 4 ~ 11 Lots of memory movement below empty space
Our scheme	(D – 2) / 2	90% prefixes Need no memory movement

We use pruning and mask extension techniques for five routing tables. Figure 12 and 13 shows the compaction result. The compaction ratio of V6Gen routing tables are 26%, 30% and 35% of 10000, 50000 and 100000 entries routing tables. The compaction ratio of Potaroo and Tilab routing tables are 28% and 25%. Therefore we can reduce 25%~35% capacity of TCAM and reduce power consumption.

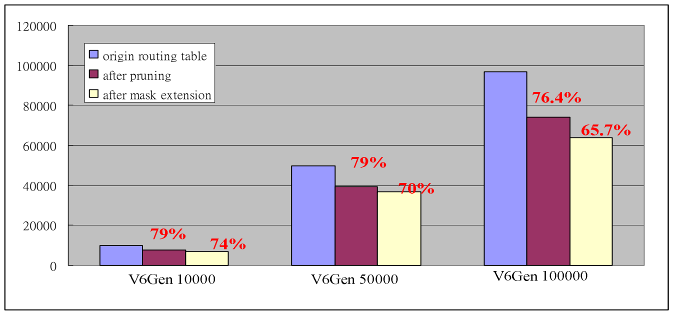

Figure 12. Compaction result of V6Gen routing tables.

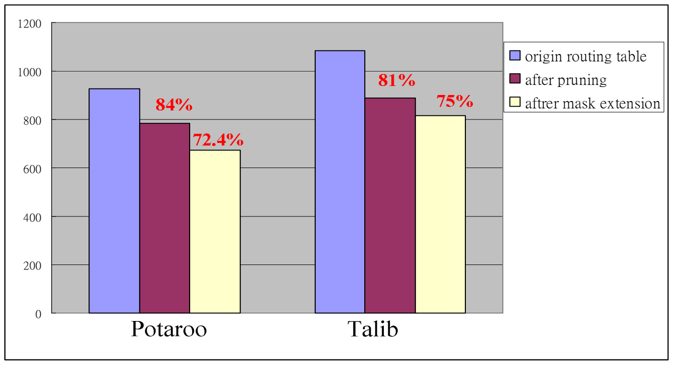

Figure 13. Compaction result of Potaroo andTilab routing tables.

VI. CONCLUSION

In this paper, we propose a fast update scheme based a TCAM-based IP lookup architecture for IPv6. The architecture contains two layers, a simple compare logic circuit and TCAM arrays. We classify the prefix into three classes. We design four storage space and two empty spaces as TCAM space management architecture. After the simulation, 90% entries need no memory movement and the worse case memory movement is (D-2)/2. Our proposed scheme overcomes the limitation of TCAM update. Besides by using pruning and mask extension algorithm, the five routing tables can reduce 25%~ 35% capacity. The proposed TCAM-based IPv6 routing lookup architecture can not only reduce capacity but also reduce power consumption.

REFERENCES

[1] Y. Rekhter and T. Li, RFC1518: "An Architecture for IP Address Allocation with CIDR," *Internet Engineering Task Force (IETF)*, Sept. 1993.

[2] J. Yu V. Fuller, T. Li and K. Varadhan, RFC1519: "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy," *Internet Engineering Task Force (IETF)*, Sept. 1993.

[3] W.Doeringer ,G .Karjoth ,and M.Nassehi, "Routing on Longest-Matching Prefixes," *IEEE/ACM Trans.Networking*,vol. 4,Feb.1996,pp. 86-97.

[4] R. Hinden and S. Deering, RFC3513: "Internet Protocol Version 6 (IPv6) ddressing Architecture," *Internet Engineering Task Force (IETF)*, April. 2003.

[5] S. Deering R. Hinden and E. Nordmark, RFC3587: "IPv6 Global Unicast Address Format," *Internet Engineering Task Force (IETF)*, August. 2003.

[6] P. Francis A.J. McAuley, "Fast routing table lookup using cams," in *INFOCOM 93*, 28 March-1 April 1993, vol. 3, pp. 1382 – 1391.

[7] Ravikumar, V.C. and Mahapatra, R.N. ,"TCAM Architecture for IP Lookup Using Prefix Properties," *Micro, IEEE*, Mar-Apr 2004 Volume: 24, Issue: 2 On page(s): 60- 69

[8] Mohammed J.Akhbarizadeh, Mehrdad Nourani, Rina Panigraphy, and Samar Sharma, "A TCAM-based parallel architecture for high speed packet forwarding," IEEE Transactions on Computers, vol.56, pp.58- 72, 2007.

[9] Kai Zheng , Chengchen Hu, Hongbin Liu and Bin Liu ,"An ultra high throughput and power efficient TCAM based IP lookup engine," *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 7-11 March 2004 Volume: 3, On page(s): 1984-1994 vol.3

[10] Panigrahy, R. and Sharma, S.,"Reducing TCAM power consumption and increasing throughput," High Performance Interconnects, 2002. Proceedings. 10th Symposium on, 2002 On page(s): 107- 112

[11] Devavrat Shah and Pankaj Gupta,"fast updating algorithms for TCAMs," *IEEE Micro* Volume 21 , Issue 1 (January 2001) Pages: 36 - 47

[12] Weidong Wu and Ruixuan Wang ,"A TCAM management scheme for IP lookup," *International Conference on Networks 2006*. ICON '06. 14th, Sept. 2006 Volume: 1, On page(s): 1-4

[13] Ravikumar, V.C. and Mahapatra, R.N. ,"TCAM Architecture for IP Lookup Using Prefix Properties" in Micro, IEEE, Mar-Apr 2004 Volume: 24, Issue: 2 On page(s): 60- 69

[14] Huan Liu "Routing table compaction in TCAM"IEEE Micro ,Jan/Feb 2002 Volume: 22, Issue: 1 On page(s): 58-64

[15] Zhenqiang Li , Dongqu Zheng and Yan Ma ," Tree, Segment Table, and Route Bucket: A Multi-Stage Algorithm for IPv6 Routing Table Lookup," *INFOCOM 2007*. 26th IEEE International Conference on Computer Communications. IEEE , 6-12 May 2007 On page(s): 24

[16] http://www.routeviews.org/

[17] http://bgp.potaroo.net/v6/as1221/bgptable.txt

[18] http://net-stats.ipv6.tilab.com/bgp/bgp-table-snapshot.txt

[19] Kai Zheng and Bin Liu,"A Scalable IPv6 Prefix Generator for Route Lookup Algorithm," *Advanced Information Networking and Applications*, 2006. AINA 2006. 20th International Conference on , 18-20 April 2006 Volume: 1