V6Gene: A Scalable IPv6 Prefix Generator for Route Lookup Algorithm Benchmark [i]

Kai Zheng, Bin Liu
Department of Computer Science, Tsinghua University, Beijing, P.R.China 100084
zk01@mails.tsinghua.edu.cn, liub@tsinghua.edu.cn

Abstract

Most conventional IPv4-based route lookup algorithms are no more suitable for IPv6 packet forwarding due to the significantly increased 128-bit-long address. However, as a result of lacking of standard IPv6 route databases, it is hard to make benchmarks for the new generation IPv6-based algorithms developing/evaluation. In this paper, based on the studies of initial IPv6 prefix distributions and the associated RFC documents, we originally develop a scalable IPv6 prefix generator, called V6Gene, for IPv6-based route lookup algorithms benchmarking. According to the RFCs and other associated standards, V6Gene generates IPv6 route prefixes from the initially assigned LIR (Local Internet Registries) prefixes collected from the real world, simulating the process of future IPv6 address block allocation from the LIRs to their subscribers. V6Gene is totally flexible for generation of all kinds of route databases with different characteristics. It is simple for implementation and can be easily integrated within other IPv6 benchmark tools/systems.

V6Gene is publicly available at http://zheng_kai.home4u.china.com/V6Gen.htm.

1. Introduction

With the rapid development of the Internet applications and the explosion of the end users, the available IPv4 addresses that can be allocated are almost exhausted [1]. Though several approaches have been made to temporally cope with the problem，such as NAT (Network Address Translation), however due to its drawbacks of not well supporting peer-to-peer applications and with other security problems, it is widely/publicly believed that there should be a new generation of IP protocol to replace IPv4. IP Version Six (IPv6) then emerge as the times require. It adopts a 128-bit address space, which provides about 2^{96} times of available addresses more than that of IPv4. Besides that, IPv6 provides very good supports for mobility and information security as well. It also offers a better support for QoS-Control than that of its counterpart, since its packet header is far more flexible and can contain more detailed information for flow/application identification.

However, the change of the protocol itself, especially the distinctively increased 128-bit address, on the other hand brings quite a lot of un-neglectable challenges for developing networking infrastructures based on IPv6. For instance, the performance of conventional route lookup algorithms (i.e., both trie-based algorithms [2][3][4][5] and TCAM-based schemes [6][7][8]) implemented in most current packet forwarding devices are sensitive to the search key (i.e., IP address) length, and, therefore, will distinctively decrease when migrated to IPv6. This means that a new generation of high performance route lookup algorithms based on IPv6 should be developed accordingly.

Yet, the major concern of this paper is not the research of finding efficient route lookup algorithms itself, but to develop a scalable benchmarking tool for such algorithms developments. Notice that in order to develop a high performance and suitable route lookup algorithm for next generation high speed packet processing, one should first inspect the route databases and make use of the characteristics of the distribution, and then use the standard route databases for benchmark and evaluate the corresponding performance, as what most researchers have done when they developed IPv4 route lookup algorithms [3][4][5][6][7][8].

However, due to the following three reasons, almost no real-world route databases can be utilized for IPv6-based algorithm benchmark currently: 1) Since IPv6 is in its initiation period, only a small portion of address blocks are assigned/allocated up to now; 2) The currently assigned blocks are mostly LIRs (Local Internet Registries, or Large ISPs) level blocks, which means that the current IPv6 prefix distribution should be quite different with its future pattern which contain mainly end subscriber level address blocks; 3) Most nowadays IPv6 based networking are for testing or

[i] This work is supported by NSFC (No. 60173009 and 60373007), China 863 High-tech Plan (No. 2002AA103011-1 and 2003AA115110), China/Ireland Science and Technology Collaboration Research Fund (CI-2003-02) and the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20040003048).

experiment purposes. Therefore, on the one hand relatively few organizations provide their IPv6 route databases for research, and on the other hand, though some of them share their route databases [10][12], the prefixes in such IPv6 route tables are somehow local, i.e., without universality for research purpose, such as benchmarking. So, providing artificially generated IPv6 route prefix database with public availability, reliability and universality are, therefore, very essential for the new generation route lookup algorithm benchmarking.

D.E.Taylor developed a policy rules generation tool, called *ClassBench* [19], for packet classification benchmarking. The basic observation is that although the real-world rule databases are always not publicly available due to security and confidentiality issues, the rule databases owners may just provide the corresponding profiles, called *seed-files*, of their rule sets instead of the confidential database. And then, based on such seed-files, the researchers may use *ClassBench* to re-generate similar rule set for benchmarking of classification algorithm development.

In light of the idea of ClassBench, in this paper, by thoroughly studies on large amount of associated RFCs and other standards for IPv6, and analysis on the characteristics of both current IPv4 prefix distribution and IPv6 initial prefix distribution, we develop a scalable IPv6 prefix generator, called *V6Gene* for route lookup algorithm benchmarking. V6Gene simulates the process of IPv6 addresses allocation from LIRs to their end users. IPv6 route prefixes are generated from real-world LIR level prefixes (i.e. the LIR level prefixes are treated as *seed-prefixes*), according to the configuration setup by the users. V6Gene is totally flexible and can be easily utilized in other IPv6 based integrated test bed or benchmark systems.

2. Definitions and Terms

2.1 Trie, Prefix Node, Internal Node, Prefix Leaf

A Binary *Trie* is introduced to represent the prefix space, with each node for a possible prefix. The prefix of a route table entry defines a path in the trie ending in some node, which is called the *Prefix Nodes* in this paper. If a node itself is not a prefix node but its descendants include prefix nodes, we call it an *Internal Node*. We name a prefix node as a *Prefix Leaf* if it has no descendants prefix nodes. Fig.1 depicts an example of the definitions introduced above. For simplicity we suppose that the address length is 7bits.

2.2 Prefix Depth and Prefix Level

Depth of a prefix (node) is defined as the number of its ancestor nodes in the prefix trie, and *Depth(i)* of the prefix trie denotes the set of all prefixes (node) with their depths equals *i*. Depth of a prefix node is actually the same of the length of the corresponding route prefix. For instance, in the example of Fig.1 the depths of prefix node A, D, and F are 0,4, and 6 respectively.

For a given prefix node *n*, there may be multi paths from *n* to its multi descendant leaf nodes. Among these paths, let $P_{\max}$ be the one containing the most prefix nodes. The number of prefix nodes (excluding node *n* itself) in $P_{\max}$ is called the *Prefix Level* of prefix node *n*. And |*Level(i)*| denotes the number of prefixes (nodes) with their level equals *i*. It is actually a parameter representing the level of a specific subnet hierarchy. In the example of Fig.1 the level of prefix node A, D and F are 2, 0 and 1 respectively. And |Level(0)|=6, |Level(1)|=2, and |Level(2)|=1 .

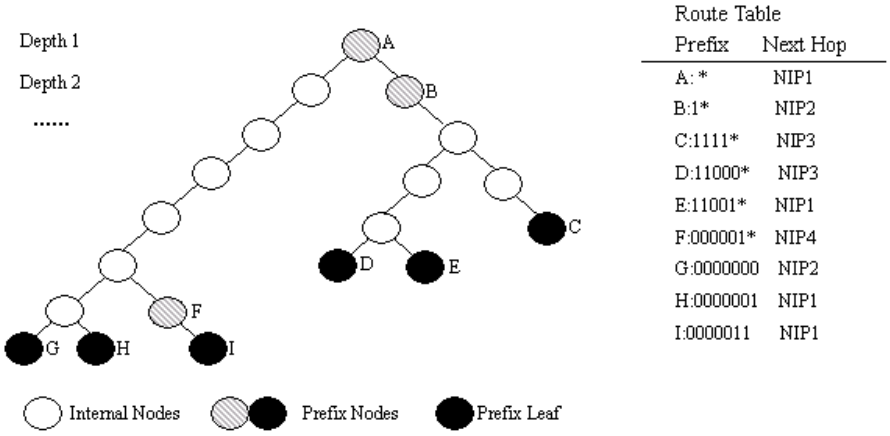
2.3 RGR and GAT

2.3.1 Random Generating Ratio (RGR).

The prefix generation is a random process. As will be introduced in the latter sections, general speaking, the generator random generates IPv6 prefixes from specific seed prefixes collected from real world, simulating the process of IPv6 address block assignments, e.g., from the LIRs to small subscribers. However notice that prefixes may also be allocated from certain newly assigned LIRs, so we also need to generate certain amount of IPv6 prefixes without regarding to the seed prefix file. To make the generator flexible, we introduce the parameter RGR: RGR is defined as the ratio of the number of prefixes to be generated without regarding to the seed prefix file to the number of all prefixes to be generated. RGR represents the degree of relation of the generated prefix table to the seed table.

2.3.2 Generation Accuracy Tolerance (GAT).

GAT is a vector, each sub-value of which is defined as the tolerance of variance between a specific parameter of generated route table and the one setup by the user. The generation outcomes should satisfy all constrains $GAT_i \le | PG_i - PS_i | / PS_i$, where PG_i represents each of the parameters of the generated table, PG_i represents each of the parameters setup by the user. Such parameters include the number of prefixes to generate, the number of total next hop IP addresses in the table, etc. The reason of introducing GAT is that the generated outcome should satisfy more than one constrains setup by the user. Note that some of them may conflict with each others, so it can not be guaranteed that all parameters be achieved, exactly.



Route Table

Prefix	Next Hop
A: *	NIP1
B:1*	NIP2
C:1111*	NIP3
D:11000*	NIP3
E:11001*	NIP1
F:000001*	NIP4
G:0000000	NIP2
H:0000001	NIP1
I:0000011	NIP1

Fig 1. An example routing table and the corresponding binary trie built from it.

3. Features of Route Prefix Distributions

3.1 Real-world IPv4 Prefix Distribution

Study on IPv4 prefix distribution should be very helpful in estimating future IPv6 characteristics, because of two main considerations: 1) The topology of the Internet should not be largely altered during the migration from IPv4 to IPv6; 2) There are many things in common of IPv4 and IPv6 address block allocation/assignment. In what follows, we provide some observation on real-world IPv4 prefix distributions. In order to ensures that the characteristics discussed here are not specific to particular routers or time interval, we pick four typical route tables collected from four famous route service projects [9][11][13][14], which are both spatially and temporally widely distributed, as described in Table I.

TABLE **1**. Four real-world route tables.

Name of Data Base	Date	Number of Prefixes	Number of Next Hop
Mae-West [13]	2001-03	33,960	45
SD_NAP [14]	2001-06	3,935	2
Route View [9]	2003-10	123,384	4
RRC06 [11]	2003-11	131,372	35

3.1.1 Distribution on Prefix Lengths and Depths

As is shown in Fig. 2 (note the logarithmic scale on the y-axis), despite the elapse of time, the prefix length distributions keep a relatively stable form: The historical 24-bit Class C Prefix still dominates the number of entries (about 50% alone); the ratio of prefixes longer than 24-bit is very tiny (less than 1% in each of the four cases); over 90% of the prefixes are between 18-bit and 24-bit.

3.1.2 Distribution on Prefix Level

Fig.3 depicts the distribution on prefix level. We can see that the ratio of prefix population decreases logarithmically with the growth of prefix level (note the logarithmic scale on the y-axis). Most real-world IPv4 route databases are with only five to six prefix levels and the majority (e.g., over 90%) of the prefixes are in Level 0 (i.e., they are prefix leaves).

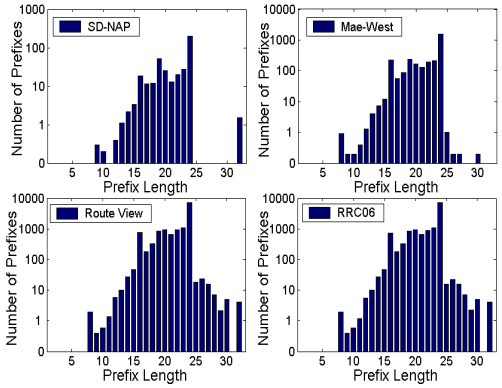


Fig 2. Prefix distributions on prefix length. Please note the logarithmic scale on the y-axis. We find that the distribution is extremely uneven across the scope of prefix length and intervals.

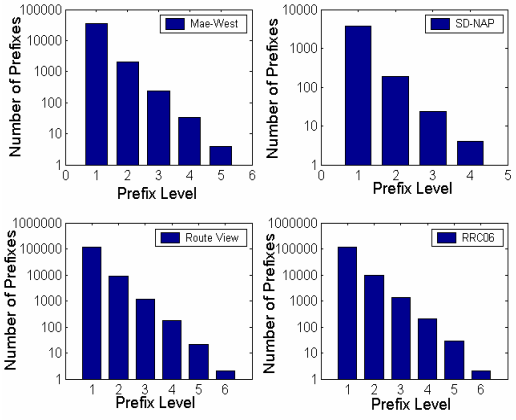


Fig 3. Prefix distributions on prefix level.

3.2 IPv6 Prefix Characteristics

As is mentioned before, IPv6's being in its initiation period leads to the relatively few number of available prefix databases that can be utilized for study [1]. And its current distribution should be quite different from its future patterns, according to our studies on the RFCs and other associated documents. In this sub-section, we first introduce the initial IPv6 prefix distribution characteristics, and then through a thorough survey of the associated RFCs [15-18] and RIPE documents [1], we estimate the "future-like" IPv6 prefix distribution and come to some useful conclusions for developing the IPv6 prefix generator.

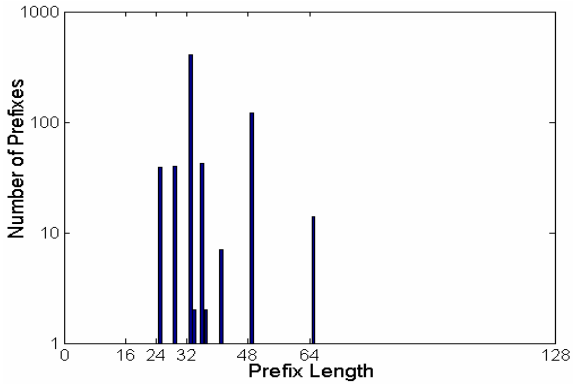
3.2.1 Current (initial) IPv6 prefix distribution.

Fig. 4(a) depicts the IPv6 prefix distribution on prefix length of a real-world IPv6 global route table (Route-View IPv6 route table, Data: 2004-10-3, Size: 680 Prefixes[ii]. [10]). We can see that the majority are '/32' prefixes, which is referred to as the "initial IPv6 allocation blocks" [1]. As mentioned in [1], this kind of IPv6 address blocks are allocated to the LIRs who: 1) plan to provide IPv6 connectivity to organizations to which it will assign '/48's by advertising that connectivity through its single aggregated address allocation; 2) have a plan for making at least 200 '/48' assignments to other organizations within two years (the last information is essential for the IPv6 prefix generation, which will be introduced shortly). Some even shorter prefixes (length from 16-31) were assigned to high-level subscribers according to RFC 2374[15] before it was replaced by RFC 3587 [18]. In RFC 2374 and RFC 2928 [16], IPv6 address blocks were organized in a complex aggregatable hierarchy which includes the TLA (Top Level Aggregation) '/16' blocks, sub-TLA blocks, NLA (Next Level Aggregation) '/48' blocks, SLA (Site Level Aggregation) '/64' blocks and the Interface Level address ('/128').

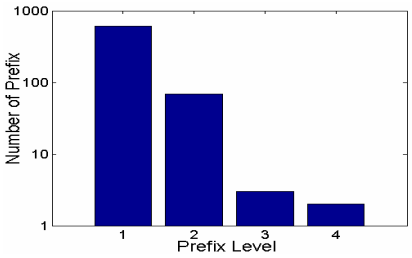
Fig.4(b) depicts the IPv6 prefix distribution on prefix level, where is very similar to the cases of IPv4. And we can see that there are only four prefix levels (i.e., subnet levels). This information will be useful for V6Gene: A specific seed prefix may have only 2-3 levels of children prefixes.

3.2.2 Future IPv6 prefix distribution estimation

RFC 3578 (up-to-dated) replaces RFC 2374 and simplifies the aggregatable IPv6 address hierarchy. Now there are only three levels of prefixes: the Global Routing Prefix (4-48thbits. Note that the 1-3thbits of IPv6 unicast address should be '001'), the Subnet ID (49-64th bits) and the Interface ID (65-128thbits).



(a) On Length



(b) On Level

Fig 4. Real-world (Initial) IPv6 prefix distribution. Please note the logarithmic scale on the y-axis.

According to RFC 3177 (IAB/IESG recommendation on IPv6 address allocation to sites) [17] and RIPE 267 [1], the IPv6 address blocks should be allocated to subscribes following these rules: 1) '/48' in the general case, except for very large subscribes, which could receive a '/47' or multiple '/48s'; 2) '/64' when it is known that one and only one subnet is needed by design; 3) '/128' when it is ABSOLUTELY known that one and only one device is connecting. Also defined by RFC 3177, "the middle 16 bits (i.e., 49-64th bit) of an address indicate the subnet ID", since "the operational benefits of a consistent width subnet field were deemed to be outweigh the drawbacks" [1]. This shows that a standard '/48s' address block can be 'subnetted' into at most 16 levels, indicating the subnet/prefix levels of practical IPv6 route table will not be large. Note that there are even more subnet levels under the CIDR IPv4 address allocation scheme, where the subnet ID may cover from the 9th to 30th bit of the address.

From the related recommendation of RFCs and

[ii] There are totally 6700 prefixes in the original database, however, only 680 of them are unique (since the database may contain identical route prefix announced by different source routers).

RIPE documents introduced above, we come to some useful conclusions as follows, which may be very useful for developing the prefix generator or making a decision of parameter setup:

i. It is obvious but important that there is no prefix with length between 64bit and 128bit (excluding 64bit and 128bit).

ii. The majority of the prefixes should be the '/48s', and '/64s' the secondary majority. Other prefixes would be distinctly fewer than the '/48s' and '/64s'.

iii. Future (or the near future) IPv6 address blocks will be allocated to common subscribers mainly from the current assigned LIRs. This is essential for IPv6 prefix generating.

iv. Though the address length is increase, the levels of subnet/prefix would not be distinctively scaled (e.g. only 4-5 levels), due to consistent width subnet field.

4. V6Gene: The IPv6 Prefix Generator

4.1 The Overall Processing Flow of V6Gene

Fig. 5 depicts the processing flow of V6Gene. Generally speaking, the generating process is actually a simulation of the IP address block allocation process: Remember that most prefixes within the seed prefix set are LIR address blocks, so to allocate new address blocks from the LIRs to their subscribers can be regarded as to generate prefixes from the LIRs/seed prefixes. Given the number of prefixes to be generate, the distribution on prefix length/level and the GAT allowed, the program run iteratively as the alteration of generation and modification, until the constrains are all satisfied.Notice that part of the prefixes is generated randomly, which means they are generation without regarding the seed file. A mergence of the two outcomes should be employed, including a process to remove the redundant/invalid prefixes.

4.2 Details of V6Gene

4.2.1 Initiation

In this step, V6Gene will read in all the configurations, including all the associated parameters (i.e., number of prefixes, number of distinct next-hop, and RAR), distribution objectives (i.e., distributions on prefix length and prefix level), generating constrains (i.e., GAT), and the seed prefix file. Then V6Gene will first check the seed database and prune the invalid/redundant information (i.e., only keep the unique LIR prefixes), for instance as mentioned before, the seed prefix database collected from certain providers may contain identical prefixes announced by different sources. After that, based on the seed file, V6Gene will construct a binary trie, called the S*eed Prefix Trie* (SPT), which will be utilized in the later steps.

4.2.2 Generating

In this step, V6Gene will simulate the IPv6 block allocation process, which includes two parallel phases:

One is the simulation of address blocks allocation from the LIRs to the ordinary subscribers. V6Gene traverse the SPT: whenever come to a seed prefix leaf, it will trigger the generation function. This function generates a specific number of prefixes (indicated by the parameter RGR) according to the given distribution on prefix length and prefix levels; all the generated prefixes should have a same prefix, i.e., the seed prefix. Then the function will randomly assign forwarding information to each prefix generated. Such information includes the forwarding output port#, next hop IP addresses, and so on.

The other phase is the random generation, which is actually to simulate the process of IPv6 block allocation from new LIRs (which do not currently exist) to their subscribers. In this phase, a specific number of prefixes (indicated by the parameter RGR) will be generated without regarding to the seed prefix file. Two sub-steps are included: First to randomly generate LIR level prefixes, and then to generate subscriber level prefix from them.

As mention before, a verification of the outcomes will be deployed after the generations, to make sure that all the prefixes are unique and with proper forwarding information. It is obvious that some of the generated prefixes may be removed in the verification process, and this may lead to dissatisfaction of certain constrains, such as the number of prefixes, etc. So another task of the random generation phase is to make up for such dissatisfaction by additionally generating a number of prefixes. In V6Gene, generation, verification and adjustment are triggered iteratively, until all constrains are well satisfied.

4.2.3 Outputting

In this step, V6Gene will collect all the generated results and output them in compatible format (with the seed prefix file, e.g., the Route-View format [10]).

4.3 Some Discussions

1) V6Gene is totally flexible and scalable. Given different seed prefix sets, or configured with different distributions or constrains, it can generate route prefix table with different scale or for different applications.

2) For target distribution setup, the user may refer to those collected from the IPv4 real-world databases and estimated according to the RFCs. They may also create their specific target distribution.

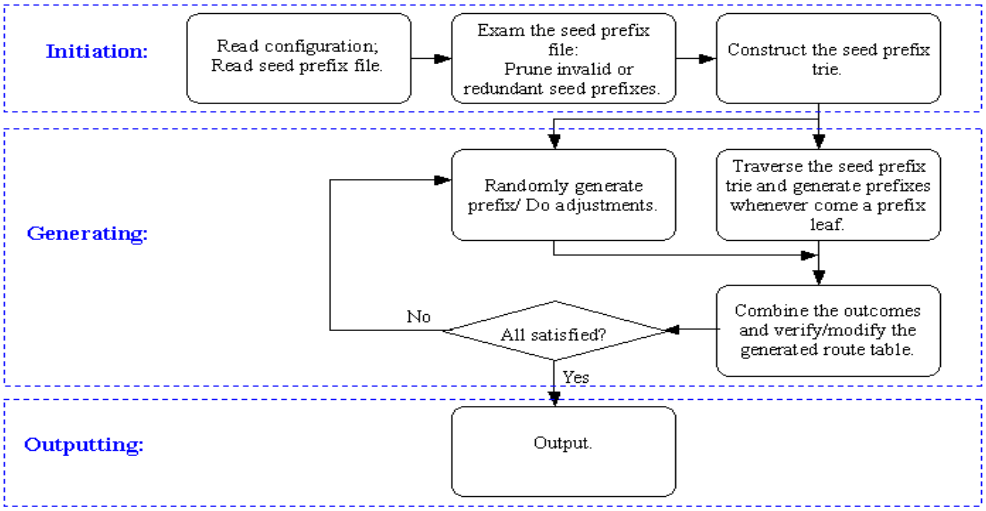


Fig 5. The processing flow of the prefix generator.

3) We implement V6Gene on a 1.8GHz Intel Pentium IV-m laptop. The result shows that it takes only a few (<10) seconds to generate (and verify) over 130,000 IPv6 prefixes. This indicates that the iteration within the process converges fast.

4) All the inputs and outputs (i.e., interface) of V6Gene are files, which shows that it can be easily adopted to collaborate with other algorithm evaluation tools or integrated within a IPv6 benchmark system.

5. Conclusions

In this paper, based on the studies of real-word prefix distributions and the associated RFC documents, we develop a scalable IPv6 prefix generator, called V6Gene. Due to the insufficiency of available real-world IPv6 route databases, V6Gene would be very useful for providing reliable and flexible benchmark for future IPv6 based application designing. V6Gene generates IPv6 route prefixes from the LIR prefixes collected from the real-world, simulating the process of IPv6 address block allocation from the LIRs to their subscribers. It is totally scalable, simple for implementation, and can be easily integrated within other IPv6 benchmark system.

6. Reference

[1] RIPE 267: APNIC, ARIN, RIPE NCC, “IPv6 Address Allocation and Assignment Policy”, Document ID: ripe-267, January 2003.

[2] D.R.Morrison, “PATRICIA -- Practical Algorithm to Retrieve Information Coded in Alphanumeric”, J.ACM, vol. 15, no.4, Oct.1968, pp. 514-34.

[3] S.Nilsson and G.Karlsson, “IP-Address Lookup Using LC-Tries”, IEEE Journal on Selected Areas in Communications, VOL. 17, NO. 6, June1999.

[4] P. Gupta, S. Lin, and N. McKeown, “Routing Lookups in Hardware at Memory Access Speed”, *Proc. of IEEE INFOCOM’98,* San Francisco, April 1998, pp. 1240-1247.

[5] M. Degermark, A. Brodnik, S. Carlsson, S. Pink, “Small Forwarding Tables for Fast Routing Lookups”, *Proc. of ACM/SIGCOMM’97*, Cannes, France, pp. 3-14, Sep. 1997.

[6] H. Liu, “Routing Table Compaction in Ternary CAM”, *IEEE Micro*, 22(1):58-64, January-February 2002.

[7] F. Zane, G. Narlikar, A. Basu, “CoolCAMs: Power-Efficient TCAMs for Forwarding Engines”, *Proc. of IEEE INFOCOM’03*, San Francisco,USA.

[8] K. Zheng, C.Hu, H.Lu, and B.Liu, “An Ultra High Throughput and Power Efficient TCAM-Based IP Lookup Engine”, *Proc. of IEEE INFOCOM*, Hong Kong, China, April, 2004.

[9] IPv4 route database from the Route-view Project (University of Oregon), http://archive.routeviews.org/bgpdata/.

[10] IPv6 route database from the Route-view Project, http:// archive.routeviews.org/route-views6/bgpdata/.

[11] IPv4 route database from the RRCC Project (Routing Registry Consistency Check Project), http://www.ripe.net/rrcc/.

[12] IPv6 route database from the Chinese CERNET BGP VIEW Project, http://bgpview.6test.edu.cn/bgp-view/.

[13] IPv4 route database from the IPMA Project (a joint effort of the University of Michigan and Merit Network), http://www.merit.edu/ipma.

[14] IPv4 route database of the SD_NAP route server (sd-nap-dmz.pch.net), http://archive.pch.net/archive/.

[15] RFC 2374: R. Hinden, M. O'Dell, S. Deering. An IPv6 Aggregatable Global Unicast Address Format. July 1998.

[16] RFC 2928: R. Hinden, S. Deering, R. Fink and T. Hain, "Initial IPv6 Sub-TLA ID Assignments", September 2000.

[17] RFC 3177: IAB, IESG, "IAB/IESG Recommendations on IPv6 Address". September 2001.

[18] RFC 3587: R. Hinden, S. Deering and E. Nordmark, "IPv6 Global Unicast Address Format", August 2003.

[19] D.E. Taylor, J.S. Turner, “ClassBench: A Packet Classification Benchmark”, *Proc. of IEEE INFOCOM* , Miami, USA, March 2005.