Tackling IPv6 Address Scalability from the Root

Mei Wang Ashish Goel Balaji Prabhakar
Stanford University
{wmei, ashishg, balaji}@stanford.edu

ABSTRACT

Internet address allocation schemes have a huge impact on the size of routing tables. Due to the imminent deployment of IPv6 and the rapidly growing number of users worldwide, it is very urgent that we have a good address allocation algorithm for IPv6.

Addresses are allocated by registries to users in the form of prefixes. Such an allocation dovetails nicely with the Longest Matching Prefix algorithm employed by routers to keep the address lookup problem scalable. However, some users inevitably grow in size and need more addresses. The additional allocations often end up creating new prefixes, causing undesirable "address fragmentation."

We further develop a simple algorithm, called Growth-based Address Partitioning (GAP), proposed in Wang (2005). GAP allocates addresses taking into account the possibility that users may grow in the future. However, it makes the unrealistic assumption that this growth rate is known at the time of the initial request. In this paper, we develop an online version of the GAP algorithm and use theory, simulations and real data from the Asia Pacific and the Chinese registries to show that the online version yields an order of magnitude less address fragmentation compared to existing allocation schemes. This is significant for reducing routing table size, increasing scalability and improving the performance of the Internet.

Categories and Subject Descriptors

H.1 [**Information Systems**]: Models and Principles

General Terms

Algorithms, Management, Theory.

Keywords

Internet address allocation

1. INTRODUCTION

IPv6'07, August 31, 2007, Kyoto, Japan.

In a recent meeting of the IETF's Internet Architecture Board dedicated to routing and addressing [1], the scalability of the routing problem was noted as an urgent issue, especially in the context of IPv6. The scalability of route lookups depends on the size and structure of routing tables which, in turn, depend on a good address allocation algorithm [4, 5, 6]. A bad algorithm can cause "address fragmentation," which refers to a single user being represented by multiple *disjoint* prefixes. Address fragmentation severely degrades the performance of longest-prefix-matching-based route lookup algorithms which perform best with a single (unfragmented) prefix per user. Thus, it is imperative we seek a good address allocation algorithm for IPv6 now.

The seemingly infinite address space offered by IPv6 may lead to the perception that a key problem in IPv4 – address fragmentation might no longer be an issue in IPv6. We argue this is not the case. First of all, the total address IPv6 provides is not as large as it seems. Only the first 64 bits out of the total 128 bits for each address are used for routing, the other 64 bits are saved for physical ID, e.g., MAC address of ethernet card. Furthermore, even if the address space seems to be immense today, history has repeatedly demonstrated that a large supply inevitably invites large consumption. Lessons from IPv4 show that it would be too late to wait until the address space becomes crowded before acting. The concern that IPv6 addresses may not be an unlimited resource has led recent studies to project IPv6 address space usage [13]. These studies show that with the current address allocation policies and a conservative projected consumption rate, IPv6 addresses will be exhausted in 60 years *without* taking emerging new applications into account. This concern has led to proposals of remedies and policy changes to make more efficient use of the address space and prolong the lifespan of IPv6 [12, 13]. Another fact worth pointing out is that address allocation is not the only factor that may lead to fragmentation, it is one of several causes such as multi-homing. In this paper, we will focus only on reducing address allocation induced fragmentation.

The Growth-based Address Partitioning (GAP) algorithm was proposed to allocate address space based on each user's growth rate [2]. Assuming this growth rate is known (an assumption we will relax in this paper), it was suggested that rapidly-growing users ought to be allowed more space to grow by having their initial prefix allotments in more open regions of the address space. This is different from an allocation scheme proposed for IPv6, called the Bisection al-

gorithm. This algorithm assigns a new request to the largest currently open region, regardless of their growth rate. Thus, the Bisection algorithm makes the implicit assumption that all users will grow at the same rate. The paper [2] showed that GAP significantly reduces the number of fragmentations relative to the Bisection method. This paper provides stronger evidence via theory and simulations.

In order to use GAP one needs to know each user's growth rate, which is not always available. In this paper, we develop an *online* version of GAP where we estimate the growth rate of each user empirically in an online fashion. Using real data from the Asia Pacific and the Chinese registries, we show that this online GAP can reduce the number of fragmentations by an order of magnitude compared to existing allocations.

The rest of this paper is organized as follows: Section 2 lays out the background for how addresses are allocated nowadays. Section 3 describes the GAP algorithm. Section 4 proves that the GAP is point-wise optimum. Section 5 presents the online version of the GAP scheme presents comparisons with other schemes on real data.

2. HOW ADDRESSES ARE ALLOCATED

2.1 Allocation hierarchy

The IP address allocation hierarchy is shown in Figure 1. At the top of the hierarchy, the whole address pool is controlled by the Internet Assigned Numbers Authority (IANA). IANA allocates large address blocks to each of the Regional Internet Registries (RIRs) serving North America (ARIN), Europe (RIPE), Asia Pacific (APNIC), Africa (AfriNIC), and Latin America and the Caribbean (LACNIC). The regional registries divide up these large address blocks into medium blocks to allocate to Local Internet Registries (LIRs), consisting mainly of Internet Service Providers (ISPs). The ISPs further assign small address blocks to their users, including companies, universities, smaller ISPs, etc.

An address allocation is defined by allocation size and location. Allocation size specifies how large of the address block or the prefix is assigned. Allocation location is where in the address pool this block is allocated to, this is the result of allocation algorithms. Thus, allocation algorithms determine how address space is partitioned. While allocation sizes can be different at each allocation hierarchy layer, a good allocation algorithm can be applied to any layer in the allocation hierarchy and any registry. We will describe the current policies on allocation sizes and allocation algorithms.

2.2 Allocation sizes

The policies on allocation sizes vary for different registries at different levels, as shown in Figure 1. For example, in the current IPv4 policies [8, 9, 10], IANA allocates to RIRs in the unit of /8. Different RIRs adopt their own policies for allocations to LIR/ISPs with unit sizes varying from /10 to /20. The sizes assigned to end users by each ISP also vary greatly. Due to historical allocation schemes, fragmentation is a common problem in IPv4 [4]: one ISP is often left with multiple prefixes.

For the 128-bit IPv6 address, the last 64 bits are assigned to interface ID [3], e.g., an ethernets MAC address. Thus,

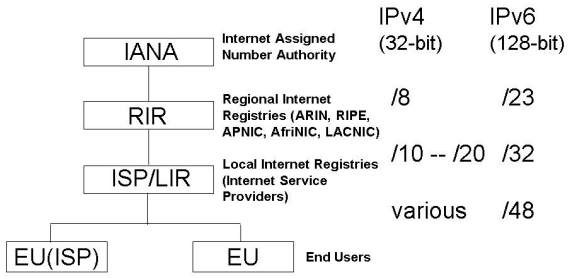

Figure 1: Address allocation hierarchy and policies on sizes.

address allocation only considers the top 64 bits. Allocation policies in terms of size are shown in Figure 1. There is a proposal [15] from major RIRs recommending a Common Address Pool (CAP) in IANA for all RIRs, instead of each RIR keeping a separate pool of addresses for allocation.

Assigning appropriate address sizes at different level has been under extensive discussions [8, 9, 10, 11, 14]. There are many issues involved in the policy decisions on sizes, such as fairness, economical issues, political issues, etc. The focus of the work in this paper will be on allocation algorithms, which we believe deserve more attention.

2.3 Allocation algorithms

There are two address allocation algorithms that are currently deployed in the real world: the sequential scheme and the bisection scheme.

Before going into these two algorithms, let us first define two terminologies that we will use often in this paper.

Address fragmentation: as mentioned before, it refers to non-continuous address blocks for one entity, i.e., more than one prefix representing a single entity in the routing table. Address fragmentation increases routing table size, decreases the efficiency of IP lookup and routing, as well as reduces network scalability.

Collision: when a user runs out of address space for further expansion, due to the proximity of the neighboring users in the address space, this phenomenon is called collision. When collision occurs, the address provider can either find a separate location on the address line or still expand the user's space by carving out the section already occupied by its neighbor. Both options lead to fragmentation. Such practice should be avoided whenever it is possible. A third option is to move either one of the two users to a different address location, which will create much hassle and may not always be viable.

2.3.1 Sequential scheme

Sequential scheme is the method used commonly in the real world practice attributed to its simplicity. In essence, each new request is allocated into the first empty space available from left to right on the address line that fits the requested size.

Sequential is a good scheme if the requested sizes are static, i.e., there is no growth of each user, the total required size for each user is fixed at the beginning of the allocation.

This is obviously not the case in reality. Each entity often requests a relatively small address block at the beginning,

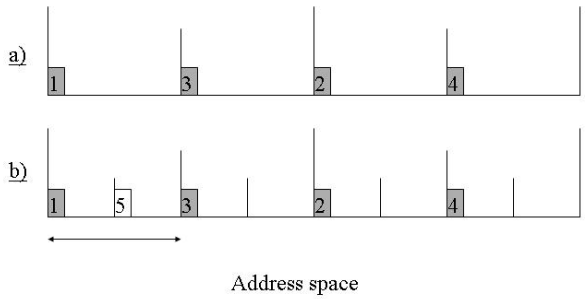

Figure 2: Bisection scheme.

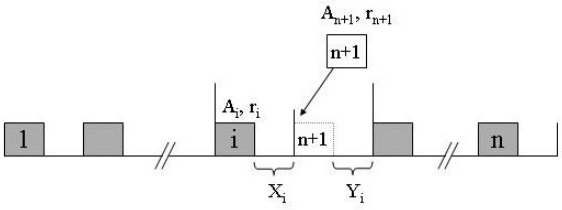

Figure 3: GAP scheme.

then comes back for more as it grows, as we will see in real allocation request patterns in section 5.

2.3.2 Bisection scheme

Bisection scheme is proposed for IPv6 address allocation practice by regional registries [18]. Address blocks are allocated in the following way: each new address block is allocated by evenly splitting the maximum available space on the address line between the existing user and the new user. This leaves maximum possible space for potential growth of both users.

An example using this scheme is illustrated in Figure 2. Each user is given an ID labeling the incoming sequence of the users. The first four users are allocated by dividing the total address space equally into four parts as shown in Figure 2(a). When the fifth user applies for an address block, it is placed to evenly split the section labeled by the arrow in Figure 2(b). The 6th, 7th, and 8th user will be placed sequentially in the spaces after the 2nd, 3rd, and 4th user, respectively. Only after the largest empty slots have been exhausted, will new allocations be assigned to bisect the smaller slots at the next level. A similar technique can be found for dynamic memory allocation in Operating Systems.

This method uniformly separates the allocated blocks to maximize the spacing between users. It treats each user equally. Regardless whether a user is big or small, as far as it comes in as the 5th user in the address requesting sequence, this user is always allocated into the space as shown in Figure 2(b).

In reality, different users are likely to require different sizes in the address space and have different growth rates. Thus this uniform partition may not be the most efficient way to utilize the address space and give each user adequate amount of space to grow, especially when there are a few very fast-growing users. These fast-growers are prone for collisions.

3. GAP: GROWTH-BASED ADDRESS PARTITIONING

To utilize the address space more efficiently and reduce collisions, Growth-based Address Partitioning (GAP) scheme [2] was proposed to make allocations based on the growth of each user. We will review the mechanism of the basic GAP scheme to layout the foundation for the theoretical proof in the next section.

When there are n exiting users, there are at most n possible address location candidates for the $(n+1)^{th}$ user. Instead of treating each user equally, as in the bisection scheme, the GAP scheme evaluates all the options and chooses the location based on the existing users' sizes and their growth-rates, the available space sizes, as well as the size and the growth-rate of the new user. For example, one can choose the location for the new user that maximizes the time before the first collision is projected to take place. Because the prefix bits are binary, the newly allocated block has to start at the middle point of the available address space.

The growth rate information is assumed to be given in the base case of GAP algorithm. We will further discuss growth rate information in Section 5. For now, let us assume we know the growth rates. As shown in Figure 3, we define:

n :	the number of existing users;
$n+1$:	the new user;
A_i :	i=1,...,n,n+1, the allocation for each user;
r_i :	i=1,...,n,n+1, the growth rate of each user;
X_i :	i=1,...,n, the empty space available for the existing user to grow, i.e., the empty space behind each allocated block to bisection point;
Y_i :	i=1,...,n, the empty space available for newly allocated user to grow;

The new location is chosen to maximize the time before anybody runs out of space to grow, i.e., the first collision occurs, using:

$$max\{min[t(X_i, A_i, r_i), t(Y_i, A_{n+1}, r_{n+1})], i = 1, ..., n\}. \quad (1)$$

$t(X, A, r)$ is the time it takes for an allocation with size A and growth rate r to fill up the space X. In other words, the new location is chosen among all available spaces that maximizes the time it takes for either the existing or the new user to run into the boundary. Function $t(X, A, r)$ can be of different forms depending on the definition of growth rate r.

Through both quantitative analysis using theoretical models and simulations, studies show that GAP offers significant advantages over the existing allocation schemes for reducing address fragmentation [2].

4. POINT-WISE OPTIMALITY OF GAP

In this section we will prove that, in an idealized setting, GAP performs at least as well as the bisection scheme in terms of the time before the first collision. We call this point-wise optimality, since this does not make any stochastic assumptions and is true at all times. The proofs in this section complement the experiment results detailed later.

We will consider the case where all requests arrive in some order at time $t = 0$ and the initial size s_i of the i-th request is 1. Let r_i denote the doubling rate of the i-th request, let

N denote the total number of requests, and let M denote the total number of bits in the address space. We will further assume that $N = 2^K$; various extensions of our proofs are possible but beyond the scope of this paper. Let $t_c^{(bisection)}$ be the time till the first collision for bisection.

For the GAP scheme, after $j \leq N$ requests have arrived, define $t_c^{(GAP,j)}$ to be the collision time if there were no more arrivals.

LEMMA 4.1. $t_c^{(GAP,j)} \geq t_c^{(bisection)}$

PROOF. We will prove this lemma using induction.

Base case ($j = 1$): In this case, $t_c^{(GAP,1)}$ is the collision time using GAP with only the first request; this is clearly no worse than bisection.

Inductive hypothesis: assume the following equation holds for j:

$$t_c^{(GAP,j)} \geq t_c^{(bisection)} \tag{2}$$

Inductive step: consider request $j + 1 \leq N$.

We can think of the M-bit address space as a tree of depth M. We will use the term subtree to denote the set of descendants of a point (including itself) in this tree. Consider the 2^K subtrees of size 2^{M-K}. Since $j < 2^K$, at least one of these subtrees must be empty. Let t' be the collision time if request $j + 1$ is assigned to the start of this free subtree of size $2^{(M-K)}$ and if there are no more requests. The proof of the inductive step follows from the following two claims:
Claim 1: $t_c^{(GAP,j+1)} \geq t'$. This claim is easily seen to be true by invoking the single step optimality of GAP.
Claim 2: $t' \geq t_c^{(bisection)}$. The proof of this claim can be divided into two cases.

Case 1: if the $(j+1)$-th request is involved in the first collision for GAP, then one of the two colliding requests (we call it request i) must have grown to a size larger than 2^{M-K}. But the bisection scheme will allocate all requests to subtrees of size 2^{M-K}, and hence the same request i will cause a collision in the bisection scheme before time t', i.e. $t' \geq t_c^{(bisection)}$.

Case 2: if request $j+1$ is not involved in the first collision in GAP, then the collision time for GAP does not change as the result of this new request, and we have $t' = t_c^{(GAP,j)}$ which is at least as large as $t_c^{(bisection)}$ by the inductive hypothesis.

Thus, we proved that the time to collision using GAP is always equal to or longer than that of using bisection under our idealized conditions. □

5. ONLINE GAP WITH REAL DATA

5.1 Online GAP

In the base version of the GAP algorithm, the growth rate information of each user is assumed to be provided at the initial request and remains constant over time. What if this information is not always available? The method we develop is Online GAP: the growth rate of each user is estimated based on the historical data of its requests. Using the current utilized address spaces and growth projections of all existing users, Online GAP allocates address blocks for each new request to prolong the time to collision. For each first-time user, we assume that its growth rate is minumum until the same user later sends in further requests. This first-time user will be allocated based on the size of its request, the existing users' current utilization and projected growth rates. As we will see in the address requesting patterns from real data, most are returning users who come back for more address space multiple times as they grow.

In the following two subsections, we will test the performance of Online GAP. The best way to test it is applying the scheme to real data. However, there are very few real IPv6 data available nowadays, while we have a long history of IPv4 allocations. We believe that a good address allocation scheme can benefit both IPv4 and IPv6. What if we had adopted this proposed allocation algorithm for IPv4 say, six months, one year, or even ten years ago? How would the results compare with that of the existing allocation? Such comparisons can be an objective judgment on how good of an allocation algorithm is.

Following this thought, we conducted experiments using data from existing IPv4 allocations from a regional registry and a country registry. These data are the historical address requesting sequences from various users. They are fed into the Online GAP engine in the identical order as they were actually requested without any knowledge of future allocations. This makes it an apple-to-apple comparison with the existing allocations.

5.2 Results from Asia Pacific data

Asia Pacific Network Information Center (APNIC) is one of the five Regional Internet Registries in the world, in charge of the address allocation and management in the Asia Pacific region.

The allocation data from APNIC span from July 1, 1985 to December 27, 2006. The address blocks are allocated to a region or a large ISP directly. We processed the data based on each country or region as an entity. Each entity has a unique ID. There are 52 entities total in this set of data.

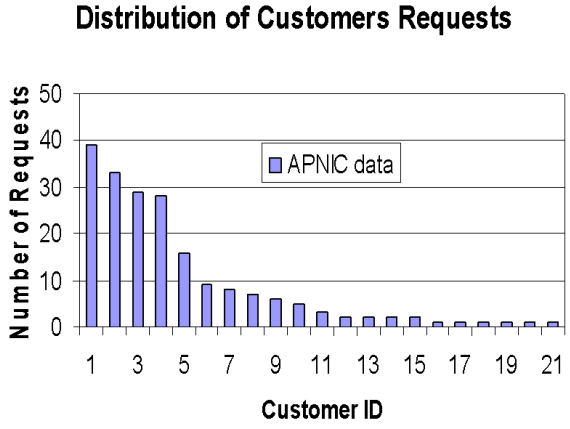

Figure 4: Histogram of user requests for APNIC allocation data.

To show an example, a block of the address 122.0.0.0/7 is used in the experiment shown here. Thus, the total address space in this block is 2^{25}. There are 21 entities assigned in this block. The assignments in this block are very recent: from May 2006 to December 2006. The histogram of the

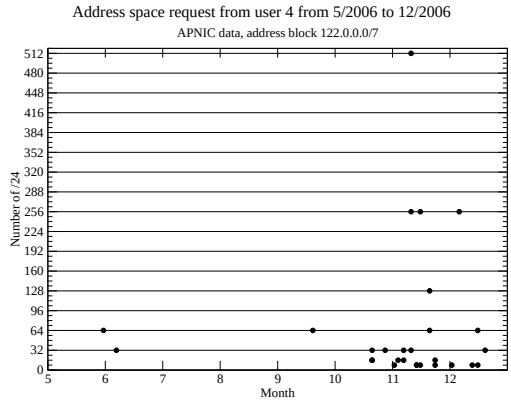

Figure 5: The request pattern of one of APNIC users.

number of requests from each entity is shown in Figure 4, sorted in descending order. The largest number of requests from a single entity during this period of time is 39. Among these 21 entities, 6 of them requested for address blocks only once. The majority of the entities (16 out of 21) requested less than 10 times. As the 80/20 rule dictates, the major part of the address space is assigned to few entities. Figure 5 shows one of the entities' request pattern as an example. This user has requested 28 times during this period of time in this block. The minimum size of allocation is 2^{11}.

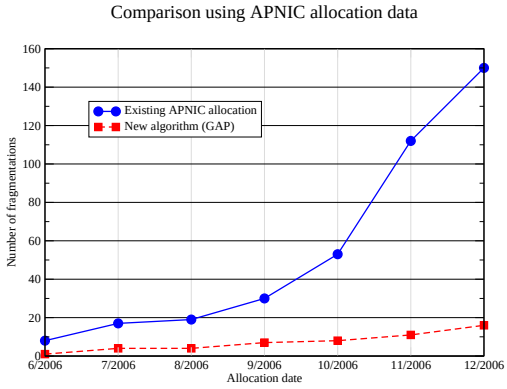

Figure 6: With real allocation data from APNIC, the number of fragmentation is significantly reduced using GAP compared to the existing allocation.

The comparison results in terms of the number of fragmentation are shown in Figure 6. Using the same set of APNIC data, GAP reduces the number of fragmentation from 150 to 16 compared to the current allocations, a 90% reduction of fragmentation.

5.3 Results from China data

China Internet Network Information Center (CNNIC) is a country registry, in charge of IP address allocation and management in China. The same methodology is used for this set of data as for APNIC data. The comparison results in Figure 7 show the number of fragmentation is reduced from 124 to 30 using GAP.

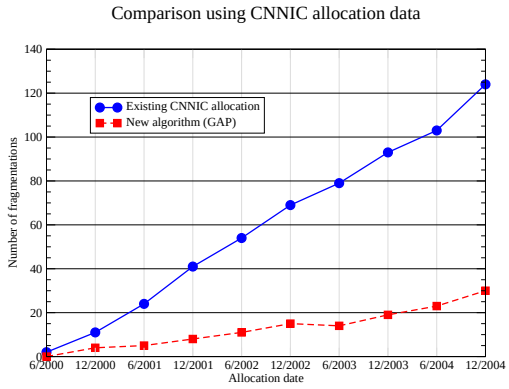

Figure 7: With real allocation data from CNNIC, the number of fragmentation is significantly reduced using GAP compared to the existing allocation.

In both cases, the online allocations using GAP fulfilled the same address allocation requests with a much higher degree of aggregation. All the results we experimented consistently demonstrate similar degree of improvement in reducing fragmentation using GAP.

What is shown here is only one block of addresses, which has limited room to enable the full advantages of GAP. Also the time span we looked at is relatively short. If larger total address space and longer time duration are given, the gain of using GAP can be even more substantial. As more time and space give the algorithm more buffer to better arrange the addresses. This is why it is critical to do the address allocation right for IPv6 from the beginning.

Also, these results are achieved without any additional information taken from users. All the growth rate information are derived empirically from the historical data. If projected growths are provided by individual organization when it requests for addresses, even larger enhancements can be accomplished.

6. CONCLUSIONS AND FUTURE WORK

We have studied the GAP algorithm for Internet address allocation using theory and real data. We have shown that GAP is point-wise optimum compared to the existing Bisection scheme. An online version of the GAP algorithm is developed; it estimates a user's growth rate based on its past requests. Experiments with real data show that the online versrion of GAP significantly reduces address fragmentation relative to other schemes.

As IPv6 is in the process of deployment, entities at different levels of the hierarchy are seeking solutions for efficient address allocation. Since its introduction, GAP has attracted the interest of regional and country IP registries, corporations and universities. In particular, CNNIC is developing a software tool based on the work in this paper that will aid in understanding the goodness of address allocation policies.

Many open questions remain; the following are some of the more important ones.

Since the GAP algorithm's performance is closely related to the accuracy of the growth rate information provided, it is worthwhile considering ways in which the growth rate

can be accurately estimated. Mining historical data (e.g., at a registry, an ISP, or a company) is one way of getting accurate information. A more direct way is to incent the users to provide an estimate of their growth rate along with their initial requests.

Other potential research directions are to design and analyze completely different address allocation methods, such as non-prefix based address allocation, provider independent address allocation, geographical-based address allocation, etc. Such schemes will be more robust to fragmentation and could lead to new address lookup algorithms (i.e. other than longest prefix matching).

Another research direction is to study the scalability of the network by combining addressing with routing, as these are clearly inter-dependent.

7. ACKNOWLEDGMENTS

The authors thank Dr. Wei Mao and his team from China Internet Network Information Center (CNNIC) for providing valuable address allocation data from China.

8. REFERENCES

[1] IETF Internet Architecture Board, Routing and Address Workshop, October 2006.

[2] Mei Wang, A growth-based Address Allocation Scheme for IPv6, Networking, 2005.

[3] R. Hinden, S. Deering. RFC2373 IP Version 6 Addressing Architecture.

[4] T. Bu, L. Gao and D. Towsley. On Characterizing Routing table growth. Proceedings of Global Internet 2002.

[5] Z. Xu, X. Meng, C. J. Wittbrodt, S. Lu, L. Zhang. IPv4 Address Allocation and the Evolution of the BGP Routing Table. UCLA Computer Science Department, 2003.

[6] H. Narayan, R. Govindan, G. Varghese. The Impact of Address Allocation and Routing on the Structure and Implementation of Routing Tables. SIGCOMM 2003.

[7] IPv4 Address Space Report, http://bgp.potaroo.net/ipv4/

[8] M. Kuhne, P. Rendek, S. Wilmot, L. Vegoda. IPv4 Address Allocation and Assignment Policies for the RIPE NCC Service Region, October 2003.

[9] ARIN. IPv4 Policies. http://www.arin.net/policy/ipv4.html

[10] Policies for IPv4 Address Space Management in Asia Pacific Region, March 2004.

[11] APNIC, RIPE, ARIN. IPv6 Address Allocation and Assignment Policy, June 2002.

[12] Andrew Dul. Proposal 2005-5: IPv6 HD Ratio, ARIN, Oct. 2005.

[13] G. Huston. Just how big is IPv6? - or Where did all those addresses go? The ISP Column, July, 2005

[14] G. Huston. Consideration of the IPv6 Allocation Unit Size, Dec 2004.

[15] G. Huston. Allocations vs. Anouncements, The ISP Column, May 2004.

[16] K. Hubbard, M. Kosters, D. Conrad, K. Karrenberg, J. Postel. RFC 2050 Internet Registry IP Allocation Guidelines, November 1996.

[17] IAB, IESG. RFC 3177 IAB/IESG Recommendations on IPv6 address Allocations to Sites, September 2001.

[18] P. Wilson, R. Plzak, A. Pawlik. IPv6 Address Space Management, October 2002.