Internet
Internet-Draft
Expires: May 19, 2003

M. Blanchet
Viagenie inc.

November 18, 2002

A Flexible Method for Managing the Assignment of Bits of an IPv6 Address Block

draft-ietf-ipv6-ipaddressassign-05

Status of this Memo

Copyright Notice

Abstract

This document proposes a method to manage the assignment of bits of an IPv6 address block or range. When an organisation needs to make an address plan for its subnets or when an ISP needs to make an address plan for its customers, this method enables the organisation to postpone the final decision on the number of bits to partition in the address space they have. It does it by keeping the bits around the borders of the partition to be free as long as possible. This scheme is applicable to any bits addressing scheme using bits with partitions in the space, but its first intended use is for IPv6. It is a generalization of RFC1219 [1] and can be used for IPv6 assignments.

Table of Contents

1. Rationale

IPv6 addresses have a flexible structure for address assignments. This enables registries, internet service providers, network designers and others to assign address ranges to organizations and networks based on different criteria, like size of networks, estimated growth rate, etc. Often, the initial assignment doesn't scale well because a small network becomes larger than expected, needing more addresses. But then, the assignment authority cannot allocate contiguous addresses because they were already assigned to another network.

RFC1219 [1] describes an allocation scheme for IPv4 where address space is kept unallocated between the leftmost bits of the subnet part and the rightmost bits of the host part of the address. This enables the network designer to change the subnet mask without renumbering, for the central bits not allocated.

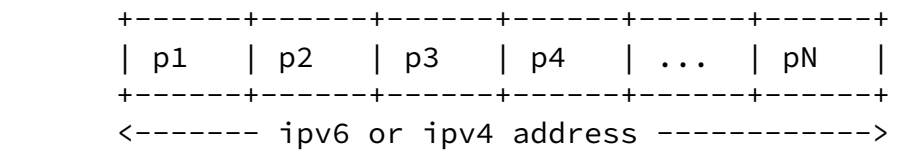
This work generalizes the previous scheme by extending the algorithm so it can be applied on any part of an IP address, which are assigned by any assignment authority level (registries, ISPs of any level, organizations, ...). It can be used for both IPv4 and IPv6.

This document does not provide any recommendation to registries on how to assign address ranges to their customers.

2. Scheme

We define parts of the IP address as p1, p2 , p3, ... pN in order, so that an IP address is composed of these parts contiguously. Boundaries between each part are based on the prefix assigned by the next level authority. Part p1 is the leftmost part probably assigned to a registry, Part p2 can be allocated to a large internet service provider or to a national registry. Part p3 can be allocated to a large customer or a smaller provider, etc. Each part can be of

different length. We define l(pX) the length of part X.



The algorithm for allocating addresses is as follows : a) for the leftmost part (p1), assign addresses using the leftmost bits first b) for the rightmost part (pN), assign addresses using the rightmost bits first c) for all other parts (center parts), predefine an arbitrary boundary (prefix) and then assign addresses using the

center bits first of the part being assigned.

This algorithm grows assigned bits in such way that it keeps unassigned bits near the boundary of the parts. This means that the prefix between any two parts can be changed forward or backward, later on, up to the assigned bits.

3. Description of the Algorithm

This section describes the assignment of leftmost bits, rightmost bits and centermost bits.

3.1 Leftmost

p1 will be assigned in order as follows :

Order	Assignment
1	00000000
2	10000000
3	01000000
4	11000000
5	00100000
6	10100000
7	01100000
8	11100000
9	00010000
...	

This is actually a mirror of binary counting.

3.2 Rightmost

pN (the last part) will be assigned in order as follows :

Order	Assignment
1	00000000
2	00000001
3	00000010
4	00000011
5	00000100
6	00000101
7	00000110
8	00000111
9	00001000
...	

3.3 Centermost

pX (where 1 < X < N) will be assigned in order as follows : (for example, with a 8 bit predefined length l(pX)=8))

Order	Assignment
1	00000000
2	00001000
3	00010000
4	00011000
5	00000100
6	00001100
7	00010100
8	00011100
9	00100000
...	

The bits are assigned using the following algorithm:

1. The first round is to select only the middle bit (and if there is

an even number of bits pick the bit following the center)

2. Create all combinations using the selected bits that haven't yet been created.

3. Start a new round by adding one more bit to the set. In even rounds add the preceding bit to the set. In odd rounds add the subsequent bit to the set.

4. Repeat 2 and 3 until there are no more bits to consider.

4. Example

As an example, a provider P1 has been assigned the 3ffe:0b00/24 prefix and wants to assign prefixes to its connected networks. It anticipates in the foreseeable future a maximum of 256 customers consuming 8 bits. One of these customers, named C2, anticipates a maximum of 1024 customer's assignments under it, consuming 10 other bits.

The assignment will be as follows, not showing the first 24 leftmost bits (3ffe:0b00/24: 00111111 11111110 00001011):

P1 assigns address space to its customers using leftmost bits:


```
10000000  : assigned to C1
01000000  : assigned to C2
11000000  : assigned to C3
00100000  : assigned to C4
...
```

C2 assigns address space to its customers (C2C1, C2C2, ...) using centermost bits:

```
0000010000 : assigned to C2C1
0000100000 : assigned to C2C2
0000110000 : assigned to C2C3
...
```

Customers of C2 can use centermost bits for maximum flexibility and then the last aggregators (should be a network in a site) will be assigned using rightmost bits.

Putting all bits together for C2C3:

```
P1                          |C2      |C2C3
00111111 11111110 00001011 01000000 00001100 00
                              <------->    <------>
                                  growing bits
```

By using this method, P1 will be able to expand the number of customers and the customers will be able to modify their first assumptions about the size of their own customers, until the "reserved" bits are assigned.

5. Implementation

The following Perl code was written by Jocelyn Picard (Jocelyn.Picard@viagenie.qc.ca) and implements this draft. This code is free and without any warranty.

```
#!/sur/bin/perl -w
use strict;

#===========================================================
# allocation(Last Prefix,Number of bits,Method)
#
#  Last Prefix    = last prefix allocated, ex: 3ffe:b00::/48
#  Number of bits = range we want to allocate
#  Method         = method to use: l,c or r (left,center,right)
#
```



```
#  Returns next prefix using selected method
#
# Note: no validation is made
#
#-----------------------------------------------------------
sub allocation {
 my ($ip,$pl)=split('/',shift);
```

```
my ($nbits,$method) = @_ ;
my ($w,@Abits,$abits);

my $i = $ip =~  s/:/:/g;
my $repl= ':0' x (9 - $i);
$ip =~ s/::/$repl/;
$ip =~ s/^:/0:/;

foreach $i (split(':',$ip)) {
 push @Abits, split('',unpack("B16", pack("n", hex($i))));
}
my $sp = int($nbits/2);

for($i=0;$i<$nbits;$i++) {
 if ($method eq "c") {
  $w = ($i % 2) ? $sp - ($i+1)/2: $sp + $i/2;
 }
 elsif ($method eq "r") {
  $w = $nbits -1 - $i;
 }
 else {
  $w = $i ;
 }
  $w += $pl - $nbits;

 if ($Abits[$w] == 0) {
  $Abits[$w] = 1;
  last;
 }
 else {
  return 0 if ($i == $nbits-1);
  $Abits[$w] = 0;
 }
}
$abits = join("",@Abits);
$ip = "";
for($i=0;$i<8;$i++) {
 $ip .= sprintf("%lx",
   unpack("n", pack("B16", substr($abits, $i * 16,16)))) . ":";
}
chop $ip;
```

```
 $ip =~ s/(:0){2,}$/::/;
 return($ip . "/$pl");
}
#================================================================
#
#Usage example: allocation of 100 /48 using "centermost" method
#
my $Prefix = '3ffe:b00::/48';
for(my $i=0;$i<100;$i++) {
 print $Prefix = allocation($Prefix,16,'c'),"\n";
}
```

6. Security Considerations

Address assignment doesn't seem to have any specific security consideration.

7. Acknowledgements

Thanks to Steve Deering, Bob Hinden, Thomas Narten, Erik Nordmark, Florent Parent and Jocelyn Picard for their very useful comments on this work.

References

[1] Tsuchiya, P., "On the assignment of subnet numbers", RFC 1219, April 1991.

[2] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

Author's Address

Marc Blanchet
Viagenie inc.
2875 boul. Laurier, bureau 300
Sainte-Foy, QC G1V 2M2
Canada

Phone: +1 418 656 9254
EMail: Marc.Blanchet@viagenie.qc.ca
URI: http://www.viagenie.qc.ca/

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.