v6ops
Internet-Draft
Expires: August 9, 2008

B. Dickson
Afilias Canada, Inc

February 6, 2008

A Survey and Analysis of Address Allocation Strategies for IPv4 and IPv6

draft-dickson-v6ops-assignment-00

Status of this Memo

Copyright Notice

Abstract

This Internet Draft describes, analyzes, and compares different strategies for allocating addresses, in a way that can be generalized for any power-of-two sized, binary address space. One such strategy is proposed as being "optimal", when viewed from the perspective of space-packing efficiency. This Draft recommends use of this technique as a "Best Current Practice", for both IPv4 and IPv6 address allocations.

Author's Note

This Internet Draft is intended to result in this draft or a related draft(s) being placed on the Informational Track for v6ops.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [2]

Table of Contents

1. Background

In the early days of the Internet, IP addresses were "classful", meaning the size of the prefix was determined by its location in the address space. Subnetting these classful blocks was frequently done, mostly in an ad-hoc manner. When Classless Inter-Domain Routing (CIDR) replaced classful, the Regional Internet Registries - who were given the task of allocating address space - started to track usage. Additional address space was allocated only if a reasonable level of efficiency in assignments was achieved. However, the process of assignments was still piece-meal and largely ad-hoc or occassionally automated with simple tools. With the dawn of widespread and large-scale deployment of IPv6, there is a new opportunity to improve on the internal assignment techiques used by ISPs.

2. Scaling problem examples

The problem space for address assignments is a classical triangle:

Efficiency (the packing problem): Efficiency is measured in terms of availability of unused space. Inefficient use is characterized by fragmentation of unused space. Optimal efficiency is achieved if none of the unused block sizes could be merged, regardless of location in the binary tree.

Expansion (the reservation problem): Expansion is the reservation of unused space adjacent to used space. A block expands when it gets merged with unused space adjacent to it. Example: Used block FEC0::2:0/112 merges with unused block FEC0::3:0/112 to become used block FEC0::2:0/111.

Existence (the renumbering problem) As soon as space is assigned, the recipient becomes a ticking time bomb. It must be presumed that their network is growing, and at some point will need more space. The recipient will not want to renumber an existing assignment, in order to receive a new assignment.

The more room is reserved for growth, the less is available for new assignments, and the lower the apparent global efficiency. This is a zero-sum game, in a finite space. However, the risk-reward, or rather cost-pain, equation pits the assignee against the assigner: any improvement in efficiency which requires a recipient to renumber will face vociferous opposition.

3. Address structures

Both IPv4 and IPv6 address space have the same properties. They are binary addressing schemes. Their respective routing tables use a binary tree (at least conceptually), and walk this tree comparing an address against the routing table until the longest match is found. This means that routes need to be sized to exactly a power of two in size, and assignments (which are the prerequisites of routes) also are powers-of-two in size. Since these properties are the same, we can consider just the generalized problem, involving powers-of-two hierarchies, when comparing methods of assigning address space.

4. Assignment Techniques

There are a number of general techniques for assignment of address space. Each have pros and cons, related to efficiency, expansion, and renumbering. Variants on each can achieve some compromise in the secondary areas, in addition to the primary benefit of the technique.

Sequential Block This technique breaks a large block into smaller blocks, and assigns prefixes of a given size all out of one sub-block, in a sequential fashion. Variants make assignments paired with reserverations ajacent in the same block, by effectively increasing the size of assignment itself. While simple to implement, this technique is neither terribly efficient, nor very flexible for growth.

Bisection This technique initially reserves the whole space for the first recipient. Thereafter, each new recipient is assigned space by splitting, or bisecting, the space reserved for one recipient, reserving half for the original recipient and the other half for the new recipient. Growth occurs within a recipient's reserved space. This technique achieves expansion at the cost of efficiency. Under bisection, unused space is *maximally* fragmented. Variants may make allowances in bisection algorithm based on size of initial assignment. Another problem with bisection is, it is non-deterministic, in that it is sensitive to the sequence in which requests are recieved - particularly when balancing new assignment requests against assignment increases due to growth.

Best Fit It uses the smallest unused block big enough to hold the requested assignment. It then repeatedly bisects that block until the exact fit for the new assignment is achieved. If the smallest is the right size, no bisection is necessary. This technique guarantees no aggregation of unused space is possible after an assignment (if it wasn't possible to aggregate before assignment). It starts with a completely aggregated empty block. Thus, it will always achieve optimal efficiency. It also exhibits the characteristic of not being order-sensitive. Regardless of the sequence of assignments, the same set of empty blocks will result - meaning the measure of efficiency does not depend on order.

In an apples-to-apples comparision, "Best Fit" will have the best efficiency. Other techniques may equal its efficiency, but it is not possible to improve on it.

5. Best Fit - Details and Example

The strategy "Best Fit", works by minimizing the sizes of unused blocks. When possible, exact matches are used, meaning the unused block is the same size as the requested block. When no exact match is found, the smallest block larger than the request, is the block used for splitting into smaller blocks. Each time the larger block is split, only one of the two halves are subsequently re-split. This is repeated until the match is exact. For example, consider a single empty block of size /N, and a request for size M (M > N). /N is split, producing empty blocks /N+1, /N+2, /N+3,... /M, and a second /M. The second /M is assigned in response to the request. If a subsequent request is made for /R, there are three possiblities:

R == M In this case, there is an exact match, of size /M, and no splitting occurs. The result does not differ depending on the order of the requests (since the requested sizes are identical.)

N < R < M In this case, there is an exact match, of size /R, and no splitting occurs. The set of empty blocks, sorted by size, is /N+1 through /R-1, /R+1 through /M.

R > M In this case, there is no exact match. The smallest empty block is /M, which is then split. The set of empty blocks, sorted by size, is /N+1 through /M-1, /M+1 through /R.

The results for the cases "R < M" and "R > M", are symmetric. If we swap which is done first, the resulting sets of empty blocks are the same. By the mathematical proof method of induction, we can see that in all cases, there will never be a case where two members of the set of empty blocks will be the same size, and that the results are not order dependent. This means that "Best Fit" is indeed completely optimal when space efficiency is considered.

6. Security Considerations

Owing to the abstract nature of this document, there are no security considerations.

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

The author wishes to acknowledge the helpful guidance of Joe Abley, Brian Haberman, and Jari Arrko. The author also thanks Marla Azinger, Scott Leibrand, Bob Hinden, and Iljitsch van Beijnum.

9. Informative References

[1] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] Fuller, V., Li, T., Yu, J., and K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC 1519, September 1993.

[4] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

Appendix A. Appendix of FIXME references

FIXME - change "FRONT" components of references above to real values
- ask WG chair, AD, or RFC Editor for help

Appendix B. Allocation Technique Examples

Using tools which do allocations according to either the "best fit" or "bisection" method, and given an empty block of a specific size, and a sequence of requests for allocations, we can observe the results readily. In the following, the allocations are kept in a file, whose structure is described in the comments block. Comments are preserved at the top. The transaction file is a list of address size requests, and the name to associate to the request. We illustrate several scenarios, using the same set of allocation requests in different sequence. The resulting allocation files are shown at various steps, so the differences between methods and the sensitivity to sequence of transactions is clearer. The resulting allocation block file, shows allocations (and optionally reservations, in the case of the bisection method). Each block has the name assigned to the allocated block, or the empty string indicating unallocated space. (Bisection uses reserved space, and does not have "unallocated" space, per se.)

Input Files:

Empty allocation file (start from scratch):

```
# File for storing tree of allocations and free blocks
# default base is 10
# default arrangement is flat (vs dotted or colon separated hierarchy)
#
# format of each line is:
# network/[reservation-]length[,customer]
#
# if no [,customer] label exists, the block is available
# if [reservation-] is specified, the following are true:
#       network/length is allocated to customer
#       network/reservation is tentatively reserved for customer,
#       but can be bisected
universe=/6
0/0
```

Transaction file containing sequential requests for new allocations:

```
# Set of requests (for batch processing of requests for allocations)
# name /size
c1 /5
c2 /6
c3 /3
c4 /6
c5 /4
c6 /3
```

Results for allocation strategy "Bisection":

```
universe=/6
0/3-5,c1
8/3-4,c5
16/3-3,c3
24/3-3,c6
32/2-6,c2
48/2-6,c4

Results for allocation strategy "Best":
universe=/6
0/5,c1
2/6,c2
3/6,c4
4/4,c5
8/3,c3
16/3,c6
24/3
32/1

Additional allocations:
c7 /2
c8 /3
c9 /3
c10 /3

Results for allocation strategy "Bisection":
Unable to allocate prefix size /2 for c7
Unable to allocate prefix size /3 for c10
universe=/6
0/3-5,c1
8/3-4,c5
16/3-3,c3
24/3-3,c6
32/3-6,c2
40/3-3,c8
48/3-6,c4
56/3-3,c9

Results for allocation strategy "Best":
universe=/6
0/5,c1
2/6,c2
3/6,c4
4/4,c5
8/3,c3
16/3,c6
24/3,c8
32/2,c7
```

48/3,c9
56/3,c10

We can see that the requests used up all of the available space, exactly. The strategy "Best" succeeded in using up all the space. The strategy "Bisect" did leave some room for growth for some allocations, but not for others. "Bisect" ultimately fragmented the space too much for allocations that would otherwise have been able to fit.

Author's Address

Brian Dickson
Afilias Canada, Inc
4141 Yonge St,
Suite 204
North York, ON M2P 2A8
Canada

Email: briand@ca.afilias.info
URI: www.afilias.info

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).