Network Working Group
Request for Comments: 1219

P. Tsuchiya
Bellcore

April 1991

On the Assignment of Subnet Numbers

Status Of This Memo

This memo suggests a new procedure for assigning subnet numbers. Use of this assignment technique within a network would be a purely local matter, and would not effect other networks. Therefore, the use of these procedures is entirely discretionary.

This memo provides information for the Internet community. It does not specify an Internet standard. Distribution of this memo is unlimited.

Overview

RFC-950 [2] specifies a procedure for subnetting Internet addresses using a bit-mask. While RFC-950 allows the "ones" in the subnet mask to be non-contiguous, RFC-950 recommends that 1) they be contiguous, and 2) that they occupy the most significant bits of the "host" part of the internet address.

RFC-950 did not specify whether different subnets of the same network may have different masks. This ambiguity was unfortunate, as it resulted in development of routing protocols that do not support different masks; see e.g., RIP [6]. The Gateway Requirements RFC [7] settled the issue in favor of allowing different masks, and therefore future routing protocols may be expected to support this feature; OSPF [3] is an example.

The network administrator must of course determine the mask for each subnet. This involves making an estimate of how many hosts each subnet is expected to have. As it is often impossible to predict how large each subnet will grow, inefficient choices are often made, with some subnets under-utilized, and others possibly requiring renumbering because of exceeded capacity.

This memo specifies a procedure for assigning subnet numbers that eliminates the need to estimate subnet size. Essentially, host bits (mask = 0) are assigned from the least significant bit working towards the most, and subnet bits (mask = 1) are assigned from the most significant bit working towards the least. As subnets grow, more host bits are assigned. As the number of subnets grows, more subnet bits are assigned. While this process does sometimes result

in new subnet masks, no host ever need change addresses.

This technique is not new, but it is also not widely known, and even less widely implemented. With the development of new routing protocols such as OSPF, it is possible to take full advantage of this technique. The purpose of this memo, then, is to make this technique widely known, and to specify it exactly.

This memo requires no changes to existing Internet standards. It does, however, require that the intra-domain routing protocol handle multiple different subnet masks.

Acknowledgments

The author would like to thank Phil Karn, Charles Lynn, Jeff Mogul, and Charles Wolverton for their helpful suggestions. Special thanks go to Joel Halpern for his painstaking debugging of the detailed specification and the examples.

1. Motivation

The Subnetting standard, RFC-950, specifies that the Host part of the formally 2-level Internet address can be divided into two fields, Subnet and Host. This gives the Internet address a third level of hierarchy, and the concomitant firewalls and savings in routing overhead. It also introduces increased inefficiency in the allocation of addresses.

This inefficiency arises from the fact that the network administrator typically over-estimates the size (number of hosts) of any single subnetwork, in order to prevent future re-addressing of subnets. It may also occur if the routing protocol being used does not handle different length subnets, and the administrator must therefore give every subnet an amount of space equivalent to that received by the largest subnet. (This RFC does not help in the latter case, as the technique herein requires different length subnets.)

The administrative hassle associated with changing the subnet structure of a network can be considerable. For instance, consider the following case. A network has three subnets A, B, and C. Assume that the lowest significant byte is the host part, and the next byte is the subnet part (that is, the mask is 255.255.255.0). Assume further that A has subnet 1.0, B has subnet 2.0, and C has subnet 3.0.

Now, assume that B grows beyond its allocation of 254 hosts. Ideally, we would like to simply change B's mask without changing any of the host addresses in B. However, the subnets numerically above

and below B are already taken by A and C. (If say 3.0 was not taken by C, B's mask could be changed from 255.0 (ff00) to 254.0 (fe00). In this case, all of B's existing addresses would still match the new subnet. Indeed, if non-contiguous masks were in use, it might be possible for B to find some other mask bit to change to 0. However, non-contiguous masks are generally not in favor, as they impose limitations on certain forwarding table lookup algorithms. Indeed, RFC-950 discourages their use.)

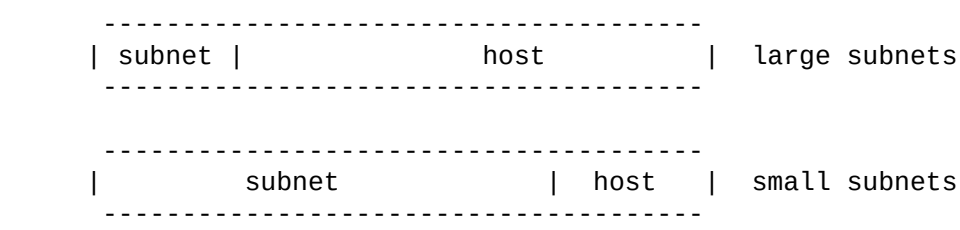
So, the choices available to the network administrator are to 1) form two subnets out of the existing one, or 2) renumber the subnet so that the subnet ends up with a smaller (fewer 1's) mask. Choice 1 can either be accomplished physically or logically. Physically forming two subnets requires partitioning the subnet and inserting a gateway between the two partitions. For obvious reasons, this is not a desirable course of action. Logically forming two subnets can be done by simply assigning another subnet number (say 4.0) to the same subnet, and assigning host addresses under the new subnet. The result of this logical partition is that the hosts with different subnet numbers will not recognize that the others are on the same subnet, and will send packets to the default gateway rather than directly to the host. In fact, this is not such a bad solution, because assuming that the gateway is capable of recognizing multiple subnet numbers on the same subnet, the gateway will simply send the host an ICMP Redirect [4], and subsequent packets will go directly to the host [1] (this may not work correctly on all hosts).

If, however, neither choice is acceptable or possible, then the network administrator must assign a new subnet number to B, thus renumbering the existing hosts, modifying the Domain Name System entries, and changing any other configuration files that have hardwired addresses for hosts in subnet B.

2. A More Flexible and Efficient Technique for Assigning Subnet Numbers

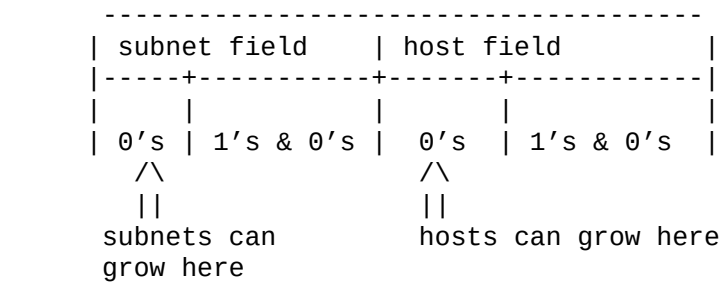
In order to help explain the new technique, we shall show what is wrong with what is currently done now. Currently, most subnets are assigned by splitting the host part of the address in two fields; the subnet field and the host field. Mask bits are one for subnet field bits, and 0 for host field bits. (In all of our addresses, the least significant bit (LSB) is on the right, the most significant bit (MSB) is on the left.)

The subnet field could be different lengths for different size subnets. For instance, say a network had two large subnets and the rest small subnets (by large subnet we mean a large number of hosts). Then the network administrator might assign two types of addresses:

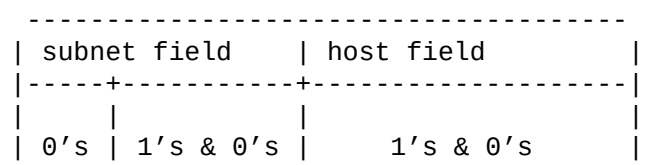


In this case, the full range of subnet numbers would not be available to the small subnets, as the bits in the small subnet that correspond to those in the large subnet could not have the same values as those in the large subnets. For instance, say that the large subnets had 4-bit subnet numbers, and the small subnets had 8-bit subnet numbers. If the large subnets had values 0001 and 0010, then subnet numbers in the range 00010000 to 00101111 could not be assigned to the small subnets, otherwise there will be addresses that would match both subnets.

In any event, a network administrator will typically assign values to the two fields in numerical order. For example, within a given subnet, hosts will be numbered 1, 2, 3, etc. Within a given network, subnets will be numbered 1, 2, 3, etc. The result is that some number of bits on the right side of the subnet and host fields will be ones for some hosts and zeros for others, and some number of bits on the left side of the subnet and host fields will be zeros for all subnets and hosts. The "all zeros" bits represent room for growth, and the "ones and zeros" bits represent bits already consumed by growth.

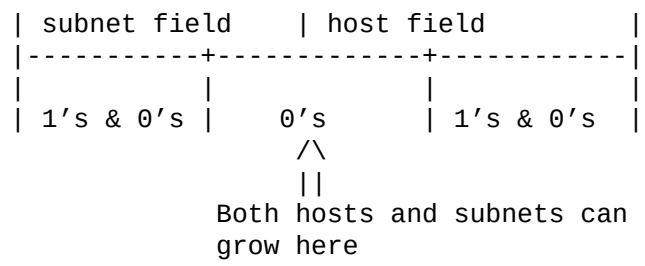


Now, let’s assume that the number of hosts in a certain subnet grows to the maximum allowed, but that there is still room in the subnet field to assign more addresses. We then have the following:



While the host field can no longer grow, there is still room in the address for growth. The problem is that because of where the growth areas are situated, the remaining growth has been effectively reserved for subnets only.

What should be done instead is to assign subnet numbers so that the ones start from the left of the subnet field and work right. In this case we get the following:



Now, both hosts and subnets individually have considerably more growing space than before, although the combined growing space is the same. Since one can rarely predict how many hosts might end up in a subnet, or how many subnets there might eventually be, this arrangement allows for the maximum flexibility in growth.

Actually, the previous figure is misleading. The boundary between the host and subnet fields is being shown in the middle of the growth area. However, the boundary could exist anywhere within the growth area. Note that it is the mask itself that determines where the boundary is. Ones in the mask indicate subnet bits, and zeros indicate host bits. We will show later that in fact the boundary should lie somewhere in the middle. Putting it there minimizes the number of times that the masks must be changed in hosts.

2.1 Specification of the New Technique

Having given the appropriate explanatory material, we can now specify the procedure for subnet number assignment. We need the following definitions:

Host-assigned Bits (h-bits): These are the bits, contiguous from

the right, for which host values, within a given subnet, contain both ones and zeros. Different subnets may have different h-bits.

Subnet-assigned Bits (s-bits): These are the bits, contiguous from the left, which 1) are not h-bits, AND 2) are required to distinguish one subnet from another, AND 3) include all bits to the left of and including the right-most one. Notice that different subnets may have different s-bits.

Growth Bits (g-bits): These are the "all zeros" bits in between the h-bits and s-bits.

s-mask: For a given subnet, the mask whereby all s-bits are one, and all g-bits and h-bits are zero.

g-mask: For a given subnet, the mask whereby all s-bits and g-bits are one, and all h-bits are zero.

Subnet Field: These are the one bits in the subnet mask (as defined in RFC-950). These bits are on the left. The subnet field must at least include all of the s-bits, and may additionally include some or all of the g-bits.

Host Field: These are the zero bits in the subnet mask. These bits are on the right. The host field must at least include all of the h-bits, and may additionally include some or all of the g-bits.

Mirror-image Counting: Normal counting, in binary, causes one bits to start at the right and work left. This is how host values are assigned. However, for subnet assignment, we want the one bits to start at the left and work right. This process is the mirror image of normal counting, where the MSB is swapped with the LSB, the second MSB is swapped with the second LSB, and so on. So, where normal counting is:

```
          0       (reserved to mean "this host")
         01
         10
        011
        100
        101
        :
        :
  11...1110
  11...1111       (reserved to mean "all hosts")
```

and so on, Mirror-image, or MI counting, is:

```
      0       (reserved to mean "this subnet")
      10
      01
      110
      001
      101
        :
        :
      011...11
      111...11        (reserved to mean "all subnets")
```

and so on. If the current MI counting value is, say, 001, the "next" MI value is 101, and the "previous" MI value is 11.

Now we can specify the algorithm. We have the following functions: Initialize(), AddSubnet(), RemoveSubnet(subnet#), AddHost(subnet#), and RemoveHost(subnet#,host#).

Notice that the algorithm is described as though one state machine is executing it. In reality, there may be a root Address Authority (RootAA) that assigns subnet numbers (Initialize, AddSubnet, and RemoveSubnet), and subnet AA, that assign host numbers within a subnet (AddHost and RemoveHost). While in general the AAs can act independently, there are two cases where "coordination" is required between the rootAA and a subnetAA. These are the cases where either the rootAA or the subnetAA "grabs" the last growth bit (in the former case because another subnet has been added, and in the latter because another host has been added). Since it is impossible for the rootAA and a subnetAA to simultaneously grab the last growth bit, either one or the other must do it.

Finally, note that the following C language style notation is used:

```
     &              bit-wise AND function
     ==             is equal to
     !=             is not equal to
     x-mask(X)      the x-mask of X (where x is s or g)
```

Initialize():

Assign the first subnet value to be 0 (the value reserved to mean "this subnet"). This is not assigned to any real subnet.

AddSubnet():

1. Find the lowest non-zero (in MI counting) non-assigned subnet number S such that (S & g-mask(Y)) != (Y & g-mask(Y)) for all existing subnet numbers Y, (Y != S).
2. If all bits in S from the rightmost one bit left are ones, then label all bits to the left of and including one bit position to the right of the rightmost one bit in S to be

s-bits. Else, label all bits to the left of and including the rightmost one bit in S to be s-bits. This prevents the "all ones" value (which is the "all subnets" broadcast address) from being assigned to a subnet. (Since no hosts have been added, the rightmost one bit is a subnet bit.)

3. Label all other bits in the address to be g-bits. (By address, we mean that part of the IP address not including the network number.)
4. Set the subnet mask to include at least all s-bits, and optionally some g-bits. The subnet mask must be contiguous. (Section 2.2 discusses the pros and cons of choosing a mask.)
5. For all existing subnet numbers Y (Y != S):
   51. If (S & s-mask(Y)) == (Y & s-mask(Y)), then:
       511. Change the leftmost g-bit of Y to an s-bit. If the rootAA and YAA (the address authority for Y) are separate AAs, then the YAA must be informed of the change of bit status. If this is the last g-bit, then this change must be coordinated with YAA.
       512. Expand the subnet mask for all hosts in Y if necessary (that is, if the subnet mask no longer includes all s-bits).

RemoveSubnet(S):

1. Consider B to be the bit position of the rightmost s-bit in S.
2. Remove S.
3. For all existing subnet numbers Y:
   31. If the bit in position B is not an s-bit, or if the bit in bit position B is a one, or if the bit in bit position B is a zero and all bits to the left of bit position B are ones, then do nothing (skip steps 32 and 33).
   32. Change the s-bit in position B to a g-bit.
   33. If for any other existing subnet numbers X (X & s-mask(Y)) == (Y & s-mask(Y)), then change the g-bit in position B back into an s-bit for Y. Else, inform YAA that of the change of bit status.

AddHost(S):

1. Create an address A consisting of subnet number S concatenated with zeros.
2. Assign to A the same h-bits, g-bits, and s-bits as the other host addresses.
3. Find the lowest non-zero (using normal counting) non-assigned host number H.
4. If all bits from the leftmost one bit to bit position 0 are ones, then execute steps 5 and 6 using bit position B equals one bit position to the left of the leftmost one bit in H. Else, execute steps 5 and 6 with bit position B equals the leftmost one bit in H. This prevents the "all ones" value

(which is the "all hosts" broadcast address) from being assigned to a host.

5. If bit position B is an s-bit, then the host cannot be added. Skip the remaining steps.
6. If bit position B is a g-bit:
   61. Change the g-bit to an h-bit for all hosts in S. Note that if this is the last g-bit, this change must be coordinated with the address authority assigning subnet numbers (see section 2.2).
   62. Modify the subnet mask in all hosts if necessary.
7. Create a new address A consisting of S concatenated with H
8. Assign A to the host.

RemoveHost(S,H):

1. Remove H.
2. If for all remaining host numbers in S, the value of the bit position of the leftmost h-bit is zero, and there is a zero in at least one of the bit positions to the right of the leftmost h-bit, then for all hosts change the leftmost h-bit into a g-bit.

It is worth noting here that this technique is a 2-level subset of the more general n-level kampai addressing [5]. The main difference here is that n-level kampai results in non-contiguous masks, while 2-level does not. In the description of kampai addressing in [5], g-bits are called a-bits, h-bits are called g-bits, and s-bits are called i-bits.

2.2 An Example

For this example, we assume a class C network, so we will only need to work with 8 bits. We start with 3 subnets, A, B, and C. Our nomenclature is h for h-bit and g for g-bit. Note that h-bits can be one or zero, but g-bits are all zero. The remaining bits are s-bits, but are shown as 1's and 0's according to the subnet number assignment. The space is just to make the addresses and masks easier to read. Finally, we number our bits 0 to 7 from right to left as shown below.

Subnet	Address	Mask
A	10gg ghhh	1111 0000
B	01gg ghhh	1111 0000
C	110g ghhh	1111 0000
	bit 7 bit 0	

We see that each subnet has at most 6 hosts (because of the three h-bits). Notice that we have chosen the masks so that there is room for growth in both hosts and subnets without requiring a mask change.

However, we have generally allowed for more growth in subnets than in hosts because adding new subnets can cause mask changes in existing subnets, while adding new hosts in a subnet only causes that subnet's mask to change.

Further, if a subnet's mask must change, but not all hosts are reconfigured at the same time, then it is less damaging if the not yet reconfigured hosts have too large a mask (too many ones) than if they have too small a mask. This is because with too large a mask, a host may think that another host which is in fact on the subnet is on another subnet. In this case, the host will send packets to the gateway, and will be redirected to the host.

However, with too small a mask, a host may think that another host which is in fact not on the subnet is on the subnet, and will ARP for that host but receive no reply. (Note that broadcasts may fail if all masks do not match.)

Finally, notice that subnet C requires three s-bits instead of just two. This is because with just two, the subnet address of C could be "11" (rather than "110"), which is a broadcast value. Step 2 of AddSubnet checks for this case.

Now, a fourth subnet, D, also with 6 hosts, is added. We get:

Subnet	Addr	Mask
A	10gg ghhh	1111 0000
B	01gg ghhh	1111 0000
C	110g ghhh	1111 0000
D	001g ghhh	1111 0000

Notice that none of the original subnets required a change in any of their status bits. This is because, when D compared its subnet number with the others (step 5 of AddSubnet(), using the s-mask), they were all different. In other words, a router would be able to distinguish an address in D from addresses in A, B, and C.

Next, a fifth subnet, E, is added. We get:

Subnet	Addr	Mask
A	100g ghhh	1111 0000
B	01gg ghhh	1111 0000
C	110g ghhh	1111 0000
D	001g ghhh	1111 0000
E	101g ghhh	1111 0000

Notice that this time, A was forced to change its leftmost g-bit (bit 5) into an s-bit, because bit 5 is needed to distinguish subnet A

from subnet E (step 511 of AddSubnet()). Changing bit 5 into an s-bit prevents hosts from being added to A to the point where bit 5 would be changed into a one (that is, step 5 of AddHost() would fail).

Notice also that if the masks in A, B, and C were originally set to 1100.0000, then the addition of E would have caused A's mask to change to 1110.0000 (Step 512 of AddSubnet()).

Next, 8 hosts each are added to subnets A and C, thus causing the right-most g-bit in each to change to an h-bit.

Subnet	Addr	Mask
A	100g hhhh	1111 0000
B	01gg ghhh	1111 0000
C	110g hhhh	1111 0000
D	001g ghhh	1111 0000
E	101g ghhh	1111 0000

Notice again that no masks have changed. If the masks for A, B, and C were originally set to 1111 1000, then they would have required changing (step 62 of AddHost()).

Next, enough hosts are added to subnet B that all of its remaining g-bits become h-bits.

Subnet	Addr	Mask
A	100g hhhh	1111 0000
B	01hh hhhh	1100 0000
C	110g hhhh	1111 0000
D	001g ghhh	1111 0000
E	101g ghhh	1111 0000

Notice here that the masks in B's subnet had to be changed to accommodate the new h-bits (step 62 of AddHost()). Notice also that if the person assigning host addresses for B (B Address Authority, or BAA) is different than the person assigning network numbers (RootAA), then BAA must coordinate the change of its last g-bit to an h-bit with the RootAA. This allows the RootAA to properly assign additional subnet numbers, as in the next step, where we add another subnet F:

Subnet	Addr	Mask
A	100g hhhh	1111 0000
B	01hh hhhh	1100 0000
C	110g hhhh	1111 0000
D	001g ghhh	1111 0000
E	101g ghhh	1111 0000

F	1110 ghhh	1111 0000

Notice that F received subnet number 1110 rather than subnet number 011 (which is what comes after 101 in MI counting). The reason is that 1) 011 is not distinguishable from B's subnet address using B's mask, and 2) we can't increase B's mask to make it distinguishable because B has already assigned hosts at bit position 5. In other words, when the comparison of step 1 in AddSubnet() was tried on number 011, the two values were equal, and so the next number was tried. In fact, no subnet numbers with 01 in bit positions 7 and 6 can be assigned (unless B loses hosts).

Next, subnet E is removed:

Subnet	Addr	Mask
A	10gg hhhh	1111 0000
B	01hh hhhh	1100 0000
C	110g hhhh	1111 0000
D	001g ghhh	1111 0000
F	1110 ghhh	1111 0000

Notice that this caused subnet A to change an s-bit back into a g-bit. This is because the equality of step 33 of RemoveSubnet() did not hold true for subnet A with respect to the remaining subnets.

References

[1] Braden, R., "Requirements for Internet Hosts -- Communication Layers", RFC 1122, USC/Information Sciences Institute, October 1989.

[2] Mogul, J., and J. Postel, "Internet Standard Subnetting Procedure", RFC 950, USC/Information Sciences Institute, August 1985.

[3] Moy, J., "OSPF Specification", RFC 1131, Proteon, October 1989.

[4] Postel, J., "Internet Control Message Protocol", RFC 792, USC/Information Sciences Institute, September 1981.

[5] Tsuchiya, P., "Efficient and Flexible Hierarchical Address Assignment", TM-ARH-018495, Bellcore, February 1991.

[6] Hedrick, C., "Routing Information Protocol" RFC 1058, Rutgers University, June 1988.

[7] Braden, R., and J. Postel, "Requirements for Internet Gateways", RFC 1009, USC/Information Sciences Institute, June 1987.

Security Considerations

Security issues are not discussed in this memo.

Author's Address

Paul F. Tsuchiya
Bellcore
435 South St.5 South St.
MRE 2L-281
Morristown, NJ 07960

Phone: 201 829-4484

EMail: tsuchiya@thumper.bellcore.com