IPv6 Prefix Primer

by Karl Auer

Notation

The thing to remember, most of all, is that an IPv6 address is just bits. All the various notations are mere conveniences for humans. But let's start with the standard notation (there are others for special purposes, but this is the main one).

An IPv6 address has 128 bits. That is a lot of bits. We need some way of writing them that is more handy, more compressed than, for example:

```
0010 0000 0000 0001 0000 1101 1011 1000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0101 0010
0000 0000 0000 0000 0000 0000 0000 0001
```

The first step is to convert groups of sixteen bits into groups of four hexadecimal ("hex") digits, separated by colons. The leftmost digits represent the highest-order bits. The above binary bits would be represented thus:

```
2001:0db8:0000:0000:0000:0052:0000:0001
```

This is an entirely "legal" representation, a well-formed address. Another, optional, step is to leave out any leading zeros in any group of four digits. That lets us compress the address quite a lot:

```
2001:db8:0:0:0:52:0:1
```

Another step, again optional, is to collapse *one* run of consecutive zero groups into a single empty group. We can do this to only *one* consecutive sequence of zero groups. If we did it to more than one sequence, there would be no way to reconstruct how many zeroes were in each sequence. If we have two sequences, we can collapse either. So now we can write this:

```
2001:db8::52:0:1
```

Handy hint: Think of the double colon as meaning "insert as many zero bits here as are needed to pad this address out to 128 bits in all".

The two optional steps are independent of each other – you can drop leading zeroes and/or compress one or more zero groups to "::".

Prefixes

The high-order bits of an IPv6 address (on the left as the address is written on paper) specify the network, the rest specify particular addresses in that network. Thus all the addresses in one network have the same first N bits. Those first N bits are called the "prefix". We use "/N" to denote a prefix N bits long. For example, this is how we write down the network containing all addresses that begin

with the 32 bits "2001:0db8":

```
2001:db8::/32
```

We use this notation whenever we are talking about a whole network, and don't care about the individual addresses in it. The above can also be read as "all addresses where the first 32 bits are "00100000000000010000110110111000". But hex is easier to read, and *much* shorter.

If we are talking about a specific address, the prefix sometimes doesn't matter, because we have specified every bit in the address. So this is an address – no need to specify a prefix:

```
2001:db8::6:1
```

However, addresses are often written with their proper prefix included, like this:

```
2001:db8::6:1/64
```

The "/64" tells us that this is an address with a 64 bit prefix. Writing addresses this way can result in confusion (see also "Subnetting traps" below), so always make sure it is clear from the context if you have written an address with a prefix length.

Subnets

You can chop any network up into smaller chunks ("subnets") by defining more bits after the prefix. For example, This is a /32 network again, and a /48 subnet of it:

```
2001:db8::/32
2001:db8:0001::/48
```

Notice that the difference is that a further 16 bits have been specified (remember each hex digit represents 4 bits). The original 32 plus the additional 16 make 48 in all.

16 bits can express 65536 distinct values, so there are 65536 distinct /48 networks in any /32 network:

```
2001:db8:0000::/48
2001:db8:0001::/48
[... 65532 subnets omitted ...]
2001:db8:FFFE::/48
2001:db8:FFFF::/48
```

Subnets and networks

The difference between a network and a subnet is really one of convenience and usage; there is no technical difference. An ISP's /32 "network" is a subnet of some larger network registry allocation, just as all those /48 networks are subnets of a /32.

The bigger the prefix, the smaller the network, in the sense that it can contain fewer hosts. A /96 is a network the size of the entire existing IPv4 Internet, because it has 32 bits' worth of host addresses in it (96 + 32 = 128). A /120 has only 8 bits left for hosts, so it's the same size as an IPv4 "class C" subnet: 256 addresses.

Consider the eight /64 subnets below (the ninth just shows how the pattern continues):

```
P:0000::/64 P:0000::/63 P:0000::/62 P:0000::/61
P:0001::/64
P:0002::/64 P:0002::/63
P:0003::/64
```

```
P:0004::/64 P:0004::/63 P:0004::/62
P:0005::/64
P:0006::/64 P:0006::/63
P:0007::/64
P:0008::/64 P:0008::/63 P:0008::/62 P:0008::/61
```

"P" represents the first 48 bits. The table columns show that the addresses from P:0000:: through P:0007:ffff:ffff:ffff:ffff can be treated as eight /64 subnets, four /63 subnets, two /62 subnets, or one /61 subnet.

Subnetting traps

Looking at the table above, we see a gap where we might have expected "P:0001::/63" in the second column on the second line. Why is that? Why can't we have "P:0001::/63"? Read on carefully, it's a bit tricky :-)

Saying we want a /63 is the same as saying that the first 63 bits of any address in that network form the prefix part, and the remainder form the host part. That means that bit 64 is in the host part, because it is not one of the first 63 bits. But in "P:0001::/63", the hex digit "1" is the last digit of the prefix. That's "0001" in binary; this hex digit in that position thus represents bit numbers 61, 62, 63 and 64. In other words, "P:0001::/63" is not valid, because one of the specified prefix bits – bit 64 – actually lies outside the 63 bits of prefix. Or, to put it another way, the number of prefix bits we specified disagrees with the prefix length we specified.

By the same logic, we can't have P:0004::/61. Hex "4" is "0100" in binary, and as the last digit of the prefix represents bits numbers 61, 62, 63 and 64, the last three of which are all in the host part of any address with a 61-bit prefix. Can you now explain why we can't have "P:0006::/62"?

Calculating subnets

How many /64 subnets are there in a /48? To work this out, just deduct the size of the network (in bits) from the size of the subnet, and raise 2 to that power. 64 - 48 = 16, and 2^{16} is 65536, so there are 65536 /64 subnets in a /48 network,

How many /64 networks are there in a /60? Again, deduct the size of the network from the size of the subnet. 64 - 60 = 4, and 2^4 is 16, so there are 16 /64 subnets in a /60.

It works for any size prefix and subnet. How many /93 subnets are there in a /91 network? 93 - 91 = 2, so there are four subnets. How many /12 subnets in a /9 network? 12 - 9 = 3, so there are eight subnets. And so on. How many /112 subnets in a /14 network?

Subnet calculations are easier with hex and IPv6 than with decimal and IPv4. Still, they can be wearisome. There are many good subnet calculators available on the web – just search for "IPv6 subnet calculator". At time of writing, there was a very good crop at http://www.subnetonline.com.

Karl Auer, April 2011

Note: This primer just discusses the basic binary bits. An IPv6 address has more structure than this, and various bit patterns – especially in the first octet – are "special". There are also conventions and standards around subnet sizes. For more comprehensive information, see particularly RFC4291.