Published on **The O'Reilly Network** (http://www.oreillynet.com/)
http://www.oreillynet.com/pub/a/onlamp/2001/05/24/ipv6_tutorial.html
See this if you're having trouble printing code examples

Introduction to IPv6

by Hubert Feyrer
05/24/2001

The future of the Internet

According to experts, the Internet as we know it will face a serious problem in a few years. Due to its rapid growth and the limitations in its design, there will be a point when no more free addresses are available for connecting to new hosts. At that point, no more new web servers can be set up, no more users can sign up for accounts at ISPs, and no more new machines can be set up to access the web or participate in online games -- some people might call this a serious problem.

Several approaches have been made to solve the problem. A very popular approach is to *not* assign a worldwide unique address to every user's machine, but rather to assign them "private" addresses, and hide several machines behind one official, globally unique address. This approach is called "Network Address Translation" (NAT, also known as "IP masquerading"). It has problems, as the machines hidden behind the global address can't be addressed, and as a result of this, opening connections to them -- which are used in online gaming, peer-to-peer networking, etc. -- is not possible. For a more in-depth discussion of the drawbacks of NAT, see [RFC3027].

A different approach to the problem of Internet addresses getting scarce is to abandon the old Internet protocol with its limited addressing capabilities, and use a new protocol that does not have these limitations. The protocol -- or actually, a set of protocols -- used by machines connected to form today's Internet is known as the TCP/IP (Transmission Control Protocol, Internet Protocol) suite, and version 4 currently in use has all the problems described above.

Switching to a different protocol version that does not have these problems of course requires for a "better" version to be available. And actually, there is a better version. Version 6 of the Internet Protocol (IPv6) fulfills future demands on address space, and also addresses other features such as privacy, encryption, and better support of mobile computing.

Assuming a basic understanding of how today's IPv4 works, this article is intended as an introduction to the IPv6 protocol. The changes in address formats and name resolution are covered. After that, it is shown how to use IPv6 -- even if your ISP doesn't offer it -- by using a simple-yet-efficient transition mechanism called 6to4. The goal is to to get online with IPv6, giving example configurations for BSD Unix and Linux.

What good is IPv6?

When telling people to migrate from IPv4 to IPv6, the question you usually hear is "Why?". There are actually a few good reasons to move to the new version:

- Bigger address space
- Support for mobile devices
- Built-in security

Related Reading:

Bigger address space

The bigger address space IPv6 offers is the most obvious enhancement it has over IPv4. While today's Internet architecture is based on 32-bit wide addresses, the new version has 128-bit technology available for addressing. Thanks to the enlarged address space, workarounds like NAT don't have to be used anymore. This allows full, unconstrained IP connectivity for today's IP-based machines as well as upcoming mobile devices like PDAs and cell phones -- all will benefit from full IP access through GPRS and UMTS.

Mobility

When mentioning mobile devices and IP, it's important to note that a special protocol is needed to support mobility, and implementing this protocol -- called "Mobile IP" -- is one of the requirements for every IPv6 stack. Thus, if you have IPv6 going, you have support for roaming between different networks, with global notification when you leave one network and enter the other one. Support for roaming is possible with IPv4 too, but there are a number of hoops that need to be jumped in order to get things working. With IPv6, there's no need for this, as support for mobility was one of the design requirements for IPv6. See [RFC3024] for some more information on the issues that need to be addressed with Mobile IP on IPv4.

Security

Besides support for mobility, security was another requirement for the successor to today's Internet Protocol version. As a result, IPv6 protocol stacks are required to include IPsec. IPsec allows authentication, encryption, and compression of IP traffic. Except for application-level protocols like SSL or SSH, all IP traffic between two nodes can be handled without adjusting any applications. The benefit of this is that all applications on a machine can benefit from encryption and authentication, and that policies can be set on a per-host (or even per-network) basis, not per application/service. An introduction to IPsec with a roadmap to the documentation can be found in [RFC2411], the core protocol is described in [RFC2401].

Changes to IPv4

After giving a brief overview the important features of IPv6, we'll go into the details of the basics of IPv6. A brief understanding of how IPv4 works is assumed, and the changes in IPv6 will be highlighted. Starting with IPv6 addresses and how they're split up, we'll go into the various types of addresses there are, what became of broadcasts; then discuss the IP layer, changes for name resolving, and what's new in DNS for IPv6.

Addressing

An IPv4 address is a 32-bit value that's usually written in "dotted quad" representation, where each "quad" represents a byte value between 0 and 255, for example:

```
127.0.0.1
```

This allows a theoretical number of 2^32 or ~4 billion hosts to be connected on the Internet. Due to grouping, not all addresses are available today.

IPv6 addresses use 128-bit technology, which results in 2^{128} theoretically addressable hosts. This allows a *really* big number of machines to be addressed, and it will fit all today's requirements plus PDAs, cell phones, and even IP phones in the near future without any sweat. When writing IPv6 addresses, they are usually divided into groups of 16 bits written as four hex digits, and the groups are separated by colons. An example is:

```
fe80::2a0:d2ff:fea5:e9f5
```

This shows a special thing -- a number of consecutive zeros can be abbreviated by a single "::" once in the v6 number. The above address is thus equivalent to `fe80:0:00:000:2a0:d2ff:fea5:e9f5` -- leading zeros within groups can be omitted.

To make addresses manageable, they are split in two parts, which are the bits identifying the network a machine is on, and the bits that identify a machine on a network or subnetwork. The bits are known as netbits and hostbits, and in both IPv4 and v6, the netbits are the "left," or most significant bits of an IP number; and the host bits are the "right," or least significant bits:

```
+--------------------+--------------------+
|      n netbits     |   128-n  hostbits  |
+--------------------+--------------------+
```

In IPv4, the border is drawn with the aid of the netmask, which can be used to mask all net/host bits. Typical examples are 255.255.0.0 which uses 16-bit for addressing the network, and 16-bit for the machine, or 255.255.255.0 which takes another 8 bits to allow addressing 256 subnets on, for example, a class B net.

When addressing switched from classful addressing to CIDR routing, the borders between net and host bits stopped being 8-bit boundaries, and as a result the netmasks started looking ugly and became unmanageable. As a replacement, the number of network bits is used for a given address, to denote the border. Thus

```
10.0.0.0/24
```

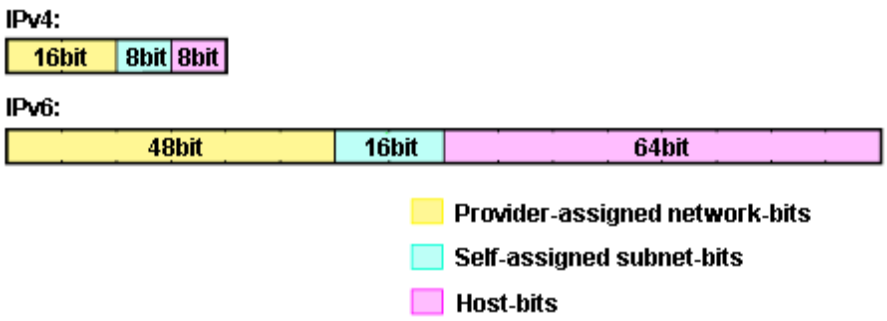
is the same as a netmask of 255.255.255.0 (24 single bits). The same scheme is used in IPv6:

```
2001:638:a01:2::/64
```

tells us that the address used here has the first (left-most) 64 bits used as the network address, and the last (right-most) 64 bits are used to identify the machine on the network. The network bits are commonly referred to as the (network) "prefix", and the prefix here would be 64 bits.

Common addressing schemes found in IPv4 are the (old) class B and class C nets. With a class C network (/24), 24 bits are assigned by your provider, which leaves 8 bits to be assigned by you. If you want to add any subnetting to that, you end up with "uneven" netmasks that are a bit tricky to deal with. Class B networks (/16) are easier cases where only 16 bits are assigned by the provider, and systems that allow subnetting, or splitting of the right-most bits into two parts -- one to address the on-site subnet, and one to address the hosts on that subnet. Usually, this is done on byte (8-bit) boundaries. Using a netmask of 255.255.255.0 (or a /24 prefix) allows flexible management even of bigger networks. Of course there is the upper limit of 254 machines per subnet, and 256 subnets.

With 128 bits available for addressing in IPv6, the scheme commonly used is the same, only the fields are wider. Providers usually assign /48 networks, which leaves 16 bits for a subnetting and 64 host bits.



IPv6 addresses have a similar structure to class B addresses.

Now while the space for network and subnets is sufficient, using 64 bits for addressing hosts seems like a

waste. It's unlikely that you will want to have several billion hosts on a single subnet, so what is the idea behind this?

The idea behind having fixed-width, 64-bit wide host identifiers is that they aren't assigned manually as in IPv4. Instead, v6 host addresses are recommended (not mandated!) to be built from so-called EUI64 addresses. EUI64 addresses are -- as the name says -- 64-bits wide, and derived from MAC addresses of the underlying network interface. For example, with Ethernet, the 6-byte (48-bit) MAC address is usually filled with the hex bits "fffe" in the middle -- the MAC address

```
01:23:45:67:89:ab
```

results in the EUI64 address

```
01:23:45:ff:fe:67:89:ab
```

which again gives the host bits for the IPv6 address.

```
::0123:45ff:fe67:89ab
```

These host bits can now be used to automatically assign IPv6 addresses to hosts, which supports autoconfiguration of v6 hosts -- all that's needed to get a complete v6 IP number is the first (net/subnet) bits. IPv6 also offers a solution to assign them automatically.

When on a network of machines speaking IP, there's usually one router which acts as the gateway to outside networks. In IPv6 land, this router will send "router advertisement" information which clients are expected to either receive during operation or solicit upon startup. The router advertisement information includes data on the router's address, and which address prefix it routes. With this information and the host-generated EUI64 address, a v6-host can calculate its IP number, and there is no need for manual address assignment. Of course, routers still need some configuration.

The advertisement information routers create is part of the Neighbor Discovery Protocol (NDP, see [RFC2461]), which is the successor to IPv4's ARP protocol. In contrast to ARP, NDP does not only do lookup of v6 addresses for MAC addresses (the neighbor solicitation/advertisement part), but also does a similar service for routers and the prefixes they serve, which is used for autoconfiguration of v6 hosts as described in the last paragraph.

Multiple addresses

In IPv4, a host usually has one IP number per network interface -- or even per machine if the IP stack supports it. Only very rare applications like web servers result in machines having more than one IP number.

In IPv6, this is different. For each interface, there is not only a globally unique IP address, but there are two other addresses that are of interest: The link-local address, and the site-local address. The link-local address has a prefix of `fe80::/64`, and the host bits are built from the interface's EUI64 address. The link-local address is used for contacting hosts and routers on the same network only, the addresses are not visible or reachable from different subnets. If desired, there's the choice of either using global addresses (as assigned by a provider), or using site-local addresses.

Site-local addresses are assigned the network address `fec0::/10`, and subnets and hosts can be addressed just as for provider-assigned networks. The only difference is, that the addresses will not be visible to outside machines, as these are on a different network, and their site-local addresses are in a different physical net (if assigned at all). As with the 10/8 network in IPv4, site-local addresses can be used, but don't have to be. For IPv6, it's most common to have hosts assigned a local link and a global IP address. Site-local addresses are rather uncommon today, and is no substitute for globally unique adresses if global connectivity is required.

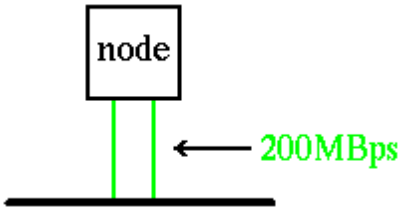
Multicasting

In IP land, there are three ways to talk to a host: unicast, broadcast, and multicast. The most common way to talk to a host is by talking to it directly using its unicast address. In IPv4, the unicast address is the "normal" IP address assigned to a single host, with all address bits assigned. The broadcast address used to address all hosts in the same IP subnet has the network bits set to the network address, and all host bits set to "1" (which can be easily done using the netmask and some bit operations). Multicast addresses are used to reach a number of hosts in the same multicast group, which can be machines spread across the Internet. Machines must join multicast groups explicitly to participate, and there are special IPv4 numbers used for multicast addresses, allocated from the 224/8 subnet. Multicast isn't used very much in IPv4, and only few applications like the MBone audio and video broadcast utilities use it.

In IPv6, unicast addresses are used the same as in IPv4, no surprise there -- all the network and host bits are assigned to identify the target network and machine. Broadcasts are no longer available in IPv6 in the way they were in IPv4, this is where multicasting comes into play. Addresses in the `ff::/8` network are reserved for multicast applications, and there are two special multicast addresses that supersede the broadcast addresses from IPv4. One is the "all routers" multicast address, the others is for "all hosts". The addresses are specific to the subnet, for example, a router connected to two different subnets can address all hosts/routers on any of the subnets it's connected to. Addresses here are:

- ff0X::1 for all hosts and
- ff0X::2 for all routers,

where "X" is the scope ID of the link here, identifying the network. Usually this starts from "1" for the "node local" scope, "2" for the first link, etc. Note that it's perfectly OK for two network interfaces to be attached to one link, thus resulting in double bandwidth:



Several interfaces attached to a link result in only one scope ID for the link.

One use of the "all hosts" multicast is in the neighbor solicitation code of NDP, where any machine that wants to communicate with another machine sends out a request to the "all hosts" group, and the machine in question is expected to respond.

Name resolution

After all the talk about addressing in IPv6, it might make one wonder if there's a proper way to abstract all those long and ugly IPv6 addresses with nice host names as one can do in IPv4, and of course there is.

Host name to IP number resolution in IPv4 is usually done in one of three ways: through a simple table in `/etc/hosts`, by using the Network Information Service (NIS, formerly YP), or via the Domain Name System (DNS).

As of this writing, NIS/NIS+ over IPv6 is currently only available on Solaris 8, for both database contents and transport, using an RPC extension.

Having a simple `address <-> name` map like `/etc/hosts` is supported in all IPv6 stacks. Depending on the implementation, `/etc/hosts` either contains v6 addresses as well, or there will be a

separate file that only maps v6 addresses to names. Examples for `/etc/hosts` that are capable of v6 addresses are the KAME-based IP stacks found in the BSD operating systems (NetBSD, FreeBSD, etc.) and `/etc/ipnodes` used by the USAGI Linux stack and Solaris. Other implementations may use different files.

For DNS, there are no fundamentally new concepts. IPv6 name resolution is done with AAAA records that -- as the name implies -- point to an entity that's four times the size of an A record. The AAAA record takes a hostname on the left side, just as A does; and on the right side, there's an IPv6 address, such as

```
noon        IN    AAAA  3ffe:400:430:2:240:95ff:fe40:4385
```

For the reverse resolution, IPv4 uses the `in-addr.arpa` zone, and below that it writes the bytes (in decimal) in reversed order (the most significant bytes are to the right). For IPv6 this is similar, only hex digits representing 4 bits are used instead of decimal numbers and resource records are also under a different domain, `ip6.int`.

So to have the reverse resolution performed for the above host, you would put a line like this into your `/etc/named.conf` file:

```
zone "0.3.4.0.0.0.4.0.e.f.f.3.IP6.INT" {
    type master;
    file "db.reverse";
};
```

and in the zone file `db.reverse` you put (besides the usual records like SOA and NS):

```
5.8.3.4.0.4.e.f.f.f.5.9.0.4.2.0.2.0.0.0 IN PTR noon.ipv6.example.com.
```

The address is reversed here and written down one hex digit after the other, starting with the least significant (right-most) one with the hex digits separated by dots, as in zone files.

One thing to note when setting up DNS for IPv6 is to take note of the DNS software version in use. BIND 8.*x* does understand AAAA records, but it does not offer name resolution via IPv6. You need BIND 9.*x* for that. Beyond that, BIND 9.*x* supports a number of resource records that are currently being discussed but not officially introduced yet. The most noticeable one here is the A6 record which makes it easier to change the provider or prefix.

In summary, this article talked about the technical differences between IPv4 and IPv6 for addressing and name resolution. Some details like IP header options, QoS, and flows were deliberately left out to simplify this explanation.

Hubert Feyrer works on operating systems, databases, and artificial intelligence at the Fachhochschule Regensburg.

Return to ONLamp.com.