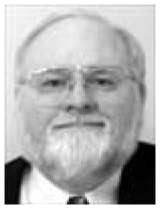
Integrating IPv6 Into Existing Small Enterprise



By Stan O. Barber
Vice President of Engineering Operations
Global IP Network Business
NTT Communications

In this article, I will be discussing some of the basic issues I encountered in actually integrating IPv6 into an existing small enterprise. While the specifics of this work may not be applicable to your enterprise, the steps involved are ones you should consider when auditing IPv6 to your existing IPv4 network.

The Existing Network

It's important before doing any addition to any network to understand the current state of that network and how it is used. There are many structured methodologies available for doing this type of assessment, but the purposes of this short article; I will use a basic inventory and a short narrative to describe the current network. For small networks (like this one), that is probably adequate. For larger networks (particularly multi-site ones), this process will likely have to be more elaborate.

The network is based around Ethernet hubs in a hierarchical tree. The network has no branches that cross-connect to each other so there is no need for the use of spanning-tree to prevent loops. Hosts are connected to the hubs using category 5, 5a or 6 cables and run at speeds between 10 and 1000Mb/sec depending on the capabilities of the host and the hub port. All these hosts fit in one IPv4 /24, so no IPv4 subnetting is used and there are no NATs in this network.

Hosts on the network are of four basic types: Unix-derived (FreeBSD), Windows (XP, 2000 and 98), Apple (Mac OS X 10.3) and HP JetDirect Printers. All of the hosts currently are running IPv4. Some are statically configured and the rest are dynamically configured using DHCP (ISC's 3.0.1) running on a FreeBSD host. The network name servers run BIND 9.1.3 (hosted on the FreeBSD platforms). Two Windows 2000 servers are also running DNS, but this is used to handle the Windows hosts that use Windows Directory Services and the DNS features integrated with it. The network is connected to the rest of the world using a FreeBSD host configured as a router and running Quagga. This site is multihomed so that Quagga (Version 0.98.5) is running BGP with two upstream providers and this router also is providing the firewall for the network using the FreeBSD firewall capabilities. The firewall is configured such that it restricts all inbound traffic to certain well-known application ports on specific hosts while allowing largely unrestricted outbound connectivity over IPv4. For this example, the network uses AS Number 63999.

Support for IPv6 for the Hosts on the Network

The following table outlines the support for IPv6 for the host types on the network.

Host Type	**Operating System**	**IPv6 Support (production quality)**
Unix-like	FreeBSD	Yes, since release 4.0
Mac OS X	Mac OS X (10.3)	Yes
Windows	Window 98	No, not from Microsoft
Windows	Windows 2000 Server	No, not from Microsoft
Windows	Windows XP	Yes, best support with Service Pack 2
JetDirect	Proprietary	Yes, but only in specific models. None of those models are on this network.

Implementation Goals

There are two primarily goals for this implementation. The first is user transparency. By this, I mean that adding IPv6 to the network should not be something that causes the users of the network to change their daily methodologies on network use. They should be able to continue using the applications they normally use just as they normally use them. Where IPv6 may be an option that can used for connectivity within an application, that option should be used where possible, but only in those cases where use of IPv6 is generally transparent to the user (e.g. connection to a web site via IPv6 should behave much the same as it would to connect to the same web site via IPv4 for those web sites that are dual stack enabled).

The second goal is to implement specific new features of IPv6 in a manner consistent with how the network is currently used for IPv4. In this case, autoconfiguration capabilities of IPv6 will be used to dynamically configure those hosts that use dynamic configuration for network access. IPsec usage originating from client hosts within the network will also be permitted. Inbound IPsec is not currently being considered, but the implementation should not prohibit changing that consideration at some future time. Mobile IPv6 implementation is currently outside the scope of this deployment effort.

Deployment Plan

With knowledge of the network, the hosts attached to it and the implementation goals, a deployment plan is the next part of the project to be performed. In order to insure that the hosts are working properly when IPv6 is configured on them, there are a couple of approaches that can be used. The first is to use some sort of testing methodology using a known working testing harness of some kind. This could be some test gear backed by a suite of testing processes. The results of the testing process would empirically indicate that the host was operating properly using IPv6. An alternative is to bring in a live commercial IPv6 network service, get the edge devices properly configured to deliver that service to your network and use it to verify that your hosts are working properly by using a suite of testing processes where the IPv6 network service provides the validation that IPv6 is working. Both approaches have benefits and risks. For this case study, I have chosen the latter.

The First Step

Since a commercial IPv6 network service is being used to validate that IPv6 is working properly, a reputable commercial IPv6 network service should be selected. The provider with the most world-wide experience in providing commercial IPv6 service would be a good choice as such a provider would most likely be able to deliver a consistent, well-proven and good performing product. That provider is NTT Communications since their commercial service has been offered worldwide since 2003 with their original work in IPv6 services started in 1996. Getting the connection was easy, since this network was already using NTT Communications as one of the two IPv4 providers. NTT Communications provisioned this on the existing connection by changing it from being just IPv4 to dual stack IPv4 and IPv6. This change was done transparently with no disruption to the

IPv4 service. NTT Communications provided (per standard ARIN guidelines) a /48 of IPv6 address space for this network to use. Under the currently guidelines, that works out to 65,535 subnets of 18,446,744,073,709,551,616 addresses. This is many more than is currently needed for this project, but it does provide for tremendous growth without having to return to the provider for more space.

An IPv6 Addressing Plan

Since all the hosts fit into an IPv4 /24, a single subnet (number 0) will be used for the statically configured hosts and a second subnet (number 1) will be used for the dynamically configured hosts. For this project, let's assume the prefix allocated from NTT Communications is 2101:418:FFDD::/48. [Note that we are using address space that has not yet been allocated to anyone in these examples.] The first host on the 0 subnet would be 2101:418:FFDD::1 and the first host on the 1 subnet would be 2101:418:FFDD:1::1. To keep things easy, the fourth octet of the IPv4 address will be used as the final component of the IPv6 address for the statically configured hosts.

An Aside on IPv6 Renumbering

While renumbering the statically configured hosts would still be a problematic as it is for IPv4, the addressing strategy for this network under IPv6 will make it possible to change the statically configured hosts to having their addressing information provided by DHCPv6 at some point which would lessen the impact of renumbering should that every be required. Renumbering for hosts using autoconfiguration is built-in to the autoconfiguration mechanism and so the change for these hosts will be trivial for both the user and the network administrator.

Configuring the Router

There are four things that need to be done here. The first is to insure that connectivity for IPv6 is running between the router and the service provider. Second, the LAN interface of the router needs to have an IPv6 address assigned to it as well. Third, BGP must be configured to exchange IPv6 routing information with the provider. Finally, routing advertisements must be configured on the router to facilitate the configuration of hosts using IPv6 autoconfiguration.

Since the router is running on a FreeBSD host configured to support both IPv6 and IPv4 (the default kernel will do this), the IPv6 interfaces are configured as the super-user. For this example, that WAN interface is called wan0 and the WAN address provided by NTT Communications for this end of the WAN connection is 2101:418:7700:5220::2. The far end is 2101:418:7700:5220::1.

```
router# ifconfig wan0 inet6 2101:418:7700:5220::2 2101:418:7700:5220::1
```

In some cases, you may need to add a prefixlen parameter. Check the instructions that came with your WAN interface card or the manual page for the WAN interface card driver for more specific information.

Now, you should be able to ping the other end.

```
router# ping6 2101:418:7700:5220:1
```

If this fails, check to be sure that you have no IPv6 firewall turned on. Generally, if IPv4 is already working on the link, there is nothing else that you need to do.

Now, you are ready to configure the interface LAN interface for IPv6. Let's assume that the lan interface is called lan0 for this example and that the address of the IPv4 interface has a fourth octet of 1. That would make the IPv6 address 2101:418:FFDD::1 for the lan0 interface.

```
router# ifconfig lan0 inet6 2101:418:FFDD::1 prefixlen 48
```

At this point, you may want to configure your firewall to deny any IPv6 traffic access to the rest of your LAN. If you are using a FreeBSD firewall, you can do this by simply running the following commands as the superuser:

```
router# kldload ip6fw

router# sh /etc/rc.firewall6 closed
```

Now, it is time to enable routing for IPv6 on your router. For FreeBSD, this is done with the following command:

```
router# sysctl net.inet6.ip6.forwarding=1

router# sysctl net.inet6.ip6.accept_rtadv=0
```

At this point, the firewall configuration needs to be change in order to finish the rest of the router configuration. Create a file in the `/etc` directory called `firewall6.conf`. The file should contain the following lines:

```
# Allow loopback communications

add 10 pass all from any to any via lo0
# but don't allow communication to IPv6 loopback address from any where
else

add 20 deny all from any to ::1
# and don't allow communication from IPv6 loopback to any where else

add 30 deny all from ::1 to any

# allow ICMPv6 to work properly (mostly works over multicast addresses)

add 40 pass ipv6-icmp from :: to ff02::/16
add 50 pass ipv6-icmp from fe80::/10 to fe80::/10
add 60 pass ipv6-icmp from fe80::/10 to ff02::/16
add 70 pass all from fe80::/10 to ff02::/16

# Stop spoofing

add 00100 deny all from  2101:418:FFDD::/48 to any in via wan0

# Allow TCP through if setup succeeded
add 07000 pass tcp from any to any established

# Allow IP fragments to pass through
add 07100 pass all from any to any frag

# put other TCP specific rules here

# Allow access to our BGP
add 07700 pass tcp from any to any 179 setup

# Allow setup of incoming email

add 07200 pass tcp from any to any 25 setup

# Allow access to our DNS

add 07300 pass tcp from any to any 53 setup
```

```
# allow access for SSH

add 07550 pass tcp from any to any 22 setup

# Allow access to our WWW

add 07600 pass tcp from any to any 80 setup

# Reject&Log all setup of incoming connections from the outside

add 08000 deny log tcp from any to any in via wan0 setup

# allow local lan tcp to anywhere

add 09000 allow any tcp from 2101:418:FFDD::/48 to any in via lan0 setup

add 09100 allow any tcp from 2101:418:FFDD::/48 to any out via wan0

# put other UDP specific rules here

# Allow DNS queries out in the world

add 10000 pass udp from any 53 to any

add 10100 pass udp from any to any 53

# Allow NTP queries out in the world

add 10200 pass udp from any 123 to any

add 10300 pass udp from any to any 123

# Allow UDP from any local lan to anywhere (facilitates traceroute6
working)

add 15000 pass udp from 2101:418:FFDD::/48 to any in via lan0

# Allow ICMPv6 destination unreach

add 20000 pass ipv6-icmp from any to any icmptypes 1

# Allow NS/NA/toobig (don't filter it out)

add 20100 pass ipv6-icmp from any to any icmptypes 2,135,136

# allow echo service and reply

add 20200 pass ipv6-icmp from any to any icmptypes 128,129

# allow time exceeded for traceroute6

add 20300 pass ipv6-icmp from any to any icmptypes 3
```

Now, you can run the following command to update the IPv6 firewall configuration so that router configuration can continue.

Router# sh /etc/rc.firewall6 /etc/firewall6.conf

To configure BGP for IPv6, enter the following after entering Quagga’s configuration mode.

```
ipv6 route 2101:418:FFDD::/48 null0

ipv6 prefix-list IPv6-to-NTT seq 5 2101:418:FFDD::/48

router bgp 63999

neighbor  2101:418:7700:5220::1 remote-as 2914

address-family ipv6

neighbor  2101:418:7700:5220::1 activate

neighbor  2101:418:7700:5220::1 next-hop-self

neighbor  2101:418:7700:5220::1 prefix-list IPv6-to-NTT

network 2101:418:FFDD::/48
```

At this point, exit the configuration mode and enter this command to verify that BGP is up.

```
show ipv6 bgp summary
```

If working there should be a number of routes in the last column between 600 and 700 prefixes. To see the prefixes, enter this command:

```
show ipv6 routes
```

Configuring Autoconfiguration Parameters

By default, Quagga does not do router advertisements. To activate this, enter the following commands in configuration mode:

```
interface lan0

no ipv6 nd suppress-ra

ipv6 nd prefix 2101:418:FFDD:1:/64
```

Finalizing Router Configuration

At this point, router and firewall configuration should be completed. To insure that the configuration persists between router reboots, the following lines should be added to the `/etc/rc.conf` file.

```
ipv6_enable="YES"

ipv6_gateway_enable="YES"

ipv6_ifconfig_lan0="2101:418:FFDD::1/48"

ipv6_ifconfig_wan0="2101:418:7700:5220::2 2101:418:7700:5220::1"

rtadvd_enable="NO"

ipv6_firewall_enable="YES"
```

```
ipv6_firewall_type="/etc/firewall6.conf"
```

You may want to schedule a regular maintenance window to restart the router to be sure that the router does restart with the properly configuration.

Configuring the End Systems for IPv6

The vast majority of the systems that will use static configuration are FreeBSD. These systems can be configured by adding the following lines to the `/etc/rc.conf` file

```
ipv6_enable="YES"

ipv6_ifconfig_lan0="2101:418:FFDD::X/48"

ipv6_defaultroute=”2101:418:FFDD::1/48"
```

Please change the `“X”` in the above line to match the last octet of the IPv4 address for each statically configured host.

For the Macintosh running OS X 10.2 and later, autoconfiguration for IPv6 is on by default. So, your Mac OS X hosts should already be enabled for IPv6.

For Windows XP systems, be sure you have service pack 2 installed. On systems that need to be enabled for IPv6, using administrative privileges type “ipv6 installed” at a command prompt. After the installation process is completed, the system should autoconfigure for IPv6 use.

Enabling DNS for IPv6

Since the main name servers for this network already run BIND 9.1.3, setting up IPv6 support is pretty simple. First, the servers must be configured to provide name service over IPv6. Add the following line to the `options` section of the `named.conf` file that named loads when it starts:

```
listen-on-v6 { any; };
```

Now, restart the name server. You can check `netstat –s –f inet6` to see if there is a server listening on port 53. There should be an entry for TCP and UDP as DNS operates on both types of connections when properly configured.

Next, you need to add the AAAA records for the statically configured servers you have setup. Typically, these entries will look like this:

```
hostname    AAAA  2101:418:FFDD::X ; remember to change the X to the last
octet of the IPv4 address
```

For inverse DNS lookups, you will need to configure a new zone in which each of the individual hexadecimal characters are separated by periods and reversed. An example using `2101:418:FFDD::/48` would have the zone be `D.D.F.F.8.1.4.0.1.0.1.2.ip6.arpa`. In this zone, you would be putting PTR records just as you do for any reverse zone. However, you must expand the :: shorthand in the address to its fully expanded form. An example using `2101:418:FFDD::1` would be like this

```
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 PTR
hostname.domain.name. ;substitute your real domain name for domain.name,
but be sure to include the trailing period.
```

The dynamically configured devices can have statically configured reverse lookups configured or dynamic DNS can be used. The scope of setting up dynamic DNS is beyond that of this article.

Other Applications

Like DNS, other applications made need a small configuration change in order to make use of IPv6. Some applications will detect and automatically use IPv6 if it is available. Typically, you can find more information about this by reading the manual pages for the application or doing a quick internet search for IPv6 and the name of the application.

Epilogue

I believe that this discussion covers the essential steps necessary to add IPv6 to a small enterprise network. While I have personally tested each of these steps to insure they are correct, it is always possible that something may have changed with the software that may invalidate the specific example. Of course, I welcome comments and update to this information. Start planning your integration of IPv6 into your network now!