Pragmatic IPv6 (Part 3)

BY HIROKI SATO

The last column introduced typical examples of IPv6 deployment for a small network with a single uplink router, such as a network in your home. It did not cover some complex configurations involving DHCPv6 and/or PPPoE because we needed some more technical knowledge of IPv6 first. Before diving into such complex cases, let's learn more by configuring your FreeBSD box. This column will specifically cover the following two topics: what to do when you cannot get IPv6 Internet access from your ISP and how to configure IPv6 in essential utilities included in the FreeBSD base system.

Another Way To Get IPv6 Internet Access

The deployment scenarios in the last column assumed that your ISP offers an IPv6 service. In that case, the router between your ISP and your home network has an IPv6 GUA[1]. All you have to do is a configuration of one or more IPv6 addresses and an IPv6 default router address pointing to the router.

So what to do if you get no IPv6 service? While the number of ISPs offering IPv6 service to their end-users is increasing, most are still optional and not their main service as of 2022. One of the ways to get access to the IPv6 Internet is the use of a tunneling connection. Figure 1 shows how "tunneling" works. Network A is an IPv6 network with IPv6 Internet access. Network B is an isolated IPv6 network. These two networks can be connected over the IPv4 Internet if they have an IPv4 address for the endpoints. Tunneling is a protocol translation technique that delivers a packet as a payload of another protocol. Namely, IPv6 packets can be delivered using IPv4 protocol, such as IPv4 TCP and IPv4 UDP. If you are familiar with the word VPN or virtual private network, you can understand tunneling as the technical foundation of VPN. Once the tunneling works, hosts on Network B can access the IPv6 Internet via Network A.

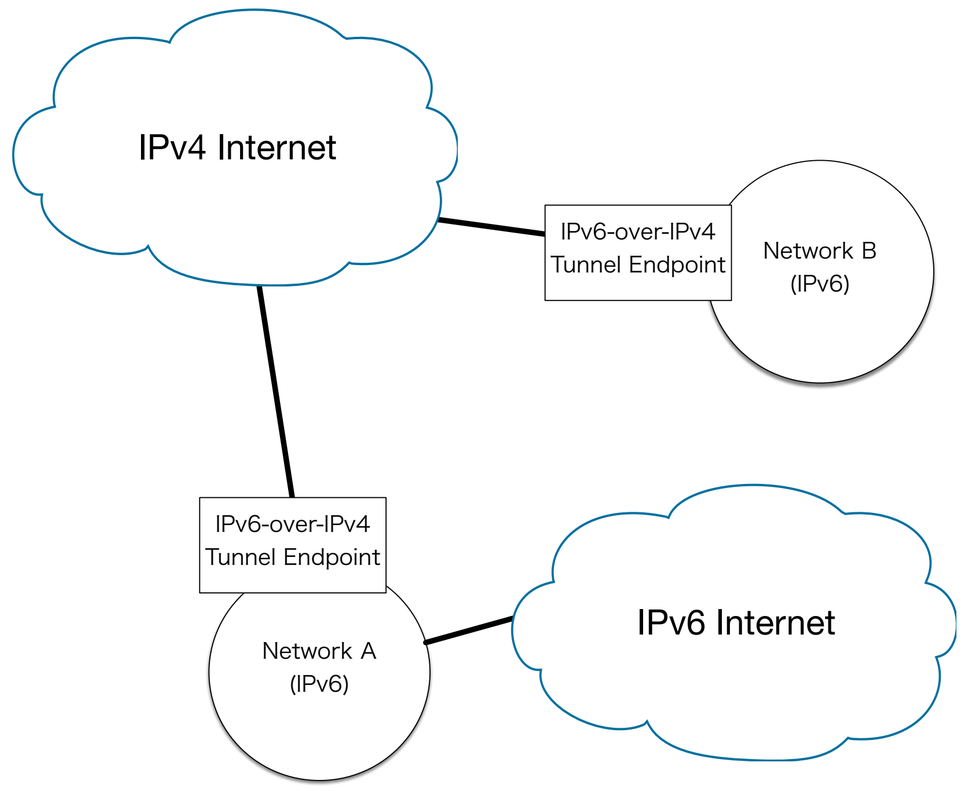

Figure 1: IPv6 networks connected over IPv4 communication

Several tunneling services support IPv6-over-IPv4 tunneling, which you can use for free. One of the reliable services, Hurricane Electric IPv6 Tunnel Broker, will be explained in the following subsections.

Hurricane Electric IPv6 Tunnel Broker

Hurricane Electric is a California-based Internet service provider focusing on Internet transit, data center colocation, and hosting services. Their network coverage is the most enormous scale in the world. It is usually unsuitable for a written article to introduce a specific Internet service like this because it rapidly becomes stale. Still, HE's IPv6 tunneling service has been maintained for over 20 years and been a good testing environment for people with no native IPv6 access. So the author recommends it as a way to get IPv6 Internet access for your experimental purpose. Although you cannot assume that it has the same reliability and performance as your native IPv4 Internet connection by your ISP, HE's service performs well, at least for personal use. And another advantage is that HE offers a /48 IPv6 address prefix. /48 is a recommended prefix length for ISPs so that their end-users can enjoy multiple LANs using the 16-bit[2]. However, many ISPs offer offer only a /64. prefix or two.

Let's see how to configure the tunneling on your FreeBSD box.

Service Account Registration

The tunneling uses two endpoints on your network and in HE's network. Between them, a virtual network will be established over the IPv4 Internet. The endpoint on your side, a FreeBSD box, will act as an IPv6 router.

You must have a global IPv4 address on your FreeBSD box as a prerequisite. The tunneling uses IP packets with protocol number 41. This number is the same as IPv6, so the packet-filtering firewall is unlikely to block them. You must not use IPv4 network address translation by using the IPv4 private address space (10.0.0.0/8, 172.16.0.0/12, or 192.168.0.0/16).

Visit https://www.tunnelbroker.net/ and create your service account. After that, choose "create regular tunnel" in the web interface. Enter the IPv4 address of your endpoint, and select one of the HE's endpoints. All IPv6 packets from your network will be delivered to HE's network once before reaching the IPv6 Internet, so you should choose the nearest one. The author lives in Tokyo, so HE's Tokyo endpoint is the best, for example. By clicking the "create tunnel" button, your endpoint on HE's side will be configured. You need to click it twice because the first click checks if the endpoint works or not by sending IPv4 ping packets from HE's network. Ensure your firewall does not block the incoming ICMP echo request/reply. The IPv4 source address is shown in the list of HE's endpoints.

Network Parameters

Four IP addresses, "Server IPv4 Address", "Server IPv6 Address", "Client IPv4 Address", and "Client IPv6 Address" will be shown on the "Tunnel Details" page.

The Server IPv4 Address is the address for the endpoint on HE's side, and the Server IPv6 Address is for the IPv6 default router that should be used on your network. The Client IPv6 Address is for your endpoint.

Note that you must be aware of two networks — one between the two routers (this network is virtual) and your LAN. See Figure 2 for more details. A yellow circle means an interface. These two networks are separated by your router, which is the endpoint simultaneously in this configuration so that you will get two IPv6 prefixes. You can see the Server and Client IPv6 Address are on the same subnet prefix. That is the virtual network between the two endpoints. The prefix for your IPv6 LAN is shown as "Routed IPv6 Prefixes". You can

configure it to the LAN side on your router. By default, you will get a /64 prefix. By clicking "Routed /48" prefix button, you will get a /48 prefix in addition to it.

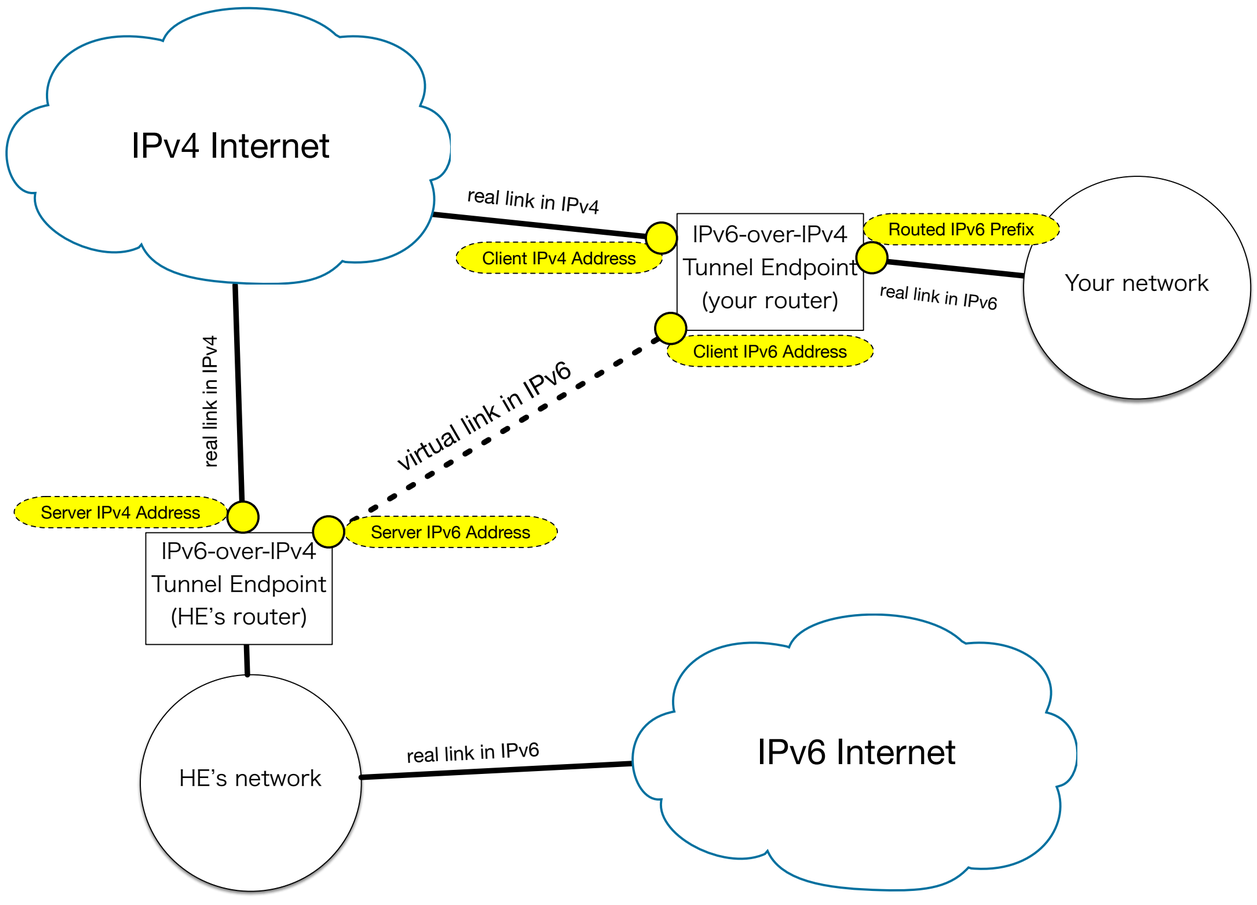

Figure 2: Tunneling between HE's network and your network

Now you are ready to configure your FreeBSD box.

Configuration on Your FreeBSD Box

The Tunnel Details page has "Example Configurations" tab and you can see a configuration example for FreeBSD 4.4 or later like this:

```
# ifconfig gif0 create
# ifconfig gif0 tunnel IPV4-CLIENT IPV4-SERVER
# ifconfig gif0 inet6 IPV6-CLIENT IPV6-SERVER prefixlen 128
# route -n add -inet6 default IPV6-SERVER
# ifconfig gif0 up
```

The keywords like IPV4-CLIENT mean the four parameters. This example is not wrong, but the following version is better:

```
# ifconfig gif0 create
# ifconfig gif0 inet tunnel IPV4-CLIENT IPV4-SERVER
# ifconfig gif0 up
# ping6 ff02::1%gif0
(hit Ctrl-C)
# ifconfig gif0 inet6 IPV6-CLIENT IPV6-SERVER prefixlen 128
# route -n add -inet6 default IPV6-SERVER
# ifconfig bge0 inet6 ROUTED-IPV6-PREFIX/64
# sysctl net.inet6.ip6.forwarding=1
```

The "**gif0**" is an instance of **gif(4)** pseudo-interface network driver on FreeBSD. An IPv6 packet is encapsulated in an IPv4 packet[3] as shown in Figure 3. A rectangular represents a bit sequence from left to right. An IPv4 packet associated with a transport protocol, such as TCP or UDP, typically consists of an IPv4 header, a transport-layer header, and the payload (data). The three parts are concatenated to form a single packet and the packet is sent over

the network. In the same way, an IPv6 packet with the same transport protocol consists of the three parts. The difference is the first element.

If we consider an IPv6 packet as data for an IPv4 packet, we can send the IPv6 packet over IPv4 network. On the sender side, a router builds an IPv4 packet that has an IPv6 packet inside, and on the receiver side, another router extracts the IPv6 packet. This occurs on HE's router and your router when using **`gif(4)`** interface. In practice, encapsulation of an IPv6 packet into an IPv4 packet is done by prepending an IPv4 header to the IPv6 packet, as shown in Figure 3 because the transport-layer header is almost the same as each other.

To configure this interface, you need two sets of addresses. First, you have to create a new **`gif(4)`** interface by using **`ifconfig gif0 create`**. Then the tunneling can be configured by using "**`tunnel`**" keyword in the **`ifconfig(8)`** utility with two addresses for the endpoints. The first address is yours, and the second address is HE's.

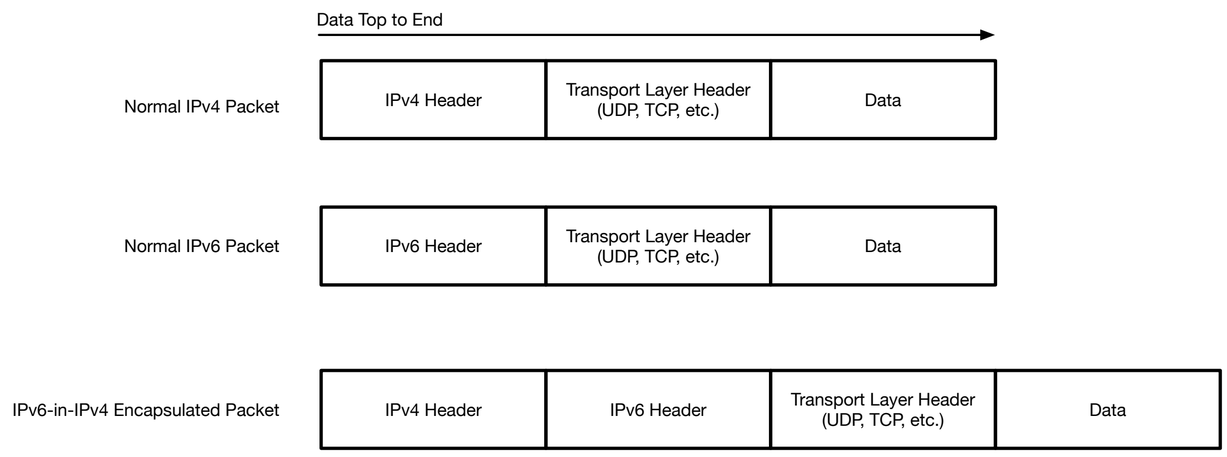

Figure 3: Encapslation used in `gif(4)` interface

After that, a virtual network is established. Note that this virtual network has no connection state, such as "connected" or "disconnected". Packets are just sent as "IP datagram" between two IPv4 addresses on demand. An IPv6 packet is encapsulated, as shown in Figure 3, and sent as an IPv4 packet to the HE's endpoint, and it will be decapsulated in the HE's network. While the IPv4 packets may be lost somewhere on Internet, the error recovery is made in the upper-layer protocol, namely the IPv6 packet inside the IP datagram. Addresses for the tunneling are often called "addresses for the outer protocol". The outer protocol, in this case, is IPv4.

To check if the virtual network works, you can send an IPv6 ping. After the second line of the example configuration, enter **`ifconfig gif0 up`** and try an IPv6 ping by using the automatically-configured LLA[4] of the **`gif0`**:

```
# ifconfig gif0 create
# ifconfig gif0 tunnel IPV4-CLIENT IPV4-SERVER
# ifconfig gif0 up
# ping6 ff02::1%gif0
PING6(56=40+8+8 bytes) fe80::80c8:4a75:123a:8a32%gif0 --> ff02::1%gif0
16 bytes from fe80::80c8:4a75:123a:8a32%gif0, icmp_seq=0 hlim=64 time=0.084 ms
16 bytes from fe80::4a52:2e06%gif0, icmp_seq=0 hlim=64 time=4.765 ms(DUP!)
16 bytes from fe80::80c8:4a75:123a:8a32%gif0, icmp_seq=1 hlim=64 time=0.081 ms
16 bytes from fe80::4a52:2e06%gif0, icmp_seq=1 hlim=64 time=2.738 ms(DUP!)
^C
--- ff02::1%gif0 ping6 statistics ---
2 packets transmitted, 2 packets received, +2 duplicates, 0.0% packet loss
round-trip min/avg/max/std-dev = 0.081/1.917/4.765/1.970 ms
```

The virtual network is working fine if you get the (`DUP!`) response. This is because you should have a reply from the router on the HE's side and another from the `gif0` interface itself. If you get no response from another side of the endpoint, double-check the configured IPv4 addresses and packet filters located between the two.

Then you need to configure actual IPv6 addresses for communication. This can be done using the `ifconfig(8)` utility, the sixth line of the example configuration. This line configures a point-to-point network that has only two IPv6 nodes. At this point, you can send an IPv6 ping to *IPV6-SERVER*.

The above configuration is sufficient if you just want access to the IPv6 Internet from the box. Your FreeBSD box can now work as an IPv6 host node. To make it a router for your /64 and /48 LANs, you have to configure the default IPv6 route to *IPV6-SERVER* and enable the IPv6 packet forwarding feature. The following part in the example does it:

```
# route -n add -inet6 default IPV6-SERVER
# ifconfig bge0 inet6 ROUTED-IPV6-PREFIX/64
# sysctl net.inet6.ip6.forwarding=1
```

This configuration assumes that the interface facing your LAN is `bge0`. *ROUTED-IPV6-PREFIX* is your prefix assigned by HE's service. You should have both /64 and /48. You can configure both on the same interface or on a different interface.

There is one thing you need to think about here. What IID[5] should be used? That is the router's address on the LAN.

You can see that IPv6 addresses used for the tunneling interface were `::1` and `::2` in their IIDs. That is one of the strategies. However, the author recommends using the all-zero address for simplicity. If you get `2001:db8::/48` from HE, pick up a /64 network in that range, say `2001:db8:1::/64`, and configure it as the router address. The IID is all-zero. This looks scary because an all-zero host address in IPv4 has a special meaning. Although it has never been clearly defined in RFC or other specifications, some traditional network implementations have recognized it as a broadcast address, not a unicast one. one. While the all-zero host address works with no problem on FreeBSD, the author guesses that you have avoided it in practice if you learned TCP/IP network in 20th century. In IPv6, the all-zero IID is completely valid. There is another reason why the all-zero address is suitable for a router, but it will be covered in the later columns.

After configuring the default router and the router address on the LAN, you can configure the hosts on the LAN. See also the last column about what options you can take. Running `rtadvd(8)` on `bge0` to enable automatic configuration is highly recommended.

Configuration in /etc/rc.conf

Once you confirm if your IPv6 tunnel works, put the configurations by hand into `/etc/rc.conf`:

```
cloned_interfaces="gif0"
ifconfig_gif0="inet tunnel IPV4-CLIENT IPV4-SERVER"
ifconfig_gif0_ipv6="inet6 IPV6-CLIENT IPV6-SERVER prefixlen 128"
ipv6_defaultrouter="IPV6-SERVER"
ipv6_gateway_enable="YES"
ifconfig_bge0="inet6 ROUTED-IPV6-PREFIX/64"
```

Before rebooting, you can check if it works by using **service(8)** command like this. Two ping6 are to check the tunnel endpoint reachability and the default router configuration:

```
# service routing start
# service netif restart gif0
# ping6 ff02::1%gif0
# ping6 IPV6-SERVER
```

That's all. You can now build IPv6-capable Internet servers on your LAN because the configured IPv6 addresses are reachable from the IPv6 Internet.

There is one additional comment about **ipv6_defaultrouter**. While the above example uses *IPV6-SERVER*, it can also be the router's LLA on HE's side, you can see by sending a ping, or you can set it to "**-interface gif0**". The former is not recommended because the LLA can change when HE changes its equipment. It is recommended to use an LLA for a static route only when it is manually configured and under your control. The latter is another way to simplify the configuration. Because **gif0** is configured as a point-to-point interface, the route to the router on the HE's side can be specified by using the interface name on your side. Doing this allows you to use a consistent configuration even if the address IPV6-SERVER is changed.

Using IPv6 in Essential Utilities

Another topic in this column is practical configuration examples for software in the FreeBSD base system. After you obtain access to the IPv6 Internet, let's make software use of IPv6.

General Rules and Pitfalls

From the user's point of view, the most significant difference is the notation of an address. Except for that, TCP and UDP work just like IPv4. So you can get started by replacing an IPv4 address in a configuration file with one in IPv6. The first column covered OpenSSH as an example. Let's see what change is required.

Configurations for the sshd(8) daemon is stored in **/etc/ssh/sshd_config**. You can edit it directly, but modifying the configuration using command-line flags allows you to keep the default configuration file intact. For example, you can put the following lines into **/etc/rc.conf** to change some of the configurations partially:

```
sshd_enable="YES"
sshd_flags=" \
  -oPort=22 \
  -oUsePAM=no \
"
```

Let's go back to the IPv6 configuration topic. The **sshd(8)** daemon listens to **tcp/22** in both IPv4 and IPv6 by default. You can see the following lines in **/etc/ssh/sshd_config**. All lines are commented out, but it means they are enabled by default:

```
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

"`::`" is an IPv6 address. This all-zero address is called "unspecified IPv6 address". Note that this is the all-zero subnet prefix and the all-zero IID simultaneously, not the same as all-zero IIDs in the previous section. This particular IPv6 address means any of IPv6 addresses. So the **`sshd(8)`** daemon listens to all of the IPv6 addresses configured on the box.

Like this, some software simply adopt raw IPv6 addresses in the configuration file. In that case, you can write an IPv6 address instead of IPv4 without further consideration. The **`sshd(8)`** is smart enough to distinguish which address family is used. If you want to restrict IPv6 addresses for the **`sshd(8)`** daemon, you can put a **`-oListenAddress`** option:

```
sshd_enable="YES"
sshd_flags=" \
  -oPort=22 \
  -oUsePAM=no \
  -oListenAddress=2001:db8::1 \
"
```

Although IPv6 addresses in a configuration file should be written in the recommended way in RFC 5952[6], almost all of software accept a redundant notation such as **`2001:0db8:0000:0000:0000:0000:0000:0001`**. And if it is an LLA, you must add **`%zoneid`** part.

However, some do not use raw IPv6 addresses because they break the backward compatibility with IPv4. Let's see **`syslogd(8)`** as an example:

```
syslogd_enable="YES"
syslogd_flags="-s -cc -b [fe80::f4:a6ff:fe43:50b%epair5b]"
```

This is one of the configurations used on the author's FreeBSD box. The **`syslogd(8)`** daemon has a **`-b`** option to choose the listening address and port number. It listens to the address and accepts incoming UDP packets as a logging information source. The address format in IPv4 is "-b *address*:*service*". The *service* is a port number or a service name listed in **`/etc/services`**. Thus the **`syslogd(8)`** daemon cannot recognize which colon is for "*service*" part if a raw IPv6 address is used.

To solve this issue, the **`syslogd(8)`** daemon uses a format like "[*ipv6-address*]:*service*". An IPv6 address must be enclosed with square brackets. If there is the **`%zoneid`** part, it must be inside the brackets. This is a popular format among software using "address:port".

Although this square-bracket format works if the software supports it, you have to be aware of the brackets are one of the meta-characters of shell programs such as **`/bin/sh`**. Actually, the above example for **`syslogd(8)`** does not work because the square brackets are sometimes interpreted as meta-characters. A pair of brackets will be replaced with a file or directory name if the string matches. It works if there is no matched name, but if there is any, the configuration fails strangely. The following is another working example:

```
syslogd_enable="YES"
syslogd_flags=" \
  -b 192.168.0.10 \
  -a 192.168.0.0/24 \
  -a [fe80::ffff:2:200%bridge0]:* \
```

```
  -b [fe80::ffff:1:202%bridge0] \
  -a [fe80::%bridge0]/64 \
"
```

You can see "*" as an argument of the **-a** option. This means the daemon accepts any UDP port. This may match one or more filenames and be replaced when the command line is evaluated. You might consider that this can be solved using "\" just before "*". It may or may not work because it depends on how the shell variable is evaluated. So you must be careful about the pathname expansion when software requires the square-bracket format. Of course, there is no problem when it appears in a configuration file.

Another point where you should pay attention is how to specify a prefix length by using a format like "/64". In the above example, the prefix length is outside the brackets. This is because the prefix length is not a part of an IPv6 address. Locations of a raw address, the zone id part, and square brackets are sometimes confusing. Remember that [**fe80::/64%bridge0**] and [**fe80::%bridge0/64**] are all wrong.

The last pitfall is a case when specifying an address by using a hostname. You can often put a hostname where an address is supposed to be specified. The hostname is usually resolved by name services available on the system. A single name can have multiple addresses because DNS supports and may return multiple A and AAAA resource records. Under such a situation, some software needs clarification about what address and which address family will be used. There is no consistent way to specify an address family of a hostname. You should have a unique hostname with a single address if you want to use hostnames.

Let's see more working examples.

DNS nameserver in /etc/resolv.conf

For simplicity, the configuration of recursive DNS servers should be handled by RA messages explained in the last column. However, if you have a DNS server on the same link, you can add an LLA manually and specify it like this:

```
nameserver fe80::ffff:1:35%bge0
```

One of the reasons why using a manually-configured LLA is that you can use the same address on different networks. It dramatically simplifies the administration.

However, LLAs in **/etc/resolv.conf** do not work for software that depends on LDNS library and does not use resolver functions in FreeBSD libc. This means that the **drill(1)** utility does not work with LLAs while there is no problem with using IPv6 GUA in **/etc/resolve.conf**. And **dhclient(8)** does not work, either. As a workaround, you need "nameserver" line by using IPv4 address or IPv6 GUA[7].

NFS and /etc/exports

The **/etc/exports** file uses raw IPv6 address format. It supports the prefix length notation for "**network**" keyword. An example is as follows:

```
/a/ftproot -alldirs -maproot=0:0 -network 2001:db8:1::/64
/a/ftproot -to -alldirs -maproot=nobody:nobody -network fe80::%lagg0/10
```

Note that NFS does not fully support LLAs because the RPC library always handles IPv6 addresses without the zone ID. The second line can be specified, and it is a correct notation,

but it does not work, unfortunately[8]. IPv6 GUA works fine. Avoid to use LLA for NFS for now. It should be fixed on FreeBSD 14.

Sendmail

The sendmail program also uses the raw IPv6 address format. And it supports the address family selection keyword for each address. An example is as follows:

```
sendmail_enable="YES"
sendmail_flags="-L sm-mta -bd -q30m \
  -ODaemonPortOptions=Family=inet,address=127.0.0.1,Name=MTA \
  -ODaemonPortOptions=Family=inet6,address=::1,Name=MTA6,M=O \
  -ODaemonPortOptions=Family=inet,address=0.0.0.0,Port=587,Name=MSA,M=E \
  -ODaemonPortOptions=Family=inet6,address=::,Port=587,Name=MSA6,M=O \
  -ODaemonPortOptions=Family=inet,address=0.0.0.0,Port=465,Name=MSA,M=s \
  -ODaemonPortOptions=Family=inet6,address=::,Port=465,Name=MSA6,M=s \
"
```

You can safely use a hostname with both a single IPv4 and a single IPv6 address simultaneously because of "**Family=**" keyword.

Configuration of the transport used by MSA[9] is handled by lines in **/etc/mail/FreeBSD.submit.mc:**

```
dnl If you use IPv6 only , change [127.0.0.1] to [IPv6::1]
FEATURE(‘msp’, ‘[127.0.0.1]’)dnl
```

In this configuration file, a modified square-bracket format is used because the sendmail program requires the square-bracket format even for IPv4. If you want to use IPv6, [**IPv6:raw-ipv6-address**] must be used.

Syslogd

The command-line options were explained in the previous section. In the configuration file, **/etc/syslog.conf**, a remote host can be specified by using the raw IPv6 address format. This is an example used to receive logs from a jail environment running ISC BIND:

```
+fe80::e:1ff:fec5:e80b%bridge100
!-named
*.notice;authpriv.none;kern.debug;lpr.info;mail.crit;news.err; /var/log/ns/messages
!named
*.* /var/log/ns/named.log
!*
+@
```

Summary

Getting access to the IPv6 Internet by using tunneling service and configuring essential software included in the FreeBSD base system to use IPv6 are explained.

Tunneling using **gif(4)** interface is not specific to Hurricane Electric IPv6 Tunnel Broker. It can be used to build your own virtual network; while it may be too primitive for a practical

purpose — it has no support for encryption or automatic configuration. FreeBSD provides a rich set of tools for network experiments.

While IPv6 configuration for userland programs typically involves only writing IPv6 addresses into the configuration file instead of IPv4, several pitfalls are listed in this column. There is also software that does not handle an IPv6 LLA correctly. Try IPv6 on your box, and if you find a problem, please report it to the author and/or the FreeBSD project.

In the next issue, more software configuration and NDP (Neighbor Discovery Protocol), one of the IPv6 core protocols you should be familiar with will be explained.

Footnotes

[1] GUA stands for "global unicast address", which is routable in the IPv6 Internet.

[2] Remember that an IPv6 GUA has the prefix and the IID, and the IID is almost always 64-bit long. If you have a 48-bit prefix, you can have 65,536 networks in your LAN by choosing the 16-bit. If you have a 64-bit prefix, you can have a single network only.

[3] This protocol is defined in RFC 4213, "Basic Transition Mechanisms for IPv6 Hosts and Routers".

[4] LLA stands for Link-Local Address. After `ifconfig gif0 up`, an LLA is automatically configured. See also the last column for more details.

[5] IID stands for Interface IDentifier. The lower bits in an IPv6 address that is unique for each host. The length is usually set to 64, but theoretically, any length equal to or shorter than 128 – (subnet prefix length) is allowed.

[6] Rules for text representation of an IPv6 address are covered in the first column. See also RFC 5952: "A Recommendation for IPv6 Address Text Representation".

[7] The author is working on these problems, and it will hopefully fixed in near future.

[8] The author is also working on these problems.

[9] MSA stands for Mail Submission Agent. This is a program to submit an email to an MTA, Mail Transfer Agent. The sendmail program can be used as an MSA or an MTA.

HIROKI SATO is an assistant professor at Tokyo Institute of Technology. His research topics include transistor-level integrated circuit design, analog signal processing, embedded systems, computer network, and software technology in general. He was one of the FreeBSD core team members from 2006 to 2022, has been a FreeBSD Foundation board member since 2008, and has hosted AsiaBSDCon, an international conference on BSD-derived operating systems in Asia, since 2007.