Pragmatic IPv6
(Part 2)

BY HIROKI SATO

Part 1 explained the basics of IPv6 protocol and how to get started using it on a FreeBSD box. You should be able to use automatically-configured link-local scope IPv6 addresses after reading it. These addresses are still powerful and helpful while they are limited to your LAN—you need no global IP address if you just want to communicate to another box on the same network. A link-local address is not routable. It is not likely an attack surface from malicious users on the Internet.

To get access to the IPv6 Internet, you need to configure at least one IPv6 global-scope unicast address. This column focuses on IPv6 deployment scenarios with routers to understand more practical configurations.

Internet in IPv4 and IPv6

After trying an `sshd(8)` example in Part 1, you probably want to try access to the IPv6 Internet. Let's look into what you need to do so and the basics of IPv6 network design.

As you know, the Internet is a global-scale interconnected network driven by the Internet protocol. To go somewhere outside your local network, you need a router that is reachable from/to the Internet. It knows "routes to the outside", and forwards IP packets from your network to other networks.

You should already have a router for the IPv4 Internet. It might be one provided by your ISP[1], or one at a data center that connects your box located there to the Internet. The ISP usually offers an connection endpoint for upstream networks that are reachable from the Internet.

Note that IPv6 is not a compatible protocol with IPv4 while it was designed as a successor. This means that IPv6 and IPv4 Internet are entirely independent, and you need an IPv6 router and an IPv6 endpoint from ISP. "IPv6 is supposed to be upper-compatible with IPv4" is one of the common misunderstandings of IPv6. This comes from the fact that most IPv6 deployments have been implemented by migrating existing IPv4 networks. For instance, you can make your public IPv4 HTTP server on FreeBSD "IPv6-ready" because FreeBSD supports both IPv4 and IPv6. However, you might not want to make it "IPv6-only" simply because people with no IPv6 connectivity cannot access your server. Thus IPv6 deployments are typically done by making existing IPv4 services and networks IPv6-capable in addition to

IPv4. This migration approach is called "dual-stack," and one of the causes that makes you believe IPv4 and IPv6 are always usable on the same machine and are somehow related.

Design of an IPv6 Network

IPv6 is independent of IPv4, so you must design an IPv6 network. Fortunately, IPv6 is almost the same as IPv4 regarding network elements, such as routers and network boundaries. Let's review how an IPv4 network works and then see specifics about IPv6.

IPv4 Local-Area Network Configuration

IPv4 has "network address" or "subnet", which is represented by a host address and a network mask (or a subnet prefix length). For instance, an IPv4 address `192.0.2.1/24` means that you have a network with an address `192.0.2.0` and the 254 host addresses from `192.0.2.1` to `192.0.2.254` are available[2] on the same network for you. The nodes with the same network address share the same L2 segment[3]. In other words, two nodes on the same segment can talk with each other with no router. If a node wants to communicate with another node with a network address outside `192.0.2.0`, it must be sent to a router which knows that destination. Thus, to configure a host, the following information is required:

- a global IPv4 host address,
- a network mask or a subnet prefix length to compute the network address,
- router information to communicate with nodes outside the network.

The router information on IPv4 end hosts is usually configured by specifying a single router on the same network as "the default router". These three elements can also be automatically configured by using DHCP[4]. DHCP is pretty famous for IPv4 end hosts though it is an optional protocol. FreeBSD has **`dhclient(8)`** as a client-side implementation.

On an IPv6 network, these three are automatically configured to some extent. Of course, you can manually configure them, but it is not recommended because the IPv6 address space is quite big. Let's see how the IPv6 network works, in detail.

Single Router Network, Manual Configuration

Figure 1a shows a simple IPv6 network that contains a router and two (or more) IPv6 hosts on the same segment. To configure this network manually, the following **`rc.conf`** variables are used on one of the IPv6 hosts:

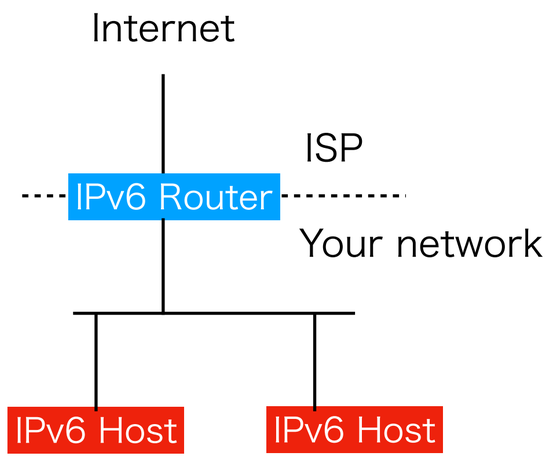

Fig 1a: A simple IPv6 network with a router.

```
ifconfig_bge0="inet 192.168.0.10/24"
ifconfig_bge0_ipv6="inet6 2001:db8:0:1::1/64"
ipv6_defaultrouter="fe80::5a52:8aff:fe10:e323%bge0"
```

This assumes the host has a network interface, `bge0`, the ISP provides the IPv6 router, and your global IPv6 prefix is `2001:db8:0:1::/64`. You can choose the host address. In this

example, **2001:db8:0:1::1/64** is chosen.

The **ifconfig_bge0** line configures an IPv4 address. As explained, IPv6 is independent of IPv4. You do not have to add an IPv4 address when using IPv6 only. For an IPv6-only configuration, this can be rewritten like the following:

```
ifconfig_bge0="up"
ifconfig_bge0_ipv6="inet6 2001:db8:0:1::1/64"
ipv6_defaultrouter="fe80::5a52:8aff:fe10:e323%bge0"
```

Note that you cannot omit the **ifconfig_bge0** line nor write IPv6 configuration in the line. This line is required to teach the **rc.d(8)** framework that the **bge0** interface exists. If this is missing, **bge0** will not be configured. We need no IPv4 address here, but it must not be empty. Thus "up" is used as a harmless sub-command for the **ifconfig(8)** utility. "up" just activates the interface. This is for historical reasons and may be changed in the future releases of FreeBSD[5], but please remember that FreeBSD releases up to 13.x, the latest release at the time of writing, require this rule. IPv6 configurations should be in the **ifconfig_bge0_ipv6** line instead.

The **ifconfig_bge0_ipv6** line is used to configure an IPv6 address and options. The rc.d(8) framework uses this line to recognize whether the interface is IPv6-ready or not. If you omit this line, IPv6 communication on the interface will be blocked. In Part 1, we had **ifconfig_bge0_ipv6="inet6 auto_linklocal"**. Even if IPv6 addresses are automatically configured, you need this line.

The **ipv6_defaultrouter** variable specifies the default router as **defaultrouter** for IPv4 does. You need an IPv6 address of the router. Usually, this information is not provided explicitly. You can find the router's address by using **ping6(8)** utility:

```
% ping6 ff02::2%bge0
PING6(56=40+8+8 bytes) fe80::5a9c:fcff:fe10:ffc2%bge0 --> ff02::2%bge0
16 bytes from fe80::5a52:8aff:fe10:e323%bge0, icmp_seq=0 hlim=255 time=0.996 ms
16 bytes from fe80::5a52:8aff:fe10:e323%bge0, icmp_seq=1 hlim=255 time=1.099 ms
^C
--- ff02::2%bge0 ping6 statistics ---
2 packets transmitted , 2 packets received , 0.0% packet loss
round -trip min/avg/max/std-dev = 0.996/1.048/1.099/0.052 ms
```

ff02::2 is the all-routers multicast address. ICMPv6 echo request packets sent by the **ping6(8)** utility to this address will be received by routers on the network, and you will receive ICMPv6 echo reply packets. You can find the addresses by observing the replies. It is **fe80::5a52:8aff:fe10:e323%bge0**.

After adding this configuration into **/etc/rc.conf**, you can run the **service(8)** utility to reconfigure **bge0**:

```
# service netif restart bge0
```

After the reconfiguration is done, you should notice that **bge0** has two addresses, **2001:db8:0:1::1/64 and fe80::xxx/64**. The latter is an automatically-configured link-lo-

cal address explained in Part 1. Remember that at least one link-local address must be configured on an IPv6-capable interface, even if you are using a global-scope IPv6 address provided by the ISP. The "xxx" part varies depending on the MAC address on the interface.

The source address is chosen from one of these two addresses based on the destination address. Now you can **ping6(8)** to **www.freebsd.org** like this:

```
% ping6 www.freebsd.org
PING6(56=40+8+8 bytes) 2001:db8:0:1::1 --> 2610:1c1:1:606c::50:25
16 bytes from 2610:1c1:1:606c::50:25, icmp_seq=0 hlim=46 time=155.715 ms
16 bytes from 2610:1c1:1:606c::50:25, icmp_seq=1 hlim=46 time=151.051 ms
16 bytes from 2610:1c1:1:606c::50:25, icmp_seq=2 hlim=46 time=152.218 ms
^C
--- wfe2.nyi.freebsd.org ping6 statistics ---
3 packets transmitted , 3 packets received, 0.0% packet loss
round -trip min/avg/max/std-dev = 151.051/152.995/155.715/1.982 ms
```

and you should see a global address, **2001:db8:0:1::1/64**. If the destination is **fe80::5a52:8aff:fe10:e323%bge0**., the link-local address will be used instead.

Note that DNS domain name resolution is performed by the name servers listed in **/etc/resolv.conf**. If you have a working configuration for IPv4, the file has a list of IPv4 addresses. If your IPv6 router works as a DNS proxy, you can put the IPv6 address like this:

```
nameserver fe80::5a52:8aff:fe10:e323%bge0
```

You can put the address similarly if your ISP provides a DNS recursive resolver.

That's all of the fully-manual configurations of an IPv6 host. You should be able to enjoy IPv6 Internet access by using your favorite software that supports IPv6.

Single Router Network, Configured by SLAAC

Figure 1b shows the same network structure, but in this case, the IPv6 hosts are configured automatically by messages from the router. NDP[6] is a part of the IPv6 core protocols which is implemented on top of ICMPv6, and defines this automatic configuration capability. The router can distribute network parameter information in RA (Router Advertisement) messages.

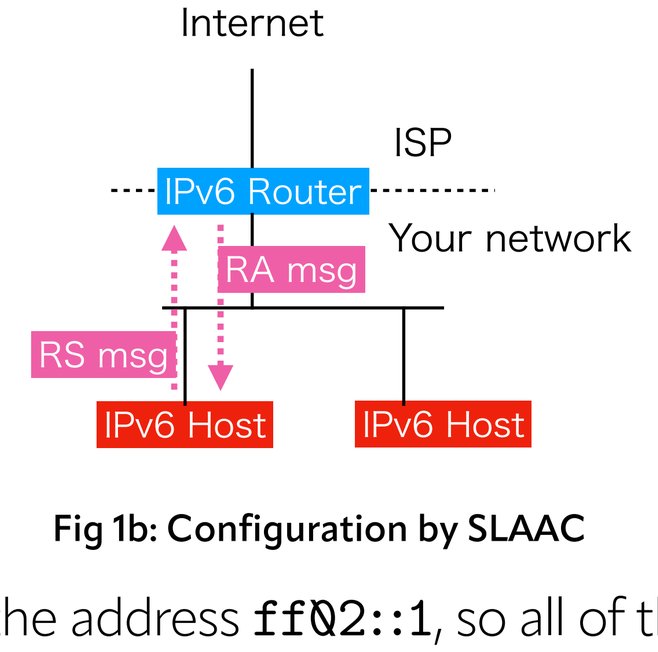

Fig 1b: Configuration by SLAAC

RA messages will be send to the address **ff02::1**, so all of the IPv6 nodes on the same network will receive them. "RS messages" in Figure 1b are Router Solicitation messages. They are used to solicit RA messages and sent to **ff02::2**, allrouters multicast address. A router will send an RA message when receiving an RS message from a host, and also send

unsolicited RA messages periodically so that the IPv6 hosts can know network information changes.

RA messages have a lot of options, which are similar to ones for IPv4 DHCP. Fundamental ones are MTU, prefix, and the default router address so that an IPv6 host can configure itself by using them.

If your router supports RA messages and you want to rely on them, the following configuration works on the IPv6 host:

```
ifconfig_bge0="up"
ifconfig_bge0_ipv6="inet6 accept_rtadv"
```

inet6 accept_rtadv enables **bge0** to accept RA messages and configure the interface. An IPv6 router usually provides prefix information of your network. When a host receives the prefix information, IID[7] will be automatically generated from the MAC address, and an IPv6 address is configured. This mechanism is called SLAAC[8]. You can see this auto-configured address in the output of **ifconfig(8)**. **"autoconf"** keyword is shown just after the address.

```
% ifconfig bge0
...
inet6 2001:db8:a743:3c00:5a9c:fcff:fe10:ffc2 prefixlen 64 autoconf
...
```

To check if your IPv6 router supports RA messages, you can use the **rtsol(8)** utility with a -D flag. It sends a RS message and shows RA messages from the routers:

```
# rtsol -D bge0
rtsol: link -layer address option has null length on bge0. Treat as not included.
rtsol: checking if bge0 is ready...
...
rtsol: received RA from fe80::5a52:8aff:fe10:e323 on bge0 , state is 2
rtsol: Processing RA
rtsol: ndo = 0x7ffffffe3b0
rtsol: ndo->nd_opt_type = 1
rtsol: ndo->nd_opt_len = 1
rtsol: ndo = 0x7ffffffe3b8
rtsol: ndo->nd_opt_type = 3
rtsol: ndo->nd_opt_len = 4
rtsol: rsid = [bge0:slaac]
rtsol: stop timer for bge0
rtsol: there is no timer
```

An RS message will also be sent just after the interface becomes "up". Note that the kernel, not this utility, will process the RA messages. So if the interface is not configured "**inet6 accept_rtadv**", messages are shown but nothing is actually configured. If you did not get "**received RA from...**" line, your IPv6 router did not respond to the RS message. In this case, you cannot use the automatic configuration.

After receiving RA messages, the SLAAC address is configured automatically. And the default router will be configured by using the message's address. Thus just putting "**`inet6 accept_rtadv`**" into **`/etc/rc.conf`** configures a global IPv6 address and the default router.

This is something like DHCP in IPv4. However, there is no server nor no "state" for RA/RS messages. The end host will configure the network parameters upon receiving the messages. While this is a more scalable method than IPv4 DHCP, you cannot control what address is actually configured on each host because they are generated from the MAC addresses. For more fine-grained control of the automatic address configuration, you will need another method, such as DHCPv6.

DNS recursive resolvers can also be configured via RA messages[9]. The kernel cannot handle this information, so the **`rtsold(8)`** daemon will handle it. The **`rtsold(8)`** daemon can be enabled by the following **`rc.conf(5)`** variables:

```
ifconfig_bge0="up"
ifconfig_bge0_ipv6="inet6 accept_rtadv"
rtsold_enable="YES"
rtsold_flags="bge0"
```

`/etc/resolv.conf` will be updated when receiving RA messages if your router provides the information.

RA messages should be enabled on a properly-configured IPv6 network with one or more routers. The absence of RA messages makes network configuration difficult because they have both network parameters and how to configure them.

IPv6 Router Configuration

The previous section assumes that the IPv6 router is provided by your ISP. You can build an IPv6 FreeBSD router by yourself, as you can do it for IPv4. Let's see what must be configured for that.

Figure 2a shows an example that your network has an IPv6 router to have two independent networks. LAN 1 and LAN 2 are connected to each other by the router, and another router provides IPv6 Internet reachability. This section explains how to configure the former one.

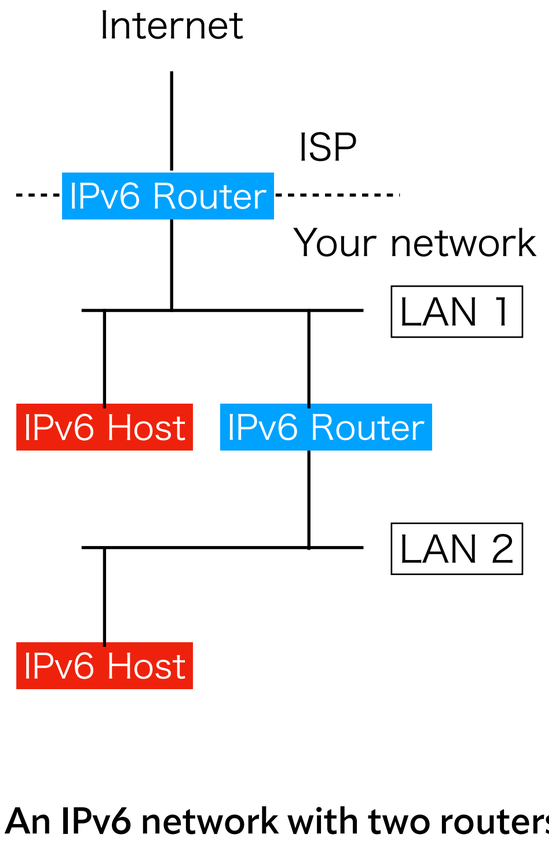

Fig 2a: An IPv6 network with two routers

To enable packet forwarding, you need **`ipv6_gateway_enable`** variable in **`/etc/rc.conf`**. This is the IPv6 counterpart of **`gateway_enable`**, which is for IPv4. Assuming **`bge0`**

and **bge1** are the network interfaces of the router—for LAN 1 and LAN2, respectively—/**etc/rc.conf** will be something like this:

```
ipv6_gateway_enable="YES"
ipv6_defaultrouter="fe80::5a52:8aff:fe10:e323%bge0"
ifconfig_bge0="up"
ifconfig_bge0_ipv6="inet6 2001:db8:0:1::1/64"
ifconfig_bge1="up"
ifconfig_bge1_ipv6="inet6 2001:db8:0:2::1/64"
```

Note that you must have a shorter prefix than 64 to configure this. For example, if the ISP offers `2001:db8::/56` for your network, you can use `2001:db8:0:1::/64` and `2001:db8:0:2::/64` by splitting the `/56` network. You might be tempted to split a `/64` prefix into longer prefixes. However, the author will not recommend using a prefix longer than 64 to design your network. This topic will be revisited in the later columns.

The router by ISP does not know the route to `2001:db8:0:2::1/64`, you have to add a static route configuration on it by using a link-local address on **bge0**. The link-local address is automatically configured, as explained in Part 1.

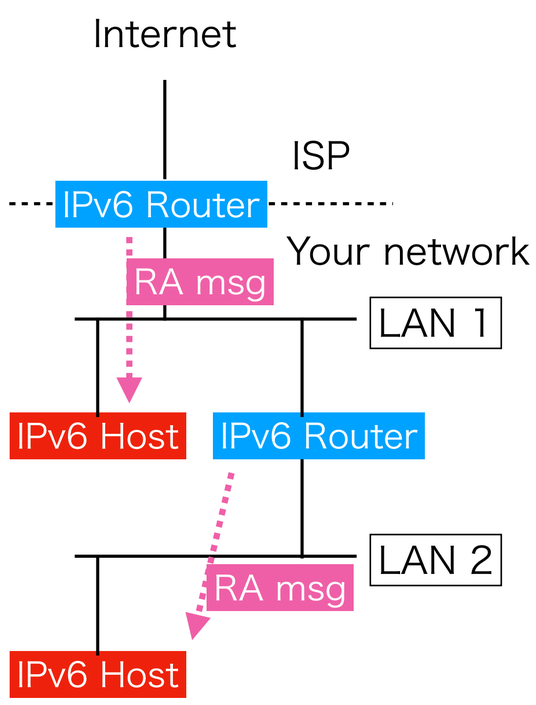

Fig 2b: Configuration by SLAAC on LAN 1 and 2

Figure 2b shows that the RA messages from these two routers. The FreeBSD router has not enabled sending RA messages yet. To send RA messages, you need the **rtadvd(8)** daemon. **rtadvd_enable** and **rtadvd_interfaces** variables enable it:

```
ipv6_gateway_enable="YES"
ipv6_defaultrouter="fe80::5a52:8aff:fe10:e323%bge0"
ifconfig_bge0="up"
ifconfig_bge0_ipv6="inet6 2001:db8:0:1::1/64"
ifconfig_bge1="up"
ifconfig_bge1_ipv6="inet6 2001:db8:0:2::1/64"
rtadvd_enable="YES"
rtadvd_interfaces="bge1"
```

IPv6 hosts on LAN 2 will receive RA messages from the router and perform the automatic configuration. The **rtadvd(8)** daemon distributes prefixes and link MTU which are con-

figured on the interface by default. More information such as DNS servers can be distributed by creating **`/etc/rtadvd.conf`**. See **`rtadvd.conf(8)`** manual page for the details.

You need to be aware that a router receives no RA message. In IPv6 specification, IPv6 nodes are categorized into hosts and routers. A host is a leaf node of the network and does not forward IPv6 packets, and a router is a multi-homed node that forwards IPv6 packets across the networks. RA messages are defined as ones sent by a router and received by a host.

This means that we cannot configure a router by using the automatic configuration capability explained in the previous section. You must not specify "**`inet6_accept_rtadv`**" on a router, and you need to configure the network parameter manually as an example shown above. If you specify **`ipv6_gateway_enable="YES"`**, the FreeBSD kernel will ignore RA messages even if "**`inet6_accept_rtadv`**" is specified.

However, this model is too restrictive under some circumstances. For example, this host-and-router model does not work well for the IPv6 router provided by the ISP. This router must be automatically configured, but there is no way to configure the default router if it does not receive RA messages. On the other hand, if a router receives RA messages to configure itself, the configuration will be screwed up quickly because of messages from other routers. Another router will change the router's default route.

To mitigate this problem, FreeBSD has adopted the following concepts:

- The "host or router" is determined on each interface, not the system-wide property,
- if the interface accepts RA messages, it is seen as "host" from other nodes.

Following this, the **`accept_rtadv`** flag can be configured on a per-interface basis. While the packet forwarding capability cannot be configured similarly, a **`sysctl net.inet6.ip6.rfc6204w3`** is provided. When it is set to 1, the kernel receives RA messages even if the packet forwarding is enabled. While these knobs are difficult to understand, the details and concrete examples will be covered in later columns.

Using DHCPv6

DHCP is also available for IPv6 and it is called DHCPv6[10]. However, it is not widely used like IPv4 because automatic configuration by RA messages and SLAAC are enough for small networks. Figure 3a shows an example of the ISP using DHCPv6. While you can use FreeBSD to implement a network with a DHCPv6 server and clients, topics related to the configuration details will be covered in the later columns. Here, we focus on how a DHCPv6-using network works.

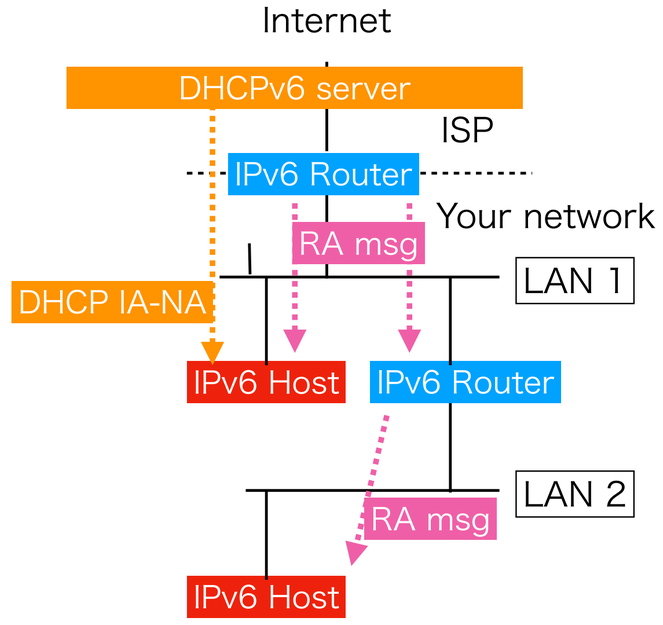

Fig 3a: Configuration by SLAAC and DHCPv6 IA-NA

First, RA messages and DHCPv6 work together, not conflict. DHCPv6 is a way to deliver information unavailable in RA messages. Some are only by RA messages, and some are only by DHCPv6. DHCPv6 can distribute IPv6 addresses. A relationship between the DHCPv6 server, a client, and distributed address information is called an IA (identity association). IAs are defined for address types, and the most notable ones are IA_NA (Non-temporary Address) and IA_PD (Prefix-Delegation).

IA_NA is similar to IPv4 DHCP—an address distributed to an interface attached to the same network as the server. A host establishes an IA_NA in Figure 3a. The host receives an IPv6 address from the DHCPv6 server. At the same time, the host receives RA messages. This means that another address by SLAAC will be configured. So the host will configure two addresses in this case.

Note that the IA_NA delivers just an address, not information about the prefix and the default router. The host still needs to receive RA messages to complete the network configuration. DHCPv6 is not a replacement for RA messages.

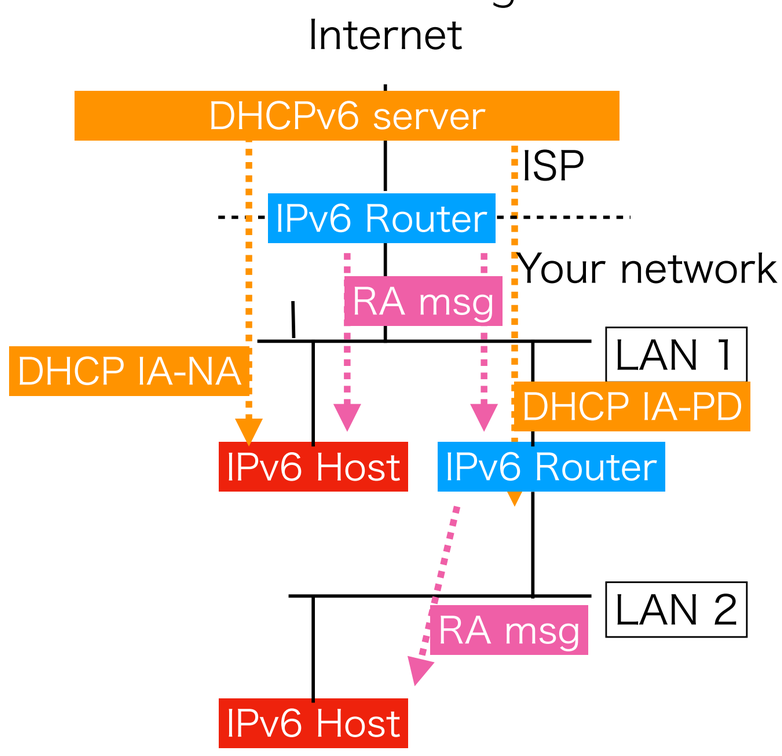

Fig 3b: DHCPv6-PD for router configuration

Figure 3b shows an IA_PD, which is designed to perform an automatic configuration of a router. It is a novel feature available in only DHCPv6. It configures a prefix on an interface on another network, LAN 2. While the IPv6 router between LAN 1 and LAN 2 establishes an IA_PD on LAN 1, the obtained prefix information is used on LAN 2. The interface on LAN 1 will be configured by RA messages. This looks like a complex behavior, but what you need to configure an IA_PD is just specifying the interface for the IA and another interface for the obtained prefix.

In both cases, service discovery of DHCPv6 is performed by the RA messages. In IPv4 DHCP, a client usually sends DHCP DISCOVER broadcast messages to the attached network to find a server. In IPv6, RA messages tell how the client should configure itself. So the configuration for a DHCPv6-using network will be like the following:

```
ifconfig_bge0="up"
ifconfig_bge0_ipv6="inet6 accept_rtadv"
rtsold_enable="YES"
rtsold_flags="-0 /usr/local/etc/dhcp6c.sh bge0"
```

The `rtsold(8)` daemon will handle a flag in RA messages which indicates whether a DHCPv6 server is deployed and the client should use it on the network. The -O option accepts a filename and it will be invoked as an executable when the flag is enabled. This option is disabled by default because FreeBSD has no DHCPv6 client in the base system. It is typically a shell script to invoke a DHCPv6 client software. This configuration works even if there is no DHCPv6 server—you do not need a specific configuration to invoke DHCPv6 client software directly.

In short, DHCPv6 is another option for automatic configuration. It is widely used to configure an IPv6 router automatically by IA_PD, which is often called DHCPv6-PD. RA messages are still used for IPv6 nodes on the same network even if a DHCPv6 server is deployed. One big reason why DHCPv6-only configuration is not enough is that DHCPv6 has no option to configure the default router.

Using PPPoE

Some ISPs are using PPPoE[11] to provide the endpoint on the customer side. As explained in the previous section, a network over Ethernet works fine for IPv4 and IPv6. However, ISP cannot control who connects to their network because no authentication is implemented on the routers. PPPoE is one of the ways to overcome this problem.

Figure 4 is a popular configuration with PPPoE. There are two routers, PPPoE router and IPv6 router, but a single physical box often realizes these two functionalities in practice. In this case, the IPv6 router and the ISP network are connected via Ethernet, but the network interface has no IPv6 address. A virtual point-to-point link over the Ethernet connection is established between the router and an endpoint on the ISP side, and all of the packets go through the link. Authentication can be performed during the negotiation of the link.

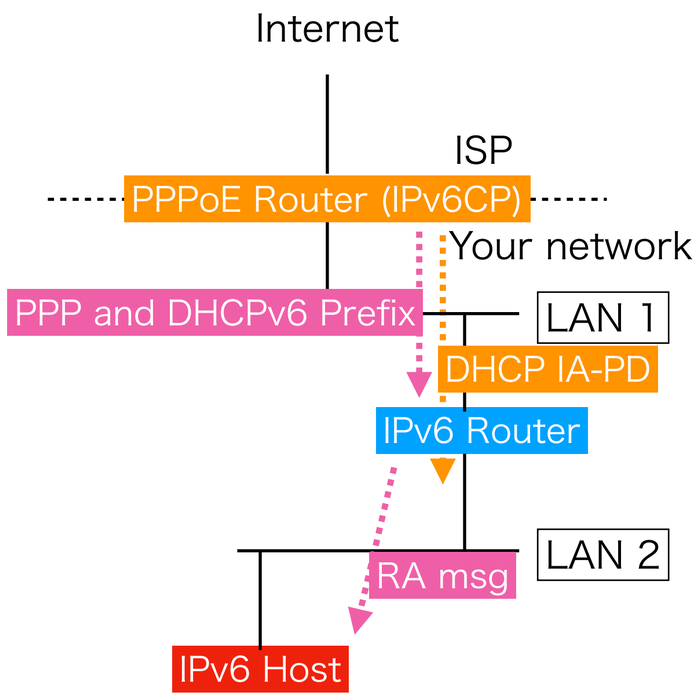

Fig 4: PPPoE Tunnel to provide IPv6 Reachability.

The point-to-point link will be established in the following way:

- The PPPoE router finds the PPPoE endpoint by sending PADI (PPP Active Discovery Initiation) to the ISP network. A PPPoE server responds, and a virtual link is established between the two,
- During the link negotiation, IPv6CP protocol (a part of PPPoE protocol) is used to get IPv6 information. It provides IPv6 IID for the WAN-facing virtual interface. Using this IID and RA messages arrived on the interface, a complete IPv6 address is generated. At this point, the PPPoE router becomes IPv6 Internet reachable,
- Using the virtual link, DHCPv6 IA_PD is established with a DHCPv6 server on the ISP side. The IA_PD configures the LAN-facing interface.

After a PPPoE link and then IA_PD are established, the LAN-facing interface can work as a IPv6 router that forwards packets to the ISP over the PPPoE virtual link. This is one of the most complex configurations for home networks. However, this is just a combination of RA messages, PPPoE, and DHCPv6. You can implement this by using FreeBSD and **net/mpd5**.

Summary

This column showed design and configuration examples of IPv6 networks that are connected to the IPv6 Internet. All of them can be implemented by using FreeBSD while the details of complex ones have not been described yet. The key of IPv6 network configurations is understanding of the automatic configuration capability.

If your ISP offers IPv6 service and you have a global IPv6 prefix, try to configure your FreeBSD box. Automatic configuration by RA messages and SLAAC is easy to configure and FreeBSD supports it out-of-the-box.

In the next column, more details of IPv6 deployments including configuration examples and ones over "tunnel" will be covered. Tunneling is a technique to establish a virtual link, and it can be used to get your FreeBSD box reachable to the IPv6 Internet. Please be patient if your ISP offers no IPv6 service and you cannot try the examples here. You will be able to use IPv6 after understanding how to configure it.

Also, IPv6 relies on multiple addresses, as shown in examples above. It is one of the significant differences from IPv4; every system administrator should know the detail of behaviors of the IPv6 core protocol. The next column will also cover what IPv6 addresses are configured and how they work, including unicast and multicast ones.

Footnotes

[1] Internet Service Provider.

[2] **192.0.2.0** and **192.0.2.255** are reserved and cannot be used as a host address in general.

[3] L2 stands for "layer 2" network. Ethernet is a typical example of L2 protocol, and the IP protocol suite works on top of it.

[4] Dynamic Host Configuration Protocol (RFC 2131).

[5] The author plans to propose a change so that **ifconfig_bge0** can accept IPv6 configurations in FreeBSD 14.x releases.

[6] Neighbor Discovery Protocol for IPv6, RFC 4861.

[7] Interface IDentifier. The lower part of an IPv6 address identifying the node. Usually 64-bit long.

[8] IPv6 Stateless Address Autoconfiguration, RFC 4862

[9] IPv6 Router Advertisement Options for DNS Configuration, RFC 8106.

[10] Dynamic Host Configuration Protocol for IPv6 (DHCPv6), RFC 8415.

[11] A Method for Transmitting PPP Over Ethernet (PPPoE), RFC 2516

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.