Pragmatic IPv6 (Part 1)

BY HIROKI SATO

Have you ever tried to use IPv6 on your FreeBSD box? While IPv4 is still the golden standard of today's Internet world, IPv6 is considered as a promising protocol which is expected to eventually replace IPv4. The first RFC of IPv6 was published in 1995, and FreeBSD has supported it for over 20 years by default. The implementation is mature, not an experimental one anymore.

Why is IPv6 less popular? Some people believe that IPv4 is enough for their purpose and do not want to change. That is a valid reason. However, I believe one of the reasons preventing IPv6 from being popular is system administrators' lack of experience. Similar to the history of IPv4, implementations and best current practices of IPv6 have also been changed over the years. This makes finding books, articles, or literature containing practical information on IPv6 challenging. As long as the pros and cons of using IPv6 are unclear, no skilled sysadmin wants to use this new technology.

This series of articles introduces (or re-introduces) IPv6 to you by showing various examples you can try on your FreeBSD box, not only the theoretical aspects. IPv6 is not a complete replacement of IPv4 and works fine with the existing IPv4 network. Even if you are already using IPv6, you should find something new or valuable.

Introduction

Before diving into the details, I would like to start this with basic concepts that you have to understand. They might be boring if you are already using IPv6, but this article assumes that the readers are familiar with IPv4 network management but not with IPv6 as yet. Definitions of the following technical terms will be explained as the very first introductory topic:

- IPv6 address format and text representation
- Subnet prefix and interface identifier
- Address scope and zone

IPv6 Address

This might be the most visible difference from IPv4. IPv6 has a 128-bit long address[1] while IPv4 has 32-bit long one. The line (1) in Figure 1 shows an example. This IPv6 address is represented in 8 fields separated by colons. You can see 4 digits in hexadecimal in each field because it has a 16-bit value.

It is a complete expression of a 128-bit value. However, there is a recommended way for text

representation of an IPv6 address[2]. The rules are as follows:

- Successive “0”s and/or leading “0”s must be omitted,
- The first longest successive “0”s longer than 16-bit must be replaced with “::”,
- Shorten as much as possible.

```
(1)  2001:0db8:0000:0000:0001:0000:0000:4444
(2)  2001:0db8:0000:0000:0001:0000:0000:4444
(3)  2001:db8:0:0:1:0:0:4444
(4)  2001:db8::1:0:0:4444
```

Figure 1: An IPv6 address in a hexadecimal format (16-bit field x 8)

(2) in figure 1 is one after the successive zeros are removed, and (3) is one after the first longest zeros are removed. (4) is the final form and what you will see in outputs from IPv6-aware software, including FreeBSD.

This convention of the address text representation is just for consistency and simplicity. You can use all representations from (1) to (4) as your input. It would be best to design your software to use (4) in the outputs. An address in a log file is a good example. It should be able to use “grep” in the format of (4).

Subnet Prefix and IID

The next keyword is "subnet prefix" and “interface identifier (IID).” Let’s revisit an IPv4 address before that. Figure 2 shows the structure of an IPv6 and IPv4 address. In IPv4, an address is represented in “dotted octect” format. There are two addresses: host address and network address. The host address identifies a "node" or an end host. The network address identifies a network segment, a domain where nodes can communicate with each other without a router. A network address is calculated by using a host address and a netmask as shown in Figure 2. If you have `192.168.2.1` as the host address and `255.255.255.0` as the netmask, then the network address is `192.168.2.0`. Machines with the same network address belong to the same network segment and can communicate directly.

```
(1)  2001:0db8:0000:0000:0001:0000:0000:4444
     subnet prefix (n bit)   interface identifier (128 – n) bit
(2)  2001:db8::1:0:0:4444/64
                             prefix length
(3)  192.168.2.1             255.255.255.0
     host address            netmask
(4)  network address = AND(host address, netmask) = 192.168.2.0
(5)  192.168.2.1/24
     subnet mask length
```

Figure 2: Subnet prefix and interface indentifier

An IPv4 netmask is designed as a 32-bit value, and it is not necessary for the “1” values in the mask to be contiguous. `255.255.255.0` is (1111 1111 1111 1111 1111 1111 0000 0000) in binary. So the leading “1” is contiguous. Most of the netmasks you see today should be contiguous because it is the standard way of routing domain engineering for the Internet. In the very early days “network class” was used and three classes A, B, and C, were defined by netmasks —"255.0.0.0," "255.255.0.0," and "255.255.255.0"— respectively. The host address determined one of these classes automatically, so you needed no netmask. This was changed at

some point, and CIDR (Classless Inter-Domain Routing) was introduced[3]. In CIDR strategy, the netmask is independent of the host address, and the leading "1"s in the mask are usually contiguous. So the netmask can be represented by using the leading "1"s length. This length is called "subnet mask length" and is often represented at the tail of an IPv4 address with a slash when you want to show the network address information at the same time (see also the last line of Figure 2).

Some modern systems no longer support the non-CIDR netmask. FreeBSD still supports it. You can try the following to see what happens. Have you ever seen an output of netstat(1) like this?

```
# ifconfig bge0 inet 192.168.2.1 netmask 255.128.255.0
# ifconfig bge0 | grep netmask
    inet 192.168.2.1 netmask 0xff80ff00 broadcast 192.255.0.255
# netstat -nrf inet | grep bge0
    192.128.0.0&0xff80ff00 link#1 U bge0
```

Let's go back to IPv6. IPv6 has almost the same concept as "host address" and "network address." A 128-bit address is divided into two parts; one is "subnet prefix," and another is "IID (interface identifier)." The subnet prefix corresponds to "network address" in IPv4. It is a 128-bit address generated by a subnet prefix and "0"s in the lower digits. You can consider a 128-bit netmask, but it is always represented by the leading "1"s length similar to IPv4 CIDR subnet mask length. In IPv6, it is called "prefix length."

You can assign an IPv6 address to an interface on your box. You need a subnet prefix, an I ID, and prefix length to do that. A prefix length for an IPv6 node (e.g., your machine) will be 64 if there is no specific reason. In the core specification of IPv6, the prefix length is variable. However, most IPv6-related protocols assume an IID is 64-bit or longer. If you do not specify the prefix length, 64 will be used as the default value. An IPv6 address assigned to your FreeBSD box is usually split into a 64-bit subnet prefix and a 64-bit IID. An interface identifier identifies the node in the same network segment.

When you get Internet access from your ISP, typically, you will receive an IPv4 address. If your ISP supports IPv6, it provides an IPv6 "prefix." They typically provide prefixes with `/48`, `/56`, `/60`, or `/64` for you. This means you can use 16 bits for your LANs in the case of a `/48` prefix since the prefix length of addresses for your machines is always `/64`.

Let's Try IPv6 on FreeBSD

Are you getting tired of abstract things? Let's go to IPv6 world on your FreeBSD box. As mentioned in the preface, FreeBSD has supported IPv6 by default for a long time. The GENERIC kernel has no runtime knob to enable or disable IPv6. All you need to use IPv6 is simply to add an IPv6 address to one of the interfaces.

Manual Configuration by using ifconfig(8)

To understand IPv6 step-by-step, let's get started configuring it by hand. `bge0` is assumed as a primary NIC on your box in the following examples. You can try them on your working NIC and IPv4 network. It will not break them at all.

Try command lines shown in Figure 3. The first `ifconfig bge0` shows the current status of `bge0`. It should show `nd6 options` line with `IFDISABLED` option. This option means "IPv6 is disabled on this interface".

The second `ifconfig bge0 inet6 -ifdisabled` command removes this option. You can double-check the status by another `ifconfig bge0` command.

By third `ifconfig` command, you should see `inet6` line. This is an IPv6 address configured to `bge0`. The `fe80::d63d:7eff:fe78:fc64%bge0` part is the address, and `prefixlen 64` part shows the prefix length. The "`%bge0`" part looks a bit odd, but it is a part of the address and will be explained later.

You can "ping" this address. The `ping(8)` command for IPv6 is named as `ping6(8)`. Note that `ping(8)` and `ping6(8)` are merged into a single command on and after FreeBSD 13.x, while `ping6(8)` is still available in the newer releases.

If an `sshd(8)` daemon is running on your machine, you can even connect using the IPv6 address. Try the last command in Figure 3. Note that if `AddressFamily` is limited to `inet` in your `sshd_config(8)` configuration file, it fails. See the output of `sockstat -6 | grep sshd` to check whether `sshd(8)` listens to the IPv6 socket.

From users' point of view, TCP/IP applications with IPv6 support like `ssh(1)` work in the same way as IPv4 except for the address format.

```
% ifconfig bge0
bge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=c019b <RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,TSO4,
    VLAN_HWTSO,LINKSTATE>
    ether d4:3d:7e:78:fc:64
    inet 192.168.100.104 netmask 0xffffff00 broadcast 192.168.100.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=29<PERFORMNUD,IFDISABLED,AUTO LINKLOCAL>

# ifconfig bge0 inet6 -ifdisabled

% ifconfig bge0
bge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=c019b <RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,TSO4,
    VLAN_HWTSO,LINKSTATE>
    ether d4:3d:7e:78:fc:64
    inet 192.168.100.104 netmask 0xffffff00 broadcast 192.168.100.255
    inet6 fe80::d63d:7eff:fe78:fc64%bge0 prefixlen 64 scopeid 0x1
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
    nd6 options=21<PERFORMNUD,AUTO LINKLOCAL>

% ping fe80::d63d:7eff:fe78:fc64%bge0
PING6(56=40+8+8 bytes) fe80::d63d:7eff:fe78:fc64%bge0 --> fe80::d63d:7eff:fe78:f-
c64%bge0
16 bytes from fe80::d63d:7eff:fe78:fc64%bge0, icmp_seq=0 hlim=64 time=0.181 ms
16 bytes from fe80::d63d:7eff:fe78:fc64%bge0, icmp_seq=1 hlim=64 time=0.107 ms
16 bytes from fe80::d63d:7eff:fe78:fc64%bge0, icmp_seq=2 hlim=64 time=0.094 ms
^C
```

```
--- fe80::d63d:7eff:fe78:fc64%bge0 ping6 statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 0.094/0.127/0.181/0.038 ms

% ssh -v fe80::d63d:7eff:fe78:fc64%bge0
OpenSSH_7.9p1, OpenSSL 1.1.1k-freebsd 24 Aug 2021
debug1: Reading configuration data /home/hrs/.ssh/config
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Connecting to fe80::d63d:7eff:fe78:fc64%bge0 [fe80::d63d:7eff:fe78:
fc64%bge0] port 22.
debug1: Connection established.
....
```

Figure 3: A command line example to add an IPv6 address to bge0 manually

IPv6 Address Autoconfiguration

An IPv6 address was configured, and it was usable. However, where does this address come from? In IPv4, you should have configured an address by specifying it literally in an `ifconfig`(8) command line. What we did was just `ifconfig bge0 inet6 -ifdisabled`. What happened?

FreeBSD supports IPv6 SLAAC (StateLess Address AutoConfiguration)[4]. This is a neat feature of IPv6 that configure an IID automatically. Remember that an IPv6 address has a prefix and an IID. The prefix is provided by the network operator (you or someone else), but the IID is what you have to choose. It must be unique on the same network.

SLAAC fills the IID part by using the MAC address[5]. It is not exactly the same as the MAC address, but a unique IID is generated based on it. You can see whether SLAAC is enabled or not by checking "nd6 options" line in the output of `ifconfig`(8). If it has "`AUTO_LINKLOCAL`," an IPv6 address is automatically generated and configured.

What about the prefix part? If you configured no IPv6 network or your ISP offered no IPv6 prefix, your LAN has no IPv6 prefix. However, the `bge0` interface has an address `fe80::d63d:7eff:fe78:fc64%bge0`. Figure 4 (1) was the output of `ifconfig`(8). It means an address shown in (2) in a full 128-bit format. The prefix is `fe80::` in the shortened representation.

The `fe80::` subnet prefix is one of the reserved prefixes for "link-local" communication. You can use this prefix freely on any network interface. Addresses within this prefix are limited to communication between directly-connected machines to the link. In IPv4, 169.254.0.0/16 is reserved for the same purpose, but it is optional and not widely used for actual communications[5]. In IPv6, the link-local subnet prefix is essential in the core protocol. This is why this address is automatically configured just by enabling IPv6 on the interface.

So you can consider that an IPv6-capable interface always has at least one address with the link-local subnet prefix. You can use this address for standard TCP/IP applications while it is also used in various vital communications.

```
(1)   inet6 fe80::d63d:7eff:fe78:fc64%bge0 prefixlen 64
(2)   fe80:0000:0000:0000:d63d:7eff:fe78:fc64%bge0/64
      prefix                    IID
```

Figure 4: An automatically-configured IPv6 address

Address Type, Scope, and Zone

The last mysterious part of the auto-configured address is `%bge0`. If you try `ping6(8)` without this part, `ping6(8)` fails. What is this?

An IPv6 address has its "scope." The scope is a topological span within which the address may be used as a unique identifier[7]. Although several scopes were proposed in IPv6 RFCs in the past, only two scopes named "global" and "link-local" are practical today. The global scope means routable on the Internet. The link-local scope is valid only on a single, specific link.

The keyword "zone" is related to "scope." Figure 5 shows an example network with two or more NICs. A zone is an extent which a scoped address must be unique. If you have two NICs and they are connected to the same network, you will have an `fe80::` prefix address on each interface automatically. In this case, these two must be unique, and this extent is called "a link-local zone." Each link-local zone is identified by using a zone identifier. On FreeBSD, the zone identifier is the scope identifier or the interface name of the NIC. You can see "scopeid" keyword in the output of `ifconfig(8)`. The value just after the keyword is the scope identifier.

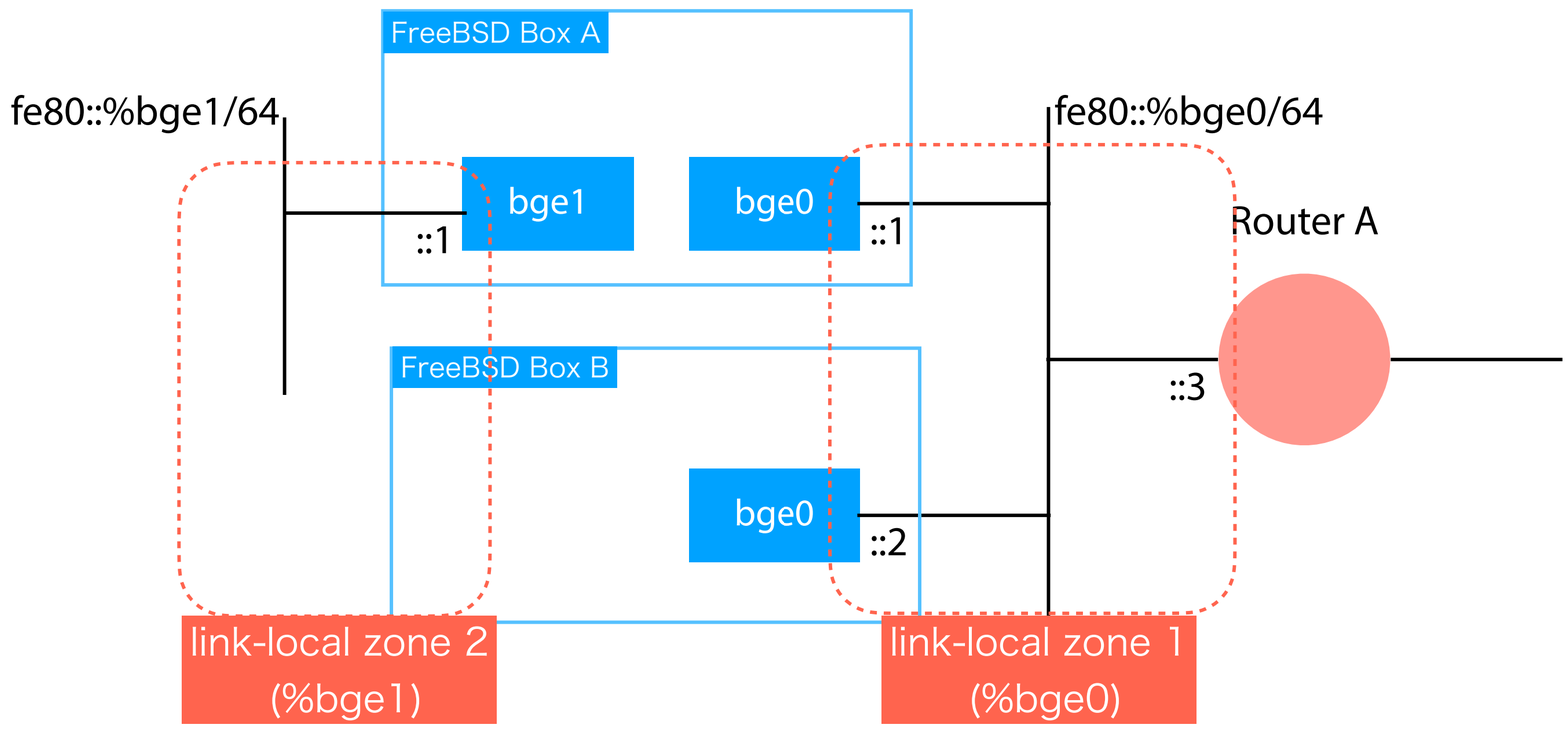

Figure 5: Relationship between zones and network segments

Let's go back to `%bge0`. This is the zone identifier of this address. Link-local scoped addresses are unique within the zone but may not be unique on the same machine. So you have to specify which zone must be used. `%bge0` part is required to specify the zone. The scope identifier of `bge0` is 1, so you can write it as %1 instead. A link-local address with no zone identifier is considered invalid[7].

And an IPv6 address has two types; "unicast" and "multicast." This is independent of the address scope. If it is unicast, the address is used for regular 1:1 communication. If multicast, the communication can have multiple receivers.

The prefix automatically determines the scope and type:

- if the first 8 bits are "1111 1111" (ff00::), it is a multicast prefix,
- if the first 10 bits are "1111 1110 10" (fe80::), it is a link-local scope and unicast prefix (called as LLA, linklocal address),
- the others are global scope, unicast prefixes (called GUA, global unicast address).

Useful Multicast Addresses

Multicast addresses are not popular in standard IPv4 deployment. In IPv6, multicast is used everywhere. More details will be covered in later issues, but I would like to introduce the following two addresses:

- `ff02::1`—link-local scope, all-nodes multicast address
- `ff02::2`—link-local scope, all-routers multicast address

Every IPv6 node "joins" the all-nodes multicast address. This means that all machines connected to the specified zone will respond to communication to this address. Let's try the command shown in Figure 6. If no other IPv6-capable device is connected to `bge0`, you will see no response except for your machine itself. If you have some, you will see multiple ICMPv6 echo replies. If you use `ff02::2`, routers on the same link will respond. If you have Apple macOS or iOS devices, you can see them because they enable IPv6 by default.

These two link-local multicast addresses are helpful for diagnostics. Remember that you can find an IPv6 neighbor by using `ff02::1`, and you can find an IPv6 router by using `ff02::2`.

```
% ping6 ff02::1%bge0
PING6(56=40+8+8 bytes) fe80::d63d:7eff:fe78:fc64%bge0 --> ff02::1%bge0
16 bytes from fe80::d63d:7eff:fe78:fc64%bge0 , icmp_seq=0 hlim=64 time=0.073 ms
16 bytes from fe80::21b:78ff:fe39:84f6%bge0 , icmp_seq=0 hlim=64 time=0.194 ms(DUP!)
16 bytes from fe80::225:90ff:fe13:503a%bge0 , icmp_seq=0 hlim=64 time=0.276 ms(DUP!)
16 bytes from fe80::a6ba:dbff:fee0:b190%bge0 , icmp_seq=0 hlim=64 time=0.351 ms(DUP!)
16 bytes from fe80::202:a5ff:fee9:4104%bge0 , icmp_seq=0 hlim=64 time=0.427 ms(DUP!)
16 bytes from fe80::202:a5ff:fee9:c87d%bge0 , icmp_seq=0 hlim=64 time=0.511 ms(DUP!)
16 bytes from fe80::8c:7aff:fe24:1c64%bge0 , icmp_seq=0 hlim=64 time=0.585 ms(DUP!)
16 bytes from fe80::202:a5ff:fee9:c3c9%bge0 , icmp_seq=0 hlim=64 time=0.658 ms(DUP!)
16 bytes from fe80::202:a5ff:fee9:3952%bge0 , icmp_seq=0 hlim=64 time=0.730 ms(DUP!)
16 bytes from fe80::202:a5ff:fee9:2e5e%bge0 , icmp_seq=0 hlim=64 time=0.802 ms(DUP!)
:
:
```

Figure 6: A ping6 command with ff02::1

IPv6 Node Configuration

An IPv6 node must have the following configuration:

- At least one link-local address on each NIC,
- a lookback address.

The loopback address is `::1`. In IPv4, `127.0.0.1` is used and configured on `lo0`. The `::1` is also automatically configured on `lo0`.

You can configure more IPv6 addresses on a single interface. In IPv6, many addresses are used on the same interface for various purpose. For example, you can add another link-local address `fe80::1/64` by using `ifconfig(8)`:

```
% ifconfig bge0 inet6 fe80::1/64
```

Unlike IPv4, this does not remove or replace the already configured IPv6 addresses. Check the output of `ifconfig(8)` and try `ping6(8)` or `ssh(1)` to use the new address. If you want to remove a specific address, you can use `-alias` flag like this:

```
% ifconfig bge0 inet6 fe80::1/64 -alias
```

What address/prefix can I use?

You might be confused about what IPv6 address you can actually use. In the IPv4 network, you probably know "private address space" is always available for your local network[9]. `10/8`, `172.16/16`, and `192.168/24` are widely used. So what about IPv6?

As mentioned in the previous sections, you can use `fe80::/64` prefix for link-local communication. You can generate addresses by using this prefix freely as long as there is no duplication. Most TCP/IP applications work with them[10]. Note that you need to add `%zoneid` part in the address if you use a link-local unicast address.

If you have a global IPv6 prefix from your ISP, you can use it in addition to the link-local prefix. If you configure the default router for IPv6, you can do IPv6 Internet communication. The global prefix is not a replacement for the link-local one; the link-local addresses are essential for the core IPv6 protocol. Do not forget that you need them even if you have global ones.

So is there no private address replacement in IPv6? Yes and no. ULA (Unique Local Address) is a reserved address space for a similar purpose[11]. However, deployment of ULA needs some more careful consideration than IPv4.

More details about IPv6 deployment scenarios, including clarification about the questions above, will be covered in the next issue.

Configuration in rc.conf

Lastly, let's see how to configure IPv6 addresses in `/etc/rc.conf`. For IPv4, `ifconfig_bge0` line is used to configure the `bge0` interface. For IPv6, `ifconfig_bge0_ipv6` is used to indicate that `bge0` is IPv6-capable. The simpliest version of the configuration is as follows. In this configuration, only one link-local address is automatically configured:

```
ifconfig_bge0="inet 192.168.0.10/24"
ifconfig_bge0_ipv6="inet6 auto_linklocal"
```

If you want to add another link-local address manually, you can add "inet6" line into `ifconfig_bge0_ipv6`. This configuration adds two link-local addresses, one by SLAAC and another by the `ifconfig_bge0_ipv6` line:

```
ifconfig_bge0="inet 192.168.0.10/24"
ifconfig_bge0_ipv6="inet6 fe80::1/64"
```

More addresses can be added in the same way as IPv4. The following example adds a global unicast address `2001:db8::1/64` by using `ifconfig_bge0_alias0` line:

```
ifconfig_bge0="inet 192.168.0.10/24"
ifconfig_bge0_ipv6="inet6 fe80::1/64"
ifconfig_bge0_alias0="inet6 2001:db8::1/64"
```

Summary

This column introduced IPv6 basics and configuration examples on FreeBSD. Once you understand the address structure and configuration by using ifconfig(8), you can use automatically-configured link-local addresses. While they are not globally routable on Internet, they are still helpful for local network communication or diagnostics.

In the next issue, deployment details using IPv6 global prefix and more configuration examples of FreeBSD and thirdparty software on IPv6-enabled network.

Footnotes

[1] RFC4291: "IP Version 6 Addressing Architecture"

[2] It is summarized in RFC 5952: "A Recommendation for IPv6 Address Text Representation"

[3] RFC 1519: "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy"

[4]RFC 4862: "IPv6 Stateless Address Autoconfiguration"

[5]The IID is not always generated by using the MAC address, but it is one of the most popular implementations. RFC 7217 and RFC 8064 have discussed several issues and a new standard. This topic will be covered in later issues of this column.

[6]RFC 3927: "Dynamic Configuration of IPv4 Link-Local Addresses". Microsoft calls this as "Automatic Private IP Addressing (APIPA)".

[7]RFC 4007: "IPv6 Scoped Address Architecture"

[8]The zone identifier “bge0” is actually interface-local, not link-local. If you have bge0 and bge1 which are connected to the same network segment, both `%bge0` and `%bge1` means the same link-local zone. IPv6 network stack on FreeBSD internally assumes interface-local and link-local are the same.
[9]RFC 1918: “Address Allocation for Private Internet”
[10]Unfortunaltely, a few applications are incompatible with LLA. They will be explained in later issues.
[11]RFC 4193: “Unique Local IPv6 Unicast Addresses”

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.