Pragmatic IPv6 (Part 4)

BY HIROKI SATO

As mentioned in earlier columns, IPv6 looks similar to IPv4 except for the address format. However, these two protocols work independently, and of course, there are features specific to each protocol. This column will take a look at IPv6's unique characteristics based on multiple addresses and its practical use, and NDP, Neighbor Discovery Protocol, which is responsible for the L2-to-L3 address resolution and discovery of hosts and routers on the same network.

Many IPv6 Addresses on Your Box

When you use IPv6 on your FreeBSD box, you will usually get multiple addresses like this:

```
% ifconfig vlan100
vlan100: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=80003<RXCSUM,TXCSUM,LINKSTATE>
        ether a4:ba:db:e0:ae:33
        inet6 2001:db8:fb5d::1prefixlen64
        inet6 fe80::a6ba:dbff:fee0:ae33%vlan100 prefixlen 64 scopeid 0x6
        inet6 fe80::ffff:2:7b%vlan100 prefixlen 64 scopeid 0x6
        inet6 fe80::ffff:2:35%vlan100 prefixlen 64 scopeid 0x6
        inet 192.168.100.1 netmask 0xffffff00 broadcast 192.168.100.255
        groups: vlan
        vlan: 100 vlanproto: 802.1q vlanpcp: 0 parent interface: lagg0
        media: Ethernet autoselect
        status: active
        nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>
```

This is a live example from one of the author's boxes. The `/etc/rc.conf` contains the following:

```
ifconfig_vlan100_ipv6="inet6 2001:db8:fb5d::1/64"
ifconfig_vlan100_alias0="inet6 fe80::ffff:2:7b/64"
ifconfig_vlan100_alias1="inet6 fe80::ffff:2:35/64"
```

You can see four IPv6 addresses in the output of `ifconfig(8)`. The `2001:db8:fb5d::1` is a GUA[1], and the other three are LLAs[2]. In `/etc/rc.conf` file, only two LLAs are explicitly specified. Why do we have four?

Automatically-configured LLA

Remember that the following will happen when an `ifconfig_IF_ipv6` is specified:

- The `IFDISABLED` flag is removed, and
- an LLA based on the L2 address of the interface will be automatically configured.

More precisely, all IPv6-capable interfaces have `AUTO_LINKLOCAL` flag by default in the kernel level, and it will configure an LLA when the interface is becoming "up". The `rc.d(8)` scripts will add `IFDISABLED` flag when no `ifconfig_IF_ipv6` is specified to prevent the interface from configuring an LLA. This is a seatbelt for people who want IPv4 only. As long as you have no `ifconfig_IF_ipv6` line, the interface will get no IPv6 address. The automatically configured LLA is an L3[3] address, so one on the same network can try to access your box over IPv6 TCP or UDP. For this reason, the LLA is not configured unconditionally.

Note that an LLA is mandatory if you want to use an IPv6 GUA. Unlike IPv4, you always have to configure an LLA. This is the reason why there is the `AUTO_LINKLOCAL` flag by default and the kernel configures one. While you can remove the LLA manually, various odd behaviors will occur.

Modified EUI-64 Format Interface Identifiers

Let's see the automatically-configured LLA again. The prefix is always `fe80::/64`. The IID is filled by using the L2 address. If you are using Ethernet, it is the IEEE 802 48-bit MAC, also known as Ethernet MAC address. The Ethernet MAC address is 48-bits long. You can find "`ether`" keyword in the output of the `ifconfig(8)` command:

```
ether a4:ba:db:e0:ae:33
inet6 fe80::a6ba:dbff:fee0:ae33%vlan100 prefixlen 64 scopeid 0x6
```

The IID looks similar to the MAC address but not the same. This is called "modified EUI-64 format interface identifier" and is generated from the 48-bit MAC address. The generation algorithm[4] is simple. Let's compare the IID to the MAC address octet-by-octet[5]:

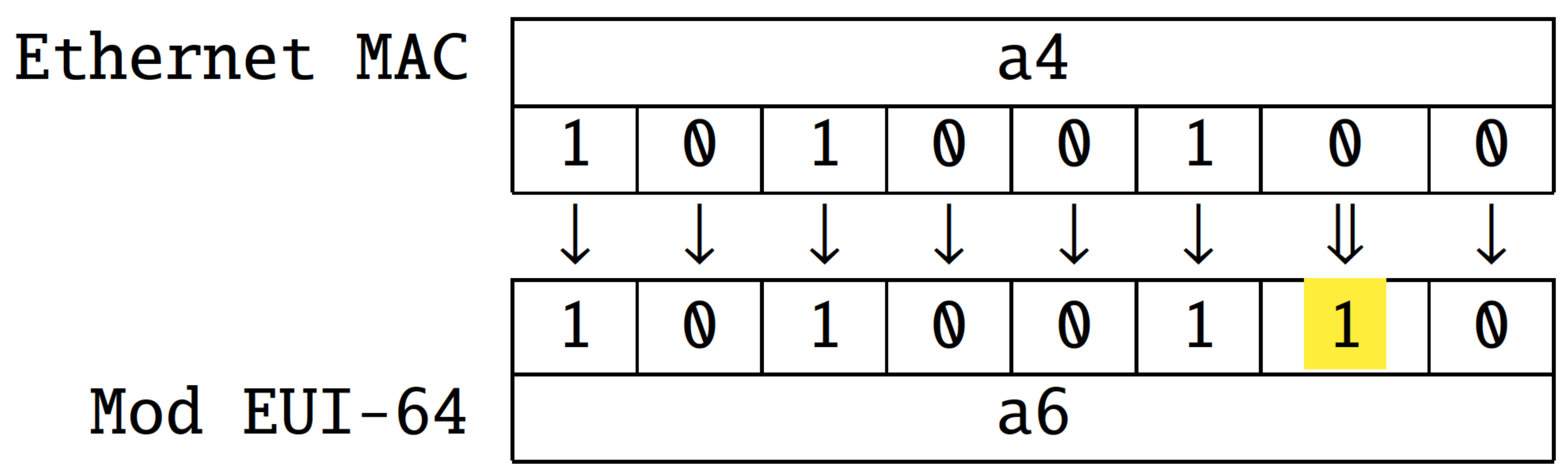
Ethernet MAC	a4	ba	db			e0	ae	33
	⇓	↓	↓			↓	↓	↓
Mod EUI-64	a6	ba	db	ff	fe	e0	ae	33

The IID is 64-bit long, so two octets must be filled. The "`0xff`" and "`0xfe`" in the center of the IID are always added. In other words, if the IID has `0xfffe` at the center, it is generated by an EUI-48 MAC address. There is one more difference—the first octet has slightly been changed. The first and second bits (from the LSB[6]) of the first octet in the MAC address have the following meanings:

first bit: "individual"(0) or "group"(1),

second bit: "universal"(0) or "local"(1).

The "individual" means unicast (i.e., 1-to-1 communication), and the "group" means multicast or broadcast (1-to-n communication). When using a real hardware NIC, not a virtual one, it has a unique MAC address assigned by the vendor. In this case, the address is universally unique, and the second bit of the octet is 0. However, in the modified EUI-64 format, the second bit is specified as an inverted value of the MAC address. So, in most cases, the second bit of the first octet is 1. The first octet, "`0xa4`" in this example, is changed in the following way. The bit array of the hexadecimal value "`0xa4`" is "`10100100`". The second bit from the rightmost bit in the array will be inverted, and you will get "`0xa6`" in the IID:



Currently, on FreeBSD, the automatic LLA and GUAs set by SLAAC use this algorithm. There are two topics you have to be aware of. One is the reason why the second bit is flipped, and another is a problem with the generated IIDs.

Problems of Modified EUI-64 IID

The reason for inverting the second bit is to make it easy to configure an address manually. The MAC address on a real hardware NIC has a "universal" bit, so the first octet of the generated IID will never be "`0x00`." Using this fact, you can configure an IID that does not conflict with automatically-configured ones. For example, "`0:0:0:1`" or "`::1`" is an address you can choose because the first octet is `0x00`. If this inversion were not defined, you would have to use something like "`0200:0:0:1`."

Although the modified EUI-64 IID is popular in IPv6 implementations, privacy is an issue. As you can imagine, the MAC address in the generated address can be used to track your network activity. While the IPv6 address space is enormous for address scanning, the EUI-64 IID address space is smaller than that. RFC 7721, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms," has extensively discussed the security and privacy aspects of the algorithm.

There are two algorithms to mitigate them. RFC 8981, "Temporary Address Extensions for Stateless Address Autoconfiguration in IPv6," defines "temporary address." The temporary address is an automatically-configured IPv6 address by SLAAC with a random IID and is valid for a short period of time. This is intended for the source address when initiating an outgoing session. It is difficult for an outside entity to predict the IID that is employed for temporary addresses. FreeBSD partially supports this extension, and you can enable it by setting the following sysctl variable:

```
# sysctl net.inet6.ip6.use_tempaddr=1
```

After enabling this, two addresses will be configured by SLAAC:

```
# ifconfig vlan84
vlan84 : flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
         options=80003<RXCSUM,TXCSUM,LINKSTATE>
         ether a4:ba:db:e0:ae:33
         inet6 2001:db8:fb5d:8001::42 prefixlen 64
```

```
        inet6 fe80::a6ba:dbff:fee0:ae33%vlan84 prefixlen 64 scopeid 0x7
        inet6 fe80::ffff:2:7b%vlan84 prefixlen 64 scopeid 0x7
        inet6 fe80::ffff:2:35%vlan84 prefixlen 64 scopeid 0x7
        inet6 2001:db8:fb5d:8001:a6ba:dbff:fee0:ae33 prefixlen 64 autoconf
        inet6 2001:db8:fb5d:8001:7c36:33b7:b967:382f prefixlen 64 autoconf
        temporary groups: vlan
        vlan: 84 vlanproto: 802.1q vlanpcp: 0 parent interface: lagg0 media:
        Ethernet autoselect
        status: active
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
```

Note that the **`vlan84`** is a different interface from **`vlan100`** in the previous example. You can see two addresses with the "autoconf" keyword. SLAAC and a modified EUI-64 IID generate the first one, and the second one has a random IID and is labeled with "temporary." The temporary address will be automatically changed once in 24 hours by default. Note that if you already have a SLAAC address and then enables the **`use_tempaddr`** variable, you need to remove the SLAAC address first.

This extension is useful to some extent, but the current FreeBSD implementation has the following problem:

- You cannot control the generation of temporary addresses per-interface basis. When enabled, all of the interfaces accepting Router Advertisement will have a temporary address.
- The address generation algorithm is based on an old specification in RFC 4941, not in RFC 8981.

There are also several pitfalls when you try to use it. This topic will be covered in later columns. At this moment, you should learn that the modified EUI-64 IID is popular, and FreeBSD uses it when autoconfiguration is performed. The auto-configured address is a normal address that can be used for TCP or UDP communication. Thus, you might want to be aware that someone on the same network segment can try to access your box using the address.

Another algorithm is a stable IPv6 interface identifier proposed in RFC 7217, "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)." This is a drop-in replacement of the modified EUI-64 IID and a solution to security and privacy issues due to the MAC address used. FreeBSD has not supported this yet, but the author is working on the implementation. This will also be covered in the later columns.

Non-Unicast Addresses

When an IPv6 address is configured, your FreeBSD box actually has more addresses. Try **`ifmcstat`** command like this:

```
% ifmcstat -i vlan84 -f inet6
vlan84 :
        inet6 fe80::a6ba:dbff:fee0:ae33%vlan84 scopeid 0x7
        mldv2 flags=2 <USEALLOW > rv 2 qi 125 qri 10 uri 3
                group ff02::1:ff67:382f%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr33:33:ff:67:38:2f
                group ff02::202%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr 33:33:00:00:02:02
                group ff02::1:ff02:35%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr 33:33:ff:02:00:35
                group ff02::1:ff02:7b%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr 33:33:ff:02:00:7b
                group ff02::1:ffe0:ae33%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr 33:33:ff:e0:ae:33
                group ff01::1%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr 33:33:00:00:00:01
                group ff02::2:a17e:3d85%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr 33:33:a1:7e:3d:85
                group ff02::2:ffa1:7e3d%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr 33:33:ff:a1:7e:3d
                group ff02::1%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr 33:33:00:00:00:01
                group ff02::1:ff00:42%vlan84 scopeid 0x7 mode exclude
                        mcast-macaddr 33:33:ff:00:00:42
```

Addresses after the keyword "group" are ones assigned to the interface **`vlan84`**. You can even try to send a ping to the addresses and get a response:

```
% ping6 ff02::1:ff67:382f%vlan84
PING6 (56=40+8+8 bytes) fe80::a6ba:dbff:fee0:ae33%vlan84 --> ff02::1:
ff67:382f%vlan84
16 bytes from fe80::a6ba:dbff:fee0:ae33%vlan84, icmp_seq=0 hlim=64
time=0.073 ms
16 bytes from fe80::a6ba:dbff:fee0:ae33%vlan84, icmp_seq=1 hlim=64
time=0.044 ms
16 bytes from fe80::a6ba:dbff:fee0:ae33%vlan84, icmp_seq=2 hlim=64
time=0.054 ms
^C
--- ff02::1:ff67:382f%vlan84 ping6 statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev=0.044/0.057/0.073/0.012ms
```

However, they were not in the output of the **`ifconfig`** command. What are they?

Well-Known Addresses and Their Application

You can see that all of the addresses have the same prefix "**`ff00::/12`**". Do you remember that in the last column, **`ping6(8)`** utility was used with the address "**`ff02::1`**" to check if the IPv6 communication works or not? "**`ff02::1%vlan84`**" is listed in the sixth entry.

An address with the prefix "**`f000::/4`**" is an IPv6 multicast address. It is used for 1-to-n communication. When you send a ping to this address, you may receive one or more responses. The prefix determines whether the address is multicast or not and the scope to which the address belongs. And the purposes of each address are also defined. All of the well-known multicast addresses and their application can be found in "IPv6 Multicast Address Space Registry"[7].

Let's see the address listed in the above example. An address with "`ff01::/16`" is an interface-local multicast address, and one with "`ff02::1/16`" is a link-local multicast address. You always need the `%zoneid` part.

"`ff02::1`" is the all-nodes address whose scope is link-local. This means that all IPv6-capable nodes on the same network have this multicast address. If you send a ping to "`ff02::1%vlan84`", you will get a lot of responses from the network on `vlan84`. A multicast address does not belong to a single node. Thus, we usually say that a host "joins" the address. All IPv6-capable hosts automatically join the all-nodes multicast address. There is no need to configure it. This is why you can always use "`ff02::1`" as a tool to check if there is an IPv6 node on the interface. "`ff02::2`" is the link-local all-routers address. This is not included in the output of `ifmcstat` command because this machine is not configured as an IPv6 router. If you send a ping to "`ff02::2%vlan84`," you can check if there is a router on `vlan84`.

"`ff01::1`" is the interface-local all-nodes address. The "interface-local" means an isolated group where only the interface belongs. You will receive a response from the same interface by sending a ping to this address.

"`ff02::202`" is the multicast address which the `rpcbind(8)` deamon uses.

Of course, these addresses are not only for the `ping6(8)` utility. They are used when ICMPv6 or some other 1-to-n communication is required. If all IPv6 nodes need to receive it, "`ff02::1`" is used. If all IPv6 routers need to do it, "`ff02::2`" is used. Therefore, most local ICMPv6 control messages will be delivered using a combination of an LLA on the host and these well-known multicast addresses.

Let's see more concrete examples. What is remaining? The following addresses are still unclear:

```
ff02::1:ff67:382f%vlan84 scopeid 0x7 mode exclude
ff02::1:ffe0:ae33%vlan84 scopeid 0x7 mode exclude
ff02::1:ff00:42%vlan84 scopeid 0x7 mode exclude
ff02::1:ff02:35%vlan84 scopeid 0x7 mode exclude
ff02::1:ff02:7b%vlan84 scopeid 0x7 mode exclude
ff02::2:a17e:3d85%vlan84 scopeid 0x7 mode exclude
ff02::2:ffa1:7e3d%vlan84 scopeid 0x7 mode exclude
```

Solicited-Node Multicast Address

The following addresses are called "Solicited-Node Multicast Address":

```
ff02::1:ff67:382f%vlan84 scopeid 0x7 mode exclude
ff02::1:ffe0:ae33%vlan84 scopeid 0x7 mode exclude
ff02::1:ff00:42%vlan84 scopeid 0x7 mode exclude
ff02::1:ff02:35%vlan84 scopeid 0x7 mode exclude
ff02::1:ff02:7b%vlan84 scopeid 0x7 mode exclude
```

A Solicited-Node Multicast Address is one with the prefix `ff02:0:0:0:0:1:ff00::/104`. This means it ranges from `ff02::1:ff00:0 to ff02::1:ffff:ffff`. These five addresses start with this prefix.

What is the purpose and how is the IID configured? The objective is NDP, Neighbor Discovery Protocol[9].

Neighbor Discovery Protocol

NDP is one of the core protocols of the IPv6 protocol suite and is responsible for the following functionalities seen in IPv4:

- ARP (L2-L3 address translation)
- ICMP Router Discovery[9]

and the following IPv6-specific features:

- DAD (Duplicate Address Detection)
- SLAAC (StateLess Address AutoConfiguration)

Before diving into the details, let's see the IID of the address format of a Solicited-Node Multicast Address. This address is generated from a unicast address. To understand the correspondence, compare the output of **`ifconfig`** and **`ifmcstat`** command:

```
inet6 2001:db8:fb5d:8001:7c36:33b7:b967:382f prefixlen 64 autoconf temporary
ff02::1:ff67:382f%vlan84 scopeid 0x7 mode exclude

inet6 2001:db8:fb5d:8001:a6ba:dbff:fee0:ae33 prefixlen 64 autoconf
ff02::1:ffe0:ae33%vlan84 scopeid 0x7 mode exclude

inet6 2001:db8:fb5d:8001::42 prefixlen 64
ff02::1:ff00:42%vlan84 scopeid 0x7 mode exclude

inet6 fe80::ffff:2:35%vlan84 prefixlen 64 scopeid 0x7
ff02::1:ff02:35%vlan84 scopeid 0x7 mode exclude

inet6 fe80::ffff:2:7b%vlan84 prefixlen 64 scopeid 0x7
ff02::1:ff02:7b%vlan84 scopeid 0x7 mode exclude
```

In short, the last three octets of a unicast address are used in the multicast address. For example, `ff02::1:ff67:382f%vlan84` has `0x67`, `0x38`, and `0x2f`. These three octets are at the last of the unicast address `2001:db8:fb5d:8001:7c36:33b7:b967:382f`. So, your box will have as many link-local multicast addresses as there are unicast addresses. While no multicast address appears in the output of **`ifconfig`** command, they are always automatically configured.

Address Resolution

Let's move on to how multicast addresses are used in NDP. One of the most crucial functionalities of NDP is L2-L3 address resolution. For IPv4, ARP[10] is responsible for this. The big difference is that ARP is a protocol of L2 network such as Ethernet, not IPv4. To communicate in IPv4, the source and destination L2 address are required. This address mapping information cannot be obtained using IPv4 due to a chicken-and-egg problem. In IPv6, using an LLA and a Solicited-Node Multicast Address, a host can initiate an IPv6 communication without knowing the destination address. The all-node address is always available.

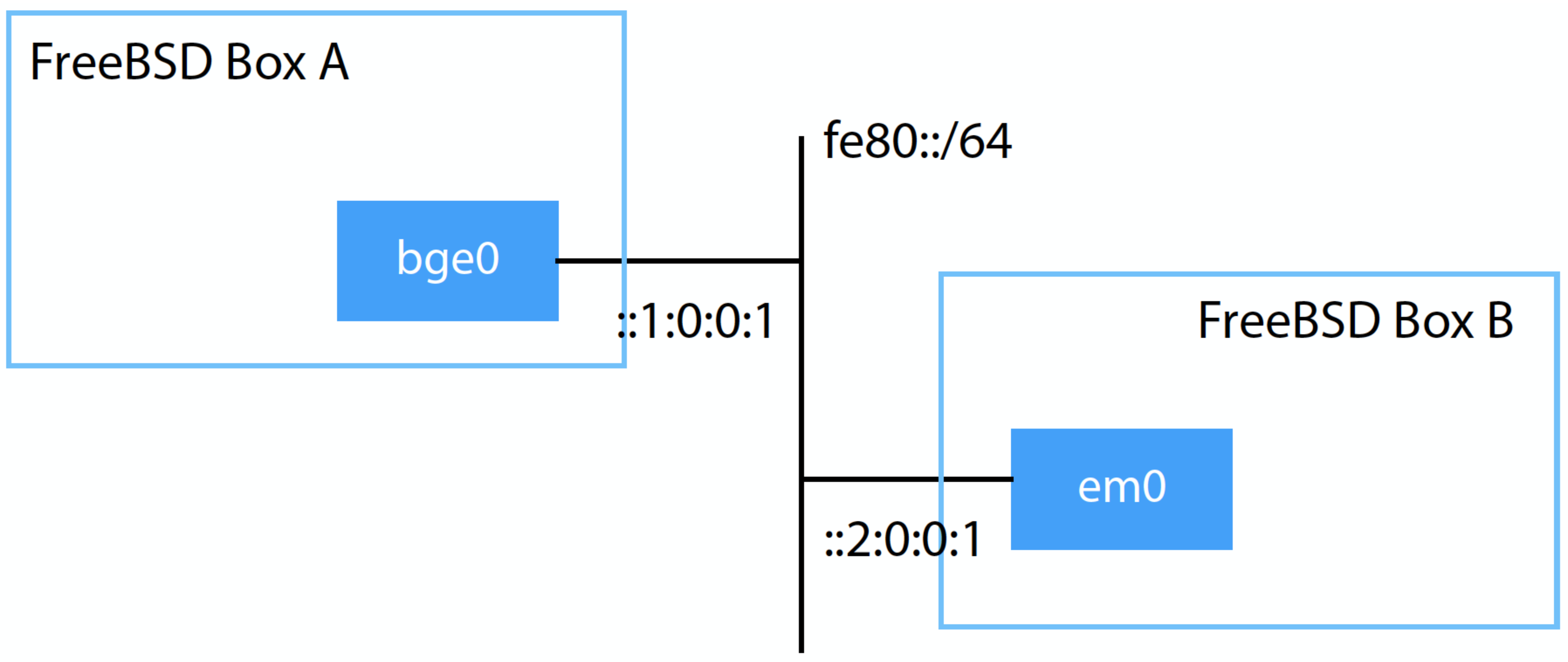


Figure 1: An example network with Box A and Box B

More specifically, the address resolution in NDP works in the following way. Figure 1 shows an example network. There are two machines, Box A and Box B.

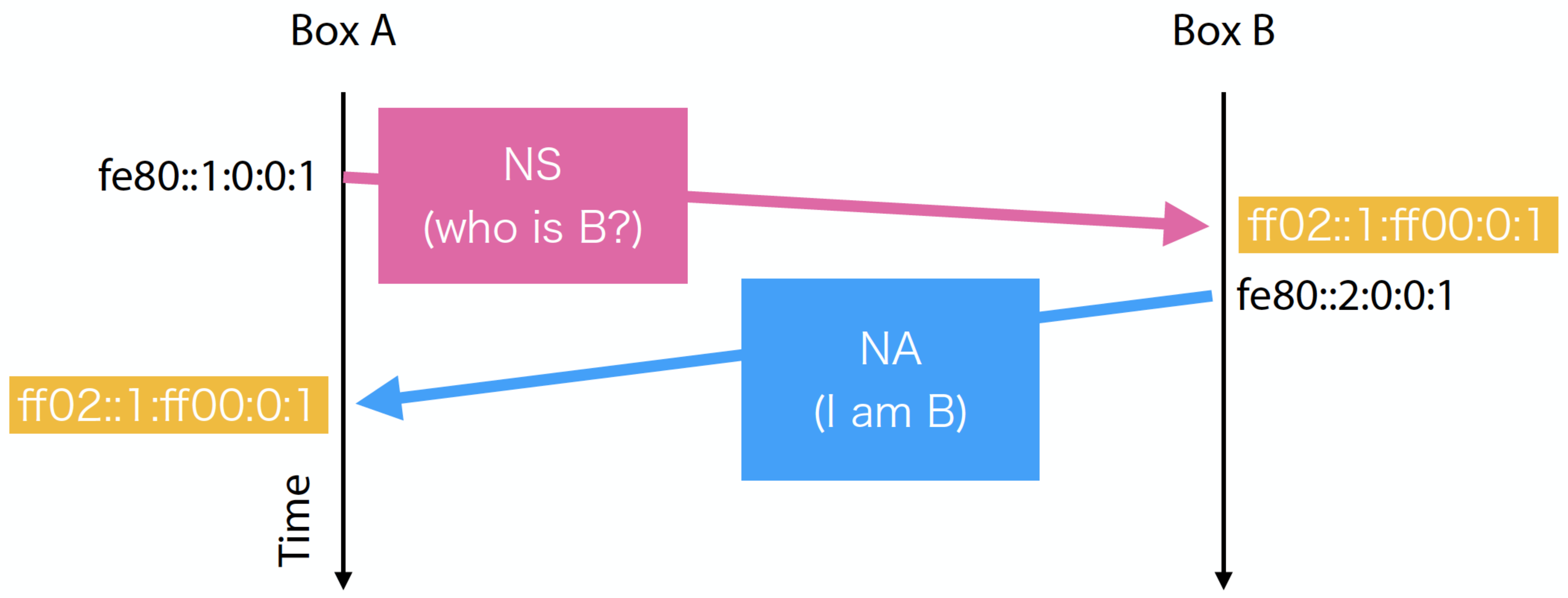


Figure 2: Timing diagram of Neighbor Solicitation and Neighbor Advertisement

When initiating a communication from A, it needs the L2 address of B. As shown in Figure 2, Box A sends a "Neighbor Solicitation" (NS) message of ICMPv6. It contains a pair of the LLA and the MAC address. The destination address of NS is Solicited-Node Multicast Address. The last three octets in the unicast addresses of A and B is `0:0:1`, so the address will be `ff02::1:ff00:0:1`. Box B will send back a "Neighbor Advertisement" (NA) message that contains another pair of the LLA and the MAC address on the Box B side. A knows B's LLA and MAC address in this exchange. Note that the Solicited-Node Multicast Address for A is the same as one for B by chance in this example. It depends on the IID of A and B.

Router Discovery and Autoconfiguration

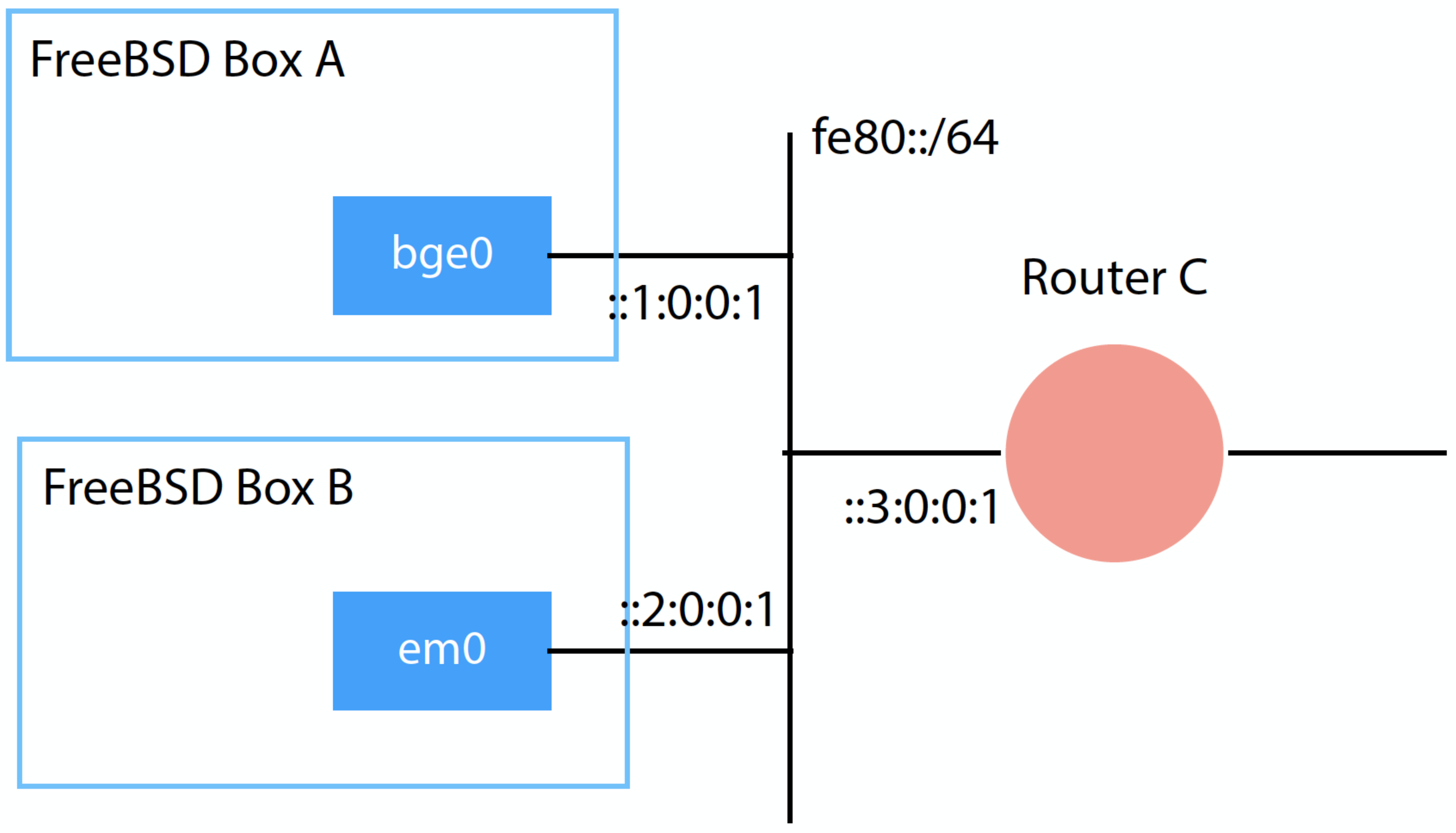


Figure 3: An example network with Box A, Box B, and Router C

Figure 3 shows another example network when there is a router. NS and NA messages are also exchanged for L2-L3 address resolution on this network. In addition, a host can discover the router's existence.

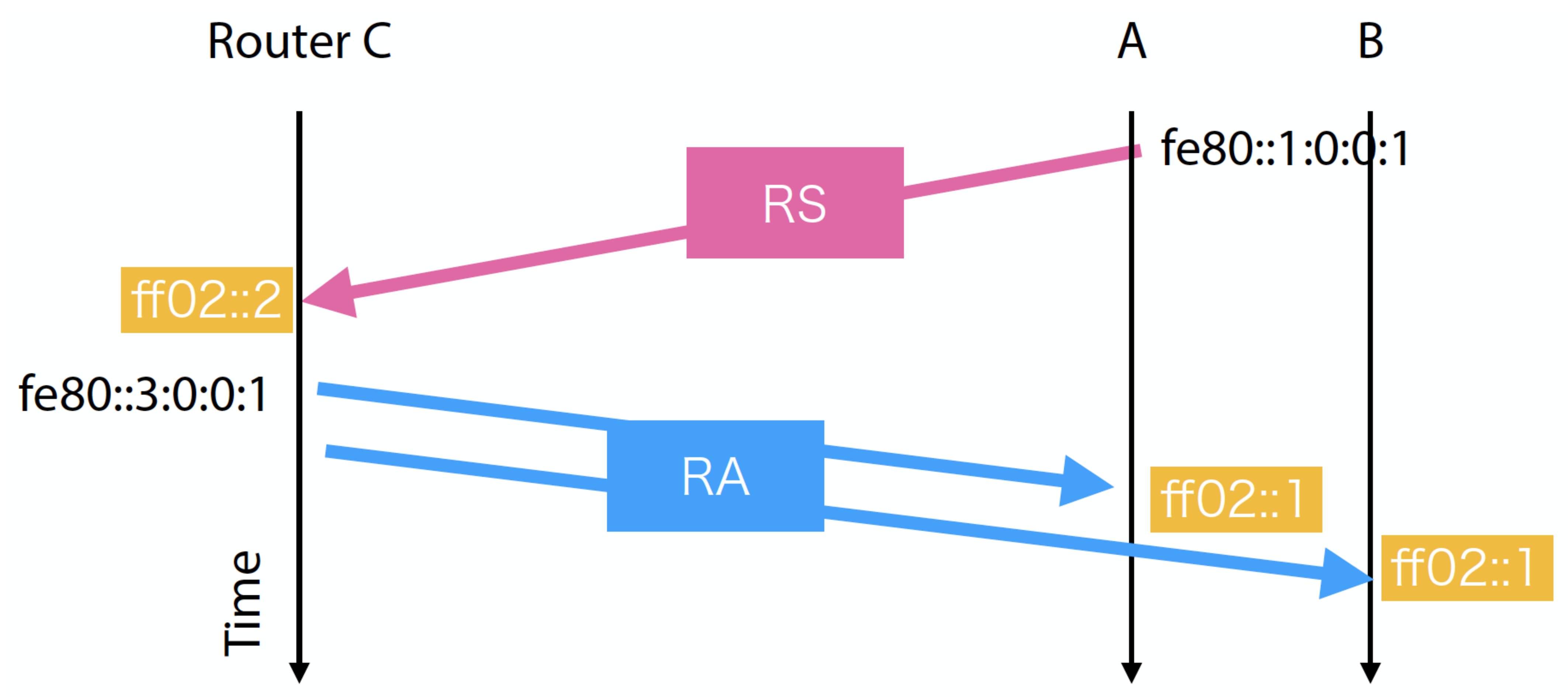


Figure 4: Timing diagram of Router Solicitation and Router Advertisement

A host can send a "Router Solicitation" (RS) message of ICMPv6 as shown in Figure 4. The destination address of RS is the all-routers multicast address. The connected routers will receive the RS and send back a "Router Advertisement" (RA) message. The destination address of RA is the all-nodes multicast address so that all hosts will receive it. An RA message contains network configuration parameters, such as MTU, subnet prefix, default router address, etc. The host nodes can configure themselves using the information. The default router address and the subnet prefix are sufficient to make the host ready to communicate with the IPv6 Internet.

As explained in the past columns, RA is sent by the `rtadvd(8)` daemon. RS can be sent by the `rtsol(8)` utility. The kernel handles NS and NA, so usually you do not need to be aware of them. Note that the kernel processes RAs only when the interface has the ACCEPT_RTADV flag. A router sends RAs periodically even if no RS is received, so you do not always need to run the rtsol(8) utility.

In this way, IPv6 uses various addresses for each specific purpose. Unlike IPv4, not all addresses are listed in the output of the `ifconfig` command. Multicast addresses can be shown using the `ifmcstat` command instead. In addition, the author wants to emphasize that an LLA on an interface is essential for NDP. Without an LLA, IPv6 does not work well. Actually, `IFDISABLED` flag, which is used to disable IPv6 communication on the interface, means disabling all of NDP traffic in the kernel. It blocks the NDP traffic only, but effectively disables IPv6.

Summary

This column has walked through how multiple addresses work in IPv6. You do not need to understand these kinds of details for communications using TCP or UDP. However, understanding of the configured addresses is quite important from the system administrator's perspective. You may want to configure additional access control lists or packet-filtering rules in some cases because an LLA on the box is another address where someone can access. On the other hand, blocking communications via LLAs may break NDP.

In the next column, several useful configuration tricks will be covered. They are based on the knowledge we know so far, and include missing parts such as DNS on IPv6 and DHCPv6.

Footnotes

[1] Global-scope Unicast Address. A routable address that consists of the prefix assigned from your ISP and the IID, interface identifier, which you have assigned or has been automatically assigned by SLAAC[2] or DHCPv6.

[2] Link-Local-scope Address. The prefix is always `fe80::/64`. This is unique only on the link and not routable.

[3] L3 stands for Layer 3 in the OSI reference model. This is a classic abstraction of communication protocols defined in `ISO/IEC 7498`. The TCP/IP protocol suite used for Internet does not fully follow this abstraction model. However, Layer 1, 2, and 3 are still helpful to understand the layer structure of Ethernet, IP, and UDP/TCP.

[4] This algorithm is explained in RFC 4291.

[5] An octet means 8-bit long data.

[6] Least Significant Bit. The lowest-order bit of a binary value.

[7] https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml

[8] This is defined in RFC 4861, "Neighbor Discovery for IP version 6 (IPv6)".

[9] RFC 1256, "ICMP Router Discovery Messages."

[10] RFC 826, "An Ethernet Address Resolution Protocol"

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 since 2007 has hosted AsiaBSDCon, an international conference on BSD-derived operating systems in Asia.