This material is translated based on the
“iDC/ISP/CATV Servers Hands-On Seminar”
by Koichi Kunitake, BeaconNC Inc.,
on December 17-18, 2009,
organized by
Task Force for IPv4 Exhaustion, Japan
http://www.kokatsu.jp/

iDC/ISP/CATV Servers

BeaconNC Inc. Data Center Dept.

Koichi Kunitake

Today's topic

- Purpose
  - Acquire knowledge for server operation in IPv6 environment on Linux OS

Note: We will use CentOS (RedHat stream) and GNU/Debian for the distribution dependent issues

Overview

- IPv6 main feature
- Basic knowledge for using IPv6 on Linux
  - IPv6 address text representation
  - IPv6 address type
  - ULA
  - Header information
  - Dual Stack
  - DNS IPv6 support

- IPv6 main feature
- Basic knowledge for utilizing IPv6 on Linux
  - IPv6 address text representation
  - IPv6 address type
  - ULA
  - Header information
  - Dual Stack
  - DNS IPv6 support

IPv6 main feature

- Huge address space (from 32bits to 128bits)
- Address auto configuration
  - Aims to lower the administration cost
- IPsec standard implementation
- MobileIP
  - More sophisticated architecture than IPv4 MobileIP

Others: Simple header architecture, disable fragmentation at the intermediate routers and introduction of extension headers

IPv6 huge address space

- Suppose IPv4 address space is 1, then IPv6 address space is…

79228162514264337593543950336

You can assign the IPv6 addresses, as many as current IPv4 addresses, to each person on the earth, and there are still good quantity of remained addresses.

- However, the netmask architecture is different in IPv4/IPv6, and cannot compare the address space that easy like above example

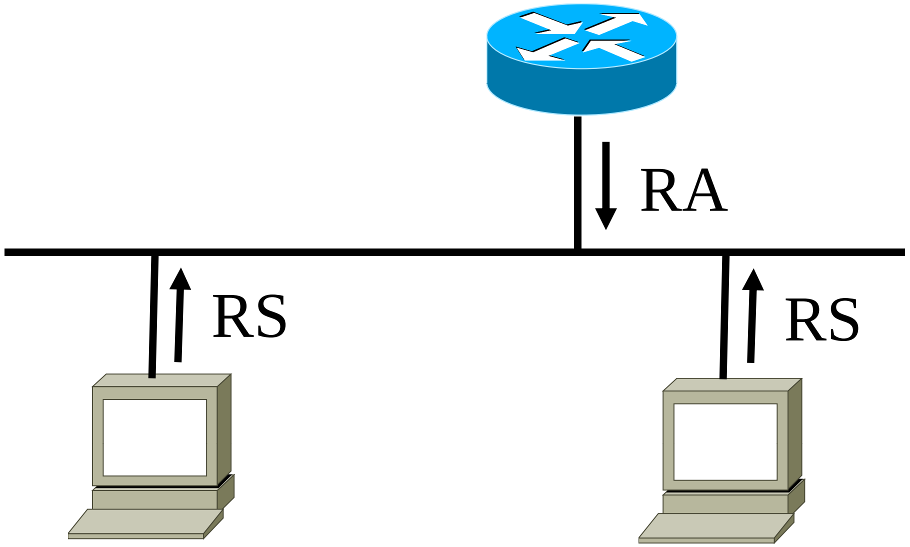
IPv6 huge address space (cont.)

- It is no use to compare the numbers of IPv4 and IPv6 addresses, but IPv6 address is huge enough
- Usable segments (*)
  - IPv4 : 1073741824
  - IPv6 : 18446744073709551616

(*) Utilize /30 in IPv4 and /64 in IPv6 for calculation. This is just to get the idea as all the address space cannot be used as global addresses

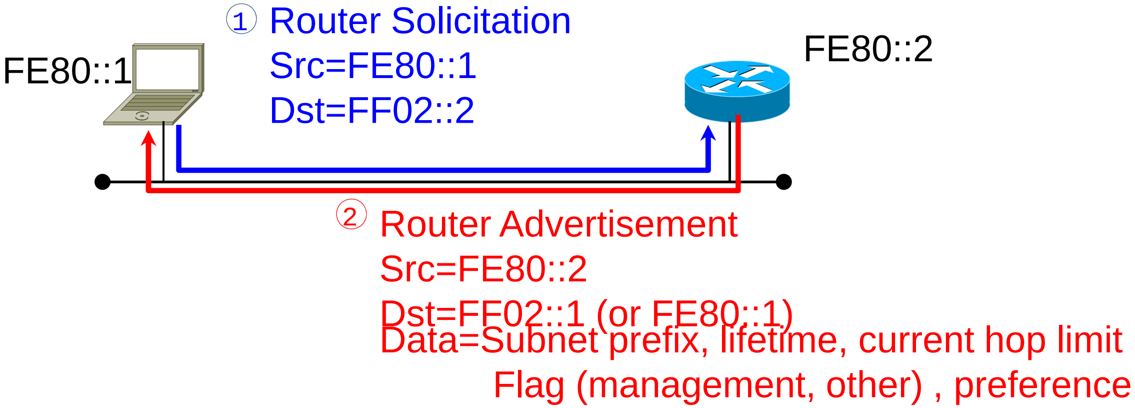
Address auto configuration

- Address auto configuration function is defined as standard function in IPv6 to support immense number of devices connected to the network
  - You can also use DHCPv6



Router Solicitation/Router Advertisement

- RS's destination address is FF02::2, and the Hop Limit is 255
- RA's destination address is FF02::1 or the RS's source address, and the Hop Limit is 255
- Current Hop Limit field in RA sets the node's hop limit
- If M-flag is 0, then stateless address will be automatically configured, and if it is 1, address will be assigned using DHCPv6
- If O-flag is 1, information other than the address will be delivered by DHCPv6
- Only the default router sets the Router Lifetime to more than 1 (less than 65535)
- DRP (Default Router Preference: RFC4191) defines how to inform default router's priority
  - High(01), Medium(00), Low(11)
  - Both routers and nodes have to support the function to enable this function



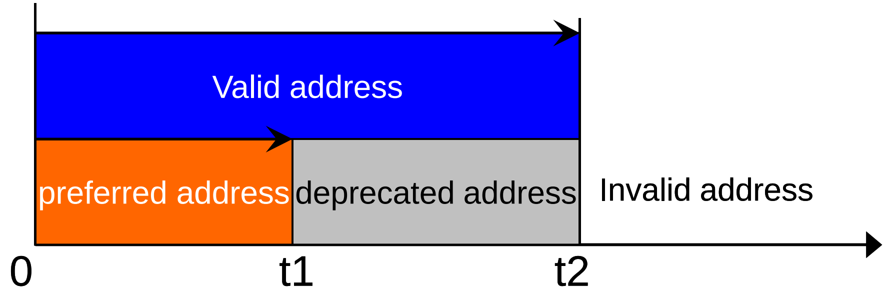
① Router Solicitation
Src=FE80::1
Dst=FF02::2

② Router Advertisement
Src=FE80::2
Dst=FF02::1 (or FE80::1)
Data=Subnet prefix, lifetime, current hop limit
Flag (management, other) , preference

IPv6 address state and lifetime

- tentative address
  - Address that will not be assigned to interfaces. Used for ND messages only. Confirms the uniqueness of addresses using DAD.
- preferred address
  - Interface’s address. Unique and communication-able address.
- deprecated address
  - Valid address, but initiating communications with this address is not recommended
- valid address
  - Preferred addresses and deprecated addresses
- Invalid address
  - When the valid address’s timeout occurs, it is changed to invalid address

Address valid lifetime and status change

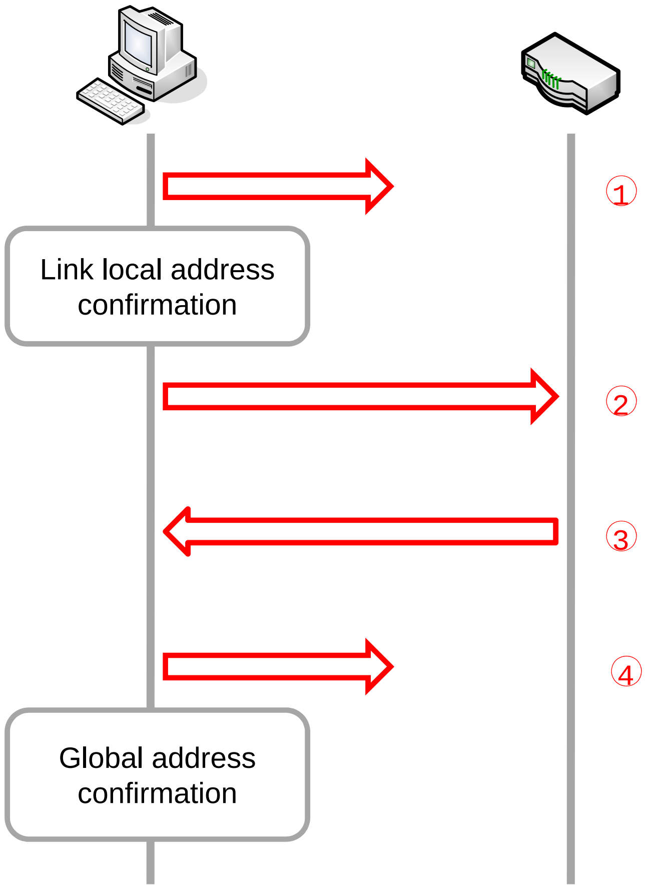


t1：preferred lifetime
t2：valid lifetime

Stateless auto-configuration flow

fe80::211:22ff:fe33:4455
2001:db8::211:22ff:fe33:4455
MAC:00:11:22:33:44:55

fe80::211:22ff:fe66:7788
2001:db8::211:22ff:fe66:7788
MAC:00:11:22:66:77:88



① Neighbor Solicitation (NS)
When there are no neighbor advertisement, the target address can be utilized
<Duplicated Address Detection>

Solicited-Node Multicast

Src MAC	00:11:22:33:44:55
Dst MAC	33:33:FF:33:44:55
Src IPv6	:: (undefined address)
Dst IPv6	ff02::1:ff33:4455
ICMPv6 Type Target	135 fe80::211:22ff:fe33:4455

② Router Solicitation (RS)
All router Multicast
Send to (ff02::2)

Src MAC	00:11:22:33:44:55
Dst MAC	33:33:00:00:00:02
Src IPv6	fe80::211:22ff:fe33:4455
Dst IPv6	ff02::2
ICMPv6 Type	133

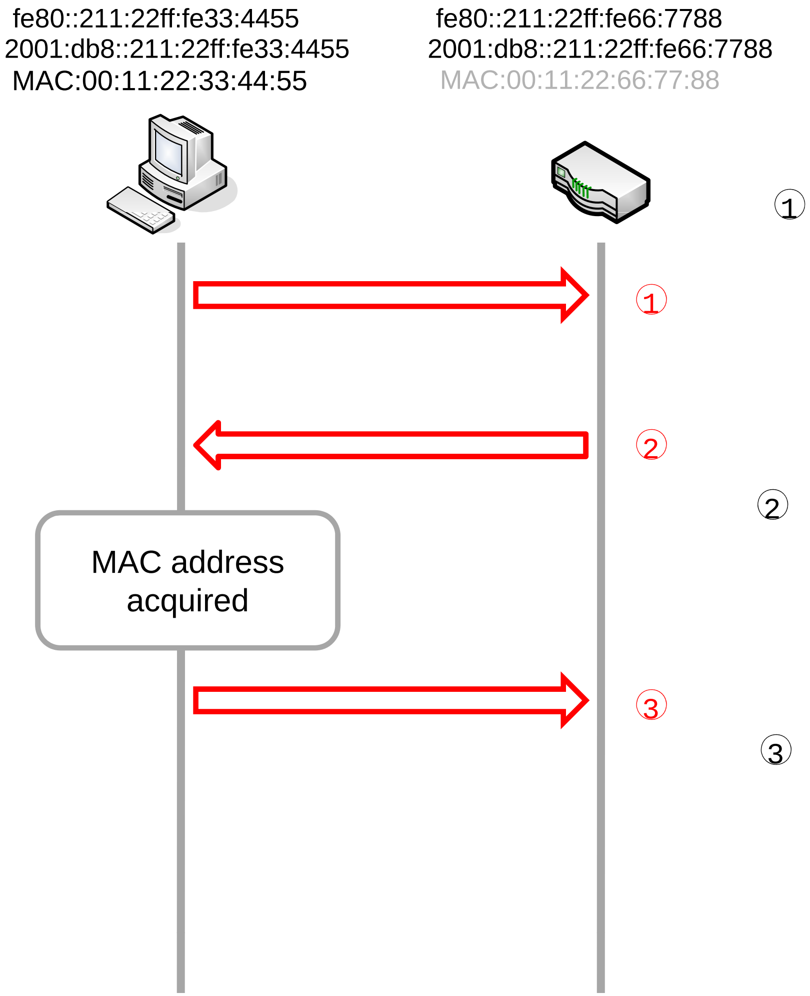
③ Router Advertisement
All node multicast
Send to (ff02::1)
Generate global address utilizing the acquired prefix

Src MAC	00:11:22:66:77:88
Dst MAC	33:33:00:00:00:01
Src IPv6	fe80::211:22ff:fe66:7788
Dst IPv6	ff02::1
ICMPv6 Type Prefix	134 2001:db8::

④ Neighbor Solicitation
When there are no neighbor advertisement, the target address can be utilized.
When there are replies, have to re-construct the address
<Duplicated Address Detection>

Src MAC	00:11:22:33:44:55
Dst MAC	33:33:FF:33:44:55
Src IPv6	:: (undefined address)
Dst IPv6	ff02::1:ff33:4455
ICMPv6 Type Target	135 2001:db8::211:22ff:fe33:4455

Link layer address resolution flow



fe80::211:22ff:fe33:4455
2001:db8::211:22ff:fe33:4455
MAC:00:11:22:33:44:55

fe80::211:22ff:fe66:7788
2001:db8::211:22ff:fe66:7788
MAC:00:11:22:66:77:88

MAC address acquired

① Neighbor Solicitation (NS)
Search communication partner's MAC address
When there are no neighbor advertisement, the node is not onlink

Src MAC	00:11:22:33:44:55
Dst MAC	33:33:FF:66:77:88
Src IPv6	fe80::211:22ff:fe33:4455
Dst IPv6	ff02::1:ff66:7788
ICMPv6 Type Target	135 2001:db8::211:22ff:fe66:7788

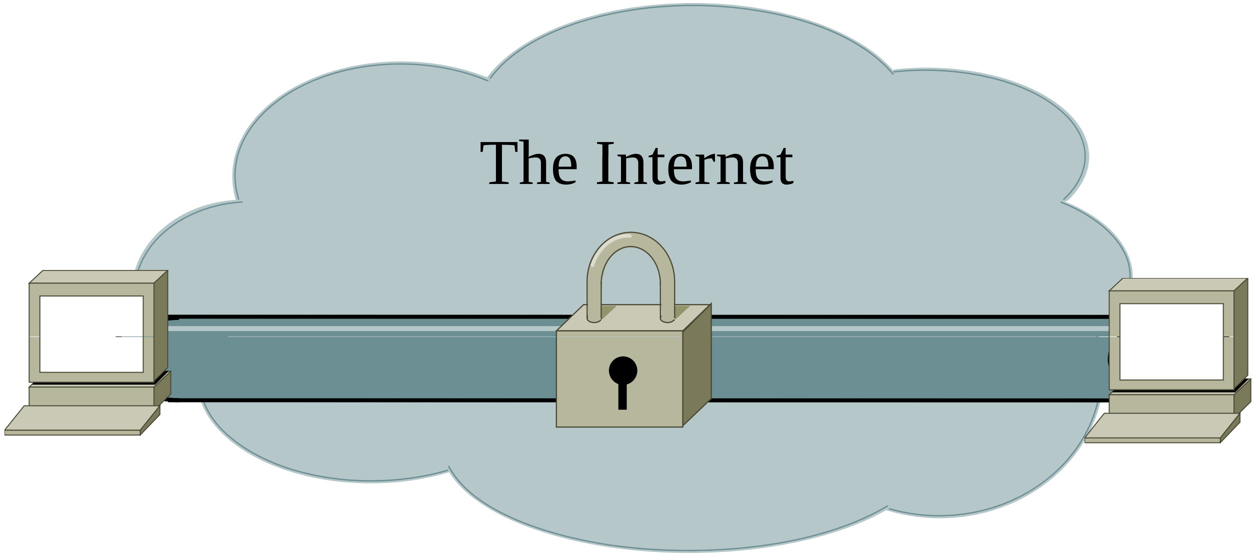
② Neighbor Advertisement
Node with the target address replied.
Any node can reply

Src MAC	00:11:22:66:77:88
Dst MAC	00:11:22:33:44:55
Src IPv6	fe80::211:22ff:fe66:7788
Dst IPv6	fe80::211:22ff:fe33:4455
ICMPv6 Type Target Target MAC	136 2001:db8::211:22ff:fe66:7788 00:11:22:66:77:88

③ Start communication

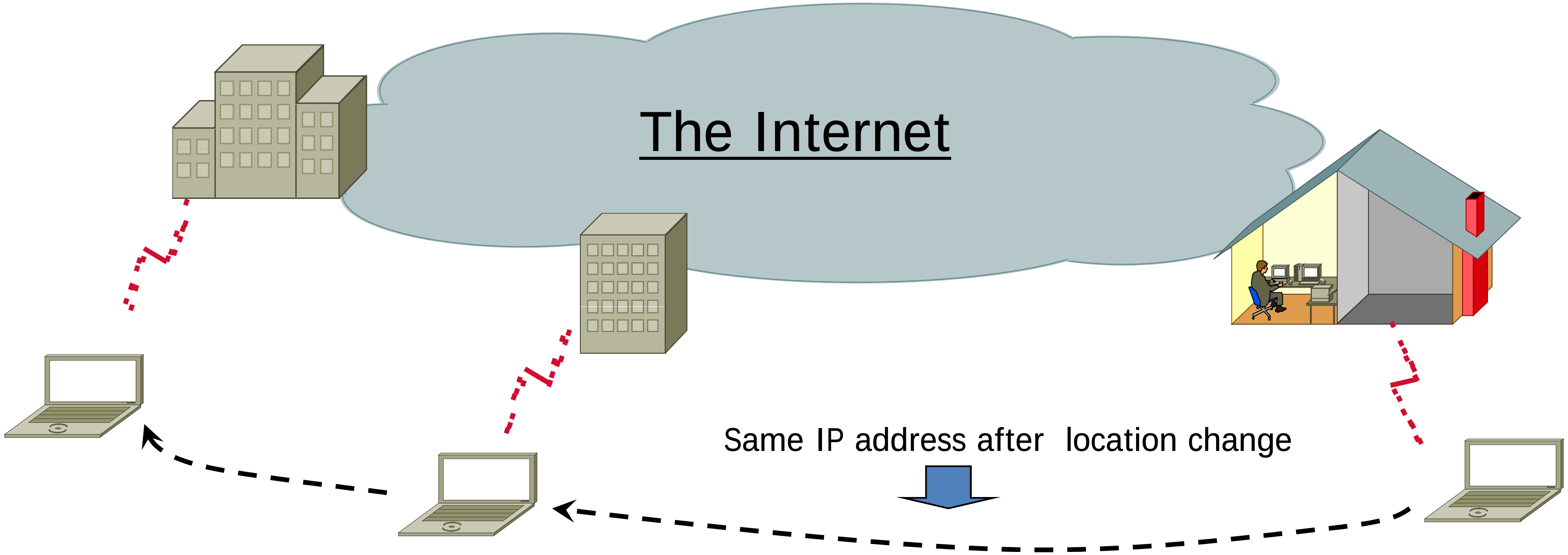
IPsec: standard function

- Guarantee secure communication at the IP layer
  - – No need to remodel the applications



MobileIP

- Compatibility with ubiquitous
  - Same IP address can be utilized anywhere
  - More realistic system than IPv4 MobileIP



Basic knowledge for utilizing IPv6 on Linux

- IPv6 main feature
- Basic knowledge for utilizing IPv6 on Linux
  - **IPv6 address text representation**
  - IPv6 address type
  - ULA
  - Header information
  - Dual Stack
  - DNS IPv6 support

IPv6 address text representation

- IPv4 address text representation
  - Example) 192.0.2.1

    Decimal number notation, utilize “.” as delimiter

- IPv6 address text representation
  - Example) fe80:0000:0000:0000:02d0:b7ff:fea0:beea

    Hex number notation, utilize “:” as delimiter

Abbreviation in IPv6

fe80:0000:0000:0000:02d0:b7ff:fea0:beea

- Leading zero in each part can be abbreviated

  fe80:0000:0000:0000:**2d0**:b7ff:fea0:beea

- Compress the zeros with "::" Can be done only once for each address

  fe80**::**02d0:b7ff:fea0:beea

Therefore, the notation becomes fe80::2d0:b7ff:fea9:beea

How about the following case?

- 2001:0db8:0000:0000:fff0:0000:0000:000f

  2001:db8::fff0:0:0:f

  or

  2001:db8:0:0:fff0::f

Note) Cannot be written as 2001:db8::fff0::f

- IPv6 main feature
- Basic knowledge for utilizing IPv6 on Linux
  - IPv6 address text representation
  - IPv6 address type
  - ULA
  - Header information
  - Dual Stack
  - DNS IPv6 support

IPv6 address type

IPv6 address classification (Example)

- Behavior
- Scope
- Special purpose address

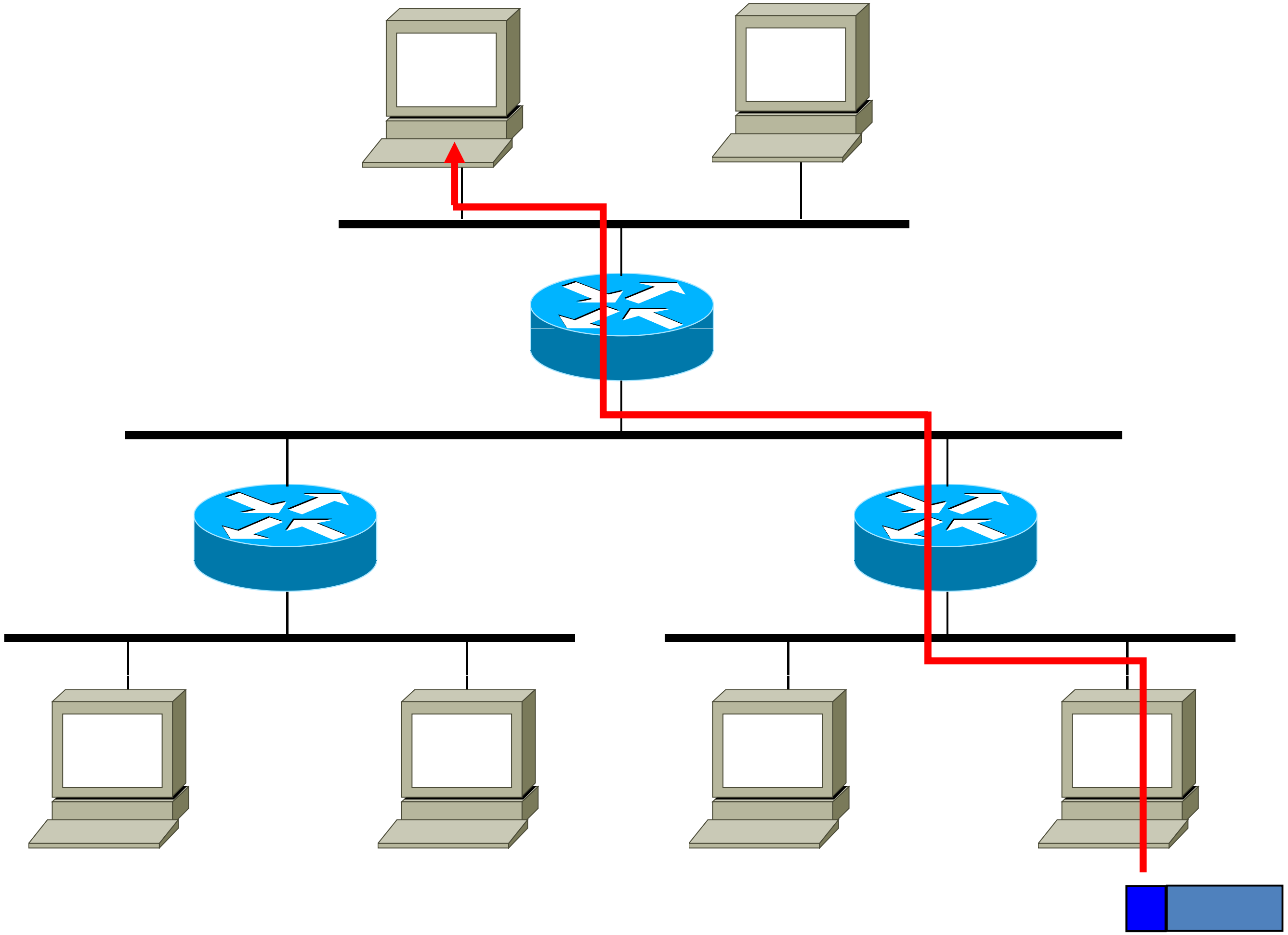
Classification using packet behavior

- Unicast Address
- Multicast Address
- Anycast Address

Unicast Address

- The address for each network interface
- Used for one to one communication (Generally, this address is used)

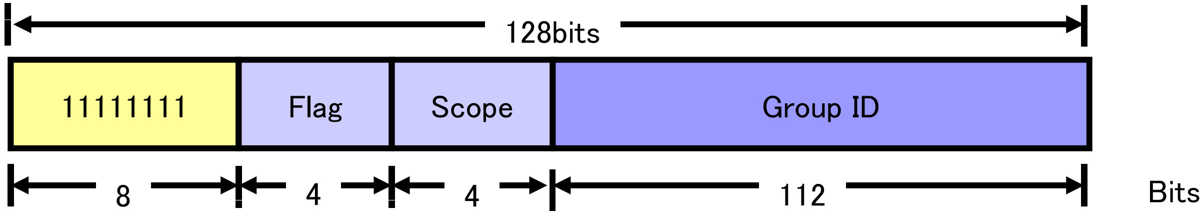
Unicast Address



Multicast Address

- Specifies a group address
  - When a packet is transmitted to a multicast address, all the interface that is in the specified multicast group will receive the packet.
  - IPv6 has no broadcast address like IPv4, but IPv6 uses multicast for same purpose.

Multicast Address

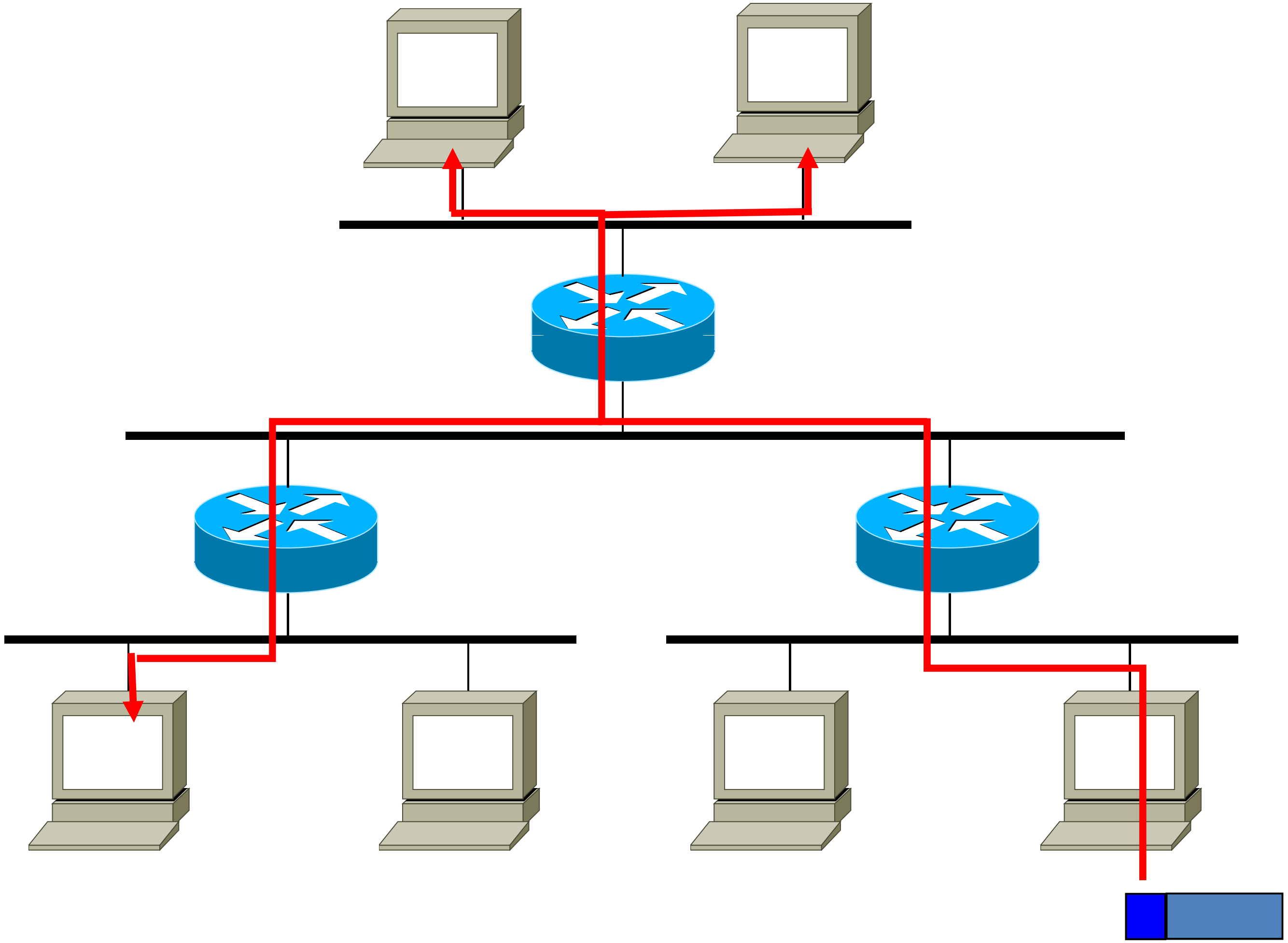


T=0 permanent multicast address
T=1 temporary multicast address

Multicast scope: 4-bit multicast scope value is used to limit the scope of the multicast group.

0 reserved
1 node-local scope
2 link-local scope
3 (unassigned)
4 (unassigned)
5 site-local scope
6 (unassigned)
7 (unassigned)
8 organization-local scope
9 (unassigned)
A (unassigned)
B (unassigned)
C (unassigned)
D (unassigned)
E global scope
F reserved

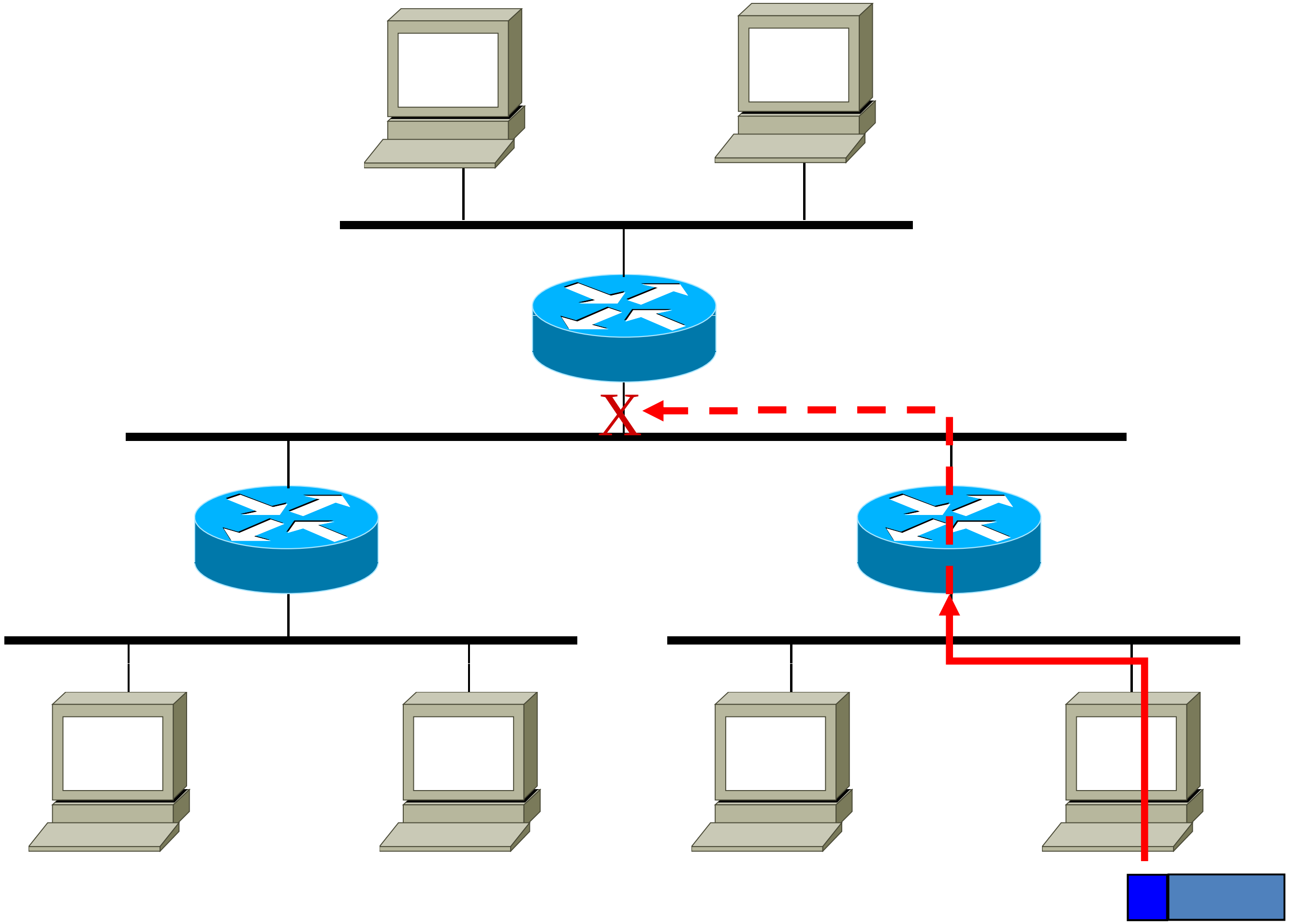
Multicast Address



Anycast Address

- An address that specifies a group address (similar to multicast address)
  - The difference with the Multicast Address is that the packet will not be send to all the interfaces belonging to the group. When delivered to one interface, the packet will not be sent to other interfaces.

Anycast Address

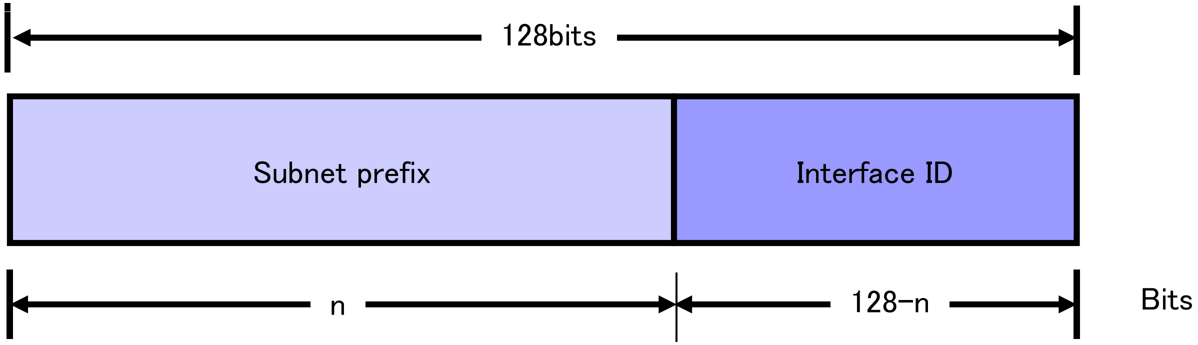


Classification by scope

- Global Address
- Site-local Address (obsolete)
  - Changed to Unique Local Address (ULA)
- Link-Local Address

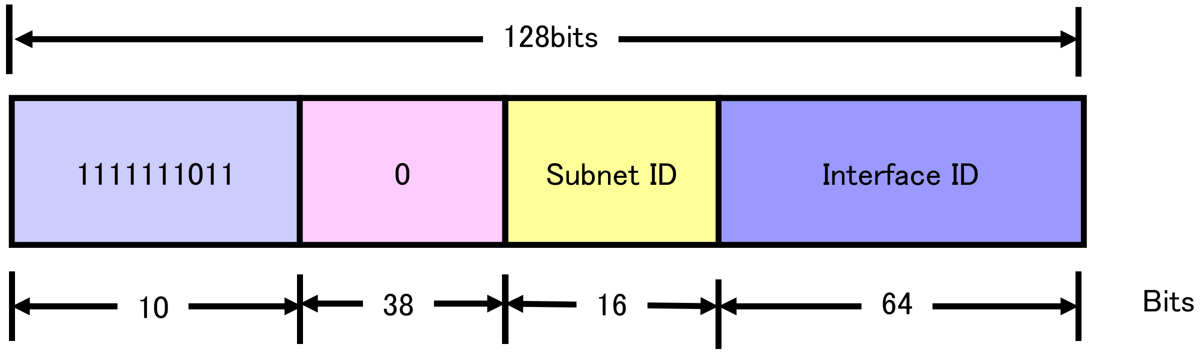
Global Address

- Same with IPv4 Global Address
- Worldly unique identifier



Site-Local Address (Obsolete)

- Private Address in IPv4
- Packets from this address will not transmitted to other organizations
- Got obsolete as there are many problems (taken over by Unique Local IPv6 Unicast Address)

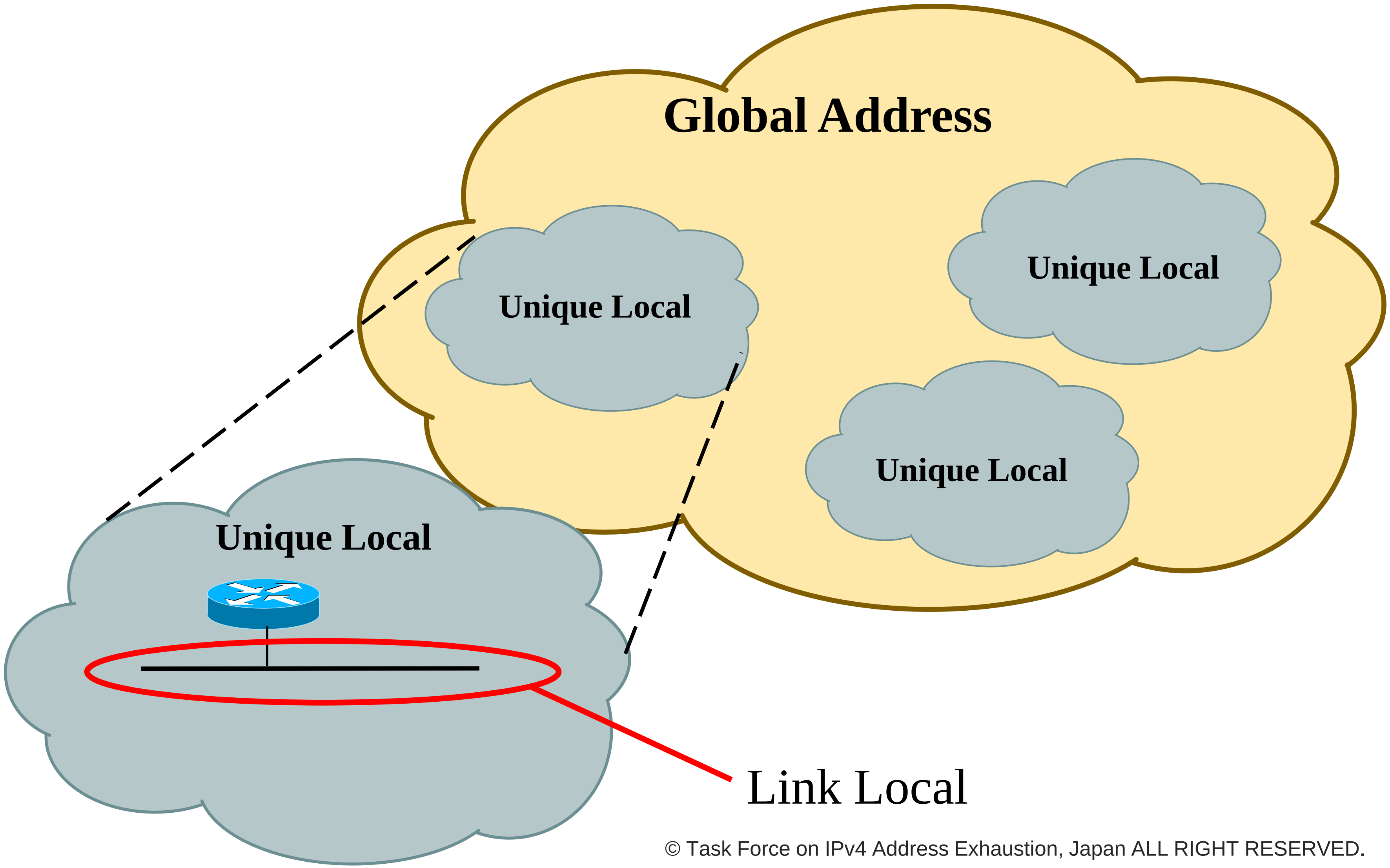


Unique Local IPv6 Unicast(ULA) is different than Site-Local Address, and its address can be used like Private Address. The cases are VERY RARE that ULA addresses are duplicated with other organizations

Link-Local Address

- Address that is valid only within the node’s directly connected link
- Automatically generated even when there are no routers that sends RA in the neighboring link
- Well-known Link-Local Address
  - ff02::1 (This is also Multicast Address)
    - – All the nodes in one link are in this address group
  - Have to be utilized with scope ID
    - – Example: “ping6 –I eth0 ff03::1”, “ssh fe80::dead:beaf%eth0”

Each address scope



Special Purpose Address

- Unspecified Address
- Loopback address
- IPv4-compatible address
- IPv4-mapped address

Unspecified Address

- Indicates the absence of an address
- System initializing host with no IP address sometimes uses this address as source address

0000:0000:0000:0000:0000:0000:0000:0000

⇩

::

Loopback address

- Used by a node sending an IPv6 packet to itself. Same with IPv4’s 127.0.0.1

0000:0000:0000:0000:0000:0000:0000:0001

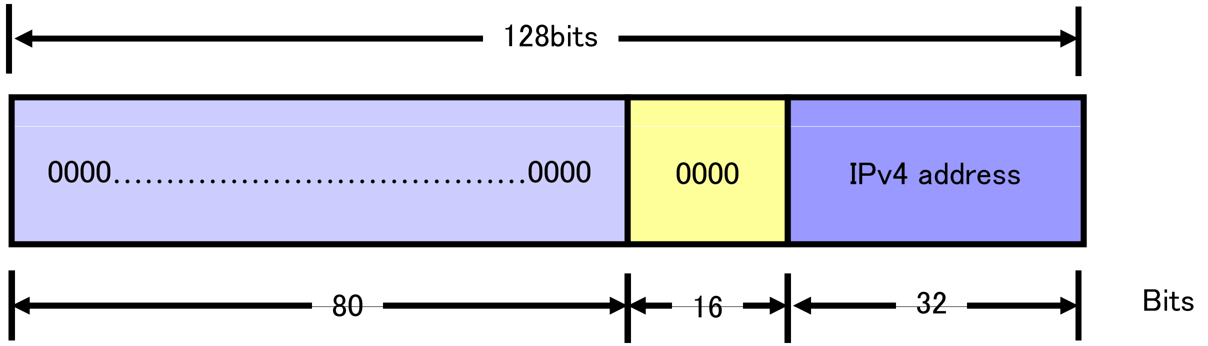
⇩

::1

IPv4-compatible Address (obsolete)

- Utilized for communications among IPv6 nodes connected only to IPv4 network
  - IPv4 address is described in binary digit notation for easy understanding

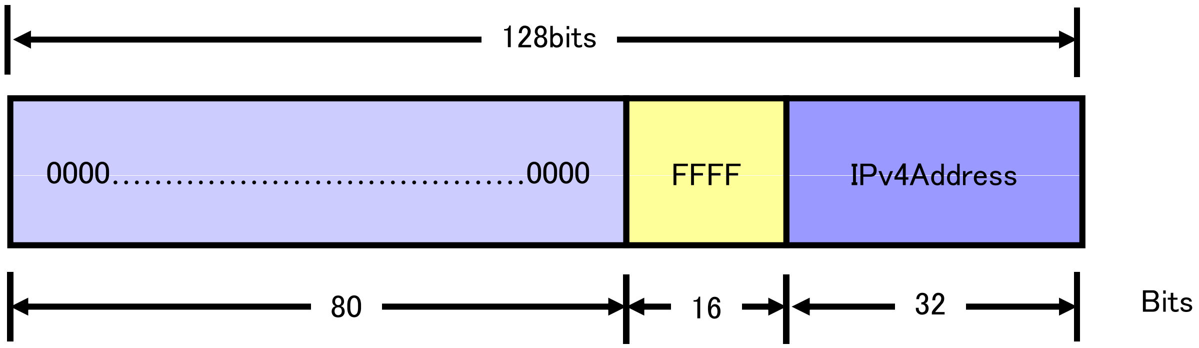
Example) “192.168.0.1” => “::192.168.0.1”



IPv4-mapped Address

- Utilized for communications between IPv6 nodes with nodes supporting only IPv4 address
  - IPv4 address is described in binary digit notation for easy understanding

Example: “192.168.0.1” => “::ffff:192.168.0.1”



- IPv6 main feature
- Basic knowledge for utilizing IPv6 on Linux
  - IPv6 address text representation
  - IPv6 address type
  - ULA
  - Header information
  - Dual Stack
  - DNS IPv6 support

Unique Local IPv6 Unicast Addresses

- Globally unique prefix
- Can be filtered easily by site boundaries
- Can be utilized without external link (e.g. connection to ISP)
- Even when the routes are leaked to outside network, they will not be duplicated with other addresses
- Applications can utilize this address as same as global addresses

Format

7	41	16	64
Prefix	Global ID	Subnet ID	Interface ID

- Prefix : Example (FC00::/7)
- Global ID :Global Identifier
- Subnet ID :ID for subnet within a site
- Interface ID

Global ID

- Utilizes MD5 based pseudo-random algorithm
- Assumes 2 prefix
  - FC00::/8   Centrally assigned
  - FD00::/8   Locally assigned
- Centrally assigned
- Locally assigned

Centrally assigned

- Address allocation organization assures there will be no address conflict
- Can be utilized permanently without periodic payment
- However, service charge of 10 euro (cannot be refund) is required for each allocation

…… are discussed (but not finalized yet)

Locally assigned

- Global ID is generated using Pseudo-random algorithm
- No need to register to the address allocating organizations
  - – How to generate ULA by yourself?
    - Utilize ULA Generator

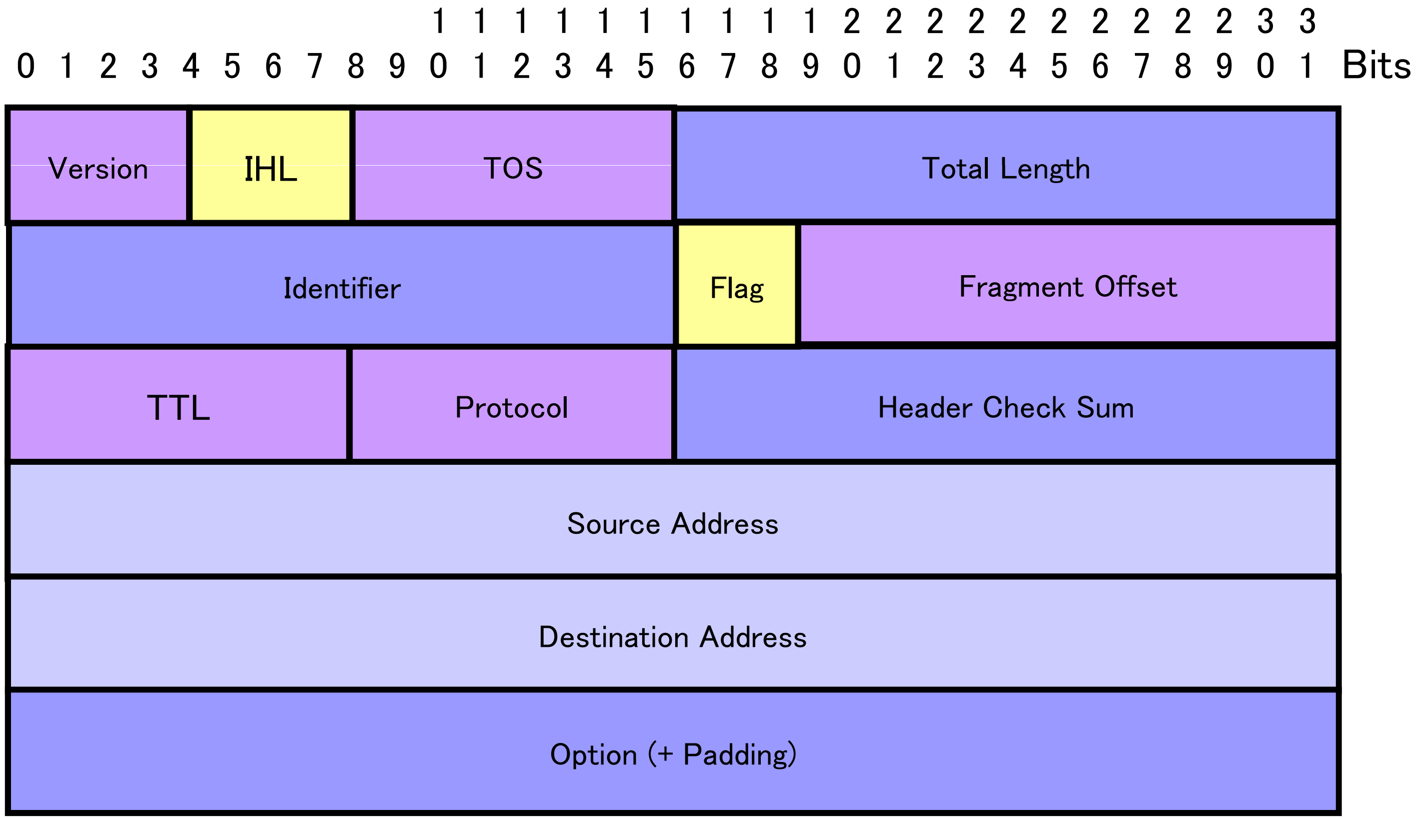
      http://www.kame.net/~suz/gen-ula.html

- IPv6 main feature
- Basic knowledge for utilizing IPv6 on Linux
  - IPv6 address text representation
  - IPv6 address type
  - ULA
  - Header information
  - Dual Stack
  - DNS IPv6 support

Header Information

- Simplified header format
  - 40 byte fixed length (Utilizes extension headers)
- Stopped utilizing check sum
  - UDP check sum is indispensable

IPv4 header



```
                    1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 Bits
```

Version | IHL | TOS | Total Length

Identifier | Flag | Fragment Offset

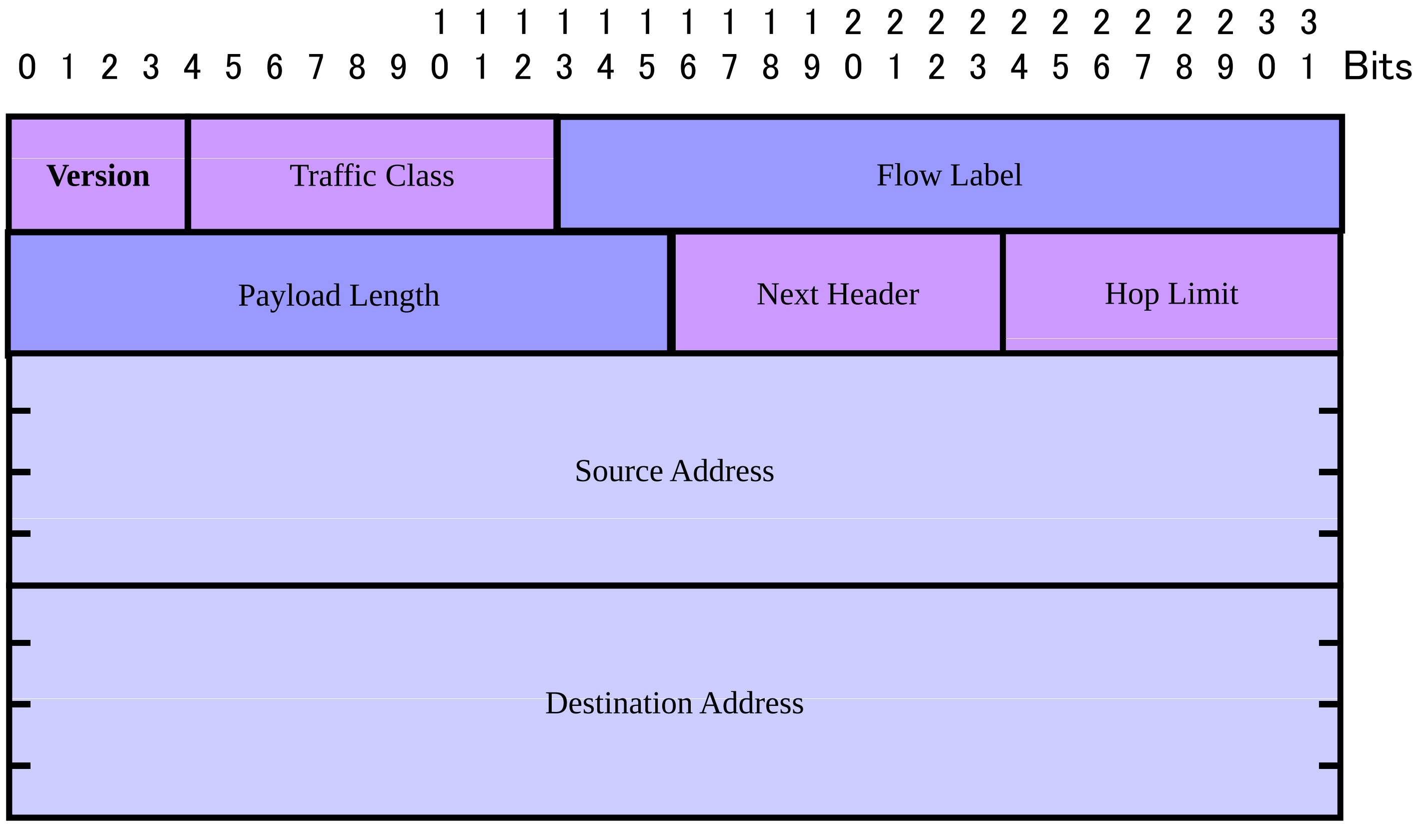
TTL | Protocol | Header Check Sum

Source Address

Destination Address

Option (+ Padding)

IPv6 header



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 Bits

Version

Traffic Class

Flow Label

Payload Length

Next Header

Hop Limit

Source Address

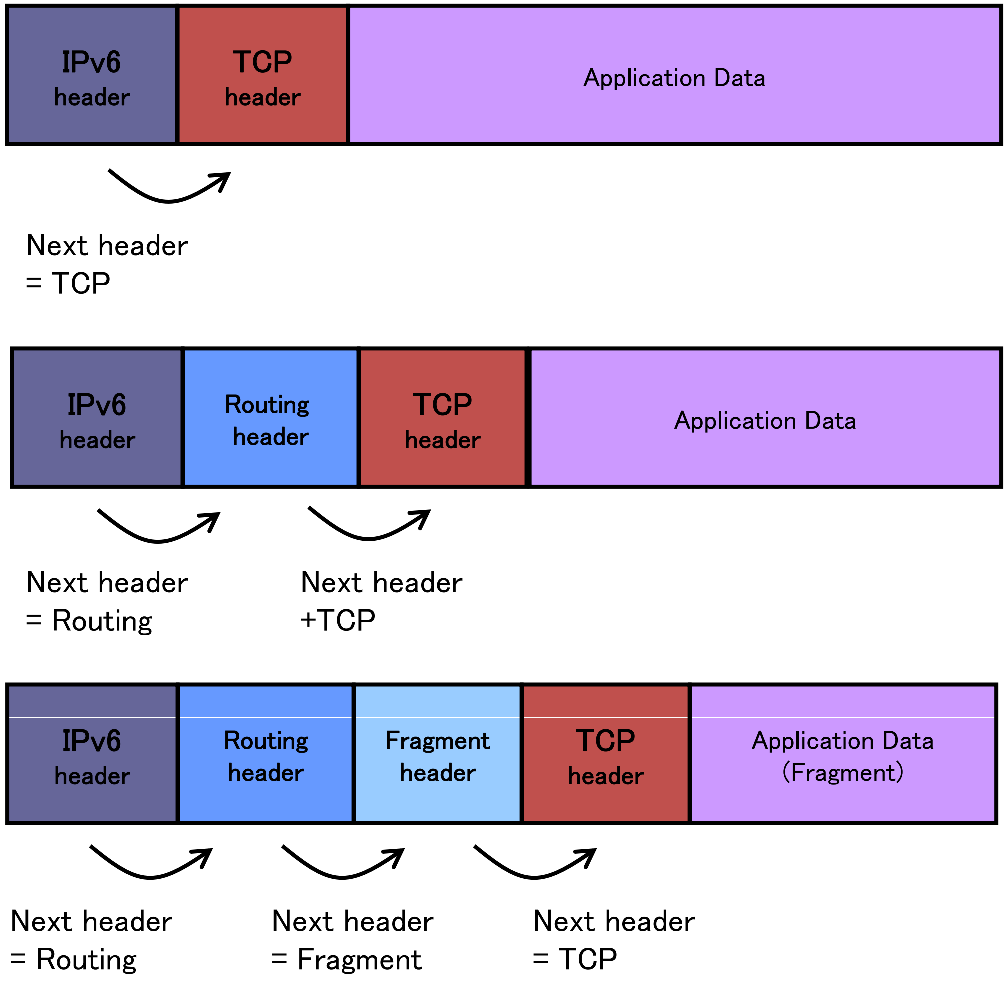
Destination Address

- Simplified compared to IPv4
- Option headers became obsolete, and extension headers were implemented instead

Extension header

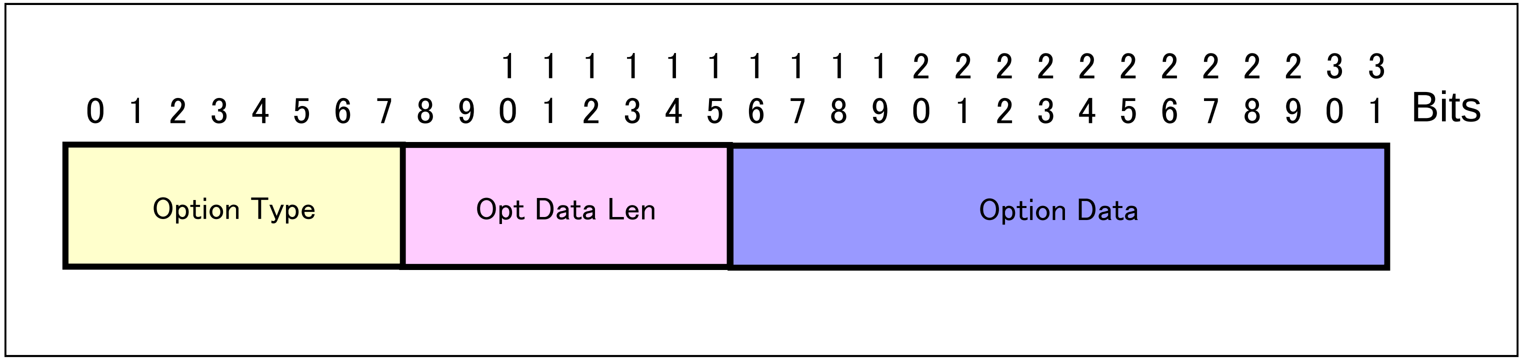
- Hop-by-Hop Options Header
  - Includes information that have to be referred by every node on the intermediate route.
- Destination Options Header
  - Specify the recompiling order at the destination address node
- Routing Header
  - Specifies at least one or more intermediate nodes
- Fragment Header
  - Specifies the packet is fragmented. Unlike IPv4, the packets will not be fragmented at the intermediate routers.
- Authentication Header
- Encapsulating Security Payload Header
  - These 2 headers are used by IPsec to enable secure communication at IP level

Beaded extension headers



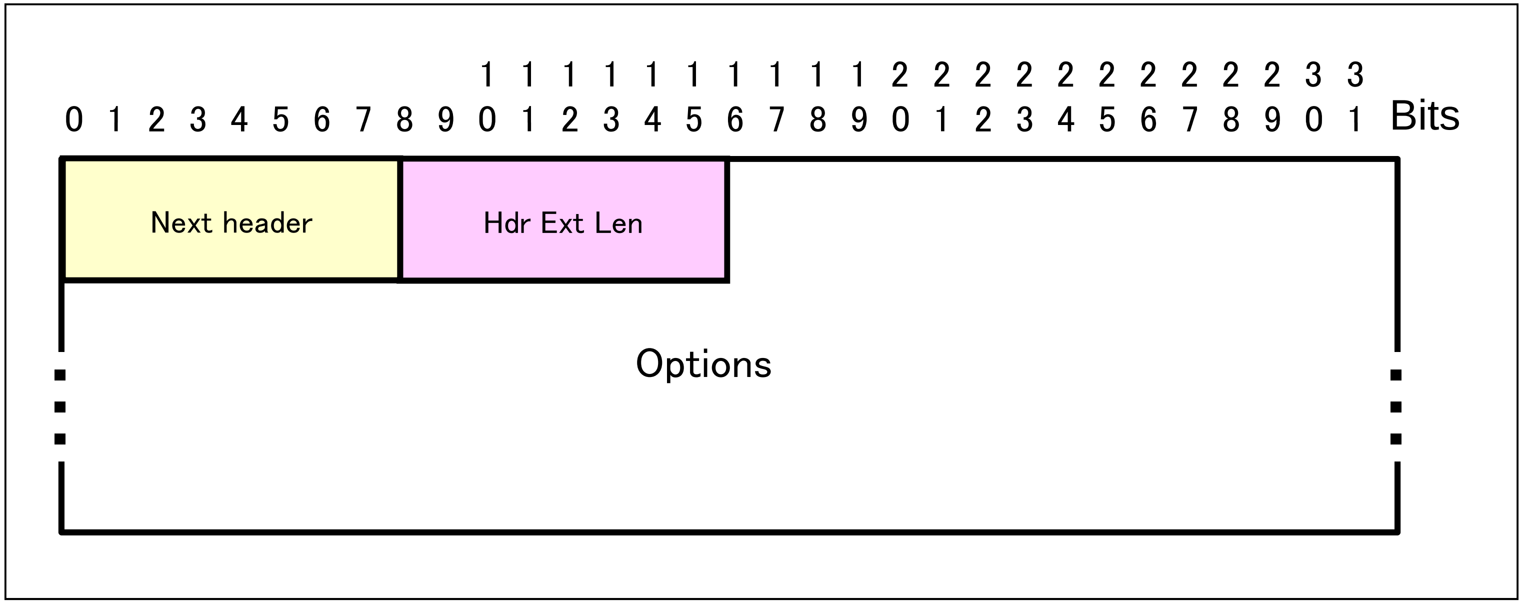
Extension header detail

- Basic Type



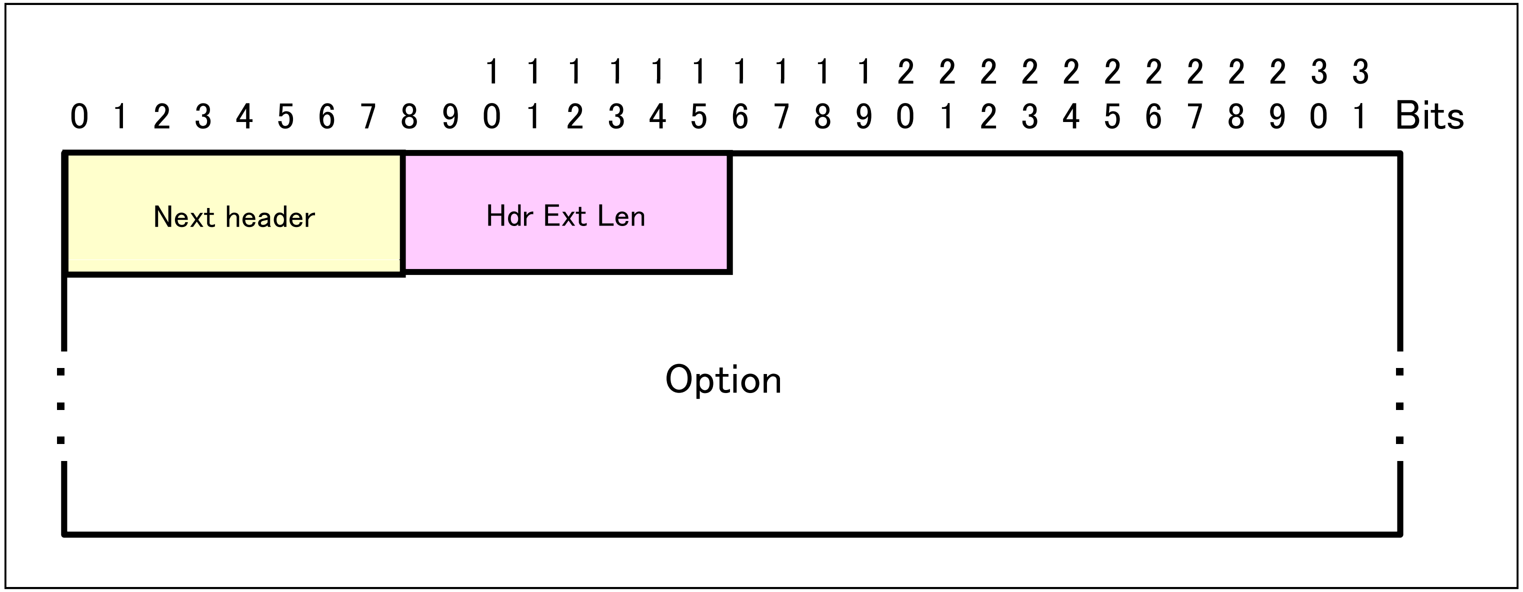
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Option Type	Opt Data Len	Option Data

Hop-by-Hop options header



- The options will be processed at all intermediate routers

Destination Headers Option



```
                    1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  Bits
```

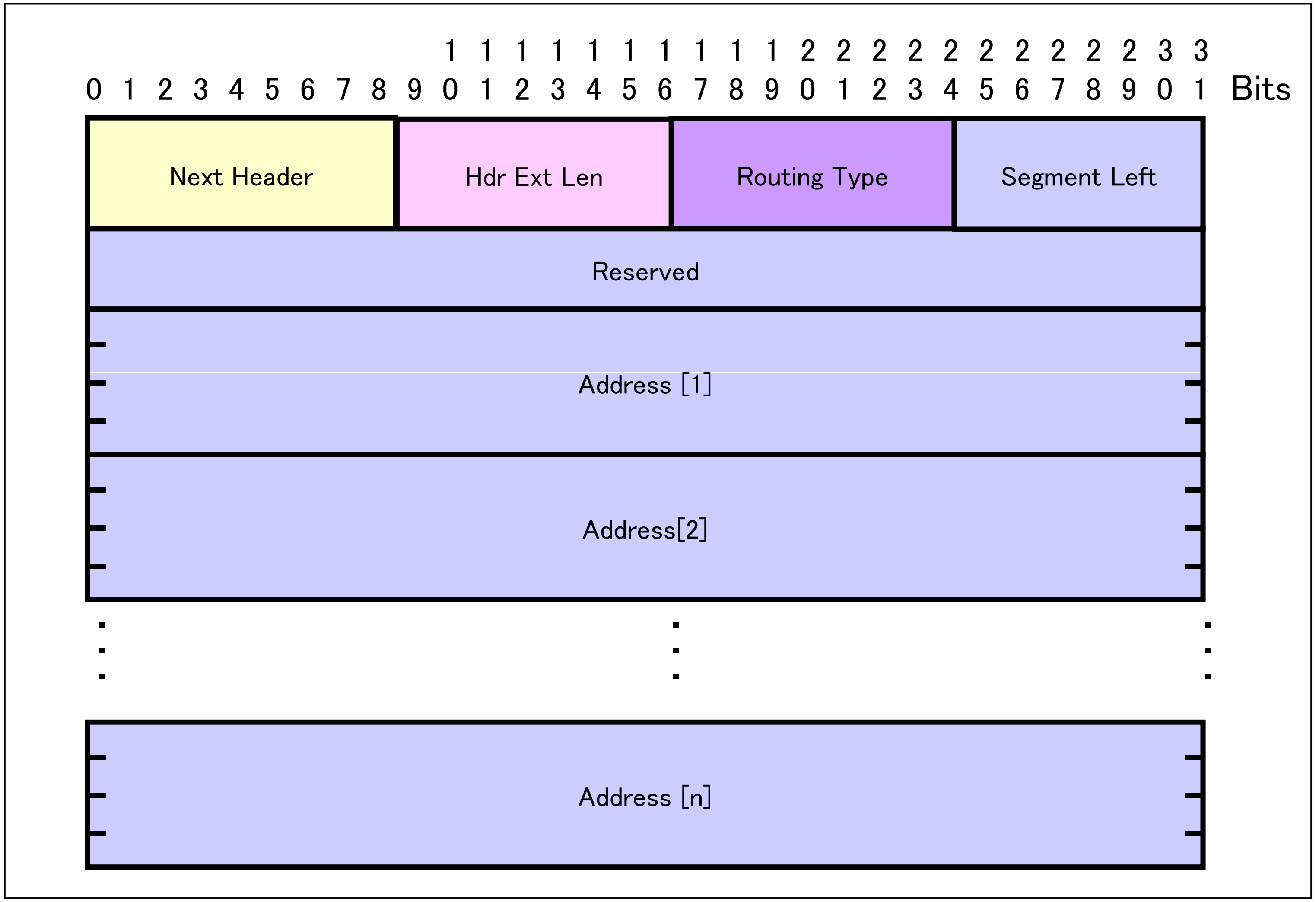
Next header

Hdr Ext Len

Option

- The options will be processed only at the destination node

Routing Header



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 Bits

Next Header	Hdr Ext Len	Routing Type	Segment Left

Reserved

Address [1]

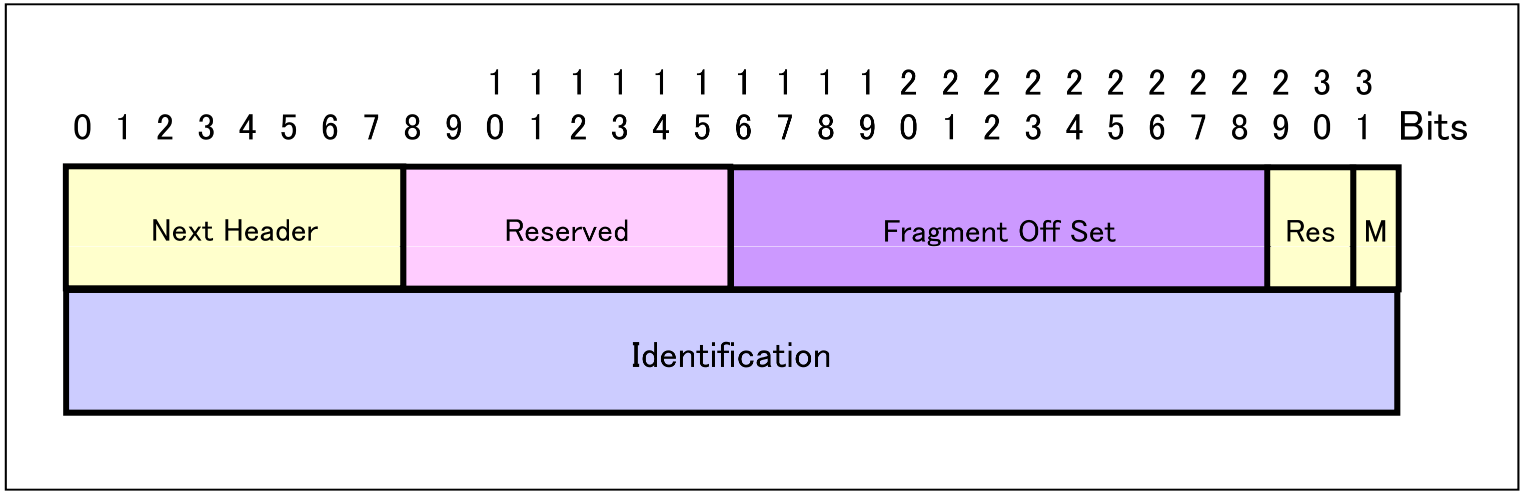
Address[2]

⋮

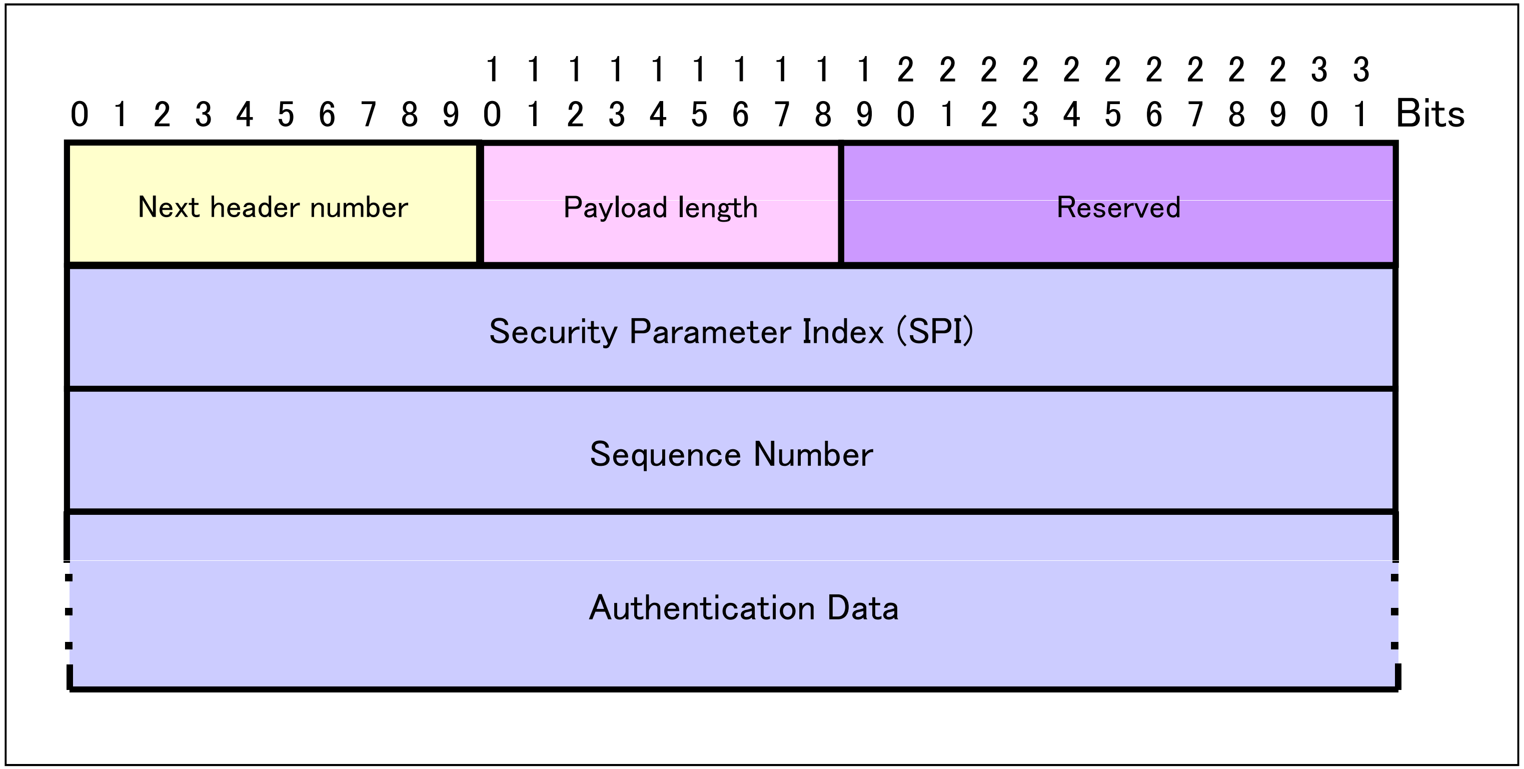
Address [n]

- A option that specifies at least one or more intermediate nodes

Fragment Header



Authentication Header



1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 Bits

Next header number

Payload length

Reserved

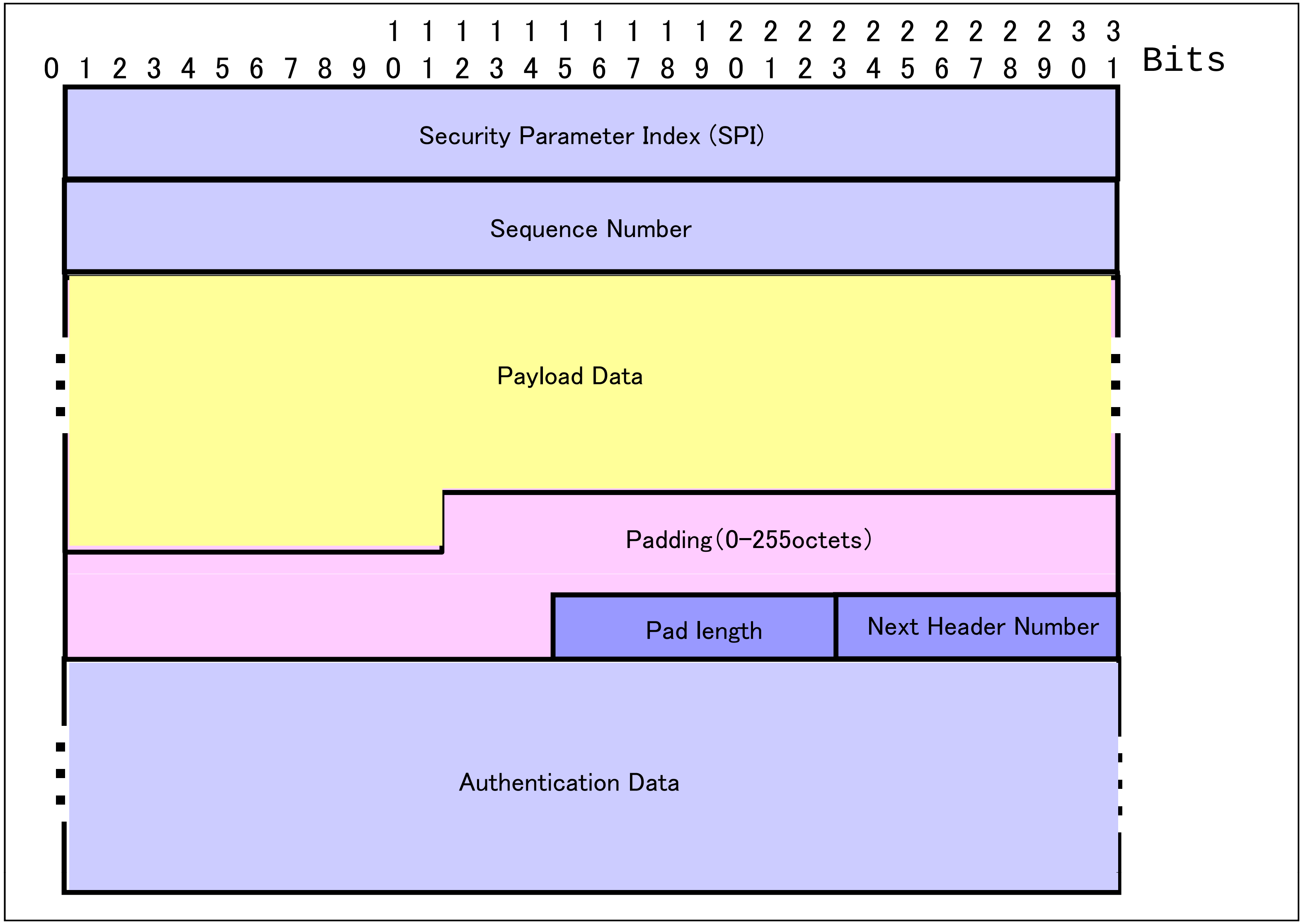
Security Parameter Index (SPI)

Sequence Number

Authentication Data

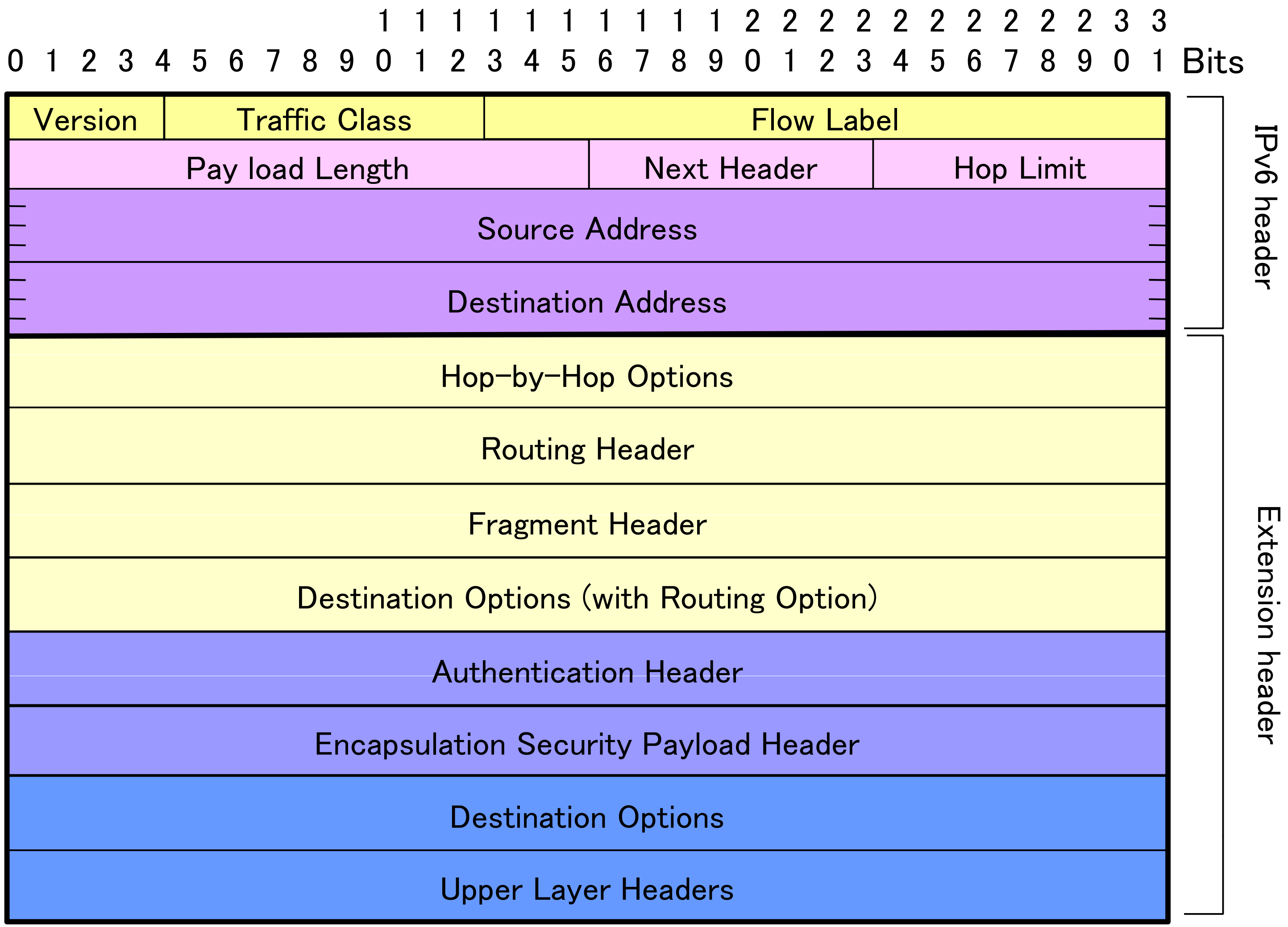
- Option to authenticate the communication counterpart (Used in IPsec)

Encapsulating Security Payload



- An option used to communicate with encrypted data (Used in IPsec)

Recommended order of extension headers



1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 Bits

Version | Traffic Class | Flow Label

Pay load Length | Next Header | Hop Limit

Source Address

Destination Address

IPv6 header

Hop-by-Hop Options

Routing Header

Fragment Header

Destination Options (with Routing Option)

Authentication Header

Encapsulation Security Payload Header

Destination Options

Upper Layer Headers

Extension header

- IPv6 main feature
- Basic knowledge for utilizing IPv6 on Linux
  - IPv6 address text representation
  - IPv6 address type
  - ULA
  - Header information
  - Dual Stack
  - DNS IPv6 support

What is Dual Stack

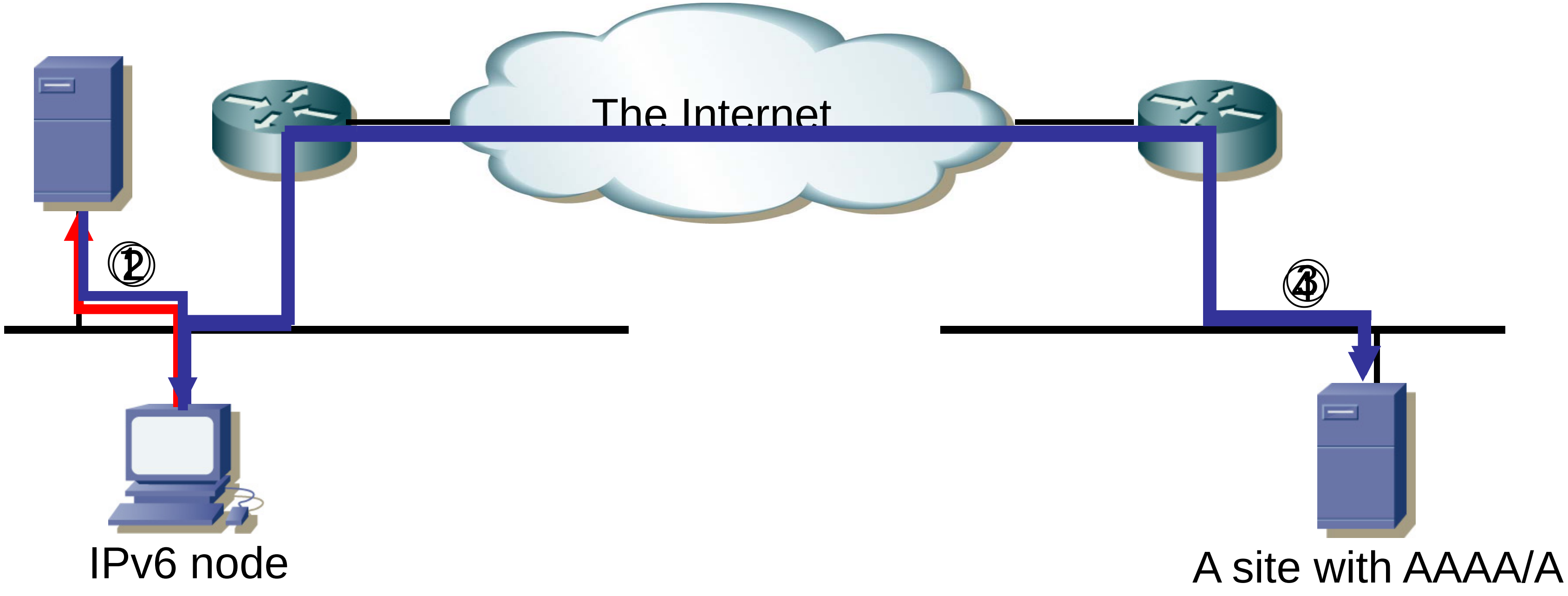
Nodes have IPv6/IPv4 addresses, and able to communicate in both addresses

```
www.example.com.      IN   A    192.0.2.1
mail.example.com.     IN   A    192.0.2.1
                      IN   AAAA 2001:db8::1
```

- What is AAAA?
  - – One of DNS records, which specifies IPv6 addresses

Establishing TCP connections

1. Inquire the addresses
2. AAAA RR or A RR will be returned
3. Connect through IPv6
4. Fallback to IPv4 when IPv6 connection cannot be established



Confirm the behavior from the program

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int sock,err;
struct addrinfo hints, *res0, *res;


memset(&hints, 0, sizeof(hints));
hints.ai_family = PF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;


/* Acquire AAAA and A record by getaddrinfo */
err = getaddrinfo("www.linux-ipv6.org", "http", &hints, &res0);


if (err) {
   fprintf(stderr, "error : %s", gai_strerror(err));
   freeaddrinfo(res0);
   exit(1);
}


/* Continue until the connection is established using the result of
      getaddrinfo*/
for (res = res0; res; res = res->ai_next) {
       sock = socket (res->ai_family, res->ai_socktype, res->ai_protocol);
       if (sock < 0)
               continue;


       if (connect(sock, res->ai_addr, res->ai_addrlen) < 0) {
               close (sock);
               continue;
       }
       break;
}
freeaddrinfo(res0);
 .
 .
 .
```

- "IPv6 support" is equal to "address family independence"

- IPv6 main feature
- Basic knowledge for utilizing IPv6 on Linux
  - IPv6 address text representation
  - IPv6 address type
  - ULA
  - Header information
  - Dual Stack
  - DNS IPv6 support

What is DNS’s IPv6 support

- RR support
  - Returning AAAA
    - Reverse Lookup is same with PTR, but utilize “ipv6.arpa” to ”in-addr.arpa”. ipv6.int was used before instead of ipv6.arpa, but not any more
  - Supported on bind8.4 or later, bind9 or later
- IPv6 transport support
  - Conduct query transmission over IPv6 protocol
  - Register AAAA to your own FQDN

* AAAA reply can be transmitted using IPv4. You have to check the IPv6 ready DNS server means if the RR is supported, the transport is supported, or both of them.

OS support on DNS query

- Revised DNS resolver (To keep the IPv4 communication quality)
  - Prioritize A record (FreeBSD, Windows Vista)
    - May become a problem when IPv6 prevail
  - Do not resolve AAAA record when NXDOMAIN is returned when resolving A record (Windows Vista)
  - Determine the AAAA record process waiting time according to the A record response time (FreeBSD, Windows Vista)
    - Minimize the timeout duration when there are no AAAA record
- Suppress AAAA record resolution
  - Do not resolve the name with AAAA query unless the host have global IPv6 address (Windows Vista)

DNS query order

- Query order differs by OS
  - FreeBSD-5.5R
    - Conduct the IPv4 and IPv6 name resolution one by one, and ends when the name is resolved
  - Windows XP SP2
    - Conduct all the IPv6 name resolution, and then move to IPv4 name resolution
  - Windows Vista
    - Conduct all the IPv4 name resolution first, and then conduct IPv6 name resolution without those with NXDOMAIN result

IPv6 configuration and its confirmation on Linux

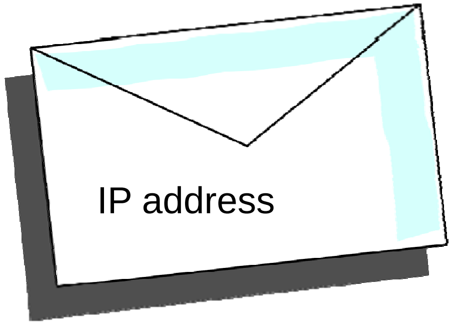
- Network configuration basics
  - Basics of routing
  - IPv4/IPv6 address configuration and its confirmation
  - IPv4/IPv6 routing confirmation
  - Bonding
  - Configuration files
  - Tunneling
- How to confirm the execution status

IPv6 configuration and its confirmation on Linux

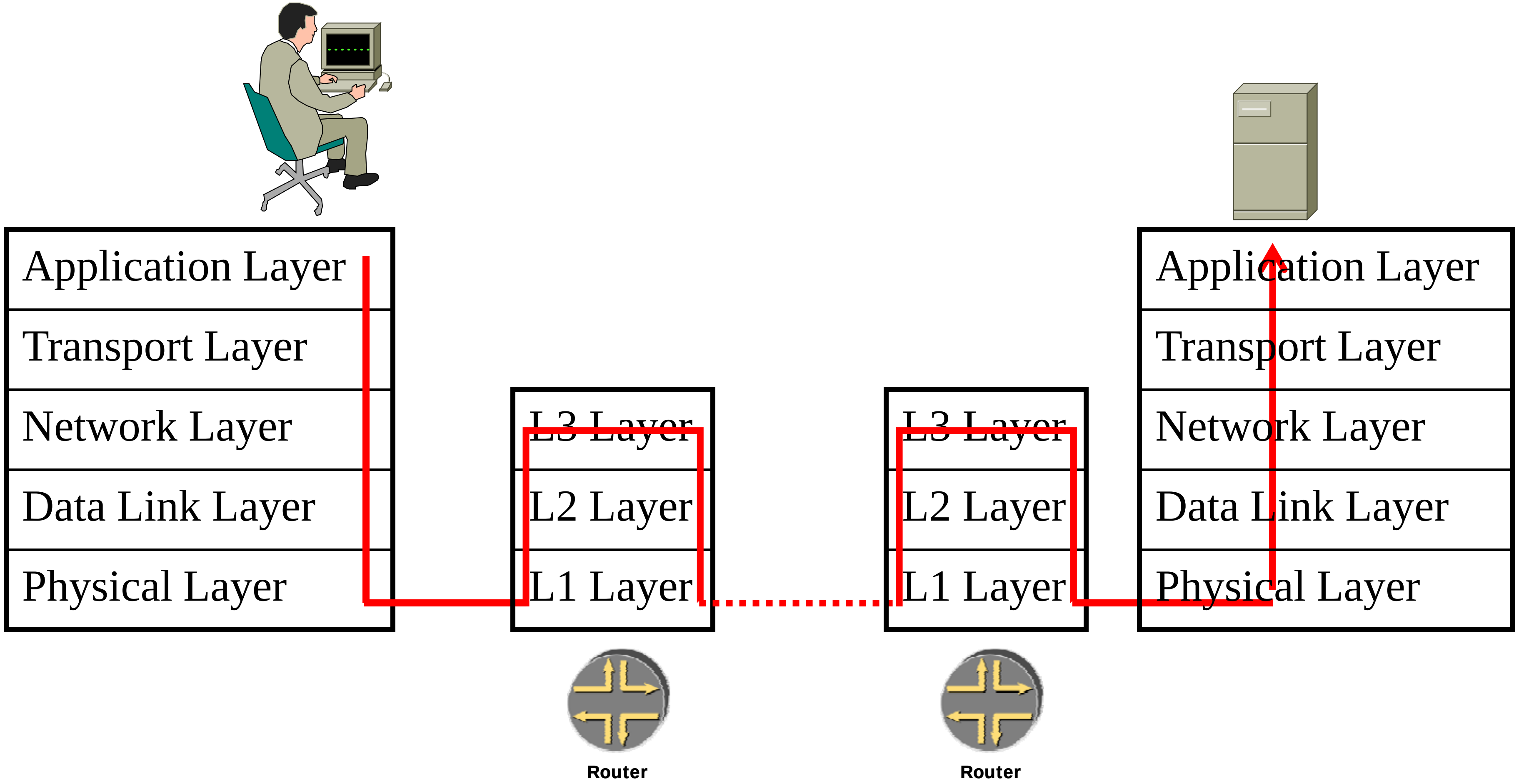
- Network configuration basics
  - – Basics of routing
  - – IPv4/IPv6 address configuration and its confirmation
  - – IPv4/IPv6 routing confirmation
  - – Bonding
  - – Configuration files
  - – Tunneling
- How to confirm the execution status

Network Configuration Basics

- In the Internet world, the data is encapsulated in a IP packet and transmitted in a bucket relay method
  - The destination name/address written on the packet is IP address



How the packet is transmitted



Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

L3 Layer
L2 Layer
L1 Layer

L3 Layer
L2 Layer
L1 Layer

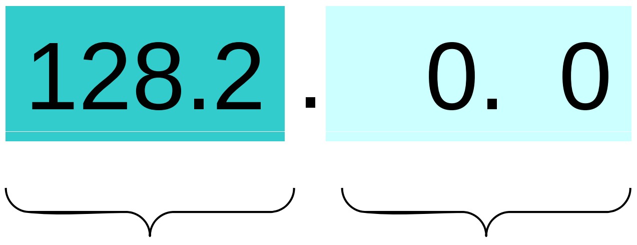
Application Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

Class full world

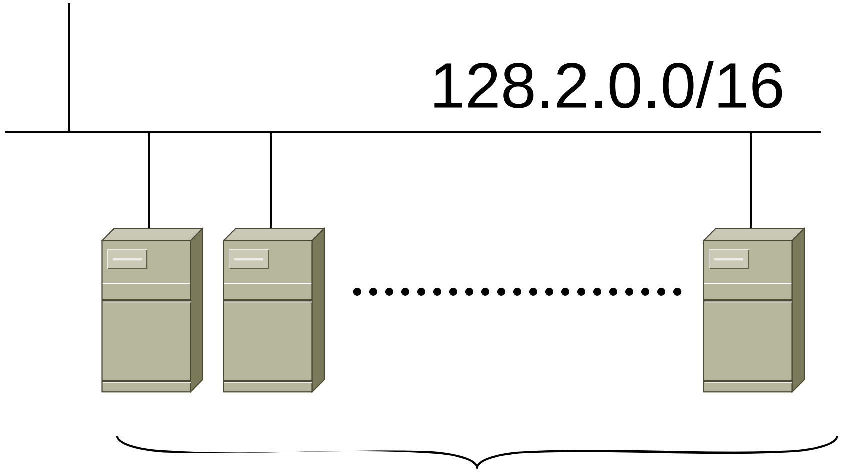
The IP addresses were separated to classes in the original scheme

	IP Address Range	netmask	length
Class A	1.0.0.0-126.255.255.255	255.0.0.0	/8
Class B	128.0.0.0-191.255.255.255.255	255.255.0.0	/16
Class C	192.0.0.0-223.255.255.255	255.255.255.0	/24

Example: Class B



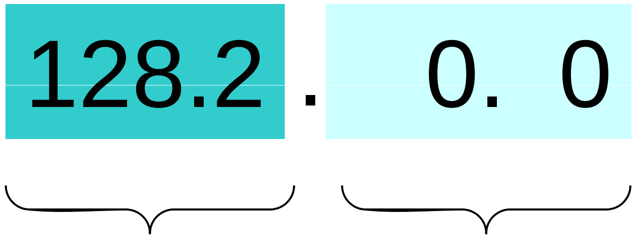
Network Address Host Address



2^16-2 = 65534 addresses

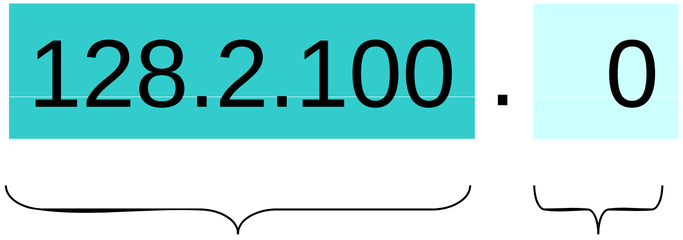
Emergence of subnets

Example: Class B



Network Address Host Address

Example: Extend the network part using subnet scheme



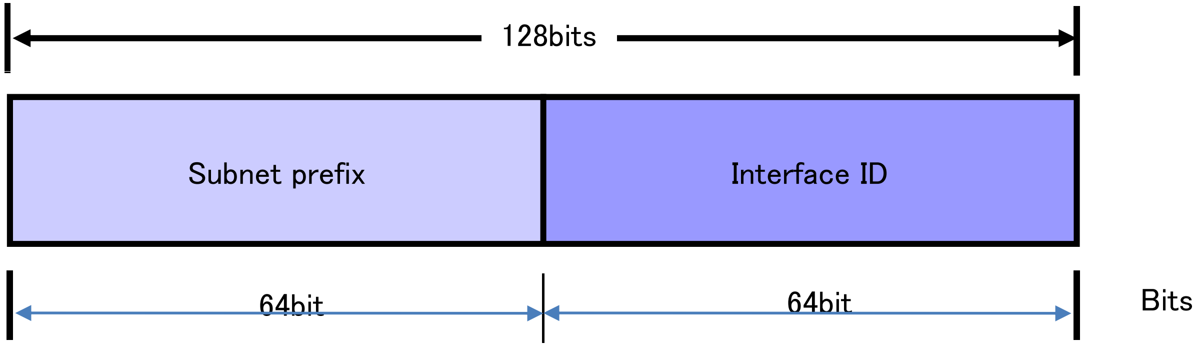
Network Address Host Address

2^8-2 = 254 host address can be utilized

length	Sub netmask	Hosts
/16	255.255.0.0	65534
/17	255.255.128.0	32766
/18	255.255.192.0	16382
/19	255.255.224.0	8190
/20	255.255.240.0	4094
/21	255.255.248.0	2046
/22	255.255.252.0	1022
/23	255.255.254.0	510
/24	255.255.255.0	254
/25	255.255.255.128	126
/26	255.255.255.192	62
/27	255.255.255.224	30
/28	255.255.255.240	14
/29	255.255.255.248	6
/30	255.255.255.252	2

In IPv6?

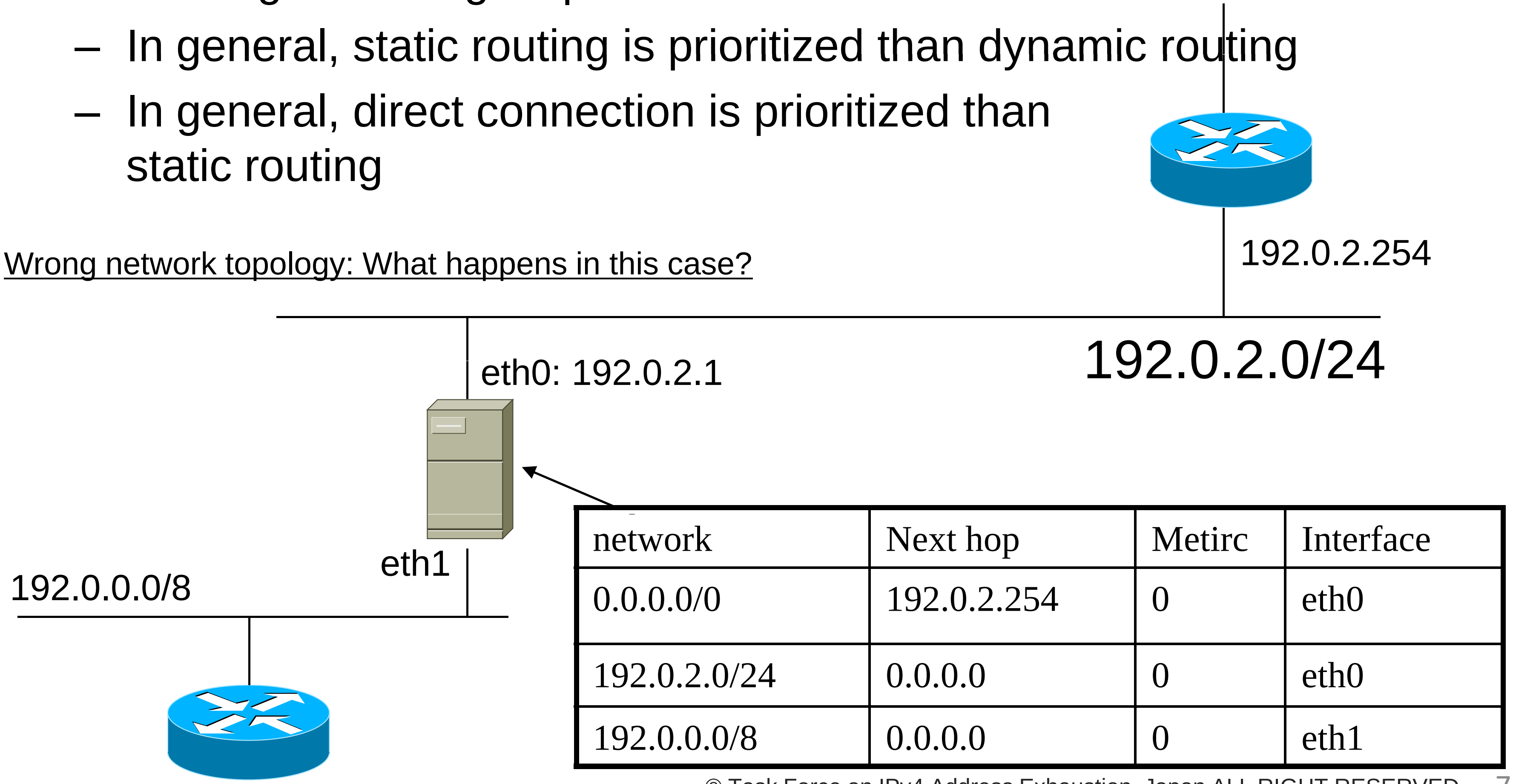
- It of course utilizes classless
- You can consider the netmask to be /64 fixed as long as you utilize it in servers



Routing Basics

- Static routing
  - Routing uses longest prefix match
  - In general, static routing is prioritized than dynamic routing
  - In general, direct connection is prioritized than static routing

Wrong network topology: What happens in this case?



network	Next hop	Metirc	Interface
0.0.0.0/0	192.0.2.254	0	eth0
192.0.2.0/24	0.0.0.0	0	eth0
192.0.0.0/8	0.0.0.0	0	eth1

- Network configuration basics
  - Basics of routing
  - IPv4/IPv6 address configuration and its confirmation
  - IPv4/IPv6 routing confirmation
  - Bonding
  - Configuration files
  - Tunneling
- How to confirm the execution status

Configuring IP address

- Following commands are commonly used to configure IP address
  – ifconfig
    - format： ifconfig *interface [aftype] options | address …*
  – ip
    - format：ip *[OPTIONS] OBJECT { COMMAND | help }*

Configuring IPv4 address - ifconfig -

- Example of IPv4 address configuration

```
IPv4 address configuration
# ifconfig eth0 192.0.2.1 netmask 255.255.255.0

Confirm the configuration
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:00:XX:XX:XX:XX
          inet addr:192.0.2.1  Bcast:192.0.2.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:169

Activate the interface
# ifconfig eth0 up

Inactivate the interface
# ifconfig eth0 down
```

IPv6 address configuration - ifconfig-

- Example of IPv6 address configuration

```
IPv6 address configuration (You have to set the interface to “up” before the configuration)
# ifconfig eth0 add 2001:db8::80/64

Deleting IPv6 address
# ifconfig eth0 del 2001:db8::80/64

Confirm the configuration
# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:XX:XX:XX:XX:XX
          inet addr:192.0.2.1  Bcast:192.0.2.255  Mask:255.255.255.0
          inet6 addr: 2001:db8::80/64 Scope:Global
          inet6 addr: fe80::2d0:xxxx:xxxx:xxxx/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:169

Shutting down the interface (Global address will be deleted with shut down in case of IPv6)
# ifconfig eth0 down
```

IPv4/IPv6 address configuration – ip command –

```
Configuring IPv4/IPv6 addresses
# ip addr add 192.0.2.1/24 dev eth0
# ip addr add 2001:db8::80/64 dev eth0

Confirm the configuration
# ip addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global eth0
    inet6 fe80::202:xxx:xxx:xxx:xxx/64 scope link
       valid_lft forever preferred_lft forever
    inet6 2001:db8::80/64 scope global
       valid_lft forever preferred_lft forever

Activate the interface
# ip link set eth0 up

Inactivate the interface
# ip link set eth0 down
```

IPv4/IPv6 address configuration / lifetime

```
# ip addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.1/24 brd 192.0.2.255 scope global eth0
    inet6 fe80::202:xxx:xxx:xxx:xxx/64 scope link
       valid_lft forever preferred_lft forever
    inet6 2001:db8::80/64 scope global
       valid_lft forever preferred_lft forever
```

Confirm the IPv4/IPv6 L2 address

```
In case of IPv4: Instead of ARP command
# ip -4 neigh show
192.0.2.1 dev eth0 lladdr 00:00:XX:00:XX:XX REACHABLE

In case of IPv6:
# ip -6 neigh show
2001:db8:0:1::dead:beaf dev eth0 lladdr 00:00:XX:XX:XX:XX router REACHABLE
fe80::XXX:XXXX:XXXX:XXXX dev eth0 lladdr 00:00:XX:XX:XX:XX router REACHABLE
```

Routing configuration and confirmation utilizing route command

```
Set the default route
# route add –A inet default gw 192.0.2.254 dev eth0
# route add –A inet6 default gw fe80:x:x:x:x:x dev eth0

Set the route for each network
# route add –net 192.0.2.0 netmask 255.255.255.0 gw 192.168.0.1 dev eth0
# route add –A inet6 2001:db8::/64 gw fe80::2d0:b7ff:fea0:beea dev eth0

Confirm the routing
# route -n –A inet6
# route –n –A inet
```

Configuration and confirmation of routing with ip command

```
Set the Default route
# ip route add default via 10.0.0.1 dev eth0
# ip route add default via fe80::202:b3ff:fe32:faa2 dev eth0

Set the route for each network
# ip route add 192.0.2.0/24 via 10.0.0.1 dev eth0
# ip route add 2001:db8::/48 via fe80::202:b3ff:fe32:faa2 dev eth0

Confirm the routing
# ip -4 route show
# ip -6 route show
```

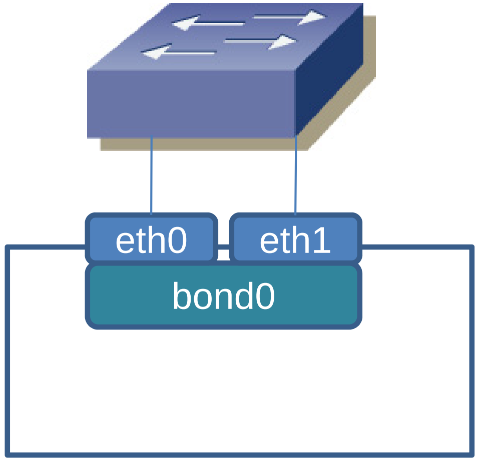
- Network configuration basics
  - – Basics of routing
  - – IPv4/IPv6 address configuration and its confirmation
  - – IPv4/IPv6 routing confirmation
  - – Bonding
  - – Configuration files
  - – Tunneling
- How to confirm the execution status

bonding

- Utilized to establish redundant networks
  - No problem for utilization in IPv6
- Example of redundancy using active-backup

/etc/modprobe.d/bonding

```
alias bond0 bondingoptions
bond0 miimon=100 mode=1
```



Confirmation

```
$ cat /proc/net/bonding/bond0
```

- Network configuration basics
  - Basics of routing
  - IPv4/IPv6 address configuration and its confirmation
  - IPv4/IPv6 routing confirmation
  - Bonding
  - <u>Configuration files</u>
  - Tunneling
- How to confirm the execution status

About configuration files (in general)

Files for name resolution

/etc

- hosts – Matching list of IP address and host names
- resolv.conf – Resolver configuration file
- host.conf – Resolver configuration file

```
127.0.0.1       hoge.example.jp   localhost hoge

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

```
search example.jp
nameserver 192.0.2.254
```

```
order hosts,bind
multi on
```

About configuration files (in general)

Files used for resolving service names, etc

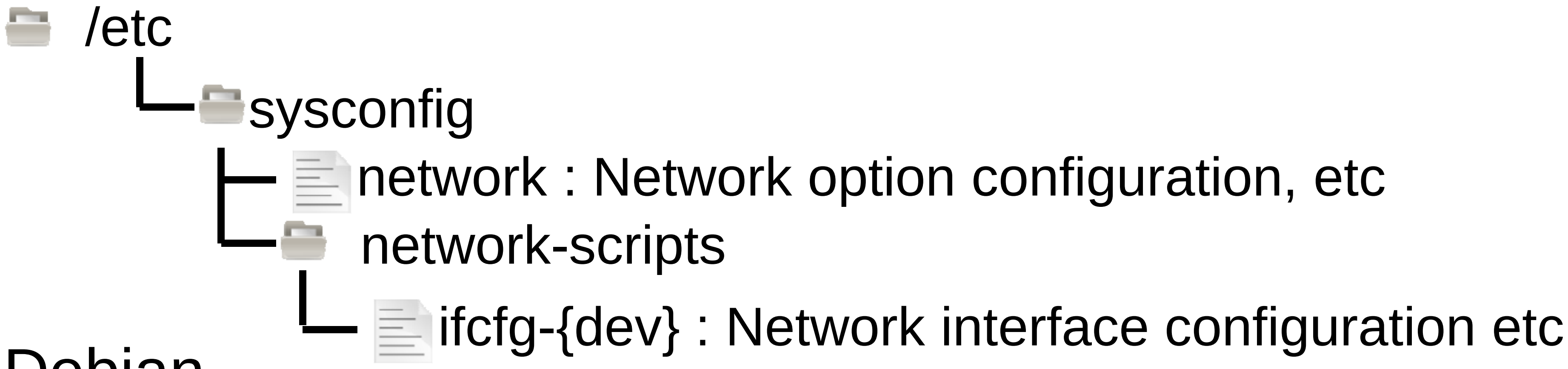
/etc
- services : Network service list
- protocols : Protocol definition file

```
tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp
msp
chargen
chargen
ftp-data
ftp
fsp
ssh
ssh
- Omitted -
```

```
ip      0       IP              # internet protocol, pseudo protocol number
#hopopt 0       HOPOPT          # IPv6 Hop-by-Hop Option [RFC1883]
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ``IP'')
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
egp     8       EGP             # exterior gateway protocol
```

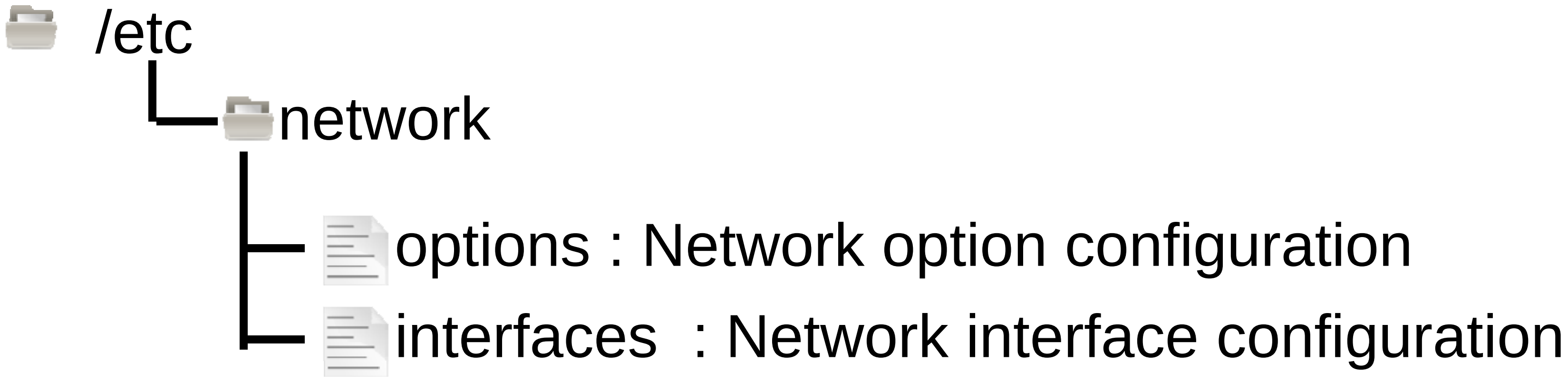
About configuration files

CentOS



- /etc
  - sysconfig
    - network : Network option configuration, etc
    - network-scripts
      - ifcfg-{dev} : Network interface configuration etc

Debian



- /etc
  - network
    - options : Network option configuration
    - interfaces : Network interface configuration

* Recent Debian/Ubuntu will not reflect the configuration when network-manager package is installed.

Network configuration on CentOS

/etc/sysconfig/network

```
HOSTNAME=hoge.example.jp

# IPv4
NETWORKING=yes
GATEWAY=192.0.2.254

#IPv6
NETWORKING_IPV6=yes
IPV6_DEFAULTGW=fe80::xxxx%eth0
#IPV6_AUTOCONF=no
#IPV6FORWARDING=no
```

HOSTNAME
  Write FQDN

NETWORKING
  yes: Run rc script network

GATEWAY
  Specify the IPv4 network default gateway

NETWORKING_IPV6
  yes: Enable IPv6 network

IPV6_DEFAULTGW
  Specify the IPv6 network default gateway

Network configuration on CentOS

/etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETWORK=192.0.2.0
NETMASK=255.255.255.0
IPADDR=192.0.2.1
IPV6INIT=yes
#IPV6_AUTOCONF=no
IPV6ADDR=2001:db8::80/64
ETHTOOL_OPTS="autoneg off speed 100 duplex full"
```

DEVICE
Physical interface name

BOOTPROTO
none : unused
bootp: utilize BOOTP
dhcp : utilize DHCP

ONBOOT
yes : Activate on boot
no : Not activate on boot

IPADDR
IPv4 address without netmask

NETWORK
Specify network address for IPv4

NETMASK
Specify netmask for IPv4

IPV6INIT
yes: Enable IPv6 initialization
No : Do not enable IPv6 initialization

IPV6ADDR
IPv6 address / netmask

ETHTOOL_OPTS
Specify options to ethtool

Network configuration on Debian (obsolete)

/etc/network/options

```
ip_forward=no
spoofprotect=yes
syncookies=no
```

ip_forward

yes: Configure when utilizing Linux as a Router

spoofprotect

yes: Reverse Path is verified. Configuration recommended when connected to Stub network

syncookies

yes: Issue cookies to deal against TCP syn flooding attack

/etc/network/options became obsolete, and replaced to /etc/sysctl.conf

/etc/sysctl.conf

```
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.tcp_syncookies = 1
```

/etc/network/interface

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.0.2.1
        netmask 255.255.255.0
        gateway 192.0.2.254
        # dns-serarch example.jp
        # dns-nameservers 192.0.2.254
        up ethtool -s eth0 autoneg off speed 100 duplex full

iface eth0 inet6 static
        address 2001:db8::80
        netmask 64
        gateway fe80::xxx%eth0
```

pre-up
Run command before bringing the interface up

up
Command executed when interface is up

post-up
Run command after bringing the interface up

pre-down
Run command before taking the interface down

down
Command executed when interface is down

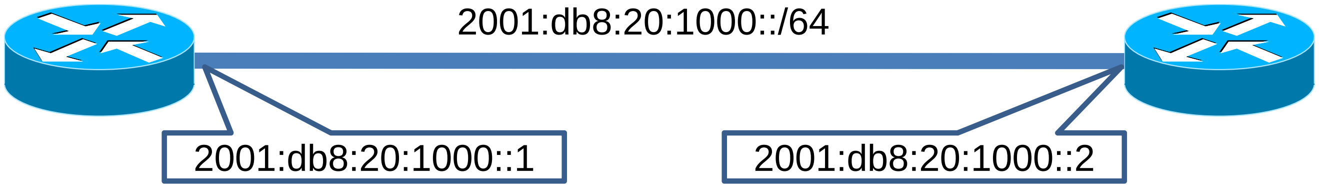
post-down
Run command after taking the interface down

Also See man interfaces

- Network configuration basics
  - Basics of routing
  - IPv4/IPv6 address configuration and its confirmation
  - IPv4/IPv6 routing confirmation
  - Bonding
  - Configuration files
  - Tunneling
- How to confirm the execution status

Tunneling

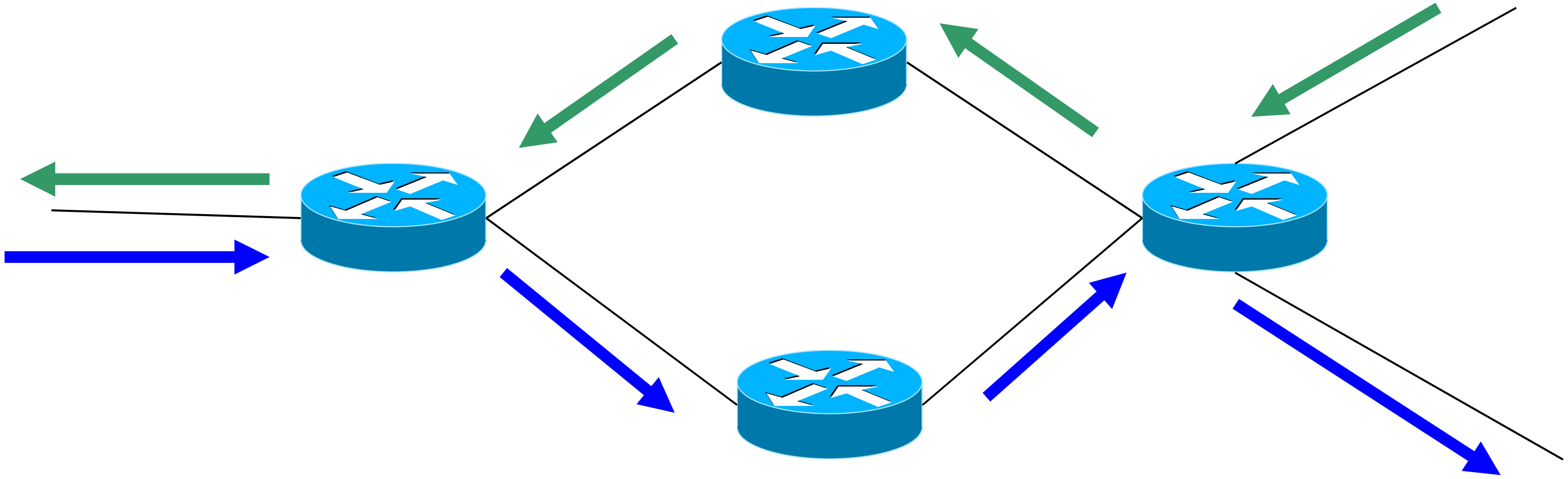
- Tunnel is point-to-point connection
  - When the address destination is not the device, it will consider it is for the tunnel counterpart
  - It is normal to assign /64 to networks
    - When a packet with destination address 2001:db8:20:1000::3 is transmitted…



- Network configuration basics
  - Basics of routing
  - IPv4/IPv6 address configuration and its confirmation
  - IPv4/IPv6 routing confirmation
  - Bonding
  - Configuration files
  - Tunneling
- How to confirm the execution status

Connection confirmation

- Easy connection confirmation method
  - Connection confirmation using ping
  - Connection confirmation using traceroute
    - You can only see the outbound route by traceroute



Outbound and inbound route may differ…

Connection confirmation using Ping(ICMP)

- Ping to the default gateway

```
Connection confirmation of IPv4 route
$ ping -c 10 192.0. 2.254
PING 192.0.2.254 (192.0.2.254) 56(84) bytes of data.
64 bytes from 192.0.2.254: icmp_seq=1 ttl=255 time=2.09 ms
64 bytes from 192.0.2.254: icmp_seq=2 ttl=255 time=2.04 ms
64 bytes from 192.0.2.254: icmp_seq=3 ttl=255 time=4.30 ms
64 bytes from 192.0.2.254: icmp_seq=4 ttl=255 time=2.00 ms
64 bytes from 192.0.2.254: icmp_seq=5 ttl=255 time=2.01 ms
64 bytes from 192.0.2.254: icmp_seq=6 ttl=255 time=2.02 ms
64 bytes from 192.0.2.254: icmp_seq=7 ttl=255 time=2.03 ms
64 bytes from 192.0.2.254: icmp_seq=8 ttl=255 time=2.62 ms
64 bytes from 192.0.2.254: icmp_seq=9 ttl=255 time=3.84 ms
64 bytes from 192.0.2.254: icmp_seq=10 ttl=255 time=4.77 ms

--- 192.0.2.254 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9036ms
rtt min/avg/max/mdev = 2.009/2.776/4.774/1.038 ms
```

Connection confirmation using Ping(ICMP)

- Ping to the default gateway

```
Connection confirmation of IPv6 route
$ ping6 -c 10 fe80::2000:1 -I eth0
PING fe80::2000:1(fe80::2000:1) from fe80::2d0:b7ff:fea0:beea
eth1: 56 data bytes
64 bytes from fe80::2000:1: icmp_seq=1 ttl=64 time=2.38 ms
64 bytes from fe80::2000:1: icmp_seq=2 ttl=64 time=7.71 ms
64 bytes from fe80::2000:1: icmp_seq=3 ttl=64 time=7.47 ms
64 bytes from fe80::2000:1: icmp_seq=4 ttl=64 time=2.41 ms
64 bytes from fe80::2000:1: icmp_seq=5 ttl=64 time=2.39 ms
64 bytes from fe80::2000:1: icmp_seq=6 ttl=64 time=3.91 ms
64 bytes from fe80::2000:1: icmp_seq=7 ttl=64 time=5.00 ms
64 bytes from fe80::2000:1: icmp_seq=8 ttl=64 time=2.29 ms
64 bytes from fe80::2000:1: icmp_seq=9 ttl=64 time=2.30 ms
64 bytes from fe80::2000:1: icmp_seq=10 ttl=64 time=2.32 ms

--- fe80::2000:1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9036ms
rtt min/avg/max/mdev = 2.292/3.822/7.711/2.069 ms
```

Hasn’t got the speed?

- Confirm the Ethernet Card is configured as Full Duplex

```
Confirm the current configuration
# ethtool eth0

Configure to Full-duplex
# ethtool -s eth0 autoneg off speed 100 duplex full
```

```
Confirm the current configuration
# mii-tool eth0

Configure to Full-duplex
# mii-tool -F 100baseTx-FD eth1
```

Control with sysctl

- IPv6 auto-configuration will be executed when the interface is brought to up. Disable the unnecessary configurations beforehand

```
# sysctl –w net.ipv6.conf.eth0.accept_ra=0
```

- Dealing against TCP Syn flooding attack

```
# sysctl –w net.ipv4.tcp_syncookies=1
```

- Ignore ICMP destined to Broadcast

```
# sysctl –w net.ipv4.icmp_echo_ignore_broadcasts
```

Confirm the execution status

Server status check

- Which programs are running?
  - ps command

```
# ps aux | less
```

- Listening to which ports?
  - netstat
  - fuser

Server status check (netstat)

- Utilize netstat to display server’s listen port

Major options

-l, --listening
  Show only listening sockets
-p, --program
 Show the PID and name of the program to which each socket belongs
-n, --numeric
 Show numerical addresses instead of trying to determine symbolic
 host, port or user names.
-t,--tcp
  Show host TCP statistics
-u,--udp
  Show host UDP statistics

Server status check (netstat)

Execution example

```
# netstat -ltupn
Proto Recv-Q Send-Q Local Address        Foreign Address     State       PID/Program name
tcp        0      0 127.0.0.1:993        0.0.0.0:*           LISTEN     3785/famd
tcp        0      0 127.0.0.1:111        0.0.0.0:*           LISTEN     3175/portmap
tcp        0      0 192.0.2.1:53         0.0.0.0:*           LISTEN     3380/named
tcp        0      0 127.0.0.1:53         0.0.0.0:*           LISTEN     3380/named
tcp        0      0 0.0.0.0:5432         0.0.0.0:*           LISTEN     3619/postmaster
tcp        0      0 0.0.0.0:25           0.0.0.0:*           LISTEN     3596/master
tcp        0      0 127.0.0.1:953        0.0.0.0:*           LISTEN     3380/named
tcp6       0      0 :::80                :::*                LISTEN     11254/apache2
tcp6       0      0 :::53                :::*                LISTEN     3380/named
tcp6       0      0 :::22                :::*                LISTEN     3659/sshd
tcp6       0      0 :::5432              :::*                LISTEN     3619/postmaster
udp        0      0 0.0.0.0:32768        0.0.0.0:*                      3380/named
udp        0      0 127.0.0.1:161        0.0.0.0:*                      3653/snmpd
udp        0      0 192.0.2.1:53         0.0.0.0:*                      3380/named
udp        0      0 127.0.0.1:53         0.0.0.0:*                      3380/named
udp        0      0 127.0.0.1:111        0.0.0.0:*                      3175/portmap
udp6       0      0 :::32769             :::*                           3380/named
udp6       0      0 :::53                :::*                           3380/named
```

* You can specify address family with ”-A” option (inet or inet6)

Notes in netstat

- Due to netstat character display limitation, IPv6 addresses longer than certain length will not be displayed
- net-tool_1.60-22 or later in Debian package can display all the information with ”-W/--wide” options

Server status check with ss

- Options are almost same with netstat, but IPv6 addresses will not be abbreviated

```
$ ss -ltun
Netid  Recv-Q Send-Q          Local Address:Port            Peer Address:Port
tcp    0      50                  127.0.0.1:3306                       *:*
tcp    0      128                         *:111                        *:*
tcp    0      128                        :::80                        :::*
tcp    0      5                   127.0.0.1:33843                      *:*
tcp    0      3                192.0.2.136:53                         *:*
tcp    0      3                   127.0.0.1:53                         *:*
tcp    0      3                          :::53                        :::*
tcp    0      128                 127.0.0.1:631                        *:*
tcp    0      128                 127.0.0.1:5432                       *:*
tcp    0      128                       ::1:953                       :::*
tcp    0      128                 127.0.0.1:953                        *:*
tcp    0      100                        :::25                        :::*
tcp    0      100                         *:25                         *:*
tcp    0      3                           *:1723                       *:*
```

Server status check (fuser)

- Specify the listening process through server listening port using fuser

```
# fuser -vn tcp 80

                     USER        PID ACCESS COMMAND
80/tcp:              root       4699 F.... apache
                     www-data   4706 F.... apache
                     www-data   4707 F.... apache
                     www-data   4708 F.... apache
                     www-data   4709 F.... apache
                     www-data   4710 F.... apache
                     www-data   8407 F.... apache
                     www-data   8408 F.... apache
                     www-data   8409 F.... apache
```

Basic service configurations

- DNS (bind9)
- SMTP (postfix)
- POP server (dovecot)
- Apache
- NTP

- **DNS(bind9)**
- SMTP(postfix)
- POP server (dovecot)
- Apache
- NTP

DNS

- To enable BIND9 IPv6 transport, configure the listen-on-v6

```
options {
            directory "/var/named";
            listen-on-v6 { any; };
…
..
.
```

DNS

- ACL configuration example
  - You can input specific addresses like IPv4

```
acl “slaves” {
        192.0.2.1;      // slave server
        2001:db8::53;   // slave server
        127.0.0.1;      // for debug
        ::1;            // for debug
};
```

DNS

- AAAA RR registration example

```
;; Server
;;   example.jp
www    IN      A      192.0.2.1
www    IN      AAAA   2001:db8::1
```

- AAAA RR confirmation

```
$ dig www.example.jp AAAA
```

DNS

- IPv6 reverse lookup configuration

```
;; IPv6 PTR for 2001:db8::/64
zone “0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ipv6.arpa” {
        type master;
        file “2001.0db8.0000.000.reverse”;
        allow-transfer { slaves; };
        allow-query { any };
}
```

```
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0   IN   PTR   www.example.jp.
```

- IPv6 reverse lookup configuration confirmation

```
$ dig –x 2001:db8::1
```

DNS: Configuration at resolver

- Configure IPv6 address to /etc/resolv.conf, same as IPv4

```
search example.jp
nameserver 192.0.2.254
nameserver 2001:db8::53
```

- DNS(bind9)
- **SMTP(postfix)**
- POP server (dovecot)
- Apache
- NTP

Enable IPv6 in postfix

- /etc/postfix/main.cf

```
# inet_protocols = ipv4
# inet_protocols = ipv4, ipv6 # equivalent to all
# inet_protocols = ipv6
inet_protocols = all
```

That’s all…

Limit the addresses to Listen

- /etc/postfix/main.cf

```
# inet_interfaces = alll
# inet_interfaces = loopback-only
inet_interfaces = 127.0.0.1, [::1], [2001:db8::25]
```

Fix the sending address

- /etc/postfix/main.cf *

```
smtp_bind_address6 = 2001:db8::25
```

* You can also configure at master.cf

Characters that have to be enclosed in []

- When configuring postfix match list such as mynetworks and debug_peer_list, you have to enclose the IPv6 addresses with [] not to mix with “type:table” format.

```
# mynetworks = hash:/etc/postfix/network_table
mynetworks = 127.0.0.0/8 [::1]/128
```

- DNS(bind9)
- SMTP(postfix)
- **POP server (dovecot)**
- Apache
- NTP

Enabling IPv6 in Dovecot

- No need for special configuration. Too much complex configuration related to listen address specification cannot be done in CentOS5 group Dovecot.
- /etc/dovecot.conf

```
# Listen to all interfaces IPv4 address when
# specifying “*”
#Listen = *
#
# Listen to all interfaces IPv6 address when
# specifying “[::]”.  Some OS listen to all
# interface IPv4 with this configuration.
#Listen = [::]
```

- DNS(bind9)
- SMTP(postfix)
- POP server（dovecot)
- **Apache**
- NTP

Apache configuration

- There are no special configurations
  - – Supports IPv6 from Apache2.0
  - – Will mention about the IPv6 dependencies from here after

Listen

Addresses have to be enclosed with [] unlike IPv4

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

Access control using ACL / address

- Note that you cannot configure as 2001:db8:0:1000/64. You have to specify the network address.

```
AuthName “Staff Only”
AuthType Basic
AuthUserFile “/var/www/www.example.jp/.htpasswd”
Require valid-user
Order Deny,Allow
Deny from all
Allow from 192.168.1.1
Allow from 2001:db8:0:1000::/64
Satisfy Any
```

Address base VirtualHost

You have to enclose the addresses with [] unlike IPv4, same as Listen

```
#<VirtualHost *:80>
<VirtualHost [2001:db8:0:1000::80]:80>
   ServerName www.example.co.jp
  …
  ..
  .
</VirtualHost>
```

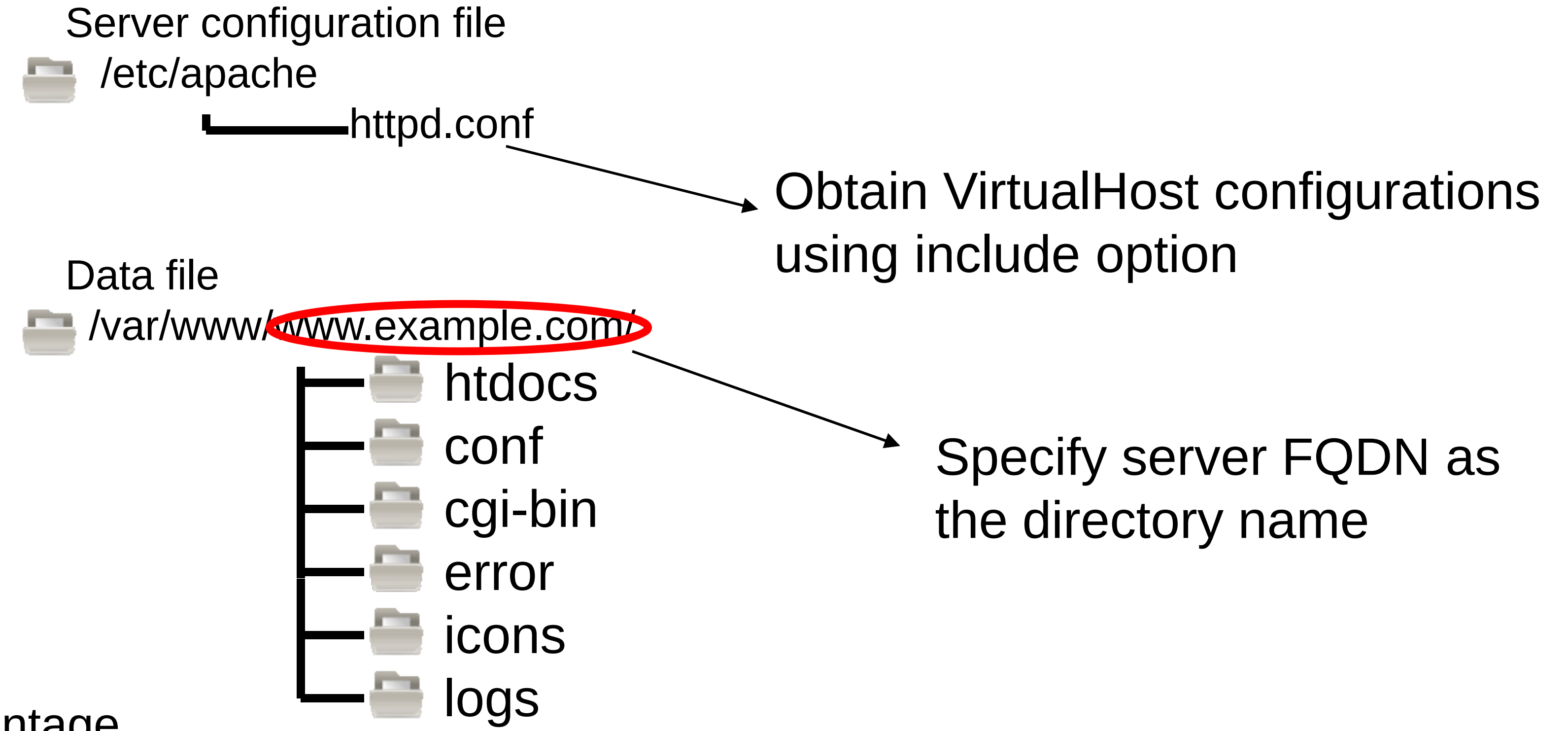
Access log

Below is the common log output example

```
2001:db8:0:1000:211:24ff:dead:beaf - - [04/Nov/2008:09:30:59 +0900] "GET /favicon.ico HTTP/1.1" 404 272
192.0.2.1 - - [04/Nov/2008:09:35:59 +0900] "GET /favicon.ico HTTP/1.1" 404 272
```

* When running Apache on Linux, IPv6 socket processes IPv4 connections, but the IPv4 address on the log is described 192.0.2.1 and not ::ffff:192.0.2.1

Configuration files layout example



Server configuration file

/etc/apache

httpd.conf

Obtain VirtualHost configurations using include option

Data file

/var/www/www.example.com/

htdocs

conf

cgi-bin

error

icons

logs

Specify server FQDN as the directory name

- Advantage

  Even when the number of operating VirtualHost changes, directory layout does not change

  Full backup of each VirtualHost can be done by specifying the directory

- Disadvantage

  As this is not the standard configuration, you have to double check such as when using the logrorate

- DNS (bind9)
- SMTP (postfix)
- POP Server (dovecot)
- Apache
- **NTP**

NTP

- Specify upper level NTP server
  - You can specify using FQDN and IPv6 addresses

```
server    ntp1.v6.mfeed.ad.jp
server    2001:3a0:0:2005::57:123
```

NTP

- Limit the query
  - With restrict command, you can utilize IPv4/IPv6 addresses, but you are supposed to specify the addresses with ”-4”, “-6” options

```
#restrict -4 default kod notrap nomodify nopeer noquery
#restrict -6 default kod notrap nomodify nopeer noquery
#restrict 127.0.0.1
#restrict ::1
restrict -4 192.0.2.0 mask 255.255.255.0 knod notrap nomodify nopeer noquery
restrict -6 2001:db8:: mask ffff:ffff:ffff:ffff:: knod notrap nomodify nopeer
#restrict 192.168.123.0 mask 255.255.255.0 notrust
```

NTP

- Synchronization check can be done using ntpq command same as IPv4. (But the information will be abbreviated)

```
$ ntp -pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 2001:3a0:0:2001 210.173.160.86   2 u   57   64    1    3.195    9.845   0.002
 2001:3a0:0:2005 210.173.160.56   2 u   56   64    1    3.173    9.871   0.002
```

```
$ ntp –pn
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 2001:3a0:0:2001 210.173.160.56   2 u   29   64   77    3.130    5.788   2.341
*2001:3a0:0:2005 210.173.160.86   2 u   33   64   77    3.173    9.871   4.693
```

- **Measuring IPv6 traffics (MRTG)**
- IPv6 ready tools
  - Smokeping
  - Nagios

Measuring the traffic using MRTG

- If the following Perl module is already installed, MRTG can acquire the data using IPv6 (cfgmaker is also IPv6 ready)
  - Socket6
  - IO::Socket::INET6
- When measuring the data in IPv6, do not forget the following configuration

```
EnableIPv6: yes
```

Traffic monitoring of IPv6

- Normal MIB cannot acquire IPv6 only traffic, so you have to work on it
- You can acquire IPv6 only traffic by measuring the packets using ip6tables

MRTG configuration

- You can create graphs using data from MRTG with external scripts not just SNMP

```
Target[Linux]: `/usr/local/bin/v6counter.pl`
```

Using ip6tables

```
#!/usr/bin/perl
# for IPv6 traffic counting
#
use strict;
use warnings;

##############################################
# Edit following values.
##############################################
my $ip6tables_cmd = '/sbin/ip6tables';
my $head_cmd      = '/usr/bin/head';

##############################################
# Main Routine
#
my $input;
my $output;

{
  foreach my $TARGET qw/INPUT OUTPUT/ {
    chomp(my $data = `$ip6tables_cmd -vnx -L $TARGET --line-number | $head_cmd -1`);
    my @counter = split(/¥s+/, $data);
    if ($data =~ /INPUT/) {
        $input = $counter[6];
    } else {
        $output = $counter[6];
    }
  }
  print "$input¥n$output¥n";
}
```

- Measuring IPv6 traffics (MRTG)
- **IPv6 ready tools**
  - Smokeping
  - Nagios

smokeping

- Measuring the network latency
  - Port monitoring other than ICMP monitoring is also possible with addition of probe
- Site
  - http://oss.oetiker.ch/smokeping/

IPv6 support on Smokeping

- Specify fping6 at Probe

```
+ FPing6
binary = /usr/sbin/fping6
```

- Specify the configured Fping6 at Probe

When using Fping6 for standard Probe

```
probe = FPing6

menu = Top
title = Network Latency Grapher

+ Servers
menu= Server
title = Server
++ server1
menu = server1
title= server1
host = www.example.jp
```

Standard Probe is used with Fping

```
++ server1
probe=FPing6
menu = server1
title= server1
host = www.example.jp
```

Nagios

- Integrated monitoring tool
  - Flexible monitoring other than ICMP monitoring is possible using plug-ins
- Site
  - http://www.nagios.org/

Nagios IPv6 support

- Most of the plugins support IPv6 already, and you just have to specify the IPv6 address at $HOSTADDRESS$. If you're using FQDN, you have to define a commend.

```
define command {
            command_name        check_ping6
            command_line        $USER1$/check_ping –H $HOSTADDRESS$ -w
$ARG1$ -c $ARG2$ -p 5 -6
}

define command {
            command_name        check_ping4
            command_line        $USER1$/check_ping –H $HOSTADDRESS$ -w
$ARG1$ -c $ARG2$ -p 5 -4
}
```

Security

- Packet filter
  - Path MTU Discovery
  - Extension header
  - ip6tables overview
  - ip6tables configuration
- ACL
  - tcp_wrappers
  - Mapped_address
  - FQDN

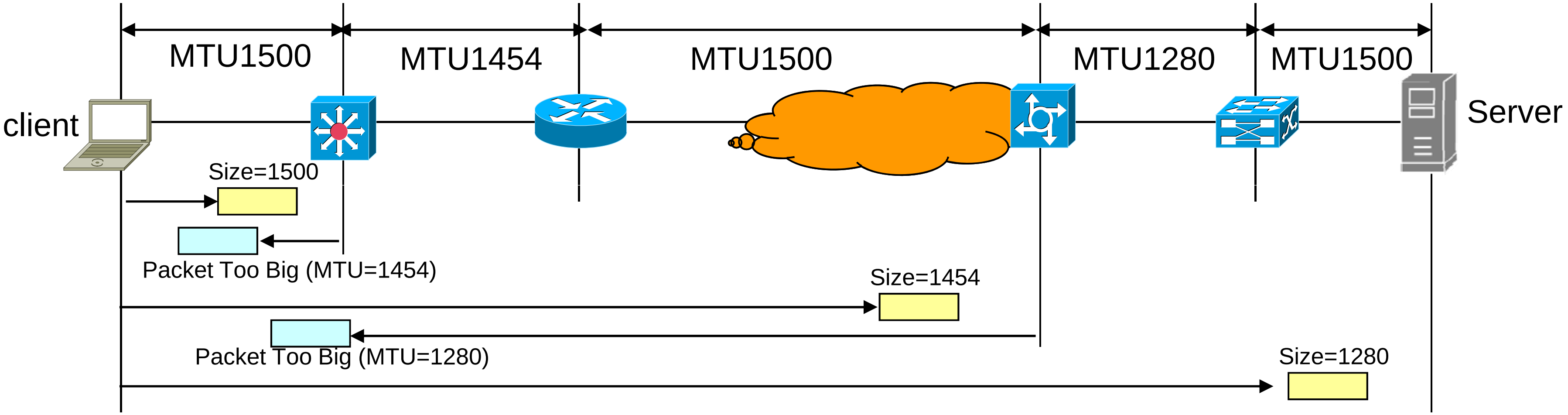
- **Packet Filter**
  - Path MTU Discovery
  - Extension headers
  - ip6tables overview
  - ip6tables configuration
- ACL
  - tcp_wrappers
  - mapped_address
  - FQDN

About packet filtering

- Unlike IPv4, the communication might stop when you filter all the ICMP messages in IPv6
- Why ICMPv6 is mandatory?
  - Fragmentation at intermediate routes is not allowed in IPv6 in order to decrease the load of intermediate routers. Therefore, Path MTU Discovery is required to transmit packets. (Path MTU Discovery requires ICMPv6)

Path MTU Discovery

- Source host assumes forwarding link's MTU as its path MTU
- When it is unable to forward packets at intermediate routes, a router discards the packet and sends back the Packet Too Big (ICMPv6 type2)message to the source address (Inserts the MTU information of the link to the next hop)
- IPv6 minimum MTU is 1280 byte
- Same with multicast
  - All the destination address's minimum MTU
- Discarded when the packets got filtered by L2SW's MTU



Basics of packet filtering

- As IPv6 assumes end-to-end communication, security should be assured at each end node.

- Precaution in IPv6
  - Do not filter ICMPv6
    - Especially type2 (Packet Too Big)
  - Pass EDNS0 and TCP53
    - As DNS response tend to be big in IPv6, these messages are necessary

- Dealing with extension header
  - Simple packet filtering does not support all features
    - RH0, fragment header, etc
  - Consideration is necessary when configuring firewalls

Packet filter 1 (Reference)

	Ingress	Egress
Mandatory	[1] Accept all ICMPv6 [2] Reject packets with following source address - Reserved address ::/8 - Deprecate-site local address fec0::/10 - Unique local address fc00::/7 - Multicast address ff00::/8 - Document address 2001:db8::/32 [3] Reject the packets with source address with its own AS's prefix (transit connection)	none
Option	[1] Limit the ICMPv6 packets destined to border interface -Prerequisite 1. Accept ICMPv6 TYPE that are used in Neighbor Discovery 2. Accept ICMPv6 TYPE =2 (Packet Too Big) used in Path MTU Discovery 3.Accept ICMPv6 TYPE = 1 (Destination Unreachable) in order to realize fast IPv6/IPv4 fallback. [2] Reject ICMPv6 other than then above described ones destined to border interfaces. *** Unable to confirm traceroute, ping otherwise [3] Reject 6bone address (obsolete) 3FFE::/16	[1] Accept all ICMPv6 [2] Reject the packets with following source address - Reserved address ::/8 - Deprecate-site local address fec0::/10 - Unique local address fc00::/7 - Multicast address ff00::/8 - Document address 2001:db8::/32 - 6bone address 3FFE::/16

Referred to: http://www.janog.gr.jp/doc/janog-comment/jc1006.txt

Packet Filter 2.1 (Reference)

Recommendations for ICMPv6 Transit Traffic

Traffic that Must Not be Dropped	Destination Unreachable (Type 1) - All codes Packet Too Big (Type 2) Time Exceeded (Type 3) - Code 0 only Parameter Problem (Type 4) - Codes 1 and 2 only Echo Request (Type 128) Echo Response (Type 129)
Traffic that Normally Should Not be Dropped	Time Exceeded (Type 3) - Code 1 Parameter Problem (Type 4) - Code 0 Home Agent Address Discovery Request (Type 144) Home Agent Address Discovery Reply (Type 145) Mobile Prefix Solicitation (Type 146) Mobile Prefix Advertisement (Type 147)
Traffic That Will Be Dropped Anyway (All these messages should never be propagated beyond the link which they were initially transmitted)	Router Solicitation (Type 133) Router Advertisement (Type 134) Neighbor Solicitation (Type 135) Neighbor Advertisement (Type 136) Redirect (Type 137) Inverse Neighbor Discovery Solicitation (Type 141) Inverse Neighbor Discovery Advertisement (Type 142) Listener Query (Type 130) Listener Report (Type 131) Listener Done (Type 132) Listener Report v2 (Type 143) Certificate Path Solicitation (Type 148) Certificate Path Advertisement (Type 149) Multicast Router Advertisement (Type 151) Multicast Router Solicitation (Type 152) Multicast Router Termination (Type 153)
Traffic for Which a Policy Should Be Defined	Seamoby Experimental (Type 150) Unallocated Error messages (Types 5-99 inclusive and 102-126 inclusive) Unallocated Informational messages (Types 154-199 inclusive and 202-254 inclusive)

Referred to: http://www.ietf.org/rfc/rfc4890.txt?number=4890

Packet Filter 2.2 (Reference)

Recommendations for ICMPv6 Local Configuration Traffic

Traffic that Must Not be Dropped	Destination Unreachable (Type 1) - All codes
	Packet Too Big (Type 2)
	Time Exceeded (Type 3) - Code 0 only
	Parameter Problem (Type 4) - Codes 1 and 2 only
	Echo Request (Type 128)
	Echo Response (Type 129)
	Router Solicitation (Type 133)
	Router Advertisement (Type 134)
	Neighbor Solicitation (Type 135)
	Neighbor Advertisement (Type 136)
	Inverse Neighbor Discovery Solicitation (Type 141)
	Inverse Neighbor Discovery Advertisement (Type 142)
	Listener Query (Type 130)
	Listener Report (Type 131)
	Listener Done (Type 132)
	Listener Report v2 (Type 143)
	Certificate Path Solicitation (Type 148)
	Certificate Path Advertisement (Type 149)
	Multicast Router Advertisement (Type 151)
	Multicast Router Solicitation (Type 152)
	Multicast Router Termination (Type 153)
Traffic that Normally Should Not be Dropped	Time Exceeded (Type 3) - Code 1
	Parameter Problem (Type 4) - Code 0
	Router Renumbering (Type 138)
Traffic That Will Be Dropped Anyway (if the service is not implemented)	Home Agent Address Discovery Request (Type 144)
	Home Agent Address Discovery Reply (Type 145)
	Mobile Prefix Solicitation (Type 146)
	Mobile Prefix Advertisement (Type 147)
	Seamoby Experimental (Type 150)
Traffic for Which a Policy Should Be Defined	Redirect (Type 137)
	Node Information Query (Type 139)
	Node Information Response (Type 140)
	Unallocated Error messages (Types 5-99 inclusive and 102-126 inclusive)

Referred to: http://www.ietf.org/rfc/rfc4890.txt?number=4890

ip6tables overview

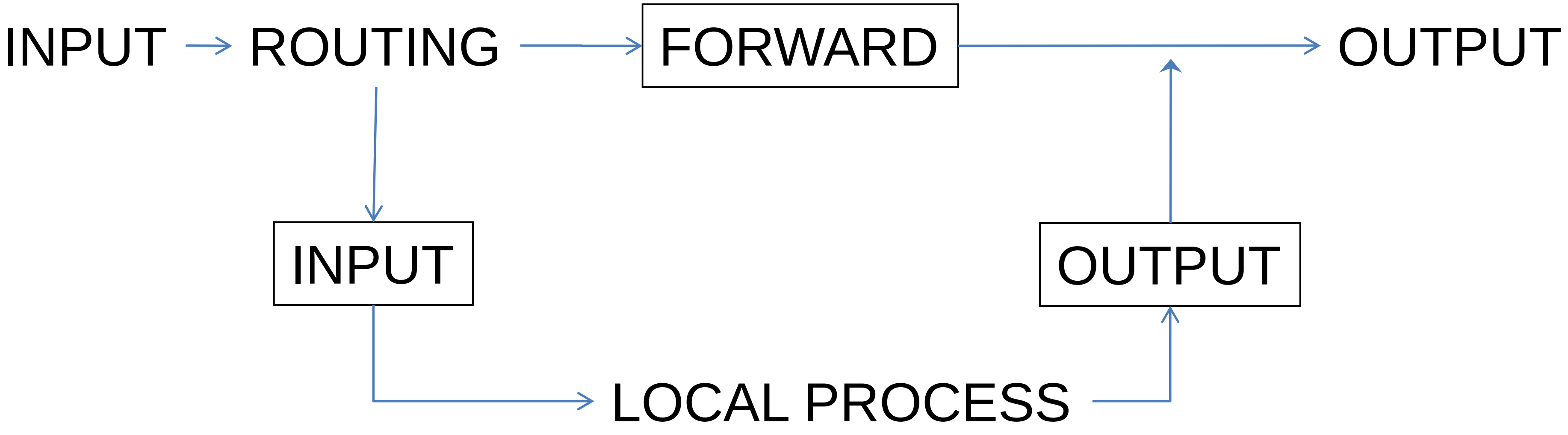
- IPv6 version of iptables
  - Packet filter rules in kernels can be configured
  - About the same with IPv4 except the NAT related configurations. Note that ICMP and ICMPv6 does not have compatibility.
  - Since Connection tracking is not supported in RHEL5/CentOS5, configuration can be done, but does not work as expected.

Example:

```
ip6tables -A TEST -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Ip6tables configuration

- Defines rules to each target



ICMPv6 configuration sample

```
# Allow some ICMPv6 types in the INPUT chain
# Using ICMPv6 type names to be clear.

ip6tables -A INPUT -p icmpv6 --icmpv6-type destination-unreachable -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type time-exceeded -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type parameter-problem -j ACCEPT


# Allow some other types in the INPUT chain, but rate limit.
ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-request -m limit --limit 900/min -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type echo-reply -m limit --limit 900/min -j ACCEPT

# Allow others ICMPv6 types but only if the hop limit field is 255.

ip6tables -A INPUT -p icmpv6 --icmpv6-type router-advertisement -m hl --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-solicitation -m hl --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type neighbor-advertisement -m hl --hl-eq 255 -j ACCEPT
ip6tables -A INPUT -p icmpv6 --icmpv6-type redirect -m hl --hl-eq 255 -j ACCEPT


# When there isn't a match, the default policy (DROP) will be applied.
# To be sure, drop all other ICMPv6 types.
# We're dropping enough icmpv6 types to break RFC compliance.

ip6tables -A INPUT -p icmpv6 -j LOG --log-prefix "dropped ICMPv6"
ip6tables -A INPUT -p icmpv6 -j DROP
```

ICMPv6 configuration sample（Cont)

```
# Allow ICMPv6 types that should be sent through the Internet.

ip6tables -A OUTPUT -p icmpv6 --icmpv6-type destination-unreachable -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type packet-too-big -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type time-exceeded -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type parameter-problem -j ACCEPT


# Limit most NDP messages to the local network.

ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbour-solicitation -m hl --hl-eq 255 -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type neighbour-advertisement -m hl --hl-eq 255 -j ACCEPT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type router-solicitation -m hl --hl-eq 255 -j ACCEPT


# If we're acting like a router, this could be a sign of problems.

ip6tables -A OUTPUT -p icmpv6 --icmpv6-type router-advertisement -j LOG --log-prefix "ra ICMPv6 type"
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type redirect -j LOG --log-prefix "redirect ICMPv6 type"
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type router-advertisement -j REJECT
ip6tables -A OUTPUT -p icmpv6 --icmpv6-type redirect -j REJECT


# Accept all other ICMPv6 types in the OUTPUT chain.

ip6tables -A OUTPUT -p icmpv6 -j ACCEPT
```

Reference: http://www.cert.org/downloads/IPv6/ip6tables_rules.txt

Security

- Packet filter
  - Path MTU Discovery
  - Extension headers
  - ip6tables overview
  - ip6tables configuration
- **ACL**
  - tcp_wrappers
  - Mapped_address
  - FQDN

tcp_wrappers

- tcp_wrappers is plural because this is provided in library format, too. This is why many distribution’s sshd can control the filter with following files without tcpd.
  - /etc/hosts.allow
  - /etc/hosts.deny

tcp_wrappers format

- IPv6 and network address have to be enclosed with [].

```
ALL:   [2001:db8:1000:2000::]/64
```

*) The format maybe different with new releases, and you should be careful when referring to old books

mapped address

- In case of an IPv6 node communicating with IPv4 node using mapped address, the remote address format will not be xx.xx.xx.xx, and failure might happen.

FQDN

- Some applications such as Apache decide access permission/rejection based on source node domain name. However, client reverse lookup may not be configured in many cases in IPv6, so you have to be careful when enabling IPv6 on servers currently in operation.

Operation

- **DNS reverse lookup**
- SMTP issues
- Log format
- Address auto-configuration
- Troubleshooting (FQDN)
- VLAN

DNS reverse lookup

- Considering the client addresses will be accessed using privacy extensions, ACL using DNS reverse lookup will not be in effect.

- DNS reverse lookup
- **SMTP issues**
- Log format
- Address auto-configuration
- Troubleshooting (FQDN)
- VLAN

SMTP issues

- There are wrong implementations that TCP fallback may not occur properly when a server of MX RR has A RR and AAAA RR. (very rare case.)

SMTP issues

- How to fix the problem

```
$ORIGIN example.jp.
;
@           IN          MX    10    mail
;
mail        IN          A           192.0.2.1
            IN          AAAA        2001:db8::25
```

Change this to……

```
$ORIGIN example.jp.
;
@           IN          MX    10    mail
                        MX    20    mail4
;
mail        IN          A           192.0.2.1
            IN          AAAA        2001:db8::25
;
Mail4       IN          A           192.0.2.1
```

- DNS reverse lookup
- SMTP issues
- **Log format**
- Address auto-configuration
- Troubleshooting (FQDN)
- VLAN

Log format

- There are no abbreviation rules in the syslog output addresses. It depends on the applications that outputs the logs. Therefore, address matching using grep sometimes does not work. (This issue is urrently under discussion in IETF)

- DNS reverse lookup
- SMTP issues
- Log format
- **Address auto-configuration**
- Troubleshooting (FQDN)
- VLAN

Address auto-configuration

- This was supposed to decrease the work load of operators who had to look over multiple client nodes.
  - However, you should not use auto-configurations in servers because the address might change when you change the NIC, etc
    - Filtering rules have to be changed with address change

- DNS reverse lookup
- SMTP issues
- Log format
- Address auto-configuration
- **Troubleshooting (FQDN)**
- VLAN

Trouble shooting (FQDN)

- If a FQDN is registered in monitoring tools, make sure the tools are not be effected by the IPv6/IPv4 fallbacks

Operation

- DNS reverse lookup
- SMTP issues
- Log format
- Address auto-configuration
- Troubleshooting (FQDN)
- **VLAN**

VLAN

- No major problem in port VLAN or tagged VLAN
- MAC address based VLAN
  - Probably not used in ISPs, but there are no feasible implementations
    - When a RA is transmitted to multicast address, all users will receive the address
    - It is better not to use it

- Q&A?

Reference

- How to be connected via IPv6?
- IPv6 terminologies

How to be connected via IPv6?

- Purchase the IPv6 transit from the upstream ISP
  - There are more than you expect
- Contract with IPv6 ready iDC
  - There are more than you expect
- Find a Tunnel Broker
  - Feel6(dtcp)
  - OCN IPv6(L2TP)

http://ipv6.blog.ocn.ne.jp/ipv6/2006/04/linuxocn_ipv62_5915.html

IPv6 terminologies

- Node
  - – IPv6 ready device
- Router
  - – A node that transfers IPv6 packets to others
- Host
  - – Any node other than routers

IPv6 terminologies cont(2)

- Upper layer
  - Upper protocol layer to IPv6 (TCP,UDP, etc)
- Link
  - Lower protocol layer to IPv6 (Ethernet, PPP, etc)
- Neighbors
  - Nodes connected on a same link
- Interface
  - Attachment of a node that is used to connect to a link

IPv6 terminologies cont(3)

- Packet
  - Payload including IPv6 header (Data part)
- Link MTU
  - Number of Maximum Transmission Unit on the link
- Path MTU
  - Minimum link MTU among all the paths from source to destination node