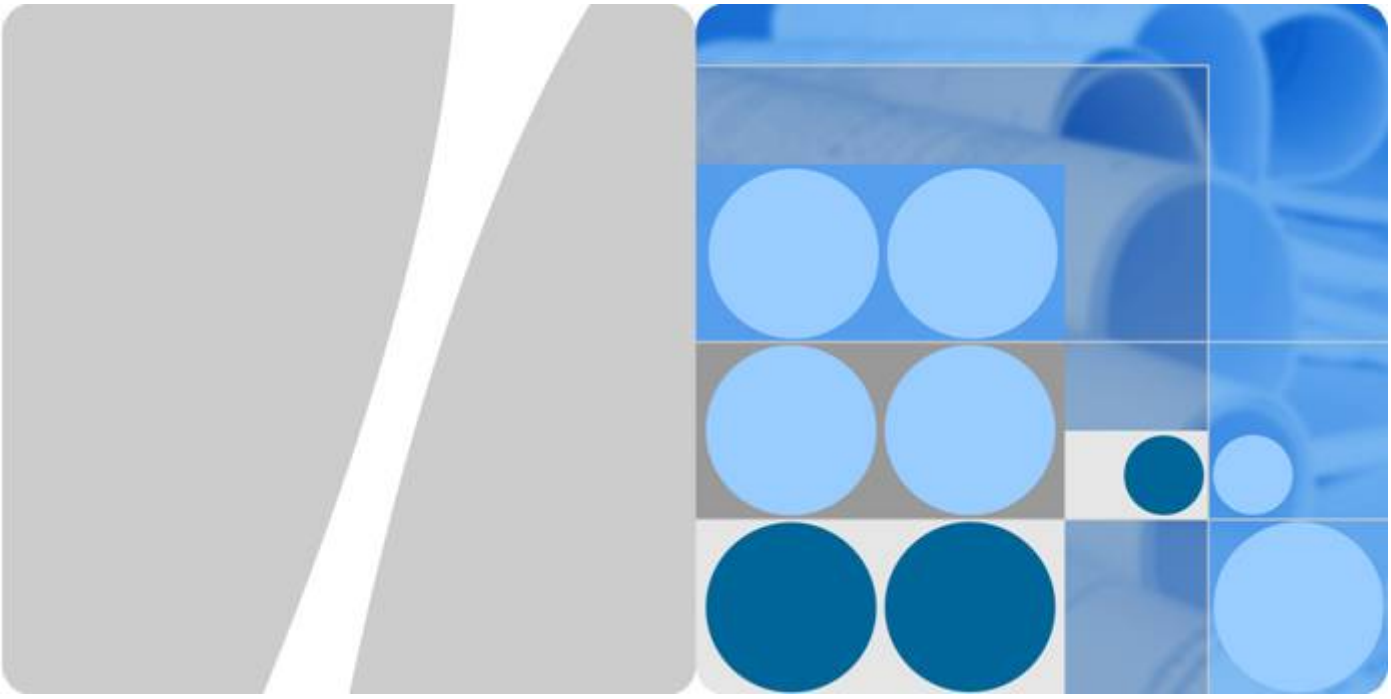


CloudEngine 12800 Series Switches

IPv6 Technical White Paper

Issue 02

Date 2016-06-15

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://e.huawei.com

Contents

1 Basic IPv6

About This Chapter

1.1 IPv6 Overview

Definition

Internet Protocol version 6 (IPv6), also called IP Next Generation (IPng), is the second-generation network layer protocol. Designed by the Internet Engineering Task Force (IETF), IPv6 is an upgraded version of Internet Protocol version 4 (IPv4).

Purpose

IPv6 was developed in response to rapidly increasing Internet use. IPv4, despite being easy to implement, simple to use, and providing good interoperability, is no longer feasible as the dominant network layer protocol. This is mainly due to IPv4 address exhaustion.

Table 1-1 shows how IPv6 overcomes many of the deficiencies found in IPv4.

Table 1-1 Comparison between IPv6 and IPv4

Item	Deficiency in IPv4	Advantage of IPv6
Address space	IPv4 addresses are 32 bits long, theoretically giving an available IP address space that contains about 4.3 billion IP addresses. The currently available IP addresses are no longer sufficient to continually support the rapid expansion of the Internet. IPv4 address resources are allocated unevenly. USA address resources account for almost half of the global address space, with barely enough addresses left for Europe, and still fewer for the Asia-Pacific area. Furthermore, the development of mobile IP and broadband technologies still requires more IP addresses. The process of IP addresses being used up is known as IP address exhaustion. While several solutions to IPv4 exhaustion are currently in place, such as Classless Inter-domain Routing (CIDR) and Network Address Translator (NAT), they all have significant disadvantages. These disadvantages prompted the development of IPv6.	IPv6 addresses are 128 bits long. A 128 bit structure allows for an address space of 2^{128} (4.3 billion x 4.3 billion x 4.3 billion x 4.3 billion) possible addresses. This vast address space makes it very unlikely that IPv6 address exhaustion will ever occur.

Item	Deficiency in IPv4	Advantage of IPv6
Packet format	The IPv4 packet header carries the Options field, including security, timestamp, and record route options. The variable length of the Options field makes the IPv4 packet header length range from 20 bytes to 60 bytes. IPv4 packets often need to be forwarded by intermediate devices. Therefore, using the Options field occupies a large amount of resources, which means this field is rarely used in practice.	Unlike the IPv4 packet header, the IPv6 packet header does not carry IHL, identifier, flag, fragment offset, header checksum, option, or padding fields, but it does carry the flow label field. This facilitates IPv6 packet processing and improves processing efficiency. To support various options without changing the existing packet format, the Extension Header information field is added to the IPv6 packet header, improving IPv6 flexibility.
Autoconfiguration and readdressing	IP addresses often need to be reallocated during network expansion or re-planning. Currently, IPv4 depends on Dynamic Host Configuration Protocol (DHCP) to provide address autoconfiguration and readdressing to simplify address maintenance.	IPv6 provides address autoconfiguration to allow hosts to automatically discover networks and obtain IPv6 addresses, improving network manageability.
Route summarization	Many non-contiguous IPv4 addresses are allocated. Routes cannot be summarized effectively due to incorrect IPv4 address allocation and planning. The increasingly large routing table consumes a lot of memory and affects forwarding efficiency. Manufacturers must continually upgrade devices to improve route addressing and forwarding performance.	The enormous address space allows for the hierarchical network design in IPv6 to facilitate route summarization and improve forwarding efficiency.
End-to-end security support	The original IPv4 framework does not support end-to-end security because security was not fully considered during the initial design.	IPv6 supports IP Security (IPSec) authentication and encryption at the network layer, providing end-to-end security.
Quality of Service (QoS) support	IPv4 has no native mechanism to support QoS, especially when regarding real-time forwarding of voice, data, and video services such as network conferencing, network telephones, and network TVs.	The Flow Label field in IPv6 guarantees QoS for voice, data, and video services.

Item	Deficiency in IPv4	Advantage of IPv6
Mobility	Due to the development of the Internet, mobile IPv4 experiences significant problems such as triangular routing and source address filtering.	Mobile IPv6 improves mobile communication efficiency and is transparent to the application layer because IPv6 has the native capability to support mobility. Unlike mobile IPv4, mobile IPv6 uses the neighbor discovery function to discover a foreign network and obtain a care-of address without using any foreign agent. The mobile node and peer node can communicate using the routing header and destination options header. This function solves the problems of triangular routing and source address filtering found in mobile IPv4.

1.2 Principles

This section explains the fundamentals of IPv6 addresses, IPv6 packets, ICMPv6, Neighbor Discovery Protocol (NDP), and path maximum transmission units (PMTUs).

1.2.1 IPv6 Addresses

IPv6 Address Formats

An IPv6 address is 128 bits long and is written as eight groups of four hexadecimal digits (base 16 digits represented by the numbers 0-9 and the letters A-F). Each group is separated by a colon (:). For example, FC00:0000:130F:0000:0000:09C0:876A:130B is a complete and valid IPv6 address.

For convenience, IPv6 addresses can be written in a compressed format. Taking the IPv6 address FC00:0000:130F:0000:0000:09C0:876A:130B as an example:

- Any leading zeroes in a group can be omitted. The example address now becomes FC00:0:130F:0:0:9C0:876A:130B.
- A double colon (::) can be used when two or more consecutive groups contain all zeros. The example address now becomes FC00:0:130F::9C0:876A:130B.

NOTE

An IPv6 address can contain only one double colon (::). Otherwise, a computer cannot determine the number of zeros in a group when restoring the compressed address to the original 128-bit address.

IPv6 Address Structure

IPv6 addresses have two parts:

- Network prefix: Corresponds to the network ID of an IPv4 address. It is comprised of *n* bits.

- Interface identifier (interface ID): Corresponds to the host ID of an IPv4 address. It is comprised of 128-*n* bits.

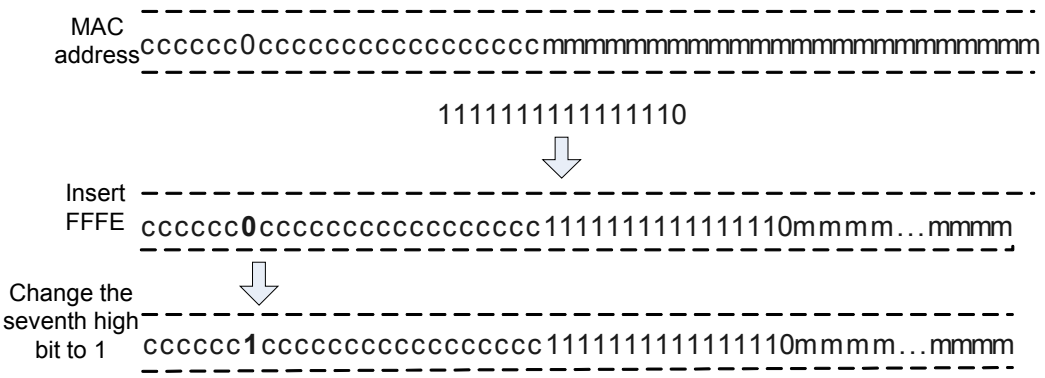
NOTE

If the first 3 bits of an IPv6 unicast address are not 000, the interface ID must contain 64 bits. If the first 3 bits are 000, there is no such limitation.

You can manually configure the interface ID, generate it through system software, or generate it in IEEE 64-bit Extended Unique Identifier (EUI-64) format. Generating an interface ID in EUI-64 format is the most common practice.

IEEE EUI-64 standards convert an interface MAC address into an IPv6 interface ID. **Figure 1-1** shows a 48-bit MAC address. When used as an interface ID, the first 24 bits (expressed by c) are a vendor identifier, and the last 24 bits (expressed by m) are an extension identifier. If the higher seventh bit is 0, the MAC address is locally unique. During conversion, EUI-64 inserts FFFE between the vendor identifier and extension identifier. The higher seventh bit also changes from 0 to 1 to indicate that the interface ID is globally unique.

Figure 1-1 EUI-64 format



For example, if the MAC address is 000E-0C82-C4D4, the interface ID is 020E:0CFF:FE82:C4D4 after the conversion.

Converting MAC addresses into IPv6 interface IDs reduces the configuration workload. When using stateless address autoconfiguration, you only need an IPv6 network prefix to obtain an IPv6 address. One defect of this method, however, is that an IPv6 address is easily calculable based on a MAC address, and could therefore be used for malicious attacks.

IPv6 Address Types

IPv6 addresses can be classified as unicast, multicast, or a new class called anycast. Unlike IPv4, there is no broadcast IPv6 address. Instead, a multicast address can be used as a broadcast address.

IPv6 Unicast Address

An IPv6 unicast address identifies an interface. Since each interface belongs to a node, the IPv6 unicast address of any interface can identify the relevant node. Packets sent to an IPv6 unicast address are delivered to the interface identified by that address.

IPv6 defines multiple types of unicast addresses, including the unspecified address, loopback address, global unicast address, link-local address, and unique local address.

- Unspecified address

  The IPv6 unspecified address is 0:0:0:0:0:0:0:0/128 or ::/128, indicating that an interface or a node does not have an IP address. It can be used as the source IP address of some packets, such as Neighbor Solicitation (NS) messages, in duplicate address detection. Devices do not forward packets with an unspecified address as the source IP address.

- Loopback address

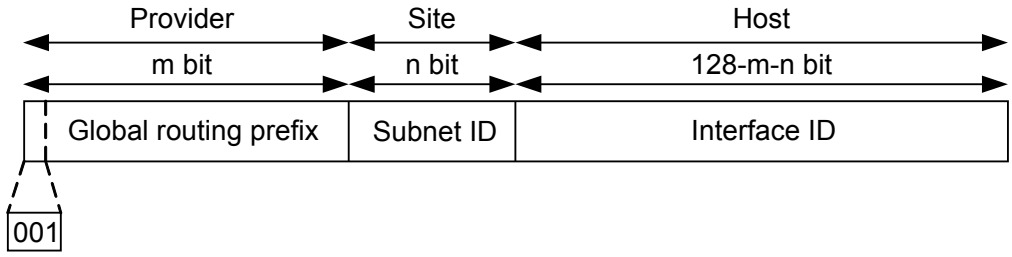
  The IPv6 loopback address is 0:0:0:0:0:0:0:1/128 or ::1/128. Similar to the IPv4 loopback address 127.0.0.1, the IPv6 loopback address is used when a node needs to send IPv6 packets to itself. This IPv6 loopback address is usually used as the IP address of a virtual interface, such as a loopback interface. The loopback address cannot be used as the source or destination IP address of packets needing to be forwarded.

- Global unicast address

  An IPv6 global unicast address is an IPv6 address with a global unicast prefix, which is similar to an IPv4 public address. IPv6 global unicast addresses support route prefix summarization, helping limit the number of global routing entries.

  Figure 1-2 shows a global unicast address consisting of a global routing prefix, subnet ID, and interface ID.

  Figure 1-2 Global unicast address format



  These components are described as follows:

  Global routing prefix: is assigned by a service provider to an organization. A global routing prefix is comprised of at least 48 bits. Currently, the first 3 bits of every assigned global routing prefix is 001.

  Subnet ID: is used by organizations to construct a local network (site). Similar to an IPv4 subnet number, there are a maximum of 64 bits for both the global routing prefix and subnet ID.

  Interface ID: identifies a device (host).

- Link-local address

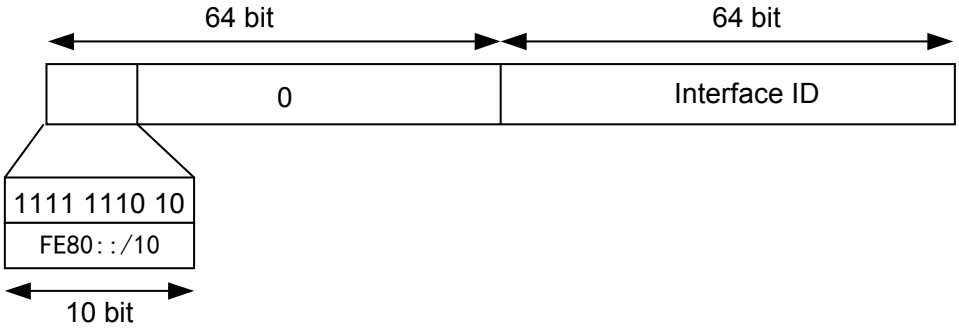
  Link-local addresses are used only in communication between nodes on the same local link. A link-local address uses a link-local prefix of FE80::/10 as the first 10 bits (1111111010 in binary) and an interface ID as the last 64 bits.

  When IPv6 runs on a node, a link-local address that consists of a fixed prefix and an interface ID in EUI-64 format is automatically assigned to each interface of the node. This mechanism enables two IPv6 nodes on the same link to communicate without any configuration, making link-local addresses widely used in neighbor discovery and stateless address configuration.

  Devices do not forward IPv6 packets with the link-local address as a source or destination address to devices on different links.

Figure 1-3 shows the link-local address format.

Figure 1-3 Link-local address format



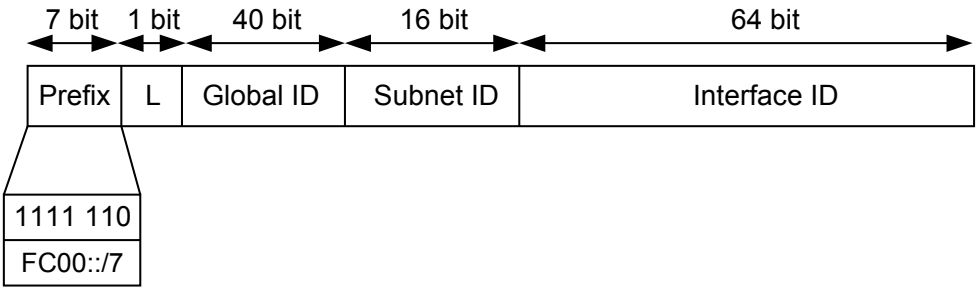
- Unique local address

  Unique local addresses are used only within a site. Site-local addresses, according to RFC 3879, have been replaced by unique local addresses.

  Unique local addresses are similar to IPv4 private addresses. Any organization that does not obtain a global unicast address from a service provider can use a unique local address. However, they are routable only within a local network, not the Internet as a whole.

  Figure 1-4 shows the unique local address format.

  Figure 1-4 Unique local address format

  These components are described as follows:

  Prefix: is fixed as FC00::/7.

  L: is set to 1 if the address is valid within a local network. The value 0 is reserved for future expansion.

  Global ID: indicates a globally unique prefix, which is pseudo-randomly allocated (for details, see RFC 4193).

  Subnet ID: identifies a subnet within the site.

  Interface ID: identifies an interface.

  A unique local address has the following features:

  - Has a globally unique prefix that is pseudo-randomly allocated with a high probability of uniqueness.

- Allows private connections between sites without creating address conflicts.
- Has a well-known prefix (FC00::/7) that allows for easy route filtering at site boundaries.
- Does not conflict with any other addresses if it is accidentally routed offsite.
- Functions as a global unicast address to applications.
- Is independent of Internet Service Providers (ISPs).

IPv6 Multicast Address

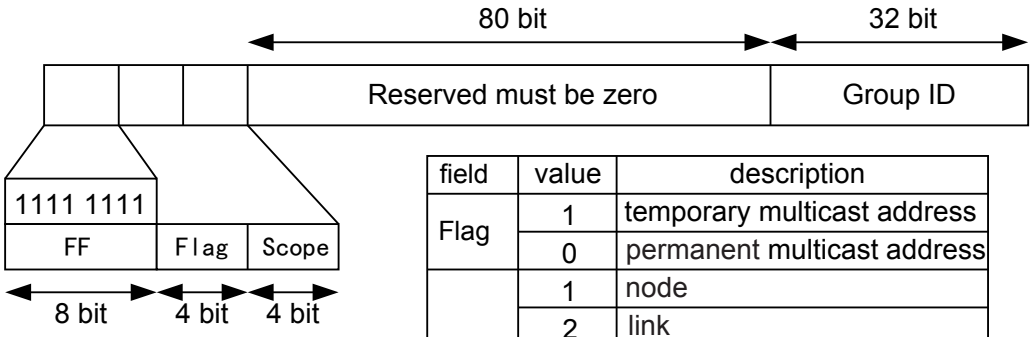
Like IPv4 multicast addresses, IPv6 multicast addresses identify groups of interfaces, which usually belong to different nodes. A node may belong to any number of multicast groups. Packets sent to an IPv6 multicast address are delivered to all the interfaces identified by the multicast address. For example, the multicast address FF02::1 indicates all nodes within the link-local scope, and FF02::2 indicates all routers within the link-local scope.

An IPv6 multicast address is composed of a prefix, a flag, a scope, and a group ID (global ID).

- Prefix: is fixed as FF00::/8.
- Flag: is 4 bits long. The high-order 3 bits are reserved and must be set to 0s. The last bit 0 indicates a permanently-assigned, well-known multicast address allocated by the Internet Assigned Numbers Authority (IANA). The last bit 1 indicates a non-permanently-assigned (transient) multicast address.
- Scope: is 4 bits long. It limits the scope where multicast data flows are sent on the network. **Figure 1-5** shows the field values and meanings.
- Group ID (global ID): is 112 bits long. It identifies a multicast group. RFC 2373 does not define all the 112 bits as a group ID but recommends using the low-order 32 bits as the group ID and setting all of the remaining 80 bits to 0s. In this case, each multicast group ID maps to a unique Ethernet multicast MAC address (for details, see RFC 2464).

Figure 1-5 shows the IPv6 multicast address format.

Figure 1-5 IPv6 multicast address format



field	value	description
Flag	1	temporary multicast address
	0	permanent multicast address
Scope	1	node
	2	link
	4	management
	5	site
	8	organization
	E	global
	the rest	unsigned or reserved

- Solicited-node Multicast Address

  A solicited-node multicast address is generated using an IPv6 unicast or anycast address of a node. When a node has an IPv6 unicast or anycast address, a solicited-node multicast address is generated for the node, and the node joins the multicast group that corresponds to its IPv6 unicast or anycast address. Each unicast or anycast address corresponds to a single solicited-node multicast address, which is often used in neighbor discovery and duplicate address detection.

  IPv6 does not support broadcast addresses or Address Resolution Protocol (ARP). In IPv6, Neighbor Solicitation (NS) packets are used to resolve IP addresses to MAC addresses. When a node needs to resolve an IPv6 address to a MAC address, it sends an NS packet in which the destination IP address is the solicited-node multicast address corresponding to the IPv6 address.

  The solicited-node multicast address consists of the prefix FF02::1:FF00:0/104 and the last 24 bits of the corresponding unicast address.

IPv6 Anycast Address

An anycast address identifies a group of network interfaces, which usually belong to different nodes. Packets sent to an anycast address are delivered to the nearest interface that is identified by the anycast address, depending on the routing protocols.

Anycast addresses implement redundancy backup and load balancing functions when multiple hosts or nodes are provided with the same services. Currently, a unicast address is assigned to more than one interface to make a unicast address become an anycast address. When sending data packets to anycast addresses, senders cannot determine which of the assigned devices will receive the packets. Which device receives the packets depends on the routing protocols running on the network. Anycast addresses are used in stateless applications, such as Domain Name Service (DNS).

IPv6 anycast addresses are allocated from the unicast address space. Mobile IPv6 applications also use anycast addresses.

NOTE

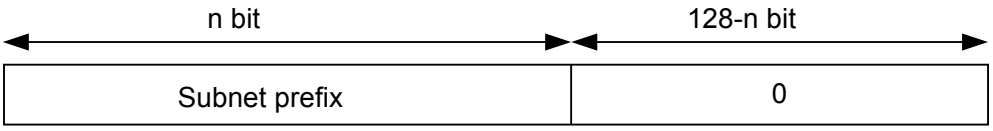
IPv6 anycast addresses can be assigned only to routing devices but not hosts. Anycast addresses cannot be used as the source IP addresses of IPv6 packets.

- Subnet-router Anycast Address

  RFC 3513 predefines a subnet-router anycast address. Packets sent to a subnet-router anycast address are delivered to the nearest device on the subnet identified by the anycast address, depending on the routing protocols. All devices must support subnet-router anycast addresses. A subnet-router anycast address is used when a node needs to communicate with any of the devices on the subnet identified by the anycast address. For example, a mobile node needs to communicate with one of the mobile agents on the home subnet.

  In a subnet-router anycast address, the *n*-bit subnet prefix identifies a subnet, and the remaining bits are padded with 0s. **Figure 1-6** shows the subnet-router anycast address format.

Figure 1-6 Subnet-router anycast address format



1.2.2 IPv6 Packet Format

An IPv6 packet has three parts: an IPv6 basic header, one or more IPv6 extension headers, and an upper-layer protocol data unit (PDU).

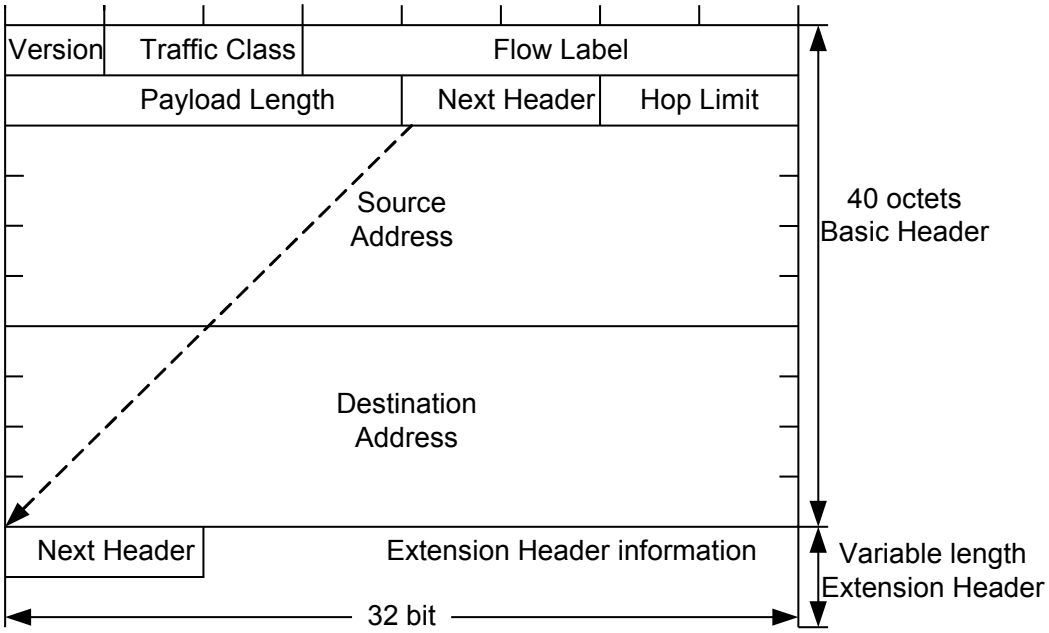
An upper-layer PDU is composed of the upper-layer protocol header and its payload, which maybe an ICMPv6 packet, a TCP packet, or a UDP packet.

IPv6 Basic Header

An IPv6 basic header is fixed as 40 bytes long and has eight fields. Each IPv6 packet must have an IPv6 basic header that provides basic packet forwarding information, and which all devices parse on the forwarding path.

Figure 1-7 shows the IPv6 basic header.

Figure 1-7 IPv6 basic header



An IPv6 basic header contains the following fields:

- Version: 4 bits long. In IPv6, the value of the Version field is set to 6.
- Traffic Class: 8 bits long. This field indicates the class or priority of an IPv6 packet. The Traffic Class field is similar to the TOS field in an IPv4 packet and is mainly used in QoS control.
- Flow Label: 20 bits long. This field was added in IPv6 to differentiate traffic. A flow label and source IP address identify a data flow. Intermediate network devices can effectively differentiate data flows based on this field.
- Payload Length: 16 bits long. This field indicates the length of the IPv6 payload in bytes. The payload is the part of the IPv6 packet following the IPv6 basic header, including the extension header and upper-layer PDU. This field has a maximum value of 65535. If the

payload length exceeds 65535 bytes, the field is set to 0, and the Jumbo Payload option in the Hop-by-Hop Options header is used to express the actual payload length.

- Next Header: 8 bits long. This field identifies the type of the first extension header that follows the IPv6 basic header or the protocol type in the upper-layer PDU.
- Hop Limit: 8 bits long. This field is similar to the Time to Live field in an IPv4 packet, defining the maximum number of hops that an IP packet can pass through. Each device that forwards the packet decrements the field value by 1. If the field value is reduced to 0, the packet is discarded.
- Source Address: 128 bits long. This field indicates the address of the packet originator.
- Destination Address: 128 bits long. This field indicates the address of the packet recipient.

Unlike the IPv4 packet header, the IPv6 packet header does not carry IHL, identifier, flag, fragment offset, header checksum, option, or padding fields, but it does carry the flow label field. This facilitates IPv6 packet processing and improves processing efficiency. To support various options without changing the existing packet format, the Extension Header information field is added to the IPv6 packet header, improving flexibility. The following paragraphs describe IPv6 extension headers.

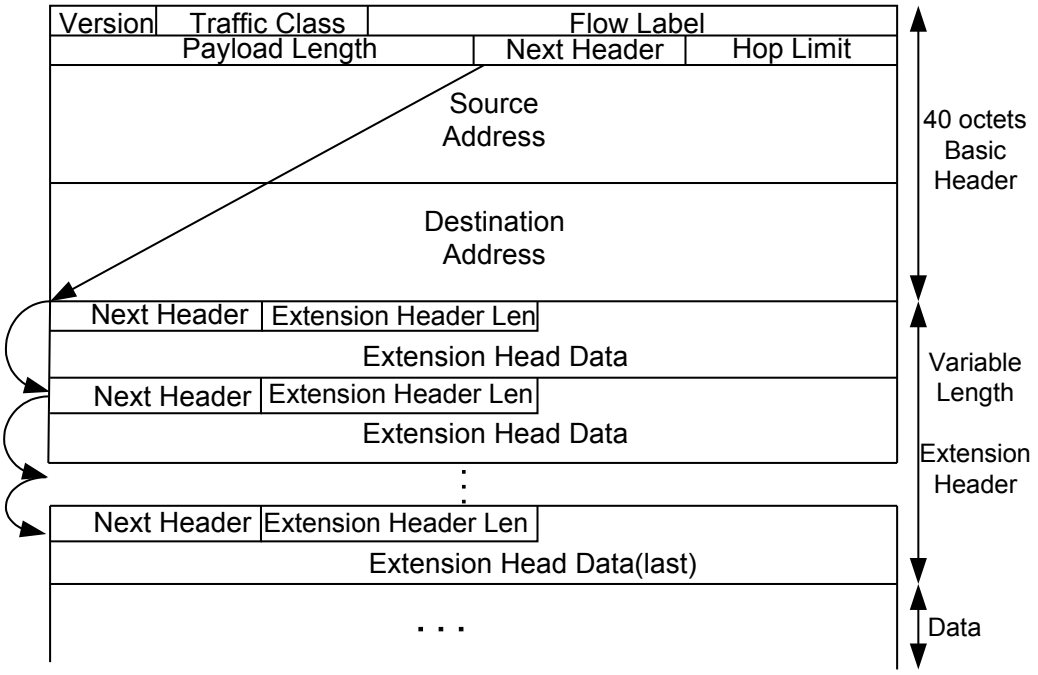
IPv6 Extension Header

An IPv4 packet header has an optional field (Options), which includes security, timestamp, and record route options. The variable length of the Options field makes the IPv4 packet header length range from 20 bytes to 60 bytes. When devices forward IPv4 packets with the Options field, many resources need to be used. Therefore, these IPv4 packets are rarely used in practice.

To improve packet processing efficiency, IPv6 uses extension headers to replace the Options field in the IPv4 header. Extension headers are placed between the IPv6 basic header and upper-layer PDU. An IPv6 packet may carry zero or more extension headers. The sender of a packet adds one or more extension headers to the packet only when the sender requests the destination device or other devices to perform special handling. Unlike IPv4, IPv6 has variable-length extension headers, which are not limited to 40 bytes. This facilitates further extension. To improve extension header processing efficiency and transport protocol performance, IPv6 requires that the extension header length be an integer multiple of 8 bytes.

When multiple extension headers are used, the Next Header field of an extension header indicates the type of the next header following this extension header. The Next Header field in the IPv6 basic header indicates the type of the first extension header, and the Next Header field in the first extension header indicates the type of the next extension header. If there are no extension headers following the current one, the Next Header field indicates the upper-layer protocol type. **Figure 1-8** shows the IPv6 extension header format.

Figure 1-8 IPv6 extension header format



An IPv6 extension header contains the following fields:

- Next Header: 8 bits long. This is similar to the Next Header field in the IPv6 basic header, indicating the type of the next extension header (if any) or the upper-layer protocol type.
- Extension Header Len: 8 bits long. This indicates the extension header length excluding the Next Header field.
- Extension Head Data: Variable length. This includes a series of options and the padding field.

RFC 2460 defines six IPv6 extension headers: Hop-by-Hop Options header, Destination Options header, Routing header, Fragment header, Authentication header, and Encapsulating Security Payload header.

Table 1-2 IPv6 extension headers

Header Type	Next Header Field Value	Description
Hop-by-Hop Options header	0	This header carries information that every node must examine along the delivery path of a packet. This header is used in the following applications: ● Jumbo payload (if the payload length exceeds 65535 bytes) ● Prompting devices to check this option before the devices forward packets. ● Resource Reservation Protocol (RSVP)
Destination Options header	60	This header carries information that only the destination node of a packet examines. Currently, this header is used in mobile IPv6.
Routing header	43	An IPv6 source node uses this header to specify the intermediate nodes that a packet must pass through on the way to its destination. This option is similar to the Loose Source and Record Route option in IPv4.
Fragment header	44	Like IPv4 packets, the length of IPv6 packets to be forwarded cannot exceed the maximum transmission unit (MTU). When the packet length exceeds the MTU, the packet needs to be fragmented. In IPv6, the Fragment header is used by an IPv6 source node to send a packet larger than the MTU.
Authentication header	51	IPSec uses this header to provide data origin authentication, data integrity check, and packet anti-replay functions. It also protects some fields in the IPv6 basic header.
Encapsulating Security Payload header	50	This header provides the same functions as the Authentication header plus IPv6 packet encryption.

Conventions for IPv6 extension headers

When a single packet uses more than one extension header, the headers must be listed in the following order:

- IPv6 basic header
- Hop-by-Hop Options header
- Destination Options header

- Routing header
- Fragment header
- Authentication header
- Encapsulating Security Payload header
- Destination Options header
- Upper-layer header

Intermediate devices determine whether to process extension headers based on the Next Header field value in the IPv6 basic header. The intermediate devices do not need to examine or process all extension headers.

Each extension header can only occur once in an IPv6 packet, except for the Destination Options header which may occur twice (once before a Routing header and once before the upper-layer header).

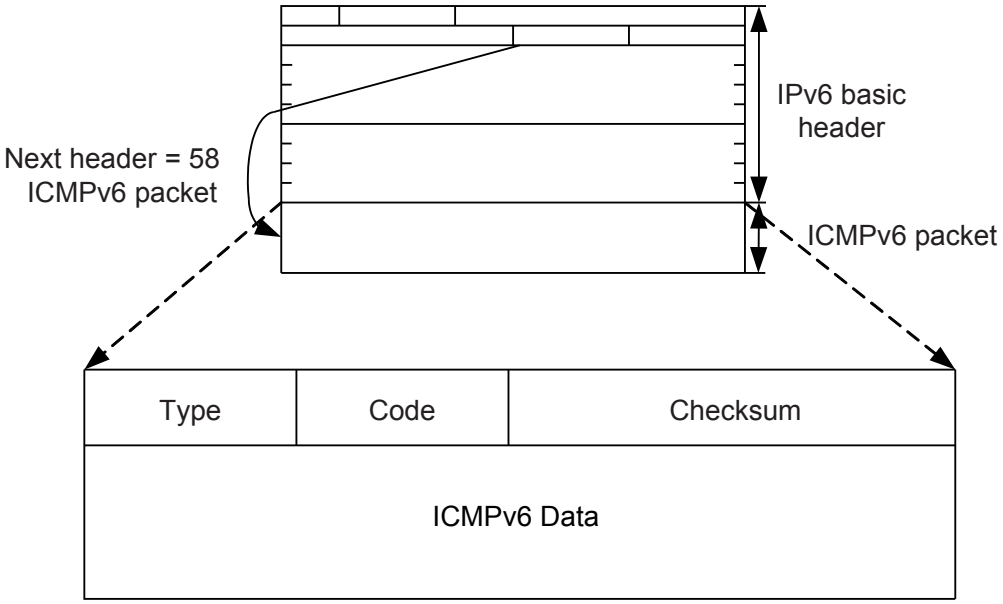
1.2.3 ICMPv6

The Internet Control Message Protocol version 6 (ICMPv6) is one of the basic IPv6 protocols.

In IPv4, ICMP reports IP packet forwarding information and errors to the source node. ICMP defines certain messages such as Destination Unreachable, Packet Too Big, Time Exceeded, Echo Request, and Echo Reply to facilitate fault diagnosis and information management. ICMPv6 provides additional mechanisms alongside the current ICMPv4 functions such as Neighbor Discovery (ID), stateless address configuration (including duplicate address detection), and Path Maximum Transmission Unit (PMTU) discovery.

The protocol number of ICMPv6 (that is, the value of the Next Header field in an IPv6 packet) is 58. **Figure 1-9** shows the ICMPv6 packet format.

Figure 1-9 Format of an ICMPv6 packet



Some fields in the packet are described as follows:

- Type: specifies a message type. Values 0 to 127 indicate the error message type, and values 128 to 255 indicate the informational message type.
- Code: indicates a specific message type.
- Checksum: indicates the checksum of an ICMPv6 packet.

Classification of ICMPv6 Error Messages

ICMPv6 error messages are generated when errors occur during IPv6 packet forwarding. They are classified into the following four types:

- Destination Unreachable message

  During IPv6 packet forwarding, if an IPv6 node detects that the destination address of a packet is unreachable, it sends an ICMPv6 Destination Unreachable message to the source node carrying information about the causes for the error message.

  In an ICMPv6 Destination Unreachable message, the value of the Type field is 1. Depending on the cause, the value of the Code field can be:

  - 0: No route to the destination device.
  - 1: Communication with the destination device is administratively prohibited.
  - 2: Not assigned.
  - 3: Destination IP address is unreachable.
  - 4: Destination port is unreachable.
- Packet Too Big message

  If an IPv6 node, during IPv6 packet forwarding, detects that the size of a packet exceeds the link MTU of the outbound interface, it sends an ICMPv6 Packet Too Big message to the source node. The link MTU of the outbound interface is carried in the message. PMTU discovery is implemented based on Packet Too Big messages.

  In a Packet Too Big message, the Type field value is 2 and the Code field value is 0.
- Time Exceeded message

  During the transmission of IPv6 packets, when a device receives a packet with a hop limit of 0 or a device reduces the hop limit to 0, it sends an ICMPv6 Time Exceeded message to the source node. During the processing of a packet to be fragmented and reassembled, an ICMPv6 Time Exceeded message is also generated when the reassembly time is longer than the specified period.

  In a Time Exceeded message, the Type field value is 3. Depending on the cause, the Code field value can be:

  - 0: Hop limit exceeded in packet transmission.
  - 1: Fragment reassembly timeout.
- Parameter Problem message

  When a destination node receives an IPv6 packet, it checks the validity of the packet. If it detects an error, it sends an ICMPv6 Parameter Problem message to the source node.

  In a Parameter Problem message, the Type field value is 4. Depending on the cause, the Code field value can be:

  - 0: A field in the IPv6 basic header or extension header is incorrect.
  - 1: The Next Header field in the IPv6 basic header or extension header cannot be identified.
  - 2: Unknown options exist in the extension header.

Classification of ICMPv6 Information Messages

ICMPv6 information messages provide diagnosis and additional host functions such as Multicast Listener Discovery (MLD) and Neighbor Discovery (ND). Common ICMPv6 information messages include Ping messages, which consist of Echo Request and Echo Reply messages.

- Echo Request messages: Echo Request messages are sent to destination nodes. After receiving an Echo Request message, the destination node responds with an Echo Reply message. In an Echo Request message, the Type field value is 128 and the Code field value is 0.
- Echo Reply messages: After receiving an Echo Request message, the destination node responds with an Echo Reply message. In an Echo Reply message, the Type field value is 129 and the Code field value is 0.

1.2.4 Neighbor Discovery

The Neighbor Discovery Protocol (NDP) is an enhancement of Address Resolution Protocol (ARP) and Internet Control Management Protocol (ICMP) router discovery in IPv4. In addition to ICMPv6 address resolution, NDP also provides the neighbor tracking, duplicate address detection, router discovery, and redirection functions.

Address Resolution

In IPv4, a host needs to obtain the link-layer address of the destination host through the ARP protocol for communication. Similar to IPv4, the IPv6 NDP protocol parses the IP address to obtain the link-layer address.

ARP packets are encapsulated in Ethernet packets. The Ethernet type value is 0x0806. ARP is defined as a protocol that runs between Layer 2 and Layer 3. ND is implemented through ICMPv6 packets. The Ethernet type value is 0x86dd. The Next Header value in the IPv6 header is 58, indicating that the packets are ICMPv6 packets. NDP packets are encapsulated in ICMPv6 packets. NDP is a Layer 3 protocol. Layer 3 address resolution has the following advantages:

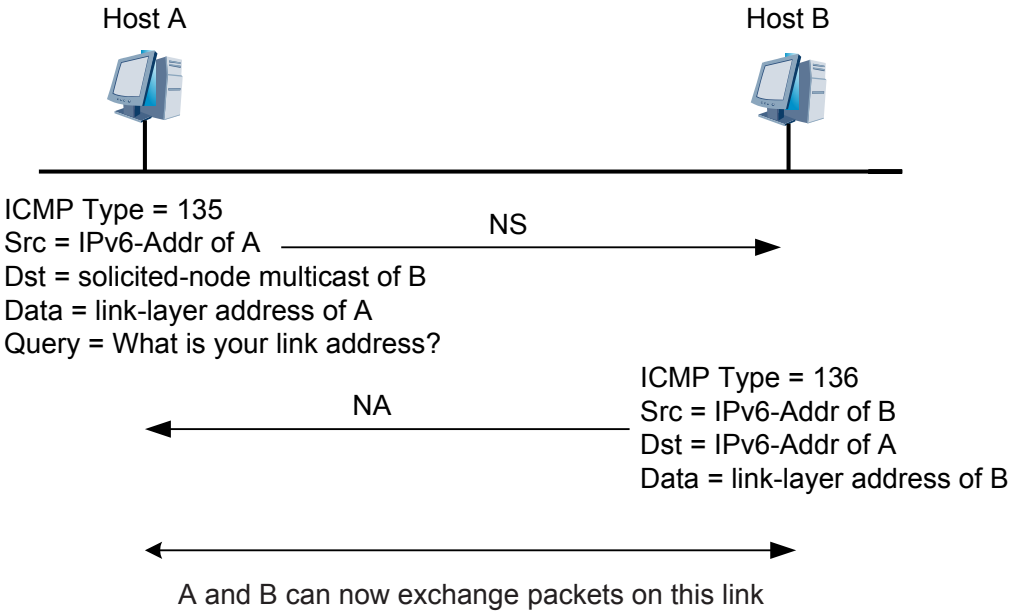
- Layer 3 address resolution enables Layer 2 devices to use the same address resolution protocol.
- Layer 3 security mechanisms are used to prevent address resolution attacks.
- Request packets can be sent in multicast mode, reducing load on Layer 2 networks.

During address resolution, Neighbor Solicitation (NS) packets and Neighbor Advertisement (NA) packets are used.

- In NS packets, the Type field value is 135 and the Code field value is 0. NS packets are similar to IPv4 ARP Request packets.
- In NA packets, the Type field value is 136 and the Code field value is 0. NA packets are similar to IPv4 ARP Reply packets.

Figure 1-10 shows the process of address resolution.

Figure 1-10 IPv6 address resolution



Host A needs to parse the link-layer address of Host B before sending packets to Host B. Host A sends an NS message with its IPv6 address as the source address and the solicited-node multicast address of Host B as the destination address. The Options field in the NS message carries the link-layer address of Host A.

After receiving the NS message, Host B replies with an NA Reply message. In the NA reply message, the source address is the IPv6 address of Host B, and the destination address is the IPv6 address of Host A (the NS message is sent to Host A in unicast mode using the link-layer address of Host A). The Options field carries the link-layer address of Host B. This is the whole address resolution process.

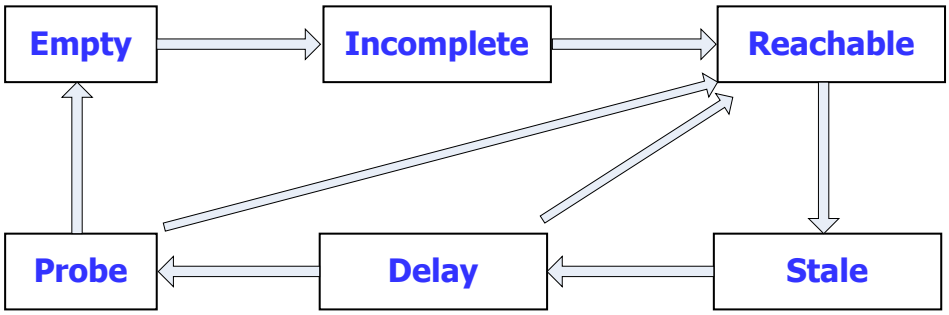
Neighbor Tracking

A neighbor state can transit from one to another. Hardware faults and hot swapping of interface cards interrupt communication with neighboring devices. Communication cannot be restored if the destination of a neighboring device becomes invalid, but it can be restored if the path fails. Nodes need to maintain a neighbor table to monitor the state of each neighboring device.

RFC2461 defines five neighbor states: Incomplete, Reachable, Stale, Delay, and Probe.

Figure 1-11 shows the transition of neighbor states. The Empty state indicates that the neighbor table is empty.

Figure 1-11 Neighbor state transition



The following example describes changes in neighbor state of node A during its first communication with node B.

1. Node A sends an NS message and generates a cache entry. The neighbor state of node A is Incomplete.
2. If node B replies with an NA message, the neighbor state of node A changes from Incomplete to Reachable. Otherwise, the neighbor state changes from Incomplete to Empty after a certain period of time, and node A deletes this entry.
3. After the neighbor reachable time times out, the neighbor state changes from Reachable to Stale, indicating that the neighbor reachable state is unknown.
4. If node A in the Reachable state receives a non-NA Request message from node B, and the link-layer address of node B carried in the message is different from that learned by node A, the neighbor state of node A changes to Stale.
5. After the aging time times out, the neighbor state changes from Stale to Delay.
6. After a period of time (5s by default), the neighbor state changes from Delay to Probe. During this time, if node A receives an NA Reply message, the neighbor state of node A changes to Reachable.
7. Node A in the Probe state sends unicast NS messages at the configured interval (1s by default) for several times (3 by default). If node A receives a Reply message, the neighbor state of node A changes from Probe to Reachable. Otherwise, the state changes to Empty, and node A deletes the entry.

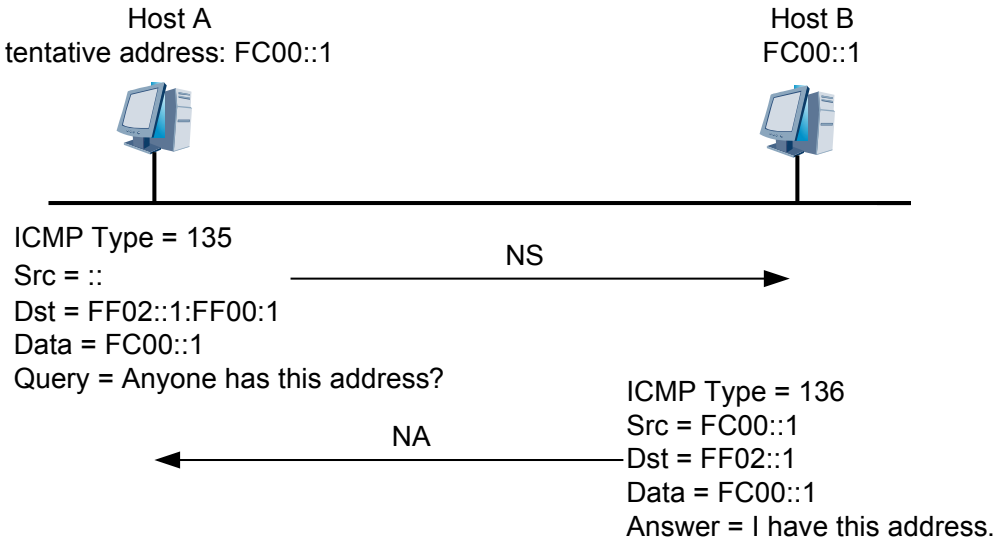
Duplicate Address Detection

Before an IPv6 unicast address is assigned to an interface, duplicate address detection (DAD) is performed to check whether another node uses the address. DAD is required if IP addresses are configured automatically. An IPv6 unicast address assigned to an interface but not verified by DAD is called a tentative address. An interface cannot use the tentative address for unicast communication but will join two multicast groups: ALL-nodes multicast group and Solicited-node multicast group.

IPv6 DAD is similar to IPv4 gratuitous ARP. A node sends an NS message that requests the tentative address as the destination address to the Solicited-node multicast group. If the node receives an NA Reply message, another node is using the tentative address for communication. This node will not use this tentative address for communication.

Figure 1-12 shows an example of DAD.

Figure 1-12 DAD example



The IPv6 address FC00::1 is assigned to Host A as a tentative IPv6 address. To check the validity of this address, Host A sends an NS message containing the requested address FC00::1 to the Solicited-node multicast group to which FC00::1 belongs. Since FC00::1 is not specified, the source address of the NS message is an unspecified address. After receiving the NS message, Host B processes the message in one of the following ways:

- If FC00::1 is a tentative address of Host B, Host B will not use this address as an interface address and will not send an NA message.
- If FC00::1 is in use on Host B, Host B sends an NA message to FF02::1 carrying IP address FC00::1. In this way, Host A can find and mark the duplicate tentative address after receiving the message so it will not take effect.

Router Discovery

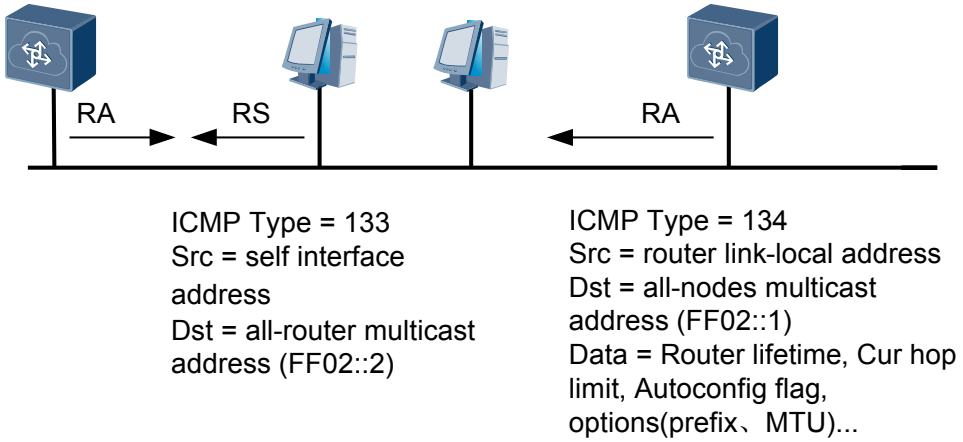
Router discovery is used to locate neighboring devices and learn their address prefixes and configuration parameters for address autoconfiguration.

IPv6 supports stateless address autoconfiguration. Hosts obtain IPv6 prefixes and automatically generate interface IDs. Router Discovery is the basis of IPv6 address autoconfiguration and is implemented through the following two types of packets:

- Router Advertisement (RA) message: Each router periodically sends multicast RA messages carrying network prefixes and identifiers on the network to declare its existence to Layer 2 hosts and devices. An RA message has a Type field value of 134.
- Router Solicitation (RS) message: After being connected to the network, a host immediately sends an RS message to obtain network prefixes. Devices on the network reply with RA messages. An RS message has a Type field value of 133.

Figure 1-13 shows the router discovery function.

Figure 1-13 Router discovery example



Address Autoconfiguration

IPv4 uses DHCP to automatically configure IP addresses and default gateways. This simplifies network management. The length of an IPv6 address is increased to 128 bits. Multiple terminal nodes require the function of automatic configuration. IPv6 allows both stateful and stateless address autoconfiguration. Stateless autoconfiguration enables hosts to automatically generate link-local addresses. Hosts automatically configure global unicast addresses and obtain other information based on prefixes in the RA message.

The process of IPv6 stateless autoconfiguration is as follows:

1. A host automatically configures the link-local address based on the interface ID.
2. The host sends an NS message for duplicate address detection.
3. If address conflict occurs, the host stops address autoconfiguration. Then addresses need to be configured manually.
4. If addresses do not conflict, the link-local address takes effect. The host then connects to the network and communicates with the local node.
5. The host either sends an RS message or receives RA messages devices periodically send.
6. The host obtains the IPv6 address based on the prefixes carried in the RA message and the interface ID.

Default Router Priority and Route Information Discovery

If there are multiple devices on the network where hosts reside, hosts need to select forwarding devices based on the destination address of the packet. In such a case, devices advertise default router priorities and route information, which allows hosts to select the optimal forwarding device based on the packet destination address.

The fields of default router priority and route information are defined in an RA message. These two fields enable hosts to select the optimal forwarding device.

After receiving an RA message containing route information, hosts update their routing tables. When sending packets to other devices, hosts check the routing table and select the optimal route.

When receiving an RA message carrying default router priorities, hosts update their default router lists. When sending packets to other devices, hosts select the device with the highest

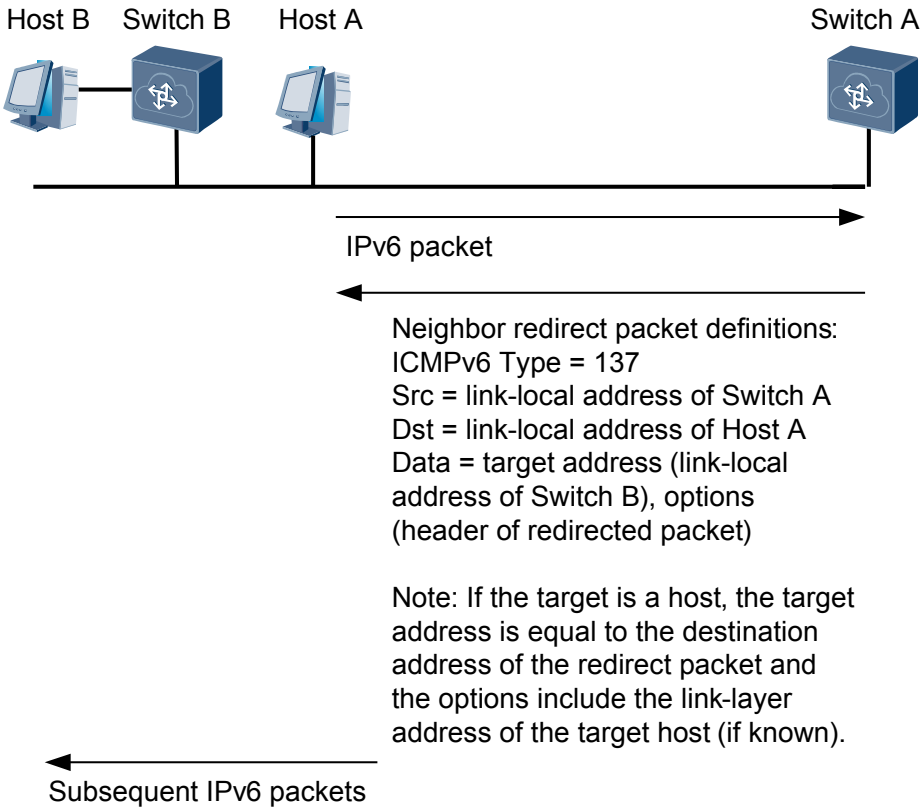
priority to forward packets from the router list. If the selected router does not work, hosts select the subsequent device in descending order of priority.

Redirection

To choose an optimal gateway device, the gateway device sends a Redirection message to notify the sender that another gateway device can send packets. Redirection messages are contained within ICMPv6 messages and have a Type field value of 137. They carry a better next hop address and destination address for packets that need to be redirected.

Figure 1-14 shows an example of packet redirection.

Figure 1-14 Packet redirection example



Host A needs to communicate with Host B. By default, Switch A sends packets from Host A to Host B. After receiving packets from Host A, Switch A discovers that sending packets directly to Switch B is more efficient. Switch A sends a Redirection message carrying the destination address of Host B to Host A to notify Host A that Switch B is a better next hop address. After receiving the Redirection message, Host A adds a host route to the default routing table. Packets sent to Host B will be sent directly to Switch B.

A device sends a Redirection message in the following situations:

- The destination address of the packet is not a multicast address.
- Packets are not forwarded to the device through routing.
- After route calculation, the outbound interface of the next hop is the interface that receives the packets.

- The device discovers that a better next hop IP address of the packet is on the same network segment as the source IP address of the packet.
- After checking the source address of the packet, the device discovers a neighboring device in the neighbor entries using this address as the global unicast address or the link-local unicast address.

1.2.5 IPv6 Secure Neighbor Discovery

In the IPv6 protocol suite, ND is significant in ensuring availability of neighbors on the local link. As network security problems intensify, how to secure ND becomes a concern. RFC 3756 defines several threats to ND security, some of which are described as follows:

Table 1-3 IPv6 ND attacks

Attack Method	Description
NS/NA spoofing	An attacker sends an authorized node (host or switch) an NS message with a bogus source link-layer address option, or an NA message with a bogus target link-layer address option. Then packets from the authorized node are sent to this link-layer address.
Neighbor Unreachability Detection (NUD) failure	An attacker repeatedly sends forged NA messages in response to an authorized node's NUD NS messages so that the authorized node cannot detect the neighbor unreachability. The consequences of this attack depend on why the neighbor became unreachable and how the authorized node would behave if it knew that the neighbor has become unreachable.
Duplicate Address Detection (DAD) attacks	An attacker responds to every DAD attempt made by a host that accesses the network, claiming that the address is already in use. Then the host will never obtain an address.
Spoofed Redirect message	An attacker uses the link-local address of the first-hop switch to send a Redirect message to an authorized host. The authorized host accepts this message because the host mistakenly considers that the message came from the first-hop switch.
Replay attacks	An attacker captures valid messages and replays them. Even if Neighbor Discovery Protocol (NDP) messages are cryptographically protected so that their contents cannot be forged, they are still prone to replay attacks.
Bogus address prefix	An attacker sends a bogus RA message specifying that some prefixes are on-link. If a prefix is on-link, a host will not send any packets that contain this prefix to the switch. Instead, the host will send NS messages to attempt address resolution, but the NS messages are not responded. As a result, the host is denied services.
Malicious last-hop switch	An attacker multicasts bogus RA messages or unicasts bogus RA messages in response to multicast RS messages to a host attempting to discover a last-hop switch. If the host selects the attacker as its default switch, the attacker is able to insert himself as a man-in-the-middle and intercepts all messages exchanged between the host and its destination.

To counter these threats, Secure Neighbor Discovery (SEND), defined in RFC 3971, specifies security mechanisms to extend ND. SEND defines Cryptographically Generated Addresses (CGAs), CGA option, and Rivest Shamir Adleman (RSA) option, which are used to ensure that the sender of an ND message is the owner of the message's source address. SEND also defines Timestamp and Nonce options to prevent replay attacks.

- CGA: A CGA contains network prefix and interface identifier. Interface identifier is generated from a one-way hash of the public key and associated parameters.
- CGA option: contains information used to verify the sender's CGA, including the public key of the sender. CGA is used to authenticate the validity of source IP addresses carried in ND messages.
- RSA option: contains the hash value of the sender's public key and contains the digital signature generated from the sender's private key and ND messages. RSA is used to authenticate the completeness of ND messages and the identity of the ND message sender.

  NOTE

  For an attacker to use an address that belongs to an authorized node, the attacker must use the public key of the authorized node for encryption. Otherwise, the receiver can detect the attempted attack after checking the CGA option. Even if the attacker obtains the public key of the authorized node, the receiver can still detect the attempted attack after checking the digital signature, which is generated from the sender's private key.
- Timestamp option: a 64-bit unsigned integer field containing a timestamp. The value indicates the number of seconds since January 1, 1970, 00:00 UTC. This option protects non-solicit notification messages and Redirect messages and ensures that the timestamp of the recently received message is the latest.
- Nonce option: contains a random number selected by the sender of a solicitation message. This option prevents replay attacks during message exchange. For example, a sender sends an NS message carrying the Nonce option and receives an NA message as a response that also carries the Nonce option; the sender verifies the NA message based on the Nonce option.

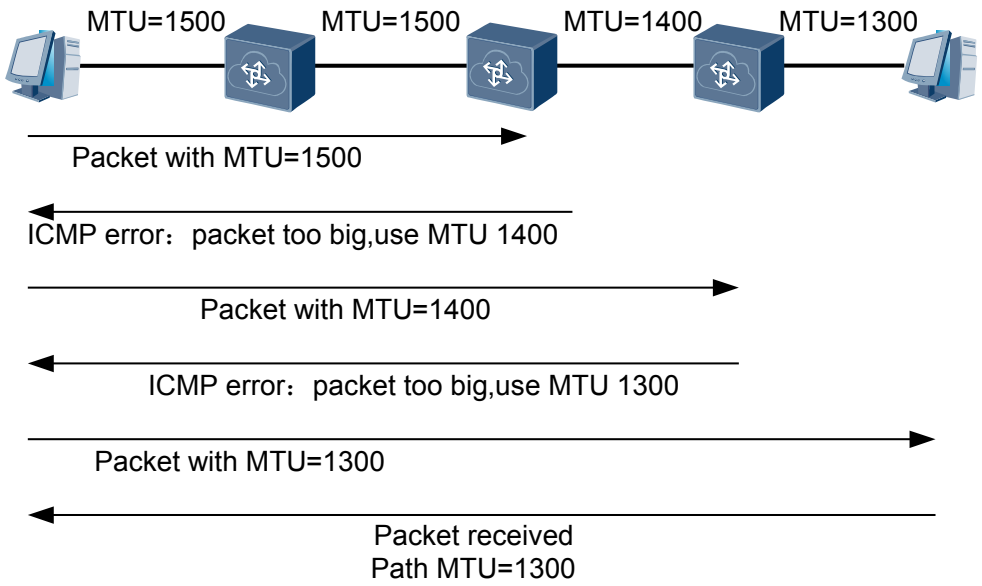
1.2.6 Path MTU

In IPv4, oversized packets are fragmented. When the transit device receives a packet exceeding the maximum transmission unit (MTU) size of its outbound interface from a source node, the transit device fragments the packet before forwarding it to the destination node. In IPv6, however, the source node fragments the packets to reduce pressure on the transit device. When an interface on the transit device receives a packet whose size exceeds the MTU, the transit device discards the packet and sends an ICMPv6 Packet Too Big message to the source node. The ICMPv6 Packet Too Big message contains the MTU value of the outbound interface. The source node fragments the packet based on the MTU and resends the packet, increasing traffic overhead. The Path MTU Discovery (PMTUD) protocol dynamically discovers the MTU value of each link on the transmission path, reducing excessive traffic overhead.

The PMTU protocol is implemented through ICMPv6 Packet Too Big messages. A source node first uses the MTU of its outbound interface as the PMTU and sends a probe packet. If a smaller PMTU exists on the transmission path, the transit device sends a Packet Too Big message to the source node. The Packet Too Big message contains the MTU value of the outbound interface on the transit device. After receiving this message, the source node changes the PMTU value to the received MTU value and sends packets based on the new MTU. This process repeats until packets are sent to the destination address. The source node obtains the PMTU of the destination address.

Figure 1-15 shows an example of PMTU discovery.

Figure 1-15 PMTU discovery



Packets are transmitted through four links with MTU values of 1500, 1500, 1400, and 1300 bytes. Before sending a packet, the source node fragments the packet based on a PMTU of 1500. When the packet is sent to the outbound interface with MTU 1400, the device returns a Packet Too Big message carrying MTU 1400. The source node then fragments the packet based on MTU 1400 and sends the fragmented packet again. The process repeats when the packet based on MTU 1400 is sent to the outbound interface with MTU 1300, the device returns another Packet Too Big message that carries MTU 1300. The source node receives the message and fragments the packet based on MTU 1300. In this way, the source node sends the packet to the destination address and discovers the PMTU of the transmission path.

NOTE

IPv6 allows a minimum MTU of 1280 bytes. Therefore, the PMTU must be greater than 1280 bytes. PMTU of 1500 bytes is recommended.

1.3 Configuration Notes

This section provides the points of attention when configuring IPv6.

Involved Network Elements

Other network elements are not required.

License Support

The IPv6 function is controlled by a license. By default, this function is disabled on a new CE12800 series switches. To use the IPv6 function, apply for and purchase the license from the device supplier.

Version Support

Table 1-4 Products and minimum version supporting IPv6

Series	Product	Minimum Version Required
CE12800	CE12804/CE12808/ CE12812	V100R002C00
	CE12816	V100R003C00
	CE12804S/CE12808S	V100R005C00

Feature Dependencies and Limitations

When deploying IPv6 on the switch, pay attention to the following:

- Each interface can be configured with only one link-local address. To prevent link-local address conflict, automatically generated link-local addresses are recommended. After an interface is configured with an IPv6 global unicast address, it automatically generates a link-local address.
- Manually configured link-local addresses have higher priority than automatically generated ones. Manually configured addresses can overwrite automatically generated ones, but automatically generated addresses cannot overwrite manually configured ones. If manually configured addresses are deleted, the overwritten automatically generated ones take effect.
- When the PMTU from the device to a specified destination node is set, the IPv6 MTU values for interfaces on all intermediate devices cannot be smaller than the configured PMTU value. Otherwise, packets are discarded.
- When both static PMTU and dynamic PMTU are configured, only static PMTU takes effect. Static PMTU entries never age.
- The interface MTU, IPv6 interface MTU, and PMTU are valid only for the packets generated on the device, not for packets forwarded by the host.
- A switch can receive ICMPv6 error packets that are sent from other devices and learns the path MTU according to the received ICMPv6 error packets.

1.4 Default Configuration

Default configuration

Parameter	Default Configuration
Maximum interval for sending RA packets	600s
Minimum interval for sending RA packets	200s
Neighbor reachable time	1200000 ms

1.5 Configuring Basic IPv6

1.5.1 Configuring IPv6 Addresses for Interfaces

Pre-configuration Tasks

Before configuring IPv6 addresses for interfaces, configure link layer protocol parameters for interfaces to ensure that the link layer protocol status on the interfaces is Up.

Configuration Process

You can perform the following configuration tasks in any sequence as required.

Configuring Global Unicast Addresses for Interfaces

Context

A global unicast address is similar to an IPv4 public address and provided for the Internet Service Provider (ISP). A global unicast address can be generated by either of the following methods:

- Generated in EUI-64 format: An IPv6 global unicast address in EUI-64 format contains a manually configured prefix and an automatically generated interface identifier.
- Configured manually: You can manually configure an IPv6 global unicast address.

NOTE

You can configure an interface with multiple global unicast addresses with different network prefixes.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
interface interface-type interface-number
```

The specified interface view is displayed.

Step 3 On an Ethernet interface, run:

```
undo portswitch
```

The interface is switched to Layer 3 mode.

By default, an Ethernet interface works in Layer 2 mode.

If an Ethernet interface already has Layer 2 configuration, this command fails to be executed on the interface. Before running this command on the interface, delete all the Layer 2 configuration of the interface.

NOTE

If many Ethernet interfaces need to be switched to Layer 3 mode, run the **undo portswitch batch** *interface-type* { *interface-number1* [**to** *interface-number2*] } &<1-10> command in the system view to switch these interfaces to Layer 3 mode in batches.

Step 4 Run:

```
ipv6 enable
```

The IPv6 function is enabled on the interface.

By default, the IPv6 function is disabled on an interface.

Step 5 Run either of the following commands to configure an IPv6 global unicast address for an interface:

- Run:

  ```
  ipv6 address { ipv6-address prefix-length | ipv6-address/prefix-length }
  ```

  An IPv6 global unicast address is manually configured.

- Run:

  ```
  ipv6 address { ipv6-address prefix-length | ipv6-address/prefix-length }
  eui-64
  ```

  An IPv6 global unicast address is generated in EUI-64 format.

You can configure a maximum of 16 global unicast addresses on a service interface and a maximum of eight global unicast addresses on a management interface.

Step 6 Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

- Run the **display ipv6 interface** [*interface-type interface-number* | **brief**] command in any view to check IPv6 information about a specified interface.
- Run the **display this ipv6 interface** command in the interface view to check IPv6 information about the current interface.

Configuring Link-local Addresses for Interfaces

Context

Link-local addresses are used in neighbor discovery or stateless autoconfiguration. An IPv6 link-local address is generated by either of the following methods:

- Generated automatically: A device automatically generates a link-local address for an interface based on the link-local prefix (FE80::/10) and link layer address of the interface.
- Configured manually: You can manually configure an IPv6 link-local address for an interface.

NOTE

- Each interface can be configured with only one link-local address. To prevent link-local address conflict, automatically generated link-local addresses are recommended. After an interface is configured with an IPv6 global unicast address, it automatically generates a link-local address.
- Manually configured link-local addresses have a higher priority than automatically generated ones. Manually configured addresses can overwrite automatically generated ones, but automatically generated addresses cannot overwrite manually configured ones. If manually configured addresses are deleted, the automatically generated ones that were previously overwritten take effect again.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
interface interface-type interface-number
```

The specified interface view is displayed.

Step 3 On an Ethernet interface, run:

```
undo portswitch
```

The interface is switched to Layer 3 mode.

By default, an Ethernet interface works in Layer 2 mode.

If an Ethernet interface already has Layer 2 configuration, this command fails to be executed on the interface. Before running this command on the interface, delete all the Layer 2 configuration of the interface.

NOTE

If many Ethernet interfaces need to be switched to Layer 3 mode, run the **undo portswitch batch** *interface-type* { *interface-number1* [**to** *interface-number2*] } &<1-10> command in the system view to switch these interfaces to Layer 3 mode in batches.

Step 4 Run:

```
ipv6 enable
```

The IPv6 function is enabled on the interface.

By default, the IPv6 function is disabled on an interface.

Step 5 Run either of the following commands to configure a link-local address for an interface:

- Run:

  ```
  ipv6 address ipv6-address link-local
  ```

  A link-local address is configured for an interface.

- Run:

  ```
  ipv6 address auto link-local
  ```

  A link-local address is automatically generated.

Step 6 Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

- Run the **display ipv6 interface** [*interface-type interface-number* | **brief**] command in any view to check IPv6 information about a specified interface.
- Run the **display this ipv6 interface** command in the interface view to check IPv6 information about the current interface.

Configuring Anycast Addresses for Interfaces

Context

IPv6 anycast addresses are allocated from the unicast address space. An anycast address identifies a group of network interfaces, which usually belong to different nodes. When using anycast addresses, pay attention to the following points:

- You can use anycast addresses as destination addresses only.
- Packets addressed to an anycast address are delivered to the nearest interface identified by the anycast address, depending on the routing protocols used.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
interface interface-type interface-number
```

The specified interface view is displayed.

Step 3 On an Ethernet interface, run:

```
undo portswitch
```

The interface is switched to Layer 3 mode.

By default, an Ethernet interface works in Layer 2 mode.

If an Ethernet interface already has Layer 2 configuration, this command fails to be executed on the interface. Before running this command on the interface, delete all the Layer 2 configuration of the interface.

NOTE

If many Ethernet interfaces need to be switched to Layer 3 mode, run the **undo portswitch batch** *interface-type* { *interface-number1* [**to** *interface-number2*] } &<1-10> command in the system view to switch these interfaces to Layer 3 mode in batches.

Step 4 Run:

```
ipv6 enable
```

The IPv6 function is enabled on the interface.

By default, the IPv6 function is disabled on an interface.

Step 5 Run:

```
ipv6 address { ipv6-address prefix-length | ipv6-address/prefix-length } anycast
```

An IPv6 anycast address is configured for the interface.

Step 6 Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

- Run the **display ipv6 interface** [*interface-type interface-number* | **brief**] command in any view to check IPv6 information about a specified interface.
- Run the **display this ipv6 interface** command in the interface view to check IPv6 information about the current interface.

1.5.2 Configuring an IPv6 Address Selection Policy Table

If multiple addresses are configured on an interface of the device, the IPv6 address selection policy table can be used to select source and destination addresses for packets.

Pre-configuration Tasks

Before configuring an IPv6 address selection policy table, complete the following tasks:

- Setting link layer protocol parameters for interfaces to ensure that the link layer protocol status of the interfaces is Up

Context

IPv6 addresses can be classified into different types based on different applications.

- Link local addresses and global unicast addresses based on the effective range of the IPv6 addresses
- Temporary addresses and public addresses based on security levels
- Home addresses and care-of addresses based on the application in the mobile IPv6 field
- Physical interface addresses and logical interface addresses based on the interface attributes

The preceding IPv6 addresses can be configured on the same interface of the switch. In this case, the device must select a source address or a destination addresses from multiple addresses on the interface. If the device supports the IPv4/IPv6 dual-stack, it also must select IPv4 addresses or IPv6 addresses for communication. For example, if a domain name maps both an IPv4 address and an IPv6 address, the system must select an address to respond to the DNS request of the client.

An IPv6 address selection policy table solves the preceding problems. It defines a group of address selection rules. The source and destination addresses of packets can be specified or planned based on these rules. This table, similar to a routing table, can be queried by using the longest matching rule. The address is selected based on the source and destination addresses.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
ipv6 address-policy [ vpn-instance vpn-instance-name ] ipv6-address prefix-length
precedence label
```

The source or destination address selection policies are configured.

By default, only default address selection policy entries are contained. These entries are prefixed with ::1, ::, 2002::, FC00::, and ::FFFF:0.0.0.0.

A maximum of 50 address selection policy entries are supported by the system.

- The *label* parameter can be used to determine the result of source address selection. The address whose *label* value is the same as the *label* value of the destination address is selected preferably as the source address.
- The destination address is selected based on both the *label* and the *precedence* parameters. If *label* values of the candidate addresses are the same, the address whose *precedence* value is largest is selected preferably as the destination address.

Step 3 Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

Run the following commands to check the previous configuration.

- Run the **display ipv6 address-policy** [**vpn-instance** *vpn-instance-name*] { **all** | *ipv6-address prefix-length* } command to check address selection policy entries.

1.5.3 Configuring ICMPv6 Packet Control

Context

Configuring ICMPv6 packet control reduces network traffic and prevents malicious attacks. Network congestion may occur when a large number of ICMPv6 error packets are sent on the network within a short period of time. To prevent network congestion, you can limit the maximum number of ICMPv6 error packets sent in a specified period using the token bucket algorithm.

You can set the bucket size and interval for placing tokens into the bucket. The bucket size indicates the maximum number of tokens that a bucket can hold. One token represents one ICMPv6 error packet. When an ICMPv6 error packet is sent, one token is taken out of the token bucket. When there are no tokens, ICMPv6 error packets cannot be sent until new tokens are placed into the token bucket.

If transmission of too many ICMPv6 error packets causes network congestion or the network is attacked by forged ICMPv6 error packets, you can disable the system from sending ICMPv6 error packets, Host Unreachable packets, and Port Unreachable packets.

Pre-configuration Tasks

Before setting rate limit for sending ICMPv6 error packets, complete the following task:

- 1.5.1 Configuring IPv6 Addresses for Interfaces

Procedure

- Control ICMPv6 messages in the system view.
  a. Run:
  ```
  system-view
  ```
  The system view is displayed.
  b. Run the following commands to configure ICMPv6 packet control.
     - Run:
     ```
     ipv6 icmp-error { bucket bucket-size | ratelimit interval } *
     ```
     Rate limit for sending ICMPv6 error packets is set.
     By default, a token bucket can hold a maximum of 10 tokens and the interval for placing tokens into the bucket is 100 ms.
     - Run:
     ```
     ipv6 icmp { icmpv6-type icmpv6-code | icmpv6-name | all } receive disable
     ```
     The system is disabled from receiving ICMPv6 messages.
     By default, the system is enabled to receive ICMPv6 messages.
     - Run:
     ```
     ipv6 icmp { icmpv6-type icmpv6-code | icmpv6-name | all } send disable
     ```
     The system is disabled from sending ICMPv6 messages.
     By default, the system is enabled to send ICMPv6 messages.
     - Run:
     ```
     ipv6 icmp rate-limit packet-too-big disable
     ```
     The device is disabled from rejecting oversized ICMPv6 error messages.
     By default, the device rejects oversized ICMPv6 error messages.
  c. Run:
  ```
  commit
  ```
  The system view is displayed.
- Control ICMPv6 messages in the interface view.
  a. Run:
  ```
  system-view
  ```
  The system view is displayed.
  b. Run:
  ```
  interface interface-type interface-number
  ```
  The specified interface view is displayed.
  c. On an Ethernet interface, run:
  ```
  undo portswitch
  ```
  The interface is switched to Layer 3 mode.
  By default, an Ethernet interface works in Layer 2 mode.

If an Ethernet interface already has Layer 2 configuration, this command fails to be executed on the interface. Before running this command on the interface, delete all the Layer 2 configuration of the interface.

NOTE

If many Ethernet interfaces need to be switched to Layer 3 mode, run the **undo portswitch batch** *interface-type* { *interface-number1* [**to** *interface-number2*] } &<1-10> command in the system view to switch these interfaces to Layer 3 mode in batches.

d. Run:

```
ipv6 enable
```

The IPv6 function is enabled on the interface.

By default, the IPv6 function is disabled on an interface.

e. Run the following commands to configure ICMPv6 packet control.

- Run:

```
ipv6 icmp hop-limit-exceeded send disable
```

The interface is disabled from sending ICMPv6 Hop Limit Exceeded messages.

By default, the function of sending ICMPv6 Hop Limit Exceeded messages configured globally also takes effect on an interface.

- Run:

```
ipv6 icmp host-unreachable send disable
```

The interface is disabled from sending ICMPv6 host-unreachable packets.

By default, the function of sending ICMPv6 host-unreachable messages configured globally also takes effect on an interface.

- Run:

```
ipv6 icmp port-unreachable send disable
```

The interface is disabled from sending ICMPv6 Port Unreachable messages.

By default, the function of sending ICMPv6 Port Unreachable messages configured globally also takes effect on an interface.

f. Run:

```
commit
```

The system view is displayed.

----End

Checking the Configuration

- Run the **display icmpv6 statistics** [**interface** *interface-type interface-number*] command to check ICMPv6 traffic statistics.

1.5.4 Configuring IPv6 Neighbor Discovery

Pre-configuration Tasks

The Neighbor Discovery Protocol (NDP) replaces the Address Resolution Protocol (ARP) and ICMP Router Discovery on an IPv4 network. Additionally, IPv6 Neighbor Discovery (ND) provides redirection and neighbor unreachability detection.

- 1.5.1 Configuring IPv6 Addresses for Interfaces

Configuration Process

You can perform the following configuration tasks in any sequence as required.

Configuring Static Neighbors

Context

To communicate with a destination host, a host needs to obtain the destination host's link-layer address. The link-layer address of a neighbor node can be obtained using the neighbor discovery mechanism, or by manually configuring static neighbor entries. A device identifies a static neighbor entry based on the IPv6 address of this neighbor and number of the Layer 3 interface connected to this neighbor. To filter invalid packets, you can create static neighbor entries, binding the destination IPv6 addresses of these packets to nonexistent MAC addresses.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
interface interface-type interface-number
```

The specified interface view is displayed.

Step 3 On an Ethernet interface, run:

```
undo portswitch
```

The interface is switched to Layer 3 mode.

By default, an Ethernet interface works in Layer 2 mode.

If an Ethernet interface already has Layer 2 configuration, this command fails to be executed on the interface. Before running this command on the interface, delete all the Layer 2 configuration of the interface.

NOTE

If many Ethernet interfaces need to be switched to Layer 3 mode, run the **undo portswitch batch** *interface-type* { *interface-number1* [**to** *interface-number2*] } &<1-10> command in the system view to switch these interfaces to Layer 3 mode in batches.

Step 4 Run:

```
ipv6 enable
```

The IPv6 function is enabled.

By default, the IPv6 function is disabled on an interface.

Step 5 Run the following commands to configure static neighbors based on the interface type.

- For a VLANIF interface, run the **ipv6 neighbor** *ipv6-address mac-address* **vlan** *vlan-id interface-type interface-number* command.

- For a GE interface, 10GE interface, 40GE interface, 100GE, or an Eth-Trunk interface, run the **ipv6 neighbor** *ipv6-address mac-address* command.
- For a GE sub-interface, 10GE sub-interface, 40GE sub-interface, 100GE sub-interface, Eth-Trunk sub-interface, run the **ipv6 neighbor** *ipv6-address mac-address* [**vid** *vlan-id*] command.

By default, no static neighbor entry is configured.

Step 6 Run:

```
commit
```

The configuration is committed.

----End

Configuring Neighbor Discovery

Context

IPv6 NDP provides address resolution, neighbor unreachability detection, DAD, router/prefix discovery, address autoconfiguration, and redirection.

NOTE

After the IPv6 function is enabled on the switch, the switch automatically implements address resolution, DAD, and redirection. Neighbor unreachability detection, router/prefix discovery, and address autoconfiguration need to be manually configured. You can also configure the switch to send RA packets to enable router/prefix discovery and address autoconfiguration, and enable the automatic detection of ND entries to check whether neighbors are reachable.

After the automatic detection of ND entries is enabled on the switch, the switch can send NS packets to check whether neighbors are reachable before aging ND entries. If neighbors are reachable, the switch updates ND entries; otherwise, the switch ages ND entries.

You can enable the switch to send RA packets. After receiving the RA packets, network nodes perform address autoconfiguration and router/prefix discovery based on the prefix and other configuration information contained in RA packets.

After the preceding configurations are complete, NDP functions work properly. You can also adjust ND parameters based on service requirements.

Procedure

Step 1 Run the following commands to enable NDP functions to work properly.

1. Run:

   ```
   system-view
   ```

   The system view is displayed.

2. Run:

   ```
   undo ipv6 nd auto-detect disable
   ```

   Automatic detection of ND entries is enabled.

   By default, automatic detection of ND entries is enabled.

3. Run:

   ```
   interface interface-type interface-number
   ```

   The specified interface view is displayed.

4. On an Ethernet interface, run:

   ```
   undo portswitch
   ```

   The interface is switched to Layer 3 mode.

   By default, an Ethernet interface works in Layer 2 mode.

   The command fails if any Layer 2 configuration exists on the interface. In such a case, clear the Layer 2 configurations on the interface before running the **undo portswitch** command.

   NOTE

   You can run the **undo portswitch batch** *interface-type* { *interface-number1* [**to** *interface-number2*] } &<1-10> command in the system view to switch the working mode of multiple Ethernet interfaces in batches.

5. Run:

   ```
   ipv6 enable
   ```

   The IPv6 function is enabled.

   By default, the IPv6 function is disabled on an interface.

6. Run:

   ```
   ipv6 nd ra halt disable
   ```

   The system is enabled to send RA packets.

   By default, the system is disabled from sending RA packets.

7. Run:

   ```
   commit
   ```

   The configuration is committed.

Step 2 (Optional) After completing the preceding steps, adjust ND parameters to meet service requirements.

Run the following commands in the system view.

Run:

```
quit
```

Return to the system view.

- In the system view, run:

  ```
  ipv6 nd hop-limit limit
  ```

  The hop limit for IPv6 unicast packets initially sent by a device is set.

  By default, the IPv6 unicast packets initially sent by a device can travel a maximum of 64 hops.

- In the system view, run:

  ```
  ipv6 nd pre-detect
  ```

  Pre-detection of ND entries is enabled.

  By default, pre-detection of ND entries is disabled, which is recommended.

- In the system view, run:

  ```
  ipv6 nd stale-timeout timeout-value
  ```

  The aging time of ND entries in STALE state is set.

  By default, the aging time of ND entries in STALE state is 1200 seconds.

Run the following commands on the interface.

Run:

```
interface interface-type interface-number
```

The specified interface view is displayed.

- Run:

  ```
  ipv6 nd stale-timeout timeout-value
  ```

  The aging time of ND entries in STALE state is set.

  By default, the aging time of ND entries in STALE state is 1200 seconds.

- Run:

  ```
  ipv6 nd ra hop-limit limit
  ```

  The hop limit for RA packets is set.

  By default, the hop limit for RA packets is 64.

- Run:

  ```
  ipv6 nd ns retrans-timer interval
  ```

  The interval for sending NS packets is set.

  By default, the interval for sending NS packets is 1000 ms.

- Run:

  ```
  ipv6 nd ra { max-interval maximum-interval | min-interval minimum-interval }
  ```

  The interval for sending RA packets is set.

  By default, the maximum interval is 600s and the minimum interval is 200s.

- Run:

  ```
  ipv6 nd ra prefix { ipv6-address prefix-length | ipv6-address/prefix-length }
  valid-lifetime preferred-lifetime [ no-autoconfig ] [ off-link ] or ipv6 nd
  ra prefix default no-advertise
  ```

  Prefix information in RA packets is configured.

  By default, an RA packet carries only the address prefix configured using the **ipv6 address** command.

- Run:

  ```
  ipv6 nd autoconfig managed-address-flag
  ```

  The managed address configuration flag (M flag) for stateful autoconfiguration in RA packets is set.

  By default, the M flag in an RA packet is not set.

- Run:

  ```
  ipv6 nd autoconfig other-flag
  ```

  The other configuration flag (O flag) for stateful autoconfiguration in RA packets is set.

  By default, the O flag in an RA packet is not set.

- Run:

  ```
  ipv6 nd nud reachable-time value
  ```

  The IPv6 neighbor reachable time is set.

  By default, the IPv6 neighbor reachable time is 1200000 ms.

- Run:

  ```
  ipv6 nd ra router-lifetime ra-lifetime
  ```

  The time to live (TTL) is set for RA packets.

  By default, the TTL of RA packets is 1800s.

- Run:

  ```
  ipv6 nd ra preference { high | medium | low }
  ```

  The priority of the default router in RA packets is set.

  By default, the priority of the default router in an RA packet is medium.

- Run:

  ```
  ipv6 nd ra route-information ipv6-address prefix-length lifetime route-lifetime [ preference { high | medium | low } ]
  ```

  Route option information in RA packets is set.

  By default, no route option information is configured in RA packets.

- Run:

  ```
  ipv6 nd dad attempts value
  ```

  The number of times NS packets are sent when the system performs DAD is set.

  By default, the number of times NS packets are sent when the system performs DAD is 1.

- Run:

  ```
  ipv6 nd ra advertised-mtu disable
  ```

  The device is configured not to contain the MTU option in RA packets.

  By default, RA packets contain the MTU option.

Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

Procedure

- Run the **display ipv6 interface** [*interface-type interface-number* | **brief**] command to check IPv6 information about a specified interface.
- Run the **display ipv6 neighbors** [*ipv6-address* | [**vlan** *vlan-id*] *interface-type interface-number* | **vpn-instance** *vpn-instance-name*] command to check information about neighbor entries.

1.5.5 Configuring IPv6 SEND

The SEcure Neighbor Discovery (SEND) protocol is a security extension of the Neighbor Discovery Protocol (NDP) in IPv6.

Pre-configuration Tasks

Before configuring IPv6 SEND, complete the following tasks:

- **1.5.4 Configuring IPv6 Neighbor Discovery**

Configuration Process

You can perform the following configuration tasks in sequence.

Configuring a CGA IPv6 Address

Context

To enable IPv6 SEND to protect ND messages that carry CGA and RSA options, you need to configure a CGA IPv6 address on an interface that sends ND messages. After receiving the packet, the peer device uses the CGA option to authenticate the validity of source IP addresses carried in ND messages and the RSA option to authenticate the completeness of ND messages.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
rsa key-pair label label-name [ modulus modulus-bits ]
```

An RSA key pair is created.

Step 3 Run:

```
interface interface-type interface-number
```

The view of the interface where a CGA IPv6 address needs to be configured is displayed.

Step 4 On an Ethernet interface, run:

```
undo portswitch
```

The interface is switched to Layer 3 mode.

By default, an Ethernet interface works in Layer 2 mode.

If an Ethernet interface already has Layer 2 configuration, this command fails to be executed on the interface. Before running this command on the interface, delete all the Layer 2 configuration of the interface.

NOTE

If many Ethernet interfaces need to be switched to Layer 3 mode, run the **undo portswitch batch** *interface-type* { *interface-number1* [**to** *interface-number2*] } &<1-10> command in the system view to switch these interfaces to Layer 3 mode in batches.

Step 5 Run:

```
ipv6 enable
```

The IPv6 function is enabled on the interface.

By default, the IPv6 function is disabled on an interface.

Step 6 Run:

```
ipv6 security rsakey-pair label-name
```

The RSA key pair is bound to the interface to generate a CGA address.

By default, an RSA key pair is not bound to an interface.

The RSA key pair is created using the **rsa key-pair label** *label-name* [**modulus** *modulus-bits*] command in step 2.

Step 7 Run:

```
ipv6 security modifier sec-level sec-value [ modifier-value ]
```

The modifier value and security level are configured for the CGA address.

By default, no modifier value or security level is set for a CGA address.

The modifier value can be manually configured only when the security level of the CGA address is 0.

Step 8 Configuring a CGA IPv6 Address.Run the following commands as required. You can configure both the CGA link local address and global unicast address or one of them.

- Run:

  ```
  ipv6 address ipv6-address link-local cga
  ```

  A CGA IPv6 address is configured.

  By default, no link-local address is configured for an interface.

- Run:

  ```
  ipv6 address { ipv6-address prefix-length | ipv6-address/prefix-length } cga
  ```

  A CGA global unicast address is configured.

  By default, no CGA global unicast addresses is configured.

Step 9 Run:

```
commit
```

The configuration is committed.

----End

Enable IPv6 SEND

Context

When IPv6 SEND is enabled, that is the strict security mode is enabled on an interface, the interface regards the received ND message insecure and discards it in the following cases:

- The received ND message does not carry a CGA or RSA option. That is, the interface that sent the ND message does not have a CGA address.
- The rate of processing the received ND message exceeds the rate limit of the system.
- The key length in the received ND message is out of the length range allowed on the interface.
- The difference between the receive time and the send time of the ND message is out of the time range allowed on the interface.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 (Optional) Run:

```
ipv6 nd security rate-limit ratelimit-value
```

The rate limit for processing received ND messages is set.

By default, no rate limit for the system to process received ND messages is set.

Step 3 Run:

```
interface interface-type interface-number
```

The interface view is displayed.

Step 4 (Optional) Run:

```
ipv6 nd security key-length { minimum keylen-value | maximum keylen-value } *
```

The key length allowed on the interface is set.

By default, the minimum key length is 512 bits and the maximum key length is 2048 bits.

Step 5 (Optional) Run:

```
ipv6 nd security timestamp { delta delta-value | drift drift-value | fuzz-factor fuzz-value } *
```

The timestamp configuration parameters are set.

By default, the maximum difference between the receive time and send time of an ND message is 300 seconds; the maximum difference between the system time of the sender and the system time of the receiver is 1%; the maximum alive time of an ND message is 1 second.

Step 6 Run:

```
ipv6 nd security strict
```

The strict security mode is enabled on the interface.

By default, the strict security mode is not enabled on an interface.

Step 7 Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

Procedure

- Run the **display ipv6 security interface** *interface-type interface-number* command to check the IPv6 SEND configurations.

----End

1.5.6 Configuring PMTU

When the device functions as the source node and sends IPv6 packets to the destination node, the device fragments packets based on PMTU. The intermediate device does not need to fragment packets, reducing the burden of the intermediate device to effectively use network resources and obtain the maximum throughput.

Pre-configuration Tasks

- **1.5.1 Configuring IPv6 Addresses for Interfaces**

Configuration Process

You can perform the following configuration tasks in any sequence as required.

Configuring Static PMTU

Context

Generally, PMTU is dynamically negotiated according to the IPv6 MTU value of an interface. In special situations, to protect devices on the network and avoid attacks from large-sized packets, you can manually configure the PMTU to a specified destination node to control the maximum length of packets forwarded from the device to the destination node.

NOTE

When the PMTU from the device to a specified destination node is set, the IPv6 MTU values for interfaces on all intermediate devices cannot be smaller than the configured PMTU value. Otherwise, packets are discarded.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 (Optional) Configure the IPv6 MTU for an interface.

1. Run:

   ```
   interface interface-type interface-number
   ```

   The specified interface view is displayed.

2. (For Ethernet interfaces) Run:

   ```
   undo portswitch
   ```

   The interface is switched to Layer 3 mode.

   By default, an Ethernet interface works in Layer 2 mode.

   The command fails if any Layer 2 configuration exists on the interface. In such a case, clear the Layer 2 configurations on the interface before running the **undo portswitch** command.

   NOTE

   You can run the **undo portswitch batch** *interface-type* { *interface-number1* [**to** *interface-number2*] } &<1-10> command in the system view to switch the working mode of multiple Ethernet interfaces in batches.

3. Run:

   ```
   ipv6 enable
   ```

   The IPv6 function is enabled on the interface.

   By default, the IPv6 function is disabled on an interface.

4. Run:

   ```
   ipv6 mtu mtu
   ```

   The MTU of IPv6 packets on an interface is set.

   By default, the MTU of IPv6 packets on an interface is 1500 bytes.

NOTE

After the MTU value is changed, run the **shutdown** and **undo shutdown** commands or the **restart (interface view)** command to restart the interface and allow the changes to take effect.

5. Run:

```
quit
```

Return to the system view.

Step 3 Run:

```
ipv6 pathmtu ipv6-address [ vpn-instance vpn-instance-name ] [ path-mtu ]
```

The PMTU for a specified IPv6 address is set.

Step 4 Run:

```
commit
```

The configuration is committed.

----End

Setting the Aging Time of Dynamic PMTU

Context

When the device functions as a source node and sends packets to a destination node, the device dynamically negotiates the PMTU with the destination node according to the IPv6 MTU values of interfaces and fragments packets based on the PMTU. After the PMTU ages out, the dynamic PMTU is deleted. The source node dynamically renegotiates the PMTU with the destination node.

NOTE

When both static and dynamic PMTUs are configured, only static PMTU takes effect. Static PMTU entries never age.

The interface MTU, IPv6 interface MTU, and PMTU are valid only for packets generated on the device, not for packets forwarded by the host.

A switch can receive ICMPv6 error packets that are sent from other devices and learns the path MTU according to the received ICMPv6 error packets.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 (Optional) Configure the IPv6 MTU for an interface.

1. Run:

```
interface interface-type interface-number
```

The specified interface view is displayed.

2. (For Ethernet interfaces) Run:

```
undo portswitch
```

The interface is switched to Layer 3 mode.

By default, an Ethernet interface works in Layer 2 mode.

The command fails if any Layer 2 configuration exists on the interface. In such a case, clear the Layer 2 configurations on the interface before running the **undo portswitch** command.

NOTE

You can run the **undo portswitch batch** *interface-type* { *interface-number1* [**to** *interface-number2*] } &<1-10> command in the system view to switch the working mode of multiple Ethernet interfaces in batches.

3. Run:

```
ipv6 enable
```

The IPv6 function is enabled on the interface.

By default, the IPv6 function is disabled on an interface.

4. Run:

```
ipv6 mtu mtu
```

The MTU of IPv6 packets on an interface is set.

By default, the MTU of IPv6 packets on an interface is 1500 bytes.

NOTE

After the MTU value is changed, run the **shutdown** and **undo shutdown** commands or the **restart (interface view)** command to restart the interface and allow the changes to take effect.

5. Run:

```
quit
```

The system view is displayed.

Step 3 Run:

```
ipv6 pathmtu age age-time
```

The aging time is set for dynamic PMTU entries.

By default, the aging time of dynamic PMTU entries is 10 minutes.

Step 4 Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

Procedure

- Run the **display ipv6 pathmtu** [**vpn-instance** *vpn-instance-name*]{ *ipv6-address* | **all** | **dynamic** | **static** } command to check all PMTU entries.
- Run the **display ipv6 interface** [*interface-type interface-number* | **brief**] command to check IPv6 information about a specified interface.

1.5.7 Configuring TCP6

Pre-configuration Tasks

Before configuring TCP6, configure link layer protocol parameters for interfaces to ensure that the link layer protocol status on the interfaces is Up.

Configuration Process

You can perform the following configuration tasks in any sequence as required.

Setting TCP6 Timers

Context

The following TCP6 timers need to be set:

- SYN-Wait timer: When SYN packets are sent, the SYN-Wait timer starts. If no response packet is received after the SYN-Wait timer expires, the TCP6 connection is terminated.
- FIN-Wait timer: When the TCP connection status changes from FIN_WAIT_1 to FIN_WAIT_2, the FIN-Wait timer starts. If no response packet is received after the FIN-Wait timer expires, the TCP6 connection is terminated.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
tcp ipv6 timer syn-timeout interval
```

The SYN-Wait timer is set for TCP6 connections.

By default, the SYN-Wait timer is set to 75s.

Step 3 Run:

```
tcp ipv6 timer fin-timeout interval
```

The FIN-Wait timer is set for TCP6 connections.

By default, the FIN_Wait timer is set to 675s.

Step 4 Run:

```
commit
```

The configuration is committed.

----End

Setting the TCP6 Sliding Window Size

Context

You can set a TCP6 sliding window size to improve network performance. The sliding window size indicates the receive or send buffer size of a TCP6 socket.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
tcp ipv6 window window-size
```

The receive or send buffer size of a TCP6 socket is set.

By default, the receive or send buffer size of a TCP6 socket is 8 KB.

Step 3 Run:

```
commit
```

The configuration is committed.

----End

Setting the MSS Value for a TCP6 Connection

Context

Setting the maximum value of Maximum Segment Size (MSS) for a TCP6 connection defines the largest TCP6 packet size, allowing TCP6 packets to be successfully forwarded by intermediate devices when no Path MTU is available.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
tcp ipv6 max-mss mss-value
```

The maximum MSS value is set for a TCP6 connection.

By default, the maximum MSS value is not configured for TCP6 connections.

Step 3 Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

Procedure

- Run the **display tcp ipv6 status** [**local-ip** *local-ip*] [**local-port** *local-port*] [**remote-ip** *remote-ip*] [**remote-port** *remote-port*] [**cid** *cid*] [**socket-id** *socket-id*] command to check the status of all IPv6 TCP connections.

----End

1.6 Maintaining IPv6

1.6.1 Clearing IPv6 Statistics

Context

NOTICE

IPv6 statistics cannot be restored after being cleared. Therefore, exercise caution before clearing IPv6 statistics.

Procedure

- Run the **reset rawip ipv6 statistics** command in the user view to clear all Raw IPv6 packet statistics.
- Run the **reset ipv6 statistics** command in the user view to clear IPv6 traffic statistics.
- Run the **reset tcp ipv6 statistics** command in the user view to clear TCP6 statistics.
- Run the **reset udp ipv6 statistics** command in the user view to clear UDP6 statistics.
- Run the **reset ipv6 pathmtu** [**vpn-instance** *vpn-instance-name*] **dynamic** command in the user view to clear dynamic PMTU entries.
- Run the **reset ipv6 neighbors** { **dynamic** | **vlan** *vlan-id* [*interface-type interface-number*] | *interface-type interface-number* } command in the user view to clear dynamic IPv6 neighbor entries.
- Run the **reset ipv6 nd security nonce** *interface-type interface-number* command in the user view to clear the Nonce value of an IPv6 SEND message on a specified interface.
- Run the **reset ipv6 nd security statistics** *interface-type interface-number* command in the user view to clear statistics on IPv6 SEND messages on a specified interface.
- Run the **reset ipv6 nd security timestamp** *interface-type interface-number* command in the user view to clear the timestamp of an IPv6 SEND message on a specified interface.

----End

1.6.2 Monitoring IPv6 Running Status

Context

You can run the following commands in any view to check IPv6 running status.

Procedure

- Run the **display ipv6 interface** [*interface-type interface-number* | **brief**] command to check IPv6 information about a specified interface.
- Run the **display ipv6 statistics** [**interface** *interface-type interface-number*] command to check IPv6 traffic statistics.

- Run the **display icmpv6 statistics** [**interface** *interface-type interface-number*] command to check ICMPv6 traffic statistics.
- Run the **display rawip ipv6 statistics** command to check Raw IPv6 packet statistics.
- Run the **display tcp ipv6 status** [**local-ip** *local-ip*] [**local-port** *local-port*] [**remote-ip** *remote-ip*] [**remote-port** *remote-port*] [**cid** *cid*] [**socket-id** *socket-id*] command to check the status of all IPv6 TCP connections.
- Run the **display tcp ipv6 statistics** command to check IPv6 TCP traffic statistics.
- Run the **display udp ipv6 statistics** command to check IPv6 UDP packet statistics.
- Run the **display udp ipv6 status** [**local-ip** *local-ip*] [**local-port** *local-port*] [**remote-ip** *remote-ip*] [**remote-port** *remote-port*] [**cid** *cid*] [**socket-id** *socket-id*] command to check information about IPv6 UDP connections.
- Run the **display ipv6 neighbors** [*ipv6-address* | [**vlan** *vlan-id*] *interface-type interface-number* | **vpn-instance** *vpn-instance-name*] command to check neighbor entries.
- Run the **display ipv6 socket** [**socket-type** *socket-type*] [**cid** *cid*] [**socket-id** *socket-id*] command to check information about IPv6 sockets.
- Run the **display ipv6 pathmtu** [**vpn-instance** *vpn-instance-name*] { *ipv6-address* | **all** | **dynamic** | **static** } command to check all PMTU entries.
- Run the **display ipv6 nd security nonce** *interface-type interface-number* command to check the Nonce value of the current secure ND transaction.
- Run the **display ipv6 nd security statistics** *interface-type interface-number* command to check the statistics on IPv6 SEND messages.
- Run the **display ipv6 nd security timestamp** *interface-type interface-number* command to check the time stamp value of the last received and accepted SEND message (RDlast) and the local time at which the last SEND message for this peer is accepted (TSlast).

----End

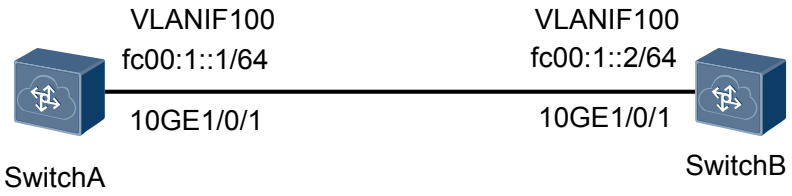
1.7 Configuration Examples

1.7.1 Example for Configuring Basic IPv6 Functions

Networking Requirements

As shown in **Figure 1-16**, 10GE1/0/1 of SwitchA connects to 10GE1/0/1 of SwitchB. The two interfaces belong to VLAN 100. An IPv6 global unicast address is configured for VLANIF 100 to allow SwitchA to communicate with SwitchB.

Figure 1-16 Networking diagram for configuring basic IPv6 functions



Configuration Roadmap

The configuration roadmap is as follows:

1. Enable the IPv6 function on switch interfaces.
2. Configure global unicast IPv6 addresses for the interfaces.

Procedure

Step 1 Create a VLAN and add interfaces to the VLAN.

Configure SwitchA.

```
<HUAWEI> system-view
[~HUAWEI] sysname SwitchA
[*HUAWEI] commit
[~SwitchA] vlan batch 100
[*SwitchA] interface 10ge 1/0/1
[*SwitchA-10GE1/0/1] port link-type trunk
[*SwitchA-10GE1/0/1] port trunk allow-pass vlan 100
[*SwitchA-10GE1/0/1] quit
[*SwitchA] commit
```

Configure SwitchB.

```
<HUAWEI> system-view
[~HUAWEI] sysname SwitchB
[*HUAWEI] commit
[~SwitchB] vlan batch 100
[*SwitchB] interface 10ge 1/0/1
[*SwitchB-10GE1/0/1] port link-type trunk
[*SwitchB-10GE1/0/1] port trunk allow-pass vlan 100
[*SwitchB-10GE1/0/1] quit
[*SwitchB] commit
```

Step 2 Enable the IPv6 function on interfaces and configure global unicast addresses for the interfaces.

Configure SwitchA.

```
[~SwitchA] interface vlanif 100
[*SwitchA-Vlanif100] ipv6 enable
[*SwitchA-Vlanif100] ipv6 address fc00:1::1/64
[*SwitchA-Vlanif100] commit
[~SwitchA-Vlanif100] quit
[~SwitchA] quit
```

Configure SwitchB.

```
[~SwitchB] interface vlanif 100
[*SwitchB-Vlanif100] ipv6 enable
[*SwitchB-Vlanif100] ipv6 address fc00:1::2/64
[*SwitchB-Vlanif100] commit
[~SwitchB-Vlanif100] quit
[~SwitchB] quit
```

Step 3 Verify the configuration.

If the preceding configurations are successful, you can view the configured global unicast addresses. The interface status and the IPv6 protocol are Up. You can also check the neighbor of the interfaces.

Check interface information on SwitchA.

```
<SwitchA> display ipv6 interface vlanif 100
Vlanif100 current state : UP
```

```
IPv6 protocol current state : UP
link-local address is FE80::C964:0:B8B6:1
  Global unicast address(es):
    FC00:1::1, subnet is FC00:1::/64
  Joined group address(es):
    FF02::1:FF00:1
    FF02::1:FFB6:1
    FF02::2
    FF02::1
  MTU is 1500 bytes
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 1200000 milliseconds
  ND retransmit interval is 1000 milliseconds
  Hosts use stateless autoconfig for addresses
```

Check interface information on SwitchB.

```
<SwitchB> display ipv6 interface vlanif 100
Vlanif100 current state : UP
IPv6 protocol current state : UP
link-local address is FE80::2D6F:0:7AF3:1
  Global unicast address(es):
    FC00:1::2, subnet is FC00:1::/64
  Joined group address(es):
    FF02::1:FF00:2
    FF02::1:FFF3:1
    FF02::2
    FF02::1
  MTU is 1500 bytes
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 1200000 milliseconds
  ND retransmit interval is 1000 milliseconds
  Hosts use stateless autoconfig for addresses
```

Ping the link-local address of SwitchB from SwitchA. You must use the parameter **-i** to specify the interface with the link-local address.

```
<SwitchA> ping ipv6 FE80::2D6F:0:7AF3:1 -i vlanif 100
  PING FE80::2D6F:0:7AF3:1 : 56  data bytes, press CTRL_C to break
    Reply from FE80::2D6F:0:7AF3:1
    bytes=56 Sequence=1 hop limit=64  time = 7 ms
    Reply from FE80::2D6F:0:7AF3:1
    bytes=56 Sequence=2 hop limit=64  time = 3 ms
    Reply from FE80::2D6F:0:7AF3:1
    bytes=56 Sequence=3 hop limit=64  time = 3 ms
    Reply from FE80::2D6F:0:7AF3:1
    bytes=56 Sequence=4 hop limit=64  time = 3 ms
    Reply from FE80::2D6F:0:7AF3:1
    bytes=56 Sequence=5 hop limit=64  time = 3 ms

  --- FE80::2D6F:0:7AF3:1 ping statistics ---
    5 packet(s) transmitted
    5 packet(s) received
    0.00% packet loss
    round-trip min/avg/max = 3/3/7 ms
```

Ping the global unicast IPv6 address of SwitchB from SwitchA. The global unicast IPv6 address can be pinged successfully.

```
<SwitchA> ping ipv6 fc00:1::2
  PING FC00:1::2 : 56  data bytes, press CTRL_C to break
    Reply from FC00:1::2
    bytes=56 Sequence=1 hop limit=64  time = 12 ms
    Reply from FC00:1::2
    bytes=56 Sequence=2 hop limit=64  time = 3 ms
    Reply from FC00:1::2
    bytes=56 Sequence=3 hop limit=64  time = 3 ms
    Reply from FC00:1::2
    bytes=56 Sequence=4 hop limit=64  time = 3 ms
```

```
  Reply from FC00:1::2
  bytes=56 Sequence=5 hop limit=64  time = 3 ms

--- FC00:1::2 ping statistics ---
  5 packet(s) transmitted
  5 packet(s) received
  0.00% packet loss
  round-trip min/avg/max = 3/4/12 ms
```

----End

Configuration File

- SwitchA configuration file

```
#
sysname SwitchA
#
vlan batch 100
#
interface Vlanif100
 ipv6 enable
 ipv6 address FC00:1::1/64
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 100
#
return
```

- SwitchB configuration file

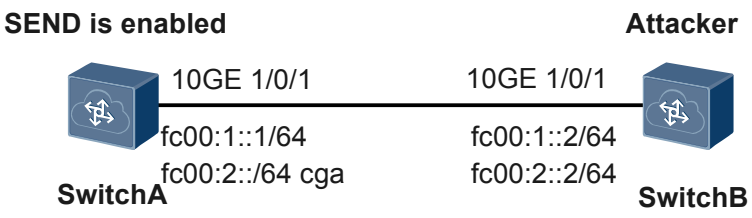
```
#
sysname SwitchB
#
vlan batch 100
#
interface Vlanif100
 ipv6 enable
 ipv6 address FC00:1::2/64
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 100
#
return
```

1.7.2 Example for Configuring IPv6 SEND

Networking Requirements

As shown in **Figure 1-17**, IPv6 SEND is configured on Switch A to ensure its security. When a network device not enabled with IPv6 SEND, such as Switch B, sends messages to Switch A, Switch A regards them invalid and discards them.

Figure 1-17 Networking diagram for configuring IPv6 SEND



Configuration Roadmap

The configuration roadmap is as follows:

1. Configure a CGA IPv6 address and a common IPv6 address on Switch A.
2. Enable the strict security mode on an interface of Switch A.
3. Configure an IPv6 address for an interface on Switch B.

Procedure

Step 1 Configure a CGA IPv6 address on Switch A.

```
<HUAWEI> system-view
[~HUAWEI] sysname SwitchA
[*HUAWEI] commit
[~SwitchA] rsa key-pair label huawei
[*SwitchA] interface 10ge 1/0/1
[*SwitchA-10GE1/0/1] undo portswitch
[*SwitchA-10GE1/0/1] ipv6 enable
[*SwitchA-10GE1/0/1] ipv6 security rsakey-pair huawei
[*SwitchA-10GE1/0/1] ipv6 security modifier sec-level 1
[*SwitchA-10GE1/0/1] ipv6 address fe80::3 link-local cga
[*SwitchA-10GE1/0/1] ipv6 address fc00:2::/64 cga
[*SwitchA-10GE1/0/1] ipv6 address fc00:1::1/64
```

Step 2 Enable the strict security mode on an interface of Switch A.

```
[*SwitchA-10GE1/0/1] ipv6 nd security strict
[*SwitchA-10GE1/0/1] commit
```

Step 3 Configure an IPv6 address of Switch B.

```
<HUAWEI> system-view
[~HUAWEI] sysname SwitchB
[*HUAWEI] commit
[~SwitchB] interface 10ge 1/0/1
[~SwitchB-10GE1/0/1] undo portswitch
[*SwitchB-10GE1/0/1] ipv6 enable
[*SwitchB-10GE1/0/1] ipv6 address auto link-local
[*SwitchB-10GE1/0/1] ipv6 address fc00:2::2/64
[*SwitchB-10GE1/0/1] ipv6 address fc00:1::2/64
[*SwitchB-10GE1/0/1] commit
```

Step 4 Verify the configuration.

If the configuration is successful, you can view that the IPv6 address and IPv6 SEND have been configured and the interface status and IPv6 protocol status are Up.

View information about 10GE 1/0/1 on Switch A.

```
[~SwitchA-10GE1/0/1] display this ipv6 interface
10GE1/0/1 current state : UP
IPv6 protocol current state : UP
link-local address is FE80::3057:B5D6:6BD6:6CA8
  Global unicast address(es):
    FC00:1::1, subnet is 1::/64
    FC00:2::2092:84CE:827B:D5A4, subnet is FC00:2::/64
  Joined group address(es):
    FF02::1:FF00:1
    FF02::1:FF7B:D5A4
    FF02::1:FFD6:6CA8
    FF02::2
    FF02::1
  MTU is 1500 bytes
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 1200000 milliseconds
  ND retransmit interval is 1000 milliseconds
  Hosts use stateless autoconfig for addresses
```

View the IPv6 SEND configuration on 10GE 1/0/1 of Switch A.

```
[~SwitchA-10GE1/0/1] display ipv6 security interface 10ge 1/0/1
 (L) : Link local address
 SEND: Security ND
 SEND information for the interface : 10GE1/0/1
------------------------------------------------------------------------------
 IPv6 address                                PrefixLength Collision Count
------------------------------------------------------------------------------
 FE80::3057:B5D6:6BD6:6CA8 (L)               10           0
 FC00:2::2092:84CE:827B:D5A4                 64           0
------------------------------------------------------------------------------
 SEND sec value : 1
 SEND security modifier value : 585D:9EA0:328:2792:B763:1DE3:BBC4:D22D
 SEND RSA key label bound : huawei
 SEND ND minimum key length value : 512
 SEND ND maximum key length value : 2048
 SEND ND Timestamp delta value : 300
 SEND ND Timestamp fuzz value : 1
 SEND ND Timestamp drift value : 1
 SEND ND fully secured mode : enable
```

View information about 10GE 1/0/1 on Switch B.

```
[~SwitchB-10GE1/0/1] display this ipv6 interface
10GE1/0/1 current state : UP
IPv6 protocol current state : UP
link-local address is FE80::2E0:E6FF:FE13:8100
  Global unicast address(es):
    FC00:1::2, subnet is 1::/64
    FC00:2::2, subnet is FC00:2::/64
  Joined group address(es):
    FF02::1:FF00:2
    FF02::1:FF13:8100
    FF02::2
    FF02::1
  MTU is 1500 bytes
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 1200000 milliseconds
  ND retransmit interval is 1000 milliseconds
  Hosts use stateless autoconfig for addresses
```

Ping the CGA link-local address of Switch A from Switch B. The ping fails because IPv6 SEND is configured on Switch A.

```
[~SwitchB-10GE1/0/1] ping ipv6 FE80::3057:B5D6:6BD6:6CA8 -i 10ge 1/0/1
  PING FE80::3057:B5D6:6BD6:6CA8 : 56  data bytes, press CTRL_C to break
    Request time out
    Request time out
    Request time out
    Request time out
    Request time out

  --- FE80::3057:B5D6:6BD6:6CA8 ping statistics ---
    5 packet(s) transmitted
    0 packet(s) received
    100.00% packet loss
    round-trip min/avg/max = 0/0/0 ms
```

Ping the CGA global unicast address of Switch A from Switch B. The ping fails because IPv6 SEND is configured on Switch A.

```
[~SwitchB-10GE1/0/1] ping ipv6 fc00:2::2092:84CE:827B:D5A4
  PING FC00:2::2092:84CE:827B:D5A4 : 56  data bytes, press CTRL_C to break
    Request time out
    Request time out
    Request time out
    Request time out
```

```
    Request time out

  --- FC00:2::2092:84CE:827B:D5A4 ping statistics ---
    5 packet(s) transmitted
    0 packet(s) received
    100.00% packet loss
    round-trip min/avg/max = 0/0/0 ms
```

Ping the common global unicast address of Switch A from Switch B. The ping fails because IPv6 SEND is configured on Switch A.

```
[~SwitchB-10GE1/0/1] ping ipv6 fc00:1::1
  PING FC00:1::1 : 56  data bytes, press CTRL_C to break
    Request time out
    Request time out
    Request time out
    Request time out
    Request time out

  --- FC00:1::1 ping statistics ---
    5 packet(s) transmitted
    0 packet(s) received
    100.00% packet loss
    round-trip min/avg/max = 0/0/0 ms
```

Disable IPv6 SEND on Switch A. The ping from Switch B to Switch A is successful. The following part provides an example of pinging the CGA global unicast address of Switch A.

```
[~SwitchA-10GE1/0/1] undo ipv6 nd security strict
[*SwitchA-10GE1/0/1] commit
[~SwitchB-10GE1/0/1] ping ipv6 fc00:2::2092:84CE:827B:D5A4
  PING FC00:2::2092:84CE:827B:D5A4 : 56  data bytes, press CTRL_C to break
    Reply from FC00:2::2092:84CE:827B:D5A4
    bytes=56 Sequence=1 hop limit=64  time = 1 ms
    Reply from FC00:2::2092:84CE:827B:D5A4
    bytes=56 Sequence=2 hop limit=64  time = 20 ms
    Reply from FC00:2::2092:84CE:827B:D5A4
    bytes=56 Sequence=3 hop limit=64  time = 1 ms
    Reply from FC00:2::2092:84CE:827B:D5A4
    bytes=56 Sequence=4 hop limit=64  time = 1 ms
    Reply from FC00:2::2092:84CE:827B:D5A4
    bytes=56 Sequence=5 hop limit=64  time = 1 ms

  --- FC00:2::2092:84CE:827B:D5A4 ping statistics ---
    5 packet(s) transmitted
    5 packet(s) received
    0.00% packet loss
    round-trip min/avg/max = 1/4/20 ms
```

----End

Configuration Files

- Configuration file of SwitchA

```
#
sysname SwitchA
#
rsa key-pair label huawei modulus 2048
#
interface 10GE1/0/1
 undo portswitch
 ipv6 enable
 ipv6 security rsakey-pair huawei
 ipv6 security modifier sec-level 1
 ipv6 address FC00:1::1/64
```

```
 ipv6 address FC00:2::/64 cga
 ipv6 address FE80::3 link-local cga
 ipv6 nd security strict
#
return
```

- Configuration file of SwitchB

```
#
sysname SwitchB
#
interface 10GE1/0/1
 undo portswitch
 ipv6 enable
 ipv6 address FC00:1::2/64
 ipv6 address FC00:2::2/64
 ipv6 address auto link-local
#
return
```

1.8 References

This section lists references of IPv6.

The following table lists the references for this document.

Document	Description	Remarks
RFC1887	An Architecture for IPv6 Unicast Address Allocation	-
RFC1970	Neighbor Discovery for IP Version 6 (IPv6)	-
RFC1981	Path MTU Discovery for IP version 6	-
RFC2375	IPv6 Multicast Address Assignments	-
RFC2147	TCP and UDP over IPv6 Jumbograms	-
RFC2460	Internet Protocol, Version 6 (IPv6) Specification	-
RFC2461	Neighbor Discovery for IP Version 6 (IPv6)	-
RFC2462	IPv6 Stateless Address Auto configuration	-
RFC2463	Internet Control Message Protocol for the Internet Protocol Version 6 Specification	-
RFC2464	Transmission of IPv6 Packets over Ethernet Networks	-
RFC2473	Generic Packet Tunneling in IPv6 Specification	-
RFC2529	Transmission of IPv6 over IPv4 Domains without Explicit Tunnels	-

Document	Description	Remarks
RFC2711	IPv6 Router Alert Option	-
RFC2893	Transition Mechanisms for IPv6 Hosts and Routers	-
RFC3056	Connection of IPv6 Domains via IPv4 Clouds	-
RFC3068	An Anycast Prefix for 6to4 Relay Routers	-
RFC3484	Default Address Selection for Internet Protocol version 6 (IPv6)	-
RFC3493	Basic Socket Interface Extensions for IPv6	-
RFC3513	IP Version 6 Addressing Architecture	-
RFC3542	Advanced Sockets API for IPv6	-
RFC3587	An Aggregatable Global Unicast Address Format	-
RFC3879	Deprecating Site Local Addresses	-
RFC4007	IPv6 Scoped Address Architecture	-
RFC4213	Basic Transition Mechanisms for IPv6 Hosts and Routers	-
RFC4291	Internet Protocol Version 6 (IPv6) Addressing Architecture	-
RFC4443	Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification	-
RFC4861	Neighbor Discovery for IP Version 6 (IPv6)	Currently, the part relating to ND proxy and ND security cannot be implemented.
RFC4862	IPv6 Stateless Address Auto configuration	Currently, hosts cannot implement RFC4862.
RFC5095	Deprecation of Type 0 Routing Headers in IPv6	-

2 IPv6 Transition Technology

About This Chapter

2.1 IPv6 Transition Technology Overview
This section describes the background and functions of IPv6 Transition Technology.

2.2 Principles
This section describes the implementation of IPv6 Transition Technology.

2.3 Configuration Notes
This section describes notes about configuring IPv6 transition technology.

2.4 Configuring IPv6 Transition Technology
This section describes the procedures for configuring IPv6 Transition Technology.

2.5 Maintaining IPv6 Transition Technology
Maintaining IPv6 transition technology includes monitoring the running status of IPv6 transition technology.

2.6 Configuration Examples
This section provides configuration examples of IPv6 transition technology, including networking requirements, configuration roadmap, and configuration procedure.

2.7 References
This section lists references of IPv6 Transition Technology.

2.1 IPv6 Transition Technology Overview

This section describes the background and functions of IPv6 Transition Technology.

During the transition from Internet Protocol version 4 (IPv4) networks to Internet Protocol version 6 (IPv6) networks, IPv6 transition technology achieves interconnection between IPv6 islands, between IPv4 islands, and between IPv6 networks and IPv4 networks.

IPv4 is the widely used Internet protocol. During initial stage of the Internet, IPv4 rapidly developed because of its simplicity, ease of implementation, and good interoperability. However, as the Internet rapidly develops, deficiency in IPv4 design becomes obvious. IPv6 is developed to overcome the deficiency of IPv4. IPv6 is incompatible with IPv4, for example, the IPv6 address format is different from the IPv4 address format, and the IPv6 packet format is different from the IPv4 packet format. Therefore, during the transition from an all-IPv4 network to an all-IPv6 network, IPv6 transition technology is required to achieve interconnection between IPv6 islands, between IPv4 islands, and between IPv6 networks and IPv4 networks.

Currently, IPv6 transition technology includes IPv4/IPv6 dual stack, IPv6 over IPv4 tunneling, and IPv6 provider edge (6PE).

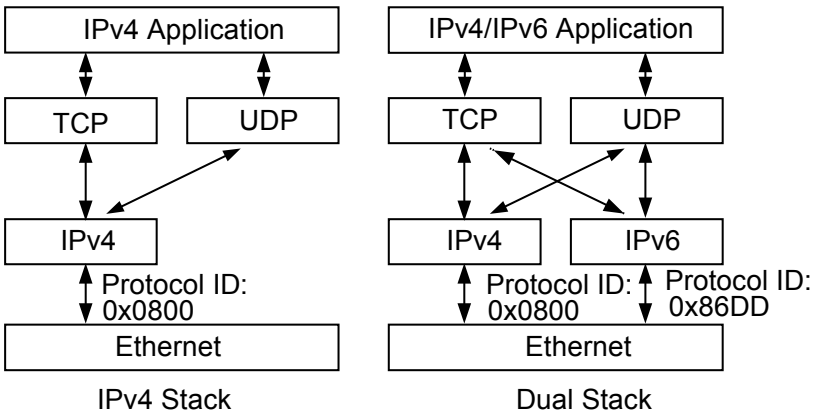
2.2 Principles

This section describes the implementation of IPv6 Transition Technology.

2.2.1 IPv4/IPv6 Dual Stack

IPv4/IPv6 dual stack is an efficient technology that implements IPv4-to-IPv6 transition. Network devices support both the IPv4 protocol stack and IPv6 protocol stack. The source device selects a protocol stack according to the IP address of the destination device. Network devices between the source and destination devices select a protocol stack to process and forward packets according to the packet protocol type. IPv4/IPv6 dual stack can be implemented on a single device or on a dual-stack network. On a dual-stack network, all devices must support the IPv4/IPv6 dual stack, and interfaces connected to the dual-stack network must have both IPv4 and IPv6 addresses configured. **Figure 2-1** shows the single-stack and dual-stack structures.

Figure 2-1 Single-stack and dual-stack structures



Support for IPv4/IPv6 dual stack:

- Multiple link protocols support IPv4/IPv6 dual stack.

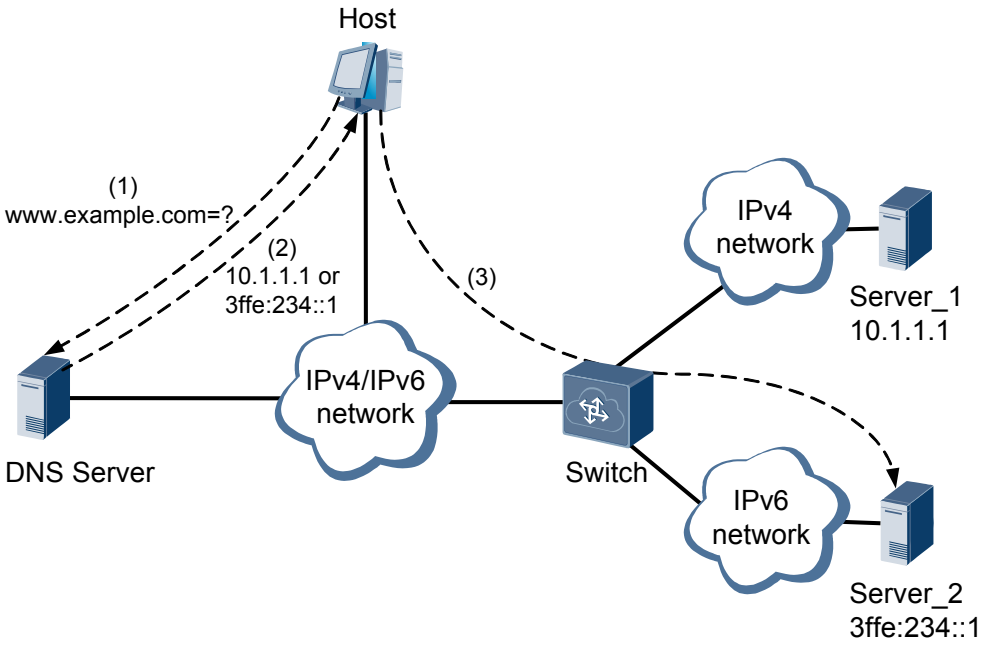
  Multiple link protocols, such as Ethernet and the Point-to-Point Protocol (PPP), support IPv4/IPv6 dual stack. In **Figure 2-1**, the link layer protocol is the Ethernet protocol. In an Ethernet frame, the protocol ID field value of 0x0800 indicates that the network layer receives IPv4 packets, and the protocol ID field value of 0x86DD indicates that the network layer receives IPv6 packets.

- Multiple applications support the dual-stack structure.

  Multiple applications, such as Domain Name System (DNS), File Transfer Protocol (FTP), and Telnet, support IPv4/IPv6 dual stack. Upper layer applications such as DNS can use the Transport Control Protocol (TCP) or User Datagram Protocol (UDP) as the transport layer protocol and prefer the IPv6 protocol stack rather than the IPv4 protocol stack as the network layer protocol.

Figure 2-2 shows a typical application of IPv4/IPv6 dual stack.

Figure 2-2 Typical application of IPv4/IPv6 dual stack



In **Figure 2-2**, the host sends a DNS request to the DNS server for the IP address of domain name www.example.com.

The DNS server searches for the IP address of the domain name and returns the mapped IP address to the host. The mapping IP address may be 10.1.1.1 or 3ffe:234::1. The DNS request sent by the host may be a class-A query or class-AAAA query. When the host is an IPv4 host, the host sends a class-A query to the DNS server for the IPv4 address of the domain name. When the host is an IPv6 host, the host sends a class-AAAA query to the DNS server for the IPv6 address of the domain name. The DNS server replies with the mapped IP address of the domain name according to the query type of the received DNS request packet.

The host accesses www.example.com according to the mapped IP address replied from the DNS server. The Switch in **Figure 2-2** supports IPv4/IPv6 dual stack. If the host needs to access network server at IPv4 address 10.1.1.1, the host can access the network server through the IPv4 protocol stack of the Switch. If the host needs to access network server at IPv6 address 3ffe:234::1, the host can access the network server through the IPv6 protocol stack of the Switch.

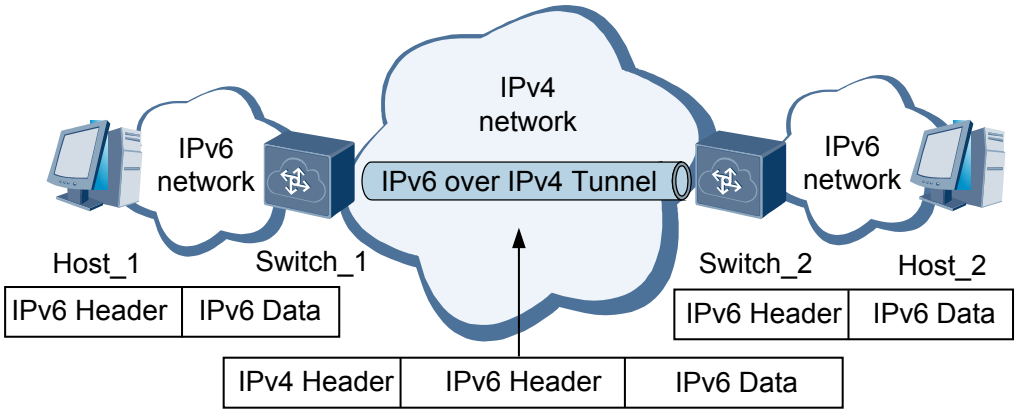
2.2.2 IPv6 over IPv4 Tunneling

Tunneling is a technology that encapsulates packets of a network protocol into packets of another network protocol for transmission. The tunneling process includes data encapsulation, transmission, and decapsulation. Tunneling technology is an important method to implement IPv4-to-IPv6 transition.

IPv4-to-IPv6 transition is a necessary trend due to IPv4 address exhaustion and IPv6 advantages. IPv6, however, is incompatible with IPv4. Existing IPv4 devices need to be replaced, which is infeasible because it requires huge cost and will interrupt existing services. Therefore, IPv4 needs to gradually transition to IPv6. During early transition, IPv4 networks are widely deployed, while IPv6 networks are isolated islands. IPv6 over IPv4 tunneling allows IPv6 packets to be transmitted on an IPv4 network, interconnecting all IPv6 islands.

Figure 2-3 shows the working mechanism of IPv6 over IPv4 tunneling technology.

Figure 2-3 IPv6 over IPv4 tunneling



In **Figure 2-3**,

1. IPv4/IPv6 dual stack is enabled and an IPv6 over IPv4 tunnel is deployed on border switches (Switch_1 and Switch_2).
2. After Switch_1 receives a packet from the connected IPv6 network, Switch_1 appends an IPv4 header to the IPv6 packet to encapsulate the IPv6 packet as an IPv4 packet if the destination address of the packet is not Switch_1 and the outbound interface of the packet is a tunnel interface.
3. On the IPv4 network, the encapsulated packet is transmitted to Switch_2.
4. Switch_2 decapsulates the packet, removes the IPv4 header, and then sends the decapsulated IPv6 packet to the connected IPv6 network.

A tunnel has a source end and a destination end. After the source end and destination end are determined, the tunnel is set up. The IPv4 address of the source end of an IPv6 over IPv4 tunnel must be configured.

- Manual tunnel: requires you to configure an IPv4 address for the destination end.

Manual tunnel

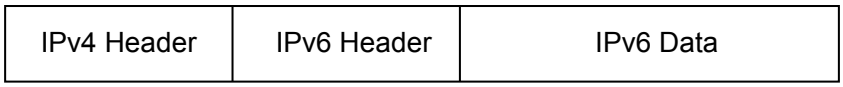
Manual tunnels are classified into IPv6 over IPv4 manual tunnels and IPv6 over IPv4 GRE tunnels based on the IPv6 packet encapsulation mode.

IPv6 over IPv4 Manual Tunnel

In a manual tunnel, an IPv6 packet is directly encapsulated into an IPv4 packet, and the source and destination addresses of the tunnel are configured to provide a point-to-point connection. Border devices on the tunnel must support IPv4/IPv6 dual stack, while other devices only need to support a single stack. The source and destination addresses of a manual tunnel must be configured. If a border device needs to set up a manual tunnel with multiple devices, multiple tunnels must be configured on the border device. Such configuration is complex. To simplify the configuration, a manual tunnel is often set up between two border devices to connect two IPv6 islands.

Figure 2-4 shows the IPv6 over IPv4 manual tunnel encapsulation format.

Figure 2-4 IPv6 over IPv4 manual tunnel encapsulation format



Packets are transmitted in an IPv6 over IPv4 manual tunnel as follows:

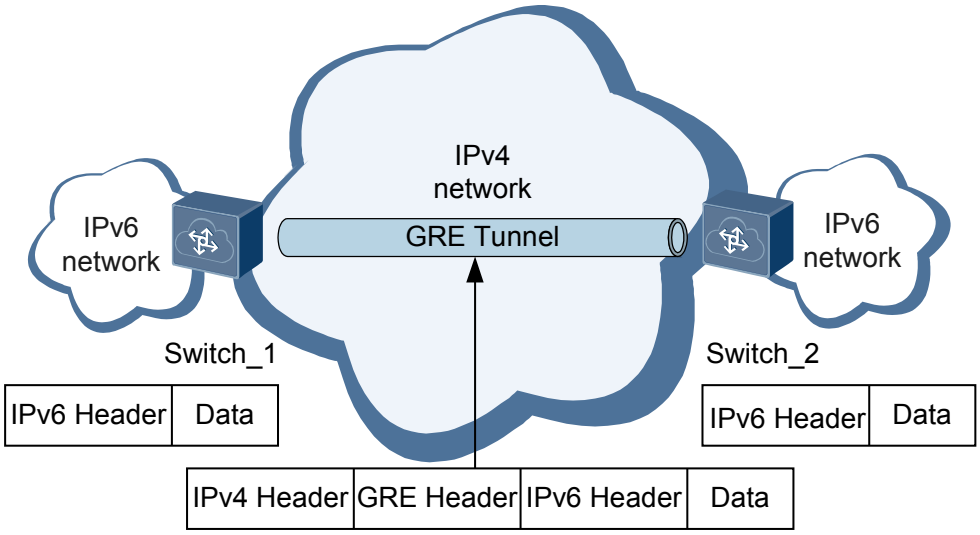
1. When a border device of the tunnel receives an IPv6 packet from the connected IPv6 network, the device searches for the IPv6 routing table according to the destination address of the IPv6 packet. If the packet is forwarded from a tunnel interface, the device encapsulates the packet according to the tunnel source and destination IPv4 addresses configured for the tunnel interface. The encapsulated packet becomes an IPv4 packet, which is then processed by the IPv4 protocol stack.
2. The IPv4 packet is forwarded to the destination end of the tunnel over an IPv4 network.
3. After the destination end of the tunnel receives the IPv4 packet, it decapsulates the packet and sends the decapsulated packet to the IPv6 protocol stack.

IPv6 over IPv4 GRE Tunnel

An IPv6 over IPv4 GRE tunnel uses standard GRE tunneling technology to provide a point-to-point connection and requires tunnel endpoint addresses to be configured. GRE tunnels can use any transport protocol and can encapsulate packets of any protocol supported by the GRE protocol, such as IPv4, IPv6, and Multiprotocol Label Switching (MPLS).

Figure 2-5 shows the working mechanism of an IPv6 over IPv4 GRE tunnel.

Figure 2-5 IPv6 over IPv4 GRE tunnel



Packets are transmitted in an IPv6 over IPv4 GRE tunnel as follows:

1. When a border device of the tunnel receives an IPv6 packet from the connected IPv6 network, the device searches for the IPv6 routing table according to the destination address of the IPv6 packet. If the packet is forwarded from a tunnel interface, the device adds a GRE header to the IPv6 packet, and then adds an IPv4 header to the GRE header according to the tunnel source and destination IPv4 addresses configured for the tunnel interface. The encapsulated packet becomes an IPv4 packet, which is then processed by the IPv4 protocol stack.
2. The IPv4 packet is forwarded to the destination end of the tunnel over an IPv4 network.
3. After the destination end of the tunnel receives the IPv4 packet, it decapsulates the packet and sends the decapsulated packet to the IPv6 protocol stack.

Compared with an IPv6 over IPv4 manual tunnel, an IPv6 over IPv4 GRE tunnel supports the Keepalive function, which enhances data transmission reliability. For details about GRE tunnels, see GRE Configuration Guide in the *CloudEngine 12800 Series Switches Configuration - Configuration Guide - VPN*.

2.2.3 6PE

IPv6 Provider Edge (6PE) technology allows ISPs to provide access services for scattered IPv6 networks over existing IPv4 backbone networks. The 6PE device converts IPv6 routes to labeled IPv6 routes and floods the routes on the ISP's IPv4/MPLS backbone network using Internal Border Gateway Protocol (IBGP) sessions. After receiving IPv6 packets entering the IPv4/MPLS backbone network, the 6PE device labels and then forwards the packets.

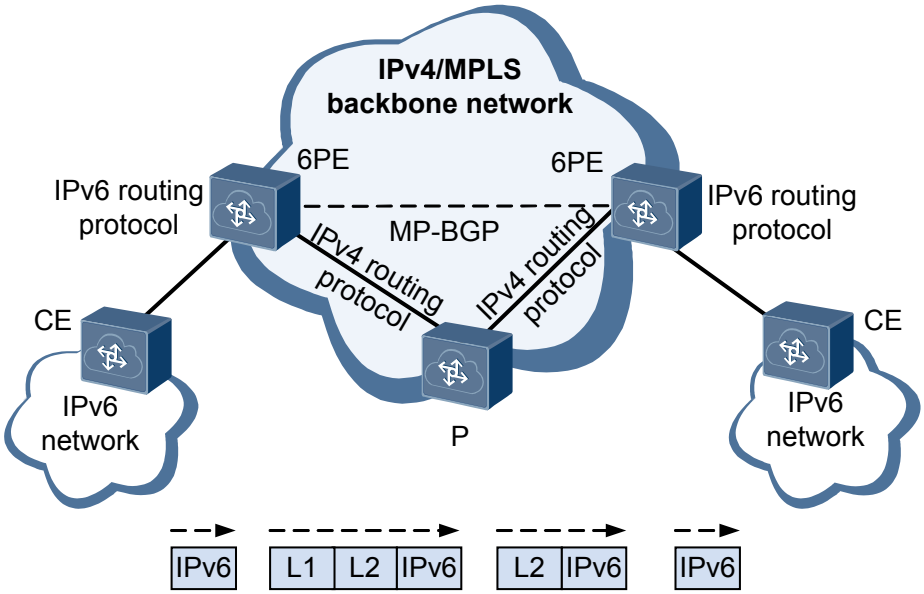
When the ISP wants to provide access services for an IPv6 network over the existing IPv4/MPLS backbone network, the ISP only needs to upgrade PE devices. Therefore, 6PE technology is a cost-effective solution to IPv4-to-IPv6 transition.

6PE only needs to be deployed on ISPs. User-side IPv6 networks can communicate with other IPv6 islands without requiring additional configurations and cannot detect the existence of the IPv4/MPLS backbone network.

Figure 2-6 shows 6PE principles.

- 6PE devices exchange IPv6 routing information with customer edges (CEs) using IPv6 routing protocols.
- 6PE devices exchange IPv6 routing information with each other using the Multiprotocol-Border Gateway Protocol (MP-BGP) and allocate MPLS labels to IPv6 prefixes.
- On the IPv4/MPLS backbone network, 6PE devices exchange IPv4 routing information with provider (P) devices using IPv4 routing protocols and establish label switched paths (LSPs) between 6PE devices using MPLS.

Figure 2-6 6PE principles



2.3 Configuration Notes

This section describes notes about configuring IPv6 transition technology.

Involved Network Elements

Other network elements are not required.

License Support

The IPv6 transition technology is controlled by a license. By default, this function is disabled on a new CE12800 series switches. To use the IPv6 function, apply for and purchase the license from the device supplier.

Version Support

Table 2-1 Products and minimum version supporting IPv6 transition technology

Series	Product	Minimum Version Required
CE12800	CE12804/CE12808/ CE12812/CE12816	V100R003C00
	CE12804S/CE12808S	V100R005C00

Feature Dependencies and Limitations

When deploying IPv6 transition technology on the switch, pay attention to the following:

- Only the Admin-VS among all VSs in port mode supports the configuration of IPv6 transition technology.
- The switch does not support multilayer IPv6 transition encapsulation or decapsulation.
- Two tunnel interfaces with the same source address and destination address cannot be configured simultaneously.
- The QoS mapping rules of an IP Tunnel are as follows:
  - Encapsulation of incoming packets:
    i. The DSCP value of the inner IP header is changed. The new DSCP value is mapped based on the internal priority and DiffServ default profile. The internal priority is obtained based on the DiffServ domain profile bound to the Layer 3 inbound interface.
    ii. The DSCP value of the outer IP header is mapped based on the internal priority and the DiffServ domain profile bound to the outbound interface. The internal priority is obtained based on the new DSCP value of the inner IP header and the DiffServ default profile.
  - Decapsulation of outgoing packets: The DSCP value of the inner IP header is mapped based on the internal priority and the DiffServ domain profile bound to the outbound interface. The internal priority is obtained based on the DSCP value of the outer IP header and the DiffServ default profile.
- CE series switches do not support packet fragmentation and reassembly. If a CE switch works as the destination endpoint of a tunnel and receives a fragmented packet, the switch fails to decapsulate the packet and discards the packet. To ensure that packets are not fragmented on a network, reduce the length of packets sent by the data source and increase the MTU of routers that support packet fragmentation on the entire network.

2.4 Configuring IPv6 Transition Technology

This section describes the procedures for configuring IPv6 Transition Technology.

2.4.1 Configuring an IPv6 over IPv4 Tunnel

You can configure IPv6 over IPv4 tunnels to interconnect IPv6 islands separated by IPv4 networks.

Pre-configuration Tasks

Before configuring an IPv6 over IPv4 tunnel, complete the following tasks:

- Configuring the IPv4/IPv6 dual stack

  NOTE

  Configure IP addresses on the source and destination ends of the tunnel to configure the IPv4/IPv6 dual stack.

  - Configure an IPv4 address for the interface that connects the device to an IPv4 network. For details, see Configuring IP Addresses for Interfaces.
  - Configure an IPv6 address for the interface that connects the device to an IPv6 network. For details, see **1.5.1 Configuring IPv6 Addresses for Interfaces**.
- Configuring a routing protocol to ensure that there are reachable IPv4 routes between the source and destination ends of the tunnel

Configuration Process

You can perform the following configuration tasks in any sequence and select one tunnel to configure.

Configuring an IPv6 over IPv4 Manual Tunnel

Context

An IPv6 over IPv4 manual tunnel functions like a permanent link between two IPv6 islands, ensuring stable connection between the two IPv6 networks.

An IPv6 over IPv4 manual tunnel needs to be configured on both ends of the tunnel.

- The tunnel source address (or source interface) and destination address must be configured.
- IPv6 addresses must be configured for tunnel interfaces and routes with the tunnel interfaces as outbound interfaces must be configured so that devices on the two IPv6 networks can communicate through the IPv6 over IPv4 manual tunnel.

On a tunnel, to ensure priorities of different services, configure a DiffServ mode so that packets in different queues are scheduled based on CoS values. For the detailed configuration procedure, see Configuring Priority Mapping on a Tunnel in the *CloudEngine 12800 Series Switches Configuration Guide - VPN*.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
interface tunnel interface-number
```

A tunnel interface is created.

Step 3 Run:

```
tunnel-protocol ipv6-ipv4
```

The tunnel mode is set to IPv6 over IPv4 manual tunnel.

Step 4 Run:

```
source { source-ip-address | interface-type interface-number }
```

A source address or source interface is specified for the tunnel.

Step 5 Run:

```
destination dest-ip-address
```

A destination address is specified for the tunnel.

Step 6 Run:

```
ipv6 enable
```

IPv6 is enabled on the tunnel interface.

Step 7 Run:

```
ipv6 address { ipv6-address prefix-length | ipv6-address/prefix-length }
```

An IPv6 address is configured for the tunnel interface.

Step 8 Run:

```
quit
```

Return to the system view.

Step 9 Choose either of the following methods to configure routes with tunnel interfaces as outbound interfaces:

- Run the **ipv6 route-static** *dest-ipv6-address prefix-length* **tunnel** *interface-number* [*nexthop-ipv6-address*] [**preference** *preference* | **tag** *tag*] * [**bfd enable** | **track** { **bfd-session** *cfg-name* | **nqa** *admin-name test-name* }] [**description** *text*] command to configure static routes.
- Configure dynamic routes. You can configure dynamic routes using the Interior Gateway Protocol (IGP) or Exterior Gateway Protocol (EGP). The Intermediate System to Intermediate System for IPv6 (IS-ISv6) protocol cannot be used to configure dynamic routes. For the configuration of dynamic routes, see the *CloudEngine 12800 Series Switches Configuration Guide - IP Routing*.

Step 10 Run:

```
commit
```

The configuration is committed.

----End

Configuring an IPv6 over IPv4 GRE Tunnel

Context

An IPv6 over IPv4 GRE tunnel functions like a permanent link between two IPv6 islands, ensuring stable connection between the two IPv6 networks.

An IPv6 over IPv4 GRE tunnel needs to be configured on both ends of the tunnel.

- The tunnel source address (or source interface) and destination address must be configured.

- IPv6 addresses must be configured for tunnel interfaces and routes with the tunnel interfaces as outbound interfaces must be configured so that devices on the two IPv6 networks can communicate through the IPv6 over IPv4 GRE tunnel.

Compared with an IPv6 over IPv4 manual tunnel, an IPv6 over IPv4 GRE tunnel supports the Keepalive function, which enhances data transmission reliability. For the configuration of the Keepalive function, see (Optional) Enabling the Keepalive Detection Function for GRE in the *CloudEngine 12800 Series Switches Configuration Guide - VPN*.

On a tunnel, to ensure priorities of different services, configure a DiffServ mode so that packets in different queues are scheduled based on CoS values. For the detailed configuration procedure, see Configuring Priority Mapping on a Tunnel in the *CloudEngine 12800 Series Switches Configuration Guide - VPN*.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
ip tunnel mode gre
```

The tunnel mode is set to GRE.

By default, the tunnel mode is VXLAN.

Only the Admin-VS in port mode supports this command, and all VSs in group mode support this command.

Step 3 Run:

```
commit
```

The configuration is committed.

NOTE

After you run the **ip tunnel mode** command to set the tunnel mode, restart the device to make the configuration take effect. A forwarding abnormality will occur if you restart only some cards or install a new card without restarting the device.

Step 4 Run:

```
system-view
```

The system view is displayed.

Step 5 Run:

```
interface tunnel interface-number
```

A tunnel interface is created.

Step 6 Run:

```
tunnel-protocol gre
```

The tunnel mode is set to GRE tunnel.

Step 7 Run:

```
source { source-ip-address | interface-type interface-number }
```

A source address or source interface is specified for the tunnel.

Step 8 Run:

```
destination [ vpn-instance vpn-instance-name ] dest-ip-address
```

A destination address is specified for the tunnel.

Step 9 Run:

```
ipv6 enable
```

IPv6 is enabled on the tunnel interface.

Step 10 Run:

```
ipv6 address { ipv6-address prefix-length | ipv6-address/prefix-length }
```

An IPv6 address is configured for the tunnel interface.

Step 11 Run:

```
quit
```

Return to the system view.

Step 12 Choose either of the following methods to configure routes with tunnel interfaces as outbound interfaces:

- Run the **ipv6 route-static** *dest-ipv6-address prefix-length* **tunnel** *interface-number* [*nexthop-ipv6-address*] [**preference** *preference* | **tag** *tag*] * [**bfd enable** | **track** { **bfd-session** *cfg-name* | **nqa** *admin-name test-name* }] [**description** *text*] command to configure static routes.
- Configure dynamic routes. You can configure dynamic routes using the Interior Gateway Protocol (IGP) or Exterior Gateway Protocol (EGP). For the configuration of dynamic routes, see the *CloudEngine 12800 Series Switches Configuration Guide - IP Routing*.

Step 13 Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

Procedure

- Run the **display ipv6 interface tunnel** *interface-number* command to check IPv6 information on the tunnel interface.

----End

2.4.2 Configuring 6PE

You can configure IPv6 provider edge (6PE) to interconnect scattered IPv6 networks over existing IPv4/MPLS backbone networks.

Pre-configuration Tasks

Before configuring 6PE, complete the following tasks:

- Configuring the IPv4/IPv6 dual stack

NOTE

Configure IP addresses for interfaces on the border device (6PE device) that connects an IPv4 network and an IPv6 network.

- Configure an IPv4 address for the interface that connects the device to an IPv4 network. For details, see Configuring IP Addresses for Interfaces.
- Configure an IPv6 address for the interface that connects the device to an IPv6 network. For details, see **1.5.1 Configuring IPv6 Addresses for Interfaces**.

- Configuring basic MPLS functions

  For details, see the *CloudEngine 12800 Series Switches Configuration Guide - MPLS*.
- Configuring a routing protocol to ensure that there are reachable IPv4 routes between 6PE devices
- Configuring a 6PE-to-CE route

Context

IPv6 provider edge (6PE) is an IPv4-to-IPv6 transition technology. You can configure 6PE to interconnect scattered IPv6 networks over existing IPv4/MPLS backbone networks.

Procedure

Step 1 Run:

```
system-view
```

The system view is displayed.

Step 2 Run:

```
bgp { as-number-plain | as-number-dot }
```

The BGP view is displayed.

Step 3 Run:

```
peer ipv4-address as-number as-number
```

An IP address and AS number of the peer are specified.

Step 4 Run:

```
peer ipv4-address connect-interface interface-type interface-number
```

The interface connected to the peer PE is specified.

Step 5 Run:

```
ipv6-family
```

The BGP-IPv6 unicast address family view is displayed.

Step 6 Run:

```
peer ipv4-address enable
```

6PE peer is enabled.

Step 7 Run:

```
peer peer-ipv4-address label-route-capability
```

The 6PE device is enabled to exchange labeled routes.

Step 8 Run:

```
commit
```

The configuration is committed.

----End

Checking the Configuration

- Run the **display mpls lsp** command to view LSP information.
- Run the **display bgp ipv6 routing-table** command to view IPv6 BGP route information.

2.5 Maintaining IPv6 Transition Technology

Maintaining IPv6 transition technology includes monitoring the running status of IPv6 transition technology.

2.5.1 Monitoring the Tunnel Running Status

Context

In routine maintenance, you can run the following command in any view to view the tunnel running status.

Procedure

- Run the **display interface tunnel** [*interface-number*] command in any view to view tunnel interface information.

----End

2.6 Configuration Examples

This section provides configuration examples of IPv6 transition technology, including networking requirements, configuration roadmap, and configuration procedure.

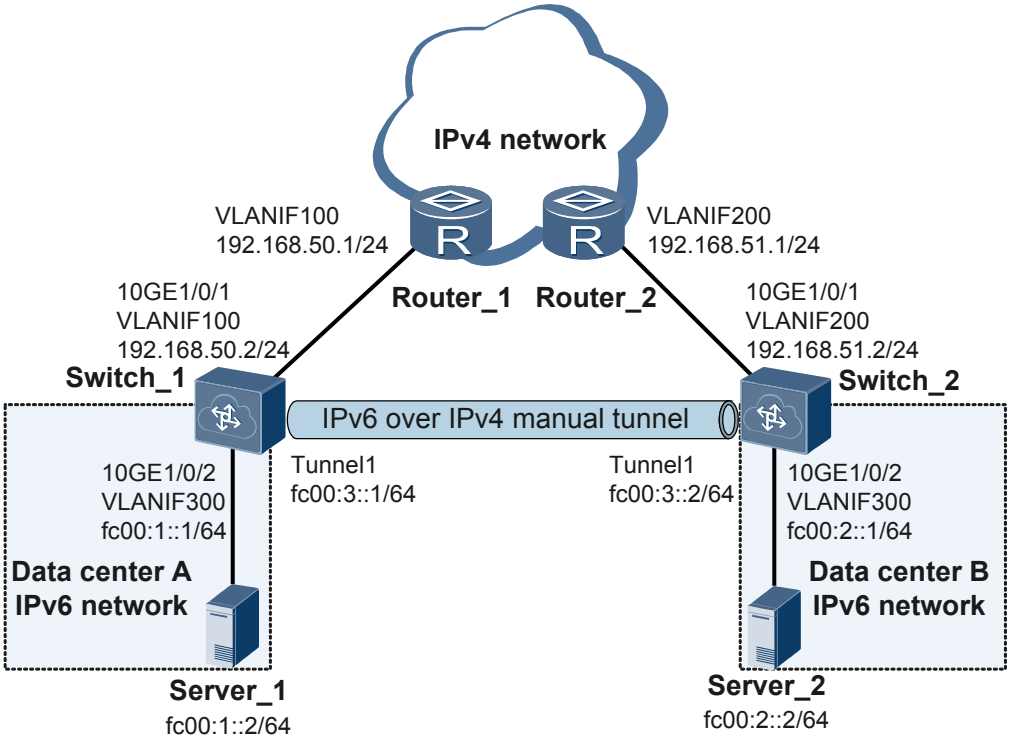
2.6.1 Example for Configuring an IPv6 over IPv4 Manual Tunnel

Networking Requirements

As shown in **Figure 2-7**, an enterprise has two data centers: data center A and data center B. The two data centers are deployed as IPv6 networks. The gateway of data center A's server Server_1 is Switch_1; the gateway of data center B's server Server_2 is Switch_2. There is an IPv4 network between the two data centers, and there are reachable routes between Router_1 and Router_2 on the IPv4 network.

The enterprise wants to establish a stable link between data center A and data center B so that Server_1 in data center A and Server_2 in data center B can communicate with each other.

Figure 2-7 Networking diagram for configuring an IPv6 over IPv4 manual tunnel



Configuration Roadmap

The configuration roadmap is as follows:

1. Configure IP addresses for interfaces on Switch_1 and Switch_2 so that Switch_1 can communicate with the IPv4 network and IPv6 network of data center A and Switch_2 can communicate with the IPv4 network and IPv6 network of data center B.
2. Configure OSPF routes to ensure that there are reachable IPv4 routes between Switch_1 and Switch_2.
3. Configure tunnel interfaces to establish an IPv6 over IPv4 manual tunnel between Switch_1 and Switch_2.
4. Configure tunnel routes using static routing to ensure that Server_1 and Server_2 can communicate through the IPv6 over IPv4 manual tunnel.

Procedure

Step 1 Configure an IP address for each physical interface.

Configure Switch_1.

```
<HUAWEI> system-view
[~HUAWEI] sysname Switch_1
[*HUAWEI] commit
[~Switch_1] vlan batch 100 300
[*Switch_1] interface 10ge 1/0/1
[*Switch_1-10GE1/0/1] port link-type trunk
[*Switch_1-10GE1/0/1] port trunk allow-pass vlan 100
[*Switch_1-10GE1/0/1] quit
```

```
[*Switch_1] interface 10ge 1/0/2
[*Switch_1-10GE1/0/2] port default vlan 300
[*Switch_1-10GE1/0/2] quit
[*Switch_1] interface vlanif 100
[*Switch_1-Vlanif100] ip address 192.168.50.2 24
[*Switch_1-Vlanif100] quit
[*Switch_1] interface vlanif 300
[*Switch_1-Vlanif300] ipv6 enable
[*Switch_1-Vlanif300] ipv6 address fc00:1::1 64
[*Switch_1-Vlanif300] quit
[*Switch_1] commit
```

Configure Switch_2.

```
<HUAWEI> system-view
[~HUAWEI] sysname Switch_2
[*HUAWEI] commit
[~Switch_2] vlan batch 200 300
[*Switch_2] interface 10ge 1/0/1
[*Switch_2-10GE1/0/1] port link-type trunk
[*Switch_2-10GE1/0/1] port trunk allow-pass vlan 200
[*Switch_2-10GE1/0/1] quit
[*Switch_2] interface 10ge 1/0/2
[*Switch_2-10GE1/0/2] port default vlan 300
[*Switch_2-10GE1/0/2] quit
[*Switch_2] interface vlanif 200
[*Switch_2-Vlanif200] ip address 192.168.51.2 24
[*Switch_2-Vlanif200] quit
[*Switch_2] interface vlanif 300
[*Switch_2-Vlanif300] ipv6 enable
[*Switch_2-Vlanif300] ipv6 address fc00:2::1 64
[*Switch_2-Vlanif300] quit
[*Switch_2] commit
```

Step 2 Configure OSPF routes to ensure that there are reachable IPv4 routes between Switch_1 and Switch_2.

Configure Switch_1.

```
[~Switch_1] ospf 1
[*Switch_1-ospf-1] area 0
[*Switch_1-ospf-1-area-0.0.0.0] network 192.168.50.0 0.0.0.255
[*Switch_1-ospf-1-area-0.0.0.0] quit
[*Switch_1-ospf-1] quit
[*Switch_1] commit
```

Configure Switch_2.

```
[~Switch_2] ospf 1
[*Switch_2-ospf-1] area 0
[*Switch_2-ospf-1-area-0.0.0.0] network 192.168.51.0 0.0.0.255
[*Switch_2-ospf-1-area-0.0.0.0] quit
[*Switch_2-ospf-1] quit
[*Switch_2] commit
```

Step 3 Configure tunnel interfaces.

Configure Switch_1.

```
[~Switch_1] interface tunnel 1
[*Switch_1-Tunnel1] tunnel-protocol ipv6-ipv4
[*Switch_1-Tunnel1] source 192.168.50.2
[*Switch_1-Tunnel1] destination 192.168.51.2
[*Switch_1-Tunnel1] ipv6 enable
[*Switch_1-Tunnel1] ipv6 address fc00:3::1 64
[*Switch_1-Tunnel1] quit
[*Switch_1] commit
```

Configure Switch_2.

```
[~Switch_2] interface tunnel 1
[*Switch_2-Tunnel1] tunnel-protocol ipv6-ipv4
[*Switch_2-Tunnel1] source 192.168.51.2
[*Switch_2-Tunnel1] destination 192.168.50.2
[*Switch_2-Tunnel1] ipv6 enable
[*Switch_2-Tunnel1] ipv6 address fc00:3::2 64
[*Switch_2-Tunnel1] quit
[*Switch_2] commit
```

Step 4 Configuring Tunnel Routes

Configure Switch_1.

```
[~Switch_1] ipv6 route-static fc00:2:: 64 tunnel 1
[*Switch_1] commit
[~Switch_1] quit
```

Configure Switch_2.

```
[~Switch_2] ipv6 route-static fc00:1:: 64 tunnel 1
[*Switch_2] commit
[~Switch_2] quit
```

Step 5 Verify the configuration.

Switch_1 can ping the IPv4 address (192.168.51.2) of VLANIF 200 on Switch_2 successfully.

```
<Switch_1> ping 192.168.51.2
  PING 192.168.51.2: 56  data bytes, press CTRL_C to break
    Reply from 192.168.51.2: bytes=56 Sequence=1 ttl=253 time=1 ms
    Reply from 192.168.51.2: bytes=56 Sequence=2 ttl=253 time=1 ms
    Reply from 192.168.51.2: bytes=56 Sequence=3 ttl=253 time=1 ms
    Reply from 192.168.51.2: bytes=56 Sequence=4 ttl=253 time=1 ms
    Reply from 192.168.51.2: bytes=56 Sequence=5 ttl=253 time=1 ms

  --- 192.168.51.2 ping statistics ---
    5 packet(s) transmitted
    5 packet(s) received
    0.00% packet loss
    round-trip min/avg/max = 1/1/1 ms
```

Switch_1 can ping the IPv6 address (fc00:3::2) of Tunnel1 on Switch_2 successfully.

```
<Switch_1> ping ipv6 fc00:3::2
  PING FC00:3::2 : 56  data bytes, press CTRL_C to break
    Reply from FC00:3::2
    bytes=56 Sequence=1 hop limit=63  time = 28 ms
    Reply from FC00:3::2
    bytes=56 Sequence=2 hop limit=63  time = 27 ms
    Reply from FC00:3::2
    bytes=56 Sequence=3 hop limit=63  time = 26 ms
    Reply from FC00:3::2
    bytes=56 Sequence=4 hop limit=63  time = 27 ms
    Reply from FC00:3::2
    bytes=56 Sequence=5 hop limit=63  time = 26 ms
  --- FC00:3::2 ping statistics ---
    5 packet(s) transmitted
    5 packet(s) received
    0.00% packet loss
    round-trip min/avg/max = 26/26/28 ms
```

Server_1 and Server_2 can ping each other.

----End

Configuration Files

- Configuration file of Switch_1

```
#
sysname Switch_1
#
vlan batch 100 300
#
interface Vlanif100
 ip address 192.168.50.2 255.255.255.0
#
interface Vlanif300
 ipv6 enable
 ipv6 address FC00:1::1/64
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 100
#
interface 10GE1/0/2
 port default vlan 300
#
interface Tunnel1
 ipv6 enable
 ipv6 address FC00:3::1/64
 tunnel-protocol ipv6-ipv4
 source 192.168.50.2
 destination 192.168.51.2
#
ospf 1
 area 0.0.0.0
  network 192.168.50.0 0.0.0.255
#
ipv6 route-static FC00:2:: 64 Tunnel1
#
return
```

- Configuration file of Switch_2

```
#
sysname Switch_2
#
vlan batch 200 300
#
interface Vlanif200
 ip address 192.168.51.2 255.255.255.0
#
interface Vlanif300
 ipv6 enable
 ipv6 address FC00:2::1/64
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 200
#
interface 10GE1/0/2
 port default vlan 300
#
interface Tunnel1
 ipv6 enable
 ipv6 address FC00:3::2/64
 tunnel-protocol ipv6-ipv4
 source 192.168.51.2
 destination 192.168.50.2
#
ospf 1
 area 0.0.0.0
  network 192.168.51.0 0.0.0.255
#
ipv6 route-static FC00:1:: 64 Tunnel1
```

```
#
return
```

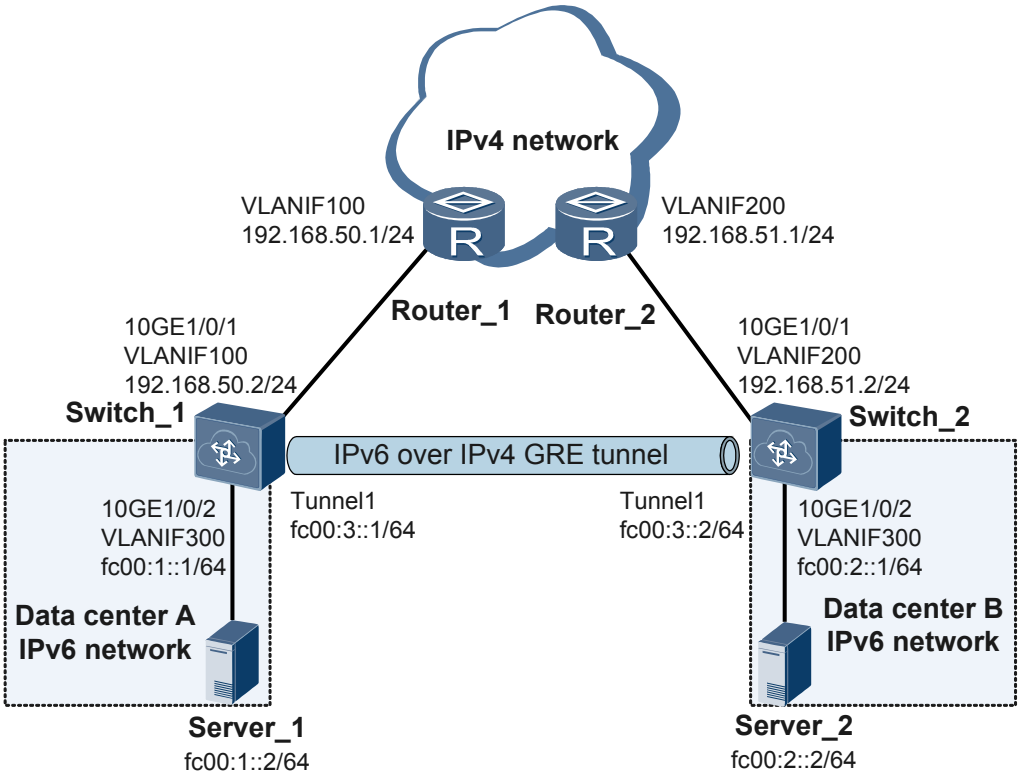
2.6.2 Example for Configuring an IPv6 over IPv4 GRE Tunnel

Networking Requirements

As shown in **Figure 2-8**, an enterprise has two data centers: data center A and data center B. The two data centers are deployed as IPv6 networks. The gateway of data center A's server Server_1 is Switch_1; the gateway of data center B's server Server_2 is Switch_2. There is an IPv4 network between the two data centers, and there are reachable routes between Router_1 and Router_2 on the IPv4 network.

The enterprise wants to establish a stable link between data center A and data center B so that Server_1 in data center A and Server_2 in data center B can communicate with each other and data can be transmitted reliably between the two data centers.

Figure 2-8 Networking diagram for configuring an IPv6 over IPv4 GRE tunnel



Configuration Roadmap

The configuration roadmap is as follows:

1. Configure IP addresses for interfaces on Switch_1 and Switch_2 so that Switch_1 can communicate with the IPv4 network and IPv6 network of data center A and Switch_2 can communicate with the IPv4 network and IPv6 network of data center B.
2. Configure OSPF routes to ensure that there are reachable IPv4 routes between Switch_1 and Switch_2.

3. Configure tunnel interfaces to establish an IPv6 over IPv4 GRE tunnel between Switch_1 and Switch_2 and configure the Keepalive function to ensure data transmission reliability between the two data centers.
4. Configure OSPFv3 routes as tunnel routes to ensure that Server_1 and Server_2 can communicate through the IPv6 over IPv4 GRE tunnel.

Procedure

Step 1 Configure an IP address for each physical interface.

Configure Switch_1.

```
<HUAWEI> system-view
[~HUAWEI] sysname Switch_1
[*HUAWEI] commit
[~Switch_1] vlan batch 100 300
[*Switch_1] interface 10ge 1/0/1
[*Switch_1-10GE1/0/1] port link-type trunk
[*Switch_1-10GE1/0/1] port trunk allow-pass vlan 100
[*Switch_1-10GE1/0/1] quit
[*Switch_1] interface 10ge 1/0/2
[*Switch_1-10GE1/0/2] port default vlan 300
[*Switch_1-10GE1/0/2] quit
[*Switch_1] interface vlanif 100
[*Switch_1-Vlanif100] ip address 192.168.50.2 24
[*Switch_1-Vlanif100] quit
[*Switch_1] interface vlanif 300
[*Switch_1-Vlanif300] ipv6 enable
[*Switch_1-Vlanif300] ipv6 address fc00:1::1 64
[*Switch_1-Vlanif300] quit
[*Switch_1] commit
```

Configure Switch_2.

```
<HUAWEI> system-view
[~HUAWEI] sysname Switch_2
[*HUAWEI] commit
[~Switch_2] vlan batch 200 300
[*Switch_2] interface 10ge 1/0/1
[*Switch_2-10GE1/0/1] port link-type trunk
[*Switch_2-10GE1/0/1] port trunk allow-pass vlan 200
[*Switch_2-10GE1/0/1] quit
[*Switch_2] interface 10ge 1/0/2
[*Switch_2-10GE1/0/2] port default vlan 300
[*Switch_2-10GE1/0/2] quit
[*Switch_2] interface vlanif 200
[*Switch_2-Vlanif200] ip address 192.168.51.2 24
[*Switch_2-Vlanif200] quit
[*Switch_2] interface vlanif 300
[*Switch_2-Vlanif300] ipv6 enable
[*Switch_2-Vlanif300] ipv6 address fc00:2::1 64
[*Switch_2-Vlanif300] quit
[*Switch_2] commit
```

Step 2 Configure OSPF routes to ensure that there are reachable IPv4 routes between Switch_1 and Switch_2.

Configure Switch_1.

```
[~Switch_1] ospf 1
[*Switch_1-ospf-1] area 0
[*Switch_1-ospf-1-area-0.0.0.0] network 192.168.50.0 0.0.0.255
[*Switch_1-ospf-1-area-0.0.0.0] quit
[*Switch_1-ospf-1] quit
[*Switch_1] commit
```

Configure Switch_2.

```
[~Switch_2] ospf 1
[*Switch_2-ospf-1] area 0
[*Switch_2-ospf-1-area-0.0.0.0] network 192.168.51.0 0.0.0.255
[*Switch_2-ospf-1-area-0.0.0.0] quit
[*Switch_2-ospf-1] quit
[*Switch_2] commit
```

Step 3 Configure the tunnel mode.

Configure Switch_1.

```
[~Switch_1] ip tunnel mode gre
[*Switch_1] commit
```

Configure Switch_2.

```
[~Switch_2] ip tunnel mode gre
[*Switch_2] commit
```

NOTE

This command takes effect only after the configuration is saved and device restarts. You can choose to restart the device immediately or after all configurations are complete.

Step 4 Configure tunnel interfaces.

Configure Switch_1.

```
[~Switch_1] interface tunnel 1
[*Switch_1-Tunnel1] tunnel-protocol gre
[*Switch_1-Tunnel1] source 192.168.50.2
[*Switch_1-Tunnel1] destination 192.168.51.2
[*Switch_1-Tunnel1] ipv6 enable
[*Switch_1-Tunnel1] ipv6 address fc00:3::1 64
[*Switch_1-Tunnel1] keepalive
[*Switch_1-Tunnel1] quit
[*Switch_1] commit
```

Configure Switch_2.

```
[~Switch_2] interface tunnel 1
[*Switch_2-Tunnel1] tunnel-protocol gre
[*Switch_2-Tunnel1] source 192.168.51.2
[*Switch_2-Tunnel1] destination 192.168.50.2
[*Switch_2-Tunnel1] ipv6 enable
[*Switch_2-Tunnel1] ipv6 address fc00:3::2 64
[*Switch_2-Tunnel1] keepalive
[*Switch_2-Tunnel1] quit
[*Switch_2] commit
```

Step 5 Configuring Tunnel Routes

Configure Switch_1.

```
[~Switch_1] ospfv3 2
[*Switch_1-ospfv3-2] router-id 1.1.1.1
[*Switch_1-ospfv3-2] quit
[*Switch_1] interface vlanif 300
[*Switch_1-Vlanif300] ospfv3 2 area 0
[*Switch_1-Vlanif300] quit
[*Switch_1] interface tunnel 1
[*Switch_1-Tunnel1] ospfv3 2 area 0
[*Switch_1-Tunnel1] quit
[*Switch_1] commit
[~Switch_1] quit
```

Configure Switch_2.

```
[~Switch_2] ospfv3 2
[*Switch_2-ospfv3-2] router-id 2.2.2.2
```

```
[*Switch_2-ospfv3-2] quit
[*Switch_2] interface vlanif 300
[*Switch_2-Vlanif300] ospfv3 2 area 0
[*Switch_2-Vlanif300] quit
[*Switch_2] interface tunnel 1
[*Switch_2-Tunnel1] ospfv3 2 area 0
[*Switch_2-Tunnel1] quit
[*Switch_2] commit
[~Switch_2] quit
```

Step 6 Verify the configuration.

Switch_1 can ping the IPv4 address (192.168.51.2) of VLANIF 200 on Switch_2 successfully.

```
<Switch_1> ping 192.168.51.2
  PING 192.168.51.2: 56  data bytes, press CTRL_C to break
    Reply from 192.168.51.2: bytes=56 Sequence=1 ttl=253 time=1 ms
    Reply from 192.168.51.2: bytes=56 Sequence=2 ttl=253 time=1 ms
    Reply from 192.168.51.2: bytes=56 Sequence=3 ttl=253 time=1 ms
    Reply from 192.168.51.2: bytes=56 Sequence=4 ttl=253 time=1 ms
    Reply from 192.168.51.2: bytes=56 Sequence=5 ttl=253 time=1 ms

  --- 192.168.51.2 ping statistics ---
    5 packet(s) transmitted
    5 packet(s) received
    0.00% packet loss
    round-trip min/avg/max = 1/1/1 ms
```

Switch_1 can ping the IPv6 address (fc00:3::2) of Tunnel1 on Switch_2 successfully.

```
<Switch_1> ping ipv6 fc00:3::2
  PING FC00:3::2 : 56  data bytes, press CTRL_C to break
    Reply from FC00:3::2
    bytes=56 Sequence=1 hop limit=63  time = 28 ms
    Reply from FC00:3::2
    bytes=56 Sequence=2 hop limit=63  time = 27 ms
    Reply from FC00:3::2
    bytes=56 Sequence=3 hop limit=63  time = 26 ms
    Reply from FC00:3::2
    bytes=56 Sequence=4 hop limit=63  time = 27 ms
    Reply from FC00:3::2
    bytes=56 Sequence=5 hop limit=63  time = 26 ms
  --- FC00:3::2 ping statistics ---
    5 packet(s) transmitted
    5 packet(s) received
    0.00% packet loss
    round-trip min/avg/max = 26/26/28 ms
```

Server_1 and Server_2 can ping each other.

----End

Configuration Files

- Configuration file of Switch_1

```
#
sysname Switch_1
#
vlan batch 100 300
#
ip tunnel mode gre
#
ospfv3 2
 router-id 1.1.1.1
 area 0.0.0.0
#
interface Vlanif100
```

```
 ip address 192.168.50.2 255.255.255.0
#
interface Vlanif300
 ipv6 enable
 ipv6 address FC00:1::1/64
 ospfv3 2 area 0.0.0.0
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 100
#
interface 10GE1/0/2
 port default vlan 300
#
interface Tunnel1
 ipv6 enable
 ipv6 address FC00:3::1/64
 tunnel-protocol gre
 keepalive
 source 192.168.50.2
 destination 192.168.51.2
 ospfv3 2 area 0.0.0.0
#
ospf 1
 area 0.0.0.0
  network 192.168.50.0 0.0.0.255
#
return
```

- Configuration file of Switch_2

```
#
sysname Switch_2
#
vlan batch 200 300
#
ip tunnel mode gre
#
ospfv3 2
 router-id 2.2.2.2
 area 0.0.0.0
#
interface Vlanif200
 ip address 192.168.51.2 255.255.255.0
#
interface Vlanif300
 ipv6 enable
 ipv6 address FC00:2::1/64
 ospfv3 2 area 0.0.0.0
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 200
#
interface 10GE1/0/2
 port default vlan 300
#
interface Tunnel1
 ipv6 enable
 ipv6 address FC00:3::2/64
 tunnel-protocol gre
 keepalive
 source 192.168.51.2
 destination 192.168.50.2
 ospfv3 2 area 0.0.0.0
#
ospf 1
 area 0.0.0.0
  network 192.168.51.0 0.0.0.255
#
return
```

2.6.3 Example for Configuring 6PE

Networking Requirements

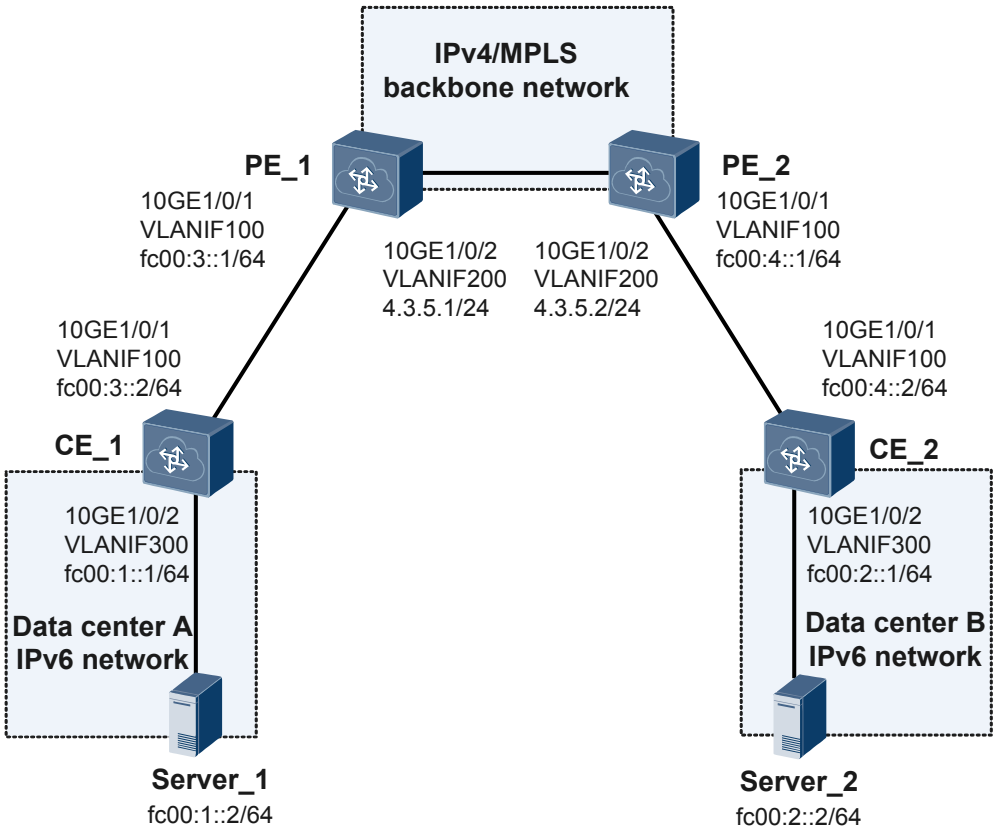
As shown in **Figure 2-9**, an enterprise has two data centers: data center A and data center B. The two data centers are deployed as IPv6 networks. The gateway of data center A's server Server_1 is CE_1; the gateway of data center B's server Server_2 is CE_2. Data center A and data center B access the IPv4/MPLS backbone network through PE_1 and PE_2. The IPv4/MPLS backbone network resides between PE_1 and PE_2.

The enterprise requires that data center A and data center B communicate through the IPv4/MPLS backbone network so that Server_1 in data center A and Server_2 in data center B can communicate with each other.

NOTE

Before using MPLS functions, purchase a license.

Figure 2-9 Networking diagram for configuring 6PE



Configuration Roadmap

The configuration roadmap is as follows:

1. Configure IP addresses for device interfaces.
2. Enable MPLS on PE_1 and PE_2.

3. Configure 6PE peers on PE_1 and PE_2.
4. Configure static routes on CE_1, CE_2, PE_1 and PE_2.

Configuration Procedure

1. Configure IP addresses for device interfaces.

   # Configure PE_1.

```
<HUAWEI> system-view
[~HUAWEI] sysname PE_1
[*HUAWEI] commit
[~PE_1] vlan batch 100 200
[*PE_1] interface 10ge 1/0/1
[*PE_1-10GE1/0/1] port link-type trunk
[*PE_1-10GE1/0/1] port trunk allow-pass vlan 100
[*PE_1-10GE1/0/1] quit
[*PE_1] interface 10ge 1/0/2
[*PE_1-10GE1/0/2] port link-type trunk
[*PE_1-10GE1/0/2] port trunk allow-pass vlan 200
[*PE_1-10GE1/0/2] quit
[*PE_1] interface vlanif 100
[*PE_1-Vlanif100] ipv6 enable
[*PE_1-Vlanif100] ipv6 address fc00:3::1 64
[*PE_1-Vlanif100] quit
[*PE_1] interface vlanif 200
[*PE_1-Vlanif200] ip address 4.3.5.1 24
[*PE_1-Vlanif200] quit
[*PE_1] interface loopback 0
[*PE_1-LoopBack0] ip address 1.1.1.9 255.255.255.255
[*PE_1-LoopBack0] quit
[*PE_1] commit
```

   # Configure PE_2.

```
<HUAWEI> system-view
[~HUAWEI] sysname PE_2
[*HUAWEI] commit
[~PE_2] vlan batch 100 200
[*PE_2] interface 10ge 1/0/1
[*PE_2-10GE1/0/1] port link-type trunk
[*PE_2-10GE1/0/1] port trunk allow-pass vlan 100
[*PE_2-10GE1/0/1] quit
[*PE_2] interface 10ge 1/0/2
[*PE_21-10GE1/0/2] port link-type trunk
[*PE_21-10GE1/0/2] port trunk allow-pass vlan 200
[*PE_21-10GE1/0/2] quit
[*PE_2] interface vlanif 100
[*PE_21-Vlanif100] ipv6 enable
[*PE_21-Vlanif100] ipv6 address fc00:4::1 64
[*PE_2-Vlanif100] quit
[*PE_21] interface vlanif 200
[*PE_2-Vlanif200] ip address 4.3.5.2 24
[*PE_2-Vlanif200] quit
[*PE_2] interface loopback 0
[*PE_2-LoopBack0] ip address 2.2.2.9 255.255.255.255
[*PE_2-LoopBack0] quit
[*PE_2] commit
```

   # Configure CE_1.

```
<HUAWEI> system-view
[~HUAWEI] sysname CE_1
[*HUAWEI] commit
[~CE_1] vlan batch 100 300
[*CE_1] interface 10ge 1/0/1
[*CE_1-10GE1/0/1] port link-type trunk
[*CE_1-10GE1/0/1] port trunk allow-pass vlan 100
[*CE_1-10GE1/0/1] quit
[*CE_1] interface 10ge 1/0/2
```

```
[*CE_1-10GE1/0/2] port default vlan 300
[*CE_1-10GE1/0/2] quit
[*CE_1] interface vlanif 100
[*CE_1-Vlanif100] ipv6 enable
[*CE_1-Vlanif100] ipv6 address fc00:3::2 64
[*CE_1-Vlanif100] quit
[*CE_1] interface vlanif 300
[*CE_1-Vlanif300] ipv6 enable
[*CE_1-Vlanif300] ipv6 address fc00:1::1 64
[*CE_1-Vlanif300] quit
[*CE_1] commit
```

Configure CE_2.

```
<HUAWEI> system-view
[~HUAWEI] sysname CE_2
[*HUAWEI] commit
[~CE_2] vlan batch 100 300
[*CE_2] interface 10ge 1/0/1
[*CE_2-10GE1/0/1] port link-type trunk
[*CE_2-10GE1/0/1] port trunk allow-pass vlan 100
[*CE_2-10GE1/0/1] quit
[*CE_2] interface 10ge 1/0/2
[*CE_2-10GE1/0/2] port default vlan 300
[*CE_2-10GE1/0/2] quit
[*CE_2] interface vlanif 100
[*CE_2-Vlanif100] ipv6 enable
[*CE_2-Vlanif100] ipv6 address fc00:4::2 64
[*CE_2-Vlanif100] quit
[*CE_2] interface vlanif 300
[*CE_2-Vlanif300] ipv6 enable
[*CE_2-Vlanif300] ipv6 address fc00:2::1 64
[*CE_2-Vlanif300] quit
[*CE_2] commit
```

2. Enable MPLS on PE_1 and PE_2.

NOTE

PEs in this example are directly connected. Run the **label advertise** command to enable the egress node to assign labels to the penultimate hop.

Enable MPLS and LDP on PE_1.

```
[~PE_1] mpls lsr-id 1.1.1.9
[*PE_1] mpls
[*PE_1-mpls] lsp-trigger all
[*PE_1-mpls] label advertise non-null
[*PE_1-mpls] quit
[*PE_1] mpls ldp
[*PE_1-mpls-ldp] quit
[*PE_1] interface vlanif 200
[*PE_1-Vlanif200] mpls
[*PE_1-Vlanif200] mpls ldp
[*PE_1-Vlanif200] quit
[*PE_1] commit
```

Enable MPLS and LDP on PE_2.

```
[~PE_2] mpls lsr-id 2.2.2.9
[*PE_2] mpls
[*PE_2-mpls] lsp-trigger all
[*PE_2-mpls] label advertise non-null
[*PE_2-mpls] quit
[*PE_2] mpls ldp
[*PE_2-mpls-ldp] quit
[*PE_2] interface vlanif 200
[*PE_2-Vlanif200] mpls
[*PE_2-Vlanif200] mpls ldp
[*PE_2-Vlanif200] quit
[*PE_2] commit
```

Configure OSPF on PE_1 to trigger the setup of an LSP.

```
[~PE_1] ospf
[*PE_1-ospf-1] area 0
[*PE_1-ospf-1-area-0.0.0.0] network 1.1.1.9 0.0.0.0
[*PE_1-ospf-1-area-0.0.0.0] network 4.3.5.0 0.0.0.255
[*PE_1-ospf-1-area-0.0.0.0] quit
[*PE_1-ospf-1] quit
[*PE_1] commit
```

Configure OSPF on PE_2 to trigger the setup of an LSP.

```
[~PE_2] ospf
[*PE_2-ospf-1] area 0
[*PE_2-ospf-1-area-0.0.0.0] network 2.2.2.9 0.0.0.0
[*PE_2-ospf-1-area-0.0.0.0] network 4.3.5.0 0.0.0.255
[*PE_2-ospf-1-area-0.0.0.0] quit
[*PE_2-ospf-1] quit
[*PE_2] commit
```

3. Configure 6PE peers on PE_1 and PE_2.

   # Configure IBGP on PE_1, enable the 6PE capability on the peer, and import direct and static IPv6 routes.

```
[~PE_1] bgp 65100
[*PE_1-bgp] peer 2.2.2.9 as-number 65100
[*PE_1-bgp] peer 2.2.2.9 connect-interface loopback 0
[*PE_1-bgp] ipv6-family
[*PE_1-bgp-af-ipv6] import-route direct
[*PE_1-bgp-af-ipv6] import-route static
[*PE_1-bgp-af-ipv6] peer 2.2.2.9 enable
[*PE_1-bgp-af-ipv6] peer 2.2.2.9 label-route-capability
[*PE_1-bgp-af-ipv6] quit
[*PE_1-bgp] quit
[*PE_1] commit
```

   # Configure IBGP on PE_2, enable the 6PE capability on the peer, and import direct and static IPv6 routes.

```
[~PE_2] bgp 65100
[*PE_2-bgp] peer 1.1.1.9 as-number 65100
[*PE_2-bgp] peer 1.1.1.9 connect-interface loopback 0
[*PE_2-bgp] ipv6-family
[*PE_2-bgp-af-ipv6] import-route direct
[*PE_2-bgp-af-ipv6] import-route static
[*PE_2-bgp-af-ipv6] peer 1.1.1.9 enable
[*PE_2-bgp-af-ipv6] peer 1.1.1.9 label-route-capability
[*PE_2-bgp-af-ipv6] quit
[*PE_2-bgp] quit
[*PE_2] commit
```

4. Configure static routes on CE_1, CE_2, PE_1 and PE_2.

   # Configure CE_1.

```
[~CE_1] ipv6 route-static :: 0 vlanif 100 fc00:3::1
[*CE_1] commit
```

   # Configure CE_2.

```
[~CE_2] ipv6 route-static :: 0 vlanif 100 fc00:4::1
[*CE_2] commit
```

   # Configure PE_1.

```
[~PE_1] ipv6 route-static fc00:1:: 64 vlanif 100 fc00:3::2
[*PE_1] commit
```

   # Configure PE_2.

```
[~PE_2] ipv6 route-sattic fc00:2:: 64 vlanif 100 fc00:4::2
[*PE_2] commit
```

5. Verify the configuration.

   # CE_1 can ping the IPv6 address of CE_2 successfully.

```
[~CE_1] ping ipv6 fc00:4::2
PING FC00:4::2 : 56  data bytes, press CTRL_C to break
```

```
    Reply from FC00:4::2
    bytes=56 Sequence=1 hop limit=63  time = 50 ms
    Reply from FC00:4::2
    bytes=56 Sequence=2 hop limit=63  time = 1 ms
    Reply from FC00:4::2
    bytes=56 Sequence=3 hop limit=63  time = 1 ms
    Reply from FC00:4::2
    bytes=56 Sequence=4 hop limit=63  time = 1 ms
    Reply from FC00:4::2
    bytes=56 Sequence=5 hop limit=63  time = 1 ms

  --- FC00:4::2 ping statistics ---
    5 packet(s) transmitted
    5 packet(s) received
    0.00% packet loss
    round-trip min/avg/max = 1/10/50 ms
```

Server_1 and Server_2 can ping each other.

Configuration Files

- Configuration file of PE_1

```
#
sysname PE_1
#
vlan batch 100 200
#
mpls lsr-id 1.1.1.9
#
mpls
 label advertise non-null
 lsp-trigger all
#
mpls ldp
#
interface Vlanif100
 ipv6 enable
 ipv6 address FC00:3::1/64
#
interface Vlanif200
 ip address 4.3.5.1 255.255.255.0
 mpls
 mpls ldp
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 100
#
interface 10GE1/0/2
 port link-type trunk
 port trunk allow-pass vlan 200
#
interface LoopBack0
 ip address 1.1.1.9 255.255.255.255
#
bgp 65100
 peer 2.2.2.9 as-number 65100
 peer 2.2.2.9 connect-interface LoopBack0
 #
 ipv4-family unicast
  peer 2.2.2.9 enable
 #
 ipv6-family unicast
  import-route direct
  import-route static
  peer 2.2.2.9 enable
  peer 2.2.2.9 label-route-capability
#
ospf 1
```

```
 area 0.0.0.0
  network 1.1.1.9 0.0.0.0
  network 4.3.5.0 0.0.0.255
#
ipv6 route-static FC00:1:: 64 Vlanif100 FC00:3::2
#
return
```

- Configuration file of PE_2

```
#
sysname PE_2
#
vlan batch 100 200
#
mpls lsr-id 2.2.2.9
#
mpls
 label advertise non-null
 lsp-trigger all
#
mpls ldp
#
interface Vlanif100
 ipv6 enable
 ipv6 address FC00:4::1/64
#
interface Vlanif200
 ip address 4.3.5.2 255.255.255.0
 mpls
 mpls ldp
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 100
#
interface 10GE1/0/2
 port link-type trunk
 port trunk allow-pass vlan 200
#
interface LoopBack0
 ip address 2.2.2.9 255.255.255.255
#
bgp 65100
 peer 1.1.1.9 as-number 65100
 peer 1.1.1.9 connect-interface LoopBack0
 #
 ipv4-family unicast
  peer 1.1.1.9 enable
 #
 ipv6-family unicast
  import-route direct
  import-route static
  peer 1.1.1.9 enable
  peer 1.1.1.9 label-route-capability
#
ospf 1
 area 0.0.0.0
  network 2.2.2.9 0.0.0.0
  network 4.3.5.0 0.0.0.255
#
ipv6 route-static FC00:2:: 64 Vlanif100 FC00:4::2
#
return
```

- Configuration file of CE_1

```
#
sysname CE_1
#
vlan batch 100 300
#
```

```
interface Vlanif100
 ipv6 enable
 ipv6 address FC00:3::2 64
#
interface Vlanif300
 ipv6 enable
 ipv6 address FC00:1::1 64
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 100
#
interface 10GE1/0/2
 port default vlan 300
#
ipv6 route-static :: 0 Vlanif100 FC00:3::1
#
return
```

- Configuration file of CE_2

```
#
sysname CE_2
#
interface Vlanif100
 ipv6 enable
 ipv6 address FC00:4::2 64
#
interface Vlanif300
 ipv6 enable
 ipv6 address FC00:2::1 64
#
interface 10GE1/0/1
 port link-type trunk
 port trunk allow-pass vlan 100
#
interface 10GE1/0/2
 port default vlan 300
#
ipv6 route-static :: 0 Vlanif100 FC00:4::1
#
return
```

2.7 References

This section lists references of IPv6 Transition Technology.

The following table lists the references for this document.

Document	Description	Remarks
RFC 2464	Transmission of IPv6 Packets over Ethernet Networks	-
RFC 2472	IP Version 6 over PPP	-
RFC 2473	Generic Packet Tunneling in IPv6 Specification	-
RFC 2529	Transmission of IPv6 over IPv4 Domains without Explicit Tunnels	-
RFC 2893	Transition Mechanisms for IPv6 Hosts and Routers	-

Document	Description	Remarks
RFC 3056	Connection of IPv6 Domains via IPv4 Clouds	-
RFC 3068	An Anycast Prefix for 6to4 Relay Routers	-
RFC 3484	Default Address Selection for Internet Protocol version 6 (IPv6)	-
RFC 3493	Basic Socket Interface Extensions for IPv6	-
RFC 4213	Basic Transition Mechanisms for IPv6 Hosts and Routers	-