Руководство пользователя

QSR-2830

Оглавление

1. CONFIGURING IP ADDRESS AND SERVICE

1.1. Understanding IP Address Configuration

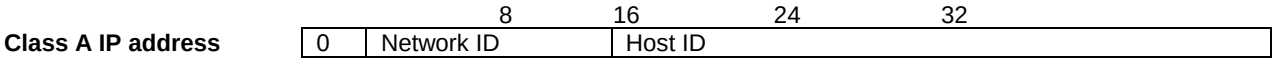
1.1.1. Overview

An IP address is made up of 32 binary bits and expressed in dotted decimal format for the convenience of writing and description. In dotted decimal format, the 32 binary bits form four octets (1 octet equals to 8 bits). Each octet is separated by a period (dot) in the range from 0 to 255. For example, 192.168.1.1 is an IP address in dotted decimal format.

An IP address is an address that IP uses for interconnection. A 32-bit IP address consists of two parts: network address and local address. According to the first several bits of the network address of an IP address, IP addresses are divided into four categories.

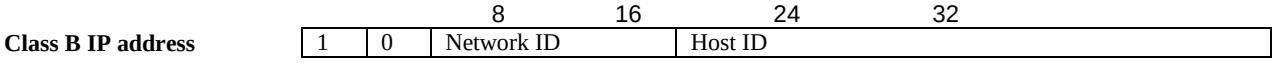
Class A: There are totally 128 Class A IP addresses. The most significant bit is 0, followed by seven bits representing a network ID and 24 bits representing a host ID.

Figure 1-1



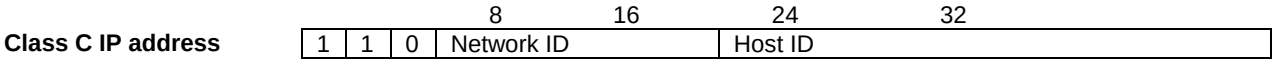
Class B: There are totally 16,384 Class B IP addresses. The two most significant bits are 10, followed by 14 bits representing a network ID and 16 bits representing a host ID.

Figure 1-2



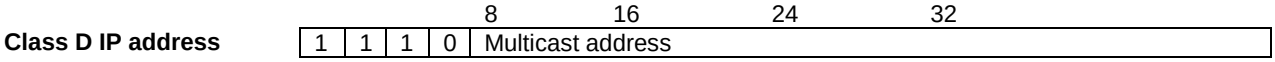
Class C: There are totally 2,097,152 Class C IP addresses. The three most significant bits are 110, followed by 21 bits representing a network ID and 8 bits representing a host ID.

Figure 1-3



Class D: The four most significant bits are 1110 and the other bits are a multicast IP address.

Figure 1-4



Note

An IP address whose four most significant bits are 1111 is prohibited. This type of IP addresses, also called Class E IP addresses, is reserved.

When you build up a network, plan IP addresses according to the real network environment. To make the network connect to the Internet, you need to apply for IP addresses from a central authority, for example, RIPE NCC (фр. Réseaux IP Européens + англ. Network Coordination Centre). Internet Corporation for Assigned Names and Numbers (ICANN) is the ultimate authority responsible for IP address allocation. However, for a private network, you do not need to apply for IP addresses. It is recommended that you assign private IP addresses for hosts in a private network.

The following table lists reserved and available addresses.

Class	Address Space	Status
Class A	0.0.0.0	Reserved
	1.0.0.0 to 126.0.0.0	Available
	127.0.0.0	Reserved
Class B	128.0.0.0 to 191.254.0.0	Available
	191.255.0.0	Reserved
Class C	192.0.0.0	Reserved
	192.0.1.0 to 223.255.254.0	Available

Class	Address Space	Status
	223.255.255.0	Reserved
Class D	224.0.0.0 to 239.255.255.255	Multicast
Class E	240.0.0.0 to 255.255.255.254	Reserved
	255.255.255.255	Broadcast

Three blocks of IP addresses are reserved for private networks and are not used on the Internet. Address translation is required for a private network using one of these IP addresses to access the Internet. The following table lists these addresses, which are defined in RFC 1918.

Class	Address Space	Status
Class A	10.0.0.0 to 10.255.255.255	1
Class B	172.16.0.0 to 172.31.255.255	16
Class C	192.168.0.0 to 192.168.255.255	256

For information on the assignment of IP addresses, TCP/UDP ports and other codes, refer to RFC 1166.

1.1.2. IP Address Configuration Task List

The IP address configuration task list includes the following tasks. Only the first one is required, and the others are optional depending on your network requirements.

1.1.2.1. Assigning IP Addresses to Interfaces

Only hosts configured with IP addresses can receive and send IP packets. If an interface is configured with an IP address, this means that the interface supports the IP protocol.

Use the following commands to assign an IP address to an interface in interface configuration mode.

Command	Function
Qtech(config-if)# **ip address** *ip-address mask*	Assigns an IP address to an interface.
Qtech(config-if)# **no ip address**	Removes the IP address configured for the interface.

A 32-bit mask identifies the network part of an IP address. In a mask, the IP address bit corresponding to 1 represents the network ID and the IP address bit corresponding to 0 represents the host ID. For example, the mask corresponding to a Class A IP address is 255.0.0.0. You can subdivide a network into multiple segments by using the mask. The goal of subnet definition is to use some bits of the host address of an IP address as the network address to reduce the number of hosts and increase the number of networks. At this point, the mask is called a subnet mask.


Note Theoretically, any bit of the host address of an IP address can be used as a subnet mask.

☑ Qtech products only support continuous subnet masks from left to right starting from the network ID.

You can configure the following features related to IP addresses in a task list. These tasks are optional depending on your actual needs.

Assigning Multiple IP Addresses to an Interface

Qtech products support assigning multiple IP addresses to an interface. Of the assigned IP addresses, one is the primary IP address and the others are secondary IP addresses. Theoretically, you can configure secondary addresses as many as you wish. A secondary IP address, however, must reside in different networks from the primary IP address or the other secondary IP addresses. The secondary IP address will be used frequently during the building of a network, for example, in the following cases:

- There are not enough host addresses for a network. For example, a LAN requires a Class C IP address to support up to 254 hosts. However, when there are more than 254 hosts in the LAN, another Class C IP address is necessary. Therefore, a host needs to connect two networks and thus needs configuring multiple IP addresses.
- Many older networks were built based on Layer 2 bridges without subnet definition. The use of secondary IP

addresses makes it easy to upgrade such a network to an IP-based routing network. An IP address is assigned for every device in a subnet.

- Two subnets of a network might be separated by another network. By creating a subnet in each separated subnets, you can connect the two separated subnets together by assigning secondary IP addresses. One subnet cannot appear on two or more interfaces in a device.

Caution

Before configuring a secondary IP address, you need to confirm that the primary IP address has been configured. If a secondary IP address is configured on one device of the network, the secondary IP addresses configured for other devices of the network must be located in the same network. If an IP address is not assigned to other devices, you can configure the IP address as the primary IP address for a device.

Use the following command to assign a secondary IP address to an interface in interface configuration mode.

Command	Function
Qtech(config-if)# **ip address** *ip-address mask* **secondary**	Assigns a secondary IP address to the interface.
Qtech(config-if)# **no ip address** *ip-address mask* **secondary**	Removes the secondary IP address configured for the interface.

Configuring the Management IP Address and the Gateway Together

Qtech Layer-2 switches allow you to configure the management IP address and the gateway by using the same command. Generally, Layer-2 switches provide the **ip default-gateway** command to configure a default gateway. Sometimes, the Layer-2 switch supports remote management via Telnet, and the management IP address and default gateway of the Layer-2 switch must be modified. In such a case, configuring either the **IP address** command or the **IP default-gateway** command will prevent you from configuring the other command (because the configuration has changed and this device can no longer be accessed via the network). Therefore, use the **gateway** keyword of the **IP address** command to modify the management IP address and default gateway.

Caution

This command is only supported on Layer-2 devices.

Use the following commands to configure the management IP address and the gateway at the same time in interface configuration mode.

Command	Function
Qtech(config-if)# **ip address** *ip-address mask* **gateway** *ip-address*	Configures the management IP address and gateway.
Qtech(config-if)# **no ip address** *ip-address mask* **gateway**	Removes the configured management IP address and gateway.

1.1.2.2. Configuring the Address Resolution Protocol (ARP)

Every device in a LAN has two addresses: a local address and a network address. The local address is contained in the header of a frame at the data link layer. More exactly, it is a data link layer address. Since this local address is handled in the MAC sub-layer of the data link layer, it is normally called an MAC address representing an IP network device on a network. The network address represents a device on the Internet and indicates the network to which the device belongs.

For mutual communication, a device in a LAN must know the 48-bit MAC address of the other device. It can obtain the MAC address according to an IP address. This process is called ARP. The reversed ARP (RARP) can resolve the IP address based on a MAC address. You can resolve an address in two ways: ARP and proxy ARP. For more information on ARP, proxy ARP and RARP, refer to RFC 826, RFC 1027, and RFC 903.

ARP binds an IP to an MAC address. It can resolve the MAC address according to an IP address. Then, the relationship between the IP address and the MAC address is stored in the ARP cache. With the MAC address, a device can encapsulate the frames of the data link layer and send them to the LAN in the Ethernet II-type by default. However, the frames can also be encapsulated into other types of Ethernet frames, such as SNAP.

The principle of RARP is similar to ARP. RARP resolves the IP address according to an MAC address. RARP is applied on diskless workstations in general.

Normally, a device can work without any special address resolution configuration. Qtech products can manage address resolution by the following configuration:

Configuring ARP Statically

ARP offers dynamic IP-to-MAC address mapping. It is not necessary to configure ARP statically in most cases. By configuring ARP Statically, Qtech products can respond to ARP requests from other IP addresses.

Use the following commands to configure static ARP in global configuration mode.

Command	Function
Qtech(config)# **arp** *ip-address mac-address arp-type*	Defines static ARP. Currently, Only *arp-type* can be set to **arpa** only.
Qtech(config)# **no arp** *ip-address*	Removes static ARP.

Setting ARP Encapsulation

Currently Qtech products only support Ethernet II ARP encapsulation, which also known as the **ARPA** keyword.

Setting ARP Timeout

ARP timeout takes effect for only the dynamically-learned IP-to-MAC address mapping. The shorter the timeout, the truer the mapping table saved in the ARP cache, but the more network bandwidth the ARP occupies. Hence the advantages and disadvantages should be weighted. Generally it is not necessary to configure the ARP timeout period unless otherwise stated.

Use the following commands to configure the ARP timeout period in interface configuration mode.

Command	Function
Qtech(config-if)# **arp timeout** *seconds*	Configures the ARP timeout period in the range from 0 to 2147483, where 0 indicates no aging.
Qtech(config-if)# **no arp timeout**	Removes the configuration.

By default, the timeout period is 3600 seconds.

Disabling IP Routing

IP routing is enabled by default. Do not execute this command unless you are sure that IP routing is not needed. Disabling IP routing will make the device lose all routes and the route forwarding function.

Run the following commands to disable IP routing in global configuration mode.

Command	Function
Qtech(config)# **no ip routing**	Disables IP routing.
Qtech(config)# **ip routing**	Enables IP routing

1.1.2.3. Configuring Broadcast Packet Processing

A broadcast packet is destined to all hosts in a physical network. Qtech products support two kinds of broadcast packets: directed broadcast and flooding. A directed broadcast packet is sent to all the hosts in a specific network, with the host ID of the destination IP addresses set to all-1s. While a flooding broadcast packet is sent to all the hosts in all networks, with all 32 bits of the destination IP address set to 1s. Broadcast packets are currently misused by some protocols, including the Internet protocol. Therefore, it is a basic responsibility for a network administrator to manage and control broadcast packets.

Forwarding flooding broadcast packets may make the network overburden and thus affect network operation. This is known as broadcast storm. There are some ways to suppress and restrict broadcast storm in the local network. However, Layer 2 network devices like bridges and switches will forward and spread broadcast packets.

The best solution to solve broadcast storm is to specify a broadcast address for each network, that is, to implement directed broadcast. This requires the IP protocol to use directed broadcast instead of flooding broadcast as much as possible.

For details about broadcast, refer to RFC 919 and RFC 922.

You can configure the following features in a task list to process broadcast packets. These tasks are optional depending on actual network needs.

Enabling Directed Broadcast-to-Physical Broadcast Conversion

A directed broadcast IP packet is one destined to the broadcast address of an IP subnet. For instance, the packet destined to 172.16.16.255 is a directed broadcast packet. However, the node that generates this packet is not a member of the destination subnet.

Upon receipt of a directed broadcast IP packet, the device indirectly connecting the destination subnet will forward the packet in the same way as forwarding a unicast packet. After the directed broadcast IP packet arrives at a device directly connecting the subnet, the device transforms the packet into a flooding broadcast IP packet (whose destination address is all-1s in general), and then send it to all the hosts within the destination subnet by means of broadcast at the link layer.

Enabling the conversion from directed broadcast to physical broadcast on an interface allows the interface to forward directed broadcast IP packets to the directly connected network. This command only affects the transmission of the directed broadcast IP packets to the final destination subnet, but not other directed broadcast packets.

You can control the forwarding of directed broadcast IP packets as required on an interface by defining ACLs. Only those IP packets matching the ACLs will experience the conversion from directed broadcast to physical broadcast.

Use the following commands to configure directed broadcast-to-physical broadcast conversion in interface configuration mode.

Command	Function
Qtech(config-if)# **ip directed-broadcast** [*access-list-number*]	Enables directed broadcast to physical broadcast conversion on the interface.
Qtech(config-if)# **no ip directed-broadcast**	Disables the conversion.

Establishing an IP Broadcast Address

Currently, the most popular way is the destination address consisting of all 1s (255.255.255.255). Qtech products can be configured to generate other IP broadcast addresses and receive all types of IP broadcast packets.

Use the following commands to set a broadcast IP address other than 255.255.255.255 in interface configuration mode.

Command	Function
Qtech(config-if)# **ip broadcast-address** *ip-address*	Creates a broadcast address.
Qtech(config-if)# **no ip broadcast-address**	Removes the configuration.

1.1.3. Monitoring and Maintaining IP Addresses

You can monitor and maintain networks according to the following task list. These tasks are optional depending on your actual needs.

1.1.3.1. Clearing Caches and Table Contents

You can remove all contents of a particular cache, table, or database, including

- Clearing the ARP cache
- Clearing the table of mapping between host names and IP addresses
- Clearing the routing table

Command	Function
Qtech# **clear arp-cache**	Clears the ARP cache.
Qtech# **clear ip route** {*network* [*mask*] | *}}	Clears the IP routing table.

1.1.3.2. Displaying System and Network Status

You can show the contents of the IP routing table, cache, and database. Such information is very helpful in network troubleshooting. You also can display information about network reachability of a local device and determine the routing path that the packets of your device are taking after leaving the device.

Use the following commands to display system and network status information in privileged user mode.

Command	Function
Qtech# **show arp** [[*ip* [*mask*] | *mac-address*] | **static** | **complete** | **incomplete**]	Shows the ARP cache table. The **complete** keyword is used to show resolved entries of dynamic ARP, and the **incomplete** keyword is used to show unresolved entries of dynamic ARP.
Qtech# **show ip arp**	Shows the IP ARP cache table.

Command	Function
Qtech# **show ip interface** [*interface-type interface-number*]	Shows interface IP address information.
Qtech# **show ip route** [*network* [*mask*]]	Shows the routing table.
Qtech#**show ip route**	Shows brief information about the routing table.
Qtech# **ping** *ip-address* [**length** *bytes*] [**ntimes** *times*] [**timeout** *seconds*]	Tests network reachability.

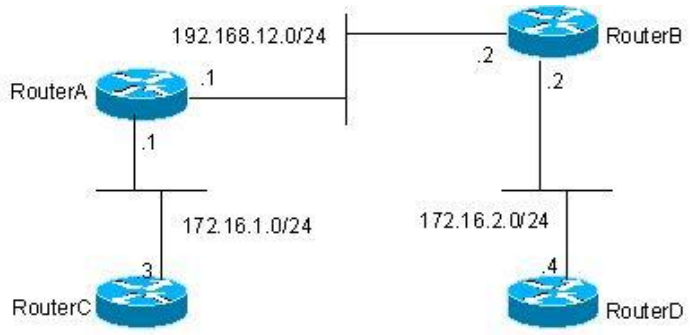
1.1.4. IP Address Configuration Examples

1.1.4.1. Example of Configuring Secondary IP Addresses

Configuration Requirements

The following figure shows IP address assignment and network device connections.

Figure 1-5 Examples of configuring secondary IP addresses.



Configure RIP. You can set the version to RIPv1 only. You can see the routes of 172.16.2.0/24 on router C and the routes of 172.16.1.0/24 on router D.

Configuration of the Routers

RIPv1 does not support classless routes. This means masks are not carried in route advertisements. 172.16.1.0/24 and 172.16.2.0/24 that belong to the same network are separated by the Class C network 192.168.12.0/24. Generally, router C and router D cannot learn detailed routes from each other. According to a feature of RIP, the mask of the route to be received should be set to the same value as that of the interface network if the route and the interface network belong to the same network. By configuring routers A and B, you can build a secondary network 172.16.3.0/24 on the network 192.168.12.0/24 to link the two separated subnets. The following presents a configuration description of routers A and B only.

Router A:

```
interface FastEthernet 0/0
ip address 172.16.3.1 255.255.255.0 secondary
ip address 192.168.12.1 255.255.255.0
!
interface FastEthernet 0/1
ip address 172.16.1.1 255.255.255.0
!
router rip
network 172.16.0.0
network 192.168.12.0
```

Router B:

```
interface FastEthernet 0/0
ip address 172.16.3.2 255.255.255.0 secondary
ip address 192.168.12.2 255.255.255.0
!
interface FastEthernet 0/1
ip address 172.16.2.1 255.255.255.0
!
router rip
network 172.16.0.0
```

```
network 192.168.12.0
```

1.2. Configuring the IP Service

1.2.1. IP Configuration Task List

IP service configuration covers the following configuration tasks. These tasks are optional depending on your actual needs.

1.2.2. Configuring the Default Gateway

☑ Run the command only on L2 devices.

If no destination IP address to which a packet will be sent is specified, the packet will be sent to the default gateway by default. Use the **show ip redirects** command to view the settings.

Use the following command to set the default gateway in global configuration mode. Use the **no** form of this command to remove the default gateway.

Command	Function
ip default-gateway *ip-address*	Sets the default gateway.

Use the following command to view the configured default gateway.

Command	Function
show ip redirects	Displays the default gateway.

1.2.3. Managing IP Connections

The IP protocol stack offers a number of services to control and manage IP connections. The Internet Control Message Protocol (ICMP) provides many of these services. Once a network problem occurs, a device or access server will send an ICMP message to the host or other devices. For detailed information on ICMP, see RFC 792.

Enabling the ICMP Destination Unreachable Message

When a router receives a non-broadcast packet destined to it and this packet uses an IP protocol that it cannot handle, it will return an ICMP destination unreachable message to the source address. Similarly, if the router is unable to forward the packet because it knows of no route to the destination address, it sends an ICMP host unreachable message. This feature is enabled by default.

Use the following command to enable or disable ICMP host unreachable messages in interface configuration mode.

Command	Function
Qtech(config-if)# **ip unreachables**	Enables ICMP protocol unreachable and host unreachable messages.
Qtech(config-if)# **no ip unreachables**	Disables ICMP protocol unreachable and host unreachable messages.

Enabling the ICMP Redirect Message

Routes are sometimes less than optimal. For example, it is possible for the device to be forced to resend a packet through the same interface on which the packet was received. If the device resends a packet through the receiving interface, it sends an ICMP redirect message to the originator of the packet, telling the originator that the gateway to this destination address is another device in the same subnet. Therefore, the originator will transmit subsequent packets based on the optimized path. This feature is enabled by default.

Use the following commands to enable or disable the ICMP redirect message in interface configuration mode.

Command	Function
Qtech(config-if)# **ip redirects**	Enables the ICMP redirect message. It is enabled by default.
Qtech(config-if)# **no ip redirects**	Disables the ICMP redirect message.

Enabling the ICMP Mask Reply Message

Occasionally, a network device needs to know the mask of a subnet on the Internet. To obtain this information, the device can send an ICMP mask request message. The device receiving the request will return an ICMP mask reply message. Qtech products can respond to the ICMP mask request message. This function is enabled by default.

Use the following commands to enable or disable the ICMP mask reply message in interface configuration mode.

Command	Function
Qtech(config-if)# **ip mask-reply**	Enables the ICMP mask reply message.
Qtech(config-if)# **no ip mask-reply**	Disables the ICMP mask reply message.

Setting the IP MTU

All interfaces have a default MTU value. All packets which are larger than the MTU have to be fragmented before they are sent on an interface. Otherwise the packets cannot be forwarded on the interface.

Qtech products allow you to adjust the MTU on an interface. Changing the MTU value can affect the IP MTU value, and the IP MTU value will be modified automatically to match the new MTU. However, changing the IP MTU value has no influence on the MTU value of the interface.

The interfaces of a device in a physical network should have the same MTU for the same protocol.

Use the following commands to set the IP MTU in interface configuration mode.

Command	Function
Qtech(config-if)# **ip mtu** *bytes*	Sets the MTU in the range from 68 to 1500 bytes.
Qtech(config-if)# **no ip mtu**	Restores the default MTU settings.

Configuring IP Source Routing

Qtech products support IP source routing. Upon receiving an IP packet, the device will check its IP header like strict source route, loose source route and recorded route, which are defined in RFC 791. If one of these options is enabled, the device performs an appropriate action. Otherwise, it sends an ICMP error message to the source and then discards the packet. Qtech products support IP source routing by default.

Use the following commands to enable or disable IP source routing in interface configuration mode.

Command	Function
Qtech(config)# **ip source-route**	Enables IP source routing.
Qtech(config)# **no ip source-route**	Disables IP source routing.

2. CONFIGURING VRF

2.1. VRF Overview

Virtual Private Networks (VPNs) provide a secure way to share bandwidths in the backbone networks of ISPs. One VPN is the collection of the sites sharing routes. The sites connect to the service provider's network through one or multiple interface links, with one VPN routing table associated with one interface. The VPN routing table is also referred to as a VPN routing or forwarding (VRF) table.

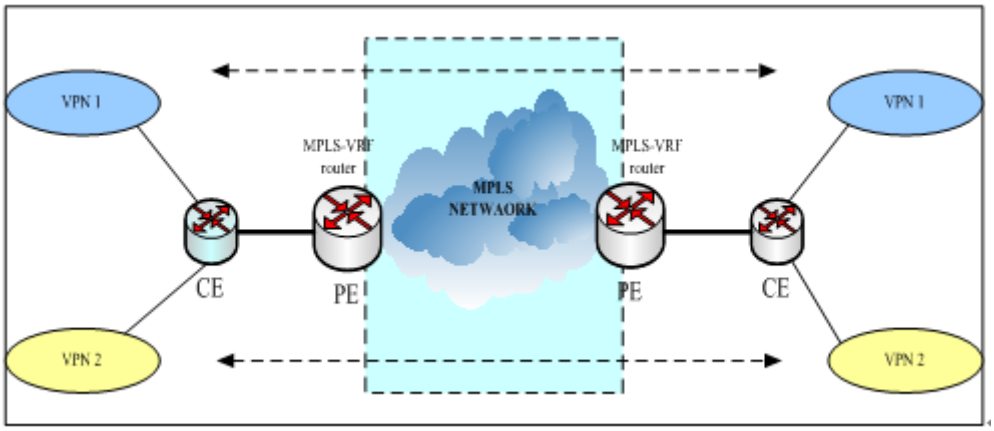

Caution

Qtech RSRs support the VRF-lite feature, which is also known as the multi-VRF CE or multi-VRF Customer Edge Device, and can implement multiple VPN route forwarding instances when serving as CEs.

2.2. Working Principles of VRF-Lite

The working principles of VRF-lite are as follows:

- CEs provide multiple access channels to PEs to support user access. CEs advertise local routes to PEs, and learns VPN remote routes from PEs.
- PEs exchange routing information with CEs via static routing and dynamic routing protocols, such as BGP, RIP, and OSPF.
- PEs may have multiple interfaces that belong to one VPN. PEs exchange VPN routing information with each other via BGP.
- PEs do not rely on the functions of CEs.
- P devices do not process VPN information. That is, VPN information is transparent to P devices.



Typical Application Model of VRF-Lite

The packet processing flow is as follows when VRF-lite is enabled on a network:

- When a CE receives a packet from a VPN, it checks interface information to query the related VRF table. If a route is successfully found in the VRF table, the CE sends the packet to a PE.
- When the ingress PE receives a packet from a CE, it queries the VRF table. If a route is successfully found in the VRF table, the ingress PE adds an MPLS label to the route and sends the packet to the MPLS network.
- When the egress PE receives an MPLS packet from the MPLS network, it removes the MPLS label and finds the related VPN routing table. The egress PE searches for a common route. If a route is successfully found, the egress PE sends the packet to the related adjacency.
- When a CE receives a packet from the egress PE, it checks the inbound interface of the packet to obtain the related VPN routing table and then searches for a route. If a route is successfully found, the packet is sent to the VPN.

2.3. Multi-Protocol VRF

You can create only single-protocol VRF supporting IPv4 in version earlier than 10.4(3). Use the **ip vrf** command to create a single-protocol IPv4 VRF. To enable single-protocol IPv4 VRF on a network interface, use the **ip vrf forwarding** command in interface configuration mode.

To support IPv6 VPNs, the 10.4(3) version has introduced multi-address-family VRFs. A multi-address-family VRF enables you to define multiple address families under the same VRF. You can apply a multi-address-family VRF to both IPv4 VPNs and IPv6 VPNs. A multi-address-family VRF is also called a multi-protocol VRF. You can use the **vrf definition** command to create a multi-protocol VRF. To enable a multi-protocol VRF on a network interface, use the **vrf forwarding** command in interface configuration mode.

It is not allowed to use the commands for configuring a single-protocol IPv4 VRF to configure a multi-protocol VRF, and vice versa.

For example, you can use the configuration command of a multi-protocol VRF to create a multi-protocol VRF named "vrf1".

```
Qtech(config)#vrf definition vrf1
```

If you try to use a configuration command of a single-protocol VRF to edit vrf1, the following prompting message will be displayed:

```
Qtech(config)#ip vrf vrf1
% Use 'vrf definition vrf1' command.
```

If you try to use a multi-protocol VRF command to edit the single-protocol VRF named "vrf2" which only supports IPv4, the following prompting message will be displayed:

```
Qtech(config)# vrf definition vrf2
% Use 'ip vrf vrf2' command.
```

The configuration commands (ip vrf and ip vrf forwarding) of single-protocol IPv4 VRFs will be reserved for a period of time until they are abandoned.

2.4. Configuring VRF-Lite

You can configure VRF-Lite as follows:

- A CE supports multiple users via VRF-Lite. Each user has its own routing table on the CE.
- Since each user has its own routing table, they may use the same IP address. This function is temporarily not supported.
- Multiple users share the physical line between a CE and a PE, and there are multiple logical interfaces on the physical line. The physical line can be implemented by multiple means.
- VRF-lite does not support functions related to MPLS-VRF. It mostly acts on CEs.
- For a PE, connecting it to multiple CEs does not differ from using VRF-lite on it.
- EBGP is recommended for route exchange between a PE and a CE. Of course, OSPF, RIP, and static routing protocols may also be used for route exchange, but that would be more complex. If the OSPF protocol is used for route exchange, you need to exercise caution during configuration. It is recommended that you enable the capability VRF-lite function when using the OSPF protocol for route exchange.

2.4.1. Creating VRF

Use the following commands to create a single-protocol IPv4 VRF.

Command	Function
Qtech(config)# **ip vrf** *vrf-name*	Creates a VRF. vrf-name shall not exceed 31 characters.
Qtech(config)# **no ip vrf** *vrf-name*	Deletes the VRF.

Use the following commands to create a multiprotocol VRF.

Command	Function
Qtech(config)# **vrf definition** *vrf-name*	Creates a multi-protocol VRF. *vrf-name* shall not exceed 31 characters.
Qtech(config-vrf)# **address-family ipv4**	Configures an IPv4 address family, namely to enable the IPv4 protocol of VRFs. No IPv4 address family is configured by default.
Qtech(config-vrf-af)# **exit-address-family**	Exits VRF address-family configuration sub-mode.
Qtech(config-vrf)# **address-family ipv6**	Configures an IPv6 address family, namely to enable the IPv6 protocol of VRFs. No IPv6 address family is configured by default.
Qtech(config-vrf-af)# **exit-address-family**	Exits VRF address-family configuration sub-mode.

2.4.2. Configuring a VRF Descriptor

Command	Function
Qtech(config-vrf)# **description** *string*	Configures a VRF descriptor, which contains at most 244 characters.

The following example configures a VRF descriptor named “vpn-a” for a single-protocol IPv4 VRF named “vrf1”.

```
Qtech(config)#ip vrf vrf1
Qtech(config-vrf)#description vpn-a
```

The following example configures a VRF descriptor named “vpn-b” for a multiprotocol VRF named “vrf2”.

```
Qtech(config)#vrf definition vrf2
Qtech(config-vrf)#description vpn-b
```

2.4.3. Enabling VRF on Interfaces

Command	Function
Qtech(config-if)# **ip vrf forwarding** *vrf-name*	Binds an interface to a single-protocol IPv4 VRF. Note that *vrf-name* cannot be a multiprotocol VRF. If the IPv6 function does not need to be enabled on an interface, you can bind the interface to a single-protocol IPv4 VRF. If the IPv6 function needs to be enabled on an interface, it is not recommended that you use this command to bind the interface to a single-protocol IPv4 VRF.

Qtech(config-if)# **no ip vrf forwarding** *vrf-name*	Cancels the binding between an interface and a single-protocol IPv4 VRF.
Qtech(config-if)# **vrf forwarding** *vrf-name*	Binds an interface to a multiprotocol VRF. Note that *vrf-name* cannot be a single-protocol IPv4 VRF. If the IPv6 function needs to be enabled on an interface, it is recommended that you bind the interface to a multiprotocol VRF instead of a single-protocol IPv4 VRF.
Qtech(config-if)# **no vrf forwarding** *vrf-name*	Cancels the binding between an interface and a multiprotocol VRF.

An interface does not belong to any VRF by default. That is, global routing applies.

Caution After you bind an interface to a single-protocol IPv4 VRF, the IPv4 address originally configured for the interface will be invalid. The binding does not affect the IPv6 address configured for the interface. In general, enable VRF first on the interface and then configure an IPv4 address.

Caution If you bind an interface to a single-protocol IPv4 VRF and enables the IPv6 protocol on the interface, the switch cannot forward IPv6 packets received on the interface. Therefore, it is recommended that you use the multiprotocol VRF if you want to bind the interface to a VRF and enable the IPv6 protocol on the interface at the same time.

Caution You cannot bind an interface to a multiprotocol VRF not configured with any address family.

Caution If you bind an interface to a multiprotocol VRF, all existing IPv4, IPv6, VRRP IPv4, and VRRP IP6 addresses configured for the interface will be deleted. In addition, the IPv6 protocol will be disabled on the interface.

Caution If you bind an interface to a multiprotocol VRF not configured with any IPv4 address family, you cannot configure any IPv4 or VRRP IPv4 address for the interface. Before configuring an IPv4 or VRRP IPv4 address for the interface, you needs to configure an IPv4 address family for the multiprotocol VRF.

Caution If you bind an interface to a multiprotocol VRF not configured with any IPv6 address family, you cannot configure any IPv6 or VRRP IPv6 address for the interface. Before configuring an IPv6 or VRRP IPv6 address for the interface, you needs to configure an IPv6 address family for the multiprotocol VRF.

Caution If you delete the IPv4 address family configured for a multiprotocol VRF, all IPv4 and VRRP IPv4 addresses of all network interfaces bound to this VRF will be deleted, so will the static IPv4 routes whose routing VRF or next-hop VRF is this VRF. If you delete the IPv6 address family configured for a multiprotocol VRF, all IPv6 and VRRP IPv6 addresses of all network interfaces bound to this VRF will be deleted, the IPv6 protocol will be disabled on the interfaces, and all the static IPv6 routes whose routing VRF or next-hop VRF is this VRF will be deleted.

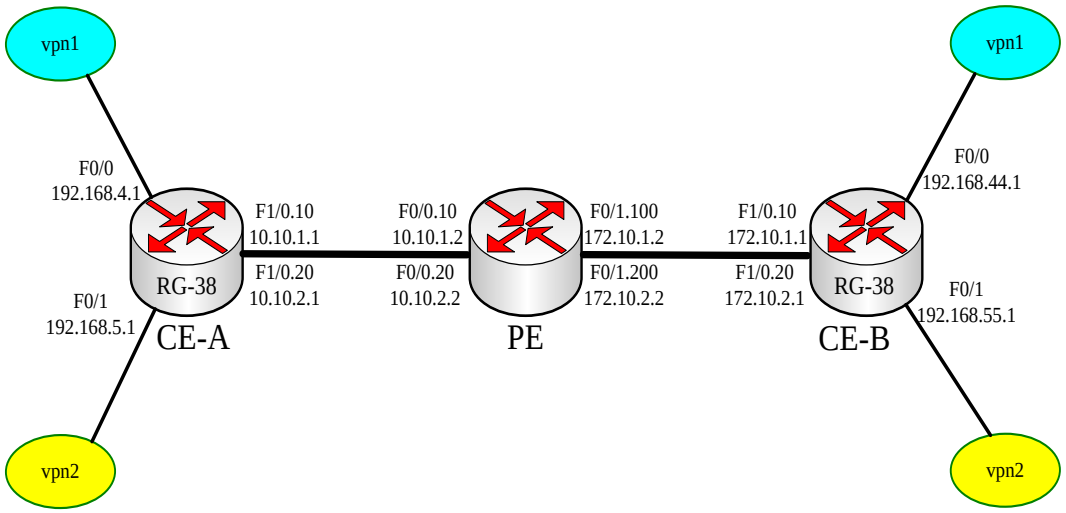
2.4.4. Configuring VRF Routes

Command	Function
Qtech(config)# **ip route vrf** *vrf-name network mask interface nexthop*	Adds a route.
Qtech(config)# **no ip route vrf** *vrf-name network mask*	Deletes a route.

You can also use routing protocols to configure routes.

2.5. VRF-Lite Configuration Examples

Example 1: As shown in the following figure, Qtech devices serve as CEs and PEs. The CEs access two VPNs named “vpn1” and “vpn2”.




```
Qtech# hostname CE-A
# Name the router.
CE-A# configure terminal
# Enter global configuration mode.
CE-A(config)# ip vrf vpn1
# Create a VRF named “vpn1”.
CE-A(config)# ip vrf vpn2
# Create a VRF named “vpn2”.
CE-A(config)# interface f0/0
# Enter interface configuration mode.
CE-A(config-if)#description  connecting-to-vpn1
# Link to vpn1.
CE-A(config-if)# ip vrf forwarding vpn1
# Enable VRF on the interface.
CE-A(config-if)# ip address 192.168.4.1 255.255.255.0
# Configure an IP address.
CE-A(config)# interface f0/1
# Enter interface configuration mode.
CE-A(config-if)# ip vrf forwarding vpn2
# Enable VRF on the interface.
CE-A(config-if)# ip address 192.168.5.1 255.255.255.0
# Configure an IP address.
CE-A(config-if)#description  connecting-to-vpn2
# Link to vpn2.
CE-A(config)# interface f1/0
# Enter interface configuration mode.
CE-A(config-if)#  no ip address
CE-A(config)# interface f1/0.10
# Enter a subinterface.
CE-A(config-if)# encapsulation dot1Q 10
# Encapsulate 802.1Q in VLAN 10.
CE-A(config-if)# ip vrf forwarding vpn1
# Enable VRF on the interface.
```

```
CE-A(config-if)# ip address 10.10.1.1 255.255.255.0
# Configure the IP address of the subinterface.
CE-A(config)# interface f1/0.20
# Enter a subinterface.
CE-A(config-if)# encapsulation dot1Q 20
# Encapsulate 802.1Q in VLAN 20.
CE-A(config-if)# ip vrf forwarding vpn2
# Enable VRF on the interface.
CE-A(config-if)# ip address 10.10.2.1 255.255.255.0
# Configure the IP address of the subinterface.
CE-A(config)# ip route vrf vpn1 192.168.44.0 255.255.255.0 10.10.1.2
# Configure a static route of vpn1.
CE-A(config)# ip route vrf vpn1 192.168.55.0 255.255.255.0 10.10.2.2
# Configure a static route of vpn2.
Qtech# hostname CE-B
# Name the router.
CE-B# configure terminal
# Enter global configuration mode.
CE-B(config)# ip vrf vpn1
# Create a VRF named “vpn1”.
CE-B(config)# ip vrf vpn2
# Create a VRF named “vpn2”.
CE-B(config)# interface f0/0
# Enter interface configuration mode.
CE-Bconfig-if)# ip vrf forwarding vpn1
# Enable VRF on the interface.
CE-B(config-if)# ip address 192.168.44.1 255.255.255.0
# Configure an IP address.
CE-B(config-if)# description  connecting-to-vpn1
# Link to vpn1.
CE-B(config)# interface f0/1
# Enter interface configuration mode.
CE-B(config-if)# ip vrf forwarding vpn2
# Enable VRF on the interface.
CE-B(config-if)# ip address 192.168.55.1 255.255.255.0
# Configure an IP address.
CE-Bconfig-if)# description  connecting-to-vpn2
# Link to vpn2.
CE-B(config)# interface f1/0
# Enter interface configuration mode.
CE-B(config-if)#  no ip address
CE-B(config)# interface f1/0.10
# Enter a subinterface.
CE-B(config-if)# encapsulation dot1Q 100
# Encapsulate 802.1Q in VLAN 100.
CE-B(config-if)# ip vrf forwarding vpn1
# Enable VRF on the interface.
CE-B(config-if)# ip address 172.10.1.1 255.255.255.0
# Configure the IP address of the subinterface.
CE-B(config)# interface f1/0.20
# Enter a subinterface.
CE-B(config-if)# encapsulation dot1Q 200
# Encapsulate 802.1Q in VLAN 200.
CE-B(config-if)# ip vrf forwarding vpn2
# Enable VRF on the interface.
CE-B(config-if)# ip address 172.10.2.1 255.255.255.0
# Configure the IP address of the subinterface.
CE-B(config)# ip route vrf vpn1 192.168.4.0 255.255.255.0 172.10.1.2
# Configure a static route of vpn1.
CE-B(config)# ip route vrf vpn1 192.168.5.0 255.255.255.0 172.10.2.2
# Configure a static route of vpn2.
# Next, configure PEs.
Router# configure terminal
```

```
Router(config)# ip vrf v1
Router(config-vrf)# rd 100:1
Router(config-vrf)# route-target export 100:1
Router(config-vrf)# route-target import 100:1
Router(config-vrf)# exit
Router(config)# ip vrf v2
Router(config-vrf)# rd 100:2
Router(config-vrf)# route-target export 100:2
Router(config-vrf)# route-target import 100:2
Router(config-vrf)# exit
Router(config)# ip cef
Router(config)# interface FastEthernet0/0.10
Router(config-if)# encapsulation dot1q 10
Router(config-if)# ip vrf forwarding v1
Router(config-if)# ip address 10.10.1.2 255.255.255.0
Router(config-if)# exit
Router(config)# interface FastEthernet0/1.100
Router(config-if)# encapsulation dot1q 100
Router(config-if)# ip vrf forwarding v1
Router(config-if)# ip address 172.10.1.2 255.255.255.0
Router(config-if)# exit
Router(config)# interface FastEthernet0/0.20
Router(config-if)# encapsulation dot1q 20
Router(config-if)# ip vrf forwarding v2
Router(config-if)# ip address 10.10.2.2 255.255.255.0
Router(config-if)# exit
Router(config)# interface FastEthernet0/1.200
Router(config-if)# encapsulation dot1q 200
Router(config-if)# ip vrf forwarding v2
Router(config-if)# ip address 172.10.2.2 255.255.255.0
Router(config-if)# exit
Router(config)# ip route vrf v1 192.168.4.0 255.255.255.0 10.10.1.1
Router(config)# ip route vrf v1 192.168.44.0 255.255.255.0 172.10.1.1
Router(config)# ip route vrf v2 192.168.5.0 255.255.255.0 10.10.2.1
Router(config)# ip route vrf v2 192.168.55.0 255.255.255.0 172.10.2.1


Example 2 : Configure a simple multiprotocol VRF.
# Create a multiprotocol VRF.
Qtech(config)#vrf definition multi-af-vrf-example
# Configure a VRF descriptor.
Qtech(config-vrf)#description This-is-an-example
# Configure an IPv4 address family.
Qtech(config-vrf)#address-family ipv4
Qtech(config-vrf-af)#exit-address-family
# Configure an IPv6 address family.
Qtech(config-vrf)#address-family ipv6
Qtech(config-vrf-af)#exit-address-family
Qtech(config-vrf)#interface VLAN 1
# Bind VLAN 1 to a multiprotocol VRF.
Qtech(config-if-VLAN 1)#vrf forwarding multi-af-vrf-example
Qtech(config-if-VLAN 1)#ip address 1.1.1.1 255.255.255.0
Qtech(config-if-VLAN 1)#ipv6 address 1000::1/64
Qtech(config-if-VLAN 1)#exit
# Configure a static IPv4 or IPv6 route of the multiprotocol VRF.
Qtech(config)#ip route vrf multi-af-vrf-example 0.0.0.0 0.0.0.0 1.1.1.2
Qtech(config)#ipv6 route vrf multi-af-vrf-example ::/0 1000::2
```

2.6. VRF-Lite Debugging

Use the following command to check the routing table of a VRF.

Command	Function
show ip route vrf *vrf-name*	Displays the routes of the specified VRF.

For details about the command syntax, see the *VRF Command Reference*.

Use the following command to clear the routing table of a VRF.

Command	Function
clear ip route vrf *vrf-name* *	Clears the routes of the specified VRF.

For details about the command syntax, see the *VRF Command Reference*.

Use the following commands to check information about a VRF in the system.

Command	Function
show ip vrf [*vrf-name*]	Displays information about an IPv4 VRF.
show vrf [brief] [vrf-name]	Displays brief information about a VRF.
show vrf ipv4 [*vrf-name*]	Displays brief information about an IPv4 VRF.
show vrf ipv6 [*vrf-name*]	Displays brief information about an IPv6 VRF.
show vrf detail [*vrf-name*]	Displays details about a VRF.

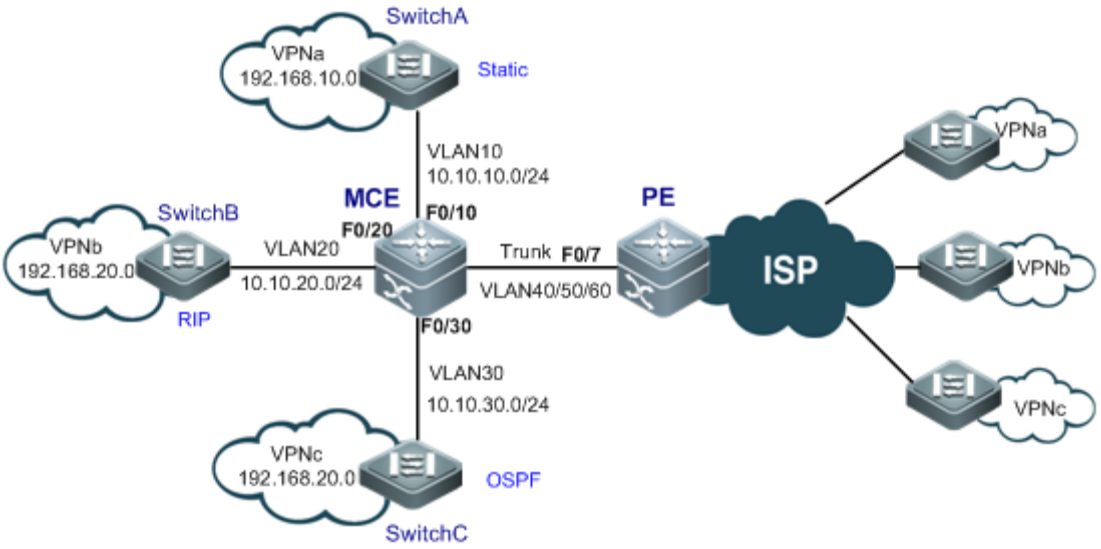
For details about the command syntax, see the *VRF Command Reference*.

2.7. MCE Configuration Example

2.7.1. Networking Topology

As shown in the following figure, VPNa, VPNb, and VPNc are located at different sites and exchange information across the backbone network.

- VPN sites access a PE via a MCE.
- Static routes are configured between the MCE and VPNa. RIP is configured between the MCE and VPNb to exchange routing information. The OSPF protocol is configured between the MCE and VPNc to exchange routing information.



Typical Topology of MCE Application

2.7.2. Networking Requirements

The MCE needs to isolate the routes of one VPN from the routes of the other VPNs. It advertises the routes of various VPNs to the PE via the static routing protocol, RIP, and the OSPF protocol respectively.

Duplicate addresses in a VPN must be supported.

2.7.3. Configuration Tips

Create multiple VRF instances on the MCE and the PE, so that the routes of different VPN services are isolated from one another. Perform the following two steps:

Configure VRF instances and bind them to various interfaces.

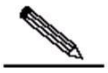
Interface	Bound VRF	Interface IP Address
Interface of the MCE that connects to VPNa (SVI 10)	VPNa	10.10.10.3
Logical interface of the MCE that connects to the PE (SVI 40)	VPNa	10.10.40.1
Logical interface of the PE that connects to the MCE (SVI 40)	VPNa	10.10.40.2
Interface of the MCE that connects to VPNb (SVI 20)	VPNb	10.10.20.3
Logical interface of the MCE that connects to the PE (SVI 50)	VPNb	10.10.50.1
Logical interface of the PE that connects to the MCE (SVI 50)	VPNb	10.10.50.2
Interface of the MCE that connects to VPNc (SVI 30)	VPNc	10.10.30.3
Logical interface of the MCE that connects to the PE (SVI 60)	VPNc	10.10.60.1
Logical interface of the PE that connects to the MCE (SVI 60)	VPNc	10.10.60.2

Configure route exchange between the MCE, VPN sites, and the PE.

VRF instances on the MCE cooperate with VRF instances on the PE to advertise VPN routes to the PE via the routing protocol between the MCE and the PE. Then the PE advertises the routes to other PEs on the network, so that each VPN site interworks with remote VPN sites on the network.

Note

In this example, VRF instances on the MCE exchange routes with VPN sites and the PE via the same routing protocol.



Note

If VRF instances on the MCE exchange routes with VPN sites and the PE via different routing protocols, to guarantee complete exchange of VPN routes, the routing protocols of VRF instances on the MCE must be redistributed, so that the MCE can advertise VPN routes to VRF instances on the PE and that the routes advertised by the PE to VRF instances on the MCE can be further advertised to VPN sites.

2.7.4. Configuration Steps

Configure VRF instances on the MCE and the PE, and bind VRF instances to interfaces.

Create VRF instances on the MCE.

Step 1: Create VRF instances named “VPNa”, “VPNb”, and “VPNc” on the MCE.

```
MCE(config)#ip vrf vpna
MCE(config-vrf)#exit
MCE(config)#ip vrf vpnb
MCE(config-vrf)#exit
MCE(config)#ip vrf vpnc
MCE(config-vrf)#exit
```

Step 2: Create VLAN 10, and add the FE interface 0/10 of the MCE that connects to Switch A to VLAN 10.

```
MCE(config)#interface fastEthernet 0/10
MCE(config-if-FastEthernet 0/10)#switchport access vlan 10
MCE(config-if-FastEthernet 0/10)#exit
```

Step 3: Bind the interface SVI 10 of VLAN 10 to VPNa, and set the IP address of the interface SVI 10 to 10.10.10.3/24.

```
MCE(config)#interface vlan 10
MCE(config-if-VLAN 10)#ip vrf forwarding vpna
MCE(config-if-VLAN 10)#ip address 10.10.10.3 255.255.255.0
MCE(config-if-VLAN 10)#exit
```

Step 4: Similarly, create VLAN 20 and VLAN 30 (see the previous two steps), bind the interface SVI 20 of the MCE that connects to Switch B to VPNb, and bind the interface SVI 30 of the MCE that connects to Switch C to VPNc. Set the IP address of the interface SVI 20 to 10.10.20.3, and the IP address of the interface SVI 30 to 10.10.30.3.

Step 5: Create VLAN 40, VLAN 50, and VLAN 60, and add the FE interface 0/7 of the MCE that connects to the PE to the three VLANs. Bind SVI 40 to VPNa, SVI 50 to VPNb, and SVI 60 to VPNc. Set the IP address of the interface SVI 40 to 10.10.40.1, the IP address of the interface SVI 50 to 10.10.50.1, and the IP address of the interface SVI 60 to 10.10.60.1.

```
MCE(config)#vlan 40
MCE(config-vlan)#exit
MCE(config)#vlan 50
MCE(config-vlan)#exit
MCE(config)#vlan 60
MCE(config-vlan)#exit
MCE(config)#interface fastEthernet 0/7
MCE(config-if-FastEthernet 0/7)#switchport mode trunk
MCE(config-if-FastEthernet 0/7)#exit
MCE(config)#interface vlan 40
MCE(config-if-VLAN 40)#ip vrf forwarding vpna
MCE(config-if-VLAN 40)#ip address 10.10.40.1 255.255.255.0
MCE(config-if-VLAN 40)#exit
MCE(config)#interface vlan 50
MCE(config-if-VLAN 50)#ip vrf forwarding vpnb
MCE(config-if-VLAN 50)#ip address 10.10.50.1 255.255.255.0
MCE(config-if-VLAN 50)#exit
MCE(config)#interface vlan 60
MCE(config-if-VLAN 60)#ip vrf forwarding vpnc
MCE(config-if-VLAN 60)#ip address 10.10.60.1 255.255.255.0
```

After the above steps are performed, VRF instances are created on the MCE.

Create VRF instances on the PE.

Step 1: Create VRF instances named “VPNa”, “VPNb”, and “VPNc” on the PE.

```
PE(config)#ip vrf vpna
PE(config-vrf)#exit
PE(config)#ip vrf vpnb
PE(config-vrf)#exit
PE(config)#ip vrf vpnc
PE(config-vrf)# exit
```

Step 2: Create VLAN 40, VLAN 50, and VLAN 60, and add the FE interface 0/7 of the PE that connects to the MCE to the three VLANs. Bind SVI 40 to VPNa, SVI 50 to VPNb, and SVI 60 to VPNc. Here, the operations for creating VLANs and adding interfaces to VLANs are omitted.

```
PE (config)#interface vlan 40
PE(config-if-VLAN 40)#ip vrf forwarding vpna
PE(config-if-VLAN 40)#ip address 10.10.40.2 255.255.255.0
PE(config-if-VLAN 40)#exit
PE(config)#interface vlan 50
PE(config-if-VLAN 50)#ip vrf forwarding vpnb
PE(config-if-VLAN 50)#ip address 10.10.50.2 255.255.255.0
PE(config-if-VLAN 50)#exit
PE(config)#interface vlan 60
PE(config-if-VLAN 60)#ip vrf forwarding vpnc
PE(config-if-VLAN 60)#ip address 10.10.60.2 255.255.255.0
```

After the above steps are performed, VRF instances are created on the PE.

Configure static routes between the MCE and VPNa, and between the MCE and the PE.

Configure a static route on Switch A (access switch for the site VPNa).

Set the IP address of the interface of Switch A that connects to the MCE to 10.10.10.2/24, and that of the interface of Switch A that connects to the site VPNa to 192.168.10.1/24. Here, the operations for adding ports to VLANs and setting interface IP addresses are omitted.

Configure a default route on Switch A, with the next hop of the outbound packet pointing to 10.10.10.3.

```
SwitchA(config)#ip route 0.0.0.0 0.0.0.0 10.10.10.3
```

Configure a static route on the MCE.

Configure a static route on the MCE for packets destined to the network segment 192.168.10.0, with the next hop pointing to 10.10.10.2, and binds the route to the instance VPNa.

```
MCE(config)#ip route vrf vpna 192.168.10.0 255.255.255.0 10.10.10.2
```

Configure static routes on the PE.

Configure two static routes on the PE, and bind them to the instance VPNa. One is for packets destined to the network segment 192.168.10.0, with the next hop pointing to 10.10.40.1. The other is for packets destined to the network segment 10.10.10.0, with the next hop also pointing to 10.10.40.1.

```
PE (config)#ip route vrf vpna 192.168.10.0 255.255.255.0 10.10.40.1
PE (config)#ip route vrf vpna 10.10.10.0 255.255.255.0 10.10.40.1
```

Configure RIP route exchange between the MCE and VPNb, and between the MCE and the PE.

Configure RIP on Switch B (access switch for the site VPNb).

Set the IP address of the interface of Switch B that connects to the MCE to 10.10.20.2/24, and that of the interface of Switch B that connects to the site VPNb to 192.168.20.1/24. Here, the operations for adding ports to VLANs and setting interface IP addresses are omitted.

```
SwitchB(config)#router rip
SwitchB(config-router)#version 2
SwitchB(config-router)#no auto-summary
SwitchB(config-router)#network 10.10.20.0 0.0.0.255
SwitchB(config-router)#network 192.168.20.0 0.0.0.255
```

Configure RIP on the MCE.

```
MCE(config)#router rip
MCE(config-router)#address-family ipv4 vrf vpnb
MCE(config-router-af)# version 2
MCE(config-router-af)# no auto-summary
MCE(config-router-af)#network 10.10.20.0 0.0.0.255
MCE(config-router-af)#network 10.10.50.0 0.0.0.255
```

Configure RIP on the PE.

```
PE(config)#router rip
PE(config-router)#address-family ipv4 vrf vpnb
PE(config-router-af)# version 2
PE(config-router-af)# no auto-summary
```

PE(config-router-af)#network 10.10.50.0 0.0.0.255 Configure OSPF route exchange between the MCE and VPNc, and between the MCE and the PE.

Configure the OSPF protocol on Switch C (access switch for the site VPNc).

Set the IP address of the interface of Switch C that connects to the MCE to 10.10.30.2/24, and that of the interface of Switch C that connects to the site VPNc to 192.168.20.1/24. Here, the operations for adding ports to VLANs and setting interface IP addresses are omitted.

```
SwitchC(config)#router ospf 1
SwitchC(config-router)#network 10.10.30.0 0.0.0.255 area 0
SwitchC(config-router)#network 192.168.20.0 0.0.0.255 area 0
```

Configure the OSPF protocol on the MCE.

```
MCE(config)#router ospf 1 vrf vpnc
MCE(config-router)#network 10.10.30.0 0.0.0.255 area 0
MCE(config-router)#network 10.10.60.0 0.0.0.255 area 0
```

Configure the OSPF protocol on the PE.

```
PE(config)#router ospf 1 vrf vpnc
PE(config-router)#network 10.10.60.0 0.0.0.255 area 0
```

2.7.5. Verification

Check routing information about the instance VPNa.

Check routing information on Switch A (access switch for the site VPNa).

```
SwitchA (config)#show ip route
Gateway of last resort is 10.10.10.3 to network 0.0.0.0
S*    0.0.0.0/0 [1/0] via 10.10.10.3
C     10.10.10.0/24 is directly connected, VLAN 10
C     10.10.10.2/32 is local host.
C     192.168.10.0/24 is directly connected, FastEthernet 0/23
C     192.168.10.1/32 is local host.
```

Check routing information about the instance VPNa on the MCE.

```
MCE#show ip route vrf vpna
Routing Table: vpna
C     10.10.10.0/24 is directly connected, VLAN 10
C     10.10.10.3/32 is local host.
C     10.10.40.0/24 is directly connected, VLAN 40
C     10.10.40.1/32 is local host.
S     192.168.10.0/24 [1/0] via 10.10.10.2
```

Check routing information about the instance VPNa on the PE.

PE#show ip route vrf vpna

```
Routing Table: vpna
S     10.10.10.0/24 [1/0] via 10.10.40.1
C     10.10.40.0/24 is directly connected, VLAN 40
C     10.10.40.2/32 is local host.
S     192.168.10.0/24 [1/0] via 10.10.40.1
```

Check routing information about the instance VPNb.

Check routing information on Switch B (access switch for the site VPNb).

SwitchB#show ip route vrf vpnb

```
Routing Table: vpnb
C     10.10.20.0/24 is directly connected, VLAN 20
C     10.10.20.2/32 is local host.
R     10.10.50.0/24 [120/1] via 10.10.20.3, 00:01:20, VLAN 20
C     192.168.20.0/24 is directly connected, FastEthernet 0/23
C     192.168.20.1/32 is local host.
```

Check routing information about the instance VPNb on the MCE.

```
MCE#show ip route vrf vpnb
Routing Table: vpnb
C     10.10.20.0/24 is directly connected, VLAN 20
C     10.10.20.3/32 is local host.
C     10.10.50.0/24 is directly connected, VLAN 50
C     10.10.50.1/32 is local host.
R     192.168.20.0/24 [120/1] via 10.10.20.2, 00:22:01, VLAN 20
```

According to the above information, the MCE has learned the private network routes of VPNb via RIP. These routes and the routes of VPNa and VPNc are separately maintained in three routing tables, thus effectively isolating the VPNs from each other and supporting address duplication inside each VPN.

Check routing information about the instance VPNb on the PE.

```
PE#show ip route vrf vpnb
Routing Table: vpnb
R     10.10.20.0/24 [120/1] via 10.10.50.1, 00:04:48, VLAN 50
C     10.10.50.0/24 is directly connected, VLAN 50
C     10.10.50.2/32 is local host.
```

```
R    192.168.20.0/24 [120/2] via 10.10.50.1, 00:02:15, VLAN 50
```

The above information shows that all the routes of the instance VPNb have been advertised to the PE.

Check routing information about the instance VPNc.

Check routing information on Switch C (access switch for the site VPNc).

```
SwitchC (config-router)#show ip route

C    10.10.30.0/24 is directly connected, VLAN 30
C    10.10.30.2/32 is local host.
O    10.10.60.0/24 [110/2] via 10.10.30.3, 00:02:42, VLAN 30
C    192.168.20.0/24 is directly connected, FastEthernet 0/23
C    192.168.20.1/32 is local host.
```

Check routing information about the instance VPNc on the MCE.

```
MCE#show ip route vrf vpnc

Routing Table: vpnc
C    10.10.30.0/24 is directly connected, VLAN 30
C    10.10.30.3/32 is local host.
C    10.10.60.0/24 is directly connected, VLAN 60
C    10.10.60.1/32 is local host.
O    192.168.20.0/24 [110/2] via 10.10.30.2, 00:01:36, VLAN 30
```

According to the above information, the MCE has learned the private network routes of VPNc via OSPF. These routes and the routes of VPNa and VPNb are separately maintained in three routing tables, thus effectively isolating the VPNs from each other and supporting address duplication inside each VPN.

Check routing information about the instance VPNc on the PE.

```
PE#show ip route vrf vpnc

Routing Table: vpnc
O    10.10.30.0/24 [110/2] via 10.10.60.1, 00:00:00, VLAN 60
C    10.10.60.0/24 is directly connected, VLAN 60
C    10.10.60.2/32 is local host.
O    192.168.20.0/24 [110/3] via 10.10.60.1, 00:00:00, VLAN 60
```

The above information shows that all the routes of the instance VPNc have been advertised to the PE.

2.7.6. Abbreviations

Abbreviation	Full Spelling
CE	Customer Edge Device
PE	Provider Edge Device
MCE	Multi-CE
VPN	Virtual Private Network
VRF	VPN Routing and Forwarding Table

3. CONFIGURING IPV4 EXPRESS FORWARDING

3.1. Understanding IPv4 Express Forwarding

3.1.1. Overview

To meet the needs of higher-end devices, we use a Prefix Tree+Adjacency Express Forwarding (REF) model to enable express forwarding. REF constitutes the mirroring of the whole core routing table instead of buffering some information

in the core routing table. Therefore, the cache does no longer need to be added to the CPU in the case of a cache failure, thus reducing the impact on the CPU and guaranteeing routing stability.

REF constructs the routing table mirroring by using the following two parts:

- Prefix Tree

The Prefix Tree is an IP prefix tree organized according to the maximum matching rule for retrieving adjacent nodes. In general, the data structure for constructing the Prefix Tree is different from the Radix Tree of the core routing table. Instead, a data structure called the M-Tries Tree is used to enable more rapid multi-step search. Using the M-Tries Tree to construct the Prefix Tree will take up more memory than using the Radix Tree. Although updating prefix and node information relatively consumes time, high retrieval performance can be obtained.

- Adjacency

Adjacency refers to an adjacent node, which contains the interface information output by routed packets, such as next-hop list, next processing part, and link-layer output encapsulation. When packets match with such an adjacent node, packets are directly encapsulated, and then the send function of this node is called to enable forwarding. To facilitate retrieval and update, the tables made up of adjacent nodes are generally organized into a hash table. To support load balancing of routes, the next list information on adjacent nodes is organized in load-balancing table form.

REF routing consists of three steps:

- REF de-encapsulates packets.
- Use the Prefix Tree to retrieve the adjacent nodes according to the routes of packets.
- After the adjacent nodes are matched, the final output interface of packets is determined according to adjacent nodes, and then packets are encapsulated according to the output interface type.

3.2. Configuring Load Balancing Policy for Express Forwarding

Express forwarding supports load balancing of packets. At present, three load balancing policies are supported. In the REF model, when the route prefix IP/MASK is associated with multiple next hops, namely, a multi-path route, the route will be associated with one load-balancing table and load is balanced according to route weights. When IP packets match with the load balancing table according to the longest prefix, express forwarding hashes the IP addresses of packets, and then one path is selected for packet forwarding. There arethree routing policies:

- Load is balanced according to destination addresses of IP packets. The destination IP addresses of packets are hashed. The route with a greater weight is more likely to be selected. By default, this policy is selected.
- Load is balanced according to destination and source IP addresses of IP packets. Destination and source IP addresses of packets are hashed and the route with a greater weight is more likely to be selected.
- Load is balanced according to packets polling. Each packet takes turn to select the path and all paths can be selected.

Use the following commands to configure the load balancing policy in global configuration mode:

Command	Function
Qtech(config)# ip ref load-sharing algorithm original	Sets the load balancing algorithm based on source and destination IP addresses.
Qtech(config)# **ip ref load-sharing packet**	Performs the load balancing according to packet polling
Qtech(config)# no ip ref load-sharing algorithm	Load is balanced according to destination addresses of packets.


Note These commands are unique to routers.

3.3. Maintaining and Monitoring the Express Forwarding Table

The express forwarding module passively receives and maintains external routing information instead of actively adding or deleting any route information. As a result, the express forwarding table provides current statistical information on routes.

3.3.1. Statistical information on Express Forwarding Packets

Statistical information on express forwarding packets refers to the statistical information on the packets processed by the express forwarding REF, including the number of forwarded packets and the number of packets dropped due to various reasons.

Command	Function
Qtech# show ip ref packet-statistic [clear]	Displays or clears the current statistical information on packets in REF.

3.3.2. Adjacency Table Information

In the express forwarding table, one type of important data table is the adjacency table. Use the following command to check current adjacency information.

Command	Function
Qtech# show ip ref adjacency [glean | local | *ip* | (interface *interface_type interface_number*) | statistic]	Displays the specified gleaned adjacency, local adjacency, adjacency corresponding to the specified IP address, adjacency associated with the specified interface, and information on all adjacency nodes.

3.3.3. Information on Packet Forwarding Path

Packets are forwarded according to IP addresses of packets. Therefore, if source IP addresses and destination IP addresses of packets are specified, the path for forwarding packets will be determined. Use the following command and specify source IP addresses and destination IP addresses of packets. The actual path for forwarding packets will be displayed. For example, you can learn whether packets are dropped, submitted to the CPU, or forwarded. Furthermore, you can know about the interface through which packets are forwarded.

Command	Function
Qtech# show ip ref exact-route [vrf *vrf_name*] *source-ipaddress dest_ipaddress*	Displays the actual forwarding path of a specified packet.

Note This command is unique to routers.

3.3.4. Routing information in the Express Forwarding Table

Express forwarding receives external route advertisements and maintains an express forwarding table, which is a mirroring of the core routing table. Use the following commands to display routing information in the express forwarding table.

Command	Function
Qtech# show ip ref route [vrf vrf_name] [default | (ip mask) | statistic]	Displays current default routing information in the express forwarding table. If the default route is not specified, all routing information in the express forwarding table is displayed, including the 0 route, default route, and common gateway route.

4. CONFIGURING TCP

4.1. Overview

The TCP module provides a reliable and connection-oriented IP-based transmission layer protocol for the application layer.

The application layer sends data streams represented in 8-bit bytes for Internet transmission to the TCP layer, which separates the data streams into packet segments with proper sizes. The maximum segment size (MSS) is generally limited by the maximum transmission unit (MTU) of the data link layer of the network to which the computer is connected. After that, TCP transmits the result packets to the IP layer, which will then transmit the said packets through the network to the TCP layer of receiving terminal.

To ensure no packet loss, TCP assigns a sequence number to each byte, and the sequence number also ensures that packets transmitted to the receiving terminal are received in sequence. The receiving terminal will then reply with an ACK to confirm the receipt of each byte. If no ACK is received within the reasonable Round Trip Time (RTT), then the corresponding byte (assumed lost) will be retransmitted by the sender.

- With regard to data accuracy and validity, TCP uses a checksum function to verify the data. The checksum must be calculated while the date is sent or received. In the meantime, MD5 authentication can also be utilized to encrypt the data.
- To ensure reliability, TCP applies the mechanisms of timeout retransmission and piggybacking.
- The sliding window protocol is applied to implement flow control. According to the protocol, all unconfirmed packets within the window will be retransmitted.
- The widely recognized TCP congestion control algorithm (also called the AIMD algorithm) is applied to implement congestion control. This algorithm mainly involves: 1) additive increase and multiplicative decrease; 2) slow start; 3) response to timeouts.

4.2. Configuring TCP

4.2.1. Changing the Timeout for Establishing a TCP Connection

Establishing a TCP connection requires a three-way handshake: The local end sends a SYN packet, the remote end responds with a SYN+ACK packet, and then the local end responds with an ACK.

- After the local end sends SYN, if the remote end does not respond with SYN+ACK, the local end will continuously retransmit SYN packets until the specified number of retransmissions is reached or until the timeout timer expires.
- After the local end sends SYN and the remote end responds with SYN+ACK, if the local end no longer responds with ACK, the remote end will keep retransmission until the specified number of retransmissions is reached or until the timeout timer expires (Such as SYN attacks).

Use the following command to configure the timeout value for SYN packets (the maximum time from SYN transmission to successful three-way handshake), namely the timeout for establishing a TCP connection.

Command	Function
Qtech(config)# ip tcp syntime-out *seconds*	Changes the timeout value for establishing a TCP connection. Range: 5 to 300 seconds; default: 20

Use the **no ip tcp syntime-out** command to restore the default value.

Note This command only applies to IPv4 TCP.

4.2.2. Changing the Buffer Size

The TCP receiving buffer is utilized to buffer the data received from the peer end. These data will be subsequently read by application programs. Generally, the window size of TCP packets implies the size of the free space in the receiving buffer. For connections involving a greater bandwidth and mass data, increasing the size of the receiving buffer will remarkably improve TCP transmission performance. The sending buffer is utilized to buffer the data of application programs. Each byte in the buffer has a sequence number, and bytes with sequence numbers acknowledged will be removed from the sending buffer. Increasing the sending buffer will improve the interaction between TCP and application programs, thus enhancing the performance. However, increasing the receiving buffer and sending buffer will result in more memory consumption of TCP.

Command	Function
Qtech(config)# ip tcp window-size *size*	Changes the size of receiving buffer and sending buffer for TCP connections. Range: 0 to 65535 bytes; default: 4096.

Use the **no ip tcp window-size** command to restore the default value.


Note This command only applies to IPv4 TCP.


Note This command does not apply to existing TCP connections. It only applies to subsequent TCP connections.


Note This command will apply to both the receiving buffer and sending buffer.

4.2.3. Prohibiting the Reset Packet When the Port is Unreachable

When the TCP module distributes TCP packets, if the TCP connection to which such packets belong cannot be found, a reset packet will be replied to the peer end to terminate the TCP connection. The attacker may initiate attacks by sending excessive port-unreachable TCP packets.

Run the following commands to prohibit or restore the reset packet sent when the port-unreachable TCP packet is received.

Command	Function
Qtech(config)# ip tcp not-send-rst	Prohibits sending a reset packet when the port-unreachable TCP packet is received.

Use the **no ip tcp not-send-rst** command to restore the default settings.


Note This command only applies to IPv4 TCP.

4.2.4. Limiting the MSS of TCP Connections

The MSS refers to the maximum size of the payload of a TCP packet, excluding TCP options.

During the three-way handshake for establishing a TCP connection, one important job is to carry out MSS negotiation. Both sides will insert an MSS option into the SYN packet to indicate the maximum size of the segment that can be received by the local end, namely the maximum size of the segment that can be sent by the remote end. Both sides will take the smaller of the MSS value sent locally and that received from the remote end as the maximum segment size of this connection.

The methods for calculating the value of the MSS option while sending SYN packets are shown below:

Non-directly connected network: MSS = Default value 536

Directly connected network: mss = egress interface MTU corresponding to the peer IP address – 20-byte IP header – 20-byte TCP header
Generally speaking, if the MTU is affected by certain applications configured on the egress interface, such applications will configure the MTU accordingly, such as the MTUs of the tunnel port and VPN port.


Note In release 10.4(3), in the SYN+ACK packet replied by the remote end of a directly connected network, the MSS option is not calculated through MTU. Instead, the default value 536 is used.


Note The MSS calculated cannot exceed the size of the receiving buffer or the IP TCP MSS configured by the user. Otherwise, the smaller of them will be used.

Note

If certain options are supported by this connection, then the size obtained after 4-byte alignment of the option must be subtracted from the MSS. For example, if the size of the MD5 option is 18 bytes, 20 bytes will be obtained after alignment.

The RMSS value obtained here is the value of the MSS option in the SYN packet sent. For example, BGP adjacency is generally established in the directly connected network, and the MSS of such a connection is 1500 – 20 – 20 – 20 = 1440.

The function of IP TCP MSS is to limit the MSS of the pending TCP connection. The negotiated MSS cannot exceed the value configured.

Command	Function
Qtech(config)# **ip tcp mss** *max-segment-size*	Limits the maximum segment size of TCP connections. Range: 68 to 10000 bytes.

Use the **no ip tcp mss** command to disable such limit.

Note

This command only applies to IPv4 TCP.

4.2.5. Enabling PMTU Discovery

The TCP Path MTU (PMTU) is implemented as per RFC1191. This feature can improve the network bandwidth utilization ratio. When the user uses TCP to transmit mass data, this feature can substantially enhance the transmission performance.

Command	Function
Qtech(config)# **ip tcp path-mtu-discovery** [**age-timer** *minutes* | **age-timer infinite**]	Enables PMTU discovery. **age-timer** *minutes*: The time interval for further discovery after discovering PMTU in the range from 10 to 30 minutes. The default value is 10. **age-timer infinite**: No further discovery after discovering PMTU.

According to RFC1191, after discovering PMTU, TCP can use a greater MSS to discover a new PMTU, and the time interval thereof is specified by the parameter **age-timer**. When the PMTU discovered by the device is smaller than the MSS negotiated, the device will try to discover a greater PMTU as per the aforementioned time interval. Such a discovery process will not end until PMTU reaches the value of MSS or until the user stops this timer. To turn off the timer, use the parameter **age-timer infinite**.

Use the **no ip tcp path-mtu-discovery** command to disable PMTU discovery.

Note

This command applies to both IPv4 TCP and IPv6 TCP.

Note

This command does not apply to existing TCP connections. It only applies to subsequent TCP connections.

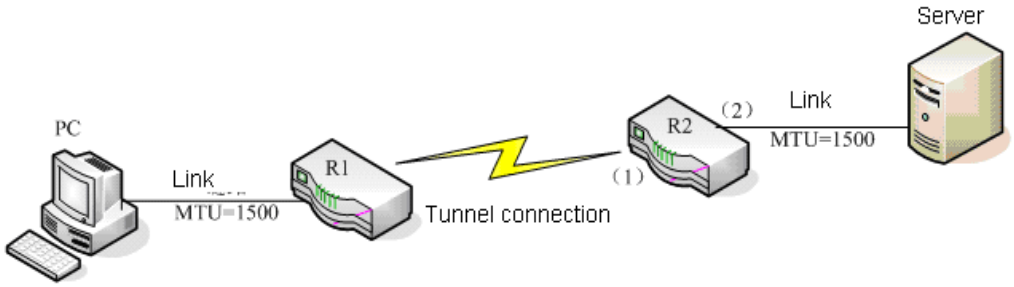
4.2.6. Configuring the MSS Option of SYN Packets Sent and Received on the Interface

The TCP Path MTU (PMTU) is implemented as per RFC1191. This feature can improve the network bandwidth utilization ratio. When the user uses TCP to transmit mass data, this feature can substantially enhance the transmission performance.

When the client initiates a TCP connection, it negotiates the maximum payload of TCP packets through the MSS Option field of the TCP SYN packet. The MSS value of client's SYN packet implies the maximum payload of TCP packets sent by the server, and vice versa.

As shown in the following figure, a PC may fail to access the server through HTTP, because the MSS of 1460 will be negotiated between the PC and the server, but such MSS cannot pass R1 and R2 (R1 and R2 are connected through a tunnel, with an MTU smaller than 1500).

Figure 1-1



In such a case, we can configure the following command on port (1) and port (2) of R2 to change the MSS Option value of the SYN packet, so as to change the MSS value negotiated for the TCP connection going through port (1) and port (2).

Command	Function
Qtech(config-if)# **ip tcp adjust-mss** *max-segment-size*	Configures the MSS Option value of SYN packets sent and received on the interface. Range: 500 to 1460 bytes.

Use the **no ip tcp adjust-mss** command to remove the configuration. In such a case, the MSS Option value of packets will not be changed when the interface sends and receives SYN packets.

Configuring this command on the interface will change the MSS option of SYN packets received or sent on the interface to the MSS value configured on the interface. It is suggested that you configure the same value on the ingress interface and egress interface, or else the MSS option of SYN packets going through the device will be changed to the smaller of the two values configured.

Note This command only applies to IPv4 TCP and is only supported by routers.

Note In release 10.4(3), the MSS value of SYN+ACK packets will not be changed.

4.3. Monitoring and Maintenance

Command	Function
Qtech# show tcp connect	Displays basic information about the current TCP connection.
Qtech# show tcp pmtu	Displays information about TCP PMTU.
Qtech# show tcp port	Displays information about the current TCP port.