Technical Guide

GRE and Multipoint VPNs

Feature Overview and Configuration Guide

Introduction

This guide describes Generic Routing Encapsulation (GRE) and its configuration. GRE is a mechanism for encapsulating any network layer protocol over any other network layer protocol. GRE can be used in point-to-point mode to provide a VPN between two sites.

Additionally, GRE can be used for Multipoint Virtual Private Networks (VPNs) using GRE in point-to-multipoint mode. Multipoint VPNs simplify configuration and allow a single tunnel interface to have multiple endpoints. For example, this means that only a single virtual tunnel interface configuration is required in a main office to connect to all other remote offices.

Products and software version that apply to this guide

This guide applies to AlliedWare Plus™ products that support GRE, running version **5.4.5** or later and Multipoint VPNs running version **5.4.9** or later.

However, implementation varies between products. To see whether a product supports a feature or command, see the following documents:

- The product's Datasheet
- The product's Command Reference

These documents are available from the above links on our website at alliedtelesis.com.

Feature support may change in later software versions. For the latest information, see the above documents.

Related documents

The following documents gives more information about using security with IPsec with Multipoint VPN on AlliedWare Plus products:

- the IPsec Feature Overview and Configuration Guide

Contents

What is GRE?

GRE is a mechanism for encapsulating any network layer protocol over any other network layer protocol. The general specification was originally described in RFC 1701, and the encapsulation of IP packets over IP is defined in RFC 1702 as a specific implementation of GRE. The GRE specification has been formalized in RFC 2784 and is commonly used for encapsulating IPv4 and IPv6 packets inside IPv4 packets. RFC 2890 extends RFC 2784 with the edition of key and sequence number. RFC 7676 specifies GRE procedures for IPv6 used as either the payload or delivery protocol (support is from version 5.4.8-2.1 onwards).

The IPv4/IPv6 protocol 47 is used when GRE packets are encapsulated in IPv4 or IPv6. GRE is widely used in VPNs as the mechanism for transporting IP packets between private IP networks across public networks with globally routed IP addresses. The advantage of GRE over other tunneling protocols is that it can encapsulate broadcast, multicast traffic (multicast streaming or routing protocols) or other non-IP protocols. GRE packets can be protected using Internet Protocol Security (IPsec) ensuring confidentiality and integrity of the tunneled traffic.

GRE is stateless and has no knowledge of the configuration or even existence of the remote tunnel endpoint. Once GRE is configured, packets are encapsulated and forwarded whether the decapsulating device is present or not.

GRE allows hosts in one private IP network to communicate with hosts in another private IP network by providing a VPN between two routers across the Internet.

The GRE connection endpoints are terminated via a Virtual Tunnel Interface (VTI) configured in each device.

Figure 1: A GRE encapsulated packet form

Delivery Header (IPv4/IPv6)	GRE Header	Payload Header (IPv4/IPv6)	Payload

Figure 2: A GRE packet header structure

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C| |K|S| Reserved0       | Ver |         Protocol Type         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Checksum (optional)      |       Reserved1 (Optional)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Key (optional)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Sequence Number (Optional)                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Virtual Tunnel Interface (VTI)

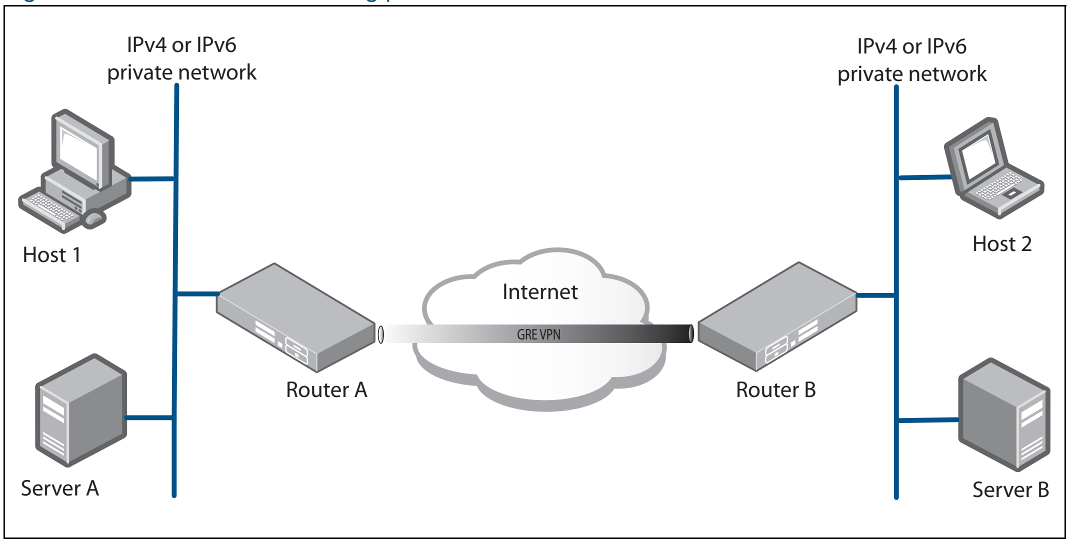
A Virtual Tunnel Interface has similar characteristics to any other interface on the device. It is virtual because it does not directly map to any of the physical interfaces on the device, but instead is actually the endpoint of a tunnel from another device. VTIs are commonly layer 3 interfaces, can have IP configuration applied directly to them and are compatible with layer 3 routing protocols. The actual tunneling mechanism depends on the protocol used (GRE, RFC2473, L2TP and so on), but commonly uses IP as its transport.

Delivery header This is the outer or encapsulating header. The IPv4/IPv6 delivery header uses protocol 47 to indicate the next header is a GRE header.

GRE Header The 4-16 byte header is placed between the delivery and payload headers. The GRE header stores the GRE version and payload protocol type. Optionally the GRE header can store a packet checksum, a tunnel key and packet sequence number.

Payload header This is the inner or encapsulated header. GRE is commonly and only used to transport IPv4 and IPv6 packets.

Figure 3: GRE IP network tunneling protocols



IP packets from the private IP network destined for a device in the remote private IP network are encapsulated by Router A and forwarded to Router B. Intermediate routers located in the Internet route the packets using addresses in the delivery protocol header. Router B extracts the original payload packet and routes it to the appropriate destination within the private network.

Supported features

- GRE over IPv4 as specified in RFC 2784.
- GRE over IPv6 as specified in RFC 7676.
- Virtual Tunnel Interfaces for terminating GRE encapsulated traffic.
- IPv4/IPv6 as the delivery protocol, used to transport the private data across the public network.
- IPv4 and or IPv6 as the payload.
- Up to 256 GRE connections can be configured, with a single GRE tunnel per VTI when point-to-point GRE is used.
- Configurable tunnel source using IPv4/IPv6 address.
- Configurable tunnel source using interface.
- Configurable tunnel destination IPv4/IPv6 address.

- Configurable tunnel destination using hostname.
- Configurable tunnel local name for IPsec phase 1 local ID, required for IPsec NAT-T.
- Configurable tunnel remote name for IPsec phase 1 remote ID, required for IPsec NAT-T.
- Configurable checksum insertion and checking (disabled by default).
- Configurable TTL value for insertion into the outer header.
- Configurable hop limit value for insertion into the outer header (defaults to 64).
- Configurable MSS Clamping.
- Configurable MTU.
- GRE IPv6 MTU default is 1456, 1452 if Checksum is configured.
- Protection of GRE IPv6 based VTI traffic using IPSec in transport mode.
- IPsec protected GRE IPv6 MTU default is 1406, 1402 if checksum is configured.
- Configurable DSCP value for insertion into the outer header (copied from the inner header by default).
- Inherit the Flow Label from the inner header (zero if inner header IPv4).
- Display of tunnel parameters via show interface tunnel (GRE) command output.
- Tunnels are compatible with dynamic IPv4 and IPv6 routing protocols (RIPv1, RIPv2, RIPNG, OSPF, OSPFv3, BGP, BGP4+).
- DF value in the outer header is always set to 1 (on).

Unsupported features

- Non-IPv4/IPv6 protocols as the delivery protocol.
- Non-IPv4/IPv6 protocols as the payload.
- Insertion or processing of Tunnel Key in the GRE header (received packets including a key are dropped).
- Insertion or processing of Sequence Numbers in the GRE header (sequence numbers in received packets are ignored).
- Insertion or processing of Source Route Entries in the GRE header (received packets including a route entry are dropped).
- Path-MTU-discovery in the underlying tunnel interface.
- Keep-alives at the GRE protocol level.
- Configurable DF value for insertion into the outer header.
- Hardware acceleration of GRE encapsulation/decapsulation processes.

GRE and Multipoint VPN

Multipoint VPN is supported using GRE in **point-to-multipoint mode**. Multipoint VPN simplifies configuration and allows a single tunnel to have multiple endpoints. For example, this means that only a single tunnel configuration is required in a main office to connect to all other remote offices.

Acronyms and terms

The following acronyms and terms are used with GRE:

Table 1: GRE Acronyms and terms

ACRONYMS AND TERMS	DESCRIPTION
Multipoint GRE	A GRE tunnel which has multiple remote endpoints.
Point-to-point tunnels	Point-to-point tunnels have a single tunnel destination, no broadcast or link layer resolution (ARP) and packets are routed via the virtual tunnel interface.
Point-to-multipoint tunnels	Point-to-multipoint tunnels have multiple tunnel endpoints, broadcast, link layer resolution (ARP) and packets are routed via the destination next hop.
Hub-and-spoke topology	In a hub and spoke network configuration, the main office has configuration for a tunnel to each remote office, and each remote office has a single tunnel connecting to the main office. Traffic between remote offices is routed via the main office.
Full-mesh topology	In a fully-meshed network configuration, each office node has a tunnel to all other office nodes.
Partial-mesh topology	In a partially meshed topology nodes have tunnels to a sub-set of all the nodes in the network.
SA	Security Association (SA) specifies the SPI, protocol (ESP, AH, IPCOMP), algorithms and keys for protecting a single flow of traffic between two IPsec peers.
ISAKMP	The Internet Security Association and Key Management Protocol (ISAKMP) provides a framework for negotiating SAs and their attributes, including secret keys.
IPsec	Internet Protocol Security (IPsec) defines the protection of IP packets using encryption and authentication. IPsec is the defacto standard for site-to-site tunnel protection due to its widespread use and performance due to hardware acceleration.
Broadcast and multicast	Broadcast and multicast are required for Address Resolution Protocol (ARP) and dynamic routing protocols, and can be emulated by replicating the packets to each of the known tunnel endpoints.
Next Hop Resolution	With a multipoint tunnel, it is no longer sufficient to say that an address is reachable over a particular tunnel; it's necessary to specify which tunnel endpoint to use. In an Ethernet environment, next hops are resolved to a MAC address. In a multipoint tunnel environment, next hops are resolved to the remote IP of the tunnel endpoint.
Endpoint authentication	This is the process of confirming that the tunnel endpoint is a known and approved device. This can be achieved via ISAKMP using pre-shared keys (PSKs) configured on the hub and each spoke. However in a multipoint environment with the possibility of a large number of endpoints, the scalability of the hub can be improved if ISAKMP uses RADIUS authentication.

Supported features

- **Multipoint GRE** Virtual Tunnel Interfaces use the command **tunnel mode gre multipoint**.
- A single Virtual Tunnel Interface (VTI) per device.
- IPv4 only as the delivery protocol.
- IPv4 only as the payload.
- Configurable tunnel source using IPv4 address.
- Configurable tunnel source using interface.
- Configurable checksum insertion and checking (disabled by default).
- Configurable TTL value for insertion into the outer header (defaults to 64).
- Configurable DSCP value for insertion into the outer header (defaults to copying from the inner header).
- Display of tunnel parameters displays in **show interface** output.
- MTU command can be used to set the MTU of a tunnel interface.
- DF bit is always set in the outer header (regardless of value in inner header).
- Tunnel is active/up with zero or more known endpoints.
- Configurable static tunnel endpoints.
- Endpoints can be specified using an IP address.
- Endpoints can be specified using a hostname which is resolved to an IP address before used.
- Failed hostname resolution is retried with intervals of 1, 2, 4, 8, 16, 32, 60, 60, 60, ... seconds.
- Endpoints can be dynamically learned from ISAKMP.
- Support broadcast and multicast by replicating packets to all known endpoints.
- Support MSS clamping on **Multipoint GRE** Virtual Tunnel Interfaces.

Unsupported features

- Non-IPv4 protocols as the delivery protocol.
- Non-IPv4 protocols as the payload.
- Insertion or processing of Tunnel Key in the GRE header (received packets including a key are dropped).
- Insertion or processing of Sequence Numbers in the GRE header (sequence numbers in received packets are ignored).
- Insertion or processing of Source Route Entries in the GRE header (received packets including a route entry are dropped).

- Path-MTU-discovery in the underlying tunnel interface.
- Keep-alives at the GRE protocol level.
- Configurable DF value for insertion into the outer header.
- Dynamic IP address allocation over the tunnel (for example, DHCP, ISAKMP).
- Gratuitous ARP support on tunnel interfaces.
- Dynamic endpoint resolution.

Device limits

The maximum tunnel endpoints apply to the following devices:

Table 2: Tunnel endpoint limits on devices

DEVICE	MAX ENDPOINTS	DEVICE	MAX ENDPOINTS
AR2010V	100	AR3050S	100
AR2050V	100	AR4050S	1000

What is Multipoint VPN?

Multipoint VPN is a term that describes the use of point-to-multipoint tunnels to enhance traditional point-to-point VPN networks.

Note: Point-to-multipoint tunnels have different characteristics to traditional point-to-point tunnels.

Table 3: Different characteristics between point-to-point and point-to-multipoint

POINT-TO-POINT	POINT-TO-MULTIPOINT
■ Single tunnel destination	■ Multiple tunnel endpoints
■ No broadcast	■ Broadcast
■ No link layer resolution (ARP)	■ Link layer resolution (ARP)
■ Route via interface	■ Route via next hop

Point-to-point

Traditional VPN network tunnels are point-to-point with a single destination. This works well if you only have two sites to connect. But if you have many sites to connect, for example a head office and many branch offices then it requires complex configuration. Each branch site needs a separate tunnel configured at the head office. The more spoke sites you add, then the more tunnels you need to configure.

Point-to-multipoint

Point-to-Multipoint GRE is used as the transport protocol for the encapsulation of packets in multipoint VPN. GRE supports multiple remote endpoints via a single VTI. It can transport both IPv4 and IPv6 packets and emulates broadcast and multicast by duplicating those packets and sending them to all known endpoints.

These tunnel interfaces support multiple remote endpoints. They are more like Ethernet in that the link layer address (LLAddress) of the next hop is required before the packet can be forwarded. Like

Ethernet, the next hop LLAddress is resolved using broadcast ARP requests. Unlike Ethernet the next hop LLAddress is resolved to an IP address, (the tunnel endpoint address), not a MAC address.

The neighbor table for a point-to-multipoint VTI contains mappings of IP addresses to tunnel endpoint IP addresses. The command **show arp** displays the neighbor table.

Broadcast packets are sent to every endpoint the local tunnel knows about. The transmitting device replicates the broadcast packet for each endpoint in its local endpoint list, (encapsulating for each remote endpoint), and finally sending the encapsulated packets to the endpoints.

Static routes should be installed with next hop addresses rather than just the tunnel interface which is fine for point-to-point tunnels.

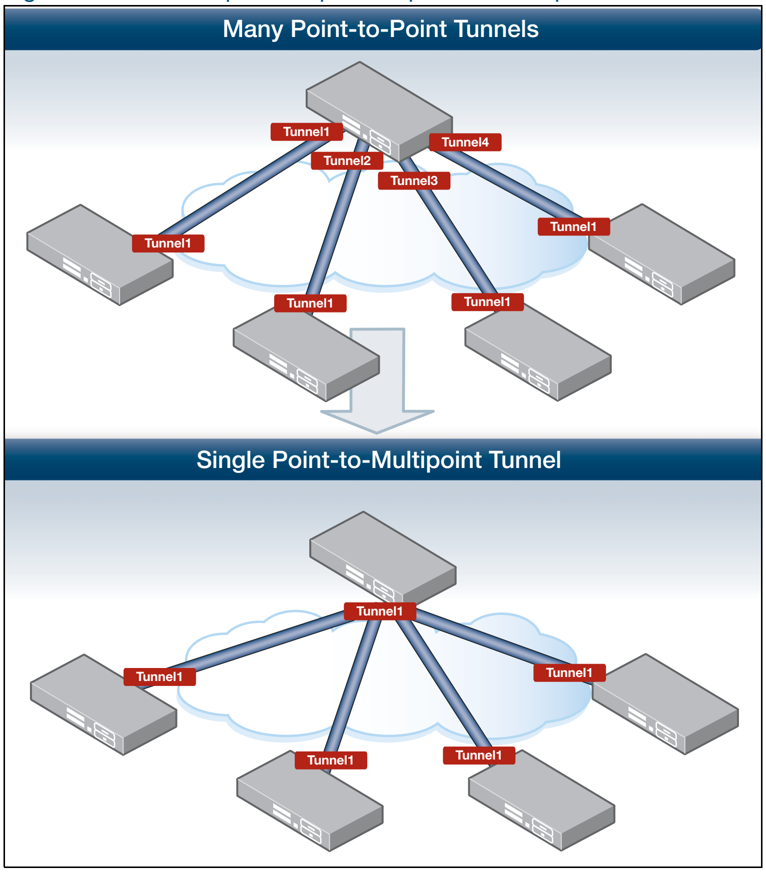
Dynamic routing protocols such as OSPF need to operate in point-to-multipoint mode.

Hub-and-spoke VPNs

Traditional Hub-and-spoke VPNs require many point-to-point tunnels on the hub device. Configurations can become large very quickly as spokes are added to the network. Apart from the configuration overhead, the internal management of many network interfaces takes away processing capability from the core job of forwarding packets.

Hub-and-spoke can be enhanced using point-to-multipoint tunnels to reduce the number of VTIs on the hub from many to one single tunnel.

Figure 4: VPNs from point-to-point to point-to-multipoint



- Multipoint VPNs can enhance existing traditional hub-and-spoke topologies by reducing the amount of configuration required on the hub device.
- Each device in a Multipoint VPN has only one tunnel configured.
- The tunnel can have Multiple tunnel endpoints.

Multipoint VPN networks are an ideal solution because only a single tunnel configuration is required for a head office to connect to all of its branch offices.

Mesh VPNs

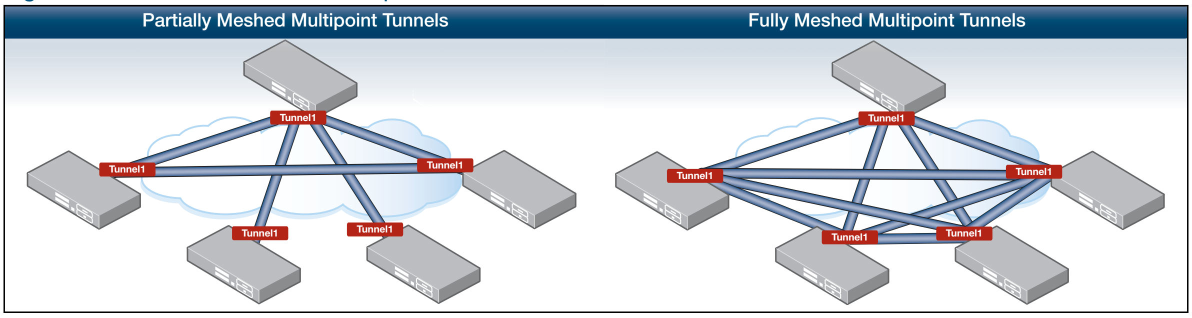
Mesh VPNs can be implemented with a single tunnel interface on each node. Mesh VPNs fall into two sub-categories.

Table 4: Partial-mesh vs full-mesh

PARTIAL-MESH	FULL-MESH
Nodes have VPN connections to a subset of all other nodes.	All Nodes have VPN connections to all other nodes.

Note: Some protocols may require extra configuration when using partially meshed VPNs if they default to assuming all nodes in a single IP subnet are reachable from every other node.

Figure 5: Partial and full-mesh Multipoint VPNs


Partially Meshed Multipoint Tunnels
Fully Meshed Multipoint Tunnels
Tunnel1

Multipoint VPN can enable the use of a full-mesh topology where any node can be easily configured to communicate with any other node over the same single tunnel endpoint.

Dynamic routing protocols such as OSPF can also be used to provide routing to each network attached to a Multipoint VPN.

How does Multipoint VPN work?

Multipoint VPN uses GRE as the transport protocol and ARP for next hop resolution. Routing can be static or dynamic. Security authentication is configured using IPsec.

Tunnel transport protocol

Multipoint VPN uses **Point-to-Multipoint GRE** as the transport protocol for the encapsulation of packets. GRE supports multiple remote endpoints on a single Virtual Tunnel Interface (VTI). It can transport both IPv4 and IPv6 packets and emulates broadcast and multicast by duplicating those packets and sending them to all known endpoints.

Next hop resolution

Multipoint VPN VTIs are different to other interface types in that the next hop for any forwarded IP address is another IP address rather than a MAC address you might see on an Ethernet interface. To enable IPv4 or IPv6 traffic over a Multipoint VPN tunnel, the VTI must be assigned an IP address. Neighbors are then resolved using ARP and can be seen in the neighbor table with an IP address as the Link Layer Address.

Routing

Multipoint VPN tunnels support both static routing and dynamic routing protocols such as, Border Gateway Protocol (BGP) and Open Shortest Path First (OSPF). OSPF interfaces must be configured in Point-to-Multipoint mode so that the OSPF hello messages are correctly distributed to all endpoints of the Multipoint VTI.

Tunnel protection

IPsec is the de facto standard for site-to-site tunnel protection due to its widespread use and performance due to hardware acceleration. IPsec tunnel protection is configured for Multipoint VPN like for any of the other Point-to-Point VTIs. Because the multipoint VTI has multiple remote peers, it is often simpler to identify pre-shared keys using the local tunnel identifier rather than remove endpoint addresses.

Each device in a multipoint VPN still establishes tunnel protection with each other device it needs to communicate with. So, although there is only one VTI in a Multipoint VPN, there will be multiple IPsec and ISAKMP SAs and peers.

Scaling Multipoint VPNs

This section explains the scaling required when setting up Multipoint VPN topologies. OSPF (Open Shortest Path First) is used for smaller topologies, and BGP (Border Gateway Protocol) is used for larger topologies.

Use of OSPF

For smaller Multipoint VPN topologies, OSPF is suitable to use as the interior routing protocol.

Dijkstra algorithm

OSPF uses Dijkstra's algorithm to compute the Shortest Path Tree (SPT). OSPF routers will each individually run SPF against their recently populated link state database, learned via neighbor exchanges. This SPF calculation and resulting tree is used to determine the shortest path within the topology, and is used to populate routing tables. And, because all routers run the same calculation on the same data (same link state database), every router has the same picture of the network.

However, running the SPF calculation is very CPU intensive. As the number of nodes and adjacencies are increased, a single topology event can impact the stability of the entire network until the routing tables of devices within the topology are converged. Additionally, it is a good idea to keep the OSPF router Link-State Advertisements (LSAs) under the IP Maximum Transmission Unit (MTU). When the MTU is exceeded, IP fragmentation occurs, which in turn adds additional and unnecessary router processing load.

OSPF Scaling limitations

To mitigate these effects, in the case of OSPF, generally it is good network design practice to ensure an OSPF area does not exceed around 50 or so routers. If they do, timers may need adjustment to ensure sufficient processing time is available for convergence.

And OSPF routes can't be summarized within a single area. OSPF allows inter-area summarization only on the Area Boarder Router (ABR), and Autonomous System Boundary Router (ASBR). So, when scaling networks using OSPF, routers are typically configured as area boarder routers - consisting backbone area, with adjacent areas attached. And, contiguous subnets are configured to ensure efficient summarization via area boarder routers, or autonomous system boundary routers can occur. And, since link-state updates that notify routers to rerun SPF calculations are only performed intra-area, this allows OSPF to scale efficiently.

However in a multipoint VPN network, the tunnel interface of the hub, and each spoke are all attached to a common area. This is typically the backbone area, as it is not possible to attach the single multipoint tunnel interface of the hub to multiple areas. This is fine for Multipoint GRE VPN topologies consisting of a smaller number of spoke devices.

For larger Multipoint VPN network topologies, alternative routing protocols, such as BGP should be considered as the summarization techniques mentioned above are not available.

Use of BGP

BGP can be used as an alternative interior routing protocol within the Multipoint VPN topology when there is a need to scale the numbers of attached spoke nodes. BGP uses Transmission Control Protocol (TCP) on port 179 to send route updates between neighbors. BGP has the efficiency, in that it does not advertise routing prefix updates unless it has to. Devices communicating via BGP are sometimes referred to as BGP speakers.

A BGP session is a TCP connection formed between two BGP speakers over which BGP messages are exchanged.

iBGP and eBGP

There are two common modes of BGP. Internal BGP (iBGP), and External BGP (eBGP). Both iBGP, and eBGP can be used for scaling Multipoint VPN networks.

- eBGP is used to advertise prefix information from one Autonomous System to another
- iBGP is used to advertise prefix information within the same Autonomous System.

iBGP

An iBGP session is formed when the two BGP speakers belong to the same Autonomous System (AS). In the case of iBGP, there is typically a TCP connection established to all other BGP speakers operating within the Autonomous System. By default, iBGP assumes there is full mesh peering between all devices within the AS. Because of this assumption, by default, iBGP does not allow prefixes received from one iBGP peer being re-advertised to other iBGP peers.

However, as each new node is added to the topology, there is a need for it to establish a peering to all other nodes within the AS - so the number of peering sessions can grow exponentially. This presents a scaling problem and can become unwieldy and rather impractical if each router needs to establish TCP connections to all other routers within the AS. To counter this, BGP speakers operating within an AS can be grouped into BGP clusters and/or confederations. By splitting the AS into smaller groups of routers, the requirement for full mesh peering of BGP sessions inside an AS is avoided.

For example, a BGP cluster consists of a combination of route reflector, and route reflector client. This is ideal in a hub and spoke Multipoint VPN topology, where the hub can be configured as the route reflector, and each spoke configured as route reflector clients. It means there is no need to configure each spoke to peer, to any other spoke within the AS. Only a single peering is required from each spoke to the hub. Routing information learned from one spoke is re-advertised by the hub to other spokes. In a non-fully meshed Multipoint VPN environment, the hub may also be optionally configured to modify the next-hop address to be itself for reflected routing information. So, each spoke directs its traffic via the nexthop IP of the hub to reach other spokes.

eBGP

The alternative approach is to configure use of eBGP. This is useful if you want to avoid the configuration complexities of iBGP, such as route reflection and next-hop address modification.

BGP Scaling

When eBGP is used, each BGP speaker is configured with a different AS Number (ASN). The ASN is a **16 bit** field, with 65,536 unique values. From this pool. the IANA reserves 1,026 of them: 64512 – 65534 for **private**, reusable ASNs. This allows scaling for up to 1024 spokes, each configured with unique private ASNs. This also ensures there is no risk of routing loops, since by default an eBGP speaker rejects incoming updates containing its own AS number.

Configuration Examples

IPv4 GRE tunnel

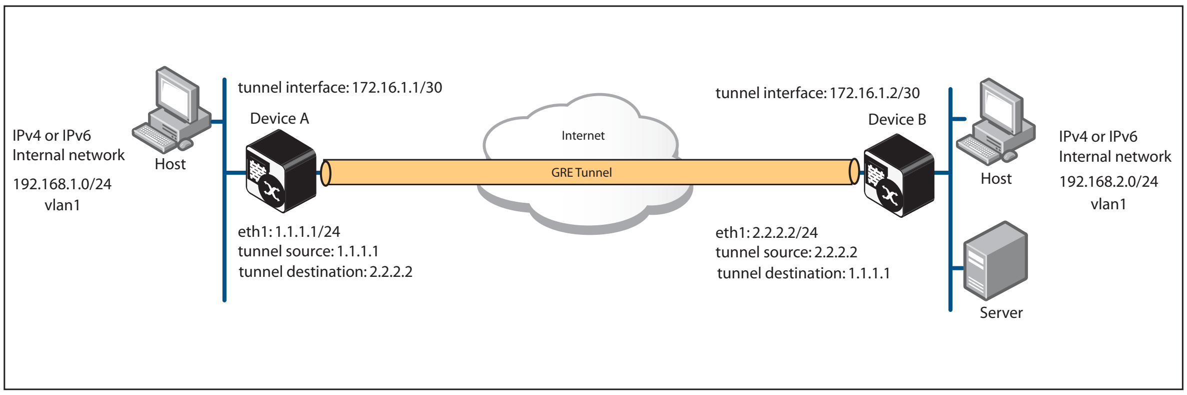
Example 1 This example shows step-by-step instructions to configure an IPv4 GRE tunnel between Device A and Device B. It assumes that the IP addresses have been configured correctly and is operational on both devices. The following table lists the parameter values in this example.

Note: Public IP addresses are used in this example.

Table 5: IPv4 address allocation

DESCRIPTION	DEVICE A	DEVICE B
IP address of Ethernet interface eth1	1.1.1.1/24	2.2.2.2/24
tunnel source IP address	1.1.1.1	2.2.2.2
tunnel destination IP address	2.2.2.2	1.1.1.1
IP address of tunnel interface	172.16.1.1/30	172.16.1.2/30

Figure 6: IPv4 GRE tunnel



Use the following steps to configure an IPv4 GRE tunnel for this example:

Step 1: **Configure Device A**

```
awplus#configure terminal
awplus(config)#interface vlan1
awplus(config-if)#ip address 192.168.1.254/24
awplus(config-if)#interface eth1
awplus(config-if)#ip address 1.1.1.1/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.1/30
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source 1.1.1.1
awplus(config-if)#tunnel destination 2.2.2.2
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.2.0/24 172.16.1.2
awplus(config)#ip route 0.0.0.0/0 1.1.1.254
```

Step 2: **Configure Device B**

```
awplus#configure terminal
awplus(config)#interface vlan1
awplus(config-if)#ip address 192.168.2.254/24
awplus(config-if)#interface eth1
awplus(config-if)#ip address 2.2.2.2/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.2/30
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source 2.2.2.2
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.1.0/24 172.16.1.1
awplus(config)#ip route 0.0.0.0/0 2.2.2.254
awplus(config)#exit
```

Step 3: **Verify connectivity**

Verify the tunnel established using the ping command to the host attached to vlan1 of the remote peer.

```
awplus#ping 192.168.2.254
```

You should receive an ICMP Echo reply message.

IPv6 GRE tunnel

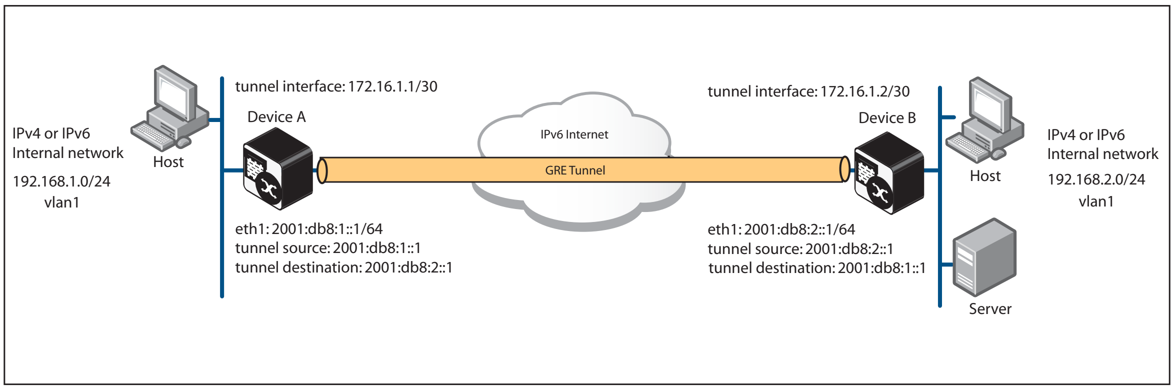
Example 2 This example shows step-by-step instructions to configure an IPv6 GRE tunnel between Device A and Device B transporting private IPv4 traffic. It assumes that the IP addresses have been configured correctly and is operational on both devices. The following table lists the parameter values in this example.

Note: Public IP addresses are used in this example.

Table 6: IPv6 address allocation

DESCRIPTION	DEVICE A	DEVICE B
IP address of Ethernet interface vlan1	192.168.1.254/24	192.168.2.254/24
tunnel source IP address	2001:db8:1::1	2001:db8:2::1
tunnel destination IP address	2001:db8:2::1	2001:db8:1::1
IP address of tunnel interface	172.16.1.1/30	172.16.1.2/30

Figure 7: IPv6 GRE tunnel



Use the following steps to configure an IPv6 GRE tunnel for this example:

Step 1: Configure Device A

```
awplus#configure terminal
awplus(config)#interface vlan1
awplus(config-if)#ip address 192.168.1.254/24
awplus(config-if)#interface eth1
awplus(config-if)#ipv6 address 2001:db8:1::1/64
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.1/30
awplus(config-if)#tunnel mode gre ipv6
awplus(config-if)#tunnel source 2001:db8:1::1
awplus(config-if)#tunnel destination 2001:db8:2::1
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.2.0/24 172.16.1.2
awplus(config)#ipv6 route ::/0 2001:db8:1::2
```

Step 2: Configure Device B

```
awplus#configure terminal
awplus(config)#interface vlan1
awplus(config-if)#ip address 192.168.2.254/24
awplus(config-if)#interface eth1
awplus(config-if)#ipv6 address 2001:db8:2::1/64
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.2/30
awplus(config-if)#tunnel mode gre ipv6
awplus(config-if)#tunnel source 2001:db8:2::1
awplus(config-if)#tunnel destination 2001:db8:1::1
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.1.0/24 172.16.1.1
awplus(config)#ipv6 route ::/0 2001:db8:2::2
awplus(config)#exit
```

Step 3: **Verify connectivity**

Verify the tunnel established using the ping command to the host attached to vlan1 of the remote peer.

```
awplus#ping 192.168.2.254
```

You should receive an ICMP Echo reply message.

Multipoint hub and spoke (single key)

Example 3 This example configures a single VTI on a Hub with a single ISAKMP key.

Step 1: **Configure the hub (point-to-multipoint)**

```
awplus#configure terminal
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 1.1.1.1/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.1.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.1/28
awplus(config-if)#tunnel mode gre multipoint
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel endpoint dynamic
awplus(config-if)#tunnel local name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.2.0/24 172.16.1.2
awplus(config)#ip route 192.168.3.0/24 172.16.1.3
awplus(config)#ip route 0.0.0.0/0 1.1.1.254
```

Step 2: **Configure spoke1 (point-to-point)**

```
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 2.2.2.2/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.2.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.2/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke1
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.0.0/16 172.16.1.1
awplus(config)#ip route 0.0.0.0/0 2.2.2.254
```

Step 3: **Configure spoke2 (point-to-point)**

```
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 3.3.3.3/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.3.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.3/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke2
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.0.0/16 172.16.1.1
awplus(config)#ip route 0.0.0.0/0 3.3.3.254
```

Show Commands to verify the configuration

Use the following show commands to check details about VTIs, neighbor tables and IPsec peers configured with Multipoint VPN.

Virtual Tunnel Interface

show interface tunnel

```
awplus#show interface tunnel1
Interface tunnel1
  Link is UP, administrative state is UP
  Hardware is Tunnel
  IPv4 address 172.16.1.1/28 broadcast 172.16.1.15
  index 15 metric 1 mtu 1434
  <UP,BROADCAST,RUNNING,MULTICAST>
  VRF Binding: Not bound
  SNMP link-status traps: Disabled
  Bandwidth 1g
  Tunnel source 1.1.1.1
  Tunnel name local node1
  Tunnel protocol/transport gre multipoint, key disabled, sequencing disabled
  Tunnel endpoint 2.2.2.2
  Tunnel endpoint 3.3.3.3
  Tunnel TTL 64
  Checksumming of packets disabled, path MTU discovery disabled
  Tunnel protection via IPsec (profile "default")
  Router Advertisement is disabled
  Router Advertisement default routes are accepted
  Router Advertisement prefix info is accepted
    input packets 56, bytes 3804, dropped 0, multicast packets 0
    output packets 121, bytes 5364, multicast packets 0, broadcast packets 0
    input average rate : 30 seconds 0 bps, 5 minutes 0 bps
    output average rate: 30 seconds 0 bps, 5 minutes 0 bps
    input peak rate 972 bps at 2019/03/14 00:30:13
    output peak rate 1.53 Kbps at 2019/03/14 00:30:13
  Time since last state change: 0 days 02:18:50
```

Neighbor Table

show arp

```
awplus#show arp
IP Address       LL Address       Interface            Port        Type
1.1.1.254        5254.b536.031d   eth1                 -           dynamic
172.16.1.2       2.2.2.2          tunnel1              -           dynamic
172.16.1.3       3.3.3.3          tunnel1              -           dynamic
```

IPsec peers

show ipsec peer

```
awplus#show ipsec peer
IPsec
  Selectors (local:remote)
    Address:  1.1.1.1 : any
    Protocol: gre:gre
    Port:     any:any
    Mark:     any:any
  Profile: default
    SPI (In:Out): c11b381d:cdfcfc96
        Selectors:  1.1.1.1/32[gre]     3.3.3.3/32[gre]
      Proto:      ESP
      Mode:       transport
      Encryption: AES256
      Integrity:  SHA256
      Expires:    28609s
IPsec
  Selectors (local:remote)
    Address:  1.1.1.1 : any
    Protocol: gre:gre
    Port:     any:any
    Mark:     any:any
  Profile: default
    SPI (In:Out): cf1b4338:c3cf1f3d
        Selectors:  1.1.1.1/32[gre]     2.2.2.2/32[gre]
      Proto:      ESP
      Mode:       transport
      Encryption: AES256
      Integrity:  SHA256
      Expires:    28400s
ISAKMP
  LocalID:  hub
  RemoteID: spoke1
    Cookies (Initiator:Responder) d67d8e23b5626547:cadf02b393a4047c
      Ver:            2             Lifetime: 78693s    State: Established
      Authentication: PSK           Group:    14
      Encryption:     AES256        NATT:     no
      Integrity:      SHA256        DPD:      yes
ISAKMP
  LocalID:  hub
  RemoteID: spoke2
    Cookies (Initiator:Responder) 654198f55f608edf:dcc76eadafe9d6aa
      Ver:            2             Lifetime: 86176s    State: Established
      Authentication: PSK           Group:    14
      Encryption:     AES256        NATT:     no
      Integrity:      SHA256        DPD:      yes
```

Multipoint hub and spoke (unique key per spoke)

Example 4 This example configures a single VTI on a Hub with a unique key for each spoke. The remote endpoints are added to the hub by creating their unique pre-shared keys for each spoke.

Step 1: **Configure the hub (point-to-multipoint)**

```
awplus#configure terminal
awplus(config)#crypto isakmp key friend1 hostname spoke1
awplus(config)#crypto isakmp key friend2 hostname spoke2
awplus(config)#interface eth1
awplus(config-if)#ip address 1.1.1.1/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.1.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.1/28
awplus(config-if)#tunnel mode gre multipoint
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel endpoint dynamic
awplus(config-if)#tunnel local name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.2.0/24 172.16.1.2
awplus(config)#ip route 192.168.3.0/24 172.16.1.3
awplus(config)#ip route 0.0.0.0/0 1.1.1.254
```

Step 2: **Configure spoke1 (point-to-point)**

```
awplus(config)#crypto isakmp key friend1 hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 2.2.2.2/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.2.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.2/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke1
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.0.0/16 172.16.1.1
awplus(config)#ip route 0.0.0.0/0 2.2.2.254
```

Step 3: **Configure spoke2 (point-to-point)**

```
awplus(config)#crypto isakmp key friend2 hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 3.3.3.3/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.3.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.3/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke2
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.0.0/16 172.16.1.1
awplus(config)#ip route 0.0.0.0/0 3.3.3.254
```

Multipoint hub and spoke with radius authentication

Example 5 This example configures a single VTI on a Hub. No configuration changes are required to add new spokes. Spoke authentication is managed via an external radius server.

Step 1: **Configure the hub (point-to-multipoint)**

```
awplus#configure terminal
awplus(config)#radius-server host 192.168.1.100 key <password>
awplus(config)#aaa authentication isakmp default group radius
awplus(config)#crypto isakmp profile spoke
awplus(config-isakmp-profile)#remote authentication eap-radius
awplus(config-isakmp-profile)#transform 1 integrity SHA1 encryption AES128
group 14
awplus(config-isakmp-profile)#exit
awplus(config)#crypto isakmp peer dynamic profile spoke
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 1.1.1.1/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.1.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.1/28
awplus(config-if)#tunnel mode gre multipoint
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel endpoint dynamic
awplus(config-if)#tunnel local name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.2.0/24 172.16.1.2
awplus(config)#ip route 192.168.3.0/24 172.16.1.3
awplus(config)#ip route 0.0.0.0/0 1.1.1.254
```

Step 2: **Configure spoke1 (point-to-point)**

```
awplus(config)#crypto isakmp profile hub
awplus(config-isakmp-profile)#local authentication eap-radius
awplus(config-isakmp-profile)#transform 1 integrity SHA1 encryption AES128
group 14
awplus(config-isakmp-profile)#exit
awplus(config)#crypto isakmp peer hub profile hub
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#crypto isakmp key friend1 hostname hub type eap
awplus(config)#interface eth1
awplus(config-if)#ip address 2.2.2.2/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.2.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.2/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke1
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.0.0/16 172.16.1.1
awplus(config)#ip route 0.0.0.0/0 2.2.2.254
```

Step 3: **Configure spoke2 (point-to-point)**

```
awplus(config)#crypto isakmp profile hub
awplus(config-isakmp-profile)#local authentication eap-radius
awplus(config-isakmp-profile)#transform 1 integrity SHA1 encryption AES128
group 14
awplus(config-isakmp-profile)#exit
awplus(config)#crypto isakmp peer hostname hub profile hub
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#crypto isakmp key friend2 hostname hub type eap
awplus(config)#interface eth1
awplus(config-if)#ip address 3.3.3.3/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.3.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.3/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke2
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.0.0/16 172.16.1.1
awplus(config)#ip route 0.0.0.0/0 3.3.3.254
```

Meshed VPN

Example 6 This example configures a single multipoint tunnel on each node and supports a single pre-shared key for the entire network. Any node can be configured to talk to any other node by company policy. Static routing, OSPF, or BGP can be used.

Step 1: **Configure node1 (point-to-multipoint)**

```
awplus#configure terminal
awplus(config)#crypto isakmp key friend hostname node1
awplus(config)#interface eth1
awplus(config-if)#ip address 1.1.1.1/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.1.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.1/28
awplus(config-if)#tunnel mode gre multipoint
awplus(config-if)#tunnel source 1.1.1.1
awplus(config-if)#tunnel local name node1
awplus(config-if)#tunnel endpoint 2.2.2.2
awplus(config-if)#tunnel endpoint 3.3.3.3
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.2.0/24 172.16.1.2
awplus(config)#ip route 192.168.3.0/24 172.16.1.3
awplus(config)#ip route 0.0.0.0/0 1.1.1.254
```

Step 2: Configure node2 (point-to-multipoint)

```
awplus(config)#crypto isakmp key friend hostname node2
awplus(config)#interface eth1
awplus(config-if)#ip address 2.2.2.2/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.2.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.2/28
awplus(config-if)#tunnel mode gre multipoint
awplus(config-if)#tunnel source 2.2.2.2
awplus(config-if)#tunnel local name node2
awplus(config-if)#tunnel endpoint 1.1.1.1
awplus(config-if)#tunnel endpoint 3.3.3.3
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.1.0/24 172.16.1.1
awplus(config)#ip route 192.168.3.0/24 172.16.1.3
awplus(config)#ip route 0.0.0.0/0 2.2.2.254
```

Step 3: Configure node3 (point-to-multipoint)

```
awplus(config)#crypto isakmp key friend hostname node3
awplus(config)#interface eth1
awplus(config-if)#ip address 3.3.3.3/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.3.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.3/28
awplus(config-if)#tunnel mode gre multipoint
awplus(config-if)#tunnel source 3.3.3.3
awplus(config-if)#tunnel local name node3
awplus(config-if)#tunnel endpoint 1.1.1.1
awplus(config-if)#tunnel endpoint 2.2.2.2
awplus(config-if)#tunnel protection ipsec
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 192.168.1.0/24 172.16.1.1
awplus(config)#ip route 192.168.2.0/24 172.16.1.2
awplus(config)#ip route 0.0.0.0/0 3.3.3.254
```

Multipoint VPN with OSPF routing

Example 7 This example shows step-by-step instructions to configure a point-to-multipoint IPv4 GRE tunnel between a main office node to a pair of remote office nodes. The point-to-multipoint VPNs in this hub-and-spoke topology are protected by IPsec. OSPF is configured to run in point-to-multipoint mode over the virtual tunnel interfaces.

Step 1: **Configure the hub (point-to-multipoint)**

```
awplus#configure terminal
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 1.1.1.1/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.1.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.1/28
awplus(config-if)#tunnel mode gre multipoint
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel endpoint dynamic
awplus(config-if)#tunnel local name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#ip tcp adjust-mss pmtu
awplus(config-if)#ip ospf network point-to-multipoint
awplus(config-if)#exit
awplus(config)#router ospf 1
awplus(config-router)#ospf router-id 172.16.1.1
awplus(config-router)#passive-interface vlan1
awplus(config-router)#network 172.16.1.0/28 area 0
awplus(config-router)#network 192.168.1.0/24 area 0
awplus(config-router)#exit
awplus(config)#ip route 0.0.0.0/0 1.1.1.254
awplus(config)#end
```

Step 2: Configure spoke1 (point-to-multipoint)

```
awplus#configure terminal
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 2.2.2.2/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.2.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.2/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke1
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#ip tcp adjust-mss pmtu
awplus(config-if)#ip ospf network point-to-multipoint
awplus(config-if)#exit
awplus(config)#router ospf 1
awplus(config-router)#ospf router-id 172.16.1.2
awplus(config-router)#passive-interface vlan1
awplus(config-router)#network 172.16.1.0/28 area 0
awplus(config-router)#network 192.168.2.0/24 area 0
awplus(config-router)#exit
awplus(config)#ip route 0.0.0.0/0 2.2.2.254
awplus(config)#end
```

Step 3: Configure spoke2 (point-to-multipoint)

```
awplus#configure terminal
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 3.3.3.3/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.3.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.3/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke2
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#ip tcp adjust-mss pmtu
awplus(config-if)#ip ospf network point-to-multipoint
awplus(config-if)#exit
awplus(config)#router ospf 1
awplus(config-router)#ospf router-id 172.16.1.3
awplus(config-router)#passive-interface vlan1
awplus(config-router)#network 172.16.1.0/28 area 0
awplus(config-router)#network 192.168.3.0/24 area 0
awplus(config-router)#exit
awplus(config)#ip route 0.0.0.0/0 3.3.3.254
awplus(config)#end
```

Note: OSPF on the spokes will cause the tunnels to initiate as they will send multicast packets out on tunnel1.

The routing tables of the hub and spoke1 can be viewed via the **show ip route** command as follows:

Example 7 IP Route tables

Hub routing table

```
awplus#show ip route
Codes: C - connected, S - static, R - RIP, B - BGP
O - OSPF, D - DHCP, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2

- candidate default
Gateway of last resort is 1.1.1.254 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 1.1.1.254, eth1
C 1.1.1.0/24 is directly connected, eth1
C 172.16.1.0/28 is directly connected, tunnel1
C 192.168.1.0/24 is directly connected, vlan1
O 192.168.2.0/24 [110/2] via 172.16.1.2, tunnel1, 00:02:09
O 192.168.3.0/24 [110/2] via 172.16.1.3, tunnel1, 00:02:09

awplus#sh ip ospf route

OSPF process 1:
Codes: C - connected, D - Discard, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2

O 172.16.1.2/32 [1] via 172.16.1.2, tunnel1, Area 0.0.0.0
O 172.16.1.3/32 [1] via 172.16.1.3, tunnel1, Area 0.0.0.0
C 192.168.1.0/24 [1] is directly connected, vlan1, Area 0.0.0.0
O 192.168.2.0/24 [2] via 172.16.1.2, tunnel1, Area 0.0.0.0
O 192.168.3.0/24 [2] via 172.16.1.3, tunnel1, Area 0.0.0.0
```

Spoke1 routing table

```
awplus#show ip route
Codes: C - connected, S - static, R - RIP, B - BGP
O - OSPF, D - DHCP, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2

- candidate default
Gateway of last resort is 2.2.2.254 to network 0.0.0.0

S* 0.0.0.0/0 [1/0] via 2.2.2.254, eth1
C 2.2.2.0/24 is directly connected, eth1
C 172.16.1.0/28 is directly connected, tunnel1
O 192.168.1.0/24 [110/2] via 172.16.1.1, tunnel1, 00:01:21
C 192.168.2.0/24 is directly connected, vlan1
O 192.168.3.0/24 [110/3] via 172.16.1.1, tunnel1, 00:01:21

awplus#show ip ospf route

OSPF process 1:
Codes: C - connected, D - Discard, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2

O 172.16.1.1/32 [1] via 172.16.1.1, tunnel1, Area 0.0.0.0
O 172.16.1.3/32 [2] via 172.16.1.1, tunnel1, Area 0.0.0.0
O 192.168.1.0/24 [2] via 172.16.1.1, tunnel1, Area 0.0.0.0
C 192.168.2.0/24 [1] is directly connected, vlan1, Area 0.0.0.0
O 192.168.3.0/24 [3] via 172.16.1.1, tunnel1, Area 0.0.0.0
```

Multipoint VPN with BGP routing

Example 8 This example configures a simple non-fully meshed Multipoint GRE VPN topology with eBGP routing. Each device is configured with a unique ASN from the reserved range, and each VPN is IPsec protected. There is eBGP peering from each spoke to the main hub.

Step 1: Configure the hub (point-to-multipoint)

```
awplus#configure terminal
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 1.1.1.1/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.1.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.1/28
awplus(config-if)#tunnel mode gre multipoint
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel endpoint dynamic
awplus(config-if)#tunnel local name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#ip tcp adjust-mss pmtu
awplus(config-if)#exit
awplus(config)#router bgp 65000
awplus(config-router)#network 192.168.1.0/24
awplus(config-router)#neighbor 172.16.1.2 remote-as 65001
awplus(config-router)#neighbor 172.16.1.2 ebgp-multihop 255
awplus(config-router)#neighbor 172.16.1.3 remote-as 65002
awplus(config-router)#neighbor 172.16.1.3 ebgp-multihop 255
awplus(config-router)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 0.0.0.0/0 1.1.1.254
```

Step 2: **Configure spoke1 (point-to-point)**

```
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 2.2.2.2/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.2.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.2/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke1
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#ip tcp adjust-mss pmtu
awplus(config-if)#exit
awplus(config)#router bgp 65001
awplus(config-router)#network 192.168.2.0/24
awplus(config-router)#neighbor 172.16.1.1 remote-as 65000
awplus(config-router)#neighbor 172.16.1.1 ebgp-multihop 255
awplus(config-router)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 0.0.0.0/0 2.2.2.254
```

Step 3: **Configure spoke2 (point-to-point)**

```
awplus(config)#crypto isakmp key friend hostname hub
awplus(config)#interface eth1
awplus(config-if)#ip address 3.3.3.3/24
awplus(config-if)#interface vlan1
awplus(config-if)#ip address 192.168.3.254/24
awplus(config-if)#interface tunnel1
awplus(config-if)#ip address 172.16.1.3/28
awplus(config-if)#tunnel mode gre
awplus(config-if)#tunnel source eth1
awplus(config-if)#tunnel destination 1.1.1.1
awplus(config-if)#tunnel local name spoke2
awplus(config-if)#tunnel remote name hub
awplus(config-if)#tunnel protection ipsec
awplus(config-if)#exit
awplus(config)#router bgp 65002
awplus(config-router)#network 192.168.3.0/24
awplus(config-router)#neighbor 172.16.1.1 remote-as 65000
awplus(config-router)#neighbor 172.16.1.1 ebgp-multihop 255
awplus(config-router)#exit
```

Configure a static route via the tunnel interface:

```
awplus(config)#ip route 0.0.0.0/0 3.3.3.254
```

The routing tables of the hub and spoke1 can be viewed via the **show ip route** command as follows:

Example 8 IP route tables

Hub routing table

```
awplus#show ip route
Codes: C - connected, S - static, R - RIP, B - BGP
   O - OSPF, D - DHCP, IA - OSPF inter area
  N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
  E1 - OSPF external type 1, E2 - OSPF external type 2
   * - candidate default

Gateway of last resort is 1.1.1.254 to network 0.0.0.0

S*   0.0.0.0/0 [1/0] via 1.1.1.254, eth1
C   1.1.1.0/24 is directly connected, eth1
C   172.16.1.0/28 is directly connected, tunnel1
C   192.168.1.0/24 is directly connected, vlan1
B  192.168.2.0/24 [20/0] via 172.16.1.2, tunnel1, 02:18:13 <- HUB learns Spoke LANs
B  192.168.3.0/24 [20/0] via 172.16.1.3, tunnel1, 02:11:08
```

Spoke1 routing table

```
awplus#show ip route
Codes: C - connected, S - static, R - RIP, B - BGP
   O - OSPF, D - DHCP, IA - OSPF inter area
  N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
  E1 - OSPF external type 1, E2 - OSPF external type 2
   * - candidate default

Gateway of last resort is 2.2.2.254 to network 0.0.0.0

S*   0.0.0.0/0 [1/0] via 2.2.2.254, eth1
C   2.2.2.0/24 is directly connected, eth1
C   172.16.1.0/28 is directly connected, tunnel1
B   192.168.1.0/24 [20/0] via 172.16.1.1, tunnel1, 02:11:14
C   192.168.2.0/24 is directly connected, vlan1
B   192.168.3.0/24  [20/0]  via 172.16.1.1,  tunnel1,  02:08:35 <- Spoke-Spoke via HUB
```