

Configure GRE Tunnels

David Davis

Takeaway: Using generic routing encapsulation (GRE) tunnels on Cisco routers can come in handy with Cisco router administration, and configuring GRE tunnels is relatively easy. David Davis has the details in this edition of Cisco Routers and Switches.

Originally developed by Cisco, generic routing encapsulation (GRE) is now a standard, defined in RFC 1701, RFC 1702, and RFC 2784. GRE is a tunneling protocol used to transport packets from one network through another network.

If this sounds like a virtual private network (VPN) to you, that's because it theoretically is: Technically, a GRE tunnel is a type of a VPN—but it isn't a secure tunneling method. However, you can encrypt GRE with an encryption protocol such as IPSec to form a secure VPN.

In fact, the point-to-point tunneling protocol (PPTP) actually uses GRE to create VPN tunnels. For example, if you configure Microsoft VPN tunnels, by default, you use PPTP, which uses GRE.

Why would I use GRE?

Why would you tunnel traffic using GRE? Here are some of the reasons:

- You need to encrypt multicast traffic. GRE tunnels can carry multicast packets—just like real network interfaces—as opposed to using IPSec by itself, which can't encrypt multicast traffic. Some examples of multicast traffic are OSPF, EIGRP, and RIPV2. Also, a number of video, VoIP, and streaming music applications use multicast.
- You have a protocol that isn't routable, such as NetBIOS or non-IP traffic over an IP network. For example, you could use GRE to tunnel IPX or AppleTalk through an IP network.
- You need to connect two similar networks connected by a different network with different IP addressing.

How do I configure GRE tunnels?

Configuring GRE tunnels on Cisco routers is relatively easy—all it takes is a few simple commands. Here's an example of a simple configuration:

Router A:

```
interface Ethernet0/1
ip address 10.2.2.1 255.255.255.0

interface Serial0/0
ip address 192.168.4.1 255.255.255.0

interface Tunnel0
ip address 1.1.1.2 255.255.255.0
tunnel source Serial0/0
tunnel destination 192.168.4.2
```

Router B:

```
interface FastEthernet0/1
ip address 10.1.1.1 255.255.255.0
```

Revised December 7, 2010

Configure GRE Tunnels

David Davis

```
interface Serial0/0
ip address 192.168.4.2 255.255.255.0
```

```
interface Tunnel0
ip address 1.1.1.1 255.255.255.0
tunnel source Serial0/0
tunnel destination 192.168.4.1
```

In this example, the two routers each have a virtual interface—the tunnel interface. This interface is its own network, just like a point-to-point T1 circuit would be. The traffic going over the tunnel network is tunneling through the serial network.

To each of the routers, it appears that it has two paths to the remote—the serial interface and the tunnel interface (running through the tunnel). This tunnel could then transmit unroutable traffic such as NetBIOS or AppleTalk. If it's going through the Internet, you could use IPSec to encrypt it.

As you can see in the output below, the tunnel interface on Router B is an interface like any other:

```
RouterB# sh ip int brie
Interface  IP-Address  OK?  Method  Status  Protocol
Ethernet0  10.1.1.1    YES  manual  up      down
Serial0    192.168.4.2 YES  manual  up      up
Serial1    unassigned  YES  unset   administratively down  down
Tunnel0    1.1.1.1    YES  manual  up      up
RouterB#
```

Troubleshooting GRE Tunnels

Because GRE takes one packet and encapsulates it in another packet, you might run into a situation where the packet you're sending is larger than your interface allows. The solution to this issue is to configure `ip tcp adjust-mss 1436` on the tunnel interface.

While GRE doesn't provide encryption, you can enable a key on each side of the tunnel using the `tunnel key` command. This is like a simple clear-text password with no encryption.

Because GRE tunnels are stateless, it's possible for one side of the tunnel to go down while the other side remains up. The solution to this problem is to enable keepalive packets on each side of the tunnel. By doing this, each side of the tunnel periodically sends a keepalive to the other side. If one side doesn't receive a keepalive in the specified time, the tunnels go down on each side.