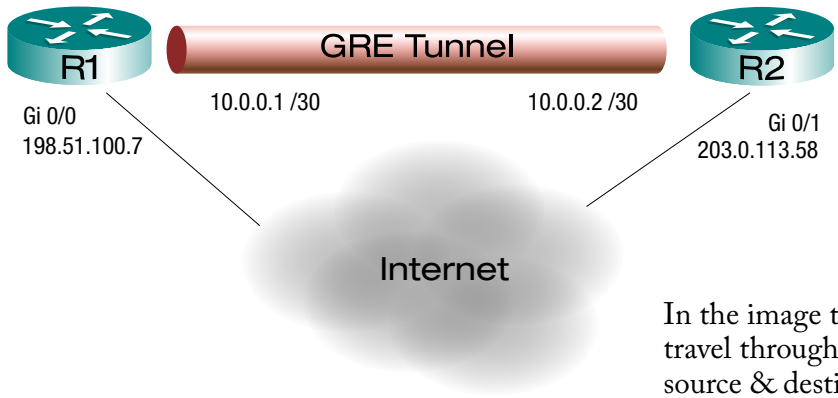
15b. GRE

ccnacookbook.com



GRE (Generic Routing Encapsulation) tunnels have no crypto, but illustrate tunnel interfaces. IPsec can be added for encryption, but not in ICND2.

In the image to the left, packets will seem to travel through the tunnel at the top, and their source & destination addresses will reflect that (10.0.0.1 and 10.0.0.2). In reality though, the IP packets will be encapsulated as data inside GRE packets with the real source and destination addresses (198.51.100.7 and 203.0.113.58) in "delivery headers." Those packets will really be routed through the internet (cloud at bottom). When they arrive and are de-encapsulated, there will no longer be any evidence that the packet didn't physically travel over the imaginary tunnel.

Once encapsulated, a GRE packet looks like this:

Delivery IP header	GRE Header	Original packet with headers

Tunnel Interfaces—Virtual interfaces that form tunnel endpoints. You can tell the router to use the tunnel by naming it in static routes or network statements for routing protocols, just like you would with a "real" interface.

- Tunnels interfaces are within the internal or secure part of the network
- Routing tables can use them as exit interfaces
- The encapsulating tunnel header is analogous to HDLC header on serial point-to-point
- Each end of a tunnel has an IP address in the same subnet
- Routing protocols can form neighbor relationships across tunnels.

CONFIGURATION

Think of what you're creating as a literal "tunnel" or shortcut, where packets enter the tunnel at some location in your network and reappear in some other router that you control, even in a branch office a continent away.

- Name the virtual tunnel interface [Line 1]—I try to avoid using the numbers 0 and 1 because IOS doesn't put a space in front of the number, making the former look like the number 10 and the latter look like a misspelled and unnumbered "tunnell."
- Give your new tunnel interface an IP address [Line 2]—This will activate IP handling on the interface, so it's necessary even if your static routes refer to the link as an exit interface by name, e.g. "tunnel 2," instead of by next-hop address. This address will also allow you to involve the tunnel in routing protocols, using network statements.
- Choose the tunnel source [Line 3]—In terms of routing, this is where your "real" packets will disappear into the tunnel. They'll be encapsulated in a delivery header and this IP address will be used as the source address of that delivery header. You can either specify an exit interface as the source [see R2, right column] or the address of that interface [R1].

- Specify the other end of the tunnel [Line 4]—this can be anywhere in the internet, and the address will match the source address used by the other end. This address will be also be used as the destination address in the delivery header to route the packet through the "real" internet, before revealing it, once de-encapsulated, at the other end of the tunnel
- Specify the kind of tunnel [line 5]—GRE is the default, so this is optional and won't display in a "show running-config"

R1	R2

`Interface Tunnel 2`
`  ip address 10.0.0.1 255.255.255.252`
`  tunnel source 198.51.100.7`
`  tunnel destination 203.0.113.58`
`  tunnel mode gre ip`
	`Interface Tunnel 3`
	`  ip address 10.0.0.2 255.255.255.252`
	`  tunnel source GigabitEthernet0/1`
	`  tunnel destination 198.51.100.7`
	Could be a hostname (DNS)

VERIFICATION

```
R1# show ip interface brief
Interface                  IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0         198.51.100.7    YES manual up                    up
Tunnel2                    10.0.0.1        YES manual up                    up
```

```
R1# show interfaces tunnel 2
Tunnel2 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 10.0.0.1/30
  MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 198.51.100.7, destination 203.0.113.58
  Tunnel protocol/transport GRE/IP
    Key disabled, sequencing disabled
    Checksumming of packets disabled
  Tunnel TTL 255, Fast tunneling enabled
  Tunnel transport MTU 1476 bytes
                    Notice the Maximum Transmission Unit is smaller than the standard 1500 for Ethernet
                    so that it can accommodate the extra headers
```

```
R1# show ip route
                    Legend omitted
      10.0.0.0/8 is variably subnetted, 2 subnets, 2 masks
C        10.0.0.0/30 is directly connected, Tunnel2
L        10.0.0.1/32 is directly connected, Tunnel2
      198.51.100.0/24 is variably subnetted, 2 subnets, 2 masks
C        198.51.100.0/24 is directly connected, GigabitEthernet0/0
L        198.51.100.7/32 is directly connected, GigabitEthernet0/0
S     203.0.113.0/24 [1/0] via 198.51.100.1, GigabitEthernet0/0
```

Traceroute only shows one hop—the inside of the tunnel. The path that the tunnel, itself, takes is invisible. For that you'd do a traceroute on the tunnel destination IP listed in "show interfaces."

```
R1# traceroute 10.0.0.2
Type escape sequence to abort.
Tracing the route to 10.0.0.2
VRF info: (vrf in name/id, vrf out name/id)
  1 10.0.0.2 4 msec *  0 msec
```

TROUBLESHOOTING

The configuration of a GRE tunnel is so simple (3 lines) that troubleshooting really boils down to checking the configuration for correctness.

Tunnel IP Address Problem (up/up)

The first thing that you should realize is that a tunnel will show as up/up even if it doesn't have an IP address; you just won't be able to send IP packets over it. That's because one of the side effects of assigning an IP address to an interface is to start an IP packet handling process on that interface. The "protocol" field refers to an OSI L2 protocol, not IP, which operates at L3.

```
1 R1# show ip interface brief
2 Interface                  IP-Address      OK? Method Status                Protocol
3 GigabitEthernet0/0         198.51.100.7    YES NVRAM  up                    up
4 GigabitEthernet0/1         unassigned      YES NVRAM  administratively down down
5 Serial0/2/0                unassigned      YES NVRAM  administratively down down
6 Serial0/2/1                unassigned      YES NVRAM  administratively down down
7 Tunnel2                    unassigned      YES manual up                    up
```

Your first hint that an up/up tunnel can't pass IP might come when a routing protocol can't form a neighbor relationship across the tunnel.

A simple ping will tell you if an up/up tunnel is really usable. In fact, that ping will test both directions at once.

Tunnel Source (up/down on misconfigured end)

A tunnel source can be specified as either an interface or an IP address. These are just two ways of referring to the exact same thing: an up/up interface with an IP address. If that underlying interface isn't up/up or lacks an IP address, your tunnel will be up/down on the router with the underlying interface problem. The router at the correctly configured end will still be up/up.

Tunnel Destination (up/down on misconfigured end)

A tunnel destination is always an IP address. Hostnames are just for making configuration easier. If you enter a hostname, your router will immediately look up the IP address using DNS and store the address in the running config. Of course, if DNS resolution fails, you'll get an error during configuration.

If the routing table has no route to the destination address, then the tunnel will be up/down. The other end of the tunnel will show as up/up because destination checking is local to each router.

Access Lists

In an extended access list, GRE is its own protocol, just like ICMP, used by ping. A "permit ip" statement will also allow GRE in the same way it allows TCP and UDP.

```
R2(config)# access-list 105 permit gre any any
```

If you use an address instead of "any," it should match the source or destination statement used in the tunnel's creation, not the IP address of the tunnel interface.

This makes it easy to overlook when creating an ACL, leaving GRE packets at the mercy of the implicit "deny any any" at the end of every ACL This is often more of an issue inbound, since outbound ACLs don't affect packets that were created *within* the same router. A GRE encapsulation would count as a packet that originated locally, even if its contents didn't.

DMVPN

DMVPN (Dynamic Multipoint VPN)—A Cisco feature that overcomes the point-to-point nature of GRE and the hub-and-spoke architecture limitations that result. Any site can directly exchange data with any other site in the same tunnel.

NHRP (Next Hop Resolution Protocol)—makes DMVPN work and eliminates significant static configuration tasks.

- One site acts as the NHRP server (the "hub")
- Initially, spokes can directly communicate only with the hub
- Spokes register their public and private IP addresses with the server (hub). These correspond to the GRE source address and the tunnel interface IP address, respectively. Thanks to dynamic registration, you don't have to add configuration to the hub every time you add a site
- For spokes to directly communicate to each other, they each
  - learn about the other's private subnets through a routing protocol
  - learn the other's public address from the NHRP server (the "hub"). This corresponds to the destination statement on a manual GRE tunnel

OUT OF SCOPE BONUS

```
R1(config-if)# tunnel mode ipsec ipv4
```

Default = tunnel mode gre ip
Option for ipv6 tunneling available