Chris Bryant, CCIE #12933 http://www.thebryantadvantage.com/

Back to Index

Tunnels, VPNs, And COPPs

Overview

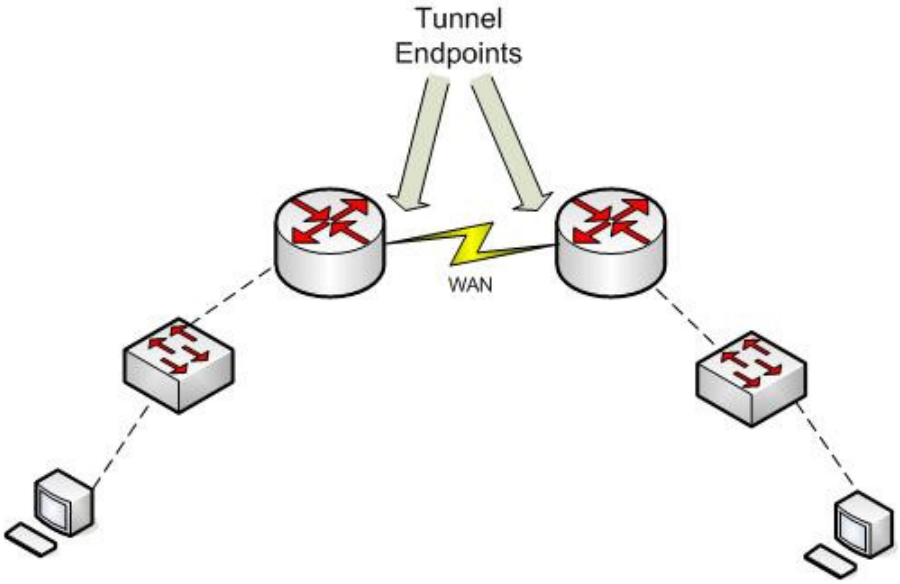
Tunnels and Virtual Private Networks (VPNs) are commonplace in today's networks. Many network admins use VPNs to connect remotely to their networks, as do today's mobile end users.

VPNs are a huge part of your ISCW studies, which may make you wonder why I'm mentioning them here in your ONT Study Guide!

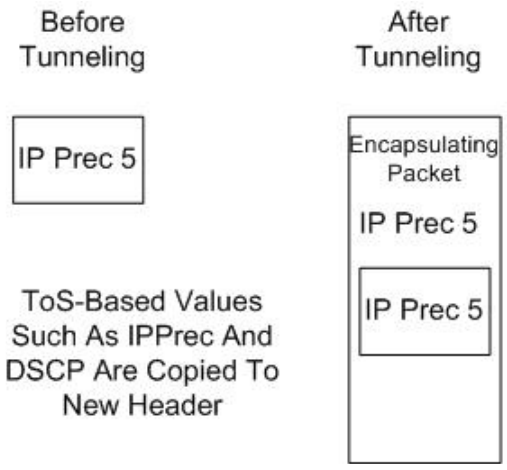
While configuring and troubleshooting VPNs are not part of the ONT exam blueprint, knowing how QoS is carried out with tunneling *is*. Before we get into a discussion of how QoS and VPNs work together, let's review just what these tunnels are!

The "V" in VPN stands for *Virtual*, and that's exactly the kind of connection you have when you "VPN in" to your network. You "tunnel" through an existing physical line to get to your destination, and that tunnel is a logical one.

The actual tunneling occurs by placing one packet into *another* packet and then transmitting that *tunneled packet* over the logical connection. The tunnels are configured on the routers - the switches and PCs in the following example don't even know that tunneling is occurring.



That all sounds great - and it is, believe me - but it does create an issue with QoS. (You were just waiting for that shoe to drop, weren't you?) I mentioned earlier that the tunneling itself takes place by placing the original packet into another packet... and that's where the problem comes in. What if a packet is carrying an IP Precedence value of 5, and then it's placed into another packet?



This would actually be okay, since the ToS will be copied from the original packet to the header of the encapsulating packet.

That's a very important concept to keep in mind, so allow me to repeat it - the ToS value is copied from the "untunneled" header to the tunnel header.

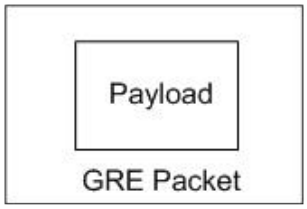
Therefore, if we're using IP Precedence or DSCP for QoS policies, we have no problems.

If we're using other values, such as a source or destination IP address or port number, we do have a potential issue, because those values are not copied to the header of the encapsulating packet.

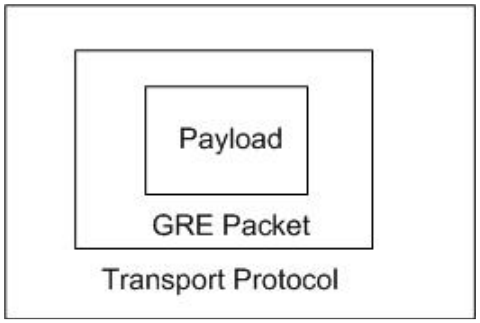
As you might expect, we've got a solution to this issue - but before we get to the solution, let's take a quick look at *GRE tunneling* and how to configure such a tunnel.

Generic Routing Encapsulation (GRE)

GRE uses IP protocol 47, is defined in RFC 1702, and follows a basic two-step process. First, the payload is encapsulated in a GRE packet...



... and the GRE packet itself is then encapsulated by the *transport protocol*, the protocol that will be used to transport the packet across the tunnel. (GRE is the *carrier protocol*, and the protocol at the very center of this process is the *passenger protocol*.)



GRE has been largely replaced in today's networks by IPSec, and IPSec actually offers us two choices - *tunnel mode* and *transport mode.* You'll learn (much) more about these options in your ISCW studies, but for now it's enough to know that all three of the tunneling options we've

discussed...

- GRE
- IPSec Tunnel mode
- IPSec Transport mode

... all copy the original ToS setting to the "outside" IP header. Therefore, if we're classifying traffic by ToS, tunneling doesn't cause any issues. But if we're using another value, an issue does exist - an issue that really wastes all of our earlier QoS configurations.

Regardless of which option you're using, the packets we send across the tunnel will have identical tunnel headers, which means they'll all be treated the same.

That means that important, jitter-sensitive voice traffic will be treated the same as "regular" data traffic. Not that regular data isn't important, but packet equality is the *opposite* of what we're trying to accomplish with QoS; the basic concept of QoS is that we don't *want* every packet treated identically.

Luckily, Cisco routers offer *QoS Preclassification*, a feature that allows a packet to be classified by its original QoS marking before the tunneling process makes that impossible. Basically, QoS Preclassification makes it possible to apply a QoS policy to tunneled packets, which would otherwise be impossible due to the encapsulation process.

QoS Preclassification

Ads for loans and credit cards often mention that you've been "pre-qualified" for that loan or card, which means that you're guaranteed to get the credit without having to go through a long approval process. QoS Preclassification is something like that, since the original packet will be guaranteed to keep its original QoS value without copying that value to its new IP header.

Before we configure QOS preclassification, we need a tunnel! We'll use GRE in this example. Our Serial interface and tunnel interface IP addresses:

R1: Serial0/1, 10.1.1.1 /24; Tunnel0, 192.168.1.1 /24

R2: Serial0/1, 10.1.1.2 /24; Tunnel0, 192.168.1.2 /24

Building a GRE tunnel is beyond the scope of the ONT exam, but let's build one and then verify it.

R1 and R2 are directly connected, with R1 as the DCE. You recall from your CCNA studies that the DCE must supply clockrate to the DTE.

Here's the serial interface config on both routers.

```
R1:

interface Serial0/1
 ip address 10.1.1.1 255.255.255.0
 clock rate 56000

R2:

interface Serial0/1
 ip address 10.1.1.2 255.255.255.0
```

Now we'll build the tunnel. When you build a tunnel, you must specify the IP address of both the source and destination interfaces of the tunnel. We have an option to use IPv6 addresses instead of IPv4, as verified by IOS Help:

```
R1(config)#int tunnel ?
  <0-2147483647>  Tunnel interface number

R1(config)#int tunnel 0
R1(config-if)#ip address 192.168.1.1 255.255.255.0
R1(config-if)#tunnel source 10.1.1.1
R1(config-if)#tunnel destination ?
  Hostname or A.B.C.D  ip address or host name
  X:X:X:X::X                    IPv6 address

R1(config-if)#tunnel destination 10.1.1.2



R2(config)#int tunnel0

*Mar  1 01:29:58.689: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Tunnel0, c
hanged state to down
R2(config-if)#ip address 192.168.1.2 255.255.255.0
R2(config-if)#tunnel source 10.1.1.2
R2(config-if)#tunnel destination 10.1.1.1

*Mar  1 01:30:12.516: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Tunnel0, c
hanged state to up
```

Note that on R2, the router generated a "line protocol down" message for the tunnel interface. Once the source and destination IP addresses were configured, the line protocol goes right back up.

Verify the tunnel interface with *show interface tunnel*. Note the highlighted portions below.

```
R2#show interface tunnel 0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 192.168.1.2/24
  MTU 1514 bytes, BW 9 Kbit, DLY 500000 usec,
```

```
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel source 10.1.1.2, destination 10.1.1.1
  Tunnel protocol/transport GRE/IP
    Key disabled, sequencing disabled
    Checksumming of packets disabled
  Tunnel TTL 255
  Fast tunneling enabled
  Tunnel transmit bandwidth 8000 (kbps)
  Tunnel receive bandwidth 8000 (kbps)
  < output truncated at this point >
```

To enable QoS preclassification on a tunnel, simply run the *qos pre-classify* command. And if you forget what this feature does, the router does a great job of reminding you!

```
R2(config)#int tunnel0
R2(config-if)#qos ?
  pre-classify  Enable QOS classification before packets are tunnel
encapsulated

R2(config-if)#qos pre-classify

R2(config)#int tunnel0
R2(config-if)#qos pre-classify
R2(config-if)#service-policy output ONTPASS
```

Note that this example shows you how to configure QoS Preclassification on a GRE tunnel. While both GRE and IPSec support preclassification, the configuration is different. When you're working with IPSec, you need to configure preclassification on the appropriate *crypto map,* as shown below.

```
R2(config)#crypto map PASSTHEONT
R2(config-crypto-map)#qos pre-classify
```

It bears repeating:

With GRE, the *qos pre-classify* command is configured on the tunnel interface.

With IPSec, the command should be placed in the appropriate crypto-map.

Applying QoS Preclassification

In the previous example, I configured Preclassification on the tunnel interface. We actually have three options for applying this feature:

- Tunnel interfaces
- Crypto Maps

- Virtual Templates

When it comes to applying the actual policy, we can apply it to either the tunnel interface or the physical interface. We've got two decisions to make before doing so:

- Do we need QoS Preclassfication?
- On which interface should the policy be applied?

Cisco's official recommendation for the various scenarios is as follows:

- If the policy is to be applied to the tunnel interface only and packets are to be classified according to the pre-tunnel header, do NOT use the *qos pre-classify* command.

- If the policy is to be applied to the physical interface and packets are to be classified according to the pre-tunnel header, DO use the *qos pre-classify* command.

- If the policy is to be applied to the physical interface and packets should be classified according to the post-tunnel header, you obviously wouldn't use the *qos pre-classify* command.

Additionally, a policy applied to a physical interface with multiple tunnels will result in all tunnels on that interface being subject to that policy.

When And Where To Use QoS Preclassification

If the policy involves values derived from the ToS - namely, IP Precedence and DSCP - QoS Preclassfication is not necessary, since the ToS value is copied from the header of the encapsulate packet to the header of the encapsulating packet.

If the policy involves source and destination IP addresses, port numbers, or other values not derived from the ToS, QoS Preclassification is necessary to ensure that high-priority packets are given the attention they deserve.

QoS Service Level Agreements (SLA)

An SLA is simply the agreement between a customer and an ISP.

The ISP promises to supply your network with a certain level of service, and you agree to supply the ISP with money.

The more money you supply, the higher level of service your network receives!

The Control Plane, The Management Plane, And The COPPs That Protect Them

A Cisco router has four overall planes:

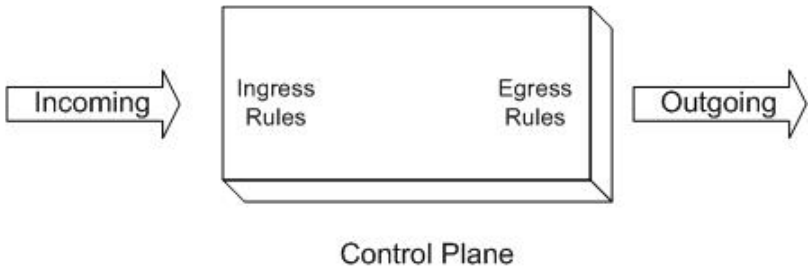
- Control
- Management
- Data
- Service

It sounds like Control Plane Policing would protect only the control plane, but in face CoPP protects both the control and management planes. The primary enemy we're defending these planes against is a DoS attack.

The number and variation of attacks on the Control Plane, as well as the need for QoS at the Control Plane level, has led Cisco to recommend CoPP configuration as a best practice. Before we configure CoPP, we should know what the Control Plane actually does!

A Cisco router's control plane is vital to the overall operation of the router. This plane handles network control traffic, including keepalives and routing update packets - and without those, we don't have much of a network!

CoPP allows us to configure QoS and security rules on the Control Plane just as if it were a regular router port. That means we can have one set of rules for incoming packets ("ingress port") and another set of rules for exiting packets ("egress port"). After we configure CoPP, the resulting service policy will be applied directly to the Control Plane rather than a physical interface.



Cisco's website lists four steps to a successful CoPP configuration:

- Define packet classification criteria

- Define the service policy
- Enter control-plane configuration mode
- Apply QoS policy

We'll walk through a simple COPP configuration here. Let's say we want to allow NTP traffic sourced from 100.1.1.1 to cross the control plane, but deny all other NTP traffic. The first step in our CoPP procedure, "define packet classification criteria", is a fancy way of saying "write an ACL and then a class-map that calls that ACL.

Here's our ACL...

```
R1(config)#access-list 125 deny udp host 100.1.1.1 any eq ntp
R1(config)#access-list 125 perm udp any any eq ntp
```

.... and here's our class-map.

```
R1(config)#class-map NTP_CLASS
R1(config-cmap)#match access-group 125
```

Now we need a policy-map that refers to that class-map, and then we'll define the action for traffic that matches that class-map. We could police the other NTP traffic, but in this case, we'll drop it.

```
R1(config)#policy-map NTP_POLICE
R1(config-pmap)#class NTP_CLASS
R1(config-pmap-c)#?
QoS policy-map class configuration commands:
  bandwidth        Bandwidth
  compression      Activate Compression
  drop           Drop all packets
  estimate         estimate resources required for this class
  exit             Exit from QoS class action configuration mode
  netflow-sampler  NetFlow action
  no               Negate or set default values of a command
  police           Police
  priority         Strict Scheduling Priority for this Class
  queue-limit      Queue Max Threshold for Tail Drop
  random-detect    Enable Random Early Detection as drop policy
  service-policy   Configure Flow Next
  set              Set QoS values
  shape            Traffic Shaping

R1(config-pmap-c)#drop
```

Remember, that drop is applied to packets that match the class-map's ACL. NTP packets from 100.1.1.1 will not match that ACL, so they'll be permitted. All other NTP packets do match that ACL, so they'll be dropped. All remaining packet types do not match that ACL, since they're hitting the implicit deny.

Now we need to apply the policy-map to the control plane. We'll first go into *control plane configuration mode* with the control-plane command. There are no other additional options for this command, as verified by IOS

Help. You may not have seen this prompt before, so watch for it on the exam. :)

```
R1(config)#control-plane ?
  <cr>

R1(config)#control-plane
R1(config-cp)#
```

We'll use the service-policy command to apply that policy to the control plane. IOS Help shows us our various options, including what happens when you try to apply a policy that you've not yet created!

```
R1(config)#control-plane
R1(config-cp)#service-policy ?
  input   Assign policy-map to the input of an interface
  output  Assign policy-map to the output of an interface

R1(config-cp)#service-policy input ?
  WORD  Policy map name

R1(config-cp)#service-policy input NTP_POLICY
Policy NTP_POLICY does not exist
error: failed to install policy map NTP_POLICY
R1(config-cp)#service-policy input NTP_POLICE
R1(config-cp)#
```

Verify with *show policy-map control-plane* and you're all set!

```
R1#show policy-map control-plane
 Control Plane

  Service-policy input: NTP_POLICE

    Class-map: NTP_CLASS (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0 bps, drop rate 0 bps
      Match: access-group 125
      drop

    Class-map: class-default (match-any)
      465 packets, 253098 bytes
      5 minute offered rate 7000 bps, drop rate 0 bps
      Match: any
```

"Hot Spots And Gotchas"

QoS Preclassification

Preclassification allows packets to be classified before tunneling and encryption make classifying on the packet's true values impossible. It's common to configure preclassification on VPNs, both those using GRE and IPSec.

When using GRE tunnels, place the command on the tunnel interface.

When using IPSec, this command is placed in the crypto map.

Preclassification can also be configured on virtual templates, but you're not responsible for configuring those on the ONT exam.

When your policy is matching on the ToS byte, you don't need to run preclassification as that byte will be copied to the outside header during the encryption process.

A service policy can be applied directly on the tunnel interface, or it can be placed on the physical interface itself. Use these Cisco guidelines to make the decision on whether the pre-classify command should be used.

- If the policy is to be applied to the tunnel interface only and packets are to be classified according to the pre-tunnel header, do NOT use the *qos pre-classify* command.

- If the policy is to be applied to the physical interface and packets are to be classified according to the pre-tunnel header, DO use the *qos pre-classify* command.

- If the policy is to be applied to the physical interface and packets should be classified according to the post-tunnel header, you obviously wouldn't use the *qos pre-classify* command.

CoPP

Control Plane Policing doesn't just protect the Control Plane, but the Management Plane as well.

CoPP can be applied to the Control Plane on an inbound or outbound basis.

What are we protecting theses planes from? Basically, DoS attacks.

We apply CoPP in control-plane configuration mode, which you may not have seen previously. Watch for the "cp" in the prompt:

```
R1(config)#control-plane
R1(config-cp)#service-policy ?
  input   Assign policy-map to the input of an interface
  output  Assign policy-map to the output of an interface
```

That is actually the last step in the CoPP configuration process. Here's an outline of everything we need to do:

- Classify the packets with ACLs
- Define the service policy with the *class-map* command
- Enter Control Plane configuration mode as shown in the previous example
- Apply the policy with the *service-policy* command.