TOPICS COVERED:

IP SECURITY

- AAA [AUTHENTICATION, AUTHORIZATION, ACCOUTING]
- ACCESS-LIST [ACL]
- CISCO ASA THEORY (ZONE BASED)
- CISCO ASA LAB
- uRPF (UNICAST REVERSE PATH FORWARDING)
- SECURITY ATTACKS
  - DATA PLANE ATTACKS
  - MANAGEMENT PLANE ATTACKS
- CONTROL PLANE SECURITY
  - Management Place Protection (MPP)
  - Control Plane Policing (CoPP)
- IPSEC SITE-to-SITE TUNNEL
- GRE TUNNEL
- DMVPN
- MPLS
- OVERLAY TUNNELS
  - LISP
  - VXLAN

AAA [AUTHENTICATION, AUTHORIZATION, ACCOUTING]

AAA

Authentication, authorization and accounting (AAA) is a system for tracking user activities on an IP-based network and controlling their access to network resources. AAA is often being implemented as a dedicated server.

This term is also referred to as the **AAA Protocol.**

1. **Authentication** refers to unique identifying information from each system user, generally in the form of a username and password. System administrators monitor and add or delete authorized users from the system.
2. **Authorization** refers to the process of adding or denying individual user access to a computer network and its resources. Users may be given different authorization levels that limit their access to the network and associated resources.
3. **Accounting** refers to the record-keeping and tracking of user activities on a computer network. For a given time period this may include, but is not limited to, real-time accounting of time spent accessing the network, the network services employed or accessed, capacity and trend analysis, network cost allocations, billing data, login data for user authentication and authorization, and the data or data amount accessed or transferred.

RADIUS vs TACACS+

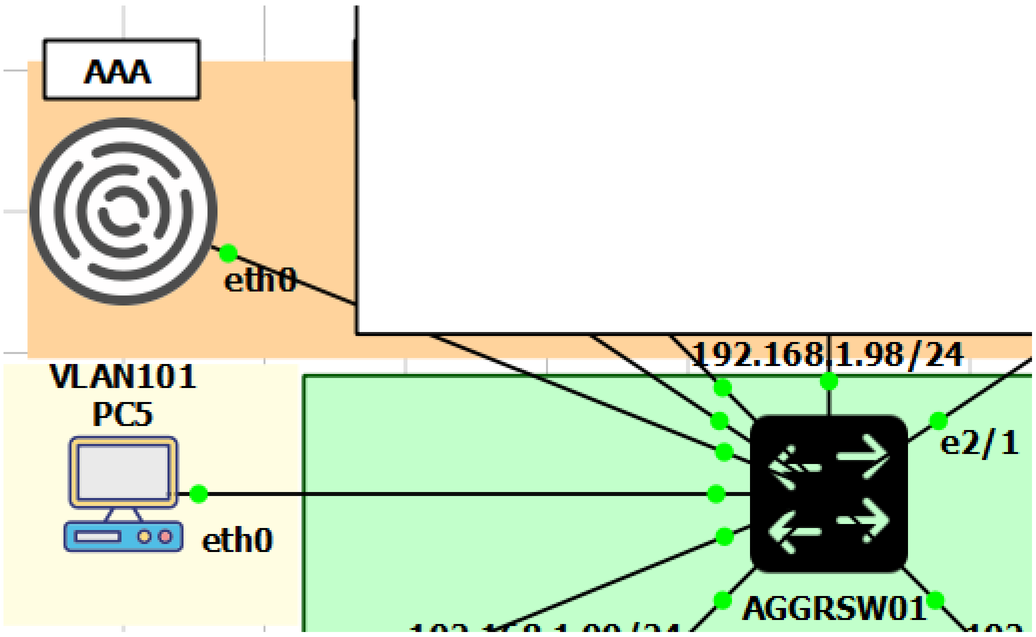
TACACS+	RADIUS
TACACS+ uses TCP	RADIUS uses UDP
TACACS+ uses TCP port 49	Uses port 1812/1645 for authenticating Uses port 1813/1646 for accounting
TACACS+ encrypts the entire communication	RADIUS encrypts passwords only
TACACS+ treats Authentication, Authorization and Accountability differently	RADIUS combines authentication and authorization
TACACS+ is Cisco proprietary protocol	RADIUS is an open protocol
TACACS+ is a heavy-weight protocol consuming more resources	RADIUS is a light-weight protocol consuming less resources
TACACS+ supports 15 privilege levels	RADIUS is limited to privilege mode
Mainly used for Device administration	Mainly used for Network access

*Note: **RADIUS** has been officially assigned UDP ports 1812 for RADIUS authentication and 1813 for RADIUS accounting by the Internet Assigned Numbers Authority (IANA). However, prior to IANA allocation of ports 1812 and 1813, ports 1645 and 1646 (authentication and accounting, respectively) were used unofficially, and became the default ports assigned by many RADIUS client/server implementations at that time. The tradition of using 1645 and 1646 for backwards compatibility continues to this day. For this reason, many RADIUS server implementations monitor both sets of UDP ports for RADIUS* requests.

The early deployment of RADIUS was done using UDP port number 1645, which conflicts with the "**datametrics**" service. The officially assigned port number for RADIUS is 1812 (in 2000).

The early deployment of RADIUS Accounting was done using UDP port number 1646, which conflicts with the "**sa-msg-port**" service. The officially assigned port number for RADIUS Accounting is 1813 (in 2000).

GNS3



CLI Commands: (TACACS+)

```
Switch (config)#
interface vlan 1
no switchport
ip add 192.168.1.101 255.255.255.0
exit

Switch (config)#username admin pri 15 password cisco

Switch (config)#enable password cisco

Switch (config)#aaa new-model

Switch (config)#aaa group server tacacs+ gns3group
Switch (config-sg-tacacs+)#server name container
Switch (config-sg-tacacs+)#exit

Switch (config)#tacacs server container
Switch (config-server-tacacs)#address ipv4 192.168.1.200
Switch (config-server-tacacs)#key gns3

Switch (config-server-tacacs)#exit

Switch (config)# aaa authentication login default group gns3group local → TACACS+

Switch (config)#aaa authentication enable default enable → enable password cisco
```

Verifications:
----

debug aaa authentication
debug tacacs

Do a self **telnet 192.168.29.102**
Username: gns3
Password: gns3

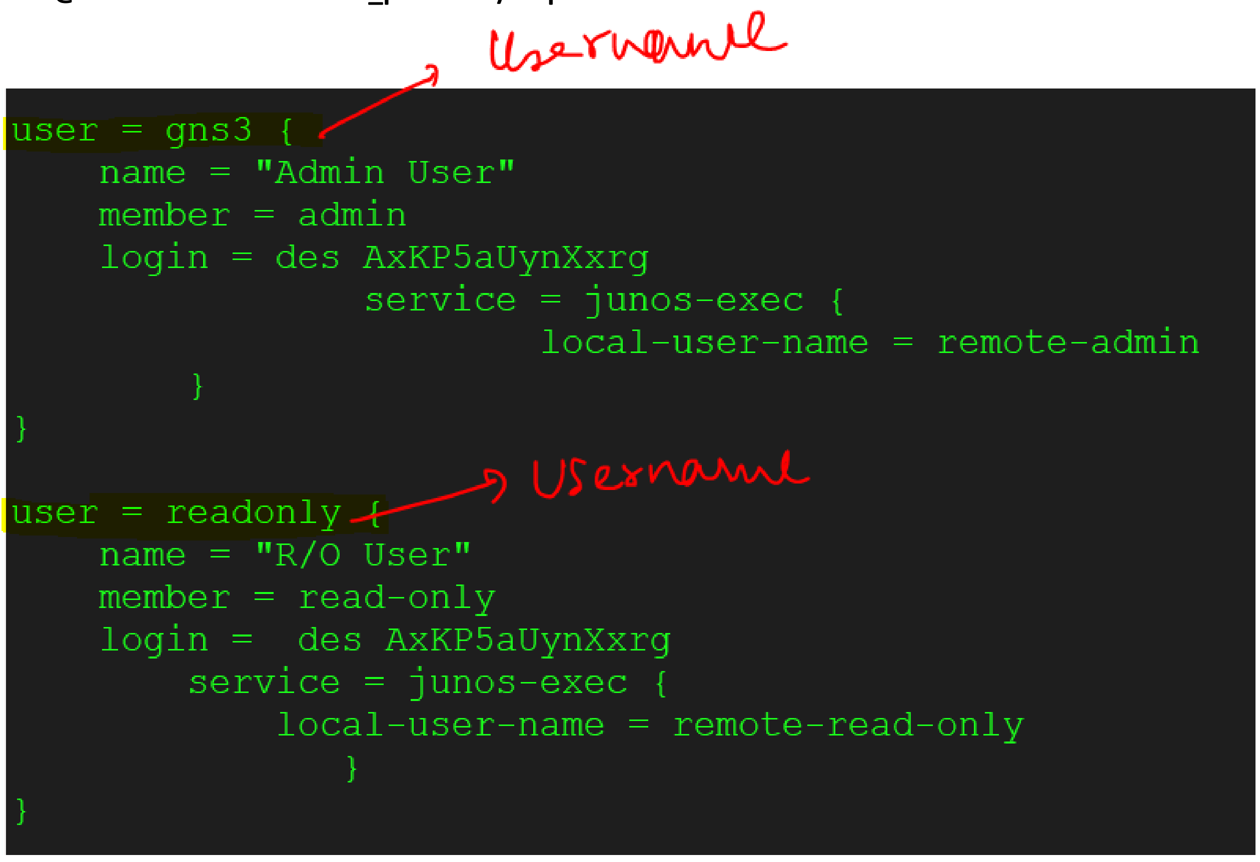
Or

Username: readonly
Password: gns3

Show users
show line

cd /etc/tacacs+
cat tac_plus.conf

root@AAA-1:~# service tacacs_plus start/stop



```
user = gns3 {
    name = "Admin User"
    member = admin
    login = des AxKP5aUynXxrg
            service = junos-exec {
                    local-user-name = remote-admin
        }
}

user = readonly {
    name = "R/O User"
    member = read-only
    login =  des AxKP5aUynXxrg
        service = junos-exec {
            local-user-name = remote-read-only
               }
}
```

CLI Commands: (RADIUS)

Switch(config)#username admin pri 15 password cisco

Switch(config)#aaa new-model

```
Switch(config)#enable password cisco

Switch(config)#aaa group server radius gns3group2
Switch(config-sg-radius)#server name radius1
Switch(config-sg-radius)#exit

Switch(config)# radius server radius1
Switch(config-radius-server)# address ipv4 192.168.29.221 auth-port 1812 acct-port 1813
Switch(config-radius-server)#key gns3

Switch(config)# aaa authentication login default group gns3group2 local → Radius

Switch(config)#aaa authentication enable default enable → enable password cisco
```

*Note: **RADIUS** has been officially assigned UDP ports 1812 for RADIUS authentication and 1813 for RADIUS accounting by the Internet Assigned Numbers Authority (IANA). However, prior to IANA allocation of ports 1812 and 1813, ports 1645 and 1646 (authentication and accounting, respectively) were used unofficially, and became the default ports assigned by many RADIUS client/server implementations at that time. The tradition of using 1645 and 1646 for backwards compatibility continues to this day. For this reason, many RADIUS server implementations monitor both sets of UDP ports for RADIUS requests.*

Verifications:

----

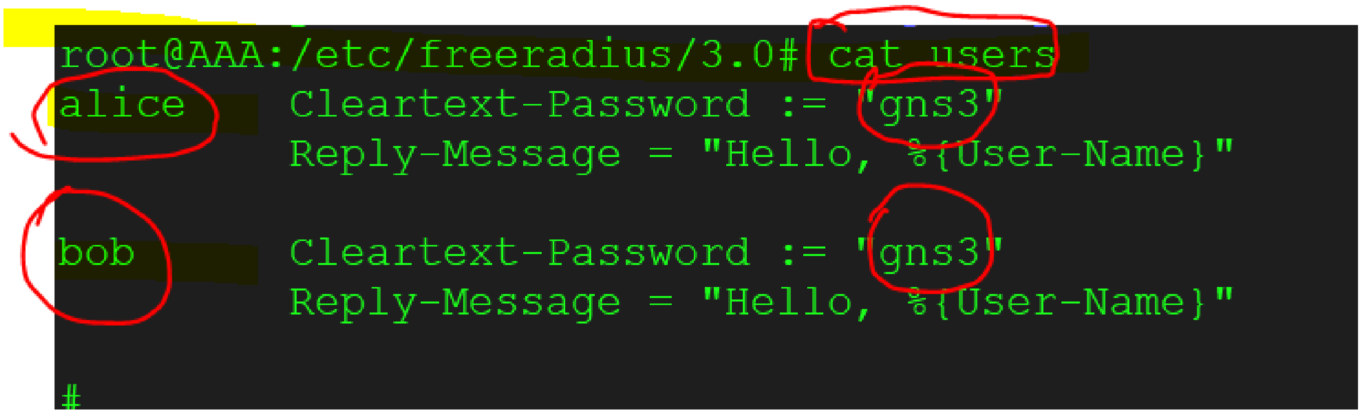
debug aaa authentication
debug radius

Do a self **telnet 192.168.29.102**
Username: bob or alice
Password: gns3

Show users
show line

root@AAA-1:~# cat /etc/freeradius/3.0/users

service freeradius start/stop



```
root@AAA:/etc/freeradius/3.0# cat users
alice      Cleartext-Password := "gns3"
           Reply-Message = "Hello, %{User-Name}"

bob        Cleartext-Password := "gns3"
           Reply-Message = "Hello, %{User-Name}"

#
```

ACCESS-LIST [ACL]

Defining an ACL

An ACL is a router configuration script (a list of statements) that controls whether a router permits or denies packets to pass, based on criteria in the packet header. To determine whether a packet is permitted or denied, it is tested against the ACL statements in sequential order. When a statement matches, no more statements are evaluated; the packet is either permitted or denied. There is an implicit deny any statement at the end of the ACL. If a packet does not match any of the statements in the ACL, it is dropped.

Types of ACLs

ACLs can be configured to filter any type of protocol traffic, including other network layer protocols such as AppleTalk and IPX. For the CCNA exam, we focus on IPv4 and IPv6 ACLs, which come in the following types:

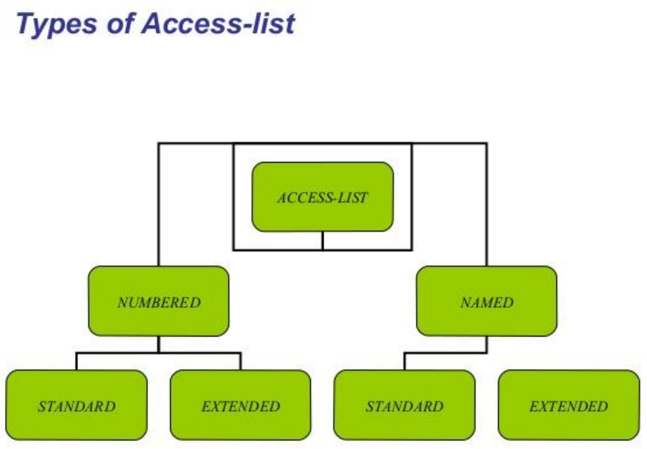
1. **Standard IPv4 ACLs: Filter traffic based on source address only (all services are blocked)**
2. **Extended IPv4 and IPv6 ACLs: Can filter traffic based on source and destination address, specific protocols, and source and destination TCP and UDP ports**

Two ways to Identify Standard or Extended ACL:

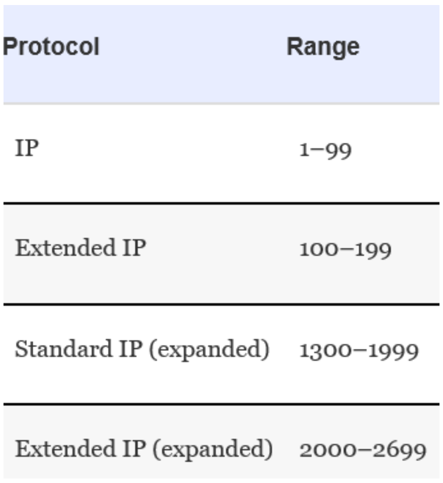
1. Numbered IPv4 ACLs: Use a number for identification
2. Named IPv4 and IPv6 ACLs: Use a descriptive name or number for identification

Named ACLs must be used with some types of Cisco IOS configurations, including IPv6 ACLs. However, they provide two basic benefits for standard and extended IPv4 ACLs:

- By using a descriptive name (such as BLOCK-HTTP), a network administrator can more quickly determine the purpose of an ACL. This is particularly helpful in larger networks, where a router can have many ACLs with hundreds of statements.
- Both numbered and named ACLs can be configured for standard as well as extended ACL implementations.



ACL IDENTIFICATIONS:



Protocol	Range
IP	1–99
Extended IP	100–199
Standard IP (expanded)	1300–1999
Extended IP (expanded)	2000–2699

Cisco IOS Software Release 12.3 introduced IP access list entry sequence numbering for both numbered and named ACLs. IP access list entry sequence numbering provides the following benefits:

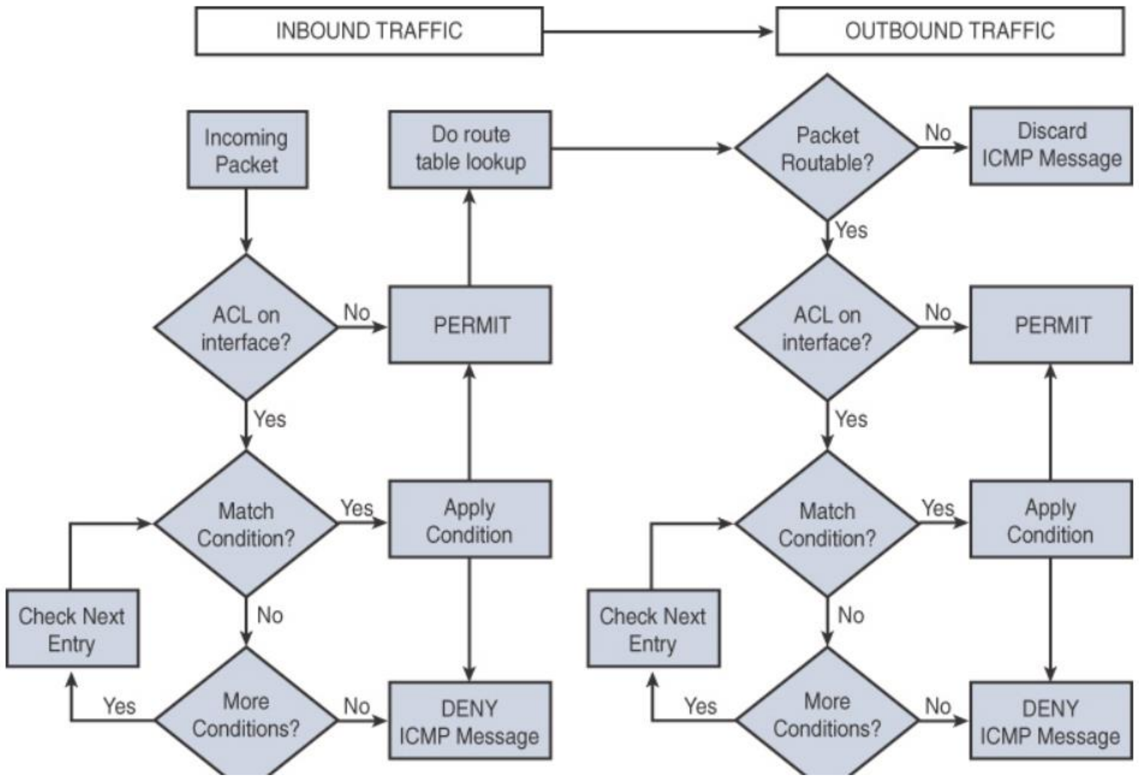
- **You can edit the order of ACL statements.**
- **You can remove individual statements from an ACL.**
- **You can use the sequence number to insert new statements into the middle of the ACL.**
- **Sequence numbers are automatically added to the ACL if they are not entered explicitly at the time the ACL is created.**

Access List Rules

The following rules apply to access lists:

1. Only one access list per interface, per protocol, and per direction is allowed.
2. An access list must contain at least one **permit** statement or all packets are denied entry into the network.
3. After a match is found, no more criteria statements are checked.
4. If an access list is referenced by a name, but the access list does not exist, all packets pass.
5. An interface or command with an empty access list applied to it permits all traffic into the network.
6. Standard access lists and extended access lists cannot have the same name.
7. Standard Access Control List should be placed near the destination devices.
   a.
8. Extended Access Control List (ACL) should be placed near the source devices.
9. Inbound access lists process packets before the packets are routed to an outbound interface.
   Inbound access lists that have filtering criteria that deny packet access to a network saves the **overhead of routing lookup**.

10. Outbound access lists process packets before they leave the device. Incoming packets are routed to the outbound interface and then processed by the outbound access list. For outbound access lists, when you configure a **permit** statement, packets are sent to the **output buffer**, and when you configure a **deny** statement, packets are discarded.
11. An access list can control traffic arriving at a device or leaving a device, but not traffic originating at a device.



TYPES OF ACLS:

1. Standard ACLs

Standard ACLs are the oldest type of ACL. They date back to as early as Cisco IOS Software Release 8.3. Standard ACLs control traffic by the comparison of the source address of the IP packets to the addresses configured in the ACL.

access-list *access-list-number* **{permit|deny}** {*host*|*source source-wildcard*|**any**}

NUMBERED STANDARD ACL:

```
interface Ethernet0/0
 ip address 10.1.1.1 255.255.255.0
 ip access-group 1 in
 exit
```

```
access-list 1 permit 10.1.1.0 0.0.0.255
```

NAMED STANDARD ACL:

```
R1(config)# ip access-list standard  blockacl
R1(config-std-nacl)# deny 172.16.40.0 0.0.0.255
R1(config-std-nacl)# permit  any
```

```
R1(config)# int fa0/1
R1(config-if)# ip access-group blockacl out
```

2. Extended Access Lists

Extended access lists are good for blocking traffic anywhere. Extended access lists test source and destination addresses and other IP packet data, such as protocols, TCP or UDP port numbers, type of service (ToS), precedence, TCP flags, and IP options. Extended access lists can also provide capabilities that standard access lists cannot, such as the following:

- Filtering IP Options
- Filtering TCP flags
- Filtering noninitial fragments of packets
- Time-based entries

NAMED EXTENDED ACL:

```
R1(config)# ip access-list extended blockacl
R1(config-ext-nacl)# deny tcp 172.16.40.0 0.0.0.255 172.16.50.0 0.0.0.255 eq 21
R1(config-ext-nacl)# deny tcp any 172.16.50.0 0.0.0.255 eq 23
R1(config-ext-nacl)# permit ip any any
```

```
R1(config)# int fa0/1
R1(config-if)# ip access-group blockacl out
```

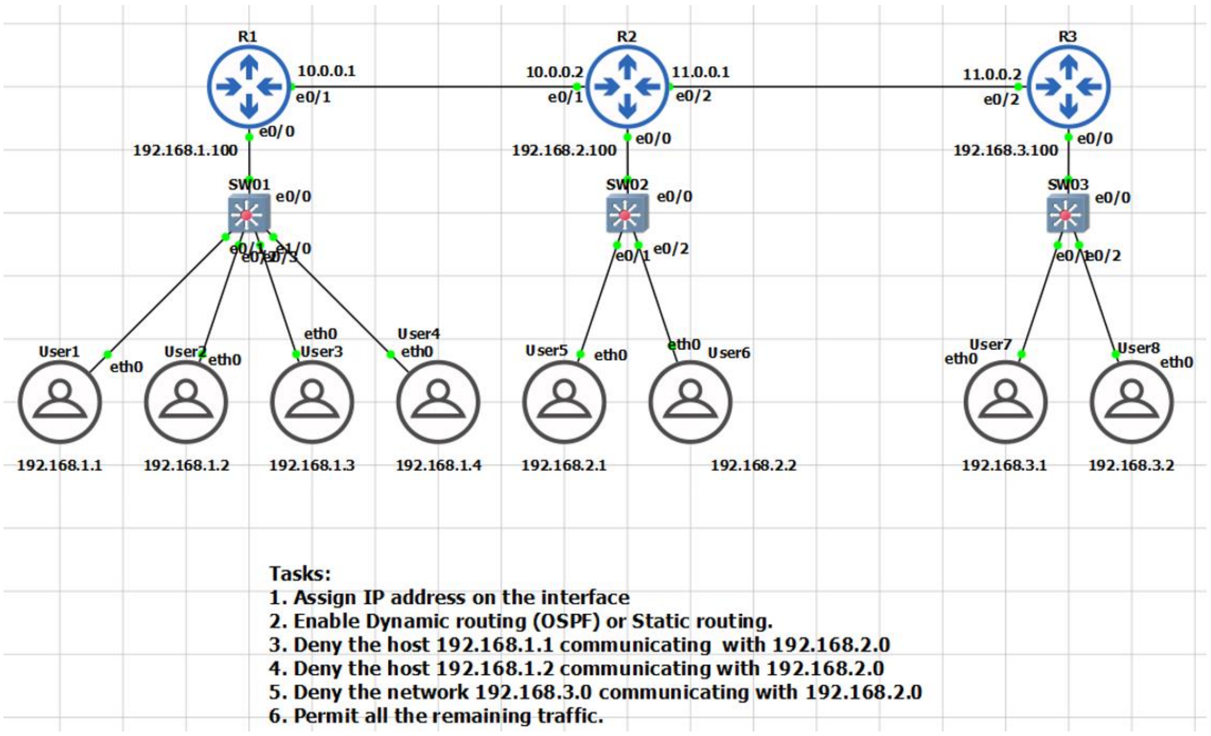
NUMBERED EXTENDED ACL:

```
R1(config)# access-list 110
        deny tcp 172.16.40.0 0.0.0.255 172.16.50.0 0.0.0.255 eq 21
R1(config)# access-list 110 permit ip any any
```

```
R1(config)# int fa0/1
R1(config-if)# ip access-group 110 out
```

HOMEWORK ASSIGNMENT

STANDARD ACL (NUMBER & NAMED)



Tasks:
1. Assign IP address on the interface
2. Enable Dynamic routing (OSPF) or Static routing.
3. Deny the host 192.168.1.1 communicating with 192.168.2.0
4. Deny the host 192.168.1.2 communicating with 192.168.2.0
5. Deny the network 192.168.3.0 communicating with 192.168.2.0
6. Permit all the remaining traffic.

ANSWERS

1& 2:

```
R1
hostname R1
interface Ethernet0/0
 ip address 192.168.1.100 255.255.255.0
no shut
interface Ethernet0/1
 ip address 10.0.0.1 255.255.255.0
no shut

router ospf 1
 router-id 1.1.1.1
 network 10.0.0.0 0.0.0.255 area 0
 network 192.168.1.0 0.0.0.255 area 0

R2
hostname R2
interface Ethernet0/1
 ip address 10.0.0.2 255.255.255.0
no shut
interface Ethernet0/0
 ip address 192.168.2.100 255.255.255.0
```

```
no shut
interface Ethernet0/2
 ip address 11.0.0.1 255.255.255.0
no shut

router ospf 1
router-id 2.2.2.2
 network 10.0.0.0 0.0.0.255 area 0
 network 11.0.0.0 0.0.0.255 area 0
 network 192.168.2.0 0.0.0.255 area 0
```

R3

```
hostname R3
interface Ethernet0/2
 ip address 11.0.0.2 255.255.255.0
no shut
interface Ethernet0/0
 ip address 192.168.3.100 255.255.255.0
no shut

router ospf 1
 router-id 3.3.3.3
 network 11.0.0.0 0.0.0.255 area 0
 network 192.168.3.0 0.0.0.255 area 0
```

3 & 4 & 5 & 6

R2

```
access-list 15 deny   192.168.1.1
access-list 15 deny   192.168.1.2
access-list 15 deny   192.168.3.0 0.0.0.255
access-list 15 permit any

int e0/0 (config)#
ip access-group 15 out
```

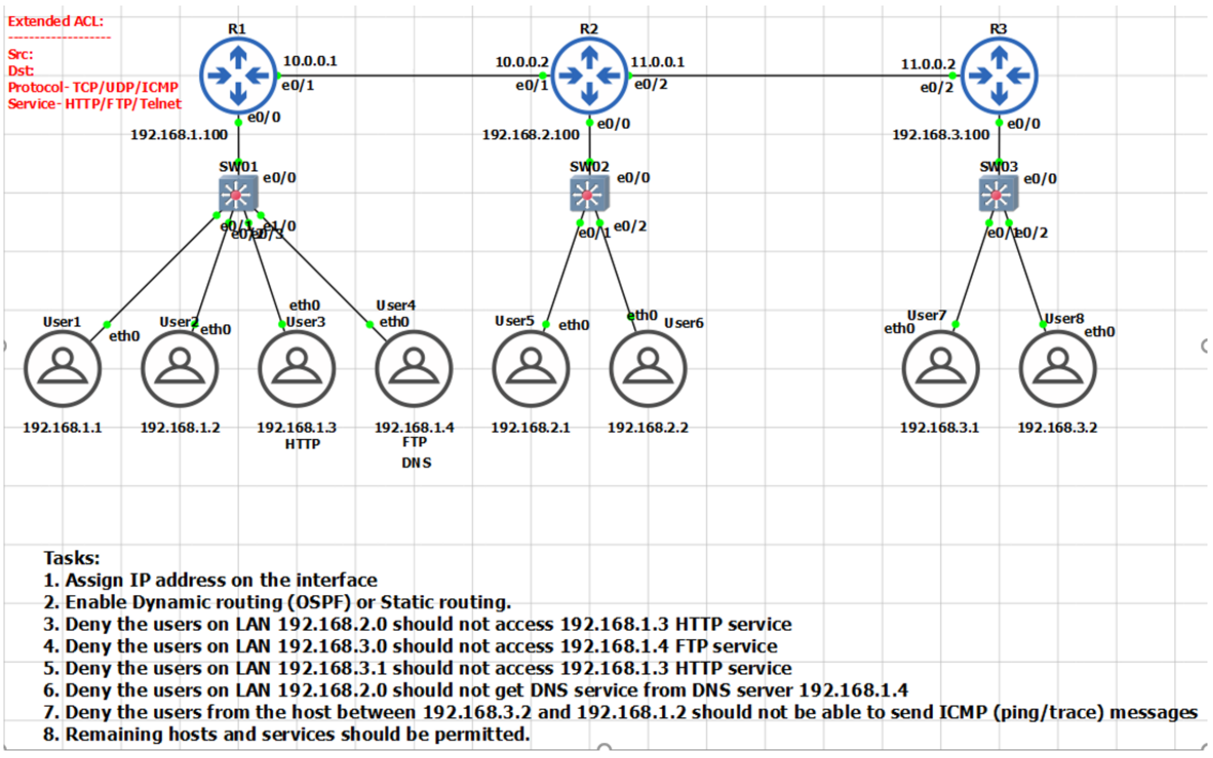
VALIDATIONS:

Ping and Validate.

Ping should not be successful!!!

HOMEWORK ASSIGNMENT #2

EXTENDED ACL (NUMBER & NAMED)



Tasks:
1. Assign IP address on the interface
2. Enable Dynamic routing (OSPF) or Static routing.
3. Deny the users on LAN 192.168.2.0 should not access 192.168.1.3 HTTP service
4. Deny the users on LAN 192.168.3.0 should not access 192.168.1.4 FTP service
5. Deny the users on LAN 192.168.3.1 should not access 192.168.1.3 HTTP service
6. Deny the users on LAN 192.168.2.0 should not get DNS service from DNS server 192.168.1.4
7. Deny the users from the host between 192.168.3.2 and 192.168.1.2 should not be able to send ICMP (ping/trace) messages
8. Remaining hosts and services should be permitted.

ANSWERS

1& 2:

Assign IP address.

3 & 4 & 5 & 6 & 7 & 8

R1

```
access-list 145 deny   tcp 192.168.2.0 0.0.0.255 host 192.168.1.3 eq www
access-list 145 deny   tcp 192.168.3.0 0.0.0.255 host 192.168.1.4 eq ftp
access-list 145 deny   tcp host 192.168.3.1 host 192.168.1.3 eq www
access-list 145 deny   udp 192.168.2.0 0.0.0.255 host 192.168.1.4 eq domain
access-list 145 deny   icmp host 192.168.3.2 host 192.168.1.2 echo
access-list 145 deny   icmp host 192.168.3.2 host 192.168.1.2 echo-reply
access-list 145 permit ip any any

int e0/0 (config)#
ip access-group 145 out
```

VALIDATIONS:

Ping and Validate.

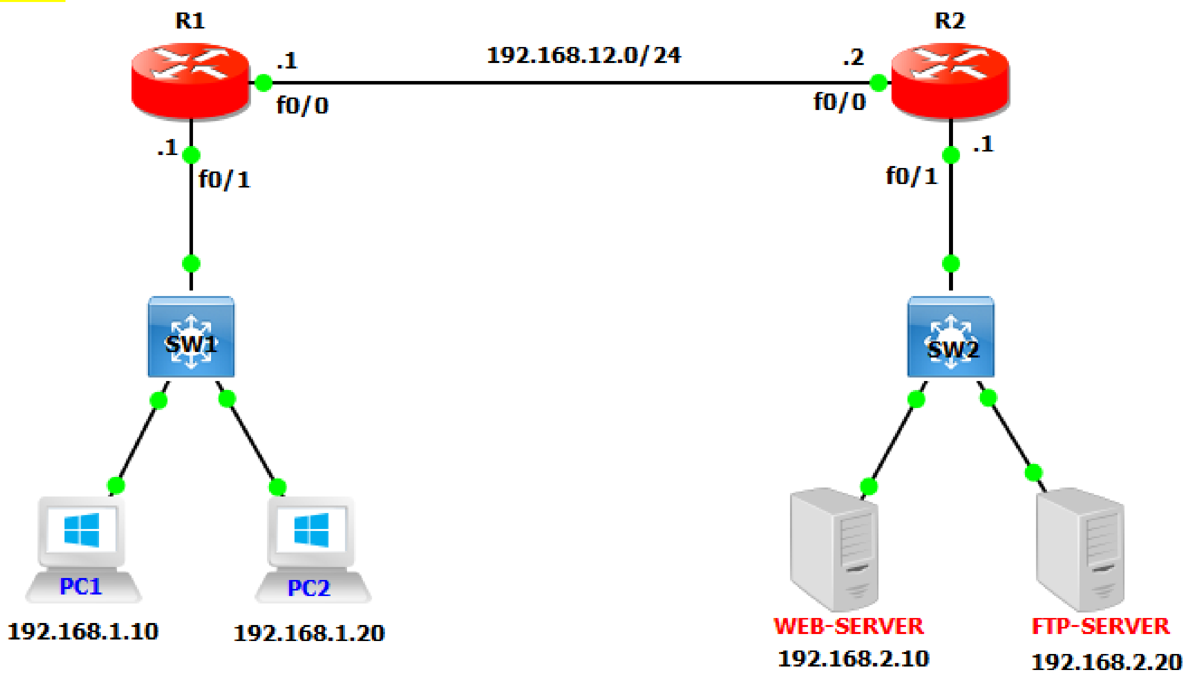
Ping should not be successful!!!

Time-Based ACL:

- Time-based ACLs that allow for network access based on time or day.
- Time-based ACLs are only active during a specified time range.
- For TACL set the time range to be either periodic or absolute.
- Configure time-based ACL, specify time range & then apply ACL.
- To verify, run show access-lists and check if the ACL is active.
- If it is in the time range and the ACL is filtering traffic.

HOMEWORK ASSIGNMENT #3

GNS3



```
R1(config)# time-range TEST-TIME
R1(config-time-range)# periodic weekdays 12:00 to 23:00
R1(config-time-range)# exit

R1(config)# access-list 100 permit ip 192.168.1.0 0.0.0.255 any time-range TEST-TIME
R1(config)# access-list 100 deny ip any any

R1(config)# interface FastEthernet 0/1
R1(config-if)# ip access-group 100 in
```

QUIZ#

1. What is the correct order of operations for an IPv4 ACL?

1. Top-down processing, execute upon the longest match, implicit deny all
2. Execute upon the longest match, top-down processing, implicit deny all

3. Implicit deny all, immediate execution upon a match, top-down processing
4. Top-down processing, immediate execution upon a match, implicit deny all

__2.__ What occurs to a packet when an ACL is applied to an interface but the packet does not match any of the entries in the ACL?

1. It is forwarded.
2. It is flooded.
3. It is dropped.
4. It is buffered.

__3.__ What does the following ACL entry accomplish when applied to an interface: **20 permit tcp 10.1.1.0 0.0.0.63 host 192.0.2.1 eq 23?**

1. Permits Telnet traffic from the device with IP address 192.0.2.1 going to any device with an IP address from 10.1.1.0 to 10.1.1.63
2. Permits Telnet traffic from any device with IP address from 10.1.1.0 to 10.1.1.63 going to the device with IP address 192.0.2.1
3. Permits SSH traffic from any device with IP address from 10.1.1.0 to 10.1.1.63 going to the device with IP address 192.0.2.1
4. Permits SSH traffic from the device with IP address 192.0.2.1 going to any device with IP address from 10.1.1.0 to 10.1.1.63

__4.__ Which command successfully filters ingress traffic using ACL 100 on an interface?

1. **access-group 100 in**
2. **access-class 100 in**
3. **ip access-group 100 in**
4. **ip traffic-filter 100 in**

__5.__ What is the correct order of operations for an IPv6 ACL?

1. Immediate execution upon a match, implicit permit icmp nd, implicit deny all, top-down processing
2. Top-down processing, immediate execution upon a match, implicit permit icmp nd, implicit deny all
3. Top-down processing, implicit permit icmp nd, immediate execution upon a match, implicit deny all
4. Implicit permit icmp nd, top-down processing, immediate execution upon a match, implicit deny all

__6.__ What happens if you add the following entry to the end of an IPv6 ACL: **deny ipv6 any any log**? (Choose two.)

1. All traffic is denied and logged.
2. All traffic that does not match an entry in the ACL is denied and logged.
3. ICMP Neighbor Discovery messages are implicitly permitted.
4. ICMP Neighbor Discovery messages are denied.

__7.__ Which command successfully filters egress traffic using an IPv6 ACL named ENARSI on an interface?

1. **access-group ENARSI out**
2. **access-class ENARSI out**
3. **ipv6 access-group ENARSI out**
4. **ipv6 traffic-filter ENARSI out**

8. Which IP prefix list matches only the default route?

1. **ip prefix-list ENARSI permit 0.0.0.0/0 le 32**
2. **ip prefix-list ENARSI permit 0.0.0.0/0 ge 32**
3. **ip prefix-list ENARSI permit 0.0.0.0/0 ge 1**
4. **ip prefix-list ENARSI permit 0.0.0.0/0**

9. Which IP prefix list matches all routes?

1. **ip prefix-list ENARSI permit 0.0.0.0/0 le 32**
2. **ip prefix-list ENARSI permit 0.0.0.0/0 ge 32**
3. **ip prefix-list ENARSI permit 0.0.0.0/0 ge 1**
4. **ip prefix-list ENARSI permit 0.0.0.0/0**

10. What routes match the following prefix list: **ip prefix-list ENARSI seq 35 deny 192.168.0.0/20 ge 24 le 28**?

1. Routes with an address from 192.168.0.0 to 192.168.15.255 with a subnet mask of 24 to 28
2. Routes within the 192.168.0.0/20 subnet with a subnet mask greater than 24 and less than 28
3. Routes with the subnet ID and mask 192.168.0.0/20
4. Routes with an address from 192.168.0.0 to 192.168.15.255 with a subnet mask of 24 or 28

ANSWERS

1. d

2. c

3. b

4. c

5. b

6. b and d

7. d

8. d

9. a

10. a

SECURITY APPLIANCE - ASA

Cisco PIX: Start: early 1994 | End of sale: 28 July 2008
CISCO ASA: Start: May 2005
CISCO FIREPOWER: Start July 2013
PALO ALTO: Start 2007
CHECKPOINT: Start 1996
FORITGATE: Start 2001
SOPHOS: Start 1990
PFSENSE: Start 2006

CISCO PIX

- End of life
- Older variant of commands

FIRST GEN ASA MODELS

- Successor to Cisco Pix
- Packet filtering firewalls
  - 5505
  - 5510
  - 5520
  - ……

→ 5585 { highest }

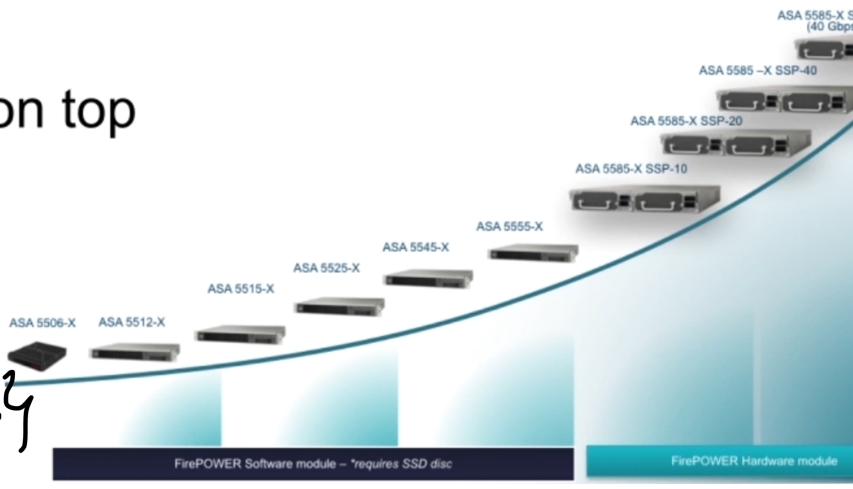
Model	5505[17]	5510	5520[17]	5540[17]	5550[17]	5580-20[17]	5580-40[17]	5585-X SSP10[17]	5585-X SSP20[17]	5585-X SSP40[17]	5585-X SSP60[17]
Cleartext throughput, Mbit/s	150	300	450	650	1,200	5,000	10,000	3,000	7,000	12,000	20,000
AES/Triple DES throughput, Mbit/s	100	170	225	325	425	1,000	1,000	1,000	2,000	3,000	5,000
Max simultaneous connections	10,000 (25,000 with Sec Plus License)	50,000 (130,000 with Sec Plus License)	280,000	400,000	650,000	1,000,000	2,000,000	1,000,000	2,000,000	4,000,000	10,000,000
Max site-to-site and remote access VPN sessions	10 (25 with Sec Plus License)	250	750	5,000	5,000	10,000	10,000	5,000	10,000	10,000	10,000
Max number of SSL VPN user sessions	25	250	750	2,500	5,000	10,000	10,000	5,000	10,000	10,000	10,000
Model	5505	5510	5520	5540	5550	5580-20	5580-40	5585-X SSP10	5585-X SSP20	5585-X SSP40	5585-X SSP60

Model	5506-X	5506W-X	5506H-X	5508-X	5512-X	5515-X	5516-X	5525-X	5545-X	5555-X	5585-X
Throughput	0.25	0.25	0.25	0.45	0.3	0.5	0.85	1.1	1.5	1.75	4-40
GB ports	8	8	4	8	6	6	8	8	8	8	6-8
Ten GB ports	0	0	0	0	0	0	0	0	0	0	2-4
Form factor	desktop	desktop	desktop	1 RU	1 RU	1 RU	1 RU	1RU	1RU	1RU	2RU

ASA-X MODELS

- "Next Gen" Firewalls
- Traditional ASA with FirePower services on top
  - Application visibility
  - URL filtering
  - IPS
  - AMP

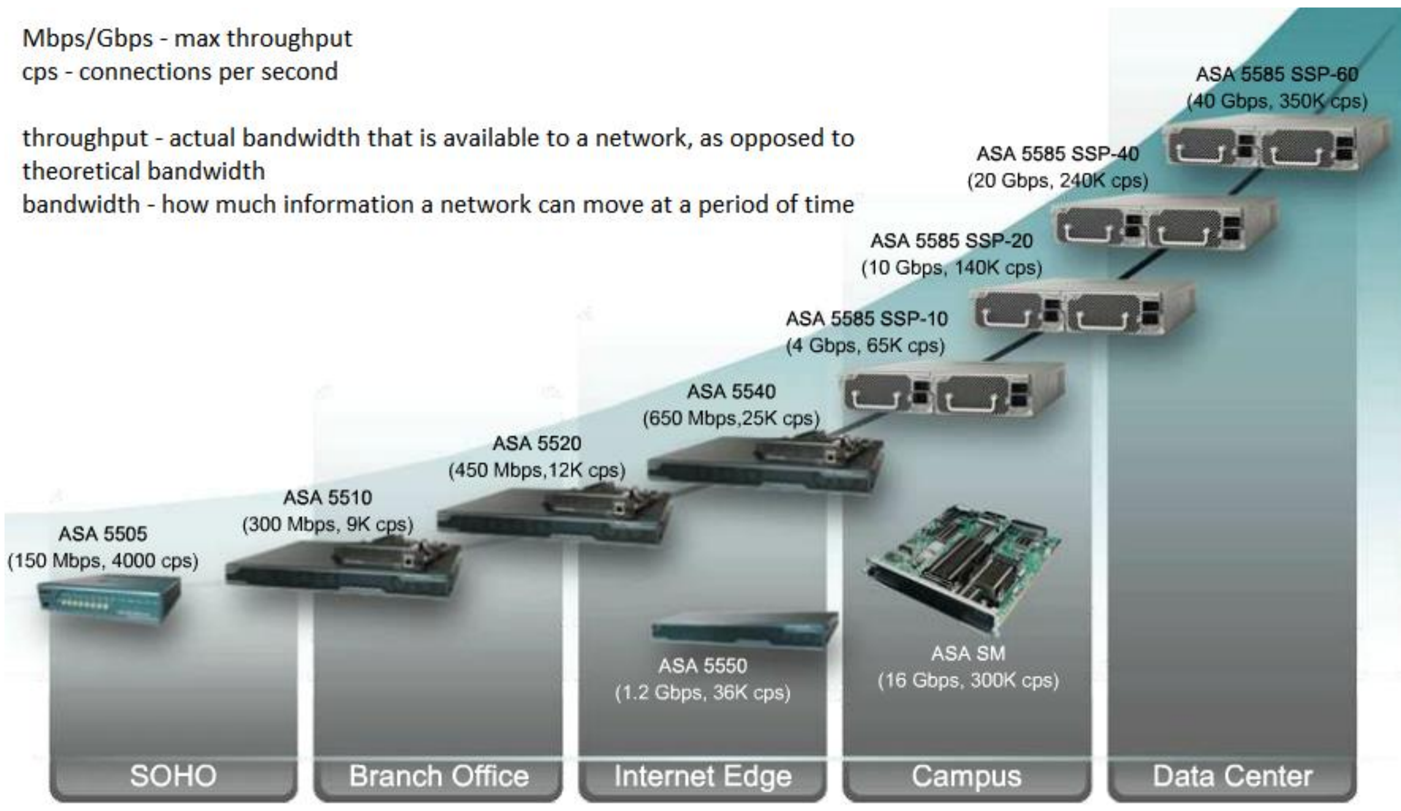
{advanced Malware Protection}



ASA Firepower Products - https://www.cisco.com/c/en/us/products/security/asa-firepower-services/index.html?dtid=osscdc000283#~software

Mbps/Gbps - max throughput
cps - connections per second

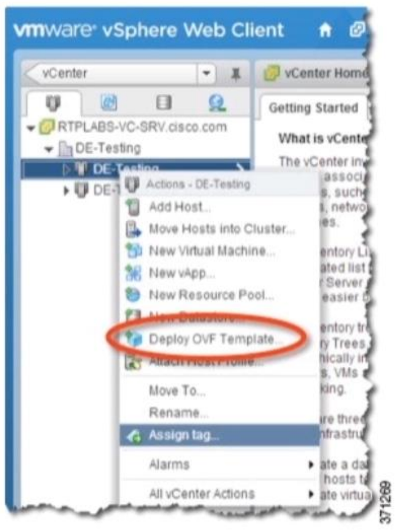
throughput - actual bandwidth that is available to a network, as opposed to theoretical bandwidth
bandwidth - how much information a network can move at a period of time



All ASA (only) Products discussion comparison - http://nhprice.com/cisco-asa-5500-specs-features-and-model-comparisons.html

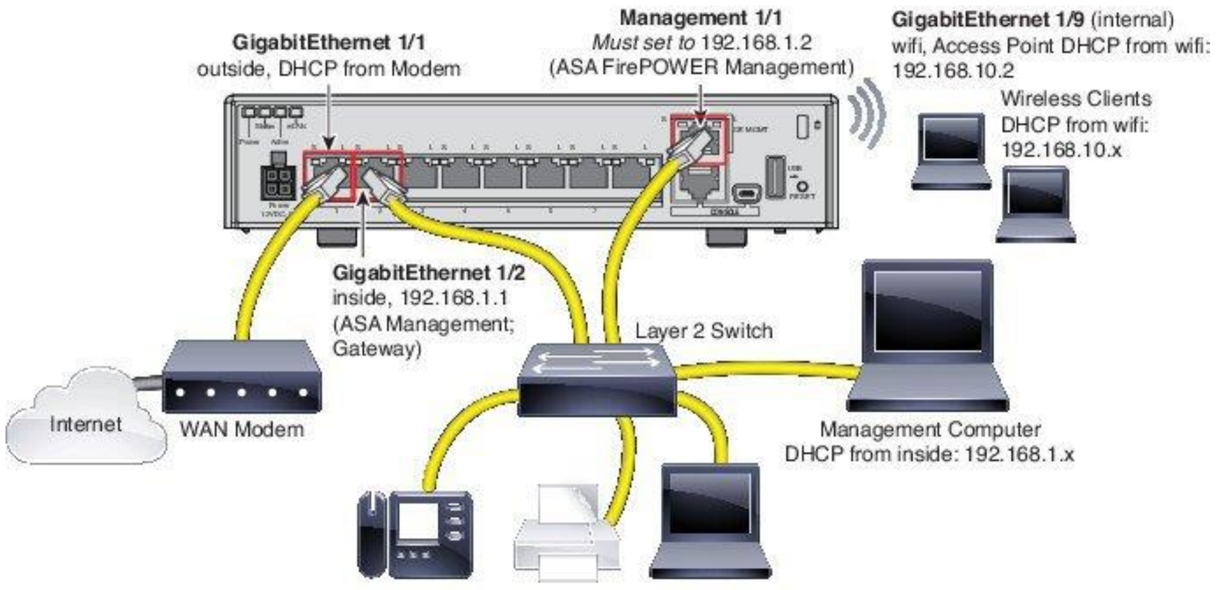
ASA-V

- Virtual
- Full ASA feature set
- No FirePower



ASA-SM

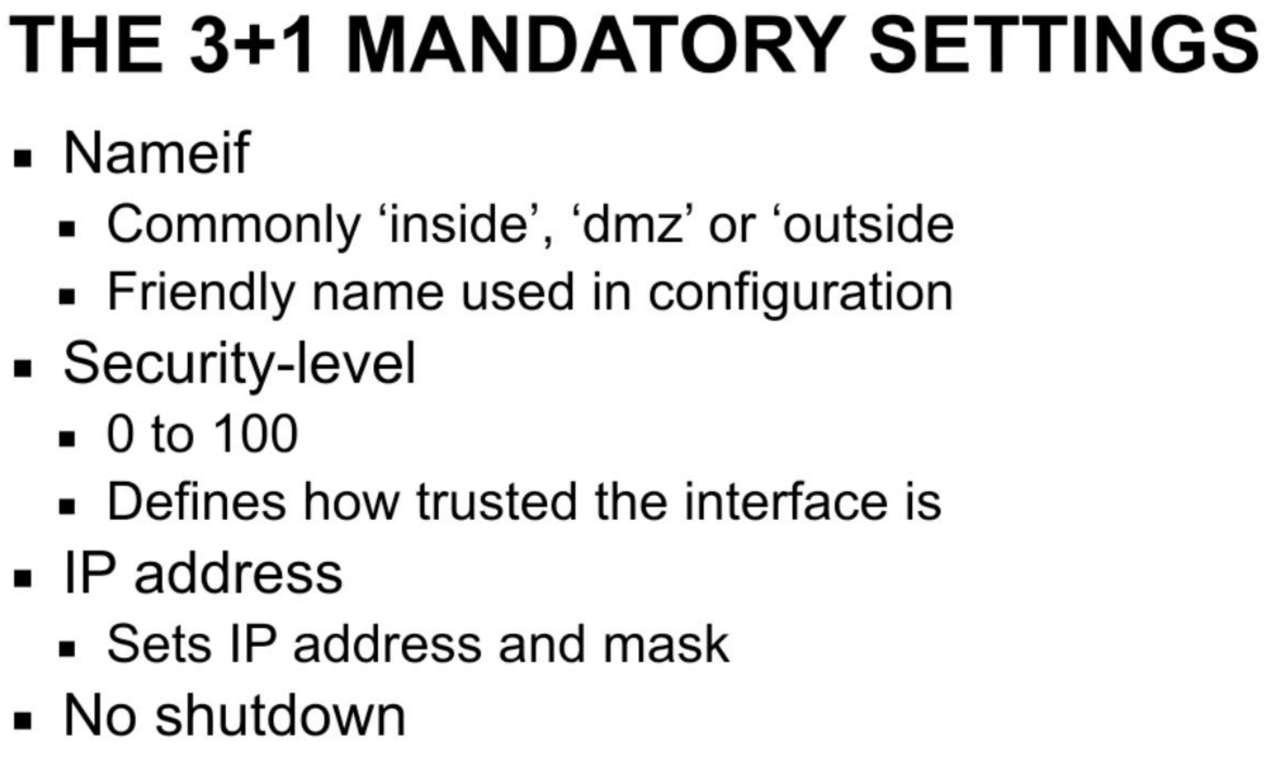
- Service Module in c6500
- Full ASA feature set
- No FirePower



Security Levels

This lesson describes the security levels concept as used in the ASA firewall appliance.

- A Security Level is assigned to interfaces (either physical or logical sub-interfaces)
- It is basically a number from 0 to 100 designating how trusted an interface is relative to another interface on the appliance.
- The higher the security level, the more trusted the interface (and hence the network connected behind it) is considered to be, relative to another interface.
- Since each firewall interface represents a specific network (or security zone), by using security levels we can assign 'trust levels' to our security zones.
- The primary rule for security levels is that an interface (or zone) with a **higher security level can access an interface with a lower security level.**
- **On the other hand, an interface with a lower security level cannot access an interface with a higher security level, without the explicit permission of a security rule (Access Control List - ACL).**



THE 3+1 MANDATORY SETTINGS

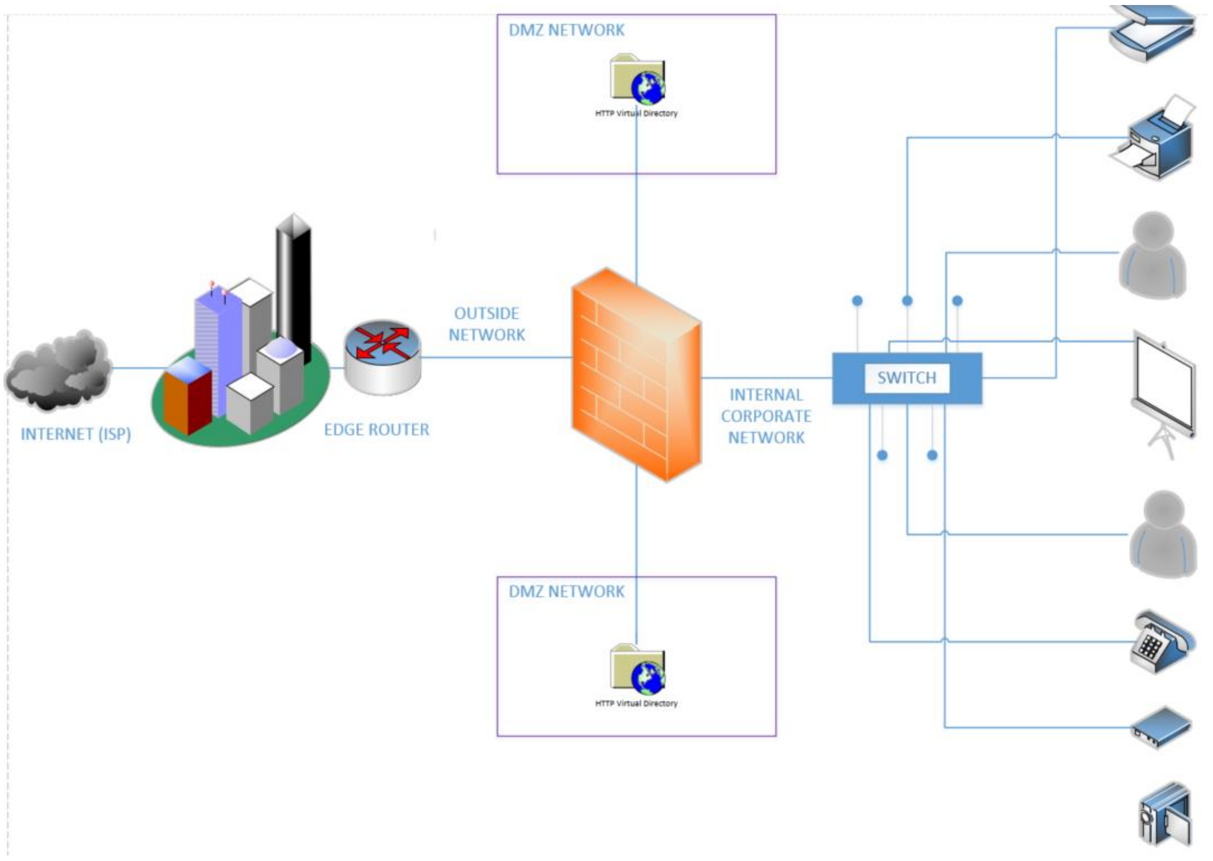
- Nameif
  - Commonly 'inside', 'dmz' or 'outside
  - Friendly name used in configuration
- Security-level
  - 0 to 100
  - Defines how trusted the interface is
- IP address
  - Sets IP address and mask
- No shutdown

Security Level Examples

- **Security Level 0**: **This is the lowest security level and it is assigned by default to the '**Outside**' Interface of the firewall. It is the least trusted security level and must be assigned accordingly to the network (interface) that we don't want it to have any access to our internal networks. This security level is usually assigned to the interface connected to the Internet. This means that every device connected to the Internet can not have access to any network behind the firewall, unless explicitly permitted by an ACL rule.**

- **Security Levels 1 to 99**: **These security levels can be assigned to perimeter security zones (e.g. DMZ Zone, Management Zone, Database Servers Zone etc).**

- **Security Level 100**: **This is the highest security level and it is assigned by default to the '**Inside**' Interface of the firewall. It is the most trusted security level and must be assigned accordingly to the network (interface) that we want to apply the most protection from the security appliance. This security level is usually assigned to the interface connecting the Internal Corporate network behind it**



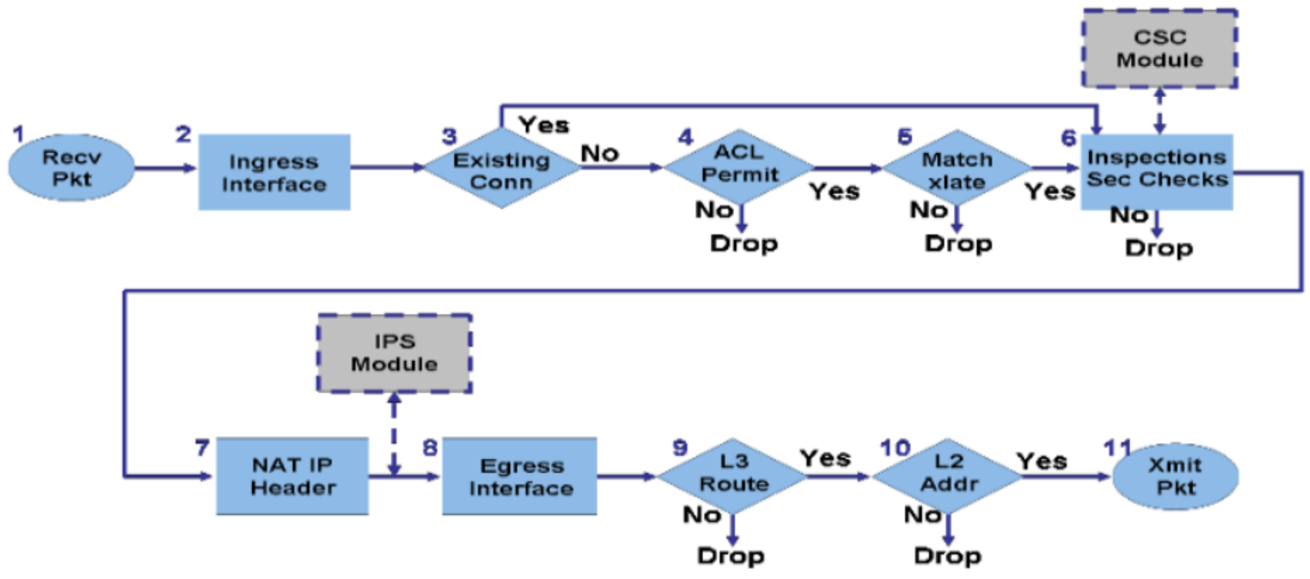
Rules for Traffic Flow between Security Levels

• Traffic from Higher Security Level to Lower Security Level**: Allow ALL traffic originating from the higher Security Level unless specifically restricted by an Access Control List (ACL). If NAT-Control is enabled on the device, then there must be a** nat/global **translation pair between High-to-Low Security Level interfaces.** NOTE: "global**" command is not supported in ASA versions 8.3 and later (more on this later).**

• Traffic from Lower Security Level to Higher Security Level: **Drop ALL traffic unless specifically allowed by an ACL. If NAT-Control is enabled on the device (more on this later), then there must be a** Static NAT **between High-to-Low Security Level interfaces.**

• Traffic between interfaces with same Security Level: **By default, this is not allowed, unless you configure the** same-security-traffic permit inter-interface **command (ASA version 7.2 and later).**

PACKET FLOW IN CISCO ASA FIREWALL:



1. The packet is reached at the ingress interface.
2. Once the packet reaches the internal buffer of the interface, the input counter of the interface is incremented by one.
3. Cisco ASA first looks at its internal connection table details in order to verify if this is a current connection. If the packet flow matches a current connection, then the Access Control List (ACL) check is bypassed and the packet is moved forward.

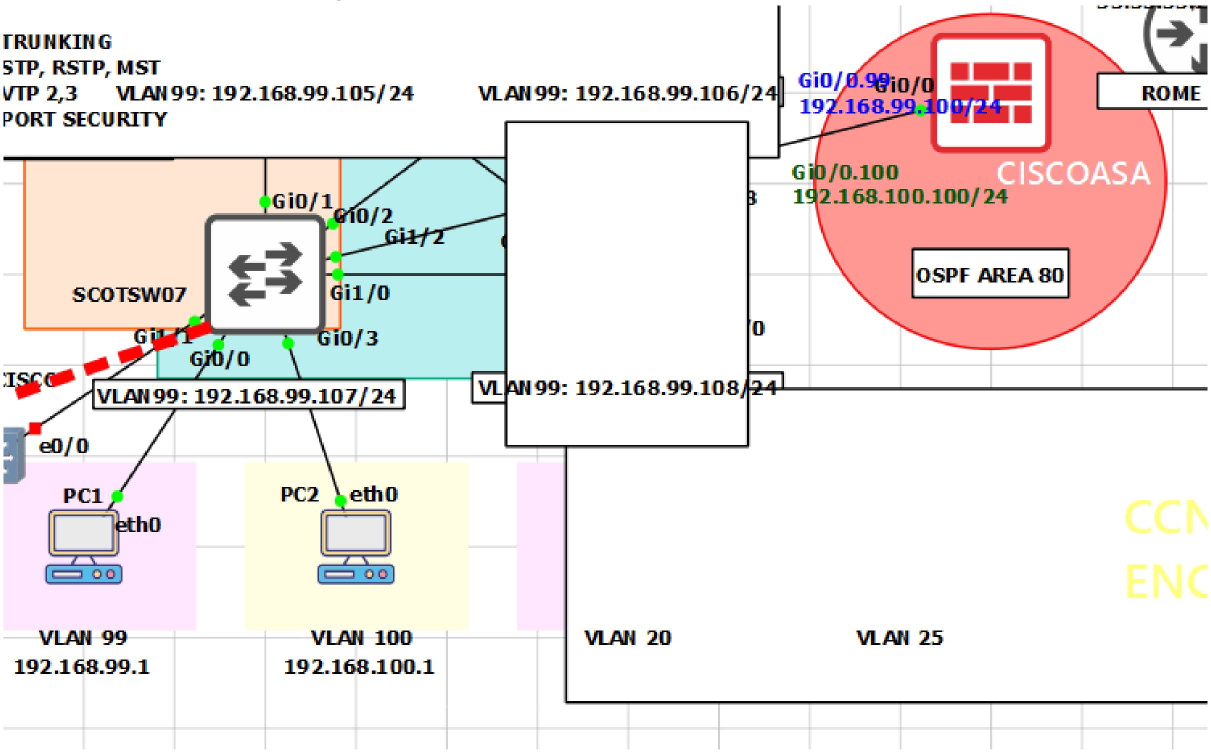
   If packet flow does not match a current connection, then the TCP state is verified. If it is a SYN packet or UDP (User Datagram Protocol) packet, then the connection counter is incremented by one and the packet is sent for an ACL check. If it is not a SYN packet, the packet is dropped and the event is logged.

4. The packet is processed as per the interface ACLs. It is verified in sequential order of the ACL entries and if it matches any of the ACL entries, it moves forward. Otherwise, the packet is dropped and the information is logged. The ACL hit count is incremented by one when the packet matches the ACL entry.
5. The packet is verified for the translation rules. If a packet passes through this check, then a connection entry is created for this flow and the packet moves forward. Otherwise, the packet is dropped and the information is logged.
6. The packet is subjected to an Inspection Check. This inspection verifies whether or not this specific packet flow is in compliance with the protocol. Cisco ASA has a built-in inspection engine that inspects each connection as per its pre-defined set of application-level functionality. If it passed the inspection, it is moved forward. Otherwise, the packet is dropped and the information is logged.

   Additional security checks will be implemented if a Content Security (CSC) module is involved.
7. The IP header information is translated as per the Network Address Translation/ Port Address Translation (NAT/PAT) rule and checksums are updated accordingly. The packet is forwarded to Advanced Inspection and Prevention Security Services Module (AIP-SSM) for IPS related security checks when the AIP module is involved.
8. The packet is forwarded to the egress interface based on the translation rules. If no egress interface is specified in the translation rule, then the destination interface is decided based on the global route lookup.
9. On the egress interface, the interface route lookup is performed. Remember, the egress interface is determined by the translation rule that takes the priority.
10. Once a Layer 3 route has been found and the next hop identified, Layer 2 resolution is performed. The Layer 2 rewrite of the MAC header happens at this stage.
11. The packet is transmitted on the wire, and interface counters increment on the egress interface.

GNS3

Cisco ASA INTERFACE Configs



CISCO ASA(config#)

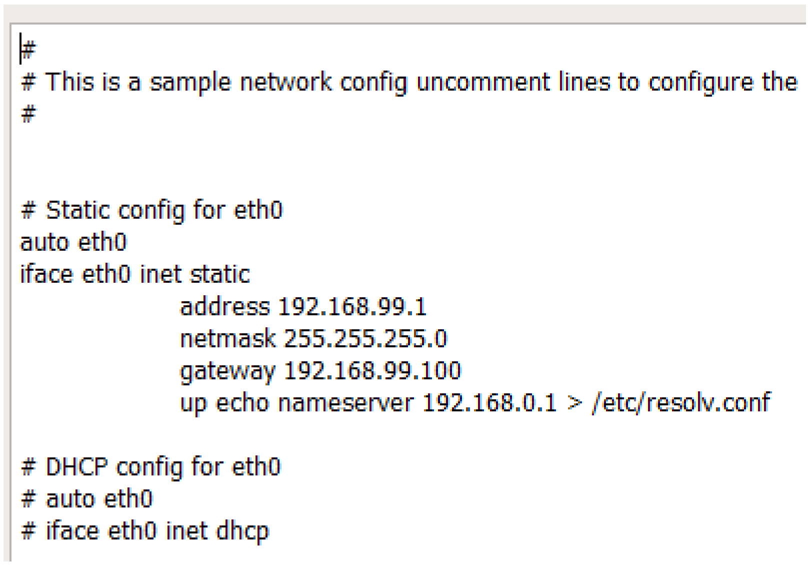
```
Interface gi0/0
no shutdown

interface GigabitEthernet0/0.99
 vlan 99
 nameif VLAN99
 security-level 100
 ip address 192.168.99.100 255.255.255.0

interface GigabitEthernet0/0.100
 vlan 100
 nameif VLAN100
 security-level 100
 ip address 192.168.100.100 255.255.255.0

policy-map global_policy
class inspection_default
inspect icmp

same-security-traffic permit inter-interface
```

SCOTSW07

```
interface GigabitEthernet1/2
```

```
switchport trunk encapsulation dot1q
switchport trunk allowed vlan 99,100
switchport mode trunk
no shutdown

interface GigabitEthernet0/0
 switchport access vlan 99
 switchport mode access
no shutdown

interface GigabitEthernet0/3
 switchport access vlan 100
 switchport mode access
no shutdown
```

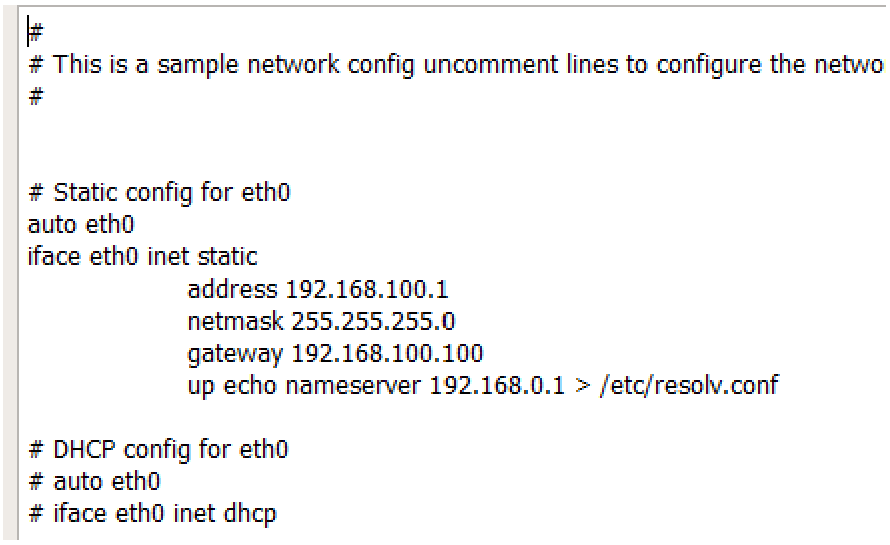
PC1:



```
#
# This is a sample network config uncomment lines to configure the
#


# Static config for eth0
auto eth0
iface eth0 inet static
                address 192.168.99.1
                netmask 255.255.255.0
                gateway 192.168.99.100
                up echo nameserver 192.168.0.1 > /etc/resolv.conf

# DHCP config for eth0
# auto eth0
# iface eth0 inet dhcp
```

```
 ip route add 0.0.0.0/8 via 192.168.99.100 dev eth0
```

PC2:



```
#
# This is a sample network config uncomment lines to configure the netwo
#


# Static config for eth0
auto eth0
iface eth0 inet static
                address 192.168.100.1
                netmask 255.255.255.0
                gateway 192.168.100.100
                up echo nameserver 192.168.0.1 > /etc/resolv.conf

# DHCP config for eth0
# auto eth0
# iface eth0 inet dhcp
```

```
ip route add 0.0.0.0/8 via 192.168.100.100 dev eth0
```

>> VALIDATION

PING PC1 to PC2 and VICE VERSA, it must be successful

TASK#2: CHANGE THE SECURITY-LEVEL

```
ciscoasa(config)#
 int gi0/0.99
security-level 50
```

Write ACL:

```
access-list VLAN99_in line 10 extended deny icmp any any log

access-group VLAN99_in in interface VLAN99

access-list VLAN99_in line 1 extended permit icmp 192.168.99.1 255.255.255.255 192.168.100.1
255.255.255.255 echo
```

VALDIATION:

```
PING 192.168.100.1 (192.168.100.1): 56 data bytes

64 bytes from 192.168.100.1: seq=192 ttl=64 time=16.626 ms

64 bytes from 192.168.100.1: seq=193 ttl=64 time=9.942 ms
```

```
64 bytes from 192.168.100.1: seq=194 ttl=64 time=40.217 ms
```

TROUBLESHOOTING

Packet-tracer

```
packet-tracer input VLAN99 icmp 192.168.99.1 8 0 192.168.100.1
```

CAPTURE

```
capture TESTING interface VLAN99 real-time match icmp host 192.168.99.1 host 192.168.100.1
```

uRPF (Unicast Reverse Path Forwarding)

- uRPF stands for Unicast Reverse Path Forwarding.
- Router checks the destination address of a packet only before routing.
- Using uRPF, the router will also look at the source address.
- Using uRPF in order to prevent traffic from spoofed IP addresses.
- This is done by verifying whether the router can reach the source IP.
- If the source address is valid, it will be forwarded.
- If the source IP is not valid, the packet will be dropped.
- uRPF is a security feature that prevents spoofing attacks.

Normally when your router receives unicast IP packets it only cares about one thing:

- *What is the destination IP address of this IP packet so I can forward it?*

If the IP packet has to be routed it will check the routing table for the destination IP address, select the correct interface and it will be forwarded. Your router really doesn't care about source IP addresses as it's not important for forwarding decisions.

Because the router doesn't check the source IP address it is possible for attackers to spoof the source IP address and send packets that normally might have been dropped by the Firewall or an access-list.

uRPF is a security feature that prevents these spoofing attacks. Whenever your router receives an IP packet it will check if it has a **matching entry in the routing table for the source IP address**. If it doesn't match, the packet will be discarded.

uRPF has two modes:
1. Strict mode
2. Loose mode

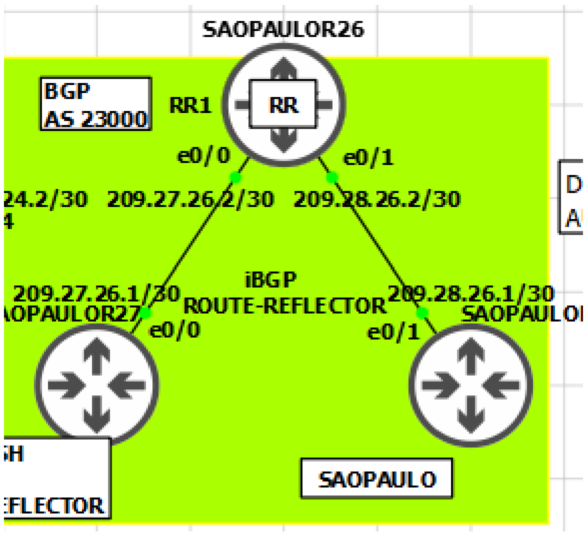
1. Strict Mode

Strict mode means that that router will perform **two checks** for all incoming packets on a certain interface:

- ➢ Do I have a matching entry for the source in the **routing table**?
- ➢ Do I use the **same interface to reach this source** as where I received this packet on?

When the incoming IP packets **passes both checks**, it will be permitted. Otherwise it will be dropped. This is perfectly fine for IGP routing protocols since they use the shortest path to the source of IP packets. The interface that you use to reach the source will be the same as the interface where you will receive the packets on.

GNS3 (strict)



CONFIG:

```
SAOPAULOR28(config)#
hostname SAOPAULOR28
interface Ethernet0/1
no shutdown
 ip address 209.28.26.1 255.255.255.0

interface Loopback1
 ip address 2.2.2.2 255.255.255.255
end

SAOPAULOR26-RR(config)#
hostname SAOPAULOR26-RR
interface Ethernet0/1
no shutdown
 ip address 209.28.26.2 255.255.255.252
 ip verify unicast source reachable-via rx
interface Ethernet0/0
no shutdown
 ip address 209.27.26.2 255.255.255.252
 ip verify unicast source reachable-via rx
```

ip route 2.2.2.2 255.255.255.255 Ethernet0/1

```
SAOPAULOR27(config)#
hostname SAOPAULOR27
interface Ethernet0/0
 ip address 209.27.26.1 255.255.255.0
no shutdown
interface Loopback1
 ip address 2.2.2.2 255.255.255.255
end
```

```
SAOPAULOR28#ping 209.28.26.2 so lo 1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 209.28.26.2, timeout is 2 seconds:
Packet sent with a source address of 2.2.2.2
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms

SAOPAULOR26-RR#show ip interface e0/0 | include verify
  IP verify source reachable-via RX

SAOPAULOR27#ping 209.27.26.2 so lo 1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 209.27.26.2, timeout is 2 seconds:
Packet sent with a source address of 2.2.2.2
.....
Success rate is 0 percent (0/5)


SAOPAULOR26-RR#show ip interface eth0/0 | include drops
   4 verification drops
   0 suppressed verification drops
```

VERIFICATIONS:

```
show ip interface e 0/0 | include verify

debug ip packet
```

SAOPAULOR28#ping 209.28.26.2 source loopback1 → no drops

SAOPAULOR27(config-if)#do ping 209.27.26.2 so lo 1 →drops seen

```
SAOPAULOR26-RR(config)#do sh ip int e0/1 | i drop
   0 verification drops
   0 suppressed verification drops
   0 verification drop-rate

SAOPAULOR26-RR(config)#do sh ip int e0/0 | i drop
   8 verification drops
   0 suppressed verification drops
               2    verification drop-rate
```

2. Loose Mode

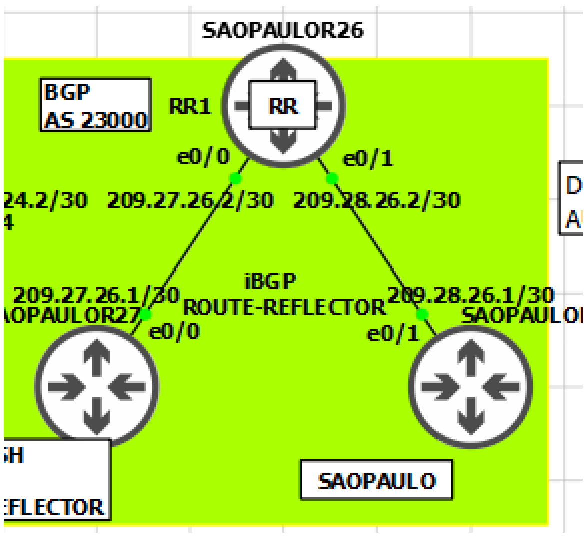
Loose mode means that the router will perform only a **single check** when it receives an IP packet on an interface:

Do I have a matching entry for the source in the **routing table**?

When it passed this check, the packet is permitted. It doesn't matter if we use this interface to reach the source or not. Loose mode is useful when you are connected to more than one ISP and you use **asymmetric routing**.

The only exception is the null0 interface, if you have any sources with the null0 interface as the outgoing interface then the packets will be dropped.

GNS3 (loose)



CONFIG:

```
SAOPAULOR28(config)#
hostname SAOPAULOR28
interface Ethernet0/1
no shutdown
 ip address 209.28.26.1 255.255.255.0

interface Loopback1
 ip address 2.2.2.2 255.255.255.255
end

SAOPAULOR27(config)#
hostname SAOPAULOR27
interface Ethernet0/0
 ip address 209.27.26.1 255.255.255.0
no shutdown
interface Loopback1
```

```
 ip address 2.2.2.2 255.255.255.255
end

SAOPAULOR26-RR(config)#
interface Ethernet0/0
no shutdown
 ip address 209.27.26.2 255.255.255.252
 ip verify unicast source reachable-via any

SAOPAULOR26-RR(config)#do sh run int e0/1
interface Ethernet0/1
no shut
 ip address 209.28.26.2 255.255.255.252
 ip verify unicast source reachable-via any

ip route 2.2.2.2 255.255.255.255 Ethernet0/1
```

VERIFICATIONS:

```
SAOPAULOR28#ping 209.28.26.2 so lo 1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 209.28.26.2, timeout is 2 seconds:
Packet sent with a source address of 2.2.2.2
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
SAOPAULOR28#

SAOPAULOR27#ping 209.27.26.2 so lo 1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 209.27.26.2, timeout is 2 seconds:
Packet sent with a source address of 2.2.2.2
.....
Success rate is 0 percent (0/5)

SAOPAULOR26-RR(config)#do sh ip int e0/0 | i drop
   0 verification drops
   9 suppressed verification drops
   0 verification drop-rate

debug ip packet
SAOPAULOR26-RR(config)#
*Nov  7 16:46:01.581: IP: tableid=0, s=2.2.2.2 (Ethernet0/0), d=209.27.26.2 (Ethernet0/0), routed via RIB
*Nov  7 16:46:01.581: IP: s=2.2.2.2 (Ethernet0/0), d=209.27.26.2 (Ethernet0/0), len 100, rcvd 3
*Nov  7 16:46:01.581: IP: s=2.2.2.2 (Ethernet0/0), d=209.27.26.2, len 100, stop process pak for forus packet
```

Loose method does not suppress until packets rcvd is from unknown sources:

```
SAOPAULOR27(config)#int lo 3
```

```
SAOPAULOR27(config-if)#ip add 3.3.3.3 255.255.255.255
SAOPAULOR27(config-if)#no shut

SAOPAULOR27#ping 209.27.26.2 so lo 3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 209.27.26.2, timeout is 2 seconds:
Packet sent with a source address of 3.3.3.3
.....
Success rate is 0 percent (0/5)

SAOPAULOR26-RR(config)#do sh ip int e0/0 | i drop
  4 verification drops
  15 suppressed verification drops
  0 verification drop-rate
```

Additional Features

Logging and Exemptions: uRPF allows the usage of an access-list so you can decide what sources it should check and if required, log the packets that are dropped using access-list logging.

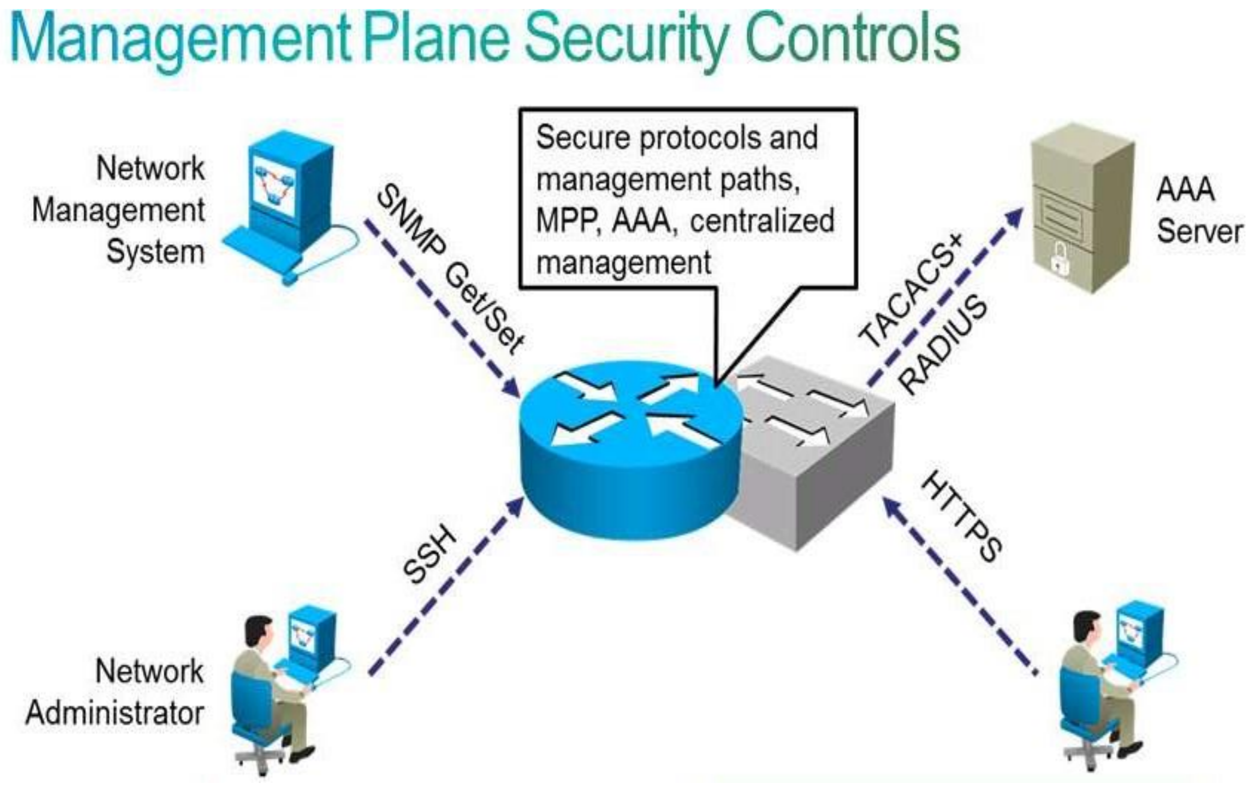
Self-pinging: Allow the router to ping itself using uRPF strict mode on the interface.

Default route: You can configure uRPF to check source IP addresses against a default route. You can use this when you want to accept all packets from your internet connection while protecting yourself against spoofed packets with source IP address from your internal network.

SECURITY ATTACKS:

Management Plane:

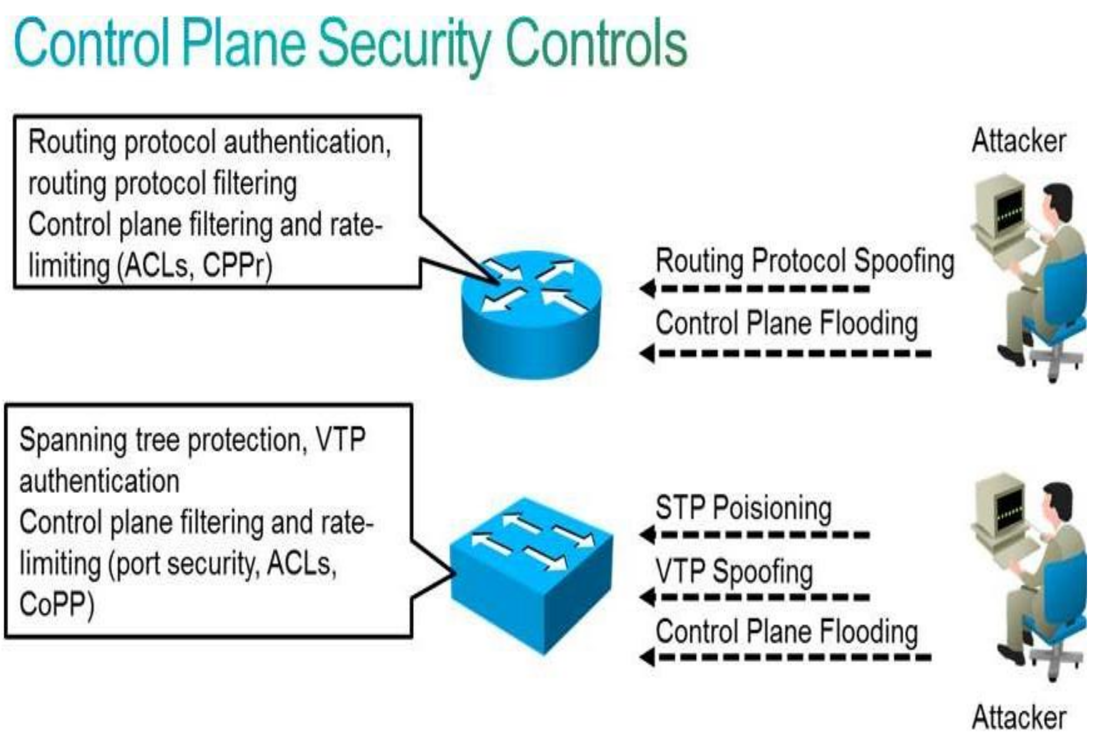
- Management Plan traffic is from the user to the device.
- Protocols & traffic that an admin uses between PC & router or switch itself.
- For example, using SSH to monitor or configure the router or switch.
- For security, use AAA, Authenticated NTP, SSH, syslog, SNMPv3, Parser views.
- Management Plane protocols are FTP, HTTP, HTTPS, SSH, SNMP, Talent, TFTP etc.


Management Plane Security Controls
Network Management System
SNMP Get/Set
Secure protocols and management paths, MPP, AAA, centralized management
TACACS+
RADIUS
AAA Server
SSH
HTTPS
Network Administrator

Control Plan:

- Control Plan traffic is from the device to the device.

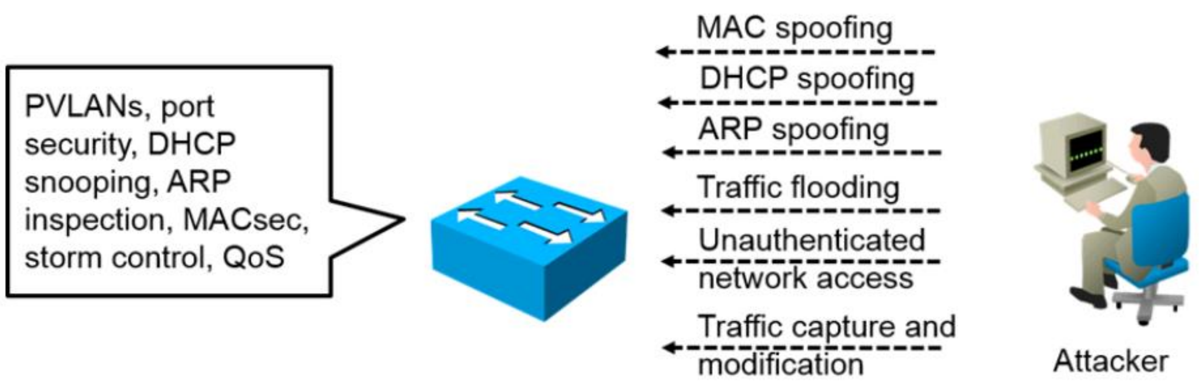
- Control plane traffic is traffic that is originated by, or destined to the router itself.
- Traffic that network devices send between each other for automatic network discovery.
- Protocols & traffic that network devices use on their own without direct interaction.
- For example, a routing protocol that can dynamically learn and share routing information.
- That the router can then use to maintain an updated routing table.
- If failure occurs in the control plane, router might lose the capability to share routing info.
- For security use CoPP, CPPr, Authenticated routing protocol updates.
- Control Plan Includes routing protocols and even ICMP messages.
- Control Plan protocol CDP, LLDP, ARP, OSPF, RIP, BGP, EIGRP etc.


Control Plane Security Controls
Routing protocol authentication, routing protocol filtering
Control plane filtering and rate-limiting (ACLs, CPPr)
Attacker
Routing Protocol Spoofing
Control Plane Flooding
Spanning tree protection, VTP authentication
Control plane filtering and rate-limiting (port security, ACLs, CoPP)
STP Poisioning
VTP Spoofing
Control Plane Flooding
Attacker

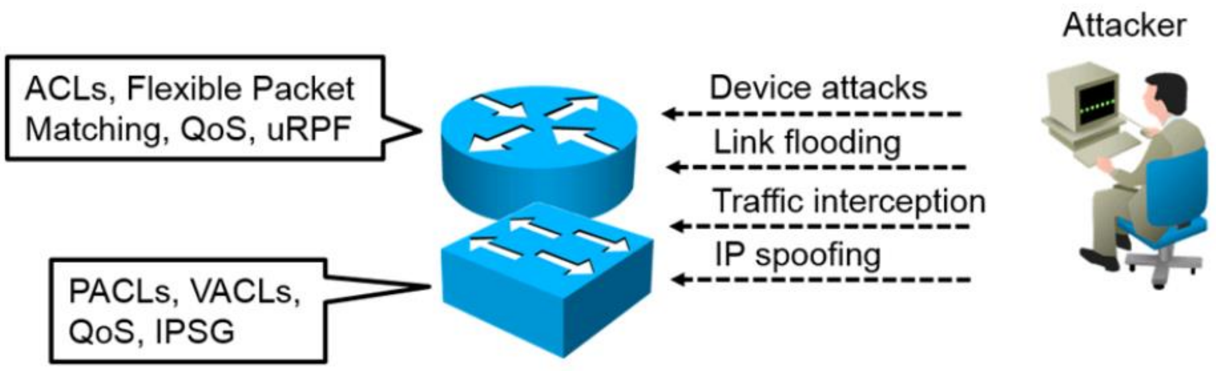
Data Plane:

- Data Plan traffic from the user to the user.
- Data plane traffic that is "just passing through" to get to other destinations.
- Traffic that is being forwarded through the network also called transit traffic.
- Example user sending traffic from one part of network to access a server in another part.
- A failure in the data plane results in the customer's traffic not being able to be forwarded.

Layer 2 Data Plane Security Controls


PVLANs, port security, DHCP snooping, ARP inspection, MACsec, storm control, QoS
MAC spoofing
DHCP spoofing
ARP spoofing
Traffic flooding
Unauthenticated network access
Traffic capture and modification
Attacker

L3 Data Plane Security Controls



Plane	Feature	Benefit
Control Plane	Control Plane Policing (CoPP)	Filter or rate limit control plane traffic with no regard to physical interface
	Control Plane Protection (CPP)	Extend CoPP with granular traffic classification
	Routing protocol authentication	Integrity of routing and forwarding
	Cisco AutoSecure	Automate device hardening
Management Plane	Authentication, authorization and accounting (AAA)	A comprehensive framework for role-based access control
	NTP, Syslog, SNMP, SSH, TLS	Secure management and reporting

	CLI views	Obtain the benefits of RBAC for command line access
Data Plane	Access control lists	Traffic filtering consistent across security device platforms
	Layer 2 controls (private VLANs, STP guards)	Protect the switching infrastructure
	Zone-based firewall, IOS IPS	Deployment flexibility in IOS form factor

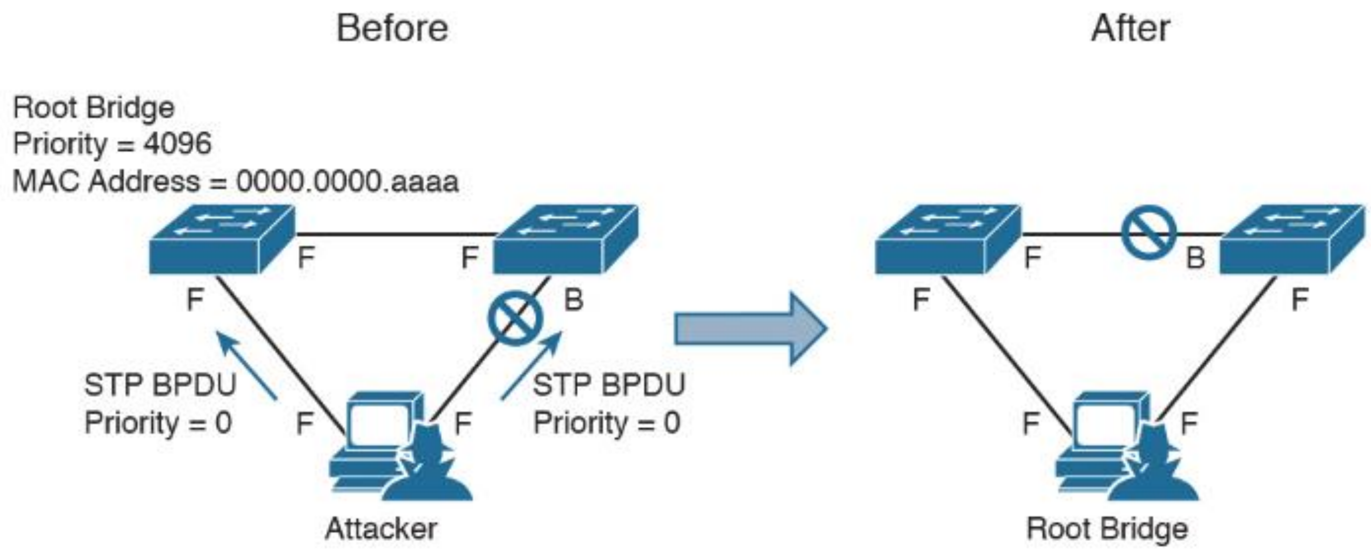
LAYER2 ATTACKS(DATA PLANES)

1] STP Attacks

A network attacker can use STP to change the topology of a network so **that the attacker's host appears to be a root bridge with a higher priority**.

The attacker sends out bridge protocol data units (BPDUs) with a better bridge ID and thus becomes the root bridge.

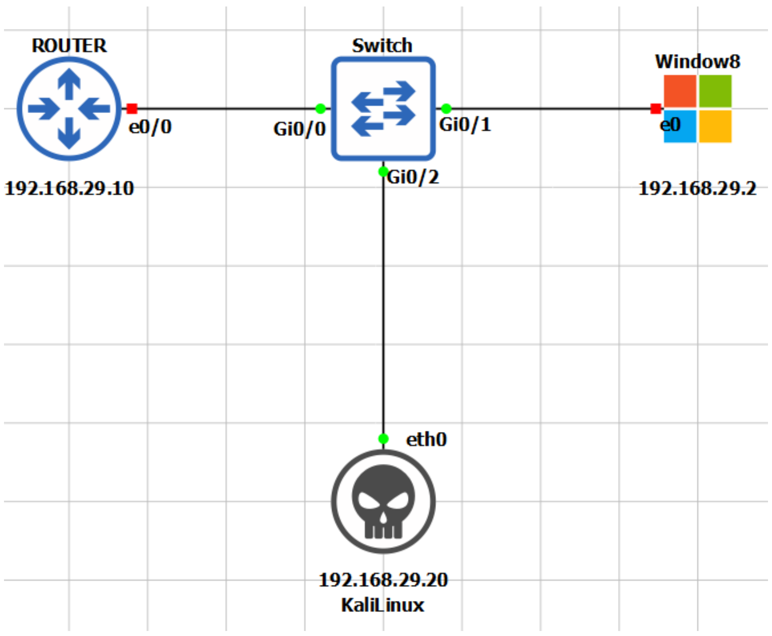
As a result, traffic between the two switches in Figure 14-1 passes through the new root bridge, which is actually the attacker system.



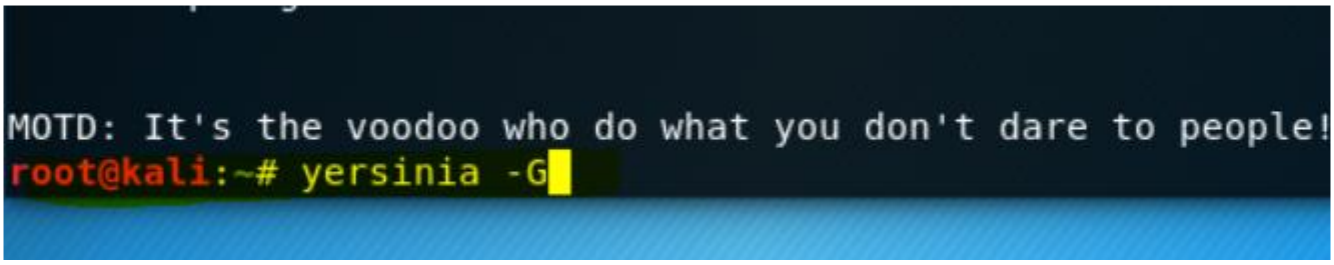
To mitigate STP manipulation attacks, use the Cisco STP stability mechanisms to enhance the overall performance of the switches and to reduce the time that is lost during topology changes.

GNS3

Step 1:

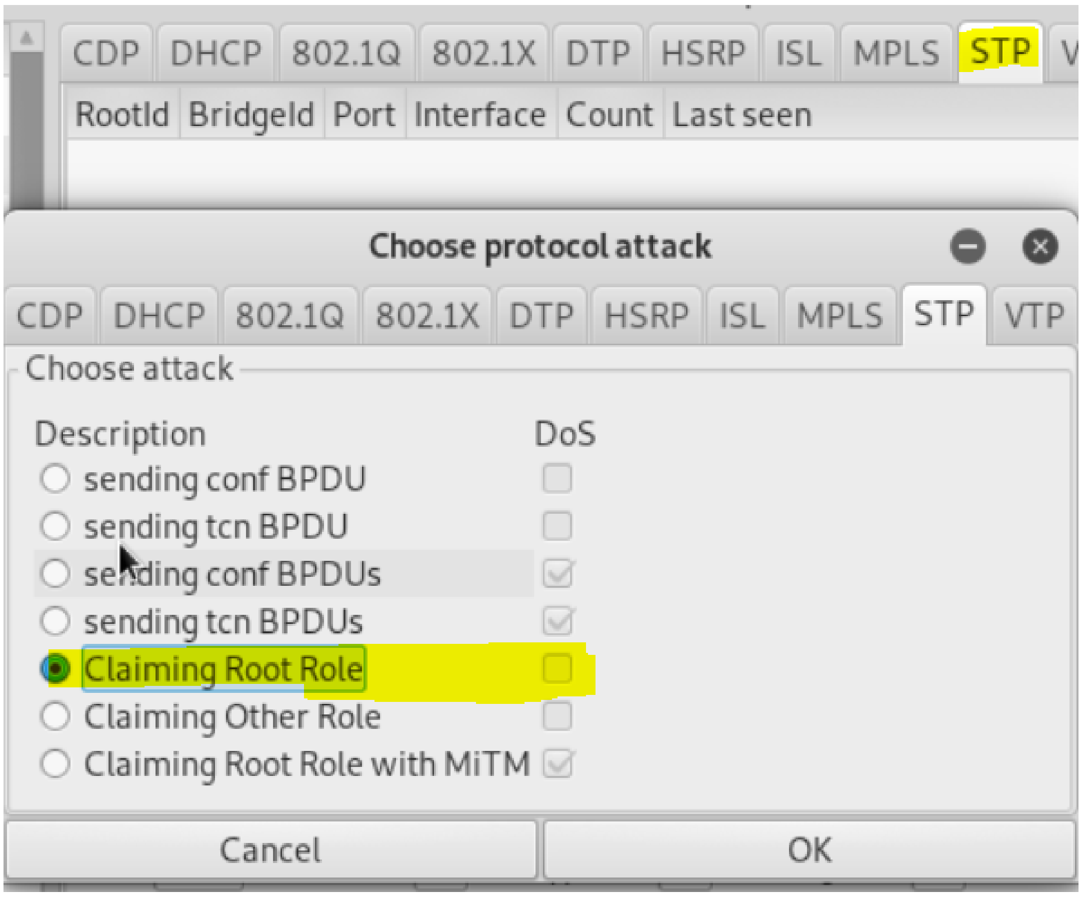


Step 2:



```
MOTD: It's the voodoo who do what you don't dare to people!
root@kali:~# yersinia -G
```

Step 3:



Choose attack

Description	DoS
sending conf BPDU	☐
sending tcn BPDU	☐
sending conf BPDUs	✓
sending tcn BPDUs	✓
[✓ Claiming Root Role]	☐
Claiming Other Role	☐
Claiming Root Role with MiTM	✓

Step 4:

Switch#show spanning-tree

```
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID    Priority    1
             Address     0c15.476e.cd00
             Cost        4
             Port        1 (GigabitEthernet0/0)
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
No more Root bridge. → Root bridge is been taken over by the Kali linux
  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0c15.476c.8500
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
```

Aging Time 300 sec

MITIGATION STEPS:

These are recommended practices for using STP stability mechanisms:

- **Root Guard:** Root guard prevents an inappropriate switch from becoming the root bridge. Root guard limits the switch ports out of which the root bridge may be negotiated. Apply to all ports that should not become root ports.

  Enable RootGuard:

  ```
  SW1(config)#int gi0/1
  SW1(config-if-range)#spanning-tree guard root
  ```

  Observe inconsistent ports:
  ```
  SW1#sh spanning-tree inconsistentports
  ```

- **PortFast:** PortFast immediately brings an interface configured as an access or trunk port to the forwarding state from a blocking state, bypassing the listening and learning states. Apply to all end-user ports. PortFast should only be configured when there is a host attached to the port, and not another switch.
- **BPDU Guard:** BPDU guard immediately error disables a port that receives a BPDU. Typically used on PortFast-enabled ports. Apply to all end-user ports.

Enable BPDUGuard:

```
SW1(config)#int gi0/1
Sw1(config-if)#spanning-tree bpduguard enable
```

Observe err-disable:
```
        SW1#sh errdisable detect | I bpdu
        SW1#sh int | i disabled
```

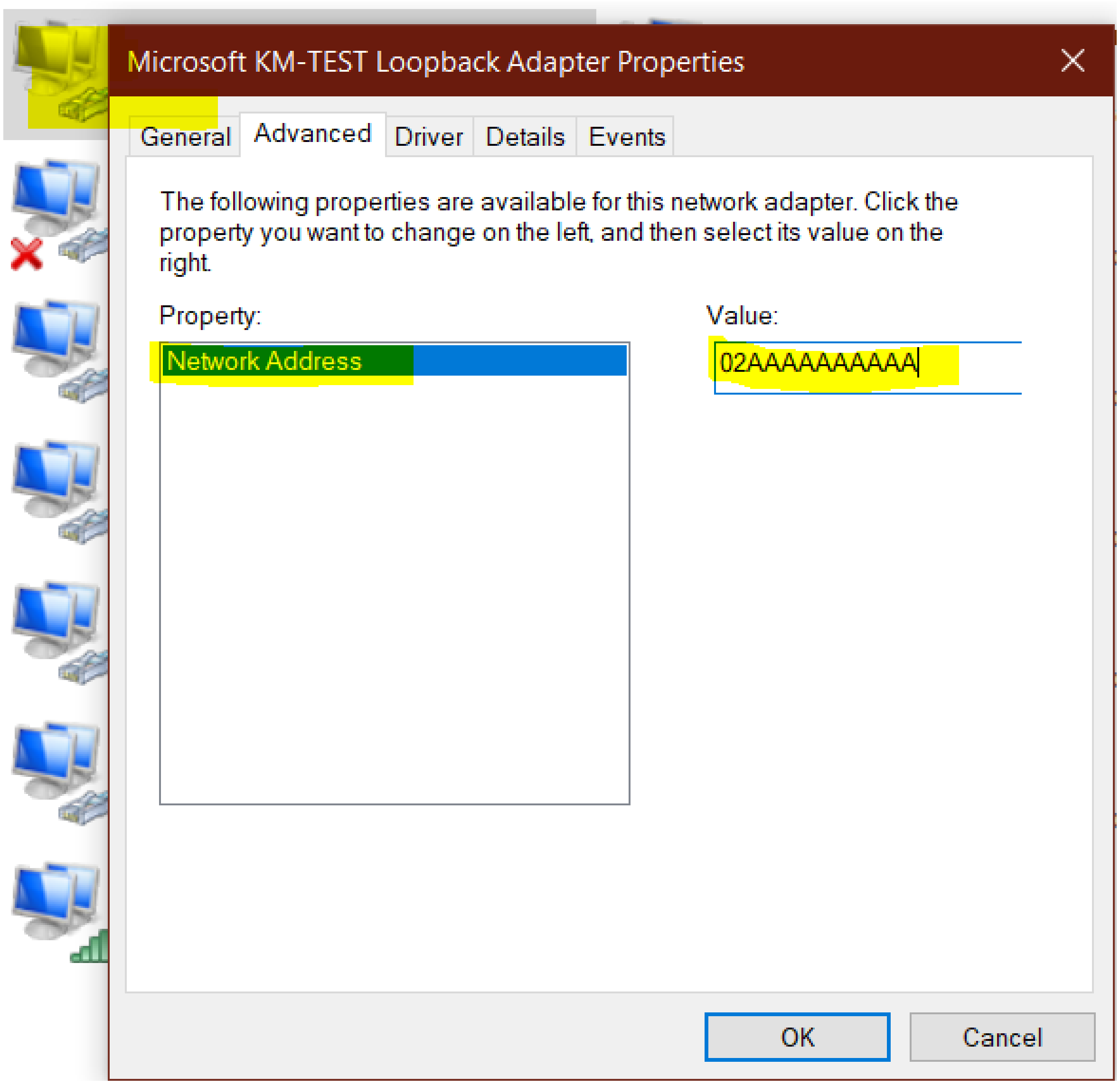
2] MAC Spoofing

MAC spoofing attacks involve the use of a known **MAC address of another host to attempt** to make the target switch forward frames that are destined for the remote host to the network attacker.

By sending a single frame with the source Ethernet address of the other host, **the network attacker overwrites the Content Addressable Memory (CAM) table entry** so that the switch forwards packets that are destined for the host to the network attacker.

Until the host sends traffic, it does not receive any traffic. When the host sends out traffic, the CAM table entry is rewritten once more so that it moves back to the original port. MAC spoofing attacks can be mitigated by **configuring port security.**

Change MAC address in Windows:



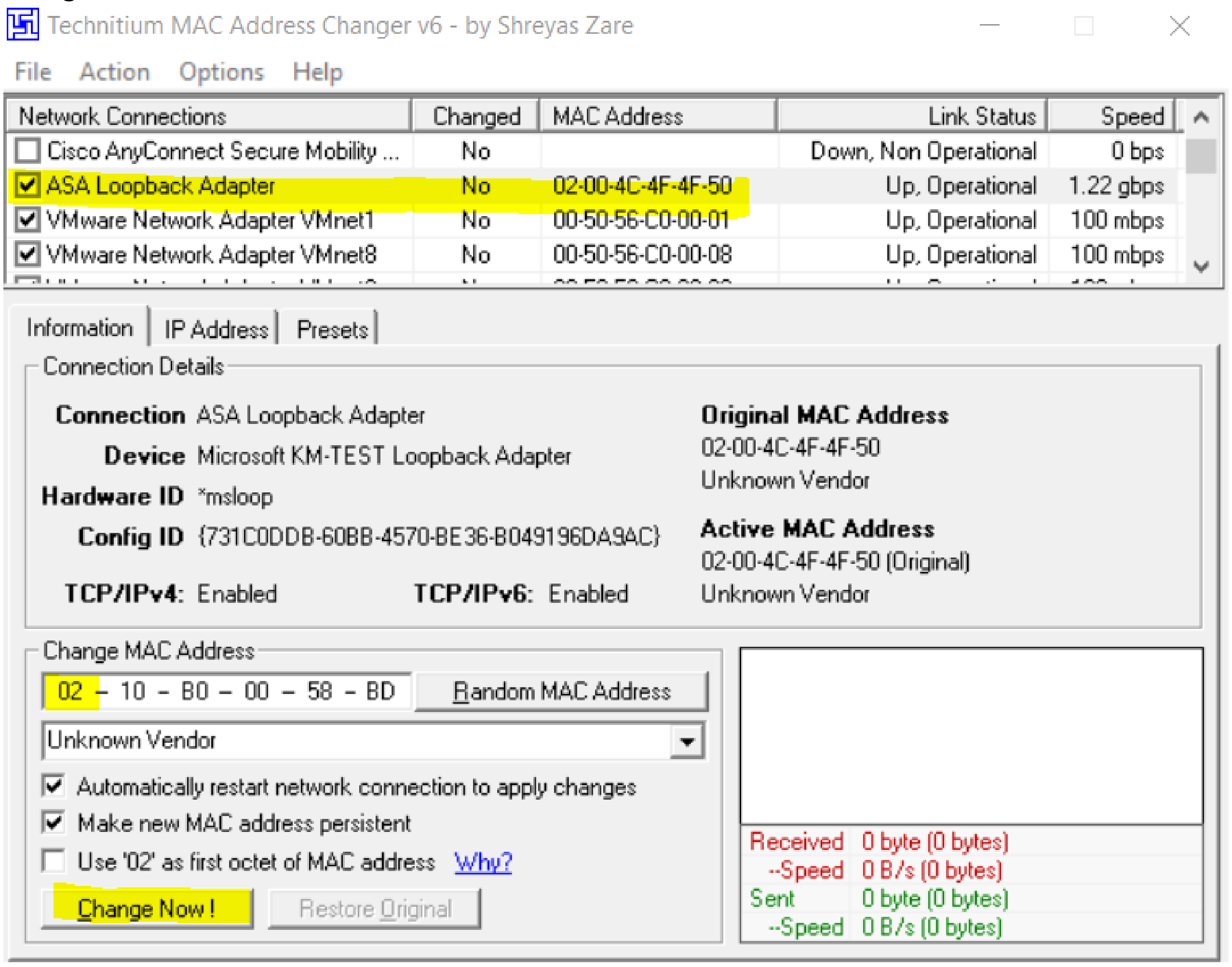
Microsoft KM-TEST Loopback Adapter Properties

The following properties are available for this network adapter. Click the property you want to change on the left, and then select its value on the right.

Property: Network Address

Value: 02AAAAAAAAAA

Change MAC address via software:



Technitium MAC Address Changer v6 - by Shreyas Zare

File Action Options Help

Network Connections	Changed	MAC Address	Link Status	Speed
☐ Cisco AnyConnect Secure Mobility ...	No		Down, Non Operational	0 bps
✓ ASA Loopback Adapter	No	02-00-4C-4F-4F-50	Up, Operational	1.22 gbps
✓ VMware Network Adapter VMnet1	No	00-50-56-C0-00-01	Up, Operational	100 mbps
✓ VMware Network Adapter VMnet8	No	00-50-56-C0-00-08	Up, Operational	100 mbps

Information | IP Address | Presets

Connection Details

Connection ASA Loopback Adapter
Device Microsoft KM-TEST Loopback Adapter
Hardware ID *msloop
Config ID {731C0DDB-60BB-4570-BE36-B049196DA9AC}
TCP/IPv4: Enabled **TCP/IPv6:** Enabled

Original MAC Address
02-00-4C-4F-4F-50
Unknown Vendor

Active MAC Address
02-00-4C-4F-4F-50 (Original)
Unknown Vendor

Change MAC Address

02 - 10 - B0 - 00 - 58 - BD   Random MAC Address

Unknown Vendor

✓ Automatically restart network connection to apply changes
✓ Make new MAC address persistent
☐ Use '02' as first octet of MAC address Why?

Change Now !   Restore Original

Received	0 byte (0 bytes)
--Speed	0 B/s (0 bytes)
Sent	0 byte (0 bytes)
--Speed	0 B/s (0 bytes)

3] CAM Table Overflows

The most important point to understanding how CAM overflow attacks work is to know that **CAM tables are limited in size.**

MAC flooding takes advantage of this limitation by **bombarding the switch with fake source MAC addresses** until the switch CAM table is full.

If enough entries are entered into the CAM table, the CAM table fills up to the point that **no new entries can be accepted.**

Note that this attack will not cause legitimate entries that were in the CAM table before the attack to be removed. As long as activity is seen from the legitimate MAC addresses, their spot in the CAM table will be reserved. But if they time out, the **attack will quickly consume the freed spot** in the CAM table and the legitimate MAC address will not be relearned.

BEFORE:

SW1#sh mac address

```
  1    0c15.4799.3c00    DYNAMIC     Gi0/1
```

SW1#sh mac address-table aging-time

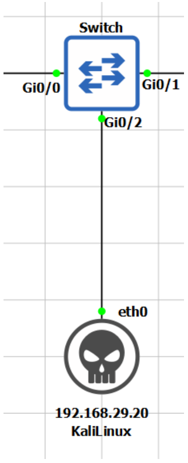
Global Aging Time: 300

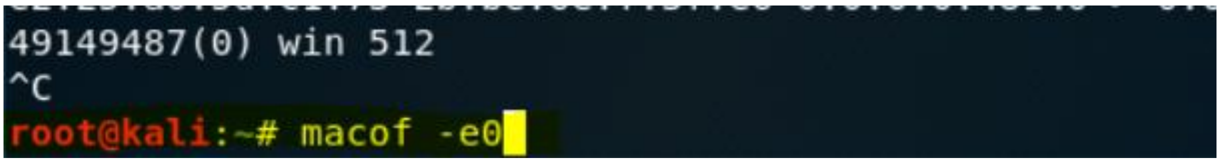
SW1#sh mac address-table count

Static Address Count : 0

Total Mac Addresses   : 3

Total Mac Address Space Available: 70152664





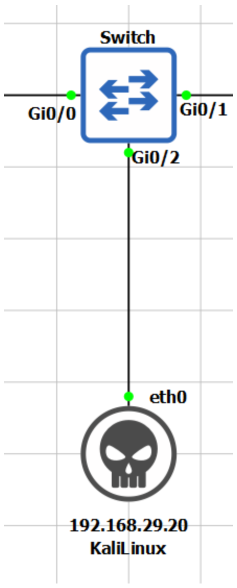
```
49149487(0) win 512
^C
root@kali:~# macof -e0
```

VALIDATE:

SW1#sh mac address-table

3] Port Security

- Port security feature protect the switch from MAC flooding attacks
- Port security feature protect the switch from DHCP starvation attacks
- Attacker start flooding the switch with very large number of DHCP requests
- Port security prevent unauthorized access & limit access, based on MAC address
- Port security is disabled by default on every interface of switch
- If port security is enabled by default only one MAC address is, allowed per interface
- If port security is enabled by default, violation is shutdown
- If port security is enabled by default, no aging is configured for recovery
- **Port security** can be configured Static, Dynamic and Sticky
- There are three different types of **violation** Shutdown, Protect and Restrict
- Port security limit (1-8192) MAC address to attach on particular port.

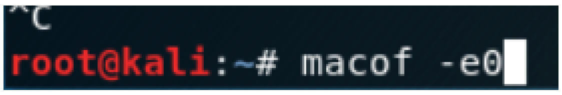


GNS3 LAB

```
SW1(config-if)#interface GigabitEthernet0/1

 switchport mode access

 switchport port-security

SW1(config-if)#sw port-security mac-address 0c15.476e.cd01

SW1(config-if)#do sh port-se add

          Secure Mac Address Table

-----------------------------------------------------------------------------

Vlan    Mac Address       Type                          Ports   Remaining Age

                                                 (mins)

----    -----------       ----                    -----    -------------

   1    0c15.476e.cd01    SecureConfigured            Gi0/1      -
```

```
---------------------------------------------------------------------------
Total Addresses in System (excluding one mac per port)     : 0
Max Addresses limit in System (excluding one mac per port) : 4096
```

RUN KALI TO OBSERVE DEBUG:



```
^C
root@kali:~# macof -e0
```

VOILATION OCCURRED:

```
*Sep 30 20:35:37.112: %PM-4-ERR_DISABLE: psecure-violation error detected on Gi0/1, putting Gi0/1 in err-disable state

*Sep 30 20:35:37.114: %PORT_SECURITY-2-PSECURE_VIOLATION: Security violation occurred, caused by MAC address 7a5b.5f60.df03 on port GigabitEthernet0/1.

*Sep 30 20:35:38.112: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to down

*Sep 30 20:35:39.114: %LINK-3-UPDOWN: Interface GigabitEthernet0/1, changed state to down
```

To enable Auto ERR DISABLE recovery

```
Switch(config)# errdisable recovery cause psecure-violation
Switch(config)# errdisable recovery interval 300
```

Verify → Switch# **show errdisable recovery**

Switch# **show errdisable recovery**

%PM-4-ERR_RECOVER: Attempting to recover from psecure-violation err-disable state on Fa0/1

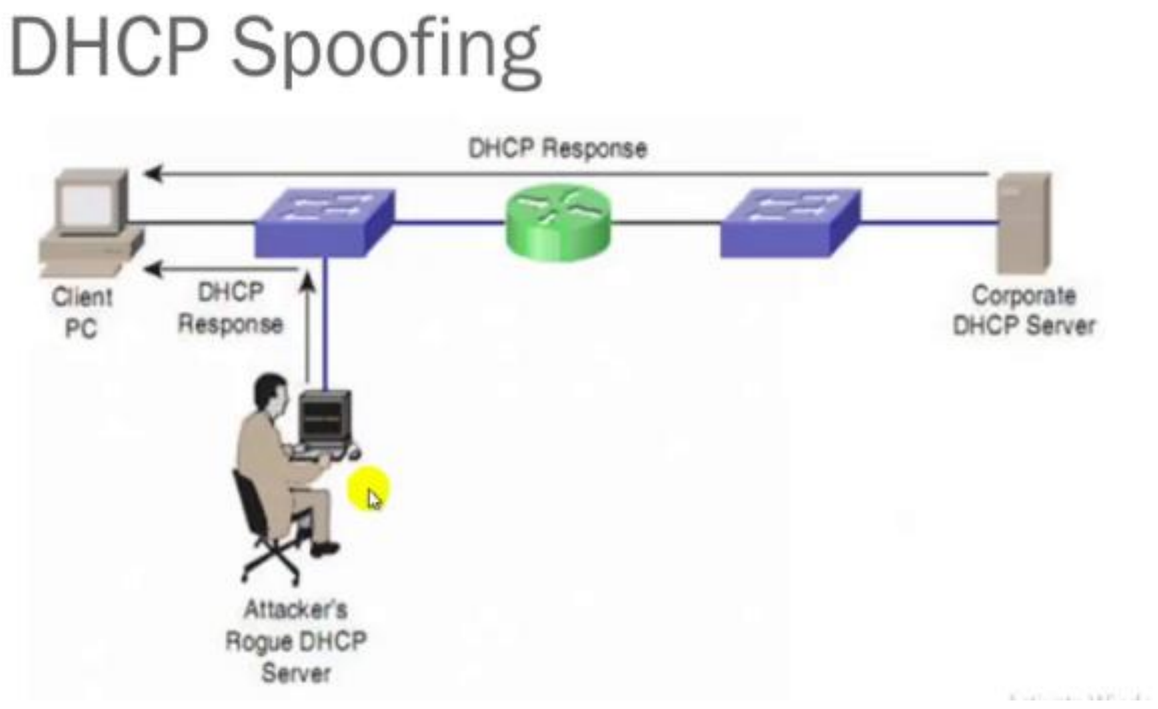
```
Switch# show interface Gi0/2
GigaEthernet0/2 is up, line protocol is up (connected)
```

4] DHCP Spoofing

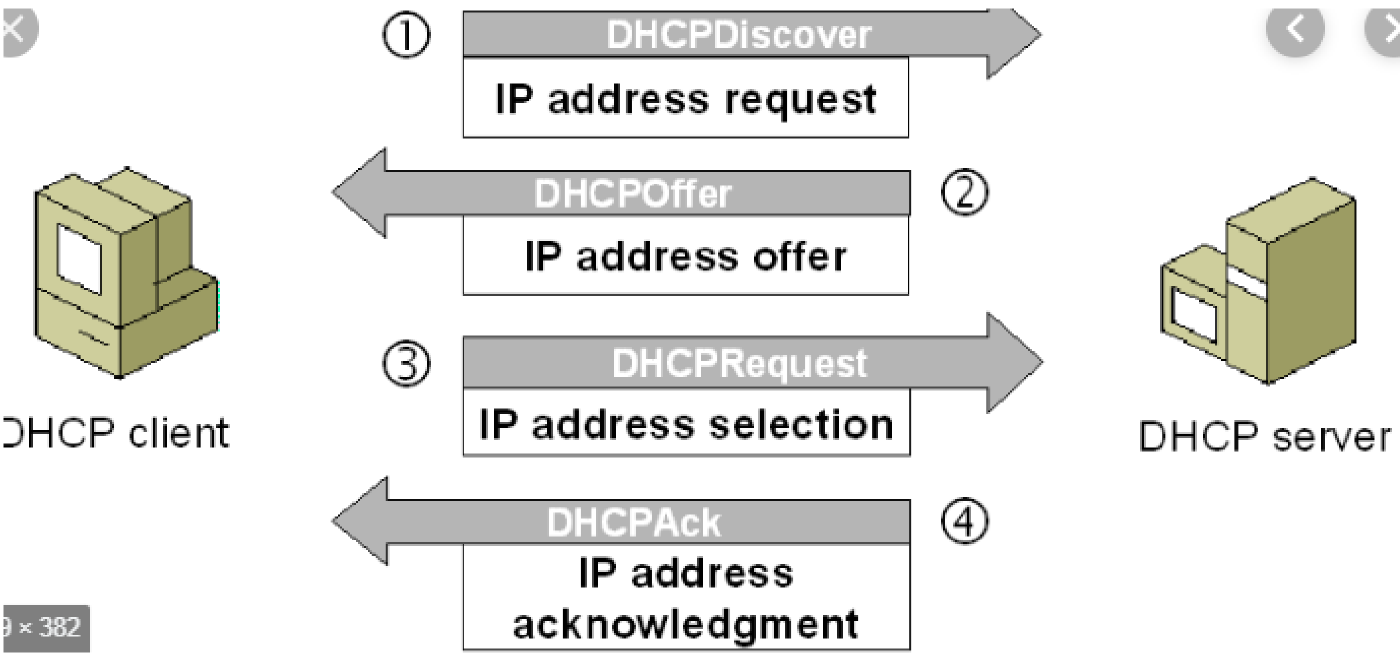
Access attacks is instant with no proper planning, Example, man-in-middle, buffer overflow, dhcp spoofing

DHCP spoofing:

- **DHCP spoofing** occurs when an attacker attempts to respond to DHCP requests and trying to list themselves (spoofs) as the default gateway or DNS server, hence, initiating a man in the middle attack. With that, it is possible that they can intercept traffic from users before forwarding to the real gateway or perform DoS by flooding the real DHCP server with request to choke ip address resources.
- This can be mitigated by configuring DHCP **Snooping** which enables specific ports only to pass DHCP traffic. All other ports will be untrusted and can only send DHCP requests. If a DHCP offer is detected in an untrusted port, it will be shut down. Let's see a sample config.

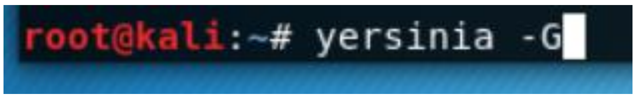


DHCP process



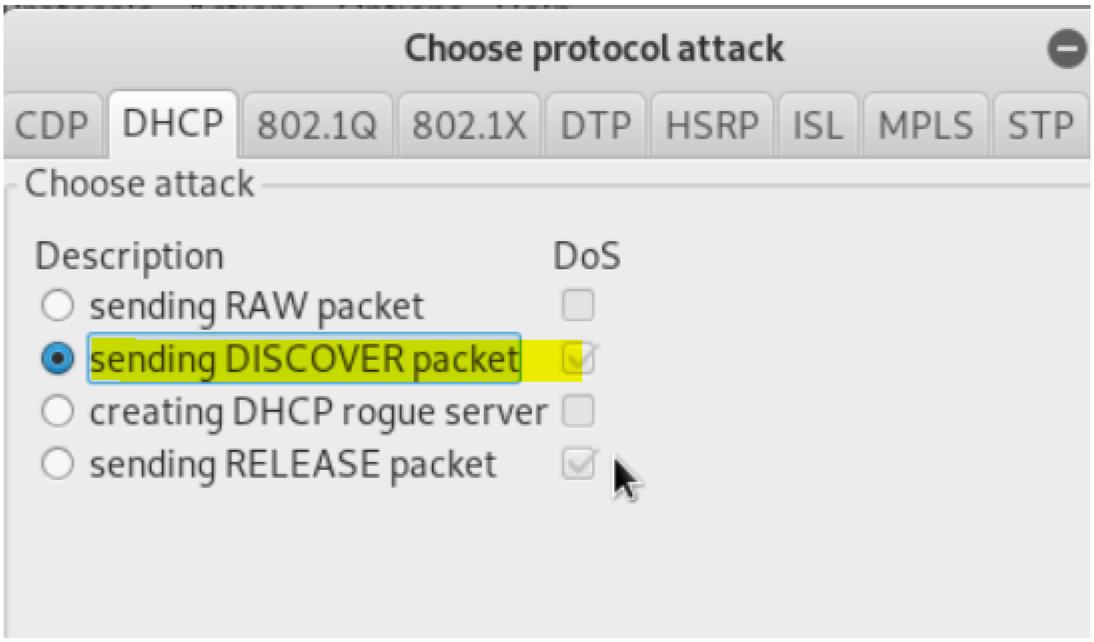
GNS3 lab:

1. **Launch Kali**



```
root@kali:~# yersinia -G
```

2. Launch attack



Choose protocol attack

CDP | DHCP | 802.1Q | 802.1X | DTP | HSRP | ISL | MPLS | STP

Choose attack

Description	DoS
○ sending RAW packet	☐
● sending DISCOVER packet	✓
○ creating DHCP rogue server	☐
○ sending RELEASE packet	✓

3. **Config on IOU Router**

```
IOU1(config)#ip dhcp pool ccnp
IOU1(dhcp-config)#network 192.168.29.0 255.255.255.0
IOU1(dhcp-config)#default-router 192.168.29.1
IOU1(dhcp-config)#dns-server 8.8.8.8
IOU1(config)#ip dhcp excluded-address 192.168.29.1 192.168.29.50
```

```
IOU1(config)#int e0/0
Ip add 192.168.29.1 255.255.255.0
No shut
```

Verify

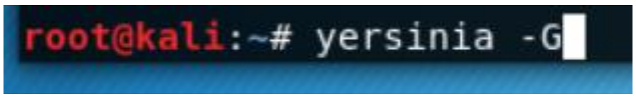
Show ip dhcp binding

Configure DHCP SNOOPING (MITIGATION PROCESS)

- SW(config)# ! Enable DHCP snooping on the switch
- **SW(config)#ip dhcp snooping**
- SW(config)# ! Enable DHCP snooping for the specific VLAN
- **SW(config)#ip dhcp snooping vlan 1**
- **SW(config)#int gi0/2**
- SW(config-if)# ! Set the port as trusted
- **SW(config-if)#ip dhcp snooping trust**
- SW(config-if)# ! Enable rate limiting to prevent flooding attacks
- **SW(config-if)#ip dhcp snooping limit rate 15**
- **SW (config) # no ip dhcp snooping information option**

GNS3 lab:

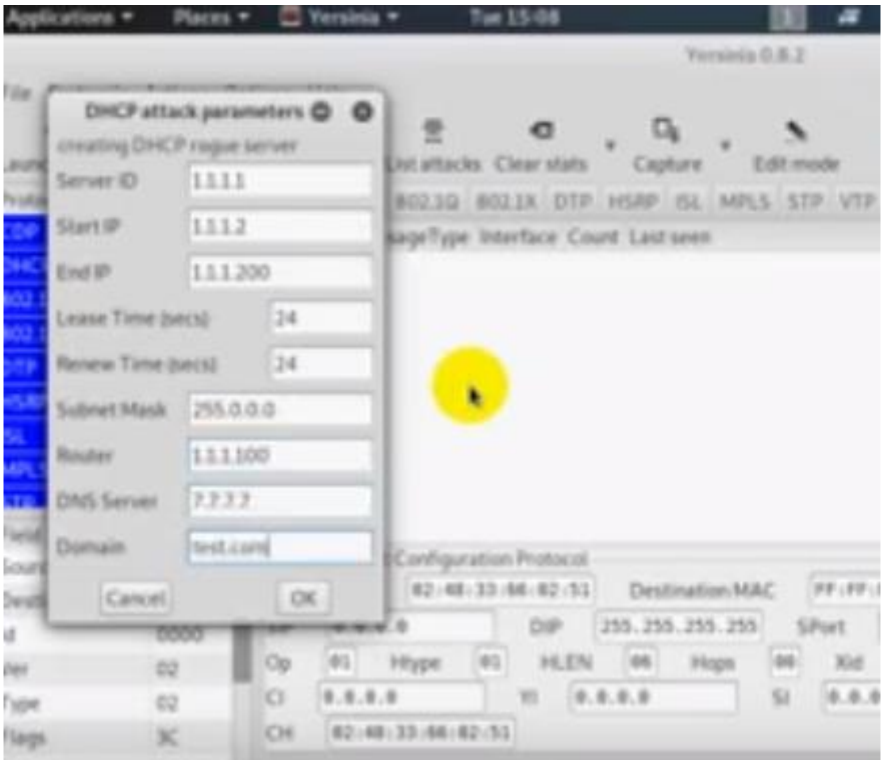
1. **Launch Kali**



```
root@kali:~# yersinia -G
```

2. Open

DHCP VOGUEE SERVER



DHCP attack parameters

creating DHCP rogue server

Server ID	1.1.1.1
Start IP	1.1.1.2
End IP	1.1.1.200
Lease Time (secs)	24
Renew Time (secs)	24
Subnet Mask	255.0.0.0
Router	1.1.1.100
DNS Server	7.7.7.7
Domain	test.com

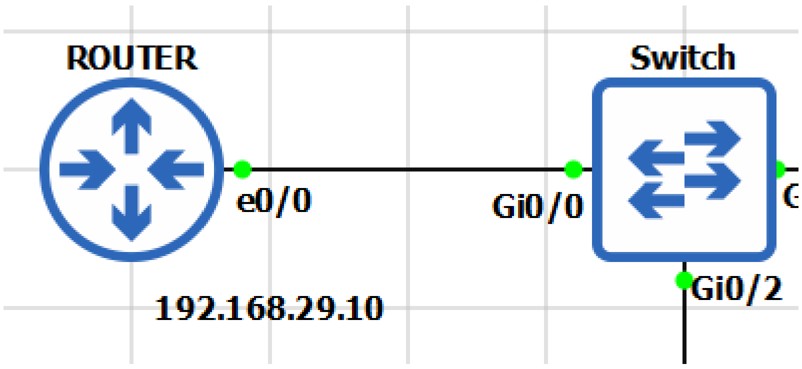
- Verify

show ip int br

show ip dhcp snooping

MANAGEMENT PLANES ATTACKS

TELNET (NON PREFERED) VS SSH(PREFERABLE)



GNS3

TELNET TEST:

ROUTER(config)#:

```
interface Ethernet0/0
 ip address 192.168.29.10 255.255.255.0
no shutdown
```

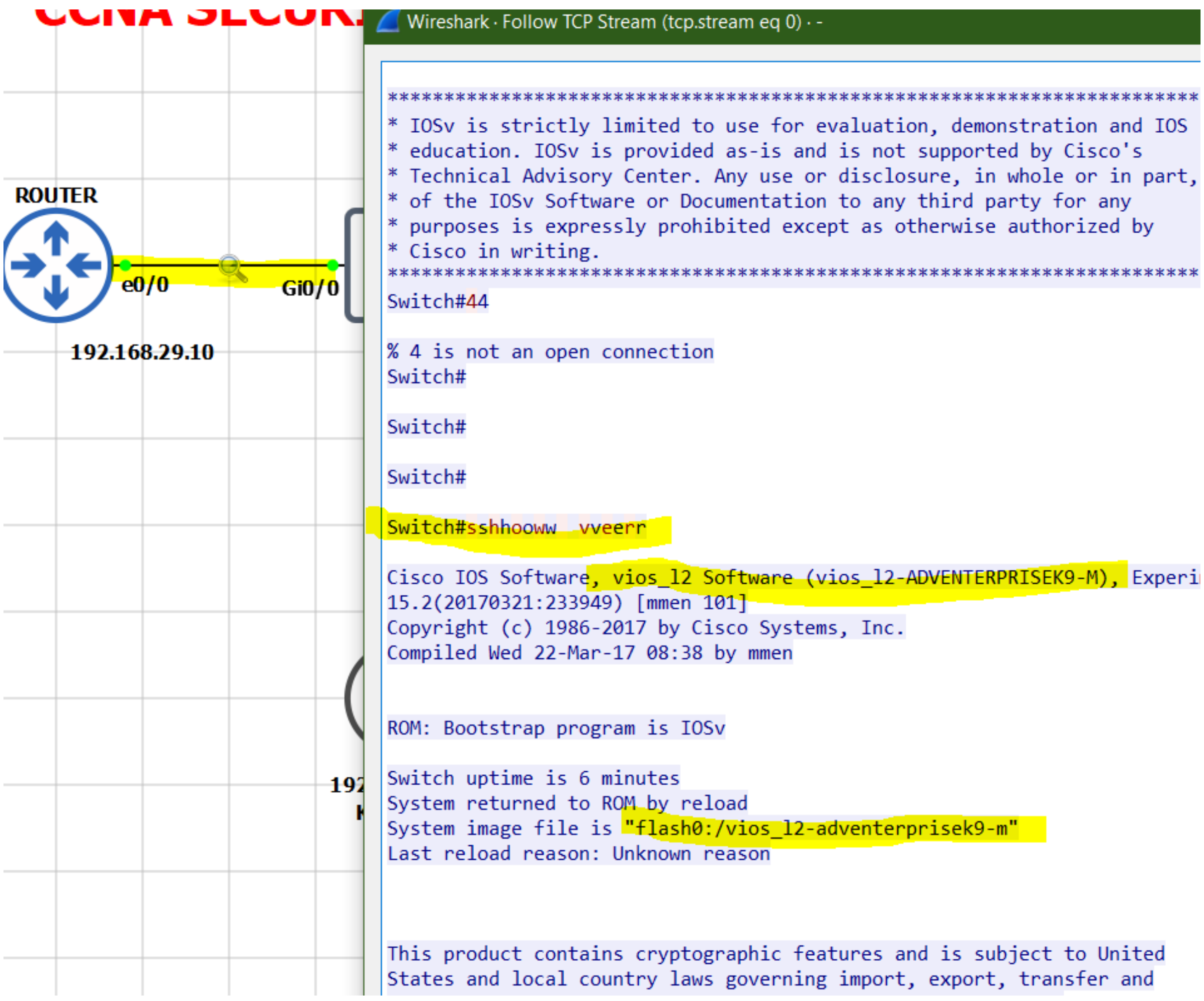
SWITCH(config)#:

```
interface GigabitEthernet0/0
 no switchport
 ip address 192.168.29.1 255.255.255.0
 exit
username admin privilege 15 password 0 cisco
line vty 0 4
 login local
 transport input telnet
```

[to be continued in next page]

TRAFFIC CAPTURED ARE ALL PLAIN-TEXT:

[WIRESHARK SCREENSHOT ATTACHED BELOW]



```
**************************************************************************
* IOSv is strictly limited to use for evaluation, demonstration and IOS
* education. IOSv is provided as-is and is not supported by Cisco's
* Technical Advisory Center. Any use or disclosure, in whole or in part,
* of the IOSv Software or Documentation to any third party for any
* purposes is expressly prohibited except as otherwise authorized by
* Cisco in writing.
**************************************************************************
Switch#44

% 4 is not an open connection
Switch#

Switch#

Switch#

Switch#sshhooww  vveerr

Cisco IOS Software, vios_l2 Software (vios_l2-ADVENTERPRISEK9-M), Experi
15.2(20170321:233949) [mmen 101]
Copyright (c) 1986-2017 by Cisco Systems, Inc.
Compiled Wed 22-Mar-17 08:38 by mmen


ROM: Bootstrap program is IOSv

Switch uptime is 6 minutes
System returned to ROM by reload
System image file is "flash0:/vios_l2-adventerprisek9-m"
Last reload reason: Unknown reason



This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
```

[to be continued in next page]

SSH TEST:
Switch(config)#line vty 0 4

Switch(config-line)#transport input ssh

Switch(config-line)#exit

Switch(config)#ip domain-name networkjourney.com

Switch(config)#cry key generate rsa modulus 1024

The name for the keys will be: Switch.networkjourney.com

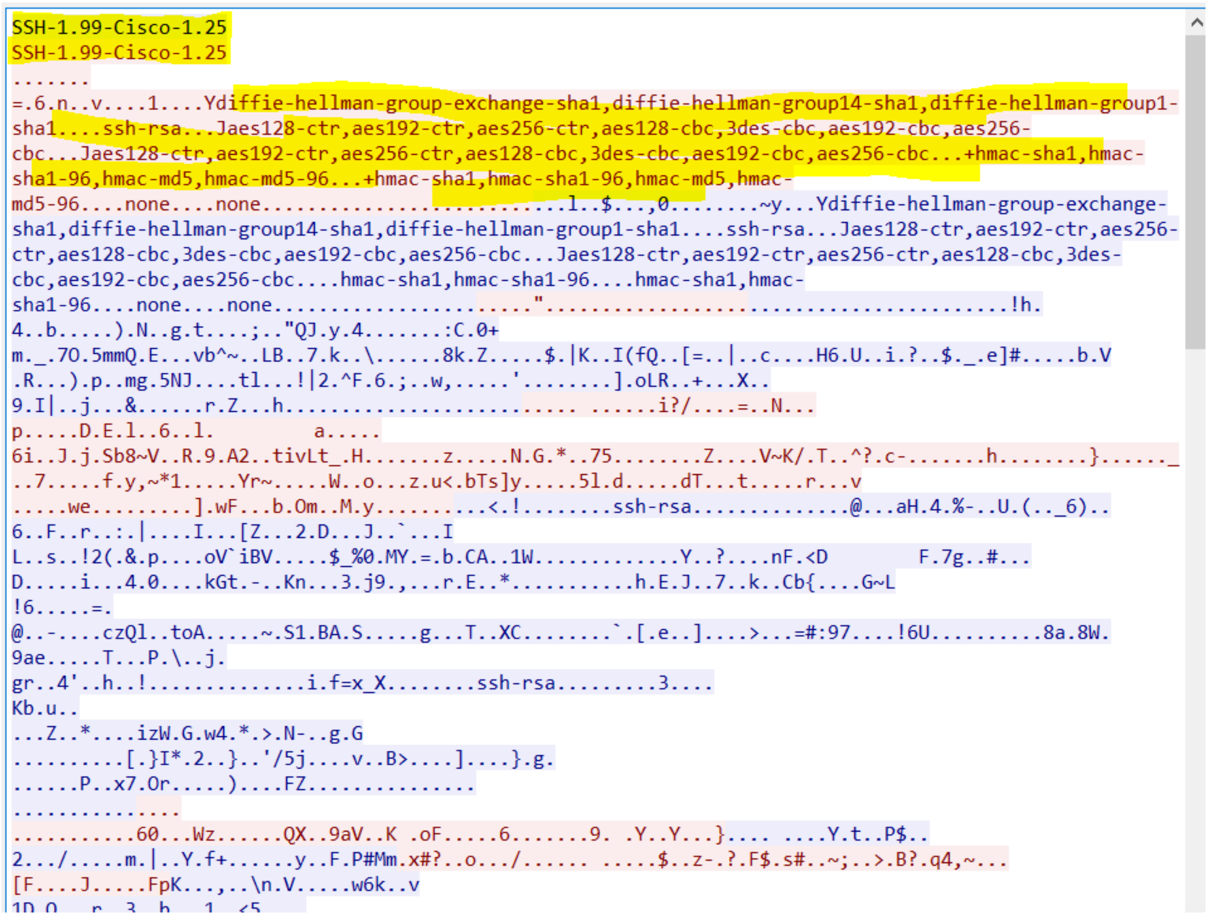
% The key modulus size is 1024 bits

% Generating 1024 bit RSA keys, keys will be non-exportable...

[OK] (elapsed time was 1 seconds)

ROUTER#ssh -l admin 192.168.29.1

SSH encrypts both data and username/password as well



```
SSH-1.99-Cisco-1.25
SSH-1.99-Cisco-1.25
.......
=.6.n..v....1....Ydiffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-
sha1....ssh-rsa...Jaes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc,aes192-cbc,aes256-
cbc...Jaes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc...+hmac-sha1,hmac-
sha1-96,hmac-md5,hmac-md5-96...+hmac-sha1,hmac-sha1-96,hmac-md5,hmac-
md5-96....none....none........................l..$...,0........~y...Ydiffie-hellman-group-exchange-
sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1....ssh-rsa...Jaes128-ctr,aes192-ctr,aes256-
ctr,aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc...Jaes128-ctr,aes192-ctr,aes256-ctr,aes128-cbc,3des-
cbc,aes192-cbc,aes256-cbc....hmac-sha1,hmac-sha1-96....hmac-sha1,hmac-
sha1-96....none....none.......................".......................................!h.
4..b.....).N..g.t....;.."QJ.y.4.......:C.0+
m._.7O.5mmQ.E...vb^~..LB..7.k..\......8k.Z.....$.|K..I(fQ..[=..|..c....H6.U..i.?..$._.e]#.....b.V
.R...).p..mg.5NJ....tl...!|2.^F.6.;..w,.....'........].oLR..+...X..
9.I|..j...&......r.Z...h........................ ......i?/....=..N...
p.....D.E.l..6..l.        a.....
6i..J.j.Sb8~V..R.9.A2..tivLt_.H.......z.....N.G.*..75........Z....V~K/.T..^?.c-.......h........}......_
..7.....f.y,~*1.....Yr~.....W..o...z.u<.bTs]y.....5l.d.....dT...t.....r...v
.....we.........].wF...b.Om..M.y.............<.!........ssh-rsa..............@...aH.4.%-..U.(.._6)..
6..F..r..:.|....I...[Z...2.D...J..`...I
L..s..!2(.&.p....oV`iBV.....$_%0.MY.=.b.CA..1W.............Y..?....nF.<D        F.7g..#...
D.....i...4.0....kGt.-..Kn...3.j9.,...r.E..*...........h.E.J..7..k..Cb{....G~L
!6.....=.
@..-....czQl..toA.....~.S1.BA.S.....g...T..XC........`.[.e..]....>...=#:97....!6U..........8a.8W.
9ae.....T...P.\..j.
gr..4'..h..!..............i.f=x_X........ssh-rsa.........3....
Kb.u..
...Z..*....izW.G.w4.*.>.N-..g.G
..........[.}I*.2..}..'/5j....v..B>....]....}.g.
......P..x7.Or.....)....FZ...............
...............
...........60...Wz......QX..9aV..K .oF.....6.......9. .Y..Y...}.... ....Y.t..P$..
2.../.....m.|..Y.f+......y..F.P#Mm.x#?..o.../...... .....$..z-.?.F$.s#..~;..>.B?.q4,~...
[F....J.....FpK...,..\n.V.....w6k..v
```

SECURE MANAGEMENT PLANE ATTACKS

(MPP) Management Place Protection

Management Plane Protection (MPP) is a security feature for Cisco IOS routers that accomplishes two things:

- Restrict the interfaces where the router permits packets from network management protocols.
- Restrict the network management protocols that the router permits.

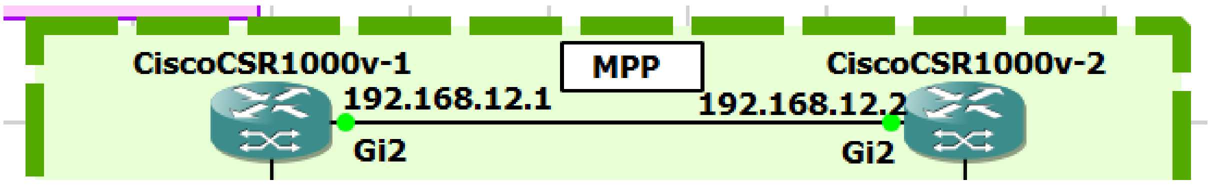
The management plane is the logical path of all traffic related to the management of the router. For example:

- Telnet
- SSH
- SNMP
- HTTP
- HTTPS

- After MPP is enabled, no interfaces except designated management interfaces will accept network management traffic destined to the device.
- Restricting management packets to designated interfaces provides greater control over management of a device, providing more security for that device.
- Other benefits include improved performance for data packets on non-management interfaces, support for network scalability.
- Need for fewer access control lists (ACLs) to restrict access to a device.
- Management packet floods on switching and routing interfaces are prevented from reaching the CPU.

GNS3:

WITHOUT MPP:



CiscoCSR1000v-2:

```
Hostname CiscoCSR1000v-2
interface Gi 2
 ip address 192.168.12.2 255.255.255.0
no shutdown
exit
username admin privilege 15 password 0 cisco
```

```
line vty 0 4
 login local
 transport input ssh
```

CiscoCSR1000v-1:

```
Hostname CiscoCSR1000v-1
interface Gi 2
 ip address 192.168.12.1 255.255.255.0
no shutdown
exit
```

ENABLE DEBUG:

```
DHCPSERVERR18# Debug ip packets
```

```
DHCPSERVERR18#

*Jul 15 16:24:41.333: IP: s=192.168.1.100 (local), d=192.168.1.1 (Ethernet0/0), len 40, sending

*Jul 15 16:24:41.333: IP: s=192.168.1.100 (local), d=192.168.1.1 (Ethernet0/0), len 40, sending full
packet
```

We see that the connection is refused, this is expected because we don't accept telnet on the VTY lines of CiscoCSR1000v-2. When you look at CiscoCSR1000v-2 you see it sends two packets to CiscoCSR1000v-1

This type of attack can be used as denial of service attack on CiscoCSR1000v-2 and also consumes percentage from the CPU resource.

WITH MPP:

Now we configure MPP, it's a subset of Control Plane Policing (COPP). so it's under the control-plane host command. Then use the management-interface command and specify the interface:

Not Supported by Older IOS code.

Supported by Cisco IOS Release 15M&T supports these features.

```
CiscoCSR1000v-2 (config)#control-plane host
(config-cp-host)#management-interface GigabitEthernet 2 allow ?
  beep Beep Protocol
  ftp File Transfer Protocol
  http HTTP Protocol
  https HTTPS Protocol
  snmp Simple Network Management Protocol
  ssh Secure Shell Protocol
  telnet Telnet Protocol
  tftp Trivial File Transfer Protocol
  tl1 Transaction Language Session Protocol
```

Let's choose telnet:

```
CiscoCSR1000v-2
(config-cp-host)#management-interface GigabitEthernet 2 allow ssh
```

Let's try telnet from CiscoCSR1000v-1:

```
CiscoCSR1000v-1#telnet 192.168.1.100
Trying 192.168.2.254 ...
% Connection timed out; remote host not responding
```

The telnet connection from DHCPRELAYSERVERR17 fails and you can see a different message, the connection now times out. DHCPRSERVERR18 silently drops the telnet packets from DHCPRELAYSERVERR17 and does not reply, hence, there is no load on CPU.

```
Verifications:
CiscoCSR1000v-2#show management-interface
Management interface GigabitEthernet2
        Protocol Packets processed
          telnet 9


CiscoCSR1000v-2#show management-interface protocol telnet
The following management-interfaces allow protocol telnet
        GigabitEthernet2 Packets processed 9
```

BENEFITS OF USING MPP:

- MPP restricts the interfaces where the router permits network management protocols.
- MPP restricts the network management protocols it permits.
- You need fewer access-lists because you don't need to add them to all your interfaces.
- MPP silently drops denied packets and does not forward them to the CPU.
-

SECURE CONTROL PLANE ATTACKS

Control Plane Policing (CoPP)

On our routers (or multilayer switches) we can use access-lists or firewalls (CBAC or zone based) to permit/deny packets that go through or to the router.

We can also use policing to rate limit that goes through our router.

What if we want to police traffic that is destined to the router? There are quite some protocols that produce packets that the router has to process:

- Routing protocols like OSPF, EIGRP, or BGP.
- Gateway redundancy protocols like HSRP, VRRP, or GLBP.
- Network management protocols like telnet, SSH, SNMP, or RADIUS.
- Packets that CEF can't forward.

The route processor inspects packets that these protocols generate on the control plane. When the route processor receives too many packets, it's possible that it can't keep up and drops packets.

When this happens, you'll see things like flapping neighbor adjacencies or timeouts when you try to connect with telnet/SSH to the router.

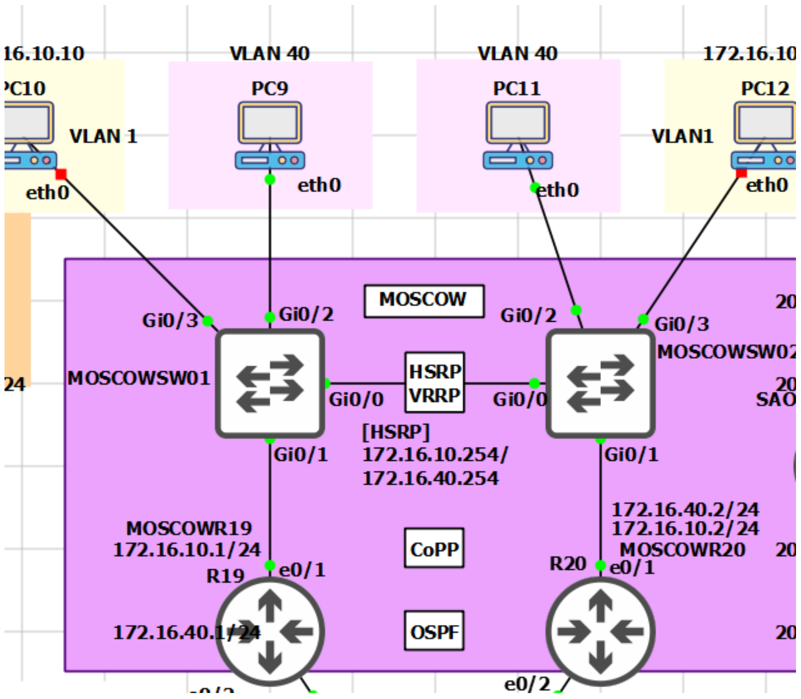
To prevent this from happening, we have a couple of options:

- rACLs (Receive Access Control List): these are standard or extended ACL that control traffic sent by line cards to the route processor. You only see this feature on high-end routers like the Cisco 12000 series.
- Control Plane Policing (CoPP): allows you to use MQC (Modular Quality of Service) framework to permit/deny or rate-limit traffic that goes to the route processor.
- Control Plane Protection (CPPr): this is an extension of CoPP. One of the things it does is separating the route processor into three sub-interfaces:
  - host
  - transit
  - CEF exception

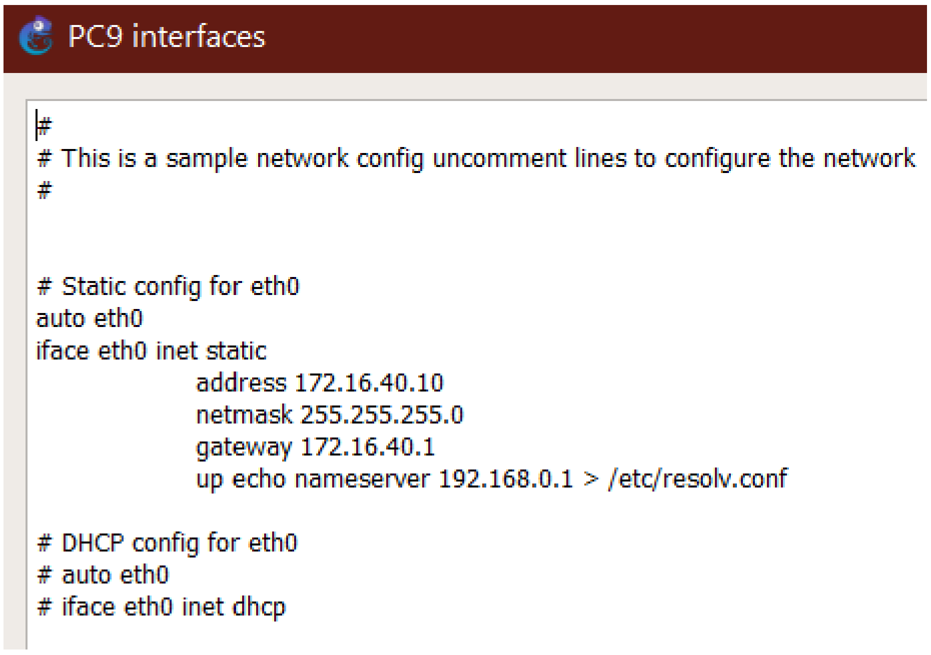
Control plane policing uses the MQC (Modular Quality of Service) so that means we have to use class-maps and a policy-map. In your class-maps, it's best to match traffic on:

- standard or extended access-lists
- DSCP or IP precedence values.

GNS3:



PC9:



```
#
# This is a sample network config uncomment lines to configure the network
#


# Static config for eth0
auto eth0
iface eth0 inet static
                address 172.16.40.10
                netmask 255.255.255.0
                gateway 172.16.40.1
                up echo nameserver 192.168.0.1 > /etc/resolv.conf

# DHCP config for eth0
# auto eth0
# iface eth0 inet dhcp
```

No special config required on MOSCOWSW01 and MOSCOWSW02 SWITCHES. Just acting as layer 2 only.

MOSCOWR19

```
hostname MOSCOWR19
interface Ethernet0/1
 ip address 172.16.40.1 255.255.255.0
no shutdown
 standby version 2
 standby 1 ip 172.16.40.254
 standby 1 priority 200
!
router ospf 1
 network 172.16.40.0 0.0.0.255 area 0
 network 192.168.32.0 0.0.0.255 area 0
!
end
```

MOSCOWR20

```
hostname MOSCOWR20
interface Ethernet0/1
 ip address 172.16.40.2 255.255.255.0
no shutdown
 standby version 2
 standby 1 ip 172.16.40.254
 standby 1 preempt
!
router ospf 1
 network 172.16.40.0 0.0.0.255 area 0
 network 192.168.32.0 0.0.0.255 area 0
!
end
```

Let’s create some access-lists that match traffic on the control plane that we can use in our class-maps:

MOSCOWR19 (config)#

```
ip access-list extended ICMP
permit icmp any any
ip access-list extended TELNET
permit tcp any any eq 23
ip access-list extended OSPF
permit ospf any any
ip access-list extended HSRP
permit udp any host 224.0.0.102 eq 1985
```

Let’s create class-maps that match the access-lists:

MOSCOWR19(config)#

```
class-map ICMP
match access-group name ICMP
```

```
class-map TELNET
match access-group name TELNET
class-map OSPF
match access-group name OSPF
class-map HSRP
match access-group name HSRP
```

Now I can create a policy-map on **MOSCOWR19**:

MOSCOWR19 (config)#

```
policy-map COPP
class ICMP
police 8000 conform-action transmit exceed-action transmit
exit
class TELNET
police 8000 conform-action transmit exceed-action transmit
exit
class OSPF
police 8000 conform-action transmit exceed-action transmit
exit
class HSRP
police 8000 conform-action transmit exceed-action transmit
exit
```

In the policy-map, I add policers for 8000 bps and the conform-action and exceed-action are both set to transmit. These policers will never drop anything but there is a good reason I configure it like this.

In a production network, you might want to run it like this for a few days to see how much of your traffic is conformed or exceeded.

We need to attach this policy-map to the control plane. We do this with the following command:

MOSCOWR19(config)#

```
control-plane
service-policy input COPP
```

Upon enabling you must information messages stating,

***Jul 16 10:37:08.295: %CP-5-FEATURE: Control-plane Policing feature enabled on Control plane aggregate path**

<u>VALIDATE:</u>

From PC9, ping MOSCOWR19 interface IP:

```
/ # ping 172.16.40.1
PING 172.16.40.1 (172.16.40.1): 56 data bytes
64 bytes from 172.16.40.1: seq=0 ttl=255 time=13.097 ms
64 bytes from 172.16.40.1: seq=1 ttl=255 time=10.012 ms
```

****[NO DROPS SO FAR, AS COPP IS ENABLED JUST TO MONITOR THE TRAFFIC FLOW]**

```
MOSCOWR19#show policy-map control-plane
 Control Plane
  Service-policy input: COPP
    Class-map: ICMP (match-all)
      13 packets, 1326 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: access-group name ICMP
      police:
          cir 8000 bps, bc 1500 bytes
        conformed 13 packets, 1326 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          transmit
        conformed 0000 bps, exceeded 0000 bps

    Class-map: TELNET (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: access-group name TELNET
      police:
          cir 8000 bps, bc 1500 bytes
        conformed 0 packets, 0 bytes; actions:
 --More--
*Jul 16 10:39:47.017: %SYS-5-CONFIG_I: Configured from console by console
          transmit
        exceeded 0 packets, 0 bytes; actions:
          transmit
        conformed 0000 bps, exceeded 0000 bps

    Class-map: OSPF (match-all)
      34 packets, 3264 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: access-group name OSPF
      police:
          cir 8000 bps, bc 1500 bytes
        conformed 34 packets, 3264 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          transmit
        conformed 0000 bps, exceeded 0000 bps

    Class-map: HSRP (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: access-group name HSRP
      police:
          cir 8000 bps, bc 1500 bytes
        conformed 0 packets, 0 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          transmit
        conformed 0000 bps, exceeded 0000 bps

    Class-map: class-default (match-any)
      249 packets, 16626 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: any
MOSCOWR19#
```

****[NO DROPS SO FAR, AS COPP IS ENABLED JUST TO MONITOR THE TRAFFIC FLOW]**

Our policy-map is up and running **but even exceeding traffic is permitted**. Let's change the exceed action for our ICMP traffic:

Let's **add a policer to the ICMP class-map to rate-limit ICMP traffic** for testing purpose:

MOSCOWR19 (config)#

```
policy-map COPP
class ICMP
police cir 8000 conform-action transmit exceed-action drop
```

**If you want to rate-limit other traffic like OSPF, TELNET, SSH, you can do it. I have only done ICMP for testing the CoPP feature.

VALIDATE:
Test by initiating PING from PC9 to R1:

```
/ # ping 172.16.40.1 -s 1024 -c 50
1032 bytes from 172.16.40.1: seq=15 ttl=255 time=7.110 ms
1032 bytes from 172.16.40.1: seq=16 ttl=255 time=6.585 ms
1032 bytes from 172.16.40.1: seq=18 ttl=255 time=4.051 ms
1032 bytes from 172.16.40.1: seq=19 ttl=255 time=15.792 ms
1032 bytes from 172.16.40.1: seq=20 ttl=255 time=12.318 ms
^C
--- 172.16.40.1 ping statistics ---
21 packets transmitted, 19 packets received, 9% packet loss

MOSCOWR19#show policy-map control-plane
 Control Plane
  Service-policy input: COPP
    Class-map: ICMP (match-all)
      282 packets, 66516 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: access-group name ICMP
      police:
          cir 8000 bps, bc 1500 bytes
        conformed 278 packets, 62236 bytes; actions:
          transmit
        exceeded 4 packets, 4280 bytes; actions:
```

```
      drop
     conformed 0000 bps, exceeded 0000 bps

  Class-map: TELNET (match-all)
   0 packets, 0 bytes
   5 minute offered rate 0000 bps, drop rate 0000 bps
   Match: access-group name TELNET
   police:
      cir 8000 bps, bc 1500 bytes
     conformed 0 packets, 0 bytes; actions:
      transmit
     exceeded 0 packets, 0 bytes; actions:
      transmit
     conformed 0000 bps, exceeded 0000 bps

  Class-map: OSPF (match-all)
   108 packets, 10368 bytes
   5 minute offered rate 0000 bps, drop rate 0000 bps
   Match: access-group name OSPF
   police:
      cir 8000 bps, bc 1500 bytes
     conformed 108 packets, 10368 bytes; actions:
      transmit
     exceeded 0 packets, 0 bytes; actions:
      transmit
     conformed 0000 bps, exceeded 0000 bps

  Class-map: HSRP (match-all)
   0 packets, 0 bytes
   5 minute offered rate 0000 bps, drop rate 0000 bps
   Match: access-group name HSRP
   police:
      cir 8000 bps, bc 1500 bytes
     conformed 0 packets, 0 bytes; actions:
      transmit
     exceeded 0 packets, 0 bytes; actions:
      transmit
     conformed 0000 bps, exceeded 0000 bps

  Class-map: class-default (match-any)
   793 packets, 53744 bytes
   5 minute offered rate 0000 bps, drop rate 0000 bps
   Match: any
```

BENEFITS OF CoPP:

CoPP (Control Plane Policing) is sued to rate limit packets to and from the route processor on the control plane.

- For classification, use:
    - standard or extended access-lists
    - DSCP or IP precedence values
    - Don’t use NBAR except to match ARP packets.
- It’s best to set the conform-action and exceed-action both to transmit, so you don’t drop legitimate traffic. Once you know how much packets are exceeding, change the values and exceed action to drop.

IPSEC SITE-TO-SITE TUNNEL

IPsec (Internet Protocol Security) is a framework that helps us to protect IP traffic on the network layer. Why? because the IP protocol itself doesn't have any security features at all. IPsec can protect our traffic with the following features:

- **Confidentiality**: by encrypting our data, nobody except the sender and receiver will be able to read our data.
- **Integrity**: we want to make sure that nobody changes the data in our packets. By calculating a hash value, the sender and receiver will be able to check if changes have been made to the packet.
- **Authentication**: the sender and receiver will authenticate each other to make sure that we are really talking with the device we intend to.

Overview of VPN Technologies

Cisco supports several types of VPN implementations on the ASA/Routers but they are generally categorized as either "**IPSec Based VPNs**" or "**SSL Based VPNs**".

The first category uses the IPSec protocol for secure communications while the second category uses SSL.

SSL Based VPNs are also called WebVPN in Cisco terminology and will be discussed in the next Chapter when we talk about the Anyconnect VPN client solution.

IPSec Based VPNs:

1. **Site-to-Site IPSec VPN:** Used to connect two or more remote LAN networks over unsecure media (e.g Internet). It runs between ASA-to-ASA or ASA-to-Cisco Router.

2. **Remote Access with IPSec VPN Client:** A VPN client software is installed on user's PC to provide remote access to the central network. It uses the IPSec protocol and provides full network connectivity to the remote user. The users use their applications at the central site as they normally would without a VPN in place.

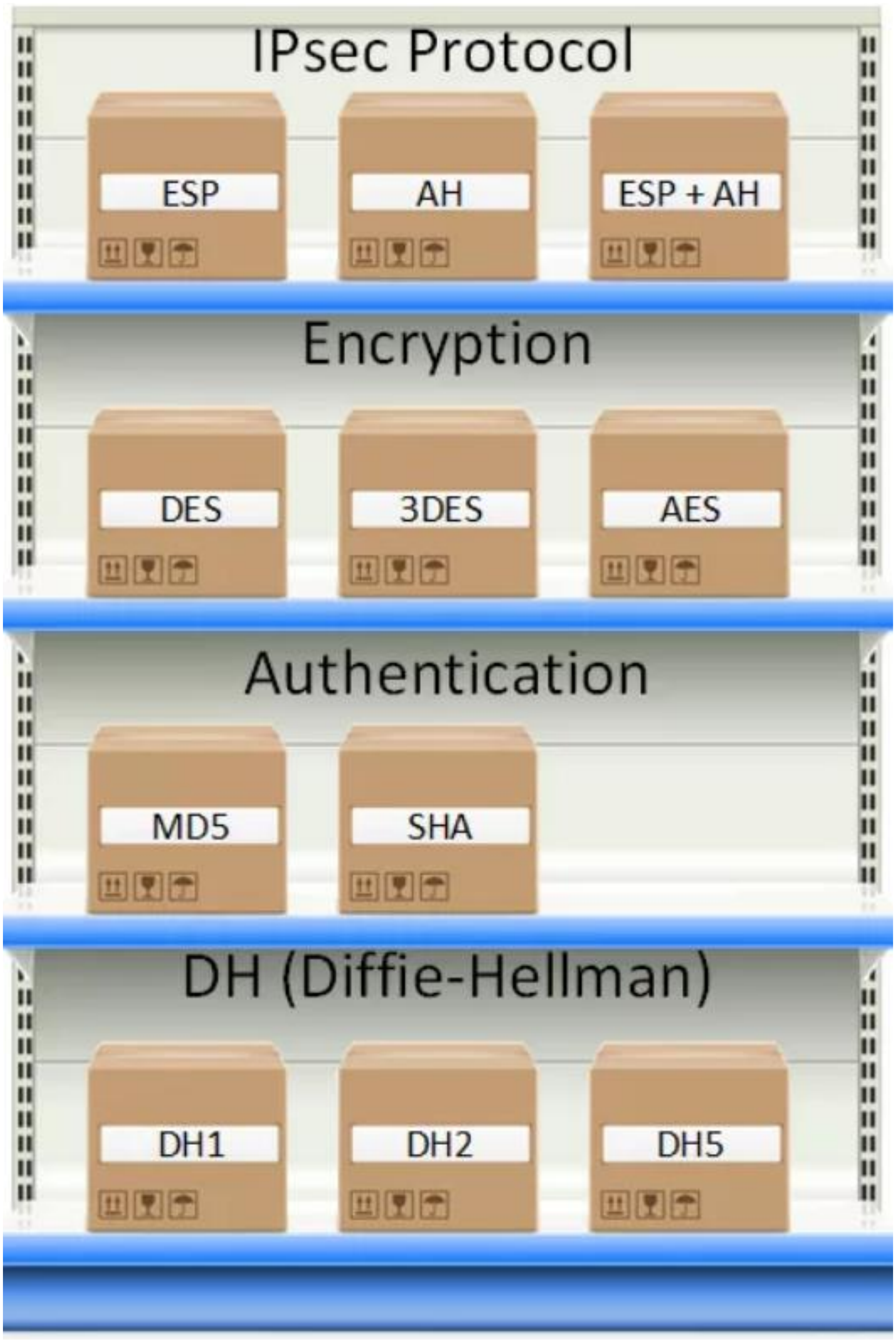
NOTE: Cisco has announced the End-of-Life of the Legacy Cisco IPSec VPN client. It is now replaced by the "Cisco Anyconnect Secure Mobility Client" which provides both secure SSL and IPSec/IKEv2 connections to the ASA for remote users.

SSL Based VPNs (WebVPN):

1. **Clientless Mode WebVPN:** This is the first implementation of SSL WebVPN supported from ASA version 7.0 and later. It lets users establish a secure remote access VPN tunnel using just a Web browser. There is no need for a software or hardware VPN client. However, only limited applications can be accessed remotely.

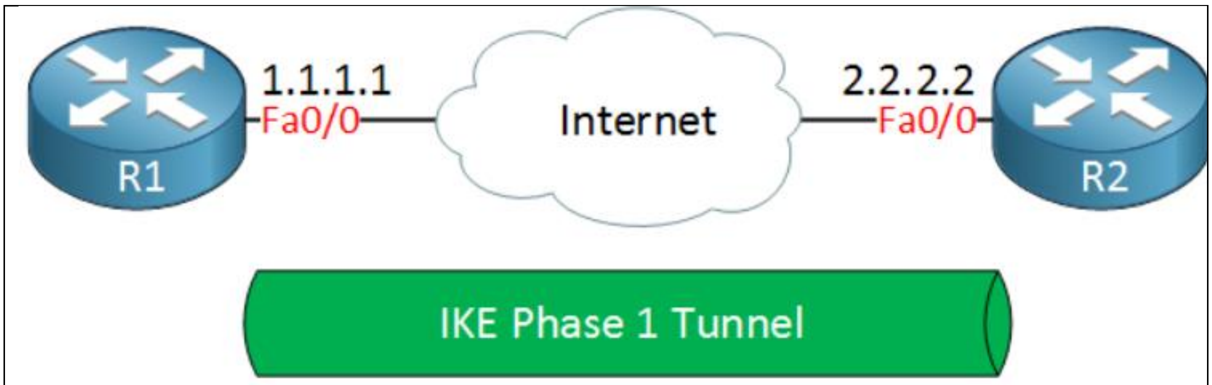
2. **AnyConnect VPN:** A special Java based client is installed on the user's computer providing an SSL secure tunnel to the central site. Anyconnect provides full network connectivity (similar with IPSec remote access client). All applications at the central site can be accessed remotely. Also, in the newest Anyconnect versions (3.x and above), the client supports also IKEv2 IPSEC to offer remote access.

I will show you how to configure two Cisco IOS routers to use IPSec in Tunnel mode. This means that the original IP packet will be encapsulated in a new IP packet and encrypted before it is sent out of the network.

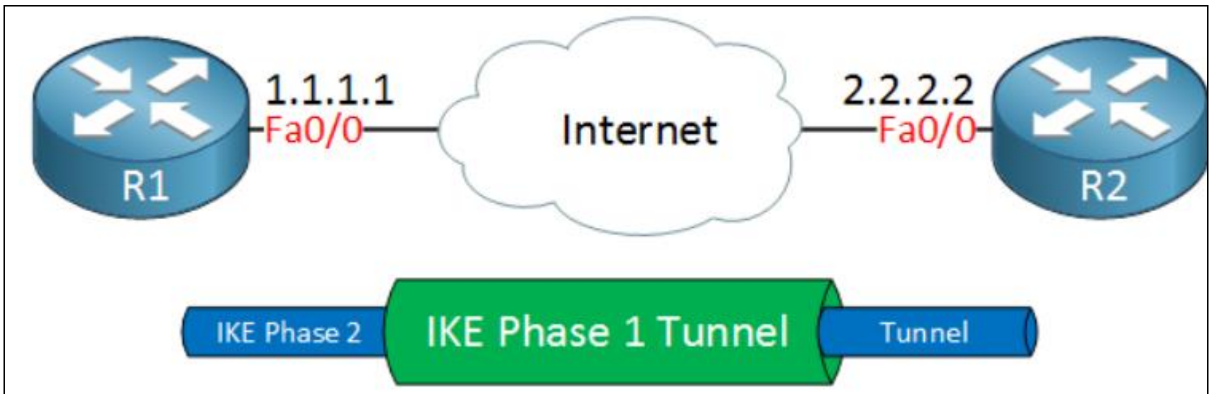


There are **two phases** to build an IPsec tunnel:

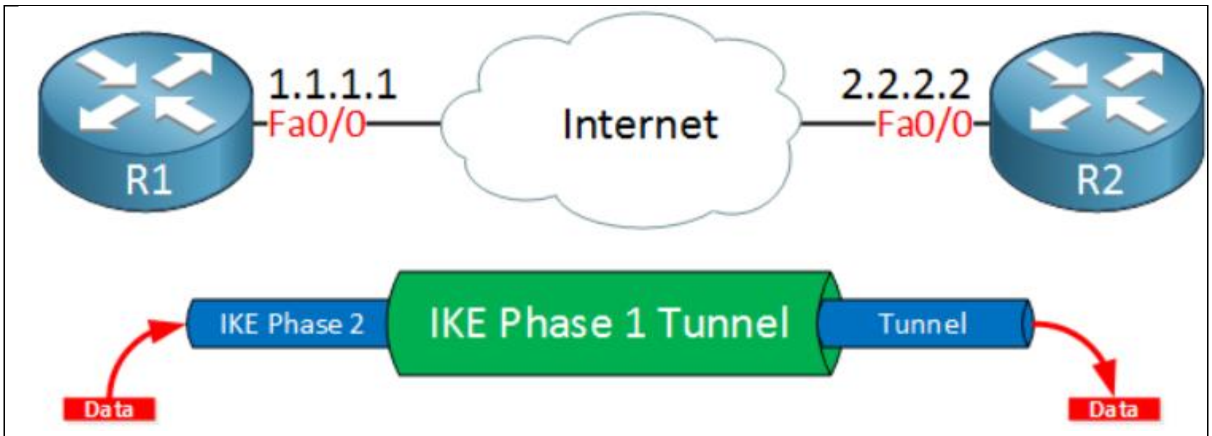
- **IKE phase 1**
- **IKE phase 2**



The IKE phase 1 tunnel is only used for **management traffic**. We use this tunnel as a *secure method to establish the second tunnel* called the **IKE phase 2 tunnel** or **IPsec tunnel** and for management traffic like keepalives.



Once IKE phase 2 is completed, we have an IKE phase 2 tunnel (or IPsec tunnel) that we can use to protect our user data. This user data will be sent through the IKE phase 2 tunnel:



IKE builds the tunnels for us but it doesn't authenticate or encrypt user data. We use two other protocols for this:

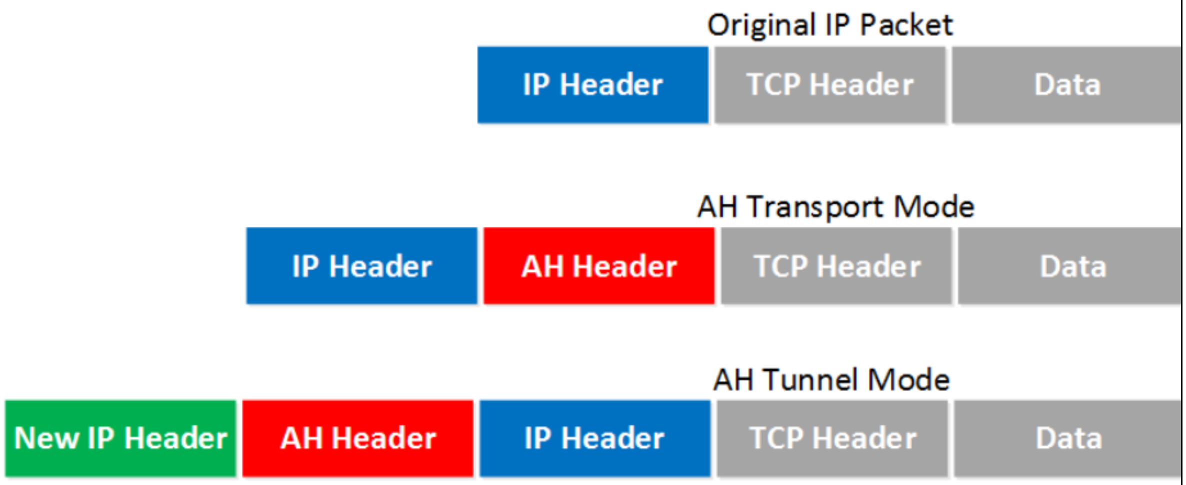
- **AH (Authentication Header)**
- **ESP (Encapsulating Security Payload)**

AH and ESP both offer authentication and integrity but only **ESP supports encryption**. Because of this, ESP is the most popular choice nowadays.

Both protocols support two different modes:

- **Transport mode**
- **Tunnel mode**

The main difference between the two is that with transport mode we will use the **original IP header** while in tunnel mode, we use a new **IP header**. Here's an example to help you visualize this:



Transport mode is often between two devices that want to protect some insecure traffic (example: telnet traffic). Tunnel mode is typically used for site-to-site VPNs where we need to encapsulate the original IP packet since these are mostly private IP addresses and can't be routed on the Internet. I will explain these two modes in detail later in this lesson.

The entire process of IPsec consists of five steps:

- **Initiation**: something has to trigger the creation of our tunnels. For example, when you configure IPsec on a router, you use an access-list to tell the router what data to protect. When the router receives something that matches the access-list, it will start the IKE process. It's also possible to manually initiate the tunnel.
- **IKE phase 1**: we negotiate a security association to build the IKE phase 1 tunnel (ISAKMP tunnel).
- **IKE phase 2**: within the IKE phase 1 tunnel, we build the IKE phase 2 tunnel (IPsec tunnel).

- **Data transfer**: we protect user data by sending it through the IKE phase 2 tunnel.
- **Termination**: when there is no user data to protect then the IPsec tunnel will be terminated after a while.

PARAMETERS REQUIRED FOR ISAKMP (PHASE I):

- **Hashing**: we use a hashing algorithm to verify the integrity, we use MD5 or SHA for this.
- **Authentication**: each peer has to prove who he is. Two commonly used options are a pre-shared key or digital certificates.
- **DH (Diffie Hellman) group**: the DH group determines the strength of the key that is used in the key exchange process. The higher group numbers are more secure but take longer to compute.
- **Lifetime**: how long does the IKE phase 1 tunnel stand up? the shorter the lifetime, the more secure it is because rebuilding it means we will also use new keying material. Each vendor uses a different lifetime, a common default value is 86400 seconds (1 day).
- **Encryption**: what algorithm do we use for encryption? For example, DES, 3DES or AES.

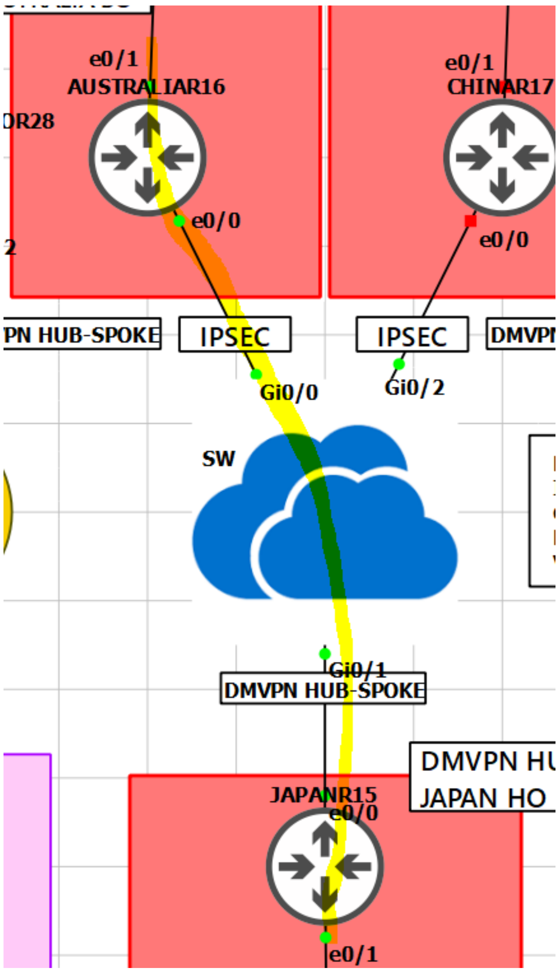
PARAMETERS REQUIRED FOR IPSEC (PHASE II):

Just like in IKE phase 1, our peers will negotiate about a number of items:

- **IPsec Protocol**: do we use AH or ESP?
- **Encapsulation Mode**: transport or tunnel mode?
- **Encryption**: what encryption algorithm do we use? DES, 3DES or AES?
- **Authentication**: what authentication algorithm do we use? MD5 or SHA?
- **Lifetime**: how long is the IKE phase 2 tunnel valid? When the tunnel is about to expire, we will refresh the keying material.
- **(Optional) DH exchange**: used for PFS (Perfect Forward Secrecy).

PFS is optional and forces the peers to run the DH exchange again to generate a new shared key in each IKE phase 2 quick mode.

GNS3 Lab:



Let's start configuring Phase 1 on both routers:

```
JAPANR15 (config)#
crypto isakmp policy 10
hash md5
authentication pre-share
group 2
encryption 3des
exit
```

The IP address of a loopback interface can be used when there are multiple paths to reach the peer's IP address:

```
crypto isakmp key cisco address 12.1.1.2
```

To configure the Phase 2, we need to define the transform-set, which specifies the hashing, the security protocol, and the encryption used for Phase 2:

```
crypto ipsec transform-set TSET esp-3des esp-md5-hmac
```

Next, we need to define the crypto ACL/proxy ID, which defines the interesting traffic:

```
access-list 100 permit ip host 1.1.1.1 host 2.2.2.2
```

In the last step, a crypto map is configured to specify the peer, crypto ACL, and the transform set. There are three choices when configuring the following crypto map:

- IPSec-ISAKMP: This is the best option. It states that we are using ISAKMP to encrypt and decrypt the key.
- IPSec-manual: This is the worst choice. It means that the key needs to be entered manually. (Can you imagine entering a 512-bit key manually?)
- GDOI: This choice is used for GETVPN configuration. It stands for group domain of interpretation.

```
crypto map TST 10 ipsec-isakmp
set peer 12.1.1.2
match address 100
set transform-set TSET
```

The final step applies the crypto map to the interface facing the other peer:

```
interface Eth 0/0
crypto map TST
```

Static Route required for bringing up VPN for testing purpose:

```
ip route 2.2.2.2 255.255.255.255 Ethernet0/0
```

Create Loopback 0

```
Interface loopback 0
Ip add 1.1.1.1 255.255.255.0
No shut
```

Configure e0/0:

```
JAPANR15 (config)#
Interface ethernet 0/0
Ip add 12.1.1.1 255.255.255.0
No shut
```

AUSTRALIAR16(config)#

```
crypto isakmp policy 10
hash md5
authentication pre-share
group 2
```

```
encryption 3des
exit
crypto isakmp key cisco address 12.1.1.1

crypto ipsec transform-set TSET esp-3des esp-md5-hmac

access-list 100 permit ip host 2.2.2.2 host 1.1.1.1

crypto map TST 10 ipsec-isakmp
set peer 12.1.1.1
match address 100
set transform-set TSET
exit

interface Eth 0/0
crypto map TST

ip route 1.1.1.1 255.255.255.255 Ethernet0/0

interface loopback 0
ip address 2.2.2.2 255.255.255.0
no shutdown
```

Let’s verify the configuration before testing:

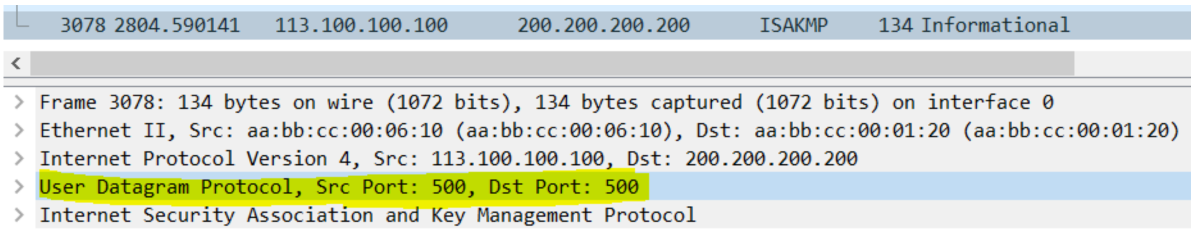
VERIFICATIONS:

ping 2.2.2.2 source loopback0

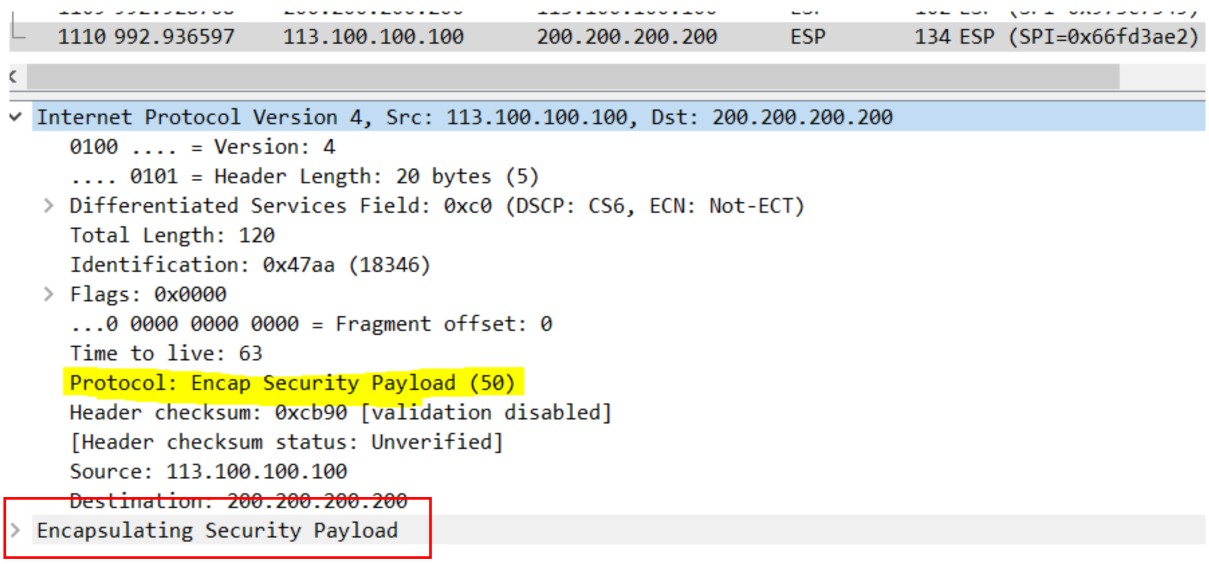
```
R15#ping 2.2.2.2 source loopback0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 1.1.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 14/17/23 ms

R15#sh cry isa sa det
Codes: C - IKE configuration mode, D - Dead Peer Detection
    K - Keepalives, N - NAT-traversal
    T - cTCP encapsulation, X - IKE Extended Authentication
    psk - Preshared key, rsig - RSA signature
    renc - RSA encryption
IPv4 Crypto ISAKMP SA

C-id  Local          Remote          I-VRF  Status Encr Hash   Auth DH Lifetime Cap.

1001  12.1.1.1       12.1.1.2               ACTIVE 3des md5    psk  2  22:56:36
    Engine-id:Conn-id =  SW:1

IPv6 Crypto ISAKMP SA
```

```
R15#sh cry ipsec sa peer 12.1.1.2 | i incr|dec
    #pkts decaps: 10, #pkts decrypt: 10, #pkts verify: 10
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
```

WIRESHARK CAPTURES:



```
3078 2804.590141   113.100.100.100      200.200.200.200      ISAKMP    134 Informational
> Frame 3078: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on interface 0
> Ethernet II, Src: aa:bb:cc:00:06:10 (aa:bb:cc:00:06:10), Dst: aa:bb:cc:00:01:20 (aa:bb:cc:00:01:20)
> Internet Protocol Version 4, Src: 113.100.100.100, Dst: 200.200.200.200
> User Datagram Protocol, Src Port: 500, Dst Port: 500
> Internet Security Association and Key Management Protocol
```



```
1110 992.936597    113.100.100.100      200.200.200.200      ESP        134 ESP (SPI=0x66fd3ae2)
v Internet Protocol Version 4, Src: 113.100.100.100, Dst: 200.200.200.200
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
    Total Length: 120
    Identification: 0x47aa (18346)
  > Flags: 0x0000
    ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 63
    Protocol: Encap Security Payload (50)
    Header checksum: 0xcb90 [validation disabled]
    [Header checksum status: Unverified]
    Source: 113.100.100.100
    Destination: 200.200.200.200
> Encapsulating Security Payload
```

GRE TUNNEL

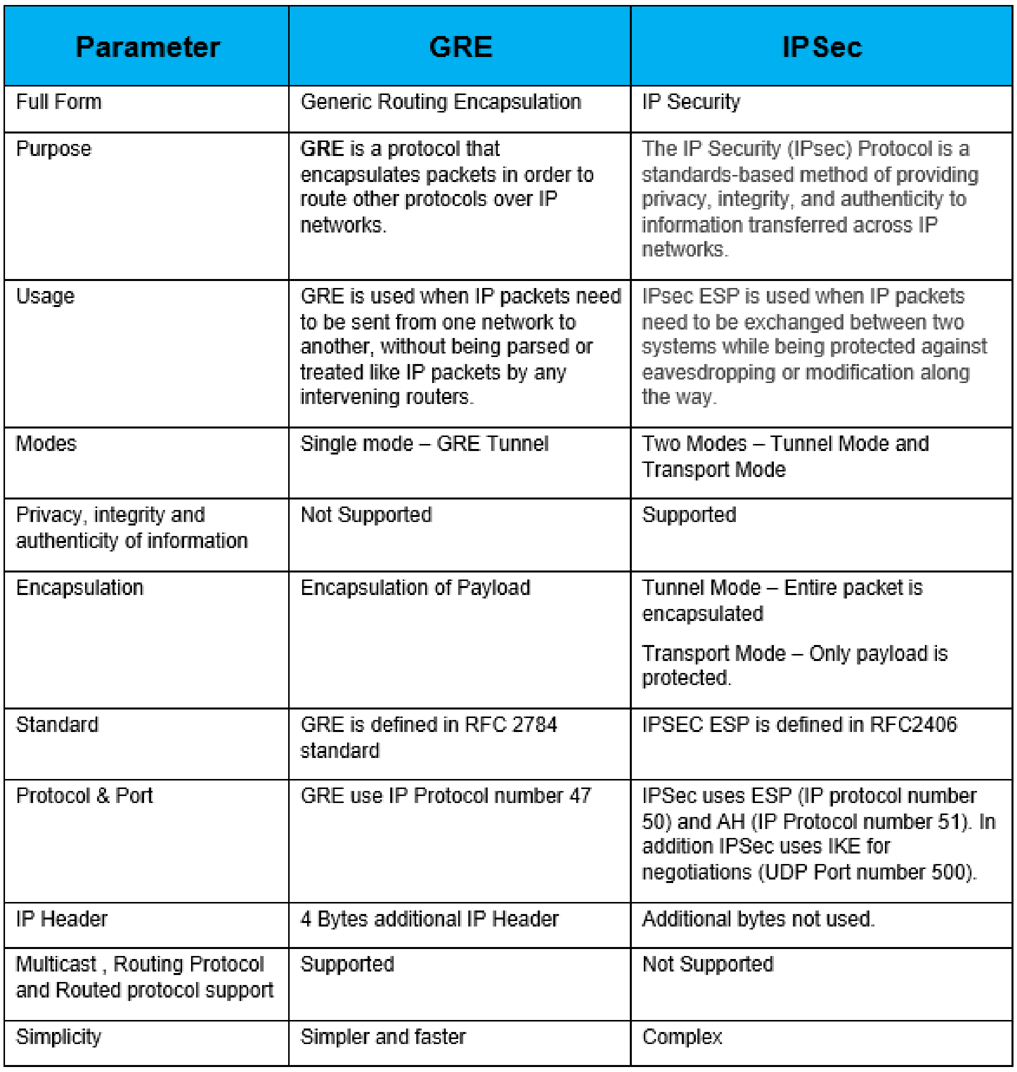
Tunnelling is a concept where we put ‘packets into packets’ so that they can be transported over certain networks. We also call this **encapsulation**.

A good example is when you have two sites with IPv6 addresses on their LAN but they are only connected to the Internet with IPv4 addresses. Normally it would be impossible for the two IPv6 LANs to reach each other but by using tunnelling the two routers will put IPv6 packets into IPv4 packets so that our IPv6 traffic can be routed on the Internet.

Another example is where we have an HQ and a branch site and you want to run a routing protocol like RIP, OSPF or EIGRP between them. We can tunnel these routing protocols so that the HQ and branch router can exchange routing information.

Basically, when you configure a tunnel, it’s like you create a **point-to-point connection** between the two devices. GRE (Generic Routing Encapsulation) is a simple tunnelling technique that can do this for us.

IPSec vs GRE TUNNEL COMPARISON:

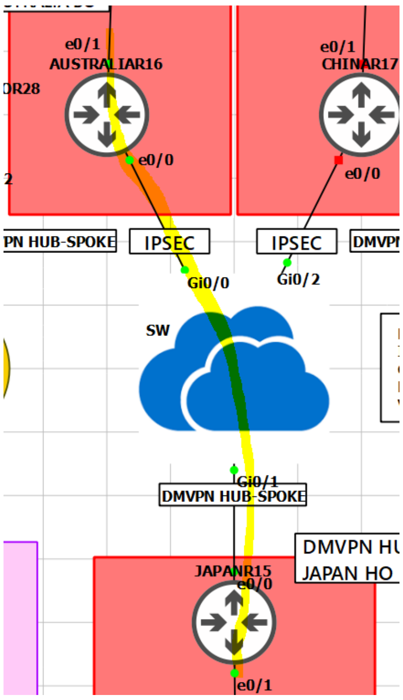


Parameter	GRE	IPSec
Full Form	Generic Routing Encapsulation	IP Security
Purpose	GRE is a protocol that encapsulates packets in order to route other protocols over IP networks.	The IP Security (IPsec) Protocol is a standards-based method of providing privacy, integrity, and authenticity to information transferred across IP networks.
Usage	GRE is used when IP packets need to be sent from one network to another, without being parsed or treated like IP packets by any intervening routers.	IPsec ESP is used when IP packets need to be exchanged between two systems while being protected against eavesdropping or modification along the way.
Modes	Single mode – GRE Tunnel	Two Modes – Tunnel Mode and Transport Mode
Privacy, integrity and authenticity of information	Not Supported	Supported
Encapsulation	Encapsulation of Payload	Tunnel Mode – Entire packet is encapsulated Transport Mode – Only payload is protected.
Standard	GRE is defined in RFC 2784 standard	IPSEC ESP is defined in RFC2406
Protocol & Port	GRE use IP Protocol number 47	IPSec uses ESP (IP protocol number 50) and AH (IP Protocol number 51). In addition IPSec uses IKE for negotiations (UDP Port number 500).
IP Header	4 Bytes additional IP Header	Additional bytes not used.
Multicast , Routing Protocol and Routed protocol support	Supported	Not Supported
Simplicity	Simpler and faster	Complex

Examples for Routed Protocol: routed protocols are IPv4, IPv6 and AppleTalk.
Examples for Routing Protocol: RIP, EIGRP, OSPF, BGP

GRE does not provide any **encryption type service**.
If security is required, **added using IPSec.**

GNS3

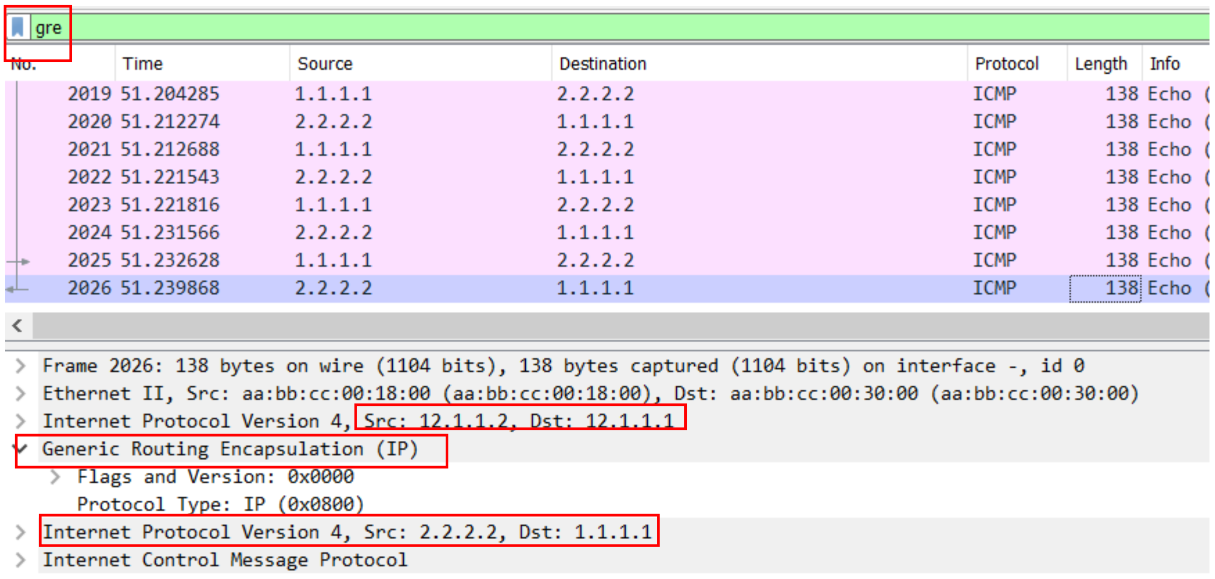


```
JAPANR15 (config)#
interface eth 0/0
ip add 12.1.1.1 255.255.255.0
no shut
inter loop 0
ip add 1.1.1.1 255.255.255.0
no shut

interface tunnel 0
ip address 4.4.4.4 255.255.255.0
tunnel source 12.1.1.1
tunnel destination 12.1.1.2
no shutdown

ip route 2.2.2.2 255.255.255.255 tunnel0
```

```
AUSTRALIAR16(config)#
interface eth 0/0
ip add 12.1.1.2 255.255.255.0
no shut
inter loop 0
ip add 2.2.2.2 255.255.255.0
no shut

interface tunnel 0
ip address 4.4.4.5 255.255.255.0
tunnel source 12.1.1.2
tunnel destination 12.1.1.1
no shutdown

ip route 1.1.1.1 255.255.255.255 tunnel0
```

VALIDATION:

```
JAPANR15 #ping 2.2.2.2 so lo 0 repeat 1000
Type escape sequence to abort.
Sending 1000, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 1.1.1.1
!!!!

AUSTRALIAR16#show interfaces tunnel 0 | i tx|packet
     reliability 255/255, txload 2/255, rxload 2/255
    Checksumming of packets disabled
  5 minute input rate 1000 bits/sec, 1 packets/sec
  5 minute output rate 1000 bits/sec, 1 packets/sec
     1010 packets input, 125240 bytes, 0 no buffer
     1010 packets output, 125240 bytes, 0 underruns

AUSTRALIAR16#show interfaces tunnel 0
Tunnel0 is up, line protocol is up
  Hardware is Tunnel
  Internet address is 4.4.4.5/24
  MTU 17916 bytes, BW 100 Kbit/sec, DLY 50000 usec,
     reliability 255/255, txload 7/255, rxload 7/255
  Encapsulation TUNNEL, loopback not set
  Keepalive not set
  Tunnel linestate evaluation up
  Tunnel source 12.1.1.2, destination 12.1.1.1
  Tunnel protocol/transport GRE/IP
<!ouput omitted--!>
```



```
gre
No.   Time        Source     Destination   Protocol  Length Info
2019 51.204285   1.1.1.1    2.2.2.2       ICMP      138 Echo (
2020 51.212274   2.2.2.2    1.1.1.1       ICMP      138 Echo (
2021 51.212688   1.1.1.1    2.2.2.2       ICMP      138 Echo (
2022 51.221543   2.2.2.2    1.1.1.1       ICMP      138 Echo (
2023 51.221816   1.1.1.1    2.2.2.2       ICMP      138 Echo (
2024 51.231566   2.2.2.2    1.1.1.1       ICMP      138 Echo (
2025 51.232628   1.1.1.1    2.2.2.2       ICMP      138 Echo (
2026 51.239868   2.2.2.2    1.1.1.1       ICMP      138 Echo (

> Frame 2026: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface -, id 0
> Ethernet II, Src: aa:bb:cc:00:18:00 (aa:bb:cc:00:18:00), Dst: aa:bb:cc:00:30:00 (aa:bb:cc:00:30:00)
> Internet Protocol Version 4, Src: 12.1.1.2, Dst: 12.1.1.1
v Generic Routing Encapsulation (IP)
   > Flags and Version: 0x0000
     Protocol Type: IP (0x0800)
> Internet Protocol Version 4, Src: 2.2.2.2, Dst: 1.1.1.1
> Internet Control Message Protocol
```

Take a close look at the source and destination IP addresses. You can see the packet between 12.1.1.2 and 12.1.1.1 and inside you will find the IP packet between 2.2.2.2 and 1.1.1.1

DMVPN TUNNEL

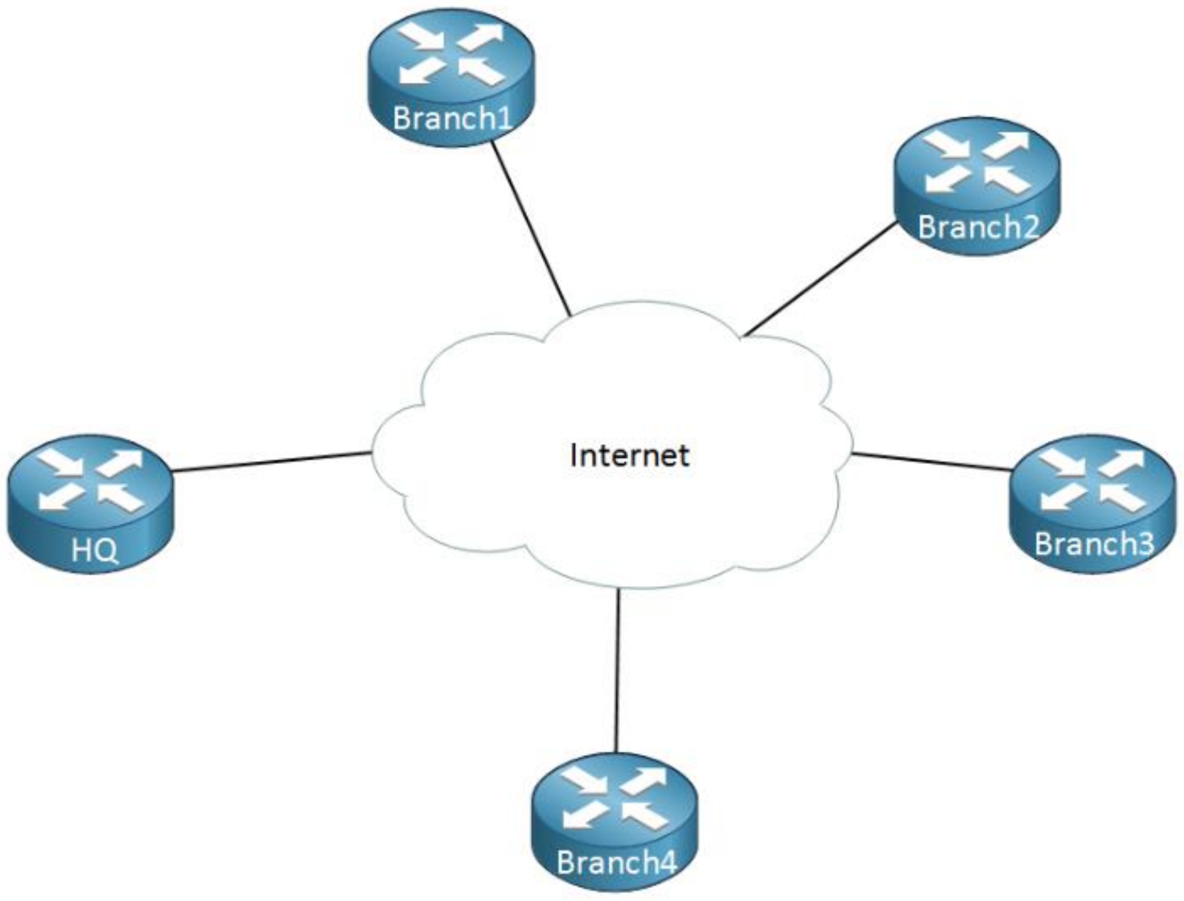
DMVPN (Dynamic Multipoint VPN) is a routing technique we can use to build a VPN network with multiple sites without having to statically configure all devices. It's a "hub and spoke" network where the spokes will be able to communicate with each other directly without having to go through the hub. Encryption is supported through IPsec which makes DMVPN a popular choice for connecting different sites using regular Internet connections. It's a great backup or alternative to private networks like MPLS VPN.

There are four pieces to the DMVPN puzzle:

- Multipoint GRE (mGRE)
- NHRP (Next Hop Resolution Protocol)
- Routing (RIP, EIGRP, OSPF, BGP, etc.)
- IPsec (not required but recommended)

1. Multipoint GRE (mGRE)

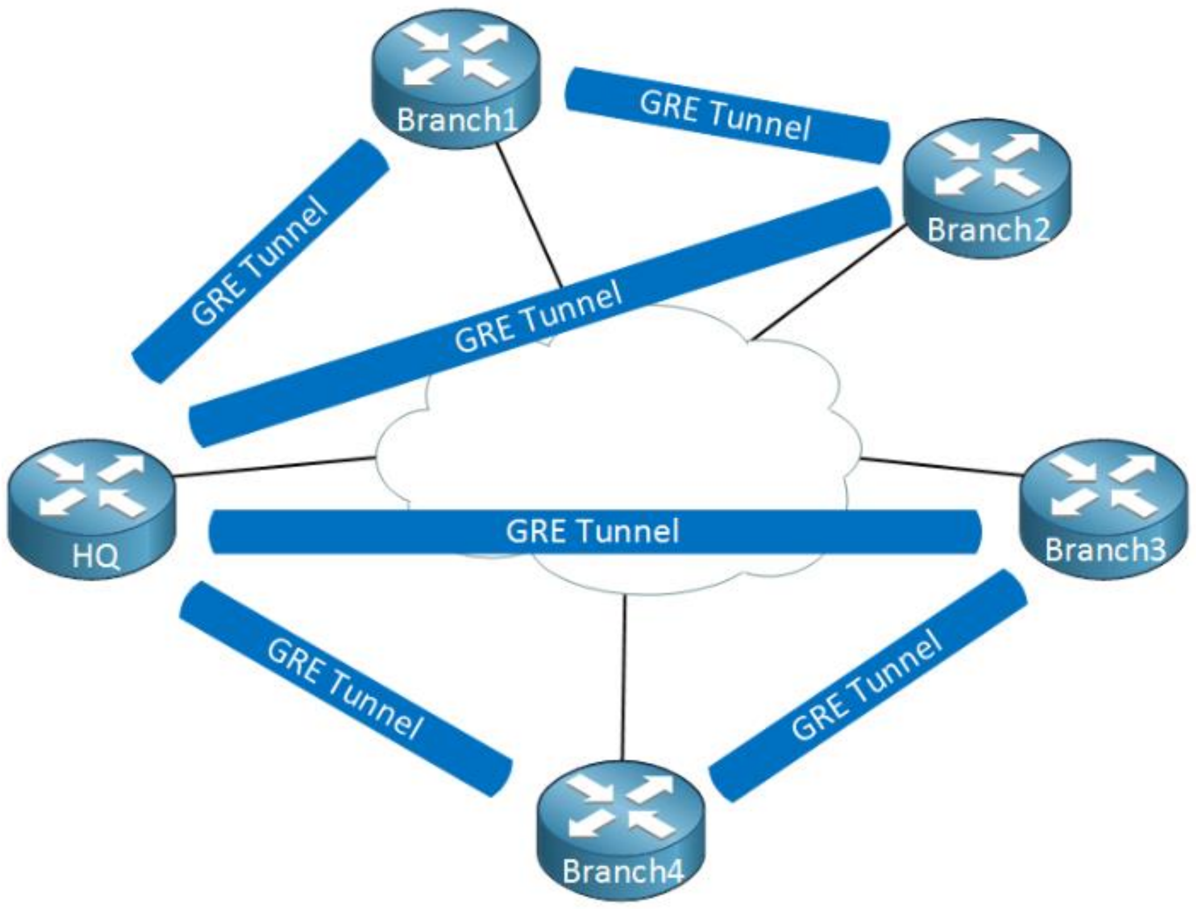
Our "regular" GRE tunnels are point-to-point and don't scale well. For example, let's say we have a company network with some sites that we want to connect to each other using regular Internet connections:



Above we have one router that represents the HQ and there are four branch offices. Let's say that we have the following requirements:

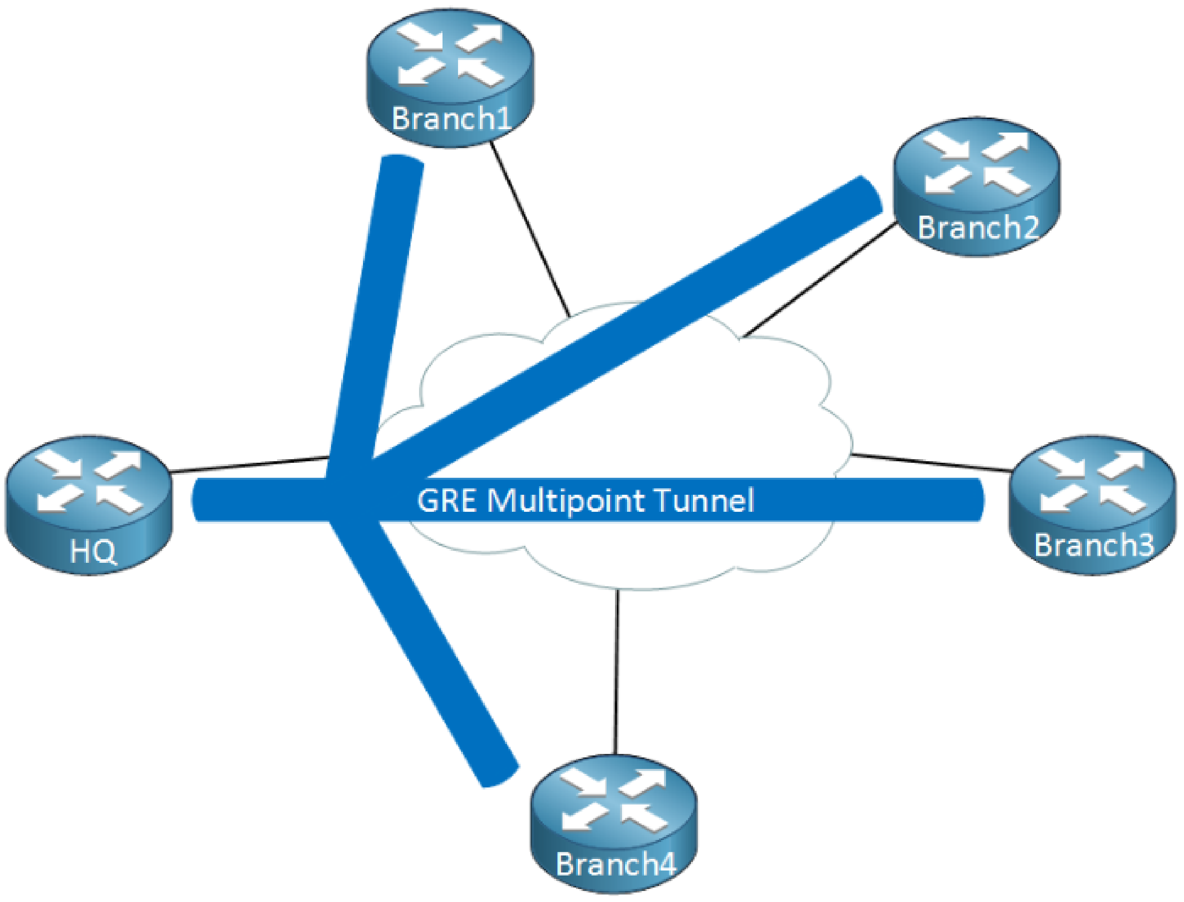
- Each branch office has to be connected to the HQ.
- Traffic between Branch 1 and Branch 2 has to be tunneled directly.
- Traffic between Branch 3 and Branch 4 has to be tunneled directly.

To accomplish this we will have to configure a bunch of GRE tunnels which will look like this:



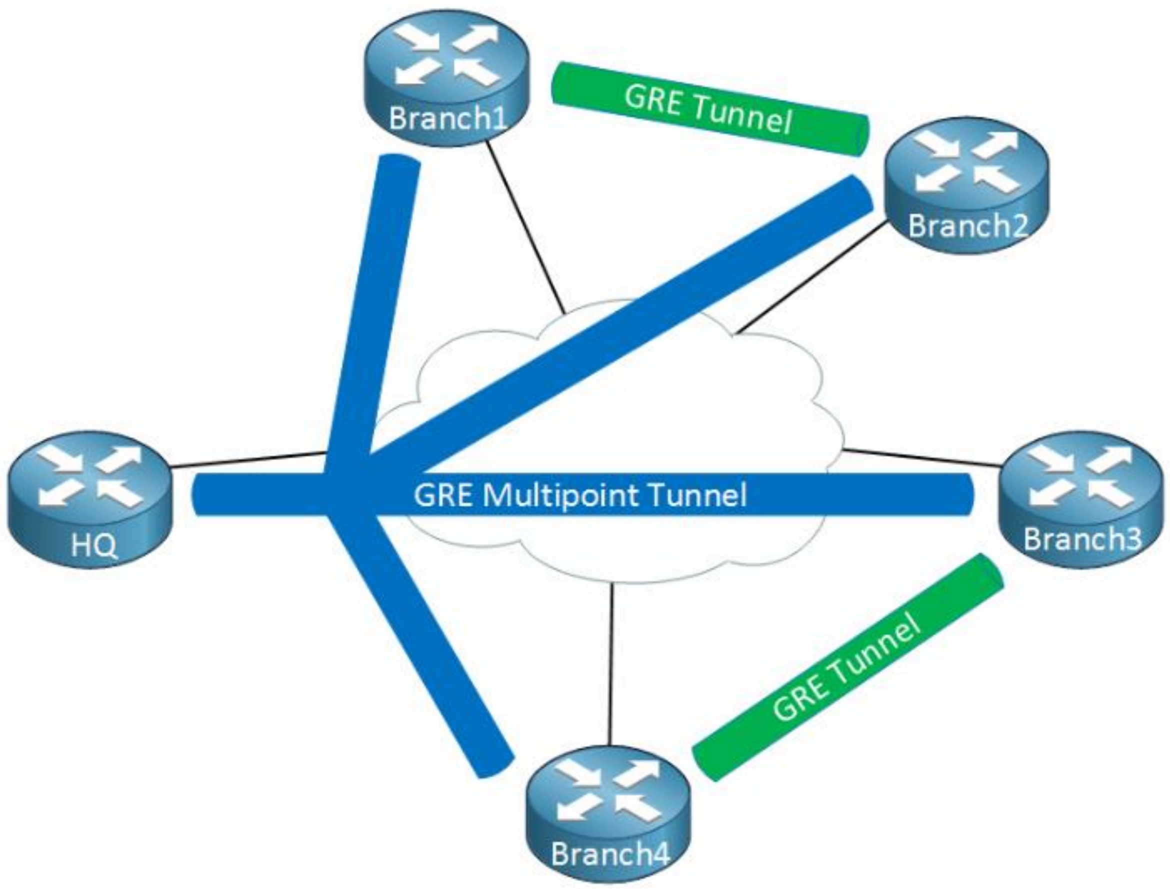
Thing will get messy quickly…we have to create multiple tunnel interfaces, set the source/destination IP addresses etc. It will work but it's not a very scalable solution.

Multipoint GRE, as the name implies allows us to have **multiple destinations**. When we use them, our picture could look like this:



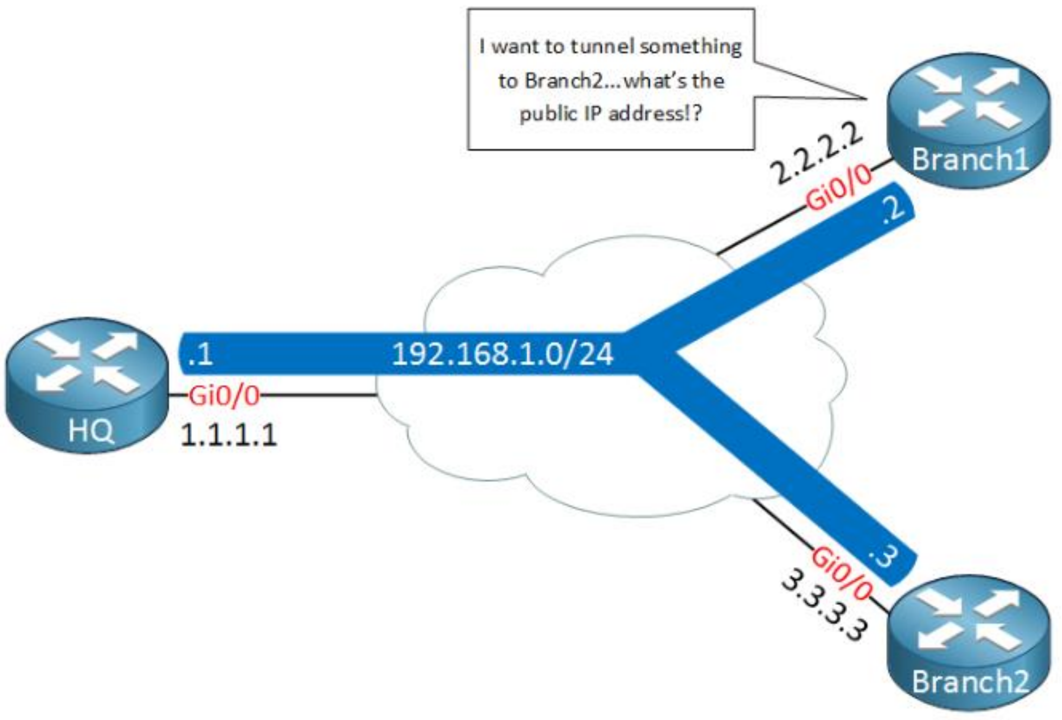
When we use GRE Multipoint, there will be only **one tunnel interface** on each router. The HQ for example has one tunnel with each branch office as its destination. Now you might be wondering, what about the requirement where branch office 1/2 and branch office 3/4 have a direct tunnel?

Right now we have a hub and spoke topology. The cool thing about DMVPN is that we use multipoint GRE so we can have multiple destinations. When we need to tunnel something between branch office 1/2 or 3/4, we automatically "build" new tunnels:



When there is traffic between the branch offices, we can tunnel it directly instead of sending it through the HQ router. This sounds pretty cool but it introduces some problems…

When we configure point-to-point GRE tunnels, we have to configure a source and destination IP address that are used to build the GRE tunnel. When two branch routers want to tunnel some traffic, how do they know what IP addresses to use? Let me show you what I'm talking about:



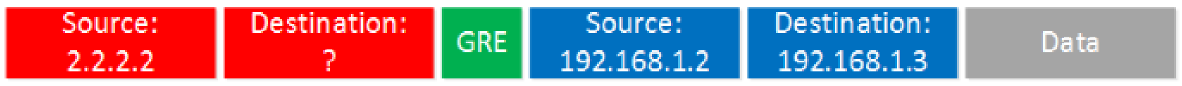
Above we have our HQ and two branch routers, branch1 and branch2. Each router is connected to the Internet and has a public IP address:

- HQ: 1.1.1.1
- Branch1: 2.2.2.2
- Branch2: 3.3.3.3

On the GRE multipoint tunnel interface, we use a single subnet with the following private IP addresses:

- HQ: 192.168.1.1
- Branch1: 192.168.1.2
- Branch2: 192.168.1.3

Let's say that we want to send a ping from branch1's tunnel interface to the tunnel interface of branch2. Here's what the GRE encapsulated IP packet will look like:



The "inner" source and destination IP addresses are known to use, these are the IP address of the tunnel interfaces. We encapsulate this IP packet, put a GRE header in front of it and then we have to fill in the "outer" source and destination IP addresses so that this packet can be routed on the Internet. The branch1 router knows its own public IP address but it has no clue what the public IP address of branch2 is…

To fix this problem, we need some help from another protocol…

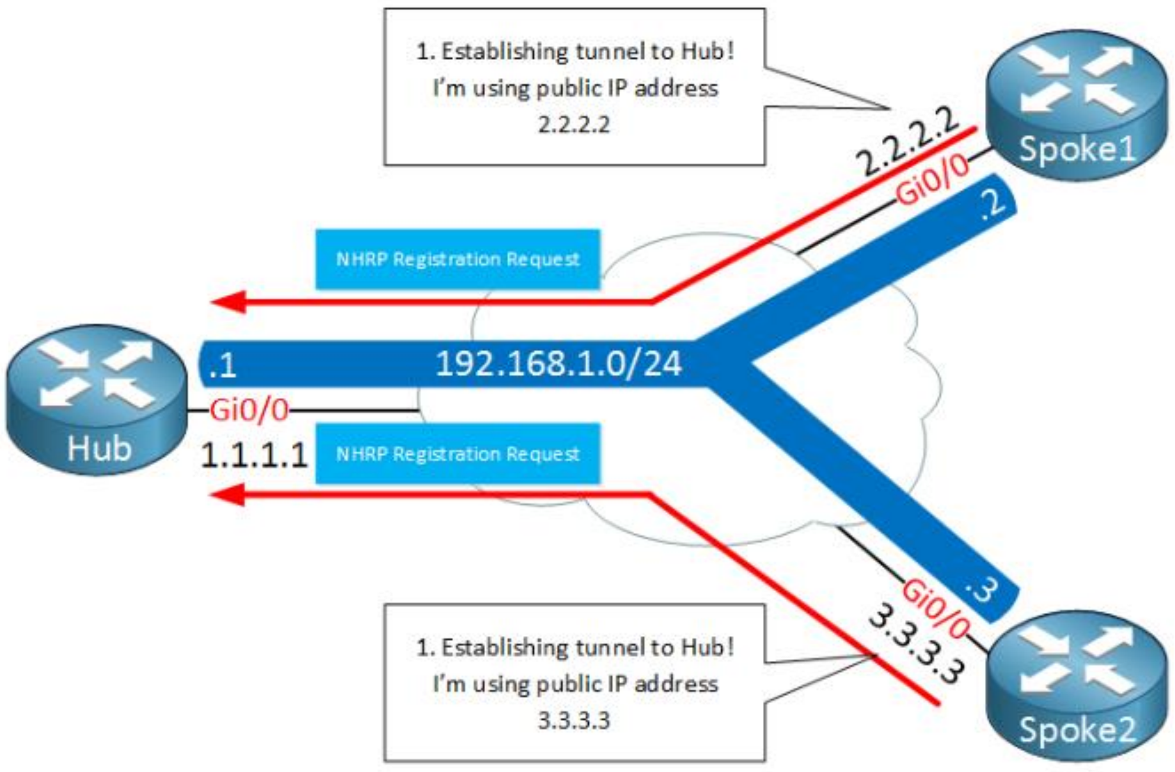
2. NHRP (Next Hop Resolution Protocol)

We need something that helps our branch1 router figure out what the public IP address is of the branch2 router, we do this with a protocol called **NHRP (Next Hop Resolution Protocol)**. Here's an explanation of how NHRP works:

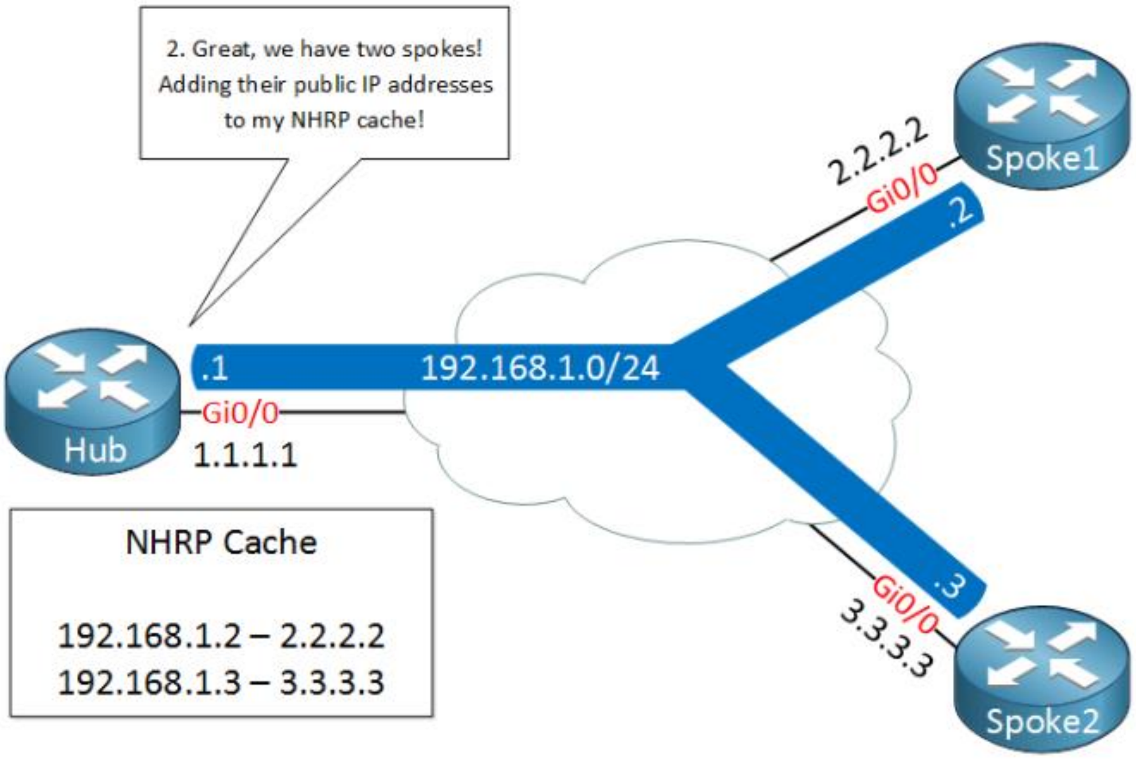
- One router will be the **NHRP server**.
- All other routers will be **NHRP clients**.
- NHRP clients register themselves with the NHRP server and **report their public IP address**.
- The NHRP server **keeps track of all public IP addresses in its cache**.
- When one router wants to tunnel something to another router, it will **request the NHRP server** for the public IP address of the other router.

Since NHRP uses this *server and clients* model, it makes sense to use a hub and spoke topology for multipoint GRE. Our hub router will be the NHRP server and all other routers will be the spokes.

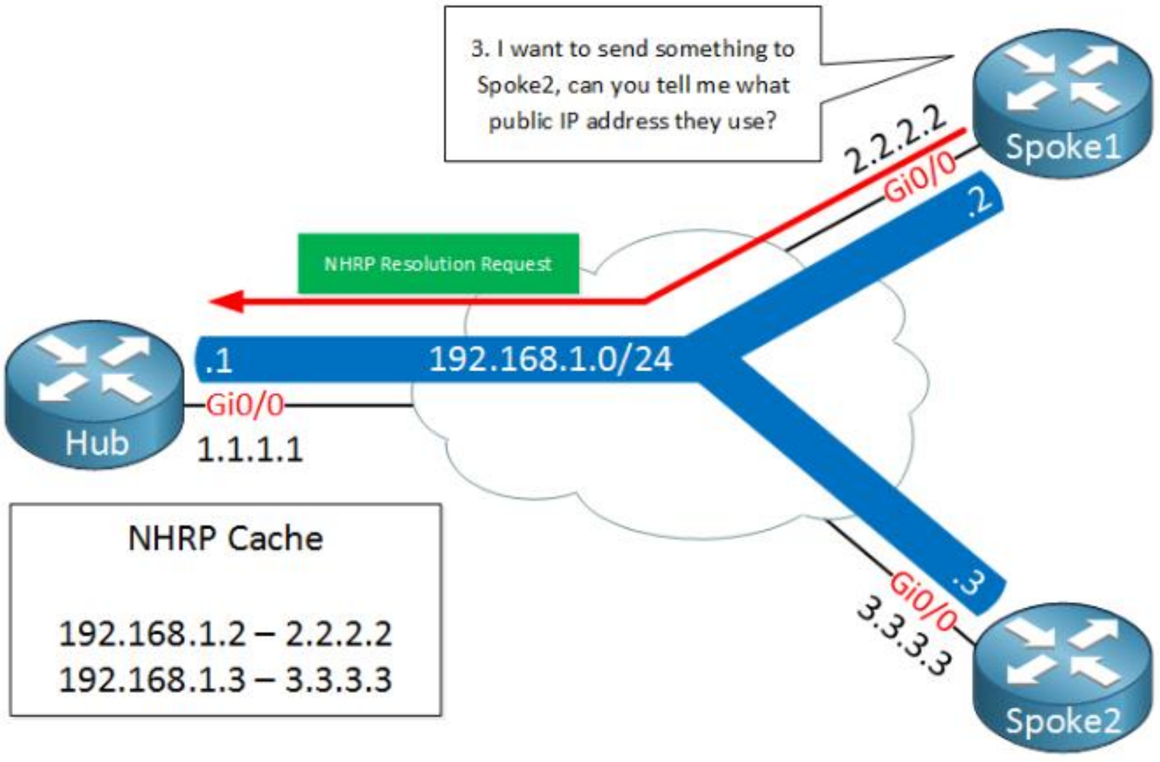
Here's an an illustration of how NHRP works with multipoint GRE:


1. Establishing tunnel to Hub! I'm using public IP address 2.2.2.2
2.2.2.2
Gi0/0
.2
Spoke1
NHRP Registration Request
.1
192.168.1.0/24
Hub
Gi0/0
1.1.1.1
NHRP Registration Request
.3
Gi0/0
3.3.3.3
Spoke2
1. Establishing tunnel to Hub! I'm using public IP address 3.3.3.3

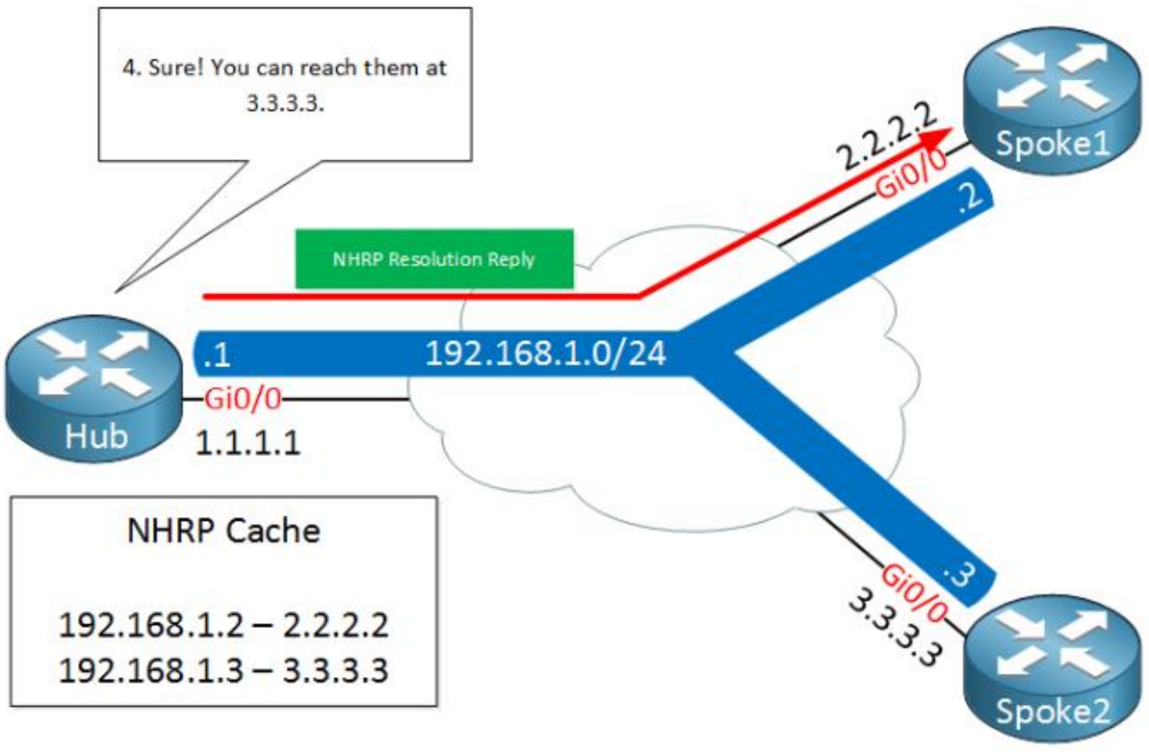
Above we have two spoke routers (NHRP clients) which establish a tunnel to the hub router. Later once we look at the configurations you will see that the destination IP address of the hub router will be **statically configured** on the spoke routers. The hub router will **dynamically** accept spoke routers. The routers will use a **NHRP registration request** message to register their public IP addresses to the hub.



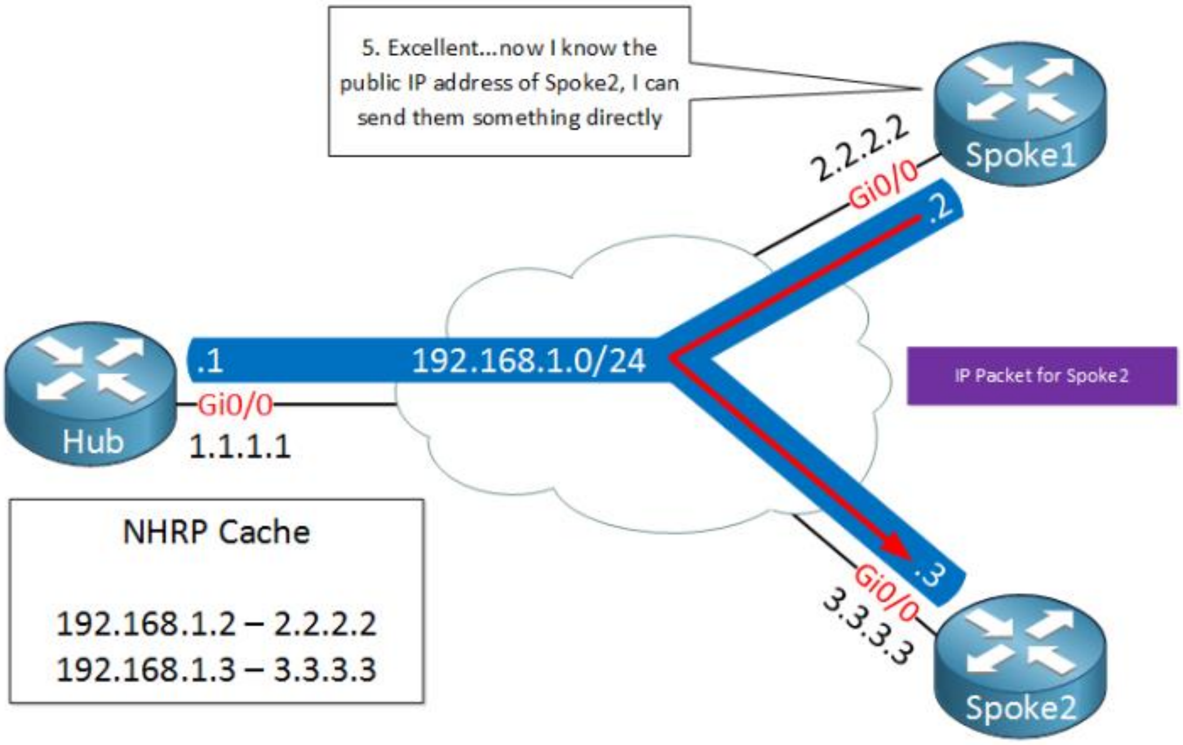
The hub, our NHRP server will create a mapping between the public IP addresses and the IP addresses of the tunnel interfaces.



A few seconds later, spoke1 decides that it wants to send something to spoke2. It needs to figure out the destination public IP address of spoke2 so it will send a **NHRP resolution request**, asking the Hub router what the public IP address of spoke 2 is.



The Hub router checks its cache, finds an entry for spoke 2 and sends the **NHRP resolution reply** to spoke1 with the public IP address of spoke2.



Spoke1 now knows the destination public IP address of spoke2 and is able to tunnel something directly. This is great, we only required the hub to figure out what the public IP address is and all traffic can be sent from spoke to spoke directly.

When we talk about DMVPN, we often refer to an **underlay** and **overlay** network:

- The underlay network is the network we use for connectivity between the different routers, for example the Internet.
- The overlay network is our private network with GRE tunnels.

NOTE:
NHRP is an old protocol (the RFC is from 1998) which was originally developed for NBMA networks like frame-relay or ATM.

Non-broadcast multiple access network (**NBMA**) A non-broadcast multiple access network (NBMA) is a computer network to which multiple hosts are attached, but data is transmitted only directly from one computer to another single host over a virtual circuit or across a switched fabric. NBMA networks do support multicast or broadcast traffic manually (pseudo-broadcasts). Some common examples of nonbroadcast network technologies include Asynchronous Transfer Mode (ATM), Frame Relay, X.25, and home power line networking.

DMVPN has different versions which we call phases, there's three of them:

- Phase 1
- Phase 2
- Phase 3

Let me give you an overview of the three phases:

Phase 1

With phase 1 we use NHRP so that spokes can register themselves with the hub. The hub is the only router that is using a multipoint GRE interface, **all spokes will be using regular point-to-point GRE tunnel interfaces**. This means that there will be **no direct spoke-to-spoke** communication, all traffic has to go through the hub!

Since our traffic has to go through the hub, our routing configuration will be quite simple. Spoke routers only need a summary or default route to the hub to reach other spoke routers.

Phase 2

The disadvantage of phase 1 is that there is no direct spoke to spoke tunnels. In phase 2, **all spoke routers use multipoint GRE** tunnels so we do have direct spoke to spoke tunneling. When a spoke router wants to reach another spoke, it will send an NHRP resolution request to the hub to find the NBMA IP address of the other spoke.

There are two requirements to make spoke to spoke tunnels work:

- Spoke routers need to have a route for the network that they are trying to reach.
- The next hop IP address of the route has to be the remote spoke.

We are using a hub and spoke topology so only the hub will exchange routing information with the spokes. Depending on the routing protocol, it's possible that the hub changes the next hop IP address of routes that it advertises to the spokes. When it does then a spoke will use the hub as the destination when it's trying to reach a remote spoke.

Summarization on the hub is not possible since the spoke routers require specific routes.

Phase 3

The final phase of DMVPN changes the way NHRP operates. The spoke routers **no longer need specific routes** to reach remote spokes and it **doesn't matter what the next hop IP address** is. When a spoke router wants to reach a remote spoke, they will forward their traffic to the hub. When the hub receives the traffic, it will realize that another spoke is the destination and it will then send a **NHRP redirect to both spokes**.

When the spokes receive the NHRP redirect, they will both send a NHRP resolution to figure out each other's NBMA IP addresses. The spoke routers will then **install a new entry in the routing table** so that they can reach each other directly.

Phase 3 vs Phase2:
- The main advantage is that you have smaller routing tables. In phase 2, each spoke router requires specific entries for networks it wants to reach behind other spoke routers.
- With phase 3, a summary route is all you need.
- no longer need specific routes
- doesn't matter what the next hop IP address

DMVPN ADDON COMPONENT:
IPSec

What about IPsec? It's not required but since you probably use DMVPN on the Internet, it's a good idea to encrypt your tunnels. Once you understand multipoint GRE, NHRP and the different DMVPN phases then you will see that IPsec is very simple to implement.

Conclusion

Understanding the "basics" of DMVPN shouldn't be too hard but this is what makes it complex:

- You need to understand the three phases and their differences.
- The impact of running different routing protocols using the three phases.

For example, we can try OSPF using its broadcast, non-broadcast, point-to-point, point-to-multipoint and point-to-multipoint non-broadcast network types on DMVPN phase 1,2 and 3. That's a lot of different variations…we also have RIP, EIGRP, iBGP and eBGP.

To help you understand DMVPN, we will see how to configure phase 1, 2 and 3 and how to configure any of the routing protocols and their different options.

LAB

GNS3 LAB

PHASE I

BASIC DMVPN CONFIG

=================

- UNDERLAY 10.10.1.98/24
- OVERLAY 172.168.1.98/24

HUB

```
hostname HUB
int fa0/0
ip add 10.10.1.98 255.255.255.0
no shut
int tun 0
ip add 172.168.1.98 255.255.255.0
tunnel mode gre multipoint
tunnel source fa0/0
ip nhrp authentication DMVPN
ip nhrp map multicast dynamic
ip nhrp network-id 1
```

- ip nhrp network-id 1 **tunnel mode**: by default the tunnel mode will be point-to-point GRE, we require a multipoint interface on the hub.
- **tunnel source**: the tunnel destinations will be dynamic but we still have to configure the source, our Gigabit0/1 interface.
- **ip nhrp authentication**: we can authenticate our NHRP traffic, it's optional but a good idea to enable. I'm using pre-shared key "DMVPN".
- **ip nhrp map multicast dynamic**: this command tells the hub router where to forward multicast packets to. Since the IP addresses of the spoke routers are unknown, we use dynamic to automatically add their IP addresses to the multicast destination list when the spokes register themselves.
- **ip nhrp network-id**: when you use multiple DMVPN networks, you need the network ID to differentiate between the two networks. This value is only locally significant but for troubleshooting reasons it's best to use the same value on all routers.

SPOKE1

```
hostname SPOKE1
int fa0/0
ip add 10.10.1.99 255.255.255.0
no shut
```

```
int tun 0
ip add 172.168.1.99 255.255.255.0
ip nhrp authentication DMVPN
ip nhrp map 172.168.1.98 10.10.1.98
ip nhrp map multicast 10.10.1.98
ip nhrp network-id 1
ip nhrp nhs 172.168.1.98
tunnel source f0/0
tunnel destination 10.10.1.98
```

```
SPOKE2
hostname SPOKE2
int fa0/0
ip add 10.10.1.100 255.255.255.0
no shut
int tun 0
ip add 172.168.1.100 255.255.255.0
ip nhrp authentication DMVPN
ip nhrp map 172.168.1.98 10.10.1.98
ip nhrp map multicast 10.10.1.98
ip nhrp network-id 1
ip nhrp nhs 172.168.1.98
tunnel source f0/0
tunnel destination 10.10.1.98
```

- **ip nhrp map**: we use this on the spoke to create a static mapping for the hub's tunnel address (172.16.123.1) and the hub's NBMA address (192.168.123.1). This will be stored in the NHRP cache of the spoke router.
- **ip nhrp map multicast**: here we specify which destinations should receive broadcast or multicast traffic through the tunnel interface. The confusing part is that you have to enter the NBMA address here. We need this command since routing protocols like RIP, EIGRP and OSPF require multicast.
- **ip nhrp nhs**: this is where we specify the NHRP server, our hub router.
- **tunnel destination**: we are using point-to-point GRE so we need to specify a destination for the tunnel, this will be the hub router.

```
Spoke1#

%LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel0, changed state to up
```

```
Hub#show dmvpn
```

```
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete

        N - NATed, L - Local, X - No Socket

        T1 - Route Installed, T2 - Nexthop-override

        C - CTS Capable

        # Ent --> Number of NHRP entries with same NBMA peer

        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting

        UpDn Time --> Up or Down Time for a Tunnel

==========================================================================



Interface: Tunnel0, IPv4 NHRP Details

Type:Hub, NHRP Peers:1,



 # Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb

 ----- --------------- --------------- ----- -------- -----

     1 192.168.123.2      172.16.123.2    UP 00:03:36     D

Hub#show ip nhrp

172.16.123.2/32 via 172.16.123.2

   Tunnel0 created 00:04:02, expire 01:55:57

   Type: dynamic, Flags: unique registered used nhop

   NBMA address: 192.168.123.2
```

```
Hub & Spoke1

#debug nhrp

NHRP protocol debugging is on
```

===========================

DMVPN Phase 1 EIGRP Routing

===========================

HUB

```
int lo 0
ip add 1.1.1.1 255.255.255.0
no shut
!
router eigrp 1
network 1.1.1.0 0.0.0.255
network 172.168.1.0 0.0.0.255
no auto-summary
```

SPOKE1

```
int lo 0
ip add 2.2.2.2 255.255.255.0
no shut
!
router eigrp 1
network 2.2.2.0 0.0.0.255
network 172.168.1.0 0.0.0.255
no auto-summary
```

SPOKE2

```
int lo 1
ip add 3.3.3.3 255.255.255.0
```

```
no shut
!
router eigrp 1
network 3.3.3.0 0.0.0.255
network 172.168.1.0 0.0.0.255
no auto-summary
```

EIGRP is a distance vector routing protocol so we have split horizon issues. The spoke routers don't see each other's networks. Let's fix this for now:

```
Hub(config)#
interface Tunnel 0
no ip split-horizon eigrp 1
```

VERIFICATION:

```
SPOKE2#ping 2.2.2.2 so lo 0
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 3.3.3.3
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 36/61/112 ms

SPOKE2#traceroute 2.2.2.2 source loopback 0
Type escape sequence to abort.
Tracing the route to 2.2.2.2
VRF info: (vrf in name/id, vrf out name/id)
  1 172.168.1.98 32 msec 20 msec 20 msec
  2 172.168.1.99 32 msec 36 msec *
```

===========================

DMVPN Phase 1 OSPF Routing - POINT TO POINT

===========================

```
HUB

router ospf 1

network 172.168.1.0 0.0.0.255 area 0

network 1.1.1.0 0.0.0.255 area 0


SPOKE1

router ospf 1

network 172.168.1.0 0.0.0.255 area 0

network 2.2.2.0 0.0.0.255 area 0


SPOKE2

router ospf 1

network 172.168.1.0 0.0.0.255 area 0

network 3.3.3.0 0.0.0.255 area 0
```

ERROR LOGS:

```
*Sep  2 14:41:49.115: %OSPF-5-ADJCHG: Process 1, Nbr 3.3.3.3 on Tunnel0 from EXCHANGE to
DOWN, Neighbor Down: Adjacency forced to reset

*Sep  2 14:41:49.211: %OSPF-5-ADJCHG: Process 1, Nbr 2.2.2.2 on Tunnel0 from EXCHANGE to
DOWN, Neighbor Down: Adjacency forced to reset
```

The default is point-to-point and we are using multipoint interfaces. Our hub router expects one neighbor, not two. It will keep establishing and tearing neighbor adjacencies with the default network type.

This is never going to work, not for any of the phases so the OSPF point-to-point network type is something you can forget about with DMVPN. Let's move on to the next one!

===========================

DMVPN Phase 1 OSPF Routing - BROADCAST

===========================

```
no router ospf 1


HUB, SPOKE1, SPOKE2(config)#

interface Tunnel 0

ip ospf network broadcast
```

There is no direct communication between spoke routers so we need to make sure that the spoke router can never be elected as DR or BDR. To enforce this we'll set their priority to 0:

```
SPOKE1, SPOKE2(config)#

interface Tunnel 0

ip ospf priority 0


clear ip ospf process
```

VERIFICATIONS:

```
SPOKE2#ping 2.2.2.2 so lo 1

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:

Packet sent with a source address of 3.3.3.3

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 48/93/216 ms


SPOKE2#traceroute 2.2.2.2 so lo 1

Type escape sequence to abort.

Tracing the route to 2.2.2.2

VRF info: (vrf in name/id, vrf out name/id)

  1 172.168.1.98 92 msec 60 msec 84 msec

  2 172.168.1.99 128 msec 96 msec *
```

===========================

DMVPN Phase 1 BGP

===========================

```
Hub(config)#

router bgp 65001

 network 1.1.1.1 mask 255.255.255.255

 neighbor 172.168.1.99 remote-as 65002

 neighbor 172.168.1.100 remote-as 65003


SPOKE1(config)#

router bgp 65002

 network 2.2.2.0 mask 255.255.255.0

 neighbor 172.168.1.98 remote-as 65001



SPOKE2(config)#

router bgp 65003

 network 3.3.3.0 mask 255.255.255.0

 neighbor 172.168.1.98 remote-as 65001
```

VERIFICATION:

```
SPOKE2#ping 2.2.2.2 so lo 1

Type escape sequence to abort.

Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:

Packet sent with a source address of 3.3.3.3

!!!!!

Success rate is 100 percent (5/5), round-trip min/avg/max = 32/45/80 ms


SPOKE2#traceroute 2.2.2.2 so lo 1
```

```
Type escape sequence to abort.

Tracing the route to 2.2.2.2

VRF info: (vrf in name/id, vrf out name/id)

  1 172.168.1.98 20 msec 24 msec 16 msec

  2 172.168.1.99 20 msec 36 msec *
```

DMVPN Phase 2 Basic Configuration

This time i'll explain how you can configure DMVPN phase 2. Once we have a basic configuration then we can try to run RIP, EIGRP, OSPF and BGP on top of it.

The configuration of DMVPN phase 1 and 2 is similar except for two key items:

- The spoke routers will now use multipoint GRE interfaces instead of point-to-point GRE interfaces.
- We don't configure a manual destination anymore on the spoke routers.
- First registration reply and registration request and then resolution request and resolution reply

That's it, those two changes make the difference between running DMVPN phase 1 or 2. Let's take a look at the configuration, here's the topology we will use:

BASIC DMVPN CONFIG PHASE II

=================

The configuration of DMVPN phase 1 and 2 is similar except for two key items:

The spoke routers will now use multipoint GRE interfaces instead of point-to-point GRE interfaces.
We don't configure a manual destination anymore on the spoke routers.

UNDERLAY 10.10.1.98/24
OVERLAY 172.168.1.98/24

```
HUB
hostname HUB
int fa0/0
ip add 10.10.1.98 255.255.255.0
no shut
int tun 0
ip add 172.168.1.98 255.255.255.0
tunnel mode gre multipoint
tunnel source fa0/0
ip nhrp authentication DMVPN
ip nhrp map multicast dynamic
ip nhrp network-id 1
```

```
SPOKE1
Spoke1(config)#
hostname SPOKE1
int fa0/0
ip add 10.10.1.99 255.255.255.0
no shut
interface Tunnel0
ip address 172.168.1.99 255.255.255.0
ip nhrp authentication DMVPN
ip nhrp map 172.168.1.98 10.10.1.98
ip nhrp map multicast 10.10.1.98
ip nhrp network-id 1
ip nhrp nhs 172.168.1.98
tunnel source fa0/0
tunnel mode gre multipoint

SPOKE2
hostname SPOKE2
int fa0/0
ip add 10.10.1.100 255.255.255.0
no shut
interface Tunnel0
ip address 172.168.1.100 255.255.255.0
ip nhrp authentication DMVPN
ip nhrp map 172.168.1.98 10.10.1.98
ip nhrp map multicast 10.10.1.98
ip nhrp network-id 1
ip nhrp nhs 172.168.1.98
tunnel source fa0/0
tunnel mode gre multipoint

HUB#sh dmvpn
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel0, IPv4 NHRP Details
Type:Hub, NHRP Peers:2,

 # Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
 ----- --------------- --------------- ----- -------- -----
     1 10.10.1.99       172.168.1.99    UP 00:00:21     D
     1 10.10.1.100      172.168.1.100   UP 00:00:14     D

HUB#show ip nhrp
172.168.1.99/32 via 172.168.1.99
   Tunnel0 created 00:00:33, expire 01:59:26
```

```
  Type: dynamic, Flags: unique registered used
  NBMA address: 10.10.1.99
172.168.1.100/32 via 172.168.1.100
  Tunnel0 created 00:00:27, expire 01:59:32
  Type: dynamic, Flags: unique registered used
  NBMA address: 10.10.1.100
```

===========================

DMVPN Phase 2 EIGRP Routing

===========================

```
HUB
int lo 0
ip add 1.1.1.1 255.255.255.0
no shut
!
router eigrp 1
network 1.1.1.0 0.0.0.255
network 172.168.1.0 0.0.0.255
no auto-summary

SPOKE1
int lo 0
ip add 2.2.2.2 255.255.255.0
no shut
!
router eigrp 1
network 2.2.2.0 0.0.0.255
network 172.168.1.0 0.0.0.255
no auto-summary

SPOKE2
int lo 0
ip add 3.3.3.3 255.255.255.0
no shut
!
router eigrp 1
network 3.3.3.0 0.0.0.255
network 172.168.1.0 0.0.0.255
no auto-summary
```

VERIFICATION

```
SPOKE2#show ip route eigrp

      1.0.0.0/24 is subnetted, 1 subnets
D        1.1.1.0 [90/27008000] via 172.168.1.98, 00:00:51, Tunnel0
```

The hub has learned the networks on the loopback interfaces of the spoke routers. The spokes learned the 1.1.1.0/24 network from the Hub but not each other's networks. Like RIP, EIGRP is a distance vector routing protocol so split horizon is preventing the hub from advertising these networks. Let's disable split horizon:

```
Hub(config)#
interface Tunnel 0
no ip split-horizon eigrp 1
```

VERIFICATION

```
SPOKE2# sh ip ro eigrp

      1.0.0.0/24 is subnetted, 1 subnets
D        1.1.1.0 [90/27008000] via 172.168.1.98, 00:03:52, Tunnel0
      2.0.0.0/24 is subnetted, 1 subnets
D        2.2.2.0 [90/28288000] via 172.168.1.98, 00:00:08, Tunnel0

SPOKE2#ping 2.2.2.2 source lo 1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 3.3.3.3
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/31/48 ms

SPOKE2#trace 2.2.2.2 source lo 1
Type escape sequence to abort.
Tracing the route to 2.2.2.2
VRF info: (vrf in name/id, vrf out name/id)
  1 172.168.1.98 20 msec 20 msec 20 msec
  2 172.168.1.99 40 msec 36 msec *
```

IMP:
When spoke1 tries to reach spoke2, it will go through the hub. That's not what we want so let's tell EIGRP not to change the next ho

```
Hub(config)#
interface Tunnel 0
no ip next-hop-self eigrp 1
```

VERIFICATION:

```
SPOKE2#ping 2.2.2.2 source lo 1
*Sep  2 16:35:56.799: %DUAL-5-NBRCHANGE: EIGRP-IPv4 1: Neighbor 172.168.1.98 (Tunnel0) is up:
new adjacency
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 3.3.3.3
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/45/84 ms

SPOKE2#trace 2.2.2.2 source lo 1
Type escape sequence to abort.
Tracing the route to 2.2.2.2
```

```
VRF info: (vrf in name/id, vrf out name/id)
  1 172.168.1.99 16 msec 8 msec *

SPOKE2#trace 2.2.2.2 source lo 1
Type escape sequence to abort.
Tracing the route to 2.2.2.2
VRF info: (vrf in name/id, vrf out name/id)
  1 172.168.1.99 24 msec 24 msec *

SPOKE2#sh ip route eigrp | i 2.2.2.
D        2.2.2.0 [90/28288000] via 172.168.1.99, 00:01:01, Tunnel0
SPOKE2#

SEPERATE DMVPN BETWEEN SPOKE2 <--> SPOKE1
SPOKE2#show dmvpn
Legend: Attrb --> S - Static, D - Dynamic, I - Incomplete
        N - NATed, L - Local, X - No Socket
        # Ent --> Number of NHRP entries with same NBMA peer
        NHS Status: E --> Expecting Replies, R --> Responding, W --> Waiting
        UpDn Time --> Up or Down Time for a Tunnel
==========================================================================

Interface: Tunnel0, IPv4 NHRP Details
Type:Spoke, NHRP Peers:2,

 # Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
 ----- --------------- --------------- ----- -------- -----
     1 10.10.1.98        172.168.1.98    UP 00:10:53     S
     1 10.10.1.99        172.168.1.99    UP 00:02:21     D

SPOKE2#show ip nhrp
172.168.1.98/32 via 172.168.1.98
   Tunnel0 created 00:11:22, never expire
   Type: static, Flags: used
   NBMA address: 10.10.1.98
172.168.1.99/32 via 172.168.1.99
   Tunnel0 created 00:02:51, expire 01:57:09
   Type: dynamic, Flags: router used
   NBMA address: 10.10.1.99

SPOKE2#show  ip nhrp nhs detail
Legend: E=Expecting replies, R=Responding, W=Waiting
Tunnel0:
172.168.1.98  RE priority = 0 cluster = 0  req-sent 1  req-failed 0  repl-recv 1 (00:13:19 ago)
```

===========================

DMVPN Phase 2 OSPF Routing - Broadcast

===========================

We can skip the first network type (point-to-point) since it doesn't work. We have seen in the DMVPN phase 1 OSPF routing lesson what happens when you try this. The hub expects one neighbor on its tunnel interface while in reality we have two neighbors.

Hub, Spoke1 & Spoke2

```
(config)#
interface Tunnel 0
ip ospf network broadcast
```

You need to make sure that the spoke routers will never be elected as DR or BDR:

Spoke1 & Spoke2

```
(config)#
interface Tunnel 0
ip ospf priority 0
```

VERIFICATION

```
SPOKE2#ping 2.2.2.2 so lo 1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 3.3.3.3
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/15/20 ms


SPOKE2#tr 2.2.2.2 so lo 1
Type escape sequence to abort.
Tracing the route to 2.2.2.2
VRF info: (vrf in name/id, vrf out name/id)
  1 172.168.1.99 28 msec 12 msec *

SPOKE2#tr 2.2.2.2 so lo 1
Type escape sequence to abort.
Tracing the route to 2.2.2.2
VRF info: (vrf in name/id, vrf out name/id)
  1 172.168.1.99 12 msec 20 msec *
```

===========================

DMVPN Phase 2 BGP

===========================

```
Hub(config)#
router bgp 65001
 network 1.1.1.1 mask 255.255.255.255
 neighbor 172.168.1.99 remote-as 65002
 neighbor 172.168.1.100 remote-as 65003

SPOKE1(config)#
router bgp 65002
 network 2.2.2.0 mask 255.255.255.0
 neighbor 172.168.1.98 remote-as 65001


SPOKE2(config)#
router bgp 65003
 network 3.3.3.0 mask 255.255.255.0
 neighbor 172.168.1.98 remote-as 65001
```

VERIFICATION

```
SPOKE2#ping 2.2.2.2 so lo 1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
Packet sent with a source address of 3.3.3.3
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 20/21/24 ms

SPOKE2#tr 2.2.2.2 so lo 1
Type escape sequence to abort.
Tracing the route to 2.2.2.2
VRF info: (vrf in name/id, vrf out name/id)
  1 172.168.1.99 40 msec 20 msec *
```

DMVPN PHASE 3

BASIC DMVPN CONFIG PHASE 3

=================

The configuration of DMVPN phase 3 and 2 is very similar. Let’s start with the following DMVPN phase 2 configuration on all routers:

UNDERLAY 10.10.1.98/24
OVERLAY 172.168.1.98/24

```
HUB
hostname HUB
int fa0/0
ip add 10.10.1.98 255.255.255.0
no shut
int tun 0
ip add 172.168.1.98 255.255.255.0
tunnel mode gre multipoint
tunnel source fa0/0
ip nhrp authentication DMVPN
ip nhrp map multicast dynamic
ip nhrp network-id 1
```

```
SPOKE1
Spoke1(config)#
hostname SPOKE1
int fa0/0
ip add 10.10.1.99 255.255.255.0
no shut
interface Tunnel0
ip address 172.168.1.99 255.255.255.0
ip nhrp authentication DMVPN
ip nhrp map 172.168.1.98 10.10.1.98
ip nhrp map multicast 10.10.1.98
ip nhrp network-id 1
ip nhrp nhs 172.168.1.98
tunnel source fa0/0
tunnel mode gre multipoint

SPOKE2
hostname SPOKE2
int fa0/0
ip add 10.10.1.100 255.255.255.0
no shut
interface Tunnel0
ip address 172.168.1.100 255.255.255.0
ip nhrp authentication DMVPN
ip nhrp map 172.168.1.98 10.10.1.98
ip nhrp map multicast 10.10.1.98
ip nhrp network-id 1
ip nhrp nhs 172.168.1.98
tunnel source fa0/0
tunnel mode gre multipoint
```

To migrate from DMVPN phase 2 to 3, we only need two commands…here's the first command:

```
Hub(config)#
interface tunnel 0
ip nhrp redirect

Spoke1 & Spoke2
(config)#
interface Tunnel 0
ip nhrp shortcut
```

The NHRP shortcut command allows the spoke routers to makes changes in the CEF entry when they receive a redirect message from the hub. You will see the NHRP redirect and shortcut in action when we look at the routing configurations.

===========================

DMVPN Phase 3 EIGRP Routing

===========================

HUB

```
int lo 0
ip add 1.1.1.1 255.255.255.0
no shut
!
router eigrp 1
network 1.1.1.0 0.0.0.255
network 172.168.1.0 0.0.0.255
no auto-summary

SPOKE1
int lo 0
ip add 2.2.2.2 255.255.255.0
no shut
!
router eigrp 1
network 2.2.2.0 0.0.0.255
network 172.168.1.0 0.0.0.255
no auto-summary

SPOKE2
int lo 0
ip add 3.3.3.3 255.255.255.0
no shut
!
router eigrp 1
network 3.3.3.0 0.0.0.255
network 172.168.1.0 0.0.0.255
no auto-summary
```

The spoke routers don't require specific entries thanks to NHRP traffic indication. I will advertise a default route on the hub router:

```
Hub(config)#
interface tunnel 0
ip summary-address eigrp 1 0.0.0.0 0.0.0.0
```

===========================

DMVPN Phase 3 OSPF Routing

===========================

We can skip the first network type (point-to-point) since it doesn't work. We have seen in the DMVPN phase 1 OSPF routing lesson what happens when you try this. The hub expects one neighbor on its tunnel interface while in reality we have two neighbors.

```
Hub, Spoke1 & Spoke2
(config)#
interface Tunnel 0
ip ospf network broadcast
```

You need to make sure that the spoke routers will never be elected as DR or BDR:

```
Spoke1 & Spoke2
(config)#
interface Tunnel 0
ip ospf priority 0
```

DMVPN WITH IPSEC

IPsec

IPsec has two phases, phase 1 and 2 (don't confuse them with the DMVPN phases).

Phase 1

We need an ISAKMP policy that matches on all our routers. Let's pick something:

```
Hub, Spoke1 & Spoke 2

(config)#crypto isakmp policy 10

(config-isakmp)#authentication pre-share

(config-isakmp)#encryption aes 128

(config-isakmp)#group 5

(config-isakmp)#hash sha256
```

When it comes to encryption we can choose between pre-shared keys or PKI. To keep it simple, I'll go for the pre-shared keys:

```
Hub(config)#crypto isakmp key DMVPN_KEY address ?

  A.B.C.D  Peer IP address

  ipv6     define shared key with IPv6 address
```

When you configure the pre-shared key you have to enter the **NBMA address**. Keep in mind that encryption occurs before multipoint GRE / NHRP. We also have to specify a peer address, we have two options here:

- Configure a pre-shared key for each "router pair" you have: this means we use a unique key for hub-spoke1, hub-spoke2 and spoke1-spoke2. This is secure but it's not a very scalable solution, the more spoke routers we add to the network, the more keys we have to configure.
- Configure a "wildcard" pre-shared key: this allows us to use a single key for all routers. This is the most convenient but it also means that if you want to change the key, you have to do it on all your routers.

I'll use the wildcard pre-shared key for our example:

```
Hub, Spoke1 & Spoke2

(config)#crypto isakmp key DMVPN_KEY address 0.0.0.0
```

Now we can worry about phase 2.

Phase 2

On each router we require a transform set that tells the router what encryption/hashing to use and if we want tunnel or transport mode:

```
Hub, Spoke1 & Spoke2

(config)#crypto ipsec transform-set DMVPN_TRANSFORM esp-aes esp-sha-hmac

(cfg-crypto-trans)#mode transport
```

I'll go for ESP with AES as the encryption algorithm and SHA for hashing. The mode is important, since we are using GRE we are already using tunnels so we can use transport mode. If you use tunnel mode then we will have even more overhead which is not required.

To apply the transform set on our tunnel interfaces we need to create a profile:

```
Hub, Spoke1 & Spoke2

(config)#crypto ipsec profile DMVPN_PROFILE

(ipsec-profile)#set transform-set DMVPN_TRANSFORM
```

Now we can apply this profile to the tunnel interfaces:

```
Hub, Spoke1 & Spoke2
```

```
(config)#interface Tunnel 0

(config-if)#tunnel protection ipsec profile DMVPN_PROFILE
```

Everything is now in place. Let’s test drive this setup

Verification

IPsec occurs before multipoint GRE and NHRP. To test if everything is working properly we should go for a fresh start. I’ll shut all the interfaces on our routers:

```
Hub, Spoke1 & Spoke2

(config)#interface Tunnel 0

(config-if)#shutdown
```

First we will enable the hub again:

```
Hub(config)#interface Tunnel 0

Hub(config-if)#no shutdown
```

And then we will enable the spoke routers again:

```
Spoke1 & Spoke2

(config)#interface Tunnel 0

(config-if)#no shutdown
```

Let’s check if the spoke routers were able to register again with the hub:

```
Hub#show dmvpn | begin Peer

Type:Hub, NHRP Peers:2,



 # Ent  Peer NBMA Addr Peer Tunnel Add State  UpDn Tm Attrb
```

```
----- --------------- --------------- ----- -------- -----

    1 192.168.123.2     172.16.123.2    UP 00:02:35     D

    1 192.168.123.3     172.16.123.3    UP 00:00:19     D
```

Stateful Switchover (SSO)

SSO Overview

The switch is supports fault resistance by allowing a redundant supervisor engine to take over if the primary supervisor engine fails. Cisco SSO (frequently used with NSF) minimizes the time a network is
unavailable to its users following a switchover while continuing to forward IP packets. The switch is supports route processor redundancy (RPR).

SSO has many benefits. Because the SSO feature maintains stateful feature information, user session information is maintained during a switchover, and line cards continue to forward network traffic with
no loss of sessions, providing improved network availability. SSO provides a faster switchover than RPR
by fully initializing and fully configuring the standby RP, and by synchronizing state information, which
can reduce the time required for routing protocols to converge. Network stability may be improved with
the reduction in the number of route flaps had been created when routers in the network failed and lost
their routing tables.

SSO Operation

SSO establishes one of the RPs as the active processor while the other RP is designated as the standby
processor. SSO fully initializes the standby RP, and then synchronizes critical state information between
the active and standby RP.
During an SSO switchover, the line cards are not reset, which provides faster switchover between the
processors. The following events cause a switchover:
• A hardware failure on the active supervisor engine
• Clock synchronization failure between supervisor engines
• A manual switchover or shutdown
An SSO switchover does not interrupt Layer 2 traffic. An SSO switchover preserves FIB and adjacency entries and can forward Layer 3 traffic after a switchover. SSO switchover duration is between 0 and

3
seconds.

Synchronization Overview
In networking devices running SSO, both RPs must be running the same configuration so that the standby RP is always ready to assume control if the active RP fails. SSO synchronizes the configuration
information from the active RP to the standby RP at startup and whenever changes to the active RP configuration occur. This synchronization occurs in two separate phases:
• While the standby RP is booting, the configuration information is synchronized in bulk from the active RP to the standby RP.
• When configuration or state changes occur, an incremental synchronization is conducted from the active RP to the standby RP.

General Restrictions
• Two RPs must be installed in the chassis, each running the same version of the Cisco IOS software.
• Both RPs must run the same Cisco IOS image. If the RPs are operating different Cisco IOS images, the system reverts to RPR mode even if SSO is configured.
• Configuration changes made through SNMP may not be automatically configured on the standby RP
after a switchover occurs.
• Load sharing between dual processors is not supported.
• The Hot Standby Routing Protocol (HSRP) is not supported with Cisco Nonstop Forwarding with Stateful Switchover. Do not use HSRP with Cisco Nonstop Forwarding with Stateful Switchover.
• Enhanced Object Tracking (EOT) is not stateful switchover-aware and cannot be used with HSRP, Virtual Router Redundancy Protocol (VRRP), or Gateway Load Balancing Protocol (GLBP) in SSO mode.
• Multicast is not SSO-aware and restarts after switchover; therefore, multicast tables and data structures are cleared upon switchover.

Configuration Mode Restrictions
• The configuration registers on both RPs must be set the same for the networking device to behave the same when either RP is rebooted.
• During the startup (bulk) synchronization, configuration changes are not allowed. Before making any configuration changes, wait for a message similar to the following:
%HA-5-MODE:Operating mode is sso, configured mode is sso.

Switchover Process Restrictions
• If any changes to the fabric configuration happen simultaneously with an RP switchover, the chassis
is reset and all line cards are reset.
• If the switch is configured for SSO mode, and the active RP fails before the standby is ready to switch over, the switch will recover through a full system reset.
• During SSO synchronization between the active and standby RPs, the configured mode will be RPR. After the synchronization is complete, the operating mode will be SSO. If a switchover occurs before the synchronization is complete, the switchover will be in RPR mode.
• If a switchover occurs before the bulk synchronization step is complete, the new active RP may be in inconsistent states. The switch will be reloaded in this case.

• Switchovers in SSO mode will not cause the reset of any line cards.
• Interfaces on the RP itself are not stateful and will experience a reset across switchovers. In particular, the GE interfaces on the RPs are reset across switchovers and do not support SSO.
• Any line cards that are not online at the time of a switchover (line cards not in Cisco IOS running state) are reset and reloaded on a switchover.

Command	**Purpose**
Router> **enable**	Enables privileged EXEC mode (enter your password if prompted).
Router# **configure terminal**	Enters global configuration mode.
Router(config)# **redundancy**	Enters redundancy configuration mode.
Router(config)# **mode sso**	Sets the redundancy configuration mode to SSO on both the active and standby RP.
Router(config-red)# **end**	Exits redundancy configuration mode and returns the switch to privileged EXEC mode.
Router# **copy running-config startup-config**	Saves the configuration changes to the startup configuration file.

MPLS

To understand MPLS there are two questions we need to answer:

1. What is MPLS?
2. Why do we need MPLS?

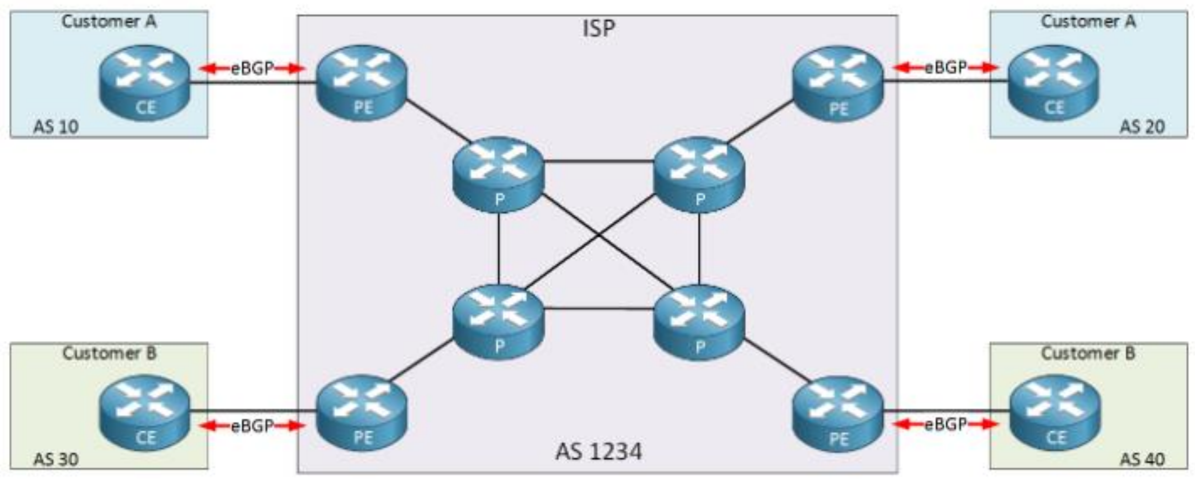
We are going to start this lesson with an explanation of why we need it and how MPLS solves some of the issues of other protocols, this will help you to understand why we use MPLS. In the second part of this lesson you will learn what MPLS is and how it actually works.

When you want to learn MPLS, you need to be very familiar with the following topics before you continue:

1. IGPs (like OSPF and EIGRP)
2. Tunneling (GRE)
3. CEF (Cisco Express Forwarding)
4. BGP (Border Gateway Protocol)

Having said that, let's get started!

Why do we need MPLS?

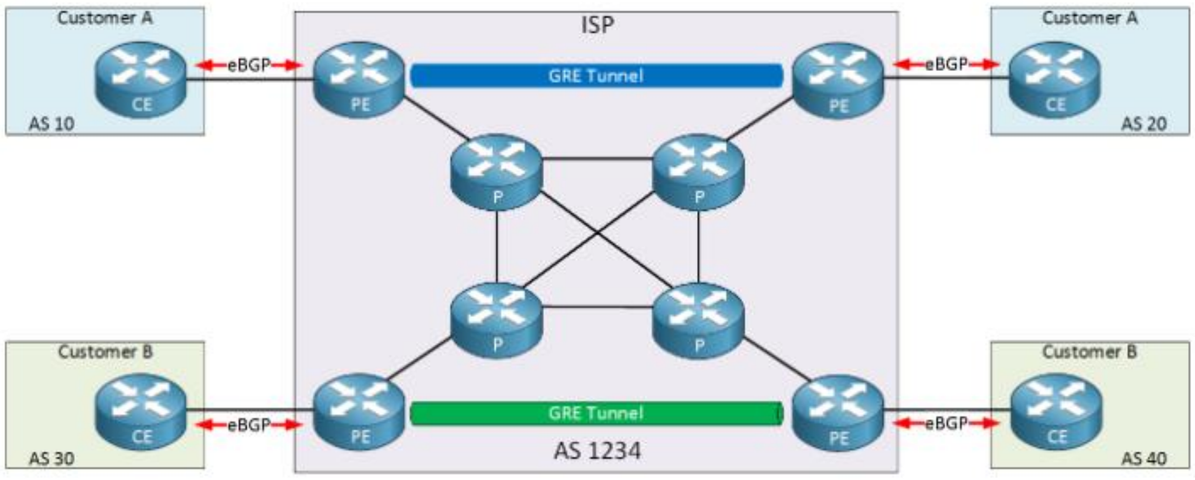


Above we have an example of an ISP with two customers called "A" and "B". The ISP only offers Internet connectivity and no other services. Each customer uses the ISP to have connectivity between their sites.

To accomplish our goal, the ISP is running eBGP between the CE (Customer Edge) and PE (Provider Edge) to exchange prefixes. This means all internal (P) routers of the ISP have to run iBGP or they don't know where to forward their packets to.

A full internet routing table currently has > 500.000 prefixes and with 8 ISP routers running iBGP, we need 28 iBGP peerings. We can reduce this number by using route reflectors or a confederation. All routers have to do lookups in the routing table for any possible destination.

Now here's something to think about…when our goal is to have connectivity between two customer sites, why should all internal P routers know about this? The only routers that need to know how to reach the customer sites are the PE routers of the provider. Why not build a tunnel between the PE routers? Take a look at the picture below:



In the picture above I added two GRE tunnels:

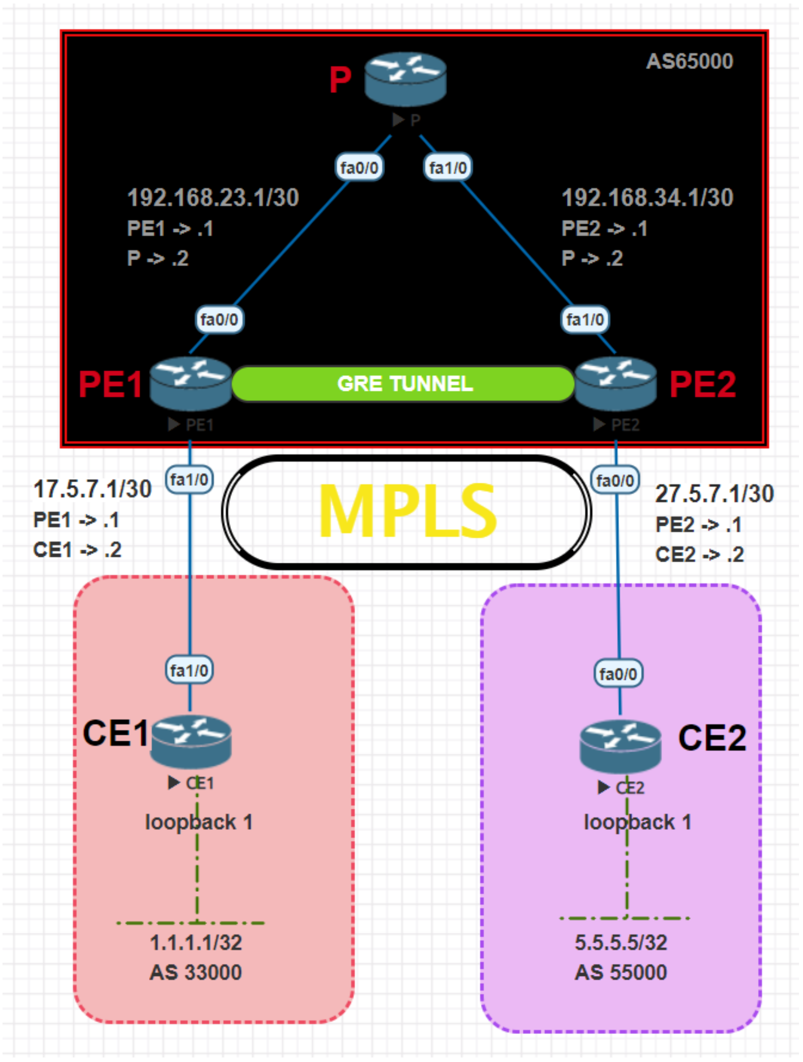
- The two PE routers at the top will use a GRE tunnel for the customer A sites.
- The two PE routers at the bottom will use a GRE tunnel for the customer B sites.

With a solution like this, we can have a **BGP free core**! There's only two places where we need BGP:

- eBGP between the PE and CE router.
- iBGP between two PE routers.

Let's take a closer look at the solution I described above.

GNS3 LAB:



PE1

```
int fa0/0
ip add 192.168.23.1 255.255.255.252
no shut
int fa1/0
ip add 17.5.7.1 255.255.255.252
no shut
int lo 1
ip add 2.2.2.2 255.255.255.255
no shut
```

PE2

```
int fa1/0
ip add 192.168.34.1 255.255.255.252
no shut
int fa0/0
ip add 27.5.7.1 255.255.255.252
no shut
int lo 1
```

```
ip add 4.4.4.4 255.255.255.255
no shut
```

P

```
int fa0/0
ip add 192.168.23.2 255.255.255.252
no shut
int fa1/0
ip add 192.168.34.2 255.255.255.252
no shut
int lo 1
ip add 3.3.3.3 255.255.255.255
no shut
```

CE1

```
int fa1/0
ip add 17.5.7.2 255.255.255.252
no shut
int lo 1
ip add 1.1.1.1 255.255.255.255
no shut
```

CE2

```
int fa0/0
ip add 27.5.7.2 255.255.255.252
no shut
int lo 1
ip add 5.5.5.5 255.255.255.255
no shut
```

========

OSPF in CLOUD

PE1(config)#

```
router ospf 1
network 192.168.23.0 0.0.0.255 area 0
network 2.2.2.2 0.0.0.0 area 0
```

P(config)#

```
router ospf  1
network 192.168.23.0 0.0.0.255 area 0
network 192.168.34.0 0.0.0.255 area 0
network 3.3.3.3 0.0.0.0 area 0
```

PE2(config)#

```
router ospf 1
network 192.168.34.0 0.0.0.255 area 0
network 4.4.4.4 0.0.0.0 area 0
```

====

eBGP config between PE & CE

CE1(config)#

```
router bgp 33000
neighbor 17.5.7.1 remote-as 65000
```

```
network 1.1.1.1 mask 255.255.255.255

CE2(config)#
router bgp 55000
neighbor 27.5.7.1 remote-as 65000

PE1(config)#
router bgp 65000
neighbor 17.5.7.2 remote-as 33000

PE2(config)#
router bgp 65000
neighbor 27.5.7.2 remote-as 55000
network 4.4.4.4 mask 255.255.255.255
```

==============

GRE config between PE1 & PE2

```
PE1(config)#
interface tunnel 0
tunnel source 2.2.2.2
tunnel destination 4.4.4.4
ip address 192.168.24.2 255.255.255.0

PE2(config)#
interface tunnel 0
tunnel source 4.4.4.4
tunnel destination 2.2.2.2
ip address 192.168.24.4 255.255.255.0
```

=============

iBGP between PE1 & PE2

```
PE1(config)#
router bgp 65000
neighbor 192.168.24.4 remote-as 65000
neighbor 192.168.24.4 next-hop-self

PE2(config)#
router bgp 65000
neighbor 192.168.24.2 remote-as 65000
neighbor 192.168.24.2 next-hop-self
```

Our PE routers will establish an iBGP peering using the IP addresses on the GRE tunnel.

VERIFICATION

```
CE2#tr 1.1.1.1 so lo 1
Type escape sequence to abort.
Tracing the route to 1.1.1.1
VRF info: (vrf in name/id, vrf out name/id)
  1 27.5.7.1 12 msec 20 msec 20 msec
```

```
 2 192.168.24.2 64 msec 64 msec 52 msec
 3 17.5.7.2 92 msec 72 msec *

CE2#ping 1.1.1.1 so lo 1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 5.5.5.5
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 68/76/84 ms
```

The outer IP header has source address 4.4.4.4 and destination address 2.2.2.2, the P router knows how to route these since it learned these addresses through OSPF.

For now, keep in mind that tunneling is used to create a BGP free core. Hold this thought while you read the next section of this lesson where we finally start talking about MPLS!

What is MPLS?

In the previous example I used a GRE tunnel but I could have used any tunneling mechanism. Besides GRE, there's IP-in-IP, Q-in-Q and…

MPLS (Multi Protocol Label Switching).

What does multi protocol label switching mean?

- **Multi protocol**: besides IP you can tunnel pretty much anything…IP, IPv6, Ethernet, PPP, frame-relay, etc.
- **Label switching**: forwarding is done based on labels, not by looking up the destination in the routing table.

MPLS can do anything that any of the other tunneling protocols support and it can do a lot more than that.

Let's start with something simple, let's replace the GRE tunnel from the previous example with MPLS so I can explain how MPLS uses labels.

First let's get rid of the GRE tunnel and the BGP peering between PE1 and PE2:

```
=======
MPLS
=======
Remove GRE, IBGP from
PE1 & PE2
(config)#no interface tunnel 0

PE1(config)#
router bgp 65000
no neighbor 192.168.24.4 remote-as 65000

PE2(config)#
router bgp 65000
```

```
no neighbor 192.168.24.2 remote-as 65000
```

=========
iBGP between PE1 and PE2 over loopback

```
PE1(config)#
router bgp 65000
neighbor 4.4.4.4 remote-as 65000
neighbor 4.4.4.4 update-source loopback 1
neighbor 4.4.4.4 next-hop-self
PE2(config)#
router bgp 65000
neighbor 2.2.2.2 remote-as 65000
neighbor 2.2.2.2 update-source loopback 1
neighbor 2.2.2.2 next-hop-self
```

=======
MPLS config

```
PE1(config)#
interface fa0/0
mpls ip

P(config)#
interface fa0/0
mpls ip
interface fa1/0
mpls ip

PE2(config)#
interface fa1/0
mpls ip
```

```
*Sep  7 18:18:55.663: %LDP-5-NBRCHG: LDP Neighbor 3.3.3.3:0 (1) is UP
```

======
VERIFICATION

```
CE2#ping 1.1.1.1 so lo 1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 5.5.5.5
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 76/81/84 ms
CE2#
```

Great, it works. Why does it work? Keep in mind there is no iBGP on the P router:

```
P#show ip cef 1.1.1.1
0.0.0.0/0
  no route
P#show ip cef 5.5.5.5
0.0.0.0/0
  no route
P#
```

Normally this traffic should be dropped since this router has no idea how it can reach 5.5.5.5. However, since we enabled MPLS we are now using labels for our forwarding decisions. Let me explain how that works.

Let us start from PE2

```
PE2#show ip rout 1.1.1.1
Routing entry for 1.1.1.1/32
  Known via "bgp 65000", distance 200, metric 0
  Tag 33000, type internal
  Last update from 2.2.2.2 01:43:39 ago
  Routing Descriptor Blocks:
  * 2.2.2.2, from 2.2.2.2, 01:43:39 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1
      Route tag 33000
      MPLS label: none
```

```
E2#show mpls forwarding-table
Local      Outgoing   Prefix          Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id    Switched      interface
16         16         2.2.2.2/32      0             Fa1/0      192.168.34.2
17         Pop Label  3.3.3.3/32      0             Fa1/0      192.168.34.2
18         Pop Label  192.168.23.0/30 0             Fa1/0      192.168.34.2
PE2#
```

```
PE2#show ip cef 1.1.1.1

1.1.1.1/32

  nexthop 192.168.34.2 FastEthernet1/0 label 16
```

```
P#show mpls forwarding-table
Local      Outgoing   Prefix          Bytes Label   Outgoing   Next Hop
Label      Label      or Tunnel Id    Switched      interface
16         Pop Label  2.2.2.2/32      15967         Fa0/0      192.168.23.1
17         Pop Label  4.4.4.4/32      16350         Fa1/0      192.168.34.1
P#
```

When the P router receives something that is tagged with label 17, then it has to be forwarded to 4.4.4.4. It's outgoing label says "pop label" which means to remove the label.

PE2 will receive a regular IP packet (without label) with destination 5.5.5.5 and it will forward it using the routing table towards CE2.

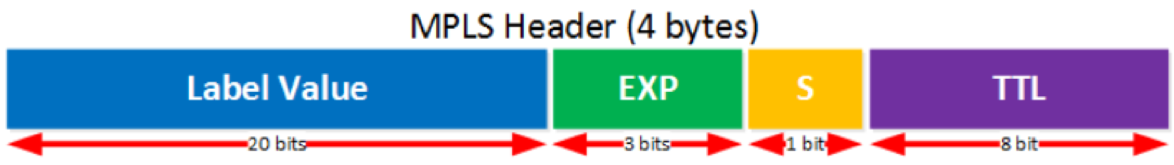
===========

MPLS is like CEF because it generates a table with mappings from incoming labels to outgoing labels and next hop. CEF on the other hand generates a table mapping the incoming packets destination to the outgoing interface and next hop. Both function based on the routing table and are generated on startup, allowing for very fast switching of packets.

On Cisco devices, CEF and MPLS work together. On the ingress edge router the IP destination network of an unlabelled packet will be looked up in the CEF table which contains a mapping to the outgoing label. This is done for efficiency so that the destination doesn't have to be looked up in the CEF table, then again in the label forwarding information base (LFIB).

MPLS Label Format

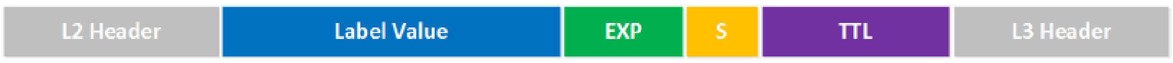
The MPLS header has been standardized, you can find it in RFC 3032. The header is pretty simple, here's what it looks like:



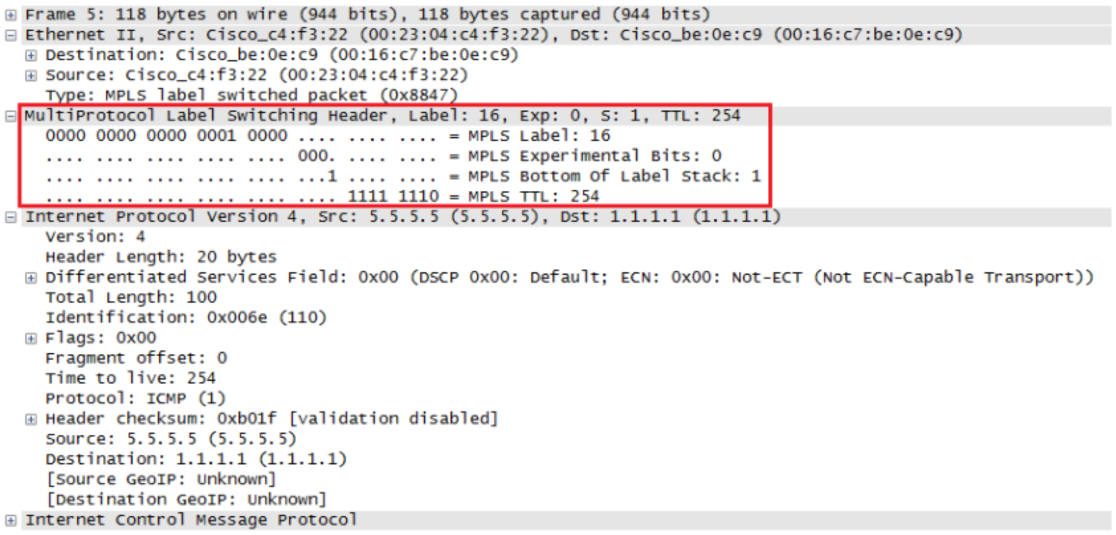
Here's what the different fields are used for:

- **Label value**: the name says it all, this is where you will find the value of the label.
- **EXP**: these are the three experimental bits. These are used for QoS, normally the IP precedence value of the IP packet will be copied here.
- **S**: this is the "bottom of stack" bit. With MPLS it's possible to add more than one label, you'll see why in some of the MPLS VPN lessons. When this bit is set to one, it's the last MPLS header. When it's set to zero then there is one or more MPLS headers left.
- **TTL**: just like in the IP header, this is the time to live field. You you can use this for traces in the MPLS network. Each hop decrements the TTL by one.

The MPLS header is added in between the L2 and L3 header:



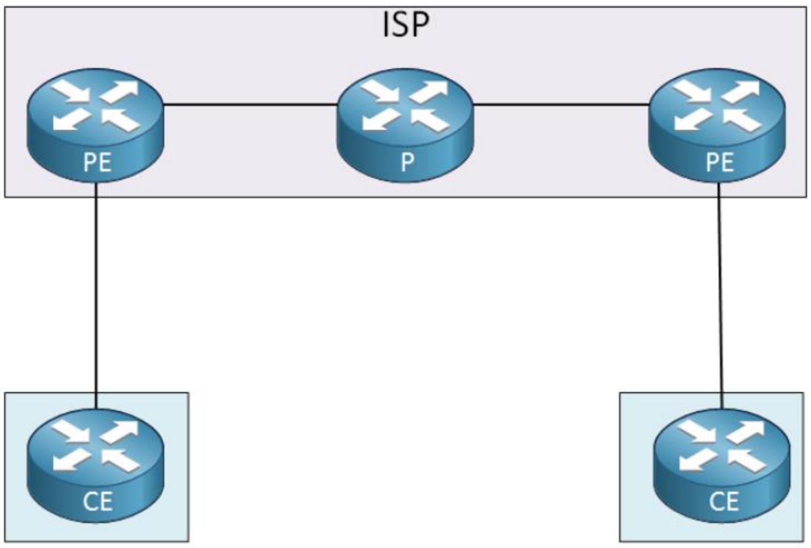
That’s why we call it a “layer 2.5” protocol. Here’s an example of what it looks like in wireshark:



```
⊞ Frame 5: 118 bytes on wire (944 bits), 118 bytes captured (944 bits)
⊟ Ethernet II, Src: Cisco_c4:f3:22 (00:23:04:c4:f3:22), Dst: Cisco_be:0e:c9 (00:16:c7:be:0e:c9)
  ⊞ Destination: Cisco_be:0e:c9 (00:16:c7:be:0e:c9)
  ⊞ Source: Cisco_c4:f3:22 (00:23:04:c4:f3:22)
    Type: MPLS label switched packet (0x8847)
⊟ MultiProtocol Label Switching Header, Label: 16, Exp: 0, S: 1, TTL: 254
    0000 0000 0000 0001 0000 .... .... .... = MPLS Label: 16
    .... .... .... .... .... 000. .... .... = MPLS Experimental Bits: 0
    .... .... .... .... .... ...1 .... .... = MPLS Bottom Of Label Stack: 1
    .... .... .... .... .... .... 1111 1110 = MPLS TTL: 254
⊟ Internet Protocol Version 4, Src: 5.5.5.5 (5.5.5.5), Dst: 1.1.1.1 (1.1.1.1)
    Version: 4
    Header Length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
    Total Length: 100
    Identification: 0x006e (110)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 254
    Protocol: ICMP (1)
  ⊞ Header checksum: 0xb01f [validation disabled]
    Source: 5.5.5.5 (5.5.5.5)
    Destination: 1.1.1.1 (1.1.1.1)
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
⊞ Internet Control Message Protocol
```

MPLS Devices and Operations

Now you know what the MPLS labels look like, let’s talk about a bit about the different devices you will encounter in a MPLS network. Here’s an overview:



Above you will find three different routers:

- **CE (Customer Edge)**: this device is the last device in the customer’s network, it could be a L2 or L3 device. In my picture I used a router but for example, it could be a switch. This device does not use MPLS.
- **PE (Provider Edge)**: this device is owned by the ISP and sits at the edge of the ISP’s network. It has an important role…it receives packets or frames from the customer and will then add a

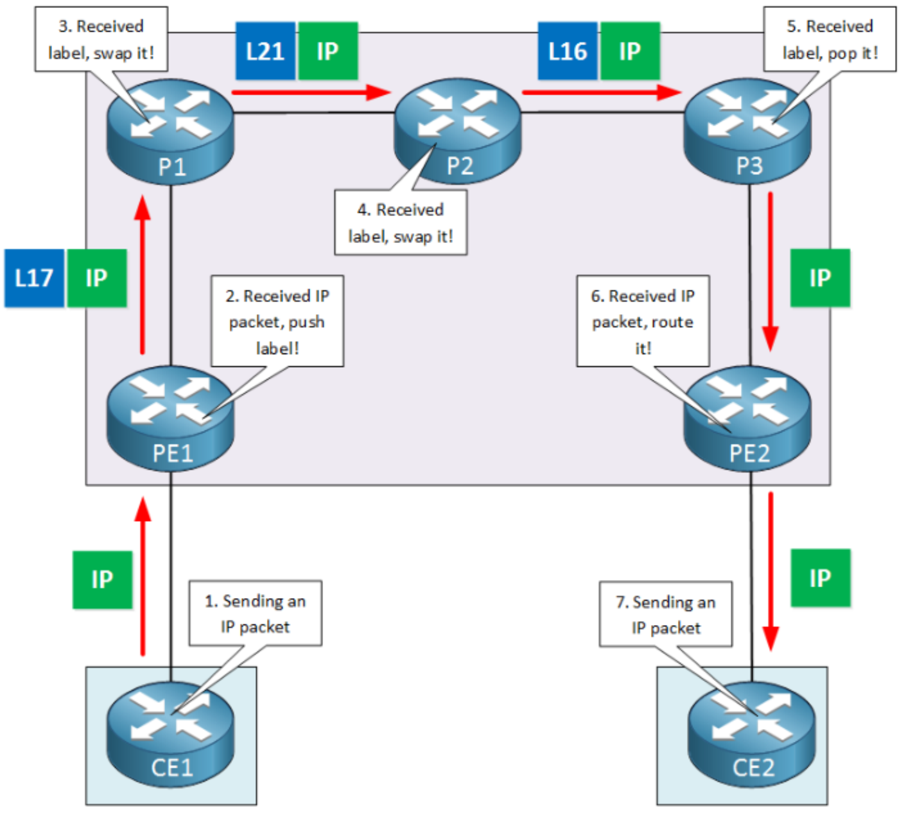
MPLS label to it and forwards towards the core. Another common name for this device is **LER (Label Edge Router)**.

- **P (Provider)**: this device connects to PE routers and other P routers. It has a simple job, it switches packets based on their labels or removes the labels. Another common name for this device is the **LSR (Label Switch Router) or transit router**.

There are three actions we can perform with labels:

- **Label push**: when we add a label to a packet, we call it a label push.
- **Label swap**: replacing a label with another value is called a label swap.
- **Label pop**: removing the label is called a label pop.

Let's look at an example of how labels are pushed, swapped and popped in a MPLS network:


3. Received label, swap it!
L21 IP
L16 IP
5. Received label, pop it!
P1
P2
P3
4. Received label, swap it!
L17 IP
2. Received IP packet, push label!
6. Received IP packet, route it!
IP
PE1
PE2
IP
1. Sending an IP packet
7. Sending an IP packet
IP
CE1
CE2

Let me add some more detail to the picture above:

1. The CE1 router is owned by the customer and connected to the ISP's PE1 router. This device doesn't have a clue what MPLS is and sends an IP packet that should end up at CE2 (another site of the customer).
2. The PE1 router receives the IP packet from the CE1 router, it will push a label on it and forwards it further into the core of the ISP network.
3. P1 receives the labeled packet from PE1, swaps the label and forwards it to P2. Labels are only locally significant, we'll talk more about this in the next lesson.
4. P2 receives the labeled packet from P1, swaps the label and forwards it to P3.
5. P3 receives the labeled packet and will pop the label, forwarding the IP packet to PE2. This is called **penultimate hop popping** and is performed to save PE2 the trouble of looking at the MPLS label.

6. PE2 receives the IP packet and forwards it to the CE2 router.
7. The CE2 router receives the IP packet and the customer is happy.

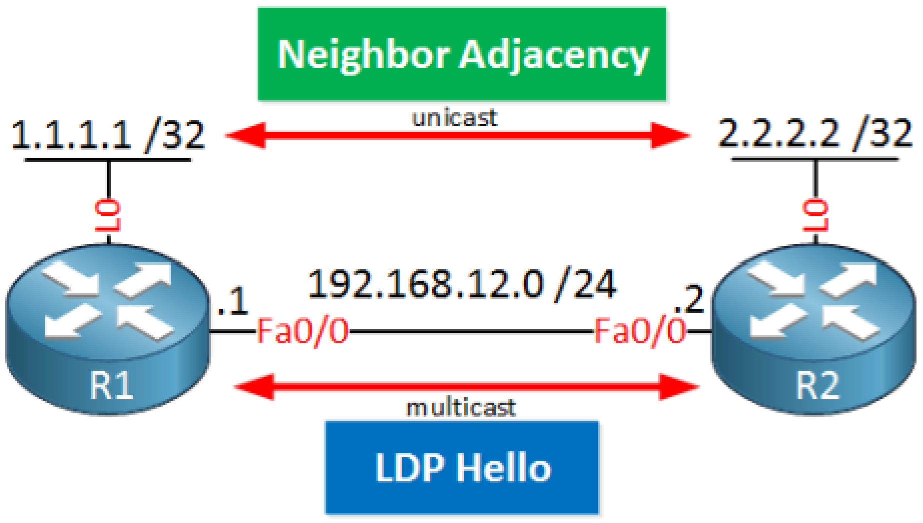
MPLS LDP (Label Distribution Protocol)

LDP is a protocol that automatically generates and exchanges labels between routers. Each router will locally generate labels for its prefixes and will then advertise the label values to its neighbors.

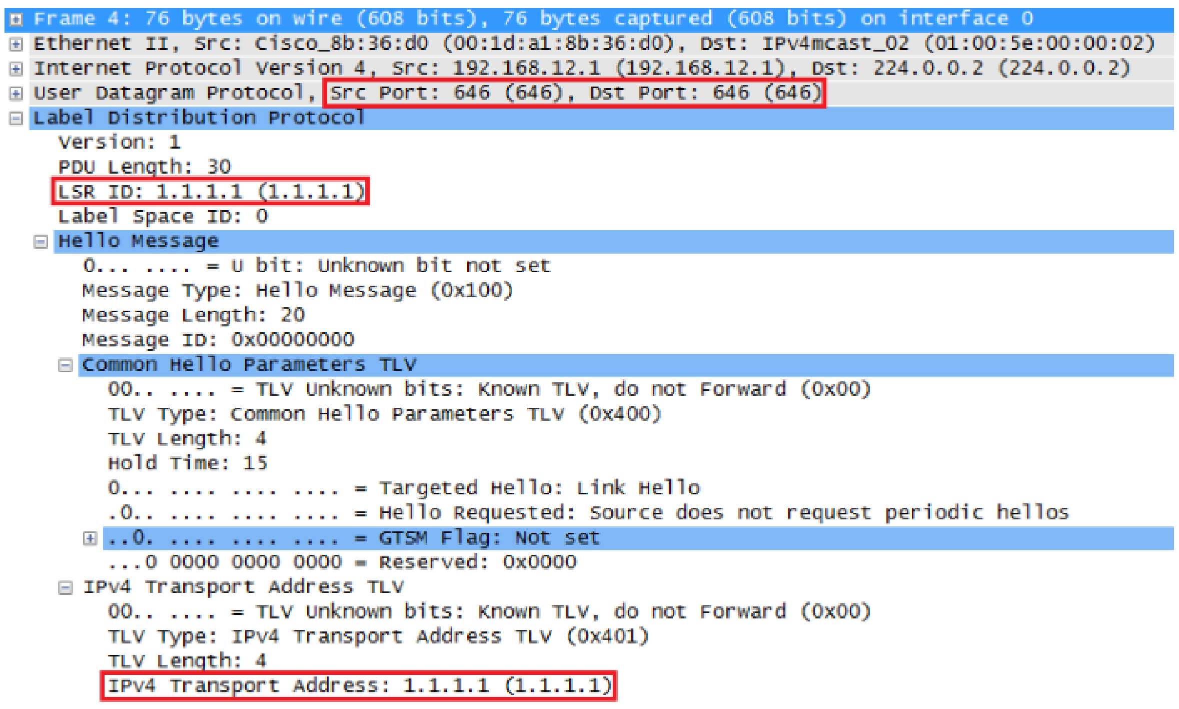
It's a standard, based on Cisco's proprietary TDP (Tag Distribution Protocol). It's pretty much the same story as 802.1Q/ISL or PaGP/LACP. Cisco created a protocol and a standard was created later. Nowadays almost everyone uses LDP instead of TDP.

Like many other protocols, LDP first establishes a **neighbor adjacency** before it exchanges label information. It works a bit different than most protocols though…

First, we send **UDP multicast hello packets** to *discover* other neighbors. Once two routers decide to become neighbors, they build the neighbor adjacency using a **TCP connection**. This connection is then used for the *exchange of label information*. Normally a loopback interface is used for the neighbor adjacency. Here's an example:



The two routers above will send multicast hello packets on their FastEthernet interfaces. Within this hello packet, they will advertise a **transport IP address**. This IP address is then used to establish the TCP connection between the two routers. Here's what the hello packet looks like in wireshark:

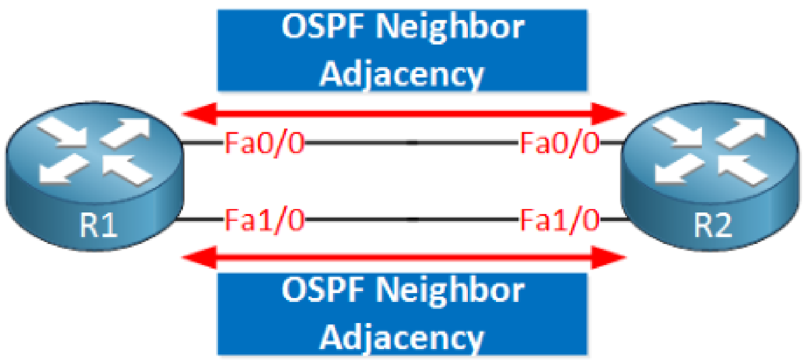


```
⊞ Frame 4: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
⊞ Ethernet II, Src: Cisco_8b:36:d0 (00:1d:a1:8b:36:d0), Dst: IPv4mcast_02 (01:00:5e:00:00:02)
⊞ Internet Protocol Version 4, Src: 192.168.12.1 (192.168.12.1), Dst: 224.0.0.2 (224.0.0.2)
⊞ User Datagram Protocol, Src Port: 646 (646), Dst Port: 646 (646)
⊟ Label Distribution Protocol
    Version: 1
    PDU Length: 30
    LSR ID: 1.1.1.1 (1.1.1.1)
    Label Space ID: 0
  ⊟ Hello Message
      0... .... = U bit: Unknown bit not set
      Message Type: Hello Message (0x100)
      Message Length: 20
      Message ID: 0x00000000
    ⊟ Common Hello Parameters TLV
        00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
        TLV Type: Common Hello Parameters TLV (0x400)
        TLV Length: 4
        Hold Time: 15
        0... .... .... .... = Targeted Hello: Link Hello
        .0.. .... .... .... = Hello Requested: Source does not request periodic hellos
      ⊞ ..0. .... .... .... = GTSM Flag: Not set
        ...0 0000 0000 0000 = Reserved: 0x0000
    ⊟ IPv4 Transport Address TLV
        00.. .... = TLV Unknown bits: Known TLV, do not Forward (0x00)
        TLV Type: IPv4 Transport Address TLV (0x401)
        TLV Length: 4
        IPv4 Transport Address: 1.1.1.1 (1.1.1.1)
```

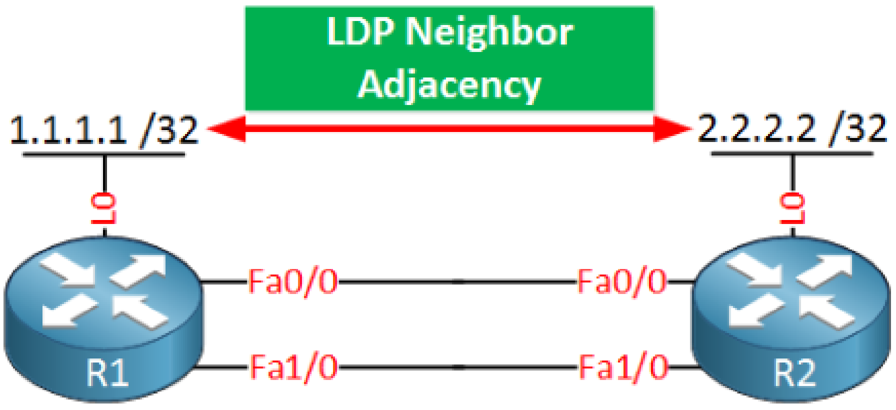
In the capture above you can see a couple of interesting things:

- The hello packets are sent to multicast address **224.0.0.2** using source/destination **UDP port 646**.
- Each router has a unique ID called the **LSR (Label Switch Router) ID**. This is similar to how most protocols select an ID, by default it will select the highest IP address on a loopback interface. If you don't have any loopback interfaces then we will use the highest IP address on a physical interface.
- At the bottom you find the **transport address**. This is what we use to build the actual TCP connection. Like the LSR ID, the router selected the IP address on the loopback interface as the transport address.

This is different compared to how routing protocols like OSPF or EIGRP form neighbor adjacencies. For example, when you run OSPF then your routers will form neighbor adjacencies on all interfaces that run OSPF:



LDP will only form a single neighbor adjacency, no matter how many interfaces you have in between your routers:



LDP Data Plane

All these tables allow us to check the control plane but what about the data plane? We can use a quick traceroute to see if we are using label switching:

```
R1#traceroute 3.3.3.3 source 1.1.1.1
Type escape sequence to abort.
Tracing the route to 3.3.3.3
VRF info: (vrf in name/id, vrf out name/id)
  1 192.168.12.2 [MPLS: Label 201 Exp 0] 0 msec 0 msec 4 msec
  2 192.168.23.3 0 msec 0 msec *
```

When you use traceroute on your MPLS devices then you can see the labels that we use. The path that we use here is called the **LSP (Label Switched Path)**.

Cisco Locator ID Separation Protocol (LISP)

Cisco Locator ID Separation Protocol (LISP) is a mapping and encapsulation protocol, originally developed to address the routing scalability issues on the Internet.

Internet routing tables have grown exponentially, putting a burden on BGP routers. Routing on the Internet is meant to be hierarchical, but because of disaggregation, a full Internet routing table nowadays contains over 800.000 prefixes.

Disaggregation is the *opposite* of aggregation (route summarization). We inject more specific routes when there is an aggregate (summary route). There are two main reasons why this happens:

- **Multihoming**: Customers connect to two different ISPs and advertise their provider-independent address space (PI) to both ISPs.
- **Traffic engineering**: A common practice for ingress traffic engineering is to advertise a more specific route. This works, but it increases the size of the Internet routing table.

You need powerful routers with enough RAM and TCAM to store all prefixes in the Internet routing table. Injecting more specific prefixes also increases the risk of route instability. We need routers with powerful CPUs to process changes in the routing table.

With traditional IP routing, an IP address has two functions:

- **Identity**: To identify the device.
- **Location**: The location of the device in the network; we use this for routing.

LISP separates these two functions of an IP address into two separate functions:

- **Endpoint Identifier (EID)**: Assigned to hosts like computers, laptops, printers, etc.
- **Routing Locators (RLOC):** Assigned to routers. We use the RLOC address to reach EIDs.

Cisco created LISP, but it's not a proprietary solution, it's an open standard, defined in RFC 6830. Originally it was designed for the Internet, but nowadays, you also see LISP in other environments like data centers, IoT, WAN, and the campus (Cisco SD-Access).

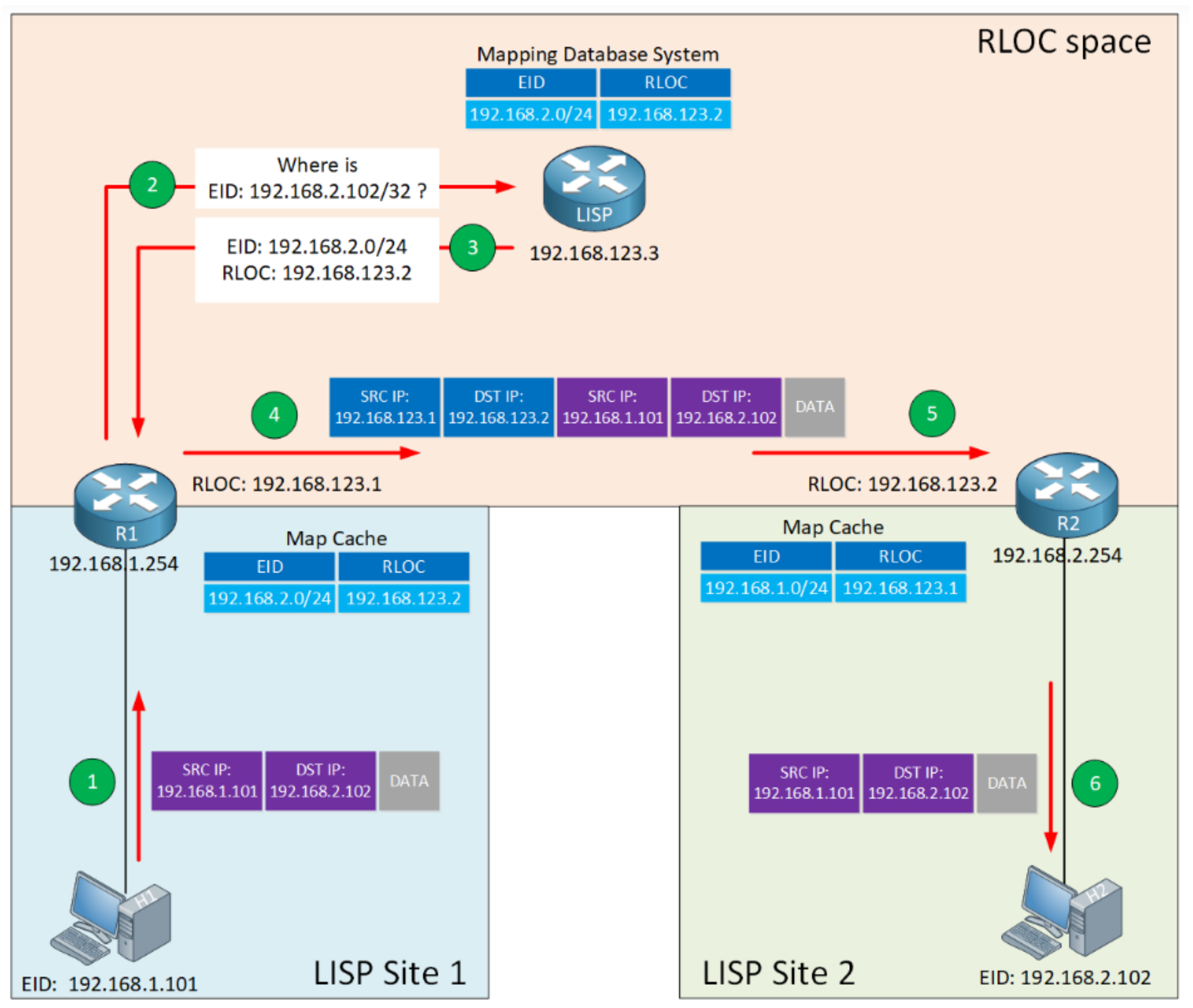
In this lesson, you will learn about the different LISP components and how it operates.

1. LISP Overview

LISP is a **map and encapsulation protocol**. There are three essential environments in a LISP environment:

- **LISP sites**: This is the EID namespace, where EIDs are.
- **non-LISP sites**: This is the RLOC namespace where we find RLOCs. For example, the Internet.
- **LISP mapping service**: This is the infrastructure that takes care of EID-to-RLOC mappings.

Here is a high-level simplified overview of how LISP works:



Let me explain what you see above:

- We have two LISP sites, site 1 and site 2.
  - In each site, there is a host and a router configured to use LISP.
    - The hosts have an EID address:
      - H1 EID 192.168.1.101
      - H2 EID 192.168.2.102.
    - The routers have an RLOC address:
      - R1 RLOC 192.168.123.1
      - R2 RLOC 192.168.123.2
- The RLOC space is a non-LISP area. For example, the Internet.

When H1 wants to send an IP packet to H2, here's what happens:

1. H1 doesn't have anything to do with LISP and sends an IP packet to its default gateway (R1).
2. R1 receives the IP packet and asks the LISP mapping system where it can find EID 192.168.2.102.
3. The mapping system replies with an EID-to-RLOC mapping.
4. R1 now knows that it can reach EID 192.168.2.102 through RLOC 192.168.123.2. The router encapsulates the IP packet with LISP encapsulation and transmits the packet.

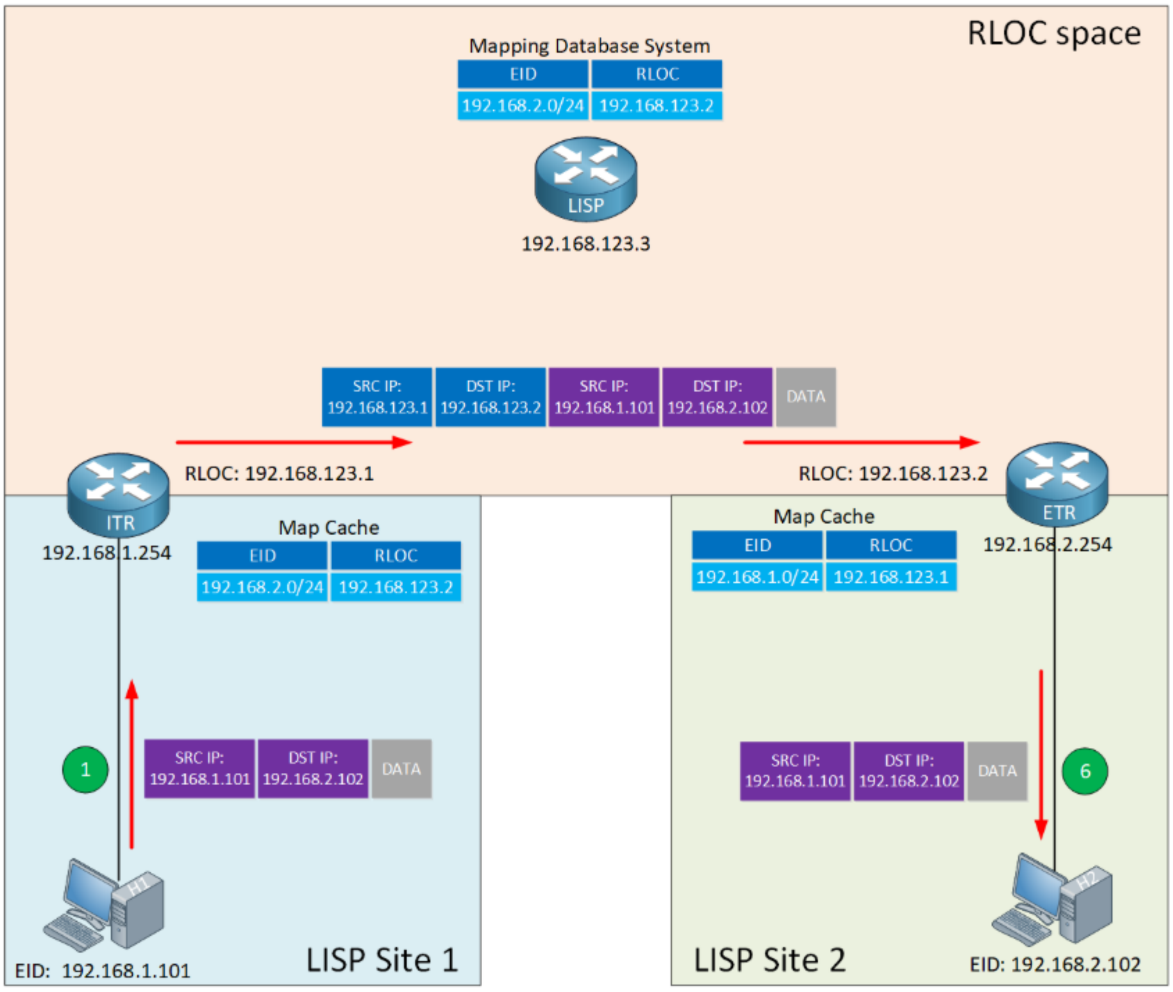
5. R2 receives the LISP encapsulated IP packet, de-encapsulates it, and forwards the original IP packet to H2.

A very simplified one-sentence explanation is that LISP is a tunneling protocol that uses a DNS-like system to figure out to which router they should send IP packets.

The LISP routers that encapsulate and de-encapsulate have a name:

- **Ingress Tunnel Router (ITR)**: Router, which encapsulates IP packets.
- **Egress Tunnel Router (ETR)**: Router, which de-encapsulates LISP encapsulated IP packets.
- **Tunnel Router (xTR)**: Router which performs both the ITR and ETR functions.

I added the ITR and ETR functions in the picture below:



Keep in mind that the hosts (computers, laptops, printers) don't know *anything* about LISP.

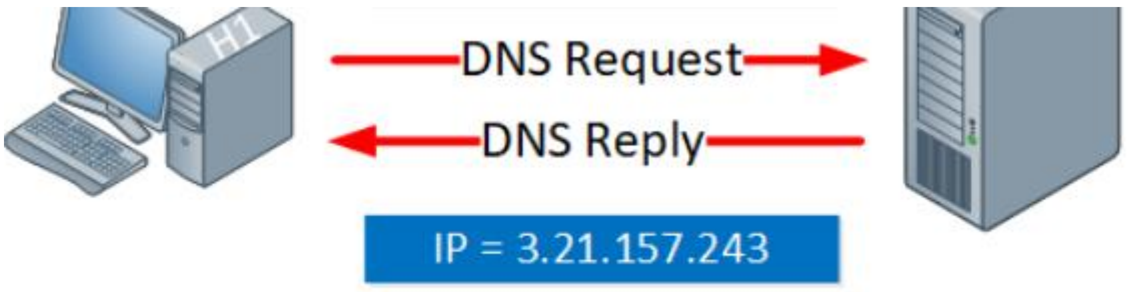
- From the LISP router's perspective: Every endpoint (host) has an EID.
- From the host's perspective: It has a regular IP address. The host doesn't even know what LISP is.

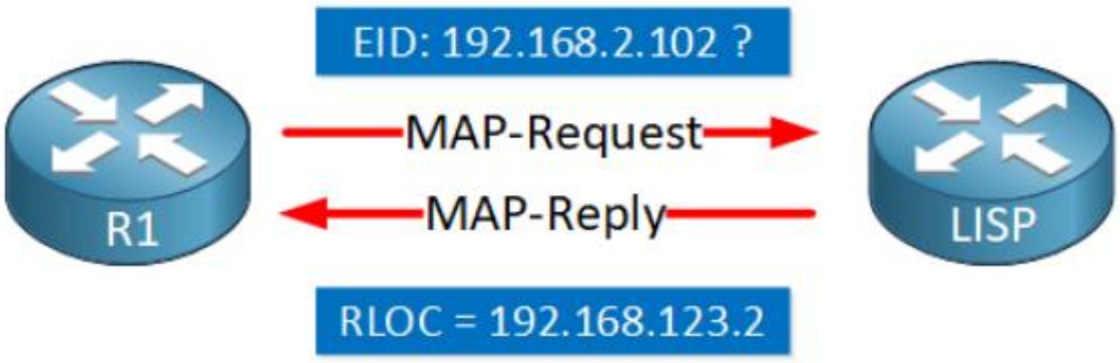
Now you know the basics of LISP, let's add some more detail to this story.

2. LISP Control Plane

The LISP control plane is similar to how DNS works:

- DNS resolves a hostname to an IP address.
- **LISP resolves an EID to an RLOC**.

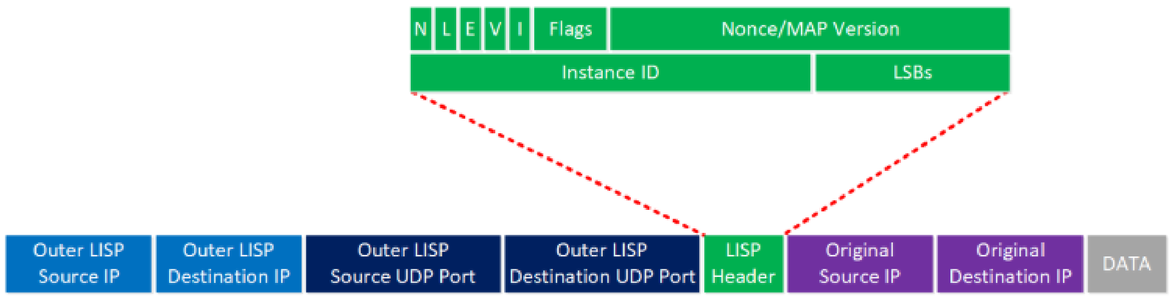




With traditional IP routing, we install prefixes in the routing table. **LISP doesn't install EID-prefixes in the routing table**. Instead, LISP uses a distributed **mapping system** where we **map EIDs to RLOCs**. We store these mappings in a distributed EID-to-RLOC database. When an ITR needs to find an RLOC address, it sends a Map-Request query to the mapping system.

3. LISP Data Plane

Once an ITR has figured out which RLOC to use to reach an EID, it encapsulates the IP packet. Let's take a closer look at how LISP encapsulates IP packets:



Let me walk you through the main headers. When the ITR receives the IP packet from a host, it adds the following headers:

- **LISP Header:** This header includes some LISP information needed to forward the packet. I won’t cover every bit and field here, but the instance ID is worth mentioning. The instance ID is a 24-bit value that has a similar function as the Route Distinguisher (RD) in MPLS VPN. The instance ID is a unique identifier, which keeps prefixes apart when you have overlapping (private) EID addresses in your LISP sites.
- **Outer LISP UDP header**: The source port is selected by the ITR to prevent traffic from one LISP site to another LISP site to take the same path if you have equal-cost multipath (ECMP) links to the destination. Different source ports prevent polarization. The destination port is 4341.
- **Outer LISP IP header**: Contains the source and destination RLOC IP addresses needed to route the packet from the ITR to ETR.

EIDs and RLOCs can be **IPv4 or IPv6 addresses,** so the LISP data-plane supports any of the following encapsulation combinations:

- IPv4 RLOCs with IPv4 EIDs.
- IPv4 RLOCs with IPv6 EIDs.
- IPv6 RLOCs with IPv6 EIDs.
- IPv6 RLOCs with IPv4 EIDs.

For a detailed explanation of all fields in the LISP header, check out RFC 6830.

LISP Operation

You now know what LISP RLOCs and EIDs are, what an ITR and ETR do, and that LISP uses a mapping system for the control plane and how LISP encapsulates IP packets on the data plane.

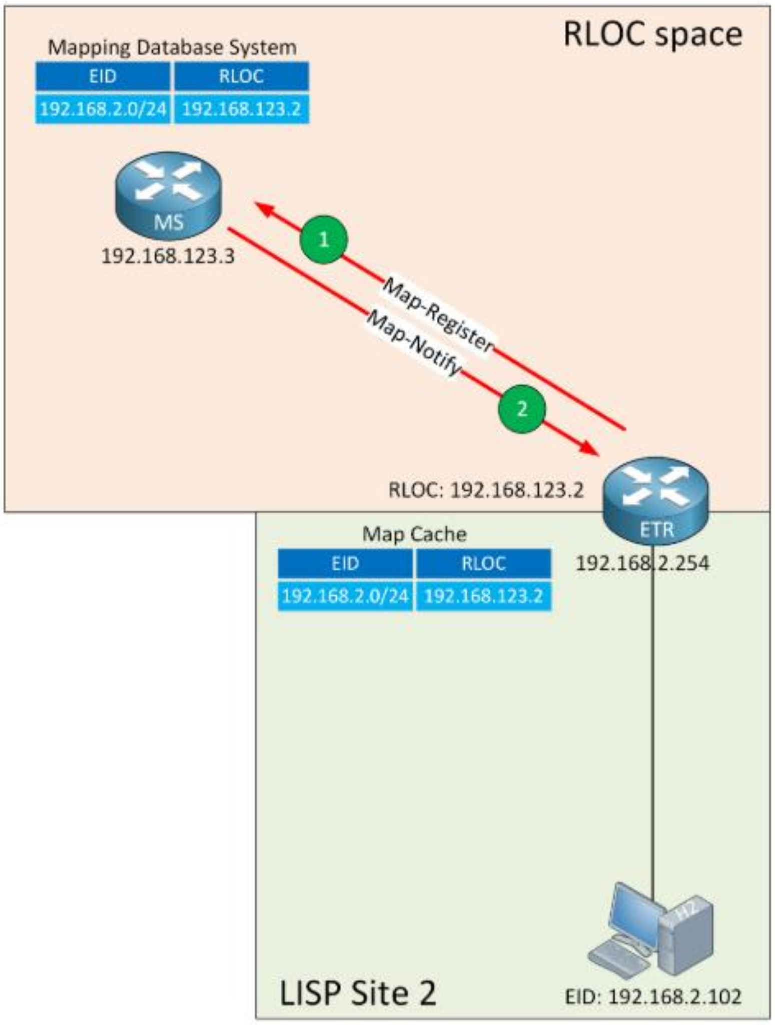
Let's dive even deeper and look at the exact steps of how the LISP mapping system works.

Mapping System

Map-Register and Map-Notify

When you configure an ETR, the router registers the EID-prefixes with the device that is responsible for keeping track of EID-to-RLOC mappings; the **Map-Server (MS)**.

Here is an illustration of the registration process:



The ETR sends a **Map-Register** message to the MS which contains:

- **EID-prefix**: 192.168.2.0/24.

- **RLOC address**: 192.168.123.2.
- **UDP source port**: Chosen by the ETR.
- **UDP destination port**: 4342.

The MS sends a reply called the **Map-Notify** to the ETR and confirms that it received and processed the Map-Register message. The Map-Notify message uses **UDP source and destination port 4342**.

Map-Request and Map-Reply

ITRs use Map-Request messages to request an EID-to-RLOC mapping. The Map-Reply provides the mapping.

Two functions provide a role here:

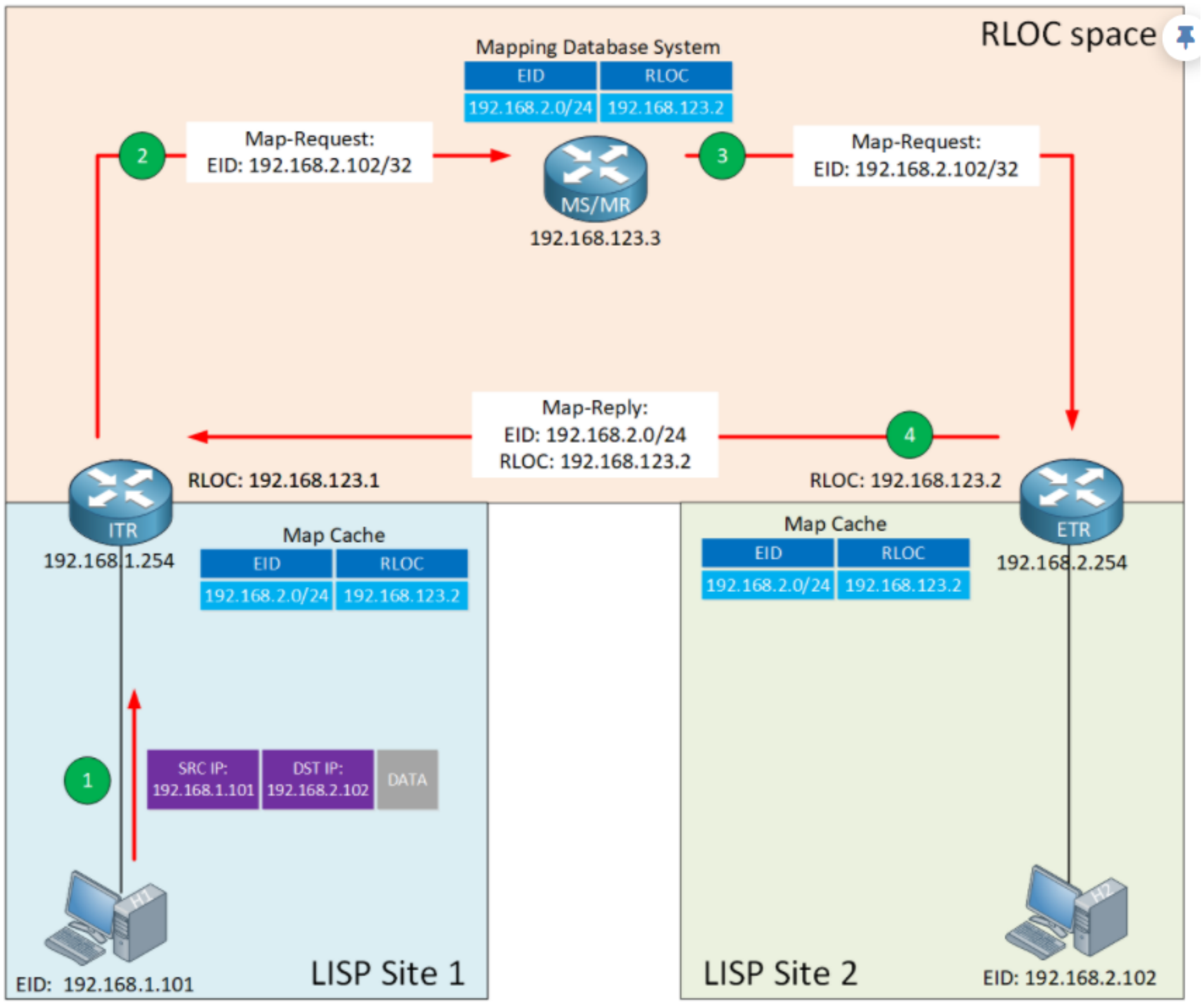
- **MS**
- **MR (Map-Resolver)**

We talked about the MS before. It's the device where ETRs register their EID-prefixes and which stores EID-to-RLOC mappings.

When the ITR needs an EID-to-RLOC mapping, it sends a Map-Request **to the MR**.

When the MR receives a Map-Request, and it has an entry in its local database, then the MR responds with a Map-Reply. When it doesn't have an entry, then the MR forwards the Map-Request to an MS. The MS forwards the Map-Request to the ETR, which answers the Map-Request with a Map-Reply directly.

In smaller environments, we combine the **MR and MS functions** into a single router. We call this an **MR/MS**. Here is an illustration of this process:



Let me walk you through this process:

Step 1

Within a LISP site, we use traditional routing. Let's say H1 in LISP site 1 wants to communicate with H2 in LISP site 2. These hosts don't know anything about LISP. If you had multiple routers in between H1 and the ITR, they would use regular IP routing to reach the ITR.

Step 2

The ITR receives the IP packet from H1 with destination 192.168.2.102. It does a lookup in its FIB table and asks itself the following questions:

- Is there an entry that matches 192.168.2.102? If so:
    - Use regular IP routing.
    - If not, we use LISP encapsulation if any of the following three checks is true:
        - We have a default route.
        - We don't have a route

        - We have a route with Null0 as the next-hop.
- Is the source IP a registered EID-prefix in the local map-cache?
    - If not, forward the packet with regular IP routing.
    - If so, the ITR:
        - Selects a UDP source port.
        - Sets the **UDP destination port to 4342.**
        - Sends an encapsulated Map-Request to the MR for 192.168.2.102.

Step 3

The MR and MS functions are on the same device. When the MR/MS receives the Map-Request, it forwards it to the ETR, which registered the EID-prefix.

Step 4

When the ETR receives the Map-Request, it creates and transmits a Map-Reply which includes:

- EID-to-RLOC mapping:
    - EID: 192.168.2.0/24
    - RLOC: 192.168.123.2
- UDP source port 4342.
- UDP destination port is the one that the ITR selected as the UDP source port for the Map-Request.

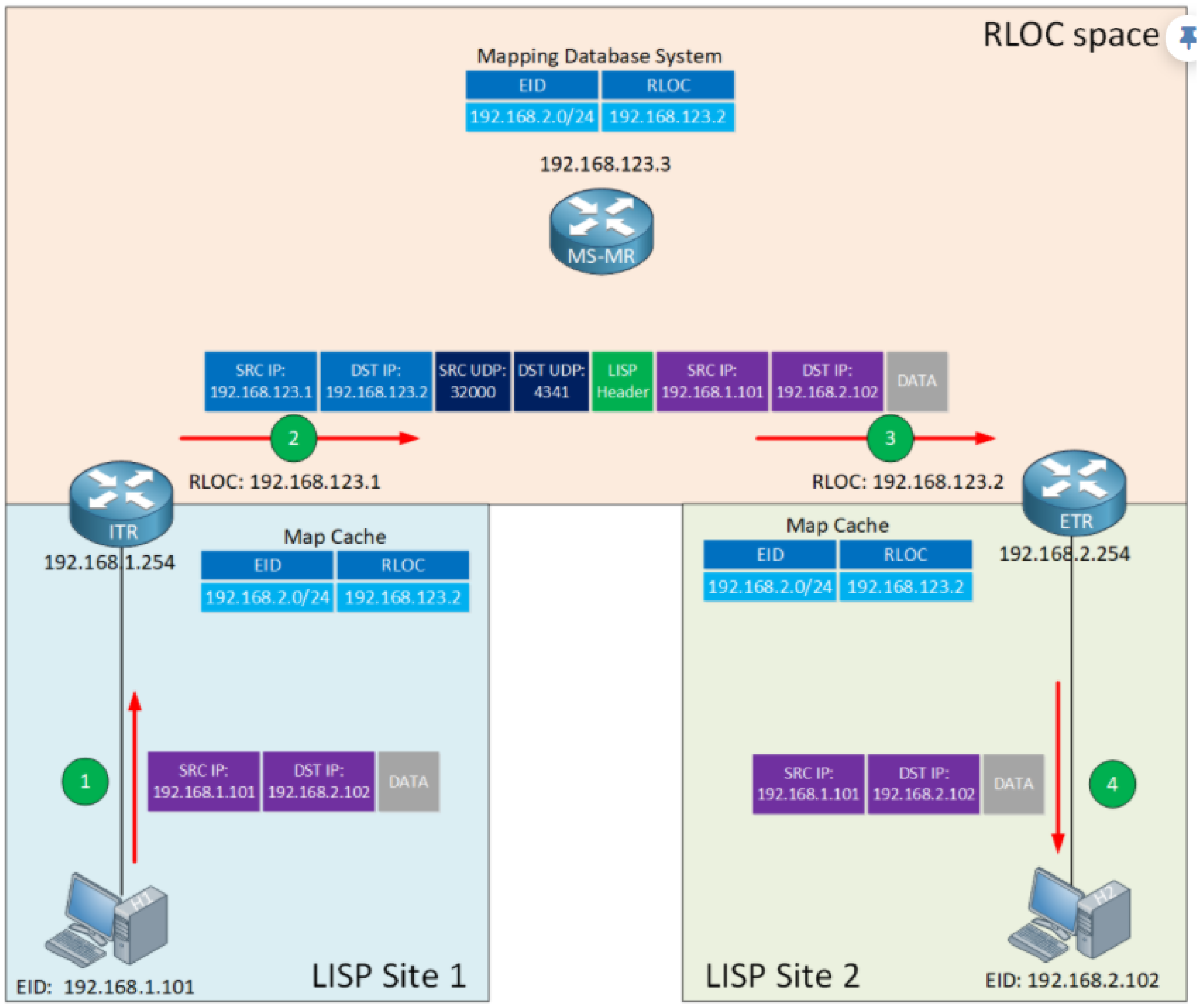
The ETR sends the Map-Reply directly to the ITR. However, it's also possible that the **ETR requests the MS to answer the Map-Request on its behalf**.

To do this, the ETR has to enable the "**proxy Map-Reply" flag (P-bit) in the Map-Register** message.

The ITR receives the Map-Reply from the ETR (or the MS if the ETR requested a proxy Map-Reply) and installs the EID-to-RLOC mapping in its local map-cache. The ITR also programs its FIB and is now ready to forward LISP encapsulated traffic.

LISP Data Path

Let's see what it looks like when the ITR encapsulates the IP packet from H1 with LISP. Here is an overview:



Let me explain the steps:

Step 1

The ITR receives a packet from H1 with EID 192.168.1.101 destined for H2 with EID 192.168.2.102.

Step 2

The ITR checks its FIB, finds a matching entry, encapsulates the IP packet from H1, and transmits the LISP encapsulated IP packet to the ETR:

- **Source IP**: the RLOC address from ITR.
- **Destination IP**: the RLOC address from the ETR.
- **Source UDP port**: selected by ITR.

- **Destination UDP port**: 4341.

Step 3

The ETR receives and de-encapsulates the LISP encapsulated IP packet, then forwards the IP packet to H2.

PETR

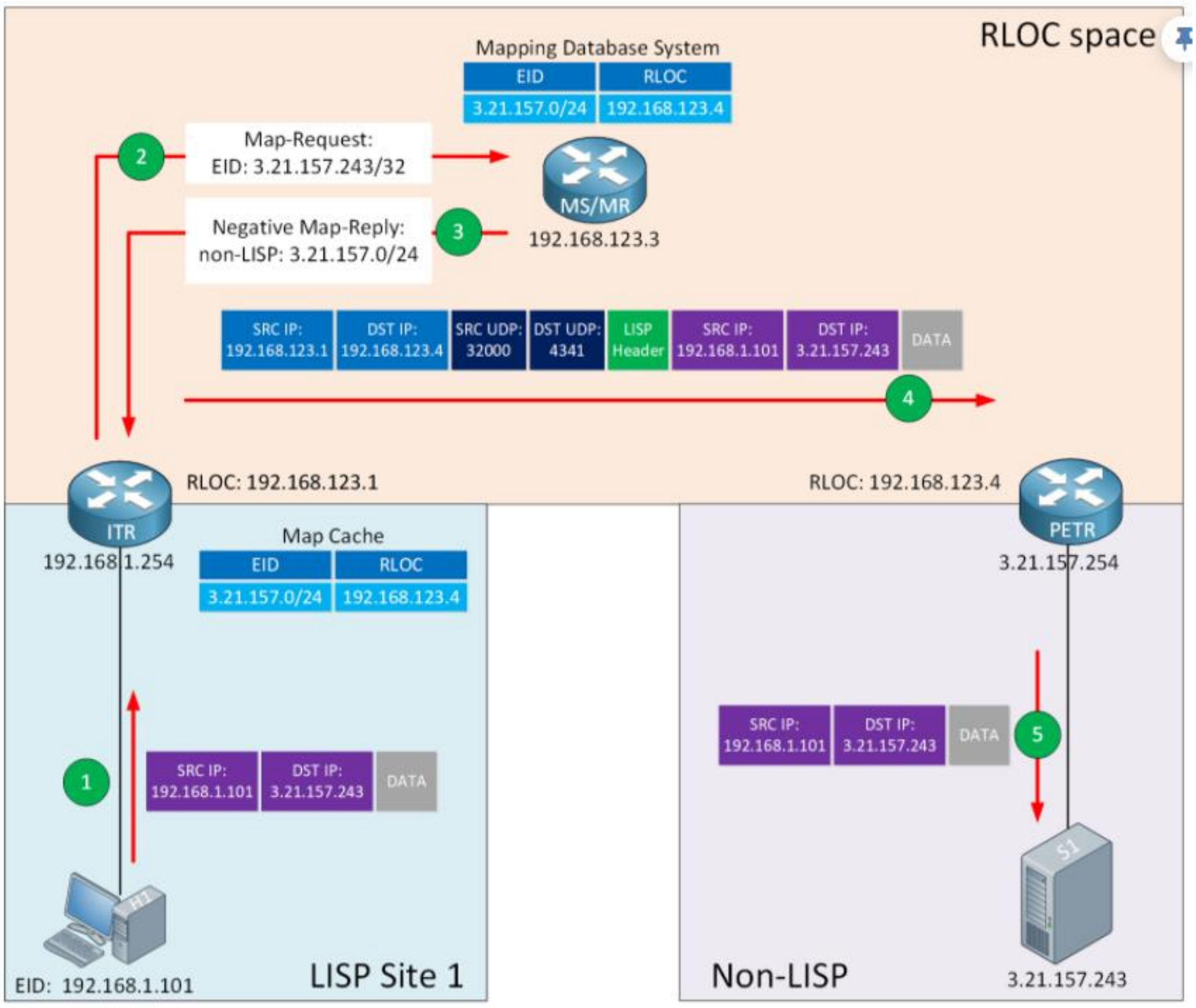
The **Proxy ETR (PETR)** is a router that connects to a **non-LISP site** like the Internet or a Data Center. Because the PETR connects with non-LISP sites, it **doesn't register EID prefixes** with the mapping system.

When an ITR sends a Map-Request, and the EID is not in the mapping database system of the MS, here's what happens:

- The MS calculates the shortest prefix, which matches the requested destination but which doesn't match any LISP EIDs.
- The MS adds the calculated non-LISP prefix in a Negative Map-Reply.
- The ITR can add this prefix in its map-cache and install it in the FIB.

From now on, the ITR can send traffic, which matches the non-LISP prefix directly to the PETR.

Let's look at an example. I replaced LISP site 2 with a non-LISP site and replaced the ETR with a PETR:



Let me explain the steps:

Step 1

H1 wants to send an IP packet to destination IP address 3.21.157.243 (a server on the Internet named S1), so it forwards its packet to the ITR.

Step 2

The ITR doesn't know how to reach 3.21.157.243 and sends a Map-Request to the MR to figure out what RLOC to use.

Step 3

The MR forwards the Map-Request to the MS. The MS replies with a Negative Map-Reply and includes a calculated non-LISP prefix. When the ITR receives the Negative Map-Reply, it installs the non-LISP prefix in its mapping cache and FIB.

Step 4

The ITR encapsulates the IP packet from H1 and forwards it to the PETR. The RLOC adress of the PETR is locally configured on the ITR.

Step 5

The PETR de-encapsulates the LISP encapsulated IP packet and forwards the IP packet to 3.21.157.243.

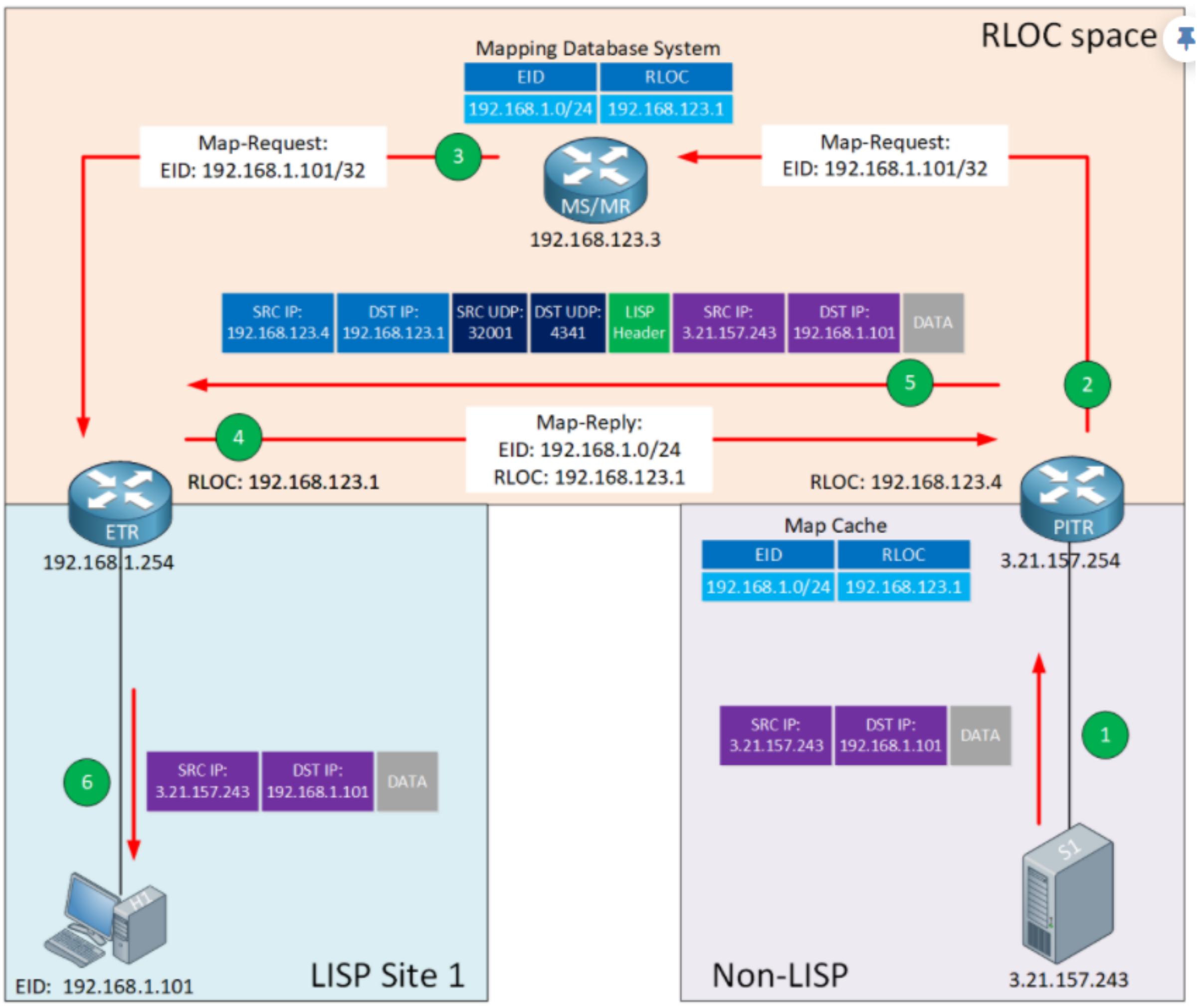
PITR

The **Proxy ITR (PITR) receives traffic from non-LISP sites destined to LISP EIDs.** They behave similarly to ITRs:

- They resolve the mapping for the destination EID.
- They encapsulate and forward traffic to the destination RLOC.

The PITR sends a Map-Request to the MR, and when it receives the Map-Reply, it encapsulates the IP packets with LISP and transmits it to the ETR.

Let’s look at an illustration:



Let me explain what we see above:

Step 1

The PITR receives traffic from S1 with destination 192.168.1.101 (H1).

Step 2

The PITR sends a Map-Request to the MR to figure out what the RLOC is for EID: 192.168.1.101.

Step 3

The MR receives the Map-Request, and forwards it to the MS. The MS forwards the Map-Request to the ETR.

Step 4

The ETR replies to the PITR with a Map-Reply which contains the EID-to-RLOC mapping:

- EID: 192.168.1.101

- RLOC: 192.168.123.253

Step 5

The PITR encapsulates the IP packet and forwards it to the ETR.

Step 6

The ETR receives the LISP encapsulated IP packet, de-encapsulates it, and forwards the IP packet to H1.

A router that performs both the PETR and PITR functions is a **Proxy xTR (PxTR) router**.

Introduction to Virtual Extensible LAN (VXLAN)

Virtual eXtensible Local Area Network (VXLAN) is a tunneling protocol that tunnels Ethernet (layer 2) traffic over an IP (layer 3) network.

Traditional layer 2 networks have issues because of three main reasons:

- **Spanning-tree.**
- **Limited amount of VLANs.**
- **Large MAC address tables.**

Spanning-tree blocks any redundant links to avoid loops. Blocking links to create a loop-free topology gets the job done, but it also means we pay for links we can't use. We could switch to a layer 3 network, but some technology requires layer 2 networking.

The **VLAN ID is 12-bit,** which means we can create 4094 VLANs (0 and 4095 are reserved). Only 4094 available VLANs can be an issue for data centers. For example, imagine we have a service provider with 500 customers. With 4094 available VLANs, they can only offer 8 VLANs to each customer.

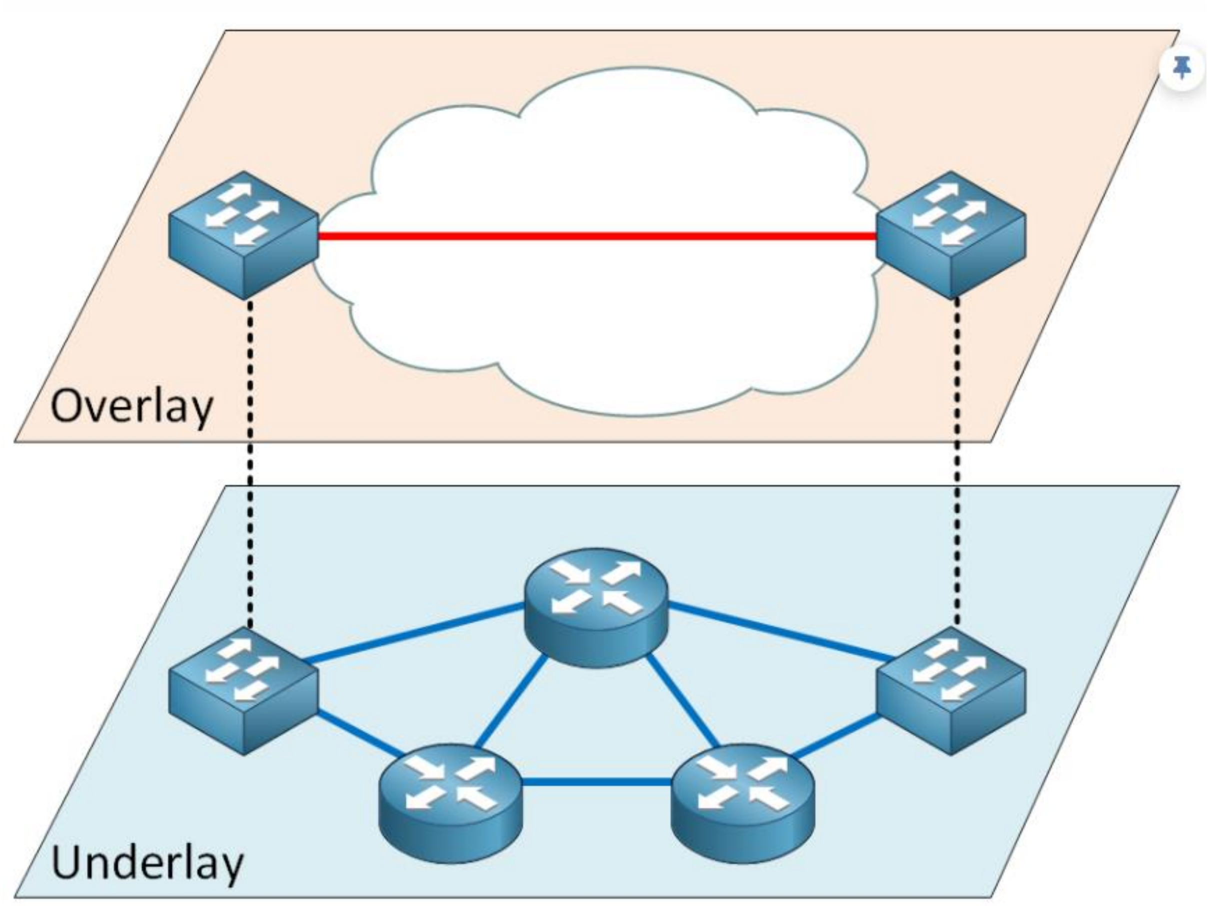
Because of server virtualization, the number of addresses in the MAC address tables of our switches has grown exponentially. Before server virtualization, a switch only had to learn one MAC address per switchport. With server virtualization, we run many virtual machines (VM) or containers on a single physical server. Each VM has a virtual NIC and a virtual MAC address. The switch has to learn **many MAC addresses on a single switchport**.

A Top of Rack (ToR) switch in a data center could connect to 24 or 48 physical servers. A data center could have many racks, so each switch has to store the MAC addresses of all VMs that communicate with each other. We require much larger MAC address tables compared to networks without server virtualization.

In this lesson, I'll explain what VXLAN is, how it works, and how it solves the above layer 2 issues.

Overlay vs Underlay

VXLAN uses an **overlay and underlay network:**



An overlay network is a **virtual** network that runs on top of a **physical** underlay network. Even if you never heard about this terminology before, you have probably seen it. A GRE tunnel is a simple example of an overlay network. The GRE tunnel runs on top of a physical underlay network.

With VXLAN, the overlay is a layer 2 Ethernet network. The underlay network is a **layer 3 IP network**. Another name for the underlay network is a *transport network*.

The underlay network is simple; its only job is to get packets from A to B. We don't use any layer 2 here, **only layer 3**. When we use layer 3, we can use an IGP like OSPF or EIGRP and load balance traffic on redundant links.

Another advantage is that the overlay and underlay network are independent. The overlay network is virtual and requires an underlay network, but whatever changes you make in the overlay network won't affect the underlay network. You can add and remove links in the underlay network, and as long as your routing protocol can reach the destination, your overlay network will remain unchanged.

VNI

The **VXLAN Network Identifier (VNI)** identifies the VXLAN and has a similar function as the VLAN ID for regular VLANs. We use **24 bits for the VNI,** which means we can create 16,777,215 (~16 million) VXLANs. That's a lot, compared to those 4094 VLANs with a 12-bit VLAN ID. We can create plenty of VXLANs, which means a large service provider with even thousands of customers can use as many VXLANs per customer as needed.

VTEP

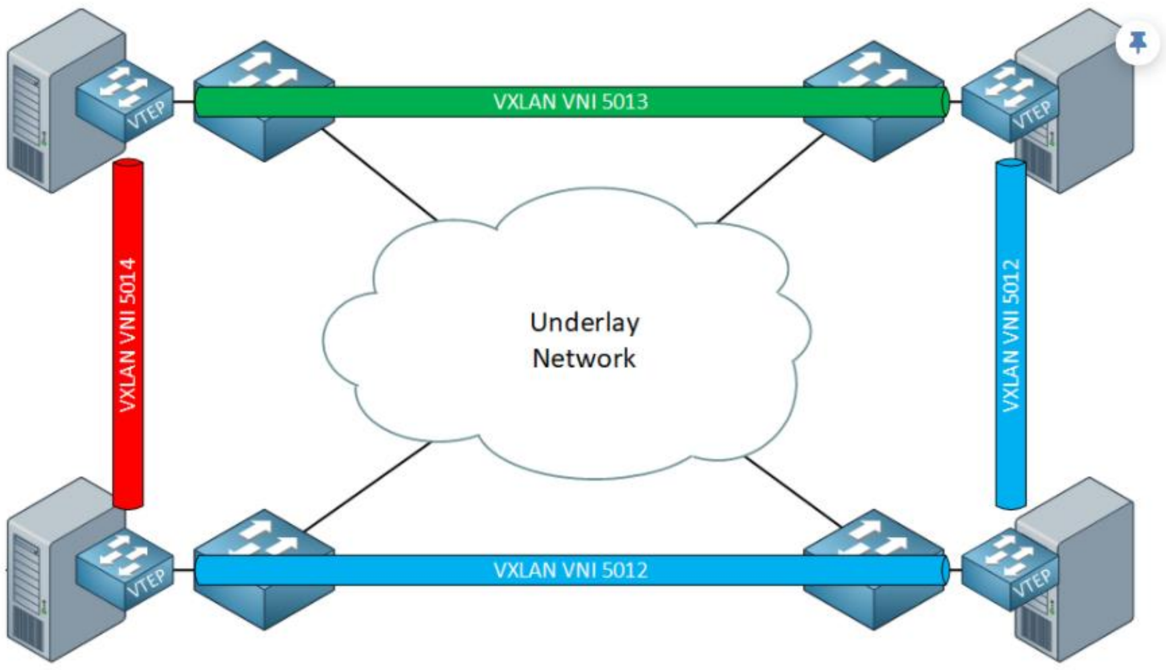
The **VXLAN tunnel endpoint (VTEP)** is the device that's responsible for encapsulating and de-encapsulating layer 2 traffic. This device is the connection between the overlay and the underlay network. The VTEP comes in two forms:

- **Software (host-based)**
- **Hardware (gateway)**

Let's look at these two options.

Software

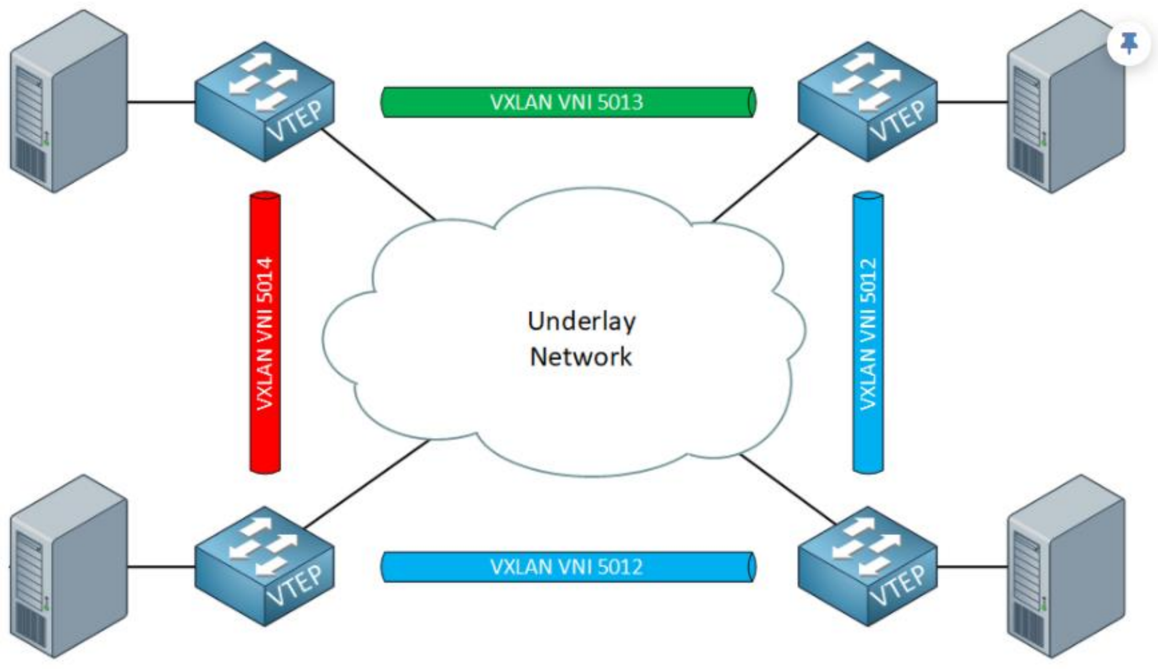
When I'm talking about hosts, I mean hypervisors like VMWare's ESXi or Microsoft's Hyper-V. These hypervisors use virtual switches, and some of them support VXLAN. Here's an illustration:



The VXLAN tunnels are between the virtual switches of the hypervisors. The underlay network is unaware of VXLAN.

Hardware

A hardware VTEP is a router, switch, or firewall which supports VXLAN. We also call a hardware VTEP a **VXLAN gateway** because it combines a regular VLAN and VXLAN segment into a single layer 2 domain. Some switches have VXLAN support with **ASICs, offering better VXLAN performance** than a software VTEP. Here's what it looks like:



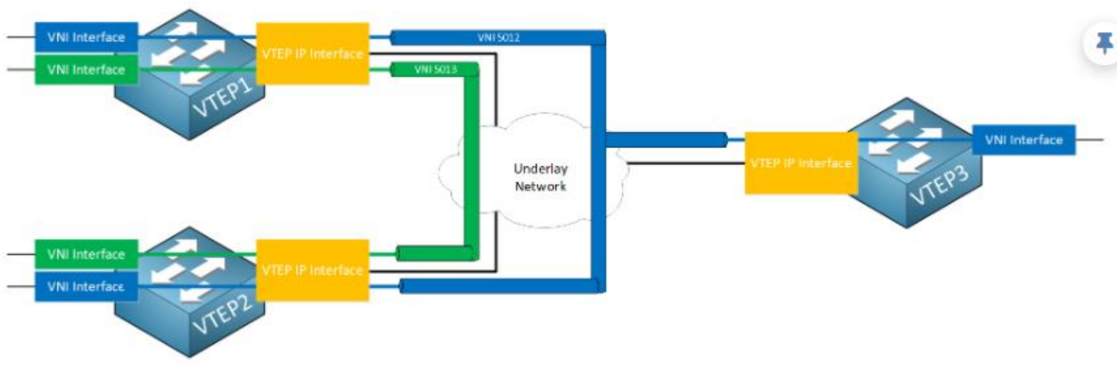
In the above picture, the VXLAN tunnels are between the physical switches. The devices that connect to the physical switches are unaware of VXLAN.

Interfaces

Each VTEP has two interfaces types:

- **VTEP IP interface:** Connects the VTEP to the underlay network with a unique IP address. This interface encapsulates and de-encapsulates Ethernet frames.
- **VNI interface:** A virtual interface that keeps network traffic separated on the physical interface. Similar to an SVI interface.

A VTEP can have **multiple VNI interfaces, but they associate with the same VTEP IP interface**. Here's a picture to help you visualize this:

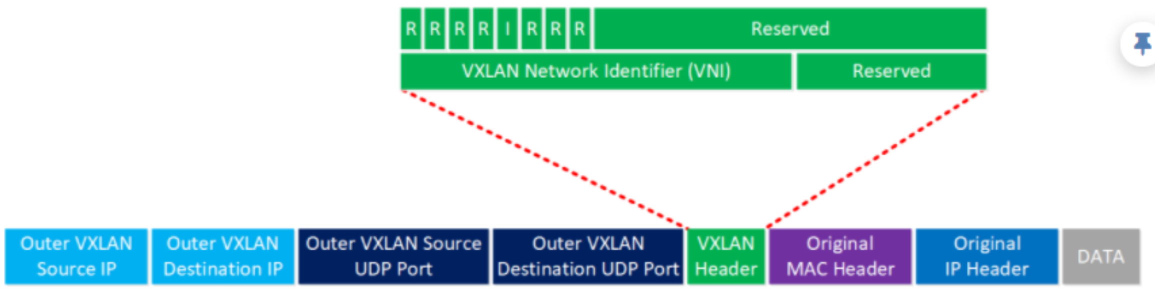


Let me explain what you see above:

- We have three VTEP devices, and each VTEP has a VTEP IP interface that connects to the underlay network.
- All VTEP devices have a VNI interface for VNI 5012 to create a layer 2 segment.
- VTEP1 and VTEP2 also have another VNI interface for VNI 5013 to create another layer 2 segment.

VXLAN Frame Format

Let's take a closer look at the VXLAN frame and header:

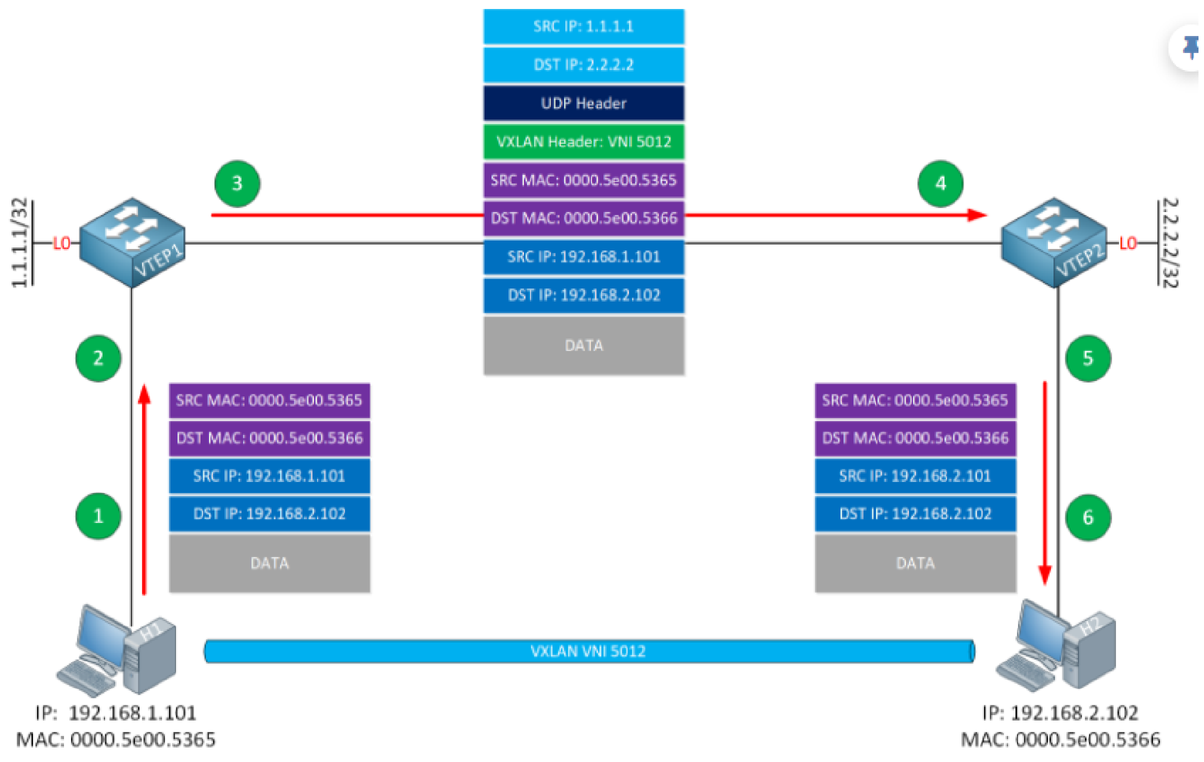

R R R R I R R R
Reserved
VXLAN Network Identifier (VNI)
Reserved
Outer VXLAN Source IP
Outer VXLAN Destination IP
Outer VXLAN Source UDP Port
Outer VXLAN Destination UDP Port
VXLAN Header
Original MAC Header
Original IP Header
DATA

When a VTEP encapsulates an Ethernet frame, it adds a VXLAN header. In this header, we find the VNI and some flags.

The official UDP port number for VXLAN is **4789**. However, it's possible that you also run into UDP port 8472. When VXLAN was first implemented in Linux, there was no official port number yet, and many vendors used port 8472.

Packet Walkthrough

Let's look at an example of how VXLAN encapsulates and de-encapsulates an Ethernet frame. Here's the topology:



H1 and H2 are regular hosts and unaware of VXLAN. VTEP1 and VTEP2 are two switches that act as VTEP devices. We use VNI 5012 to encapsulate Ethernet frames between H1 and H2.

Let me walk you through this process:

- H1 transmits an Ethernet frame, destined for H2.
- VTEP1 receives the Ethernet frame on its VNI interface and performs the following actions:
    - Look up the VNI (5012 in my example) to which H1 is attached.
    - Find the mapping between the destination MAC address and remote VTEP IP address.
    - Add the VXLAN header with VNI 5012.
    - Add the UDP header.
    - Add the outer IP header and set the VTEP IP addresses of VTEP1 and VTEP2.
    - Transmit the IP packet on the underlay network.
- VTEP2 receives the IP packet on its VTEP interface and performs the following actions:
    - De-encapsulate the IP packet.
    - Verify whether the VNI is correct and check if there is a host that uses the destination MAC address.
    - Forward the original Ethernet frame towards H2.
- H2 receives the Ethernet frame.

Control Plane

In the packet walkthrough example, I explained that the VTEP device looks up the mapping to figure out what VTEP IP address to use to reach the destination MAC address. I didn't explain how VTEP1 learned this mapping information. Let's see how this works.

With a traditional VLAN, the first time two hosts communicate with each other, it goes like this:

- H1 sends an ARP request.
- Switches in between H1 and H2 learn the MAC address of H1.
- Switches flood the ARP request.
- H2 receives the ARP request.
- H2 answers with an ARP reply.
- Switches learn the MAC address of H2.

With VXLAN, each VTEP has a VXLAN mapping (forwarding) table that maps a destination MAC address to a remote VTEP IP address. How do VTEP devices learn MAC addresses? There are different control plane solutions. Cisco supports these four options:

- VXLAN with static unicast VXLAN tunnels.
- VXLAN with multicast underlay.
- VXLAN with MP-BGP EVPN.
- VXLAN with LISP.

The first option is simple. You manually configure the VXLAN mapping table. This works, but it's not a scalable solution. The VXLAN standard describes the second solution, where we use a multicast "flood and learn" solution on the underlay.

Here's how it works:

- Each VNI maps to a multicast group.
- The VTEP devices join the multicast group.
- When VTEP1 receives the ARP request, it transmits it to the multicast group.
- VTEP2 receives the ARP request and learns the MAC address of H1.
- VTEP2 stores the MAC address of H1 and the IP address of VTEP1 in the mapping table.
- When VTEP2 receives the ARP reply from H2, it uses the information in the mapping table to send a unicast packet to VTEP1.

The MP-BGP EVPN solution is popular in data centers and private clouds. VXLAN with LISP is a popular choice for campus networks. For example, Cisco's SD-Access uses VXLAN with LISP on the control plane.