

# **CCIE**

# **Routing and Switching**

# **Quick Review Kit**

# **Ver.2 / Vol. 1**



**By: Krzysztof Załęski**  
**CCIE R&S #24081**

**ver. 20111227**



## **Copyright information**

CCIE Routing and Switching Quick Review Kit ver.2 vol.1

By Krzysztof Załęski

CCIE R&S #24081, CCVP

<http://www.inetcon.org>

[cshyshtof@gmail.com](mailto:cshyshtof@gmail.com)

### **ver. 20111227**

This Booklet is NOT sponsored by, endorsed by or affiliated with Cisco Systems, Inc.

Cisco, Cisco Systems, CCIE, CCVP, CCIP, CCNP, CCNA, the Cisco Systems logo, the CCVP logo, the CCIE logo are trademarks or registered trademarks of Cisco Systems, Inc. in the United States and certain other countries.

All terms mentioned in this book, known to be trademarks or service marks belong to their appropriate right owners.

This Booklet is designed to help CCIE candidates to prepare themselves for the CCIE written and/or the lab exam. However, this is not a complete study reference. It is just a series of the author's personal notes, written down during his pre-lab, and further studies, in a form of mind maps, based mainly on CISCO Documentation for IOS 12.4T. The main goal of this material is to provide quick and easy-to-skim method of refreshing one's existing knowledge. All effort has been made to make this Booklet as precise and correct as possible, but no warranty is implied. CCIE candidates are strongly encouraged to prepare themselves using other comprehensive study materials like Cisco Documentation ([www.cisco.com/web/psa/products/index.html](http://www.cisco.com/web/psa/products/index.html)), Cisco Press books ([www.ciscopress.com](http://www.ciscopress.com)), and other well-known vendor's products, before going through this Booklet. The autor of this Booklet takes no responsibility, nor liability to any person or entity with respect to loss of any information or failed tests or exams arising from the information contained in this Booklet.

This Booklet is available for free, and can be freely distributed in the form as is. Selling this Booklet in any printed or electroic form i prohibited. For the most recent version of this document, please visit <http://www.inetcon.org>

Did you enjoy this booklet? Did it help you in achieveing your goal? You can share your gratitude :-> here: <http://amzn.com/w/28VI9LZ9NEJF1>

# Table of Contents

## Volume I

### Data-link technologies

Frame Relay	5
FR Features	6
PPP	7
PPPoE	8

### Switching

VLAN	9
VTP, VTPv3	10
PVST+	11
RSTP	12
MST	13
Port Channel	14
L2 Convergence	14
STP Port Protection	15
SPAN	16
Macro	16
Bridging	16
35x0 features	17
Ethernet frames	17

### IP Services

IPv4	18
ICMP	18
UDP	18
TCP	19
Fragmentation/MTU	20
Static routing	21
Distance	21
Default route	21
Redistribution	21
Policy Based Routing	21
Route Map	21
Adv Obj Tracking	22
ODR	22
GRE	22
Backup interface	22
NTP	23
ARP	23
Neighbor discovery	24
WCCP	24
OER/PfR basics	25
OER/PfR measuring	26
OER/PfR learning	26
OER/PfR policy	27
OER/PfR control	27
Cisco HSRP	28
Cisco GLBP	29
VRRP	29
IDRP	29
DRP	29
NAT part 1	30
NAT part 2	31
Management part 1	32
Management part 2	33
Management part 3	34
DHCP	35
EEM	36

## Volume II

### Routing

#### IPv6

#### Multicast

## Volume III

### Quality-of-Service

#### Security

#### MPLS

(G) – global command  
(IF) – interface command  
(RM) – route-map command  
(CM) – class-map command  
(PM) – policy-map command

**(IF) encapsulation frame-relay [ietf]**

Default encapsulation is CISCO. When connecting Cisco devices with non-Cisco devices, you must use IETF. Cisco device will automatically understand both incoming encapsulations: cisco and ietf. The difference is in FR header following LAPP header (format differences and NLPID values indicating type of payload – only used by endpoint devices)

**(IF) frame-relay interface-dlci <#> [ietf]**

Encapsulation can be set per VC, if some go to Cisco and some for non-cisco device

Ping to local interface travels to the other side of VC and comes back the same way (to remote site first), so RTT is twice larger than pinging remote IP

In NBMA networks local-L2 => remote-L3 mapping is required for proper communication between endpoints (local router must know how to construct L2 header to contact remote IP). Since it's NBMA, broadcast L2 address does not exist like for LAN (ff:ff:ff:ff:ff:ff)

LMI triggers InARP. If LMI is disabled, InARP will not work

InARP starts for every DLCI once LMI reports it with status ACTIVE (Here is my IP on this PVC, what's yours?)

P2P interfaces ignore InARP messages as they only have one DLCI so they know L2 mapping

InARP flows only across VC, it is not forwarded by routers. IP is required on intf to send InARP

**(IF) frame-relay map ip <remote-ip> <dlci> [broadcast]**

Static mapping is required if InARP is not enabled or between spokes. You may also need mapping for local IP to be able to ping it. Static mapping overrides dynamic

**(IF) no frame-relay inverse-arp**

Disable dynamic learning of L2-to-L3 mapping. If subinterfaces are used, it must be configured on subinterfaces, not physical interface

**(IF) no frame-relay inverse-arp ip <dlci>**

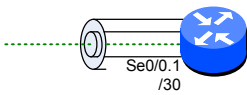
Not only stops sending mapping on that DLCI, but also ignores

**clear frame-relay inarp** **show frame-relay map**

In some cases mappings do not get cleared, reload is required

L2-to-L3 mapping not required, as only one DLCI is allowed on p2p intf.

Broadcast capability is automatically enabled



Only ONE DLCI can be assigned to p2p interface. Adding next one will not overwrite existing one, but error message will be displayed

Non-Broadcast Multi-access (NBMA); many devices in shared subnet, but without broadcast capability. One subnet (ex. /24) with many hosts connected with separate DLCIs.

DLCIs do not have to be manually defined (interface-dlci), as all DLCIs go to physical intf by default.

Can cause problems with split-horizon (many DLCIs, but one interface, so updates are not propagated between spokes)



**interface serial0/0/1 multipoint**

**frame-relay interface-dlci <id>**  
There can be many DLCIs assigned to multipoint subinterface (it acts like physical interface)

Inverse-arp is enabled only on DLCIs assigned to multipoint interface with IP address configured. DLCIs left on physical interface do not run inverse-arp

When InARP is used, it can map DLCI-to-IP only from spokes to hub. InARP is not passed through hub router, so for spokes to communicate separate static mapping is required

Spokes can talk to each other only via Hub. When static mapping is enabled on spoke for hub and other spoke, only mapping for Hub needs broadcast keyword. Enabling broadcast for every static mapping causes multiple packets to be sent to remote destinations (they will be dropped, but bandwidth is wasted)

# Frame-Relay

## Encap.

## InARP

## Point-to-point

## Interface types

## Multipoint

## Hub-and-spoke

## Header

LAPP header – Link Access Procedure for Frame-Relay

DLCI – 10 bits (0-1023) – identifier local to each interface

EA – Extended address – up to 2 additional bytes of header

8	7	6	5	4	3	2	1
DLCI						C/R	EA
DLCI				FECN	BECN	DE	EA

FECN – Forward Explicit Congestion Notification – set towards receiver. For unidirectional traffic BECN cannot be set, so Q922 test frame can be generated by routers as reaction for FECN (FECN reflection)

BECN – Backward Explicit Congestion Notification – set toward sender in returning frames

DE – Discard Eligible – frame may be dropped by the FR switch if DE is set to 1 (during overbooked congestion inside FR network, all frames can be dropped, even with DE=0, but DE=1 is dropped first). Intelligent QoS is required on router interfaces to make sure important traffic fits contracted CIR. DE is set for every packet above CIR

## LMI

**(IF) keepalive <sec>**

Enables LMI. By default enabled

Status Enquiry is sent from DTE to FR Switch once interface comes up. Switch responds with Status describing PVCs

Type-1 – keepalive (10 sec), 3 misses and LMI is down

Type-0 – Complete information about VCs, every 6th message

**q933a:** ITU Annex A, Available DLCIs 16-991 (LMI DLCI: 0)

**ansi:** Annex D, Available DLCIs 16-991 (LMI DLCI: 1023)

**cisco:** Available DLCIs 16-1007 (LMI DLCI: 0)

**(IF) frame-relay lmi-type <type>**

LMI can be autosensed

Any DLCI announced by LMI, not associated with subintf are assumed to be bound to physical intf

**(IF) frame-relay lmi-n391dte <count>**

Full status (type 0) messages frequency (default every 6 cycles)

**show frame-relay pvc**

PVC status

ACTIVE – PVC is working fine on both ends

INACTIVE – other end of PVC is experiencing a problem

DELETED – DLCI is configured locally, but it's not received via LMI (misconfigured on local FR switch)

STATIC – LMI is disabled, mappings are statically configured

## Back2Back

1) The same DLCI on both sides

Disable LMI (**no keepalive**). PVCs will be shown as **STATIC**

Router A and B:  
**(IF) frame-relay interface-dlci 101**

2) If DLCIs are to be different on both sides

Router A:  
**(IF) frame-relay map ip <ip> 102** (encapsulate)  
**(IF) frame-relay interface-dlci 201** (expect)

Router B:  
**(IF) frame-relay map ip <ip> 201** (encapsulate)  
**(IF) frame-relay interface-dlci 102** (expect)

3) Frame-relay switching

**keepalive** must be enabled on both sides

Router A:  
**(G) frame-relay switching**  
**(IF) frame-relay intf-type dce**  
**(IF) frame-relay interface-dlci 201**

Router B:  
**(IF) frame-relay interface-dlci 201**

## FR Switching

**(G) frame-relay switching**

Frame Relay switching must be globally enabled to route PVCs between interfaces

**(IF) frame-relay intf-type dce**

Routing is not allowed between two DTE interfaces, at least one must be DCE

**(IF A) frame-relay route <incoming DLCI> interface <outgoing IF> <outgoing DLCI>**

DLCI routing must be configured bi-directionally, that is on both interfaces. Always configured on physical interface

**(G) connect <name> serial <nr> <dlci> serial <nr> <dlci>**

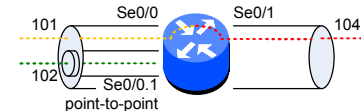
Alternate method of defining how to switch DLCIs

CBWFQ applied on physical interface to do per-VC shaping (match fr-dlci) does NOT work for switched DLCIs

**frame-relay intf-type dce**

**frame-relay route 101 interface Serial0/1 104**

**frame-relay route 104 interface Serial0/0 101**



# FR Features

## Broadcast Queue

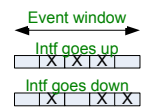
- FR can send broadcast over PVCs. It's not one packet to all destinations, but it's replicated to all PVCs (pseudo-broadcast)
- Managed independently of the normal interface queue
- STP and BPDUs are not transmitted using the broadcast queue
- (IF) frame-relay broadcast-queue <size> <Bps> <packet-rate>**
- The broadcast queue has priority when transmitting at a rate below the configured maximum, so it has a guaranteed minimum bandwidth. Two rate limits are to avoid flooding the interface with broadcasts. Limiting starts when whichever rate hits first.
- The byte rate should be less than (1) N/4 times the minimum remote access rate, where N is the number of DLCIs to which the broadcast must be replicated, and (2) 1/4 the local access rate (B/s)
- Queue size should be set, so that one complete routing update from each protocol and for each DLCI can be stored. As a general rule, start with 20 packets per DLCI
- The defaults for the serial interfaces are: size of 64 packets, 256kbps, and 36pps

## FR Autoinstall

- When the serial interface is connected to the network, the AutoInstall process begins automatically
- Router being configured over FR will send BOOTP request for IP address (DHCP in LAN, SLARP in HDLC)
- AutoInstall using Frame Relay can be initiated over only the first serial interface on the router
- Staging (intermediate) router must have FR map configured. Defined IP will be assigned to remote router
- frame-relay map ip <remote IP> <DLCI> broadcast** (NBMA)
- frame-relay interface-dlci <dlci> protocol ip <remote ip>** (P2P)
- Helper-address on staging router is required if configured router needs to upload config via TFTP. Router with TFTP server should have directed-broadcast enabled on Ethernet
- AutoInstall will attempt to download configuration files in the following order: network-confg, cisco.net.cfg, router-confg, router.cfg, cisconftr.cfg. The process will be repeated 3 times.

## End-to-end Keepalive (EEK)

- If keepalive is rcvd within defined timers, success-event is logged. Otherwise, error-event is logged. To bring up intf, 3 successes in a row must appear. To bring down, any 3 events within event-window
- map-class frame-relay <name>**
- frame-relay end-to-end keepalive mode {reply | request | bidir}**
- frame-relay end-to-end keepalive timer {recv | send} <sec>**
- frame-relay end-to-end keepalive event-window {recv | send} <#>**
- frame-relay end-to-end keepalive error-threshold {recv | send} <#>**
- frame-relay end-to-end keepalive success-events {recv | send} <#>**
- show frame-relay end-to-end keepalive [interface <if> <dlci>]**



## Bridging

- bridge <id> protocol ieee**
- interface <intf>**
- bridge-group <id>**
- frame-relay map bridge <dlci> broadcast**
- Static mapping is required on multipoint interfaces
- Can be used to emulate p2p link on multipoint interface or to enable LFI on FRF.8 links (FR to ATM interworking)

## Fragmentation

- map-class frame-relay <name>**
- frame-relay fragment-size <#>**
- Fragment size = delay \* BW
- Must be added on both sides, as 2 bytes fragmentation header is added
- show frame-relay fragment**
- Legacy – requires shaping with dual FIFO for interleaving
- MLPPP required for FRF.8 FR-to-ATM interworking
- frame-relay fragment <#>**
- Fragmentation configured directly on interface with no FRTS (>12.2.13T)
- IOS automatically creates dual FIFO

## PPPoFR

- Can be used to emulate p2p link on multipoint interface or to enable LFI on FRF.8 links (FR to ATM interworking)
- interface serial0/0**
- frame-relay interface-dlci <dlci> ppp virtual-template <id>**
- interface virtual-template <id>**
- ip address <ip> <mask> | ip unnumbered loopback0**
- interface multilink <ML-id>**
- ppp multilink**
- ppp multilink group <ML-id>**
- interface virtual-template <VT-id>**
- ppp multilink group <ML-id>**
- Virtual-access interface is created (cloned) after virtual-template is bound to DLCI. As this interface is p2p then no L2-to-L3 mapping is required even if used on physical multipoint interface
- Remote peer's /32 IP is shown in routing table as connected (PPP behavior)
- On multipoint interface each DLCI must be assigned to the same virtual-template interface because all endpoints must be in the same subnet. Separate virtual-access interface will be created for each DLCI

Address 0xFF	Control 0x03	Protocol	Payload	FCS
1B	1B	2B		2B

# PPP

**LCP** – to establish, configure, and test the data link connection – mandatory phase

**NCP** – for establishing and configuring different network layer protocols (IPCP, CDPCP) – mandatory phase

Authentication (PAP/CHAP) – optional phase. Authentication method is negotiated during LCP, but authentication itself is after LCP

**(G) no peer neighbor route**  
Peers' IP addresses are sent in IPCP negotiation and they show up as /32 connected networks in addition to /30 subnets. Host routes received from peer can be discarded with this command.

## Features

RTA:  
**(IF) ip address negotiated**

RTB (option A):  
**(IF) peer default ip address <remote ip>**

RTB (option B):  
**(G) ip address-pool local**  
**(G) ip local pool <name> <first IP> <last IP>**  
**(IF) peer default ip address pool <name>**

Address IP can be sent to peer (like DHCP). Such address is always seen as /32 host route

**PAP** (Password Authentication Protocol) is a 2-way authentication method, sending clear-text login and password (request-response). Can be uni- or b-directional

## PAP

**(IF) ppp authentication pap**  
Router with this command requests other side to authenticate with PAP

**(IF) ppp pap sent-username <username> password <password>**  
Send hostname and a password in response to PAP request

**(IF) ppp pap wait**  
The router will not authenticate to a peer that requests PAP authentication until the peer has authenticated itself to the router (bi-directional authentication configuration required)

**(IF) ppp pap refuse [callin]**  
All attempts by the peer to force authentication with PAP are refused. The callin option specifies that the router refuses PAP but still requires the peer to authenticate itself with PAP

**CHAP** is a 3-way handshake authentication method based on challenge-response. No clear-text passwords are sent across the link Done upon initial link establishment and may be repeated any time after the link has been established

## CHAP

**(IF) ppp authentication chap**  
Router with this command requests the otreh side to authenticate with CHAP

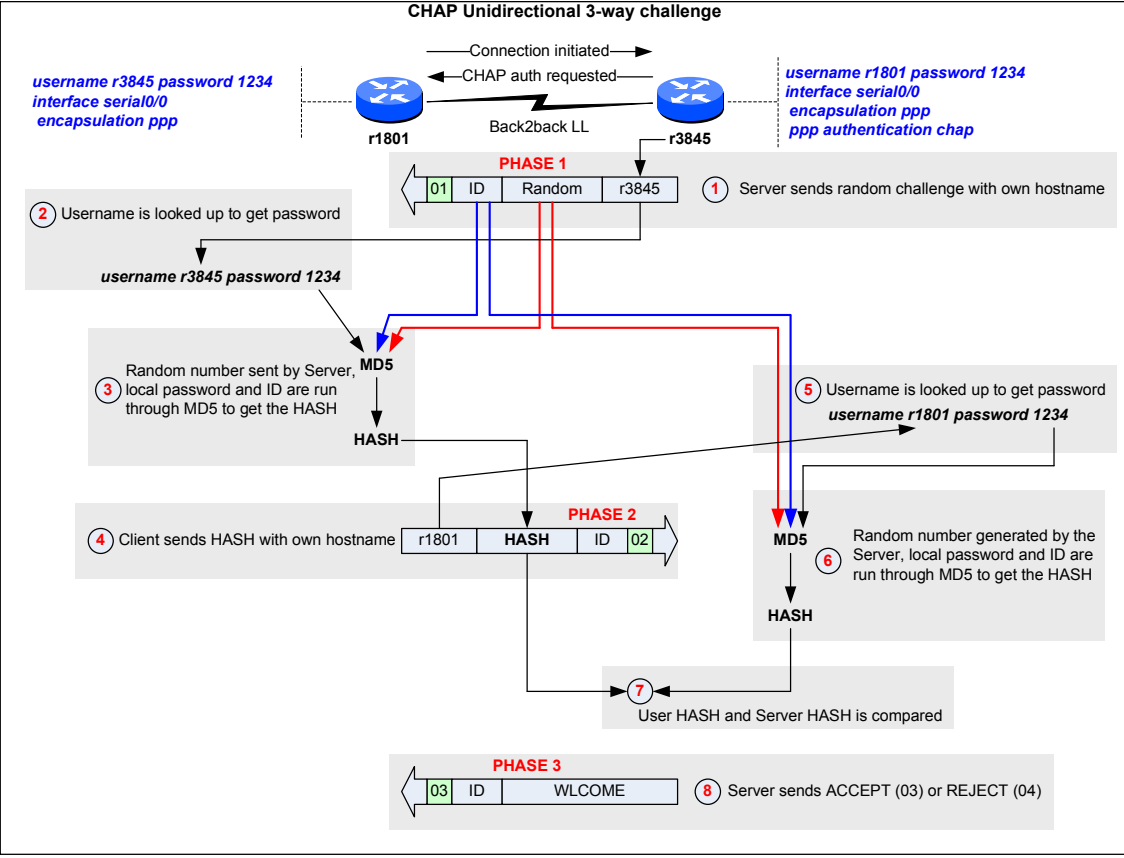
**(IF) ppp chap hostname <name>**  
Send alternate hostname as a challenge. By default, real hostname is sent as username

**(IF) ppp chap password <pass>**  
This password is used if global username is not configured

**(IF) ppp direction [callin | callout]**  
Forces a call direction. Used when a router is confused as to whether the call is incoming or outgoing (when connected back-to-back)

**(IF) ppp chap refuse [callin]**  
All attempts by the peer to force authentication with CHAP are refused. The callin option specifies that the router refuses CHAP but still requires the peer to answer CHAP challenges

**(IF) ppp chap wait**  
The router will not authenticate to a peer that requests CHAP authentication until the peer has authenticated itself to the router CHAP will fail if hostnames are the same on both sides



### PAP/CHAP Authentication

One way authentication. If two-way PAP authentication is required it has to be configured the oposite way

<p>Client:</p> <pre>hostname R1 interface serial0/0 ! Client sends username and password via PAP ppp pap sent-username R1 password cisco</pre>	<p>Server:</p> <pre>hostname R2 username R1 password cisco interface serial0/0 ! server requests client to authenticate with PAP ppp authentication pap</pre>
<p>Two-way authentication, R2 requests R1 to auth using PAP, and R1 requests R2 to auth using CHAP</p> <p>Client:</p> <pre>hostname R1 username R2 password cisco interface serial0/0 ! Client sends username and password via PAP ppp pap sent-username R1 password cisco</pre> <p>! Client requests server to authen. with CHAP ppp authentication chap</p>	<p>Server:</p> <pre>hostname R2 username R1 password cisco interface serial0/0 ! server requests client to authenticate with PAP ppp authentication pap</pre> <p>! server sends CHAP response using user R1</p>

There is a Discovery stage (Ethertype 0x8863) and a PPP Session stage (Ethertype 0x8864)

When discovery completes, both peers know PPPoE SESSION\_ID and peers' MAC which together define the PPPoE session uniquely

The client broadcasts a PPPoE Active Discovery Initiation (PADI) packet. PADI (with PPPoE header) MUST NOT exceed 1484 octets (leave sufficient room for relay agent to add a Relay-Session-Id TAG)

PADI transmit interval is doubled for every successive PADI that does not evoke response, until max is reached

Concentrator replies with PPPoE Active Discovery Offer (PADO) packet to the client containing one AC-Name TAG with Concentrator's name, a Service-Name TAG identical to the one in the PADI, and any number of other Service-Name TAGs indicating other services that the Access Concentrator offers.

Host chooses one reply (based on concentrator name or on services offered). The host then sends PPPoE Active Discovery Request (PADR) packet to the concentrator that it has chosen

Concentrator responds with PPPoE Active Discovery Session-confirmation (PADS) packet with SESSION\_ID generated. Virtual access interface is created that will negotiate PPP

The PPPoE Active Discovery Terminate (PADT) packet may be sent anytime after a session is established to indicate that a PPPoE session has been terminated

**Discovery**

**PPPoE**

**1. Virtual template**

`interface virtual-template <number>`  
`ip unnumbered <ethernet>`  
*(IF) peer default ip address dhcp-pool <name>*  
 Assign IP address to a client from local DHCP pool

**2. Broadband Group**

*(G) bba-group pppoe <name> | global*  
 Create BBA group to be used to establish PPPoE sessions. If global group is created it is used by all ports with PPPoE enabled where group is not specified.

*(BBA) virtual-template <number>*  
 Specifies the virtual template interface to use to clone Virtual Access Interfaces

**3. Enable on Interface**

*(IF) pppoe enable [group <name>]*  
 Assign PPPoE profile to an Ethernet interface. Interface will use global PPPoE profile if group is not specified

*(IF) protocol pppoe [group <name>]*  
 Assign PPPoE profile to VLAN subinterface (`encapsulation dot1q <vlan>`). Interface will use global PPPoE profile if group is not specified

*(IF) vlan-id dot1q <vlan-id> or vlan-range dot1q <start> <end>*  
`pppoe enable [group <group-name>]`  
 Enables PPPoE sessions over a specific VLAN or a range of VLANs on physical ethernet interface

`vpdn enable`  
`vpdn-group <name>`  
`request-dialin`  
`protocol pppoe`  
 Configure VPDN group – legacy, prior 12.2(13)T

**Client**

`interface dialer <number>`  
`encapsulation ppp`  
`ip mtu <mtu>` ! recommended 1492 for 8 byte PPPoE header  
`ip address negotiated`  
`dialer pool <number>`  
`dialer-group <group-number>`

*(G) dialer-list <dialer-group> protocol ip {permit | list <acl>}*  
 Defines which traffic brings up dialer interface

*(IF) pppoe-client dial-pool-number <number> [dial-on-demand] [service-name <name>]*  
 Specify the dialer interface to use for cloning. A dial-on-demand keyword enables DDR functionality (idle-timeout can be configured on dialer intf). Specific service can be requested from BRAS. Service parameters are defined in RADIUS server

If authentication is required, it is configured just like for ppp

**Limits**

*(IF) pppoe max-sessions <#> [threshold-sessions <#>]*  
 Specify maximum number of PPPoE sessions that will be permitted on Ethernet interface. Threshold defines when SNMP trap is sent. Max sessions depend on the platform.

*(BBA) sessions per-mac limit <per-mac-limit>*  
 Specifies the maximum number (default 100) of sessions per MAC address for each PPPoE port that uses the group

*(BBA) sessions max limit <pppoe-session-limit> [threshold-sessions <#>]*  
 Specifies maximum number of PPPoE sessions that can be terminated on this router from all interfaces. This command can be used only in a global PPPoE profile

*(BBA) sessions per-vlan limit <per-vlan-limit>*  
 Specifies maximum number (default 100) of PPPoE sessions for each VLAN

*(G) snmp-server enable traps pppoe*  
 If thresholds are used, SNMP traps for PPPoE must be enabled

`subscriber profile <name> [refresh <min>]`  
`pppoe service <name>`  
 Multiple services can be assigned to one profile. PPPoE server will advertise the service names to each PPPoE client that uses the configured PPPoE profile. Cached PPPoE configuration can be timed you after defined amount of time (minutes)

*(G) aaa new-model*  
*(G) aaa authorization network default group radius*  
 A subscriber profile can be configured locally on the router or remotely on a AAA server

**Services**

`bba-group pppoe`  
`service profile <name>`

**Verify**

`show interfaces virtual-access <number >`  
`clear interfaces virtual-access <number >`  
`show pppoe session [all]`  
`show pppoe summary`  
`clear pppoe {all | interface <if> [vlan <vlan>] | rmac}`

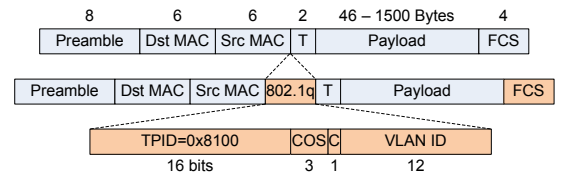
# VLAN

## Types

- Normal range 1-1001**
  - Can be configured in Server and Transparent modes
  - VLAN1 cannot be deleted, and it's name (default) cannot be changed
  - Propagated by VTP
- Extended range 1006 - 4094**
  - Supported only in Transparent and VTP v3 modes. Not propagated by VTP v1 and v2, but propagated by v3
  - Not supported in VLAN database configuration mode (*vlan database*)
  - (G) *vlan internal allocation policy [ascending | descending]*
  - Each routed port on a Catalyst 3550 switch creates an internal VLAN for its use. These internal VLANs use extended-range VLAN numbers, and such internal VLAN ID cannot be used for an extended-range VLAN. Internal VLAN IDs are in the lower part of the extended range (*show vlan internal usage*)
  - Extended VLANs cannot be pruned
- Reserved 1002 - 1005, 0, 4095**
- Native**
  - By default VLAN1 is native on all trunks (untagged frames are assigned to native VLAN)
  - (G) *vlan dot1q tag native* Not supported on ISL trunks - all frames are tagged
  - Emulates ISL behaviour on 802.1q trunks for tagging native VLAN (required for QinQ)
  - (IF) *encapsulation dot1q <vlan-id> native*
  - By default, native VLAN is terminated on physical router interface. It can be processed by a subinterface *native* keyword is used
  - (IF) *switchport trunk native vlan <id>*
  - Native VLAN, even though it is not tagged, it MUST be allowed with *switchport trunk allowed vlan* command if it is used
- Voice**
  - (IF) *switchport voice vlan <id>*
  - If port is configured as access, the switch will automatically convert it internally into a trunk
  - VLAN number is communicated to phone via CDPv2 (required for IPPhones)
  - 802.1q frame
  - 802.1p frame
  - Switch treats frames with 802.1q tag set to zero as it was an access port, but honors 802.1p COS field for prioritizing voice traffic. Traffic is then assigned back to native VLAN

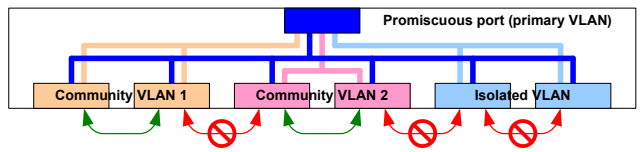
## Trunking

- DTP**
  - Switches must be in the same VTP domain. Default mode is Desirable on 3550 only. It is Auto on 3560
  - Routers do NOT understand DTP protocol. Trunk must be statically defined on switch port
  - Messages sent every 30 sec (300sec timeout) to 01-00-0c-cc-cc-cc (ISL - VLAN1, 802.1q - Native)
  - If both switches support ISL and 802.1q then ISL has priority
  - (IF) *switchport mode trunk* - always trunk, sends DTP to the other side
  - (IF) *switchport mode access* - always access, DTP is disabled
  - (IF) *switchport mode dynamic desirable* - sends negotiation DTP messages
  - (IF) *switchport mode dynamic auto* - replies to negotiation DTP messages
  - (IF) *switchport nonegotiate*
  - Disable sending of DTP messages. Can be used only if static trunking is configured
  - If DTP does not negotiate trunk, port becomes access assigned to VLAN (default 1)
  - show interface [<if>] trunk*
- ISL**
  - Cisco proprietary protocol supporting up to 1024 VLANs - deprecated
  - SA is MAC of device doing trunking; DA is 0100.0c00.0000
  - Native (non-tagged) frames received from an ISL trunk port are dropped
  - Encapsulates in 26 bytes header and recalculated 4 bytes FCS trailer (real encapsulation) - total 30 bytes added to the frame
- 802.1q**
  - IEEE standard for tagging frames on a trunk. Supports up to 4096 VLANs
  - Inserts 4 byte tag after SA and recalculates original FCS. Does not tag frames on the native VLAN
  - Canonical Format Indicator (CFI) is used only for TokenRing frames
  - TPID is in the same place as previous EtherType (T) field, indicating the frame is tagged. Real EtherType follows 802.1q tag



## Private VLANs (Cat3560)

- All hosts can be in the same subnet. VTP transparent is required (unless VTP v.3 is used)
- When you enable DHCP snooping on primary VLAN, it is propagated to the secondary VLANs
- STP runs only on primary VLAN. Community and isolated VLANs do not have STP instance
- Primary (promiscuous) VLAN**
  - All devices can access it. Isolated and community VLANs must be associated with primary VLAN
  - vlan <id> private-vlan primary private-vlan association <list>*
  - Define primary VLAN on every switch
  - interface <if> switchport mode private-vlan primary switchport primary-vlan mapping <pri> <list>*
  - Define L2 port as primary and assign secondary VLANs
- Secondary**
  - community VLAN**
    - Can talk to Primary and to each other within a community VLAN, but not to other community VLANs. There can be many community VLANs
    - (VLAN) *private-vlan community*
    - Define VLAN as community
  - isolated VLAN**
    - Can talk only to Primary. Only one isolated VLAN
    - (VLAN) *private-vlan isolated*
    - Define VLAN as isolated
  - (IF) *switchport mode private-vlan host*
  - Define L2 port as secondary VLAN
  - (IF) *switchport private-vlan host-association <pri> <sec>*
  - Assign L2 port to community or isolated VLAN



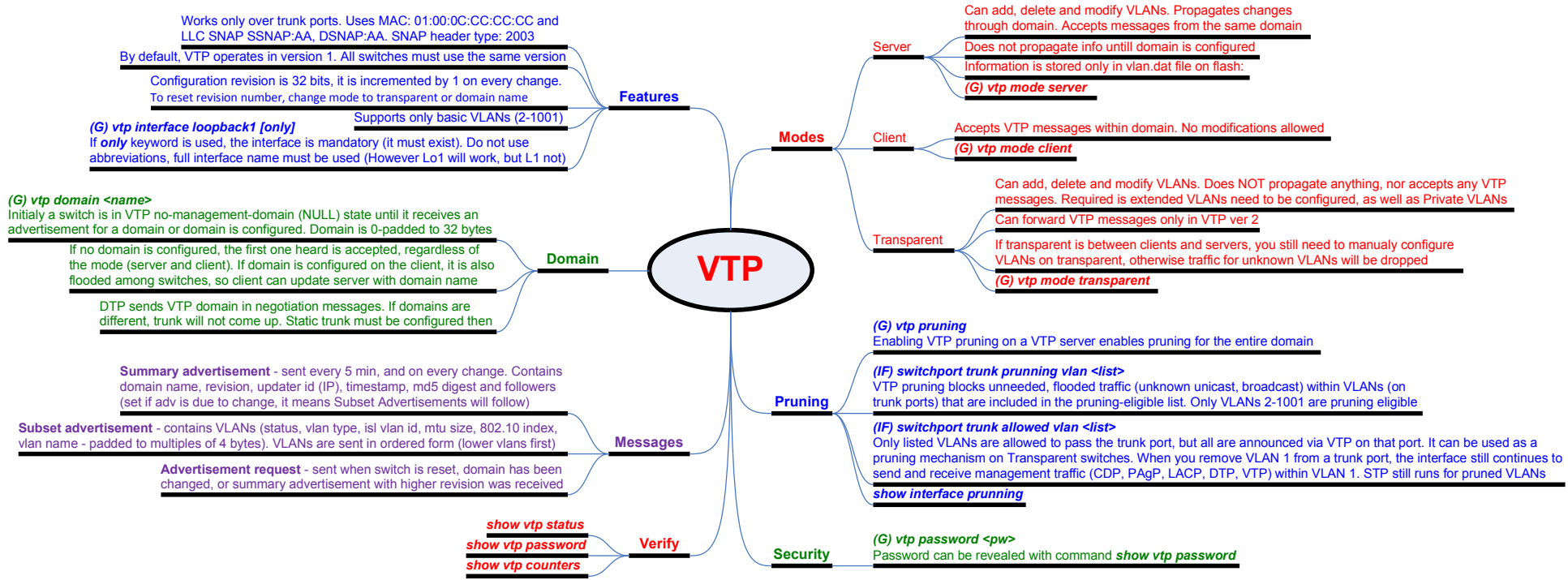
## QinQ Tunneling

- Tagged frames (EtherType 0x8100) encapsulated within additional 4 byte 802.1q header (EtherType 0x88a8), so **system mtu 1504** must be added to all switches, otherwise some protocols may not work properly (OSPF)
- The native VLAN of the IEEE 802.1Q trunks must not match any native VLAN of the nontrunking (tunneling) port on the same switch
- Use the *vlan dot1q tag native* global command to configure the edge switch, so that all packets going out IEEE 802.1q trunk, including the native VLAN, are tagged. VLAN1 is a default native VLAN, so by default this command is required.
- Supports CDP, STP, MSTP, VTP, PAgP, LACP, and UDLD
- (IF) *switchport mode dot1q-tunnel*
- (IF) *switchport access vlan <id>*
- Outer VLAN for tunneled traffic
- (IF) *l2protocol-tunnel [cdp | stp | vtp]*
- By default CDP, STP and VTP are not enabled on tunneled interfaces
- (IF) *l2protocol-tunnel point-to-point [pagp | lacp | udld]*
- Tunnel etherchannel frames. Each pair of remote ports must be in different access VLAN
- (IF) *l2protocol-tunnel cos <value>*
- COS applied to all tunneled traffic. If not defined, default is COS 5. Inner COS is preserved

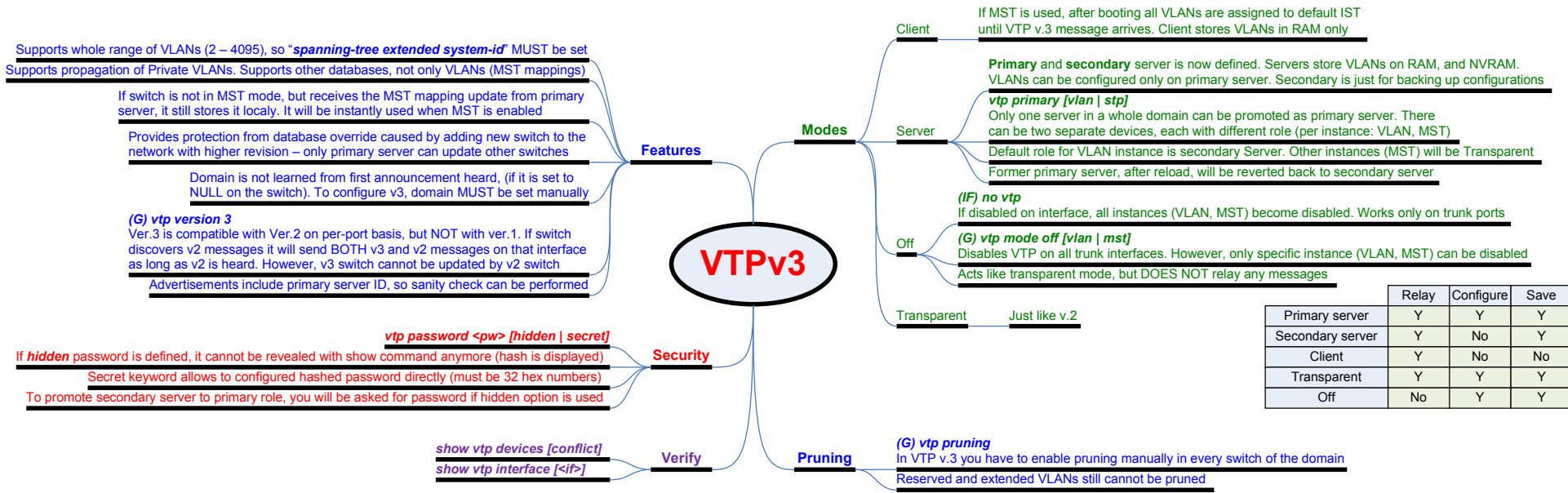
## VMPS

- (IF) *switchport access vlan dynamic*
- Switch (client) starts talking to server using VLAN Query Protocol (VQP)
- When server configured in secure mode the port is shutdown if MAC-to-VLAN mapping is not in database. In open mode, access is denied but port stays up
- 3560 can be a client and a server. 3550 can be a client only
- (G) *vmmps server <ip> [primary]*
- (G) *vmmps reconfirm <sec>* - default refresh is every 60 min
- (G) *vmmps retry <#>* - default 3 times
- show vmmps*

# VTP



# VTPv3



	Relay	Configure	Save
Primary server	Y	Y	Y
Secondary server	Y	No	Y
Client	Y	No	No
Transparent	Y	Y	Y
Off	No	Y	Y

# Cisco PVST+

Based on IEEE 802.1D standard and includes Cisco proprietary extensions such as BackboneFast, UplinkFast, and PortFast. PVST was supported only on ISL trunks

**(G) spanning-tree vlan <id> hello-time <sec>**  
 BPDU generation (default is 2 sec). Skew detection sends syslog if switch detects delay in BPDU arrival (non-root). Syslog is rate-limited 1msg/60sec, unless delay is MaxAge/2 (10 sec), then shown immediately

**spanning-tree vlan <id> forward-time <sec>** (default is 15 sec)

**spanning-tree vlan <id> max-age <sec>** (default is 20 sec)  
 Bridge waits 10 Hello misses before performing STP recalculation

Blocking (20sec) => Listening (15sec) => Learning (15 sec) => Forwarding

Bridges are not interested in local timers, they use timers send by Root Hellos.

Each BPDU sent by root, contains the Age timer. Root sets age to zero, every other switch adds 1 sec (transit delay), so BPDU shows how many hops away the root is

The max-age timer is reset on every BPDU receipt. This timer does not count down, but the counter starts from Age timer, and when it reaches max-age, BPDU is aged out. So, the further the switch, the less time is left for max-age. Ex. first switch from the root has 20 sec, second switch has 19 sec to age out BPDU, and so on

## Timers & Features

## 1. Elect the Root bridge

**Lowest Priority (Priority+VLAN+MAC) wins root election**  
 Priority – 2 bytes 32768 (0x8000)  
 ID – 6 bytes MAC

If superior (lowest) Hello is heard, own is ceased. Superior is forwarded

**(G) spanning-tree vlan <id> priority <0-61440>**

**(G) spanning-tree vlan <id> root {primary|secondary} [diameter <hop#>]**  
 - **primary:** 24576 or 4096 less than existing one (macro listens to root BPDUs)  
 - **secondary:** 28672 (always – no way to find current secondary's priority)  
 - **diameter:** causes changes to Hello, Forward delay and Maxage timers

Each switch forwards root's Hello changing some fields

Cost (total cost to the Root) – added from interface on which BPDU was received. Can be manipulated with BW, speed, and manually set on interface per VLAN  
 Forwarder's ID (Bridge ID of the switch that forwarded BPDU)  
 Forwarder's port priority – configured on interface out of which BPDU is sent  
 Forwarder's port number – outgoing interface

## 2. Determine the Root Port

- Port on which Hello was received with lowest Cost (after adding own cost)  
**(IF) spanning-tree vlan <id> cost <path-cost>** (configured on root port)
- Lowest forwarder's Bridge ID – the one who sent BPDU to us
- Lowest forwarder's port priority (default 128, in increments of 16)  
**(IF) spanning-tree vlan <id> port-priority <0-250>** (configured on DP)
- Lowest forwarder's port number

- 10Mb – 100
- 100Mb – 19
- 1Gb – 4
- 2Gb EC – 3
- 3-7Gb EC – 2
- 8Gb EC – 1
- 10Gb – 2
- 20Gb EC – 1

Switches receive BPDUs on all ports, even blocked ports. They store and relay only best BPDU (from root). If superior is heard, previous is discarded, and new one is stored and relayed.

If 10 Hellos are missed (Maxage 20 sec) the switch thinks it is a root and starts sending own Hellos again  
 Any change resulted in port to be unblocked, forces that port to go through Listening and Learning (30 sec)

If a switch receives new, different „best“ Hello on blocking port, and it still hears superior Hello on different port, it switches over the first port from blocking to DP and starts forwarding superior Hellos

Switch ignores worse BPDUs until max-age timer expires, even if his own BPDU is to be the best (in case current path to root is lost, and switch tries to declare itself as a root - only if there are no other potential ports receiving superior BPDU from current root, so the port transitions to listening and learning, otherwise, switch generates own BPDUs thinking it is a root)

## 4. Topology change

## 3. Determine Designated Ports

Only one switch can forward traffic to the same segment  
 BPDUs forwarded with lowest advertised cost (without adding own cost) define DP  
 Switch with inferior BPDU stops forwarding them to the segment  
 If advertised costs are the same the tiebreaker is exactly the same as for Root Port

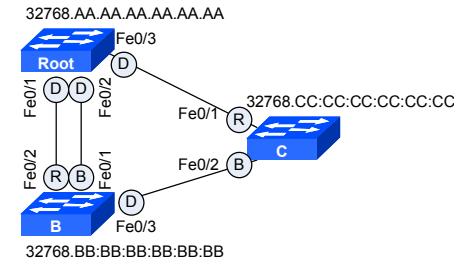
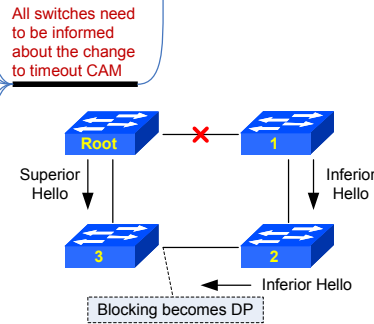
Switch sends TCN BPDU every hello time (locally defined, not from root), on root port toward Root every until ACKed by upstream switch

Upstream switch ACKs with next BPDU, setting Topology Change Ack (TCA) bit, and sends TC upward, until root is reached

When root receives TCN, it sets TCA for next BPDUs so all switches are notified

All switches use Forward Delay Timeout (15 sec) to timeout CAM (default is 300 sec) for period of MaxAge + ForwardDelay (35 sec). Root sets TC in Hellos for the period of that time

It's better than clearing MAC table, as there might be hosts successfully communicating with each other



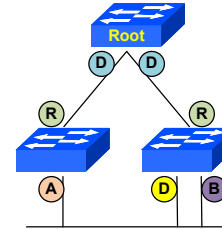
# RSTP 802.1w

## Features

- (G) spanning-tree mode rapid-pvst**
- BPDUs ver.2 is used (unused fields are now used to define port role, port state, and proposal and agreement states - 802.1d used only two bits: TC and TCACK)
- RSTP decouples the role and the state of port. No blocking and listening state (DISCARDING, LEARNING, FORWARDING)
- All switches originate Hellos all the time (keepalive). Hellos are NOT relayed
- Neighbor querying (proposal-agreement BPDU) like in backbonefast, but standardized. Convergence in less than 2 sec
- Maxage only 3 Hello misses (fast aging). Basically RSTP is not timer-based
- 802.1w is compatible with 802.1d. Port working as RTSP, when it comes up, starts a migration timer for 3 seconds. If port receives 802.1d BPDU, it transitions to 802.1d. When legacy switch is removed, RSTP switch continues working as 802.1d. Manual restart is required on that port.
- RSTP is able to actively confirm that port can safely transit to forwarding state without relying on any timers. Switch relies now on two variables: edge port and link type

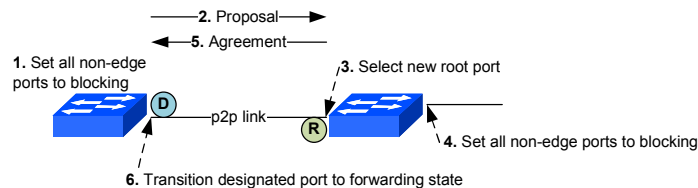
## Port roles

- New port roles used for fast convergence**
  - Backup port** – Receives better BPDU from the same switch on the segment. Provides redundant path to the same segment, and it does not guarantee a redundant path to root
  - Alternate port** – Receives better BPDU from the other switch on one segment. Provides redundant path to the root
- Port types**
  - point-to-point**
    - Full duplex port (only two switches on LAN segment) – simple and fast sync process
    - (IF) spanning-tree link-type point-to-point**
    - The p2p state can be manually forced if HDX (half-duplex) is used
  - shared edge**
    - Ports with Half Duplex (shared bus) – requires arbitration, slow and complicated sync process. Does not support RSTP and STP interoperation.
    - Defined with **spanning-tree portfast** – highly recommended



## Convergence

- Topology change**
  - Only link-up causes TC, as new path may be build. If link goes down, simple sync proces takes place. Edge ports do not generate TCN, nor sync, regardless of their state change (up or down)
  - If topology change is detected, switch sets a TC timer to twice the hello time and sets the TC bit on all BPDUs sent out to its designated and root ports until the timer expires
  - If switch receives a TC BPDU, it clears the MAC addresses on that port and sets the TC bit on all BPDUs sent out its designated and root ports (except the receiving one) until the TC timer expires (2x hello). Process contigues through whole domain
  - TCNs are never flooded to edge ports, as there are no switches there
  - Due to MAC flushing, excessive unknown unicast flooding takes place
- Sync**
  - If root port changes or better root information is received, the bridge sends a proposal only out of all downstream DP (sets proposal bit in outgoing BPDU)
  - Downstream bridge blocks all non-designated ports and authorizes upstream brodge to put his port into forwarding state. This is agreement, only if this switch does not have better root information
  - Sync stops when there is no more leaves, or Reject is received (downstream switch has better root information)
  - If designated discarding port does not receive agreement (downstream does not understand RSTP or is blocking), port slowly transitions for forwarding like 802.1d
  - Proposals are ignored on blocked ports, unless inferior BPDU is received. If local root info is better, switch immediately sends back proposal so inferior switch can quickly adapt. If local info is worse, new sync process begins.



## BPDU Frame

TCN BPDU  
Type value: 128

Protocol ID (2B)
Protocol Version ID (1B)
BPDU Type (1B)
Flags (1B)
Root ID (8B)
Root Path Cost (4B)
Bridge ID (8B)
Port ID (2B)
Message Age (2B)
Max Age (2B)
Hello Time (2B)
Forward Delay (2B)

## BPDU Flags

Topology Change (TC)	0
Proposal	1
Port Role	2
	3
	4
Learning	4
Forwarding	5
Agreement	6
Topology Change ACK	7

00: Unknown  
01: Alternate/Backup  
10: Root  
11: Designated

# MST 802.1s

## Features

- Up to 16 MST (64 RFC) instances (no platform-specific limit for number of VLANs – max 4096) – there is always one instance 0 (zero) + 15 user-defined. Instances can be numbered from 1 to 4096
- 802.1s introduces Regions (like AS in BGP) – switches in one common management. Switches belong to the same region if name, revision and vlans mappings are the same. It is not recommended to have multiple regions. Place as many switches as you can inside one MST region. Migrate core and follow to access
- VLAN-to-instance mapping is not propagated. Only digest with region name and revision number is sent
- VLANs mapped to single MSTI must have the same topology (allowed VLANs on trunks). Avoid mapping VLANs to IST(0), and never manually prune individual VLANs (belonging to the same MSTI) from trunk
- When the IST converges, the root of the IST becomes the CIST regional root
- The IST and MST instances do not use the message-age and maximum-age information in the configuration BPDUs to compute the STP topology. Instead, they use the path cost to the root and a hop-count mechanism
- Edge ports are designated by **spanning-tree portfast**
- Each switch decrements hop-count by 1. If switch receives BPDU with hop-count = 0, then it declares itself as a root of new IST instance. MST increases hop count of cascaded switches from 7 to 40 (20 is default). It also uses 802.1t long cost mode to differentiate between GE, GEC, 10G.

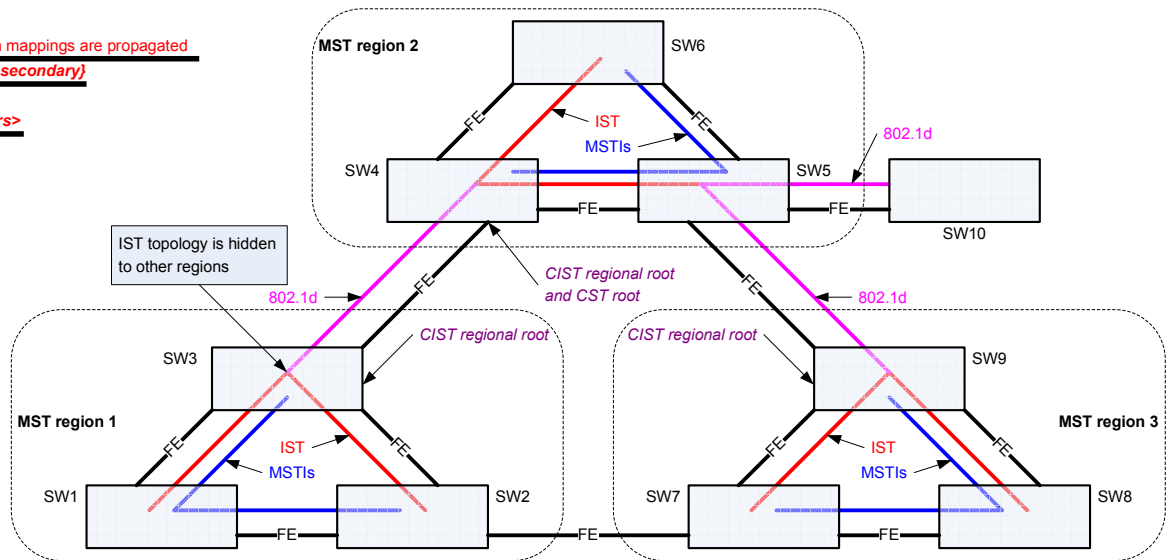
## Instances

- IST (MSTI 0)**  
Internal Spanning Tree
  - The only instance that sends and receives BPDUs. All of the other STP instance information is contained in M-records, which are encapsulated within MSTP BPDUs
  - MST Region replicates IST BPDUs within each VLAN to simulate PVST+ neighbor. First implementation of pre-standard MISTP (Cisco proprietary MST) tunneled extra BPDUs across MST
  - It is recommended to have IST root inside MST. Successful MST and PVST+ interaction is possible if MST bridge is the root for all VLANs. If MST is the root for CTS and other switch (PVST+) is the root for any of the VLANs, boundary port will become root-inconsistent
  - Represents MST region as CST virtual bridge to outside. By default, all VLANs are assigned to the IST
  - STP parameters related to BPDU transmission (hello time, etc) are configured only on the CST instance but affect all MST instances. However, each MSTI can have own topology (root bridge, port costs)
- MSTI** – Multiple Spanning Tree Instances (one or more) - RSTP instances within a region. RSTP is enabled automatically by default
- CIST** – (common and internal spanning tree) collection of the ISTs in each MST region, and the common spanning tree (CST) that interconnects the MST regions and single spanning trees
  - Each region selects own CIST regional root. It must be a boundary switch with lowest CIST external path cost
  - External BPDUs are tunneled (CIST metrics are passed unchanged) across the region and processed only by boundary switches.
  - When switch detects BPDU from different region it marks the port on which it was received as **boundary port**
  - Boundary ports exchange CIST information only. IST topology is hidden between regions
  - Switch with lowest BID among all boundary switches in all regions is elected as CST root. It is also a CIST regional root within own region

## Configuration

- (G) spanning-tree mode mst**
- spanning-tree mst configuration**
- name <name>**
- revision <number>**
- instance <id> vlan <range>**
- Must be defined on every switch. If VTPv3 is used, then mappings are propagated
- (G) spanning-tree mst <instance-id> root (primary | secondary)**
- (G) spanning-tree mst max-hops <count>**
- (G) spanning-tree mst <other STP parameters, timers>**
- (IF) spanning-tree mst pre-standard**
- If 802.1s and pre-standard MISTP ports are connected

### Final IST topology



# Port Channel

## Cisco PAgP

- Up to eight compatibly configured interfaces
- In auto-negotiation mode it may take 15 sec to form EC. It takes place before STP. Negotiation should be disabled for hosts (off)
- (IF) channel-protocol pagp**
- (IF) channel-group <1-64> mode {auto | desirable} [non-silent]**  
In silent mode etherchannel can be built even if PAgP packets are not received. The silent setting is for connections to file servers or packet analyzers
- (G) pagp learn-method {aggregation-port | physical-port}**  
How to learn the source address of incoming packets received from (aggr-port is default). If phy-port is used, then frames are sent always on the same port where MAC was learned.
- (IF) pagp port-priority <#>**  
The physical port with the highest priority (default is 128) that is operational and has membership in the same EtherChannel is the one selected for PAgP transmission

PAgP	802.1d	Behavior
on	on	No dynamic negotiation. Forced.
off	off	PortChannel negotiation disabled
auto	passive	Wait for other side to initiate
desirable	active	Initiate negotiation

## IEEE 802.3ad LACP

- LACP protocol can run only on full-duplex ports
- 16 ports can be selected, but only max 8 is used. Rest is in hot-standby
- Switch with lowest system priority makes decisions about which ports participate in bundling (switch used port-priorities)
- (IF) channel-protocol lacp**
- (IF) channel-group <1-64> mode {passive | active}**
- (IF) lacp port-priority <#>**  
Priority decides which ports are used for EX, and which remain in standby. Default 32768, lower is better. If priority is the same, Port ID is used (lower better)
- (G) lacp system-priority <#>** (lower better)  
The system priority is used in conjunction with the MAC to form the system identifier
- show lacp sys-id**

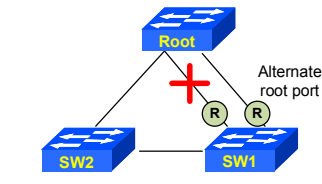
## Load balancing

- (G) port-channel load-balance {dst-ip | dst-mac | src-dst-ip | src-dst-mac | src-ip | src-mac}**  
Set the load-distribution method among the ports. Src-mac is default (XOR on rightmost bits of MAC)
- show etherchannel load-balance**

## Features

- All physical interfaces must have identical configuration. If any of speed, duplex, trunking mode, allowed vlans is different, the port is not bound to etherchannel
- LACP or PAgP check links consistency. If they are disabled, STP loop can occur (Etherchannel on one side, single links on other side)
- (Po1) port-channel min-links <#>**  
By default, etherchannel is active as long as at least one link is active. STP cost is not adjusted when links go down. You can make sure that data flow chooses hi-bandwidth redundant path in case only few links are left.
- show etherchannel {summary | detail | port-channel | protocol}**
- show interface etherchannel**

# Convergence



## Uplinkfast

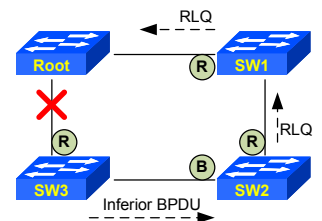
- 802.1d legacy feature used on access switch with multiple uplinks to core
- Priority is automatically set to 49152 so the switch will not become root. Port cost is set to 3000 so it will not transit any traffic
- During switchover to new RP, for each connected MAC it multicasts dummy frames with each MAC as SA forcing other switches to update CAM. Other MACs are cleared
- Tracks alternate root port (second best path) to immediately switch over
- (G) spanning-tree uplinkfast [max-update-rate <rate>]**  
If rate is 0 then no multicast flooding takes place (150 default)

## Portfast

- Immediately switches over to forwarding state. Avoid TCN generation for end hosts
- BPDU guard should be enabled on that port. Portfast does not turn off STP on that port
- (G) spanning-tree portfast default**
- (IF) spanning-tree portfast**
- (IF) switchport mode host**

## Backbonefast

- 802.1d legacy feature used for indirect link failure detection – explicit verification of inferior BPDUs. Recovery within 30 sec.
- (G) spanning-tree backbonefast**  
All switches within a domain must be configured
- If inferior BPDU is received on block port, switch SW2 sends proprietary Root Link Query messages on root and alternate (blocked upstream) ports containing SW2's root information and SW2 BID. If upstream switch has the same root information as SW2 it forwards it to root ports. Root switch confirms it's still a root with positive answer flooding on all DP
- If any switch has different information, immediate negative answer is sent, and SW2 performs root election without waiting MaxAge (only Listening and Learning). In case of positive answer blocked port changes to Listening and Learning



# STP Port Protection

**BPDUGuard**  
 Err-disable portfast port upon receiving BPDU  
 (G) `spanning-tree portfast bpduguard default`  
 Applied only to interfaces which are in portfast state  
 (IF) `spanning-tree bpduguard enable`  
`show interfaces status err-disabled`

**BPDUFILTER**  
 (G) `spanning-tree portfast bpdufilter default`  
 Applies only to interfaces in portfast state. Portfast state switches to non-portfast upon receiving BPDU. Port is not blocked.  
 (IF) `spanning-tree bpdufilter enable`  
 Port does not send any BPDUs and drops all BPDUs received

## Etherchannel guard

(G) `spanning-tree etherchannel guard misconfig`  
 A misconfiguration can occur if local interfaces are configured in an EtherChannel, but the interfaces on the other device are neither LACP, PAgP, nor ON. It is used to check consistency. If etherchannel is not detected all bundling ports go into err-disable. This command uses spanning tree (BPDU comes back on a port, meaning one of etherchannel ports on remote end is not in common channel)

## Root guard

Can be enabled on **designated ports only**. Opposite to loop guard  
 When superior BPDU is received on a DP, the port becomes root-inconsistent. Recovery after ForwardDelay sec of not receiving superior BPDU  
 Cannot be configured on backup ports when uplinkfast is configured  
 Applies to all the VLANs to which the interface belongs  
 (IF) `spanning-tree guard root`  
`show spanning-tree inconsistentports`

## Loop guard

If no BPDUs are received on a blocked port for a specific length of time (MaxAge 20 sec), Loop Guard puts that port (per VLAN) into loop-inconsistent blocking state, rather than transitioning to forwarding state  
 Unlike UDLD, loopguard protects against STP software problems (bugs, etc)  
 Can be enabled on **non-designated ports only**, which are mainly root and alternate ports. Cannot be enabled on portfast and dynamic VLAN ports. Enabling on shared links is highly not recommended.  
 Automatic recovery when BPDU is again received  
 (G) `spanning-tree loopguard default`  
 (IF) `spanning-tree guard loop`

## UDLD

Sends local port ID and remote (seen) port ID. Remote end compares with own state  
 Unlike loopguard, UDLD protects against wrong wiring, and is per-physical-port, not per-VLAN  
 (G) `udld message time <sec>`  
 Default L2 probes sent every 15 sec to mac 01:00:0C:CC:CC:CC. Must be ACKed by remote end. Dead is 3x hello.  
 Normal mode does nothing except syslog (on some platforms it may err-disable port on the side where misconfiguration detected)  
 Aggressive mode attempts to reconnect once a second 8 times before err-disabling both ends  
 If configured for the first time it is not enabled until first Hello is heard from the other side  
 (G) `udld {enable | aggressive}`  
 Enable UDLD in normal (`enable`) or aggressive mode only on all fiber-optic interfaces  
 (IF) `udld port {aggressive}`  
 Enable UDLD in normal or aggressive mode on fiber-optic (override global mode) and twisted-pair link  
`udld reset` – reset err-disable state without shutting down port  
`show udld [ <if> | neighbors]`

**SPAN**

- You cannot monitor outgoing traffic on multiple ports. Only 2 SPAN sessions per switch
- You can monitor incoming traffic on a series or range of ports and VLANs.
- Receive (Rx) SPAN – catch frames before any modification or processing is performed by the switch. Destination port still receives a copy of the packet even if the actual incoming packet is dropped by ACL or QoS drop.
- Transmit (Tx) SPAN – catch frames after all modification and processing is performed by the switch. In the case of output ACLs, if the SPAN source drops the packet, the SPAN destination would also drop the packet
- (G) monitor session <#> filter vlan <vlan-ids>**  
Limit the SPAN source traffic to specified VLANs
- (G) monitor session 1 source vlan <id> rx**  
VLAN can be only a source of traffic
- (G) monitor session 1 source interface <if> [rx | tx | both]**
- (G) monitor session 1 destination interface <if>**

**RSPAN**

- You cannot use RSPAN to monitor Layer 2 protocols (CDP, VTP, STP)
- You must create the RSPAN VLAN on all switches that will participate in RSPAN. It cannot be any of reserved VLANs (including 1)
- The reflector port (Cat 3550 only) loops back untagged traffic to the switch. It becomes unavailable. The port can be down (it's ASIC is used)
- Traffic is placed on the RSPAN VLAN and flooded to any trunk ports that carry the RSPAN VLAN
- No access ports are allowed to be configured in the RSPAN VLAN
- vlan <id>**  
**remote-span** (on source switch only)
- SW1: monitor session 1 source interface <if> [rx | tx | both]**
- SW1: monitor session 1 source vlan <id> rx**
- SW1: monitor session 1 destination remote vlan <id> reflector-port <if>**
- SW2: monitor session 1 source remote vlan <id>**
- SW2: monitor session 1 destination interface <if>**

**Macro**

**Interface Range**

- (G) define interface-range <name> <intf range>**
- (G) interface range macro <name>**

**Smartport**

- macro name USER\_PORT**  
**switchport mode access**  
**switchport access vlan \$vlanID**  
**spanning-tree portfast**
- (IF) macro apply USER\_PORT \$vlanID 10**
- After applying macro to interface, **macro description <name>** will be added to indicate that configurations were applied from macro
- show parser macro brief**  
Pre-defined macros

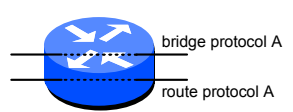
**Bridging**

**Transparent**

- Complies with the IEEE 802.1D standard
- (G) bridge <bridge-group> protocol ieee**
- (IF) bridge-group <bridge-group>**
- (G) bridge <bridge-group> acquire**  
Forward frames according to dynamically learned MAC addresses. If disabled, static mappings must be used.
- (G) bridge <bridge-group> address <mac-address> {forward | discard} [<intf>]**  
Filter frames with a specific source or destination MAC address

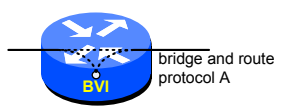
**CRB**

- Route given protocol among one group of interfaces and concurrently bridge that protocol among a separate group of interfaces
- Protocol may be either routed or bridged on a given interface, but not both
- (G) bridge crb**
- (G) bridge <bridge-group> route <protocol>**  
When CRB is enabled, you must configure explicit bridge route command for any protocol that is to be routed on the interfaces in a bridge group



**IRB**

- Routers do not support per-vlan STP, so Bridge Priority is always 32768 for every VLAN, which is lower than any value on switches, which add VLAN id, so router will be a root for all VLANs by default
- Integrated routing and bridging makes it possible to route a specific protocol between routed interfaces and bridge groups, or route a specific protocol between bridge groups
- The bridge-group virtual interface (BVI) is a normal routed interface that does not support bridging, but does represent its corresponding bridge group to the routed interface
- Packets coming from a routed interface, but destined for a host in a bridged domain, are routed to BVI and forwarded to the corresponding bridged interface
- All routable traffic received on a bridged interface is routed to other routed interfaces as if it is coming directly from BVI.
- (G) bridge irb**
- (G) interface bvi <bridge-group>**
- (G) bridge <bridge-group> route <protocol>**
- (G) bridge <bridge-group> bridge <protocol>**



# 35x0 Features

Flex Links are a pair of a Layer 2 interfaces where one interface is configured to act as a backup to the other. Users can disable STP and still retain basic link redundancy. It's a sort of UplinkFast without STP

- Preemption can be enabled so traffic goes back to primary link after it comes back up
- A backup link does not have to be the same type
- STP is automatically disabled on Flex Link ports

The MAC address-table move update feature allows the switch to provide rapid bidirectional convergence when a primary link goes down and the standby link begins forwarding traffic

- (IF) switchport backup interface <intf>
- (IF) switchport backup interface <intf> preempt mode [forced | bandwidth | off]  
forced – active always preempts; bandwidth - intf with higher BW always acts as active
- (IF) switchport backup interface <intf> preempt delay <sec> (default 35 sec)
- (IF) switchport backup interface <intf> mmu primary vlan <vlan-id>  
If not defined, the lowest VLAN is used for MAC-address move updates
- (G) mac address-table move update transmit  
Enable the access switch to send MAC address-table move updates to other switches
- (G) mac address-table move update receive  
Enable the switch to get and process the MAC address-table move updates

## FlexLink

## SVI

- Switched Virtual Interface is an L3 interface acting as a potential GW for a VLAN
- VLAN must exist in database, otherwise interface vlan <vlan ID> will be down
- If switch is a real L3 then physical interfaces can be assigned IP address (no switchport). Adding many SVIs does not make a switch an L3 switch
- Routing between SVIs is not recommended, as it takes much longer to detect a link failure (SVI uses autostate process, which delays routing convergence)
- (G) sdm prefer {default | access | vlan | routing | dual-ipv4-and-ipv6}  
If you use switch for routing make sure you adjust SDM template (Switched Database Manager). TCAM structure is then properly managed for L2/L3 entries

## Autonegotiation

- Works only if enabled on both sides
- If one side is manually configured, speed will be negotiated, but duplex not. By default, auto-port gets stuck in 100/half duplex
- If mismatch occurs, full-duplex side will face CRC errors (does not expect collisions, so it treats them as malformed frames)
- If mismatch occurs, half-duplex side will face late-collisions, as the other side is able to transmit at any time, without listening to the medium

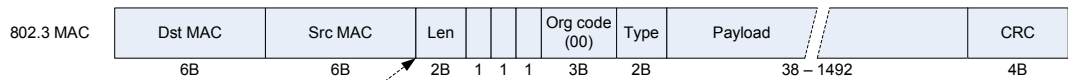
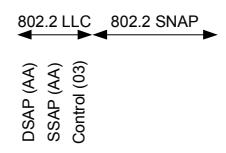
## Fallback bridging

- With fallback bridging, the switch bridges together two or more VLANs or routed ports, connecting multiple VLANs within one bridge domain
- Fallback bridging does not allow spanning trees from VLANs to collapse. Each VLAN has own SPT instance. There is also separate SPT, called VLAN-bridge SPT, which runs on top of the bridge group to prevent loops
- (G) bridge <bridge-group> protocol vlan-bridge
- (IF) bridge-group <bridge-group>
- By default, switch forwards any frames it has dynamically learned. But, the switch only forward frames whose MAC addresses are statically configured (static MAC for bridge, not for mac-address-table !!!).
- 1) no bridge <group> acquire
- 2) bridge <group> address <mac> {forward | discard} [<interface>]

## MAC notification

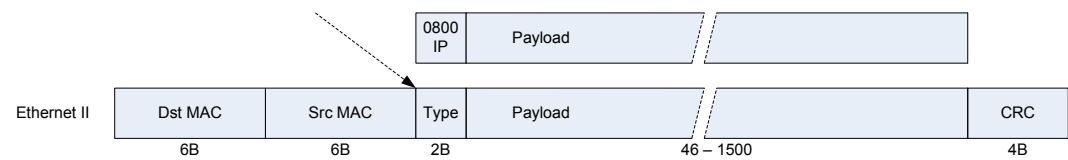
- (G) snmp-server enable traps mac-notification
- (G) mac address-table notification change
- (G) mac address-table notification change [history-size <#>] [interval <sec>]  
By default traps are sent every 1 sec. History size is 1.
- (IF) snmp trap mac-notification {added | removed}

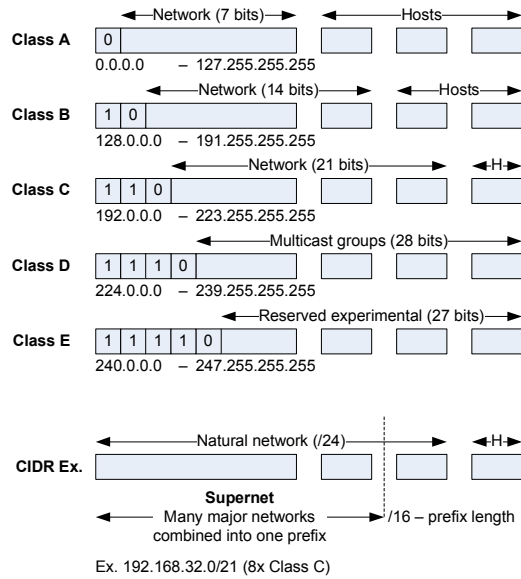
## Ethernet frames



Type/Len ranges are different to make Ethernet II and 802.3 frames distinguishable. Frames with value 0-0x05DC (1500) are 802.3, and values above that make Ethernet II

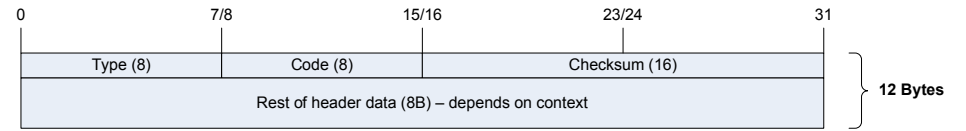
MTU for Ethernet II is 1500, but for 802.3 it is 1492 (3B LLC, 5B SNAP)





**Common networks**

0.0.0.0/8	Default network
10.0.0.0/8	Private network
127.0.0.0/8	Loopback
169.254.0.0/16	Link-Local
172.16.0.0/12	Private network
192.0.0.0/24	Reserved (IANA)
192.0.2.0/24	Test network
192.88.99.0/24	IPv6 to IPv4 relay
192.168.0.0/16	Private network
198.18.0.0/15	Network benchmark tests
198.51.100.0/24	Test network
203.0.113.0/24	Test network
224.0.0.0/4	Multicasts
240.0.0.0/4	Reserved
255.255.255.255	Broadcast



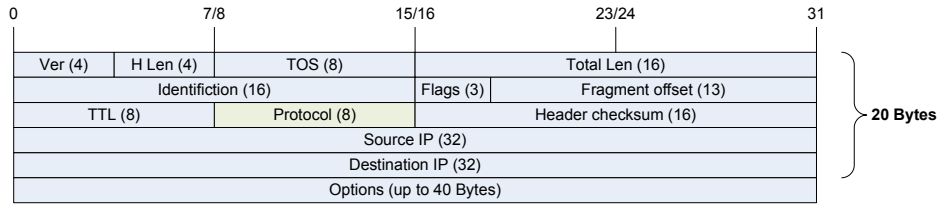
# ICMP

- Features**
  - Error message is never send if another error message is received to avoid loops. Also, it is never sent in reply to broadcast or multicast packets or other IP segments than first, as well as packets with source IP not defining single host (loopback, broadcast, all zeros, etc)
  - Error messages must include in their payload original IP header with all options and first 8 bytes of data following IP header in original packet. IP header allows to interpret those 8 bytes of data. For TCP and UDP ports are included in those 8 bytes, so for example unreachable can be generated stating which ports are unreachable
  - For unreachable message first 32 bits in ICMP payload are unused (all 0), they can be used to define MTU for PMTUD mechanism.
  - ICMP echo contains identifier which allows to distinguish between several processes sending ping message from single host. Also sequence number is included, starting from 0 incrementing by 1 with every message sent.
  - Record Route IP option stores max 9 hops. 20 bytes fixed IP header, 40 bytes left, 3 used for IP option overhead – own header, then 37 bytes available. Each IP address is 4 bytes, so 9 hops = 36 B is used.
  - Redirect contains in reserved 4 octets IP address of router to be used for sending packets to a destination network. Redirects can be generated only by routers, not hosts. Also, routers do not use redirect messages, they use routing table
  - "!" - OK, "." - (dot) timeout, "M" - usually fragmentation needed but DF set, "U" - unreachable

- Traceroute**
  - It sends UDP messages with dest port most likely not being used (above 30000). Intermediate hosts send Time Exceeded, but when datagram reached end host, even if TTL is 1, it does not generate Time Exceeded (as it is a final host), so Port Unreachable is generated
  - Hosts which receive datagram with TTL 0 or 1 must NOT forward it. If TTL=0 they drop it and sent Time Exceeded ICMP message

**Protocol #**

1	ICMP
2	IGMP
4	IP
6	TCP
17	UDP
41	IPv6
46	RSVP
47	GRE
50	ESP
51	AH
88	EIGRP
89	OSPF
102	HSRIPv2
103	PIM
112	VRRP

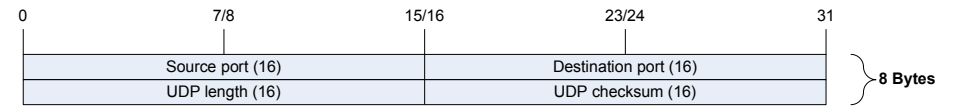


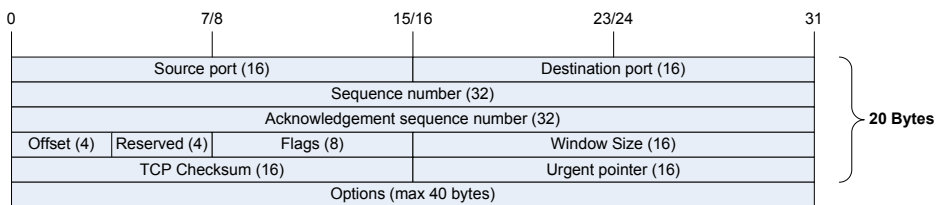
# IPv4

- Header**
  - Header Len: number of 32b/4B words – default is 5, that is 5x4 bytes = 20 bytes. Max IP header is 60 bytes (15x4B words). Padding is used to make sure header always end on 32 bits boundary
  - IP options: could be: record route, timestamp, loose and strict source routing
  - Total length: entire datagram size, including header and data, in bytes. Max 65536 B
  - Identification: used for uniquely identifying fragments of an original IP datagram when fragmentation is used
  - Flags: bit 0: Reserved, bit 1: Don't Fragment (DF), bit 2: More Fragments (MF)
  - Fragment offset: defined in 8B blocks. Specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram. The first fragment has an offset of zero. This allows a maximum offset of  $(2^{13} - 1) \times 8 = 65,528$  bytes
  - TTL: Each router decrements TTL by one. When it hits zero, the packet discarded
  - Header checksum: At each hop, the checksum of the header must be compared to the value of this field

# UDP

- Features**
  - Connectionless. No way to track lost datagrams. Upper layer must take care
  - Well fit for multimedia traffic due to small header size
  - Host is not required to receive datagram larger than 576 bytes. TCP divides data into segments, so it is not a concern, but UDP protocols often limit their payload to 512 bytes





Offset: TCP header length. The same rules apply as for IP header

Initial SNs for new sessions start with 1 and increments every 0.5 sec and at every new connection by 64000, cycling to 0 after about 9.5h. The reason for this is that each connection starts with different initial number

CWR – Congestion Window Reduced flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set and had responded in congestion control mechanism

ECE – Explicit Congestion Notification (ECN-Echo) – not the same as ECN in IP header TOS field

URG – indicates that the Urgent pointer field is significant

ACK – Acknowledges data received. All packets after the initial SYN should have this flag set

PSH – Asks to immediately push the buffered data to the receiving application. Normally, TCP waits for the buffer to exceed the MSS – can be problematic (delay) for applications sending small data

RST – Reset the connection

SYN – Exchange sequence numbers. Only the first packet sent from each end should have this flag set

FIN – No more data from sender, connection can be closed

(G) ip tcp window-size <bytes>

Window size: defines the number of bytes receiver is willing to accept before it sends ACK. Initially set to number of bytes set as ACK SN sent in 3-way handshake. Default is 4128 B

Options can be MSS, Timestamp, Selective ACK. It is exchanged only in first segments (SYN)

(G) ip tcp selective-ack

TCP might not experience optimal performance if multiple packets are lost from one window of data. Receiver returns selective ACK packets to sender, informing about data that has been received. The sender can then resend only the missing data segments

(G) ip tcp timestamp

TCP time stamp improves round-trip time estimates

TCP uses also congestion window (CWND). It is not communicated between peers. TCP sender calculates CWND by its own - varies in size much more quickly than advertised window as it reacts to congestion

TCP sender always uses the lower of the two windows to determine how much data it can send before receiving ACK

3-way handshake sets CWND = 1 and Slow Start Threshold (SSTHRESH) = 65535

1) TCP sender fails to receive ACK in time (possible lost packet)

2) TCP sender sets CWND to the size of a single segment

3) Slow start threshold SSTHRESH is set to 50% of CWND value before lost segment

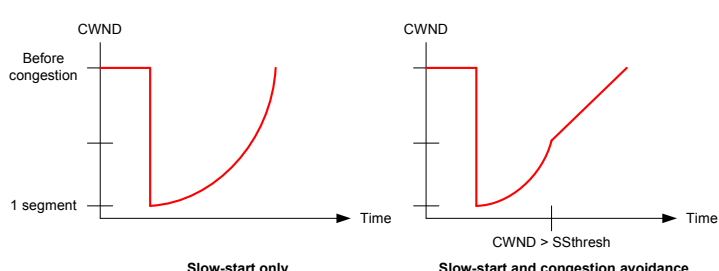
4) Slow start governs how fast CWND grows until it reaches value of SSTHRESH

5) After CWND > SSTHRESH congestion avoidance governs how fast CWND grows

CWND grows at an exponential rate during slow start

Congestion avoidance allows CWND to grow slower at a linear rate

Congestion Avoidance and Slow Start are algorithms with different objectives. In practice they are implemented together



### Connection

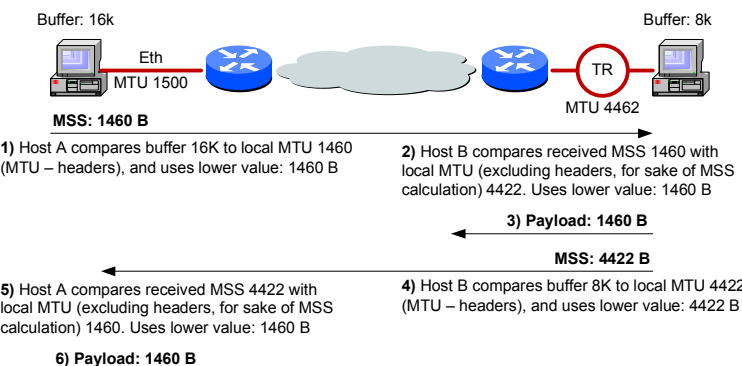
- 3-way handshake is required before data can be sent. Each side sets own SN independently, and exchanges it with the other side
- Closing connection is a 4-way. Any endpoint can send FIN to signal EoT, it must be ACKed. Since TCP is a full-duplex, other side must also send FIN and wait for ACK
- (G) service tcp-keepalive {in | out} Detect dead sessions (probe idle connections)
- (G) ip tcp synwait-time <sec> Timeout for establishing all TCP sessions from a router. Default is 30 sec. Can be used to speed up telnet timeout for non-responding hosts
- show tcp brief all [numeric]
- show tcp tcb <#>
- Show detail TCP session information. Acquire TCP from show tcp brief all

**Common port numbers**

echo	7/tcp/udp
discard	9/tcp/udp
daytime	13/tcp/udp
chargen	19/tcp/udp
bootps	67/udp
bootpc	68/udp
auth	113/tcp/udp
ntp	123/udp
netbios-ns	137/tcp/udp
netbios-dgm	138/tcp/udp
netbios-ssn	139/tcp/udp
snmp	161/udp
snmptrap	162/udp
bgp	179/tcp
syslog	514/udp
shell	514/tcp
rip	520/udp
ripng	521/udp

### MSS

- (G) ip tcp mss <#> Define MSS for TCP connections from and to a router. Default is 1460 for local destination, or 536 for remote
- TCP is a stream protocol, unlike UDP, where each write, performed by application, generates separate UDP segment. TCP collects writes and may send them all in one segment as chunks
- MSS is a largest amount of data (without headers) that TCP is willing to send in a single segment. MSS = MTU – IP header – TCP header. Should be small enough to avoid fragmentation
- Derived from local interface MTU minus TCP and IP headers. (Ex. 1460 for ethernet). Sender compares own MSS and local MTU, chooses lower one and sends this MSS to receiver
- When destination IP is non-local or other side does not set MSS, then MSS is set to 536 (20B IP and 20B TCP is added, so IP packet fits into min 576B required by RFC for host to accept)
- Received MSS is always compared only to local MTU – smaller value is used. If there is smaller MTU somewhere on the path, fragmentation will occur. PMTUD should be used to find lowest MTU on the path (tunneling on intermediate routers lowers MTU)



### Flow Control

- Receiver specifies the receive window with the amount of data it is willing to buffer. Sending host can send only up to that amount of data before it waits for an acknowledgment
- Windows is set in every segment, and is floating, depending on how fast process reads data from incoming buffer. ACK can set window to 0, which means receiver's process hasn't read data from buffer yet. A while later ACK is sent with updating window. It looks like another ACK but it's just Window Update
- Sender does not have to fill whole receiver's window. Receiver does not have to wait until whole window is filled
- Persist Timer is started after each window=0. Window Probe is sent after timer expires (no Window Update was received) at 60 sec intervals until session is terminated or new windows is advertised. To avoid sending small segments while buffer is being freed (silly window syndrome), receiver does not advertise new window until half of available buffer is free
- TCP usually does not send ACK at the same time data has been received. It waits (200ms) so maybe some data can be sent back (piggyback ACK). If there is data to be sent ACK is sent immediately. The 200ms timer goes off at fixed intervals, so ACK can wait from 1 to 200 msec, depending on when data was received
- (G) service nagle Nagle – collect data and send them in one segment to avoid tinygrams. It is not recommended for interactive applications (mouse movements).

# MTU & Fragn.

## Fragmentation

Maximum datagram length is 65k, but most links enforce lower MTU. IP packets can be fragmented to alleviate MTU differences.

When IP datagram is fragmented, it is not reassembled until it reaches final host (or router in case of tunnel endpoint if tunneled traffic is fragmented)

Dropped fragments cause whole IP packet to be retransmitted

16 bit identifier identifies whole datagram. It is the same in all fragments.

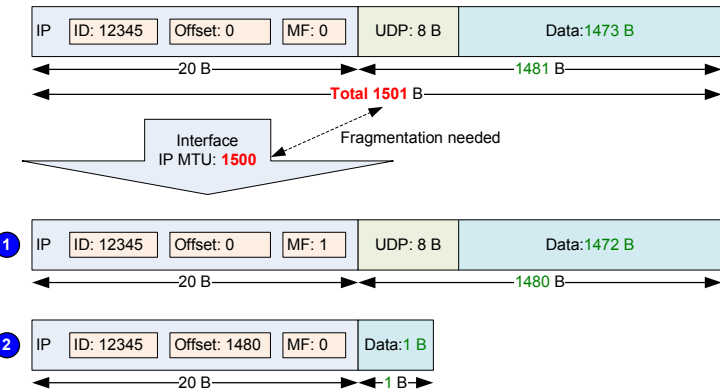
DF - used by PMTUD, 0:may fragment, 1:don't fragment

MF - 0:last fragment, 1:more fragments

13 bits fragment offset (in Bytes). First fragment starts with 0

IP header (20 bytes) is added to each fragment. Original IP datagram size can be determined only after last fragment is received

Fragmentation is problematic for receiver. Hosts don't have problems, as they have resources for this. Router reserves maximum available buffer for fragmented packet, as it has no idea how large the packet will be. This consumes scarce resources



## PMTUD

**(G) ip tcp path-mtu-discovery [age-timer {<min> | infinite}]**  
 Enable PMTUD. Default time is 10 min. It changes the default MSS to 1460 even for nonlocal nodes.

PMTUD is supported only for TCP traffic and is independent in both directions

If host supports PMTUD (in most cases it does), all packets have DF bit set

If host does not announce MSS, it is assumed 536 (for non-local destinations). It can be also saved on per-route basis

After determining MSS, host sends segments with DF set. If MTU is smaller on the path, ICMP is returned with next-hop MTU. If MTU is not included in ICMP message, IP stack must perform trial-and-error procedure to guess minimal MTU (may take few packets until MTU is guessed)

Upon receiving ICMP error, CWND is not changed, but slow-start is initiated. As path can change, hosts try larger MTU (up to announced MSS) periodically – every 10 min

**(G) ip icmp rate-limit unreachable [df] [<ms>] [log [<packets>] [<interval-ms>]]**  
 ICMP: "fragmentation needed but DF set" (3/4) messages are throttled one per 500 ms. It can be set independently for DF messages and all other ICMP messages

PMTUD may not work if firewalls are on the path, which usually filter unreachable

Allow (ACL) unreachable **permit icmp any any unreachable permit icmp any any time-exceeded**

Signal MSS **(IF) ip tcp adjust-mss <value>**  
 Better solution than clearing DF to allow fragmentation, is to signal MSS between endpoints. This is only for TCP traffic

Clear DF bit **Allow fragmentations by clearing DF bit with route map (should be used as last resort)**  
**route-map Clear-DF permit 10 match ... set ip df 0 interface <inbound if> ip policy route-map Clear-DF**

## Tunnels

**(IF) tunnel path-mtu-discovery**  
 External GRE IP header has DF always cleared, not copied from original IP. This command causes DF to be copied from original packet to GRE IP header.

**1) GRE tunnel IP MTU is 1476 (1500 – 24 bytes for GRE header), DF not set**  
 Packet 1500 is received. TCP segment is 1480, which is larger than GRE MTU 1476. Fragmentation takes place. 1st packet is 1456 (+20 IP), 2nd packet is 24 (+20 IP). Each packet is then encapsulated in GRE: 1st packet is 1500 (including 24 GRE), 2nd packet is 68 (including 24 GRE). Tunnel destination host removes GRE and forwards 2 independent IP packets to end station, which reassemble them.

**2) GRE tunnel IP MTU is 1476 (1500 – 24 bytes for GRE header), DF set**  
 Router receives 1500 with DF. Packet is dropped, and ICMP is sent back with MTU 1476 (from GRE tunnel endpoint). Packet is encapsulated with new MTU and sent

**3) GRE tunnel IP MTU is 1476 (1500 – 24 bytes for GRE header), DF set or not, some smaller MTU between GRE endpoints, no tunnel PMTUD**  
 Packet with 1476 is received. GRE is added, packet is sent as 1500. Intermediate link is 1400. Packet is fragmented (GRE header DF is 0), original IP is only in first fragment. Tunnel endpoint must reassembly those parts. Then GRE is removed and original packet is sent to end station

**4) GRE tunnel IP MTU is 1476 (1500 – 24 bytes for GRE header), DF set, some smaller MTU between GRE endpoints, tunnel PMTUD enabled**  
 Packet with 1476 is received. GRE is added and sent. Intermediate link drops packet (DF set) and sends ICMP (MTU 1400) to tunnel source (external IP header source). Router lowers tunnel MTU to 1376 (1400 – 24 GRE). As packet was dropped, host retransmits it with 1476, but this time router send ICMP to original host with new MTU 1376. Host uses new MTU

**5) Pure IPSec tunnel mode, DF cleared**  
 Packet 1500 is received. IPSec adds 52 bytes. Outgoing MTU is 1500 so packet is fragmented in a normal way

**6) Pure IPSec tunnel mode, DF is set**  
 IPSec always performs PMTUD. Encryption is always performed before fragmentation. Packet 1500 is received, 52 bytes are added by IPSec. Outgoing MTU is 1500 so packet is dropped and ICMP is sent back with MTU 1442 (1500 – 58, which is max IPSec header size). Now host sends 1442, IPSec adds 52, resulting in 1496. Now packet is sent, but intermediate links is 1400. ICMP is sent to IPSec router with MTU 1400, router lowers SA MTU to 1400. Now, when host re-sends packet with 1442, router drops and sends ICMP with MTU 1342 (1500 – 58 max IPSec header). Host now sends 1342, 52 is added, and packet is sent all the way.

**7) GRE + IPSec**  
 IPSec is usually in transport mode to carry GRE between endpoints, and GRE itself is encrypted. In transport mode we save 20 bytes. It is recommended to set **ip mtu 1400** on GRE tunnels to avoid double fragmentation

Administrative Distance	
Directly connected	0
Static to interface/NH	1
EIGRP Summary	5
eBGP	20
EIGRP Internal	90
IGRP	100
OSPF	110
ISIS	115
RIP	120
EGP	140
ODR	160
EIGRP external	170
iBGP	200
BGP local	200
Unknown (not valid)	255

# Routing features part 1

**(G) ip route <net> <mask> <gw> <AD>**  
 Floating static route is used to provide backup route in case primary route disappears (primary must have lower AD than floating static)

Static route to p2p WAN interfaces can be always used, as there is always only one receiver on the other end. Static route to LAN interface can be used only if there is a router in that LAN segment, with **ip proxy arp** enabled

Static route to interface makes this network also „connected“, so they can be advertised with **network** statements by some protocols. Only BGP and EIGRP are able to pick up such networks. Static to Null0 acts the same, as Null0 is an interface

When networks are summarized in OSPF and EIGRP, by default static summary route is created. They can be disabled

**(OSPF no discard route [internal | external])**  
**(IF) ip summary-address eigrp <#> <net> <mask> 255**

## Static routing

**distance <distance> <ip> <mask> <acl>**  
 Defined within a routing protocol (any), but is not protocol-specific. The ip/mask defines advertising router (source), and an acl defines which routes will get new distance

If AD is manipulated, and two protocols have the same AD, the tie-breaker is the default, original AD for each protocol

## Distance

**(G) ip default-gateway <ip>**  
 Used not only on switches, but also on routers with ip routing disabled. When router is booting via TFTP, ip routing is not enabled yet, so this command may be needed.

**(G) ip default-network <net>**  
 Network must be in classful form and it must be in routing table. Makes that major network a candidate default. If you specify a subnet network (which must be in routing table also), IOS will automatically install major network as a static route with subnet network as a NH. The command with major network must be issued again to mark it as candidate default

To propagate default-network with EIGRP, this network must be coming from EIGRP. If it is defined as static, it must be either redistributed or advertised with network command

RIP will automatically advertise 0.0.0.0 if gateway of last resort is set with default-network

OSPF does not understand default-network at all

**(G) ip route 0.0.0.0 0.0.0.0 <gw>**  
 EIGRP and RIP can only propagate existing 0/0 via redistributing (for example, from static). OSPF does not understand 0/0 via redistribution unless **default-information originate** is added

## Default route

## Redistribution

**Step 1:** get all routes which are in routing table and belong to redistributed protocol (**show ip route <protocol>**)

**Step 2:** get all connected routes which are covered by redistributed protocol with network command (**show ip route connected <addr> => redistributed by <protocol>**)

Chain distribution on one router is **NOT** possible. For example when redistributing EIGRP => RIP => OSPF, then EIGRP routes will be redistributed into RIP, but NOT into OSPF. Separate redistribution of EIGRP to OSPF needs to be configured

Routes redistributed from one protocol (higher AD) into another protocol (lower AD) will NOT be in the routing table on redistributing router as originated by the second protocol, although AD is lower. Route to be redistributed must be in the routing table, so it could cause endless redistribution loop

## Distribute-list

**router <IGP-protocol>**  
**distribute-list <acl> {in | out} <intf>**  
 When using extended ACL in distribute-list in IGP, the „source“ part is an update source of the route, and „destination“ is network to be matched (distributed)

**access-list <acl> permit ip <source> <source mask> <network> <network mask>**

**router <IGP-protocol>**  
**distribute-list prefix <prefix1 name> gateway <prefix2 name> {in | out}**  
 Filter prefixes in prefix1 list received from gateways listed in prefix2 list

**Class A: ip prefix-list A permit 0.0.0.0/1 ge 8 le 32 => access-list 100 permit 0.0.0.0 127.255.255.255**  
**Class B: ip prefix-list B permit 128.0.0.0/2 ge 16 le 32 => access-list 100 permit 128.0.0.0 63.255.255.255**  
**Class C: ip prefix-list C permit 192.0.0.0/3 ge 24 le 32 => access-list 100 permit 192.0.0.0 31.255.255.255**

## Match Classes

If a route is denied by ACL in „permit“ statement it doesn't mean route is not redistributed at all, it's just not matched by this entry

There is **IMPLICIT DENY** at the end of route-map

If no action or sequence number is specified when the route map is configured, the route map will default to a permit and a sequence number of 10

**match ip address 10**  
**match ip tag 2222**  
 Two different types of matches in the same route-map entry define **AND** operation (they all must match)

**match ip address 10 20**  
 Two the same types of matches in the same route-map entry define **OR** operation (any of them can match)

## Route-map

**(RM) continue <seq>**  
 Jump to specified seq or next seq if seq is not specified

If match clause exists, continue proceeds only if match is successful

If next RM entry (pointed by continue) also have continue clause but match does not occur, second continue is not processed, and next RM entry is evaluated

**Match**

- metric:** metric of the route (MED for BGP)
- route-type:** OSPF or EIGRP route type (external, internal, type 1 or 2)
- ip-address:** ACL defining specific prefix(es)
- ip-address prefix-list:** specific prefix and length (bit netmask)
- ip next-hop:** ACL defining route's next-hop (via in routing table)
- ip route-source:** ACL defining neighbor (from in routing table)
- tag:** route tag
- length:** packet length

Policy Based Routing proceeds through route-map until match is found. If no match is found or match is found in route-map deny statement, the packet not dropped, but it is forwarded according to normal destination-based process

## PBR

**(IF) ip policy route-map <name>**  
 Affects incoming packets only

**(IF) ip route-cache policy**  
 By default, PBR is process-switched unless CEF is enabled. Fast-switching is recommended if CEF is not enabled. It must be added before PBR is applied

**(IF) ip route-cache same-interface**  
 May be required if next-hop points to the same interface (ex. NBMA)

**(RM) set ip next-hop <ip> verify-availability**  
 Verify the availability of the next-hop address before attempting to forward the packet. The router will search CDP table to verify that the next-hop address is listed

**(RM) set ip next-hop <ip> track <id>**  
 next hop can be also tracked with Advanced Object Tracking. There can be many next hops defined in one route-map entry. If one fails, the next one is checked.

**(G) ip local policy route-map <name>**  
 for traffic originated by the router. It can be useful to pass router-generated traffic through ACL or CBAC. By default router-generated traffic does not pass any outbound ACLs.

# Routing features part 2

**(G) track <#> interface <iif> [line-protocol | ip routing]**  
 Interface IP routing will go down when line-protocol goes down or interface loosed IP address (assigned by DHCP or IPCP)

**(G) track <#> ip route <net>/<bits> [reachability | metric threshold]**  
 Track IP route reachability (in routing table) or route's metric. Route metric values are normalized to the range of 0 to 255, where 0 is connected and 255 is inaccessible. State is up if the scaled metric for that route is less than or equal to the up threshold. Tracking uses a per-protocol configurable resolution value to convert the real metric to the scaled metric

**(G) track resolution ip route [eigrp | isis | ospf | static] <resolution-value>**  
 Define resolutions for routes tracked with threshold. EIGRP resolution 256 - 40000000. ISIS resolution 1 - 1000. OSPF resolution 1 - 1562. Static resolution 1 to 100000

**(G) track <#> ip sla <#> [state | reachability]**  
 IP SLA tracking, in addition to up/down state, can set return codes

**(G) track <#> list {boolean {and | or} | threshold {weight | percentage}}**  
 List of tracked objects can be either ANDed or ORed. Objects can also be negated

**(G) track <#> stub-object**  
 Create dummy object that can be tracked and manipulated by EEM

**(G) track timer {interface | ip route | sla } | list | stub}{<sec> | msec <msec>}**  
 Defines interval during which the tracking process polls the tracked object. The default interval for interface polling is 1 sec, and for IP-route polling is 15 sec

```
track 1 interface serial0/0 line-protocol
track 2 interface serial0/1 line-protocol
track 12 list threshold weight
object 1 weight 5
object 2 weight 5
threshold weight up 10 down 0
Object is down if two interfaces are down
```

```
track 1 sla 1 reachability
delay down <sec> up <sec>
1. Track remote router with RTR
```

**(G) ip route 192.0.0.192 255.255.255.255 null 0 track 1**  
 2. Create bogus static routing, reacting to tracked RTR. Although the route is pointed to null0, which is always available, the route will be in the routing table only if status of tracked resource is UP

**(G) ip prefix-list TST permit 1.1.1.1/32**  
 3. Create prefix-list covering bogus route and assign it to route-map

```
route-map TST permit 10
match ip address prefix-list TST
4. Assign tracked prefix to route-map
```

**router rip**  
**default-information originate route-map TST**  
 5. Originate a default route (RIP in this example) only if route-map result is true, meaning the remote router is reachable

## Advanced Object Tracking

## Conditional 0/0 injection

## ODR

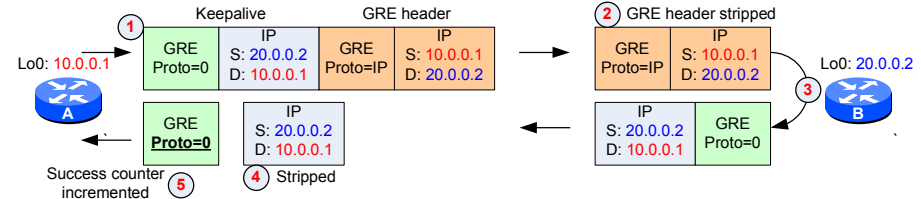
- (G) router odr**  
Configured on hub only
- Hub router can automatically discover stub networks. Stub routers use a default route to the hub (also learned via ODR: `0* 0.0.0.0 [160/1] via ...`)
- ODR carries only the network portion of the address, without a mask. Information is carried by CDP TLVs
- The metric (hop count) will never be more than 1
- Hello 60sec, Invalid 180sec – CDP timers are used. ODR advertisements stop if any other routing protocol is enabled on stub

## Backup interface

- (IF) backup interface <backup-intf>**  
The interface defined with this command can back up only one other interface. The backing up interface goes into standby mode and cannot be used to carry any traffic until activated.
- (IF) backup delay {<enable-delay> | never} {<disable-delay> | never}**  
To immediately switchover to backup interface specify delay = 0

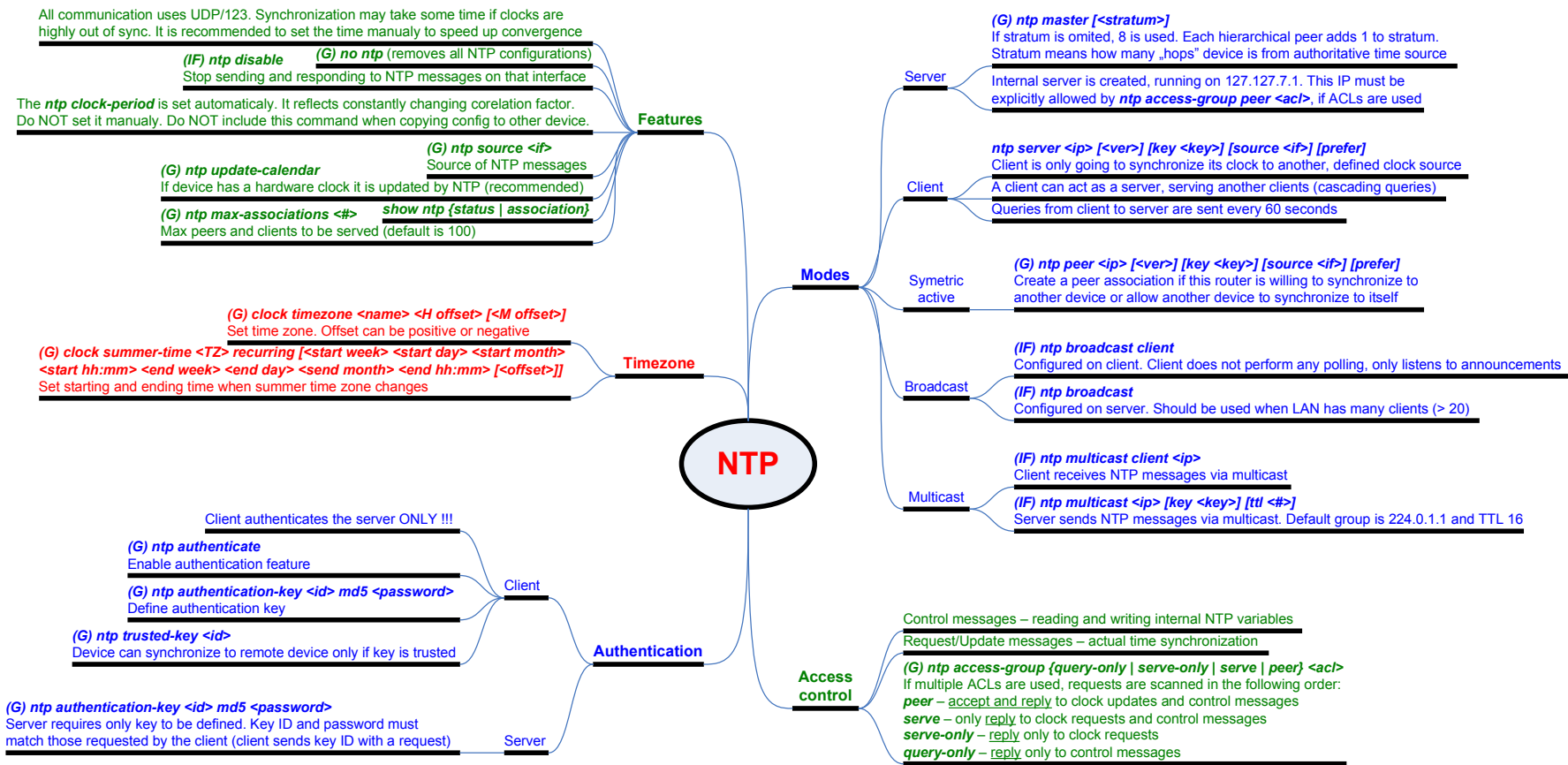
## GRE

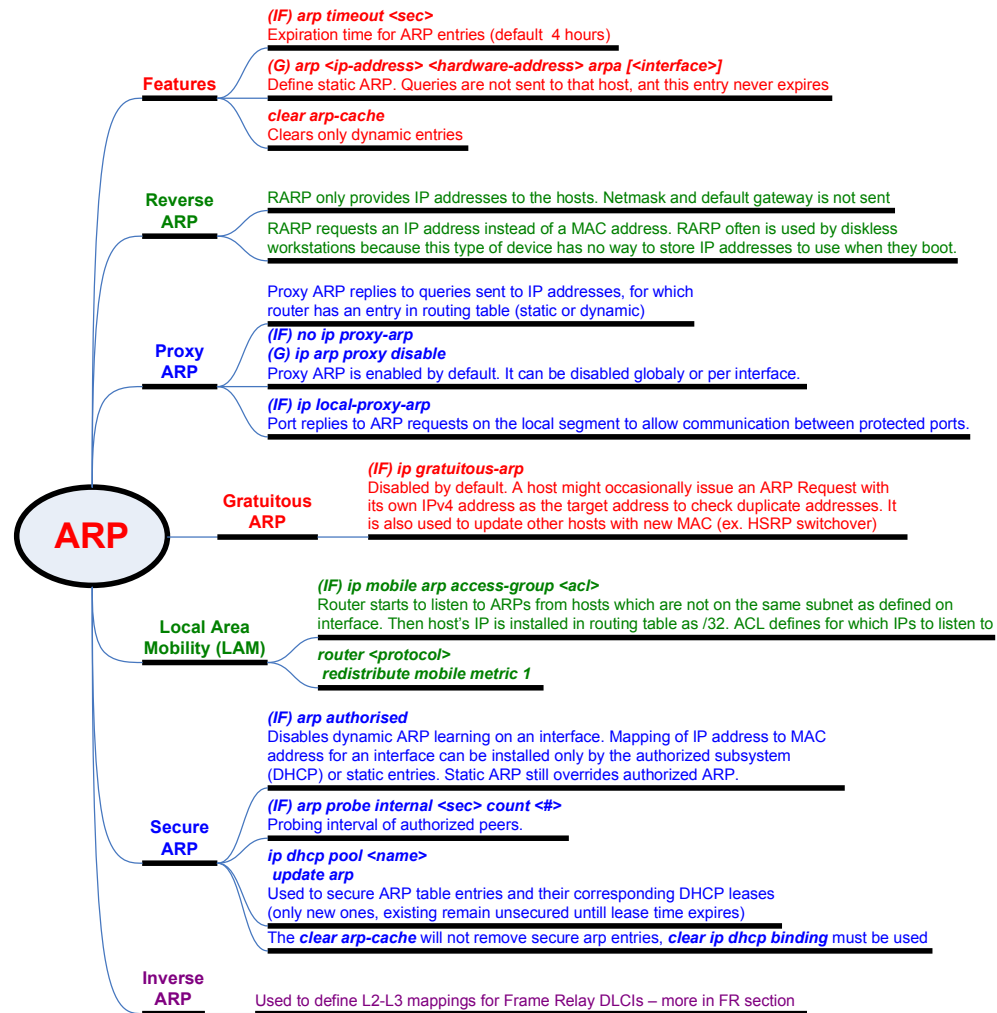
- Protocol number 47
- (IF) tunnel route-via <iif> {mandatory | preferred}**  
Tunnel route selection can be used, if there are multiple equal-cost paths to destination (only single route for tunnel destination is selected randomly). Mandatory: if there is no route via specified interface, tunnel goes down. Preferred: if there is no route via specified interface, tunnel takes next available path
- debug tunnel route-via**
- (IF) keepalive <sec> <retry count>**  
By default configured tunnel does not have the ability to bring down the line protocol of either tunnel endpoint, if the far end is unreachable. If keepalive is enabled, NAT cannot be used for GRE packets



## NSF & GR

- Non Stop Forwarding is a way to continue forwarding packets while control plane is recovering from failure**
- Graceful Restart is a way of rebuilding forwarding data in routing protocols when control plane has recovered**
- 1) If NSF capable control plane detects failure (neighbors down) it will not reset data plane, but will mark forwarding information as stale. Any traffic will be switched based on last known information
- 2) Control plane must recover before neighbor hold time expires. When control plane gets up, it signals the neighbor that it still forwards traffic, but would like to resync. This is GR message (protocol dependant)
- 3) Control plane must recover before neighbor hold time expires. When control plane gets up, it signals the neighbor that it still forwards traffic, but would like to resync
- 4) Neighbor then sends prefix updates. When done, end-of-table marker is sent
- 5) When end-of-table is seen, router recalculates topology and informs CEF, which removes stale entries





# Neighbor Discovery

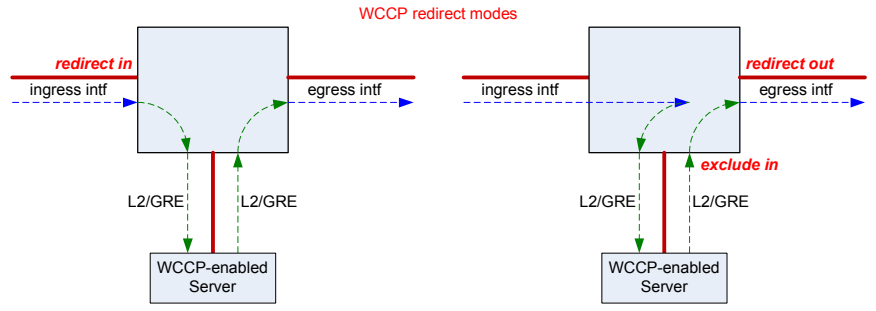
- CDP**
  - (G) `cdp run`  
(IF) `cdp enable`  
Enable CDP globally and per-interface
  - CDP runs on any media that supports the subnetwork access protocol (SNAP). CDP v2 contains 3 additional TLVs VTP domain, native vlan and interface duplex
  - Timers**
    - (G) `cdp timer <sec>`  
CDP messages advertisement interval (default 60 sec)
    - (G) `cdp holdtime <sec>`  
Inform receiving device, how long CDP messages should be stored locally (default 180)
  - (G) `no cdp advertise-v2`  
Disable V2 advertisements
  - (G/IF) `no cdp log mismatch duplex`  
Duplex mismatches are displayed for all Ethernet interfaces by default
  - (G) `cdp source-interface <if>`  
IP from this interface will be used to identify device (messages will be originated from this intf). It should not be an IP unnumbered interface
  - Verify**
    - `show cdp [interface <if> | entry <id>]`
    - `show cdp neighbors`
    - `clear cdp table`

## LLDP

- 802.1AB Link Layer Discovery Protocol runs on L2 like CDP. Composed of TLVs. Mandatory TLVs: Port description, System name, System description, System capabilities, management address
- LLDP-MED (Media Endpoint Devices) – extension to LLDP to discover devices like IP Phones (describes VLAN, QoS (network policy), Power, Inventory – SN
- (IF) `lldp med-tlv-select {inventory-management | location | network-policy | power-management}`  
By default only standard LLDP messages are sent, until LLDP-MED is heard from attached device. Then, extended TLVs are send back to device. By default all available types of TLVs are send back. They can be filtered
- (G) `lldp run`  
EnableLLDP globally
- (IF) `lldp {transmit | receive}`  
Enable/disable LLDP on interface
- (G) `network-policy profile <#>`  
Network policy defines characteristics for attached device. It is not supported on private vlan port
- Timers**
  - (G) `lldp holdtime <s>`  
How long attached device should hold policy information (default 120 sec)
  - (G) `lldp timer <s>`  
Sending frequency (default 30 sec)
  - (G) `lldp reinit <s>`  
Delay before initializing LLDP on interface (default 2 sec)
- Verify**
  - `show lldp [{entry <id> | neighbors [detail] | interface <if>}]`
  - `show network-policy profile`
  - `clear lldp {table | counters}`
- Options for `lldp med-tlv-select`:
  - `{voice | voice-signaling} [vlan {<vlan-id> | dot1p} {cos <cos> | dscp <dscp>}] | none | untagged`
  - `vlan` – native vlan for voice traffic
  - `dot1p` – use vian0
  - `none` – do not instruct the phone about vlan
  - `untagged` – phone sends untagged traffic (default)
- Options for `network-policy profile`:
  - (IF) `network-policy <#>`  
Apply policy to interface. Switchport voice vlan must be defined first
  - (IF) `lldp med-tlv-select network-policy`  
Enable LLDP to send network-policy TLVs

# WCCP

- Features**
  - WCCP works only with IPv4 networks. Uses UDP/2048
  - Up to 32 Content Engines for a router in WCCPv1. CE with lowest IP is elected as leading Content Engine
  - WCCPv1 supports only HTTP (port 80) traffic
  - In WCCPv2 (default) there can be more than one router serving Content Engine cluster
  - WCCPv2 supports MD5 authentication and load distribution
  - When WCCP forwards traffic via GRE, the redirected packets are encapsulated within a GRE header, and a WCCP redirect header. When WCCP forwards traffic using L2 (Cache Engine is on the same segment as the router), the original MAC header of the IP packet is overwritten and replaced with the MAC header for the WCCP client.
- Configuration**
  - (G) `ip wccp web-cache` (enable WCCP)
  - (G) `ip wccp web-cache group-address <multicast> password <pass>`
  - (G) `ip wccp web-cache redirect-list <acl>` - for which clients redirection is enabled
  - (G) `ip wccp web-cache group-list <acl>` - which cache engines are allowed to participate
  - (IF) `ip wccp web-cache redirect in` – select interface toward local LAN
  - (IF) `ip wccp web-cache redirect out` – select interface toward Internet)
  - (IF) `ip wccp redirect exclude in`  
Exclude interface from redirection (usually interface where WCCP server is located – redirect out mode)
  - (G) `ip wccp mode {open | closed}`  
When closed mode is enabled, and a content engine is not available, all traffic which would normally be passed through it, is blocked
  - `show ip wccp`
  - `show wccp status`



# OER/PfR Basics

## Features

- Communication between MC and BR – UDP/3949, TCP/3949
- Traditional routing uses static metrics and destination-based prefix reachability. Network recovery is based on neighbor and link failures. PfR enhances routing to select the best path based on measurements and policy
- OER monitors traffic class performance and selects the best entrance or exit for traffic class. Adaptive routing adjustments are based on RTT, jitter, packet loss, MOS, path availability, traffic load and cost policy
- Minimum CPU impact. Utilizes lot's of memory (based on prefixes). MC is the most impacted.
- The preferred route can be an injected BGP route or an injected static route
- PfR is a successor of OER. OER provided route control on per destination prefix basis. PfR expands capabilities that facilitate intelligent route control on a per application basis
- OER can learn both outside and inside prefixes.
- Master controller and Border Router can be enabled on the same router

## Master Controller

- Monitors the network and maintains a central policy database with statistics. Verifies that monitored prefix has a parent route with valid next hop before it asks BR to alter routing
- Does not have to be in forwarding path, but must be reachable by BRs
- Long-term stats are collected every 60 min. Short-term stats are collected every 5 min
- Support up to 10 border routers and up to 20 OER-managed external interfaces
- MC will not become active if there are no BRs or only one exit point exists
- Can be shutdown with `shutdown` command

### Features

#### (G) oer master

Enable OER master controller. Below commands are defined in its context

```
border <ip> [key-chain <name>]
```

At least one BR must be configured. Key chain is required when adding BR for the first time. It's optional when reconfiguring existing BR

#### interface <if> {external | internal}

Define interfaces which are used on BR (must exist on BR)

#### port <port>

Dynamic port used for communication between MC and BR. Must be the same on both sides

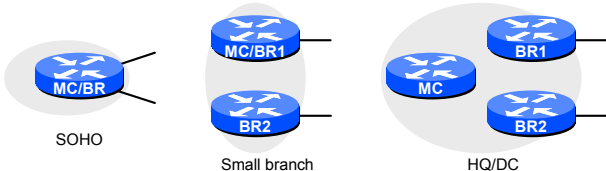
#### logging

Enables syslog messages for a master controller (notice level)

#### keepalive <sec>

Keepalive between MC and BR. Default is 60 sec.

### Config



## Border Router

- Edge router with one or more exit links to an ISP or WAN
- Enforces policy changes so it must be in the forwarding path
- Reports prefix and exit link measurements to MC
- `ip nat inside source list 1 interface virtual-template 1 overload oer`
- NAT awareness for SOHO. NAT session will remain in case of route change via second ISP

### Features

#### (G) oer border

Enable OER border router

#### port <port>

Port used between MC and BR

#### local <intf>

Identifies source for communication with an OER MC

#### master <ip> key-chain <name>

Define MC. Key chain is mandatory

### Config

## Phases Wheel

### Learn (BR)

- The list of traffic classes entries is called a Monitored Traffic Class (MTC) list. The entries in the MTC list can be profiled either by automatically learning the traffic or by manually configuring the traffic classes (both methods can be used at the same time)
- BR profiles interesting traffic which has to be optimized by learning flows that pass through a router. Non-interesting traffic is ignored
- BR sorts traffic based on delay and throughput and sends it to MC
- Next hops on each border router cannot be from the same subnet (exchange points)

### Measure (BR)

- PfR automatically configures (virtually) IP SLA ICMP probes and NetFlow configurations. No explicit NetFlow or IP SLAs configuration is required
- OER measures the performance of traffic classes using active and passive monitoring techniques but it also measures, by default, the utilization of links
- Active monitoring generates synthetic traffic to emulate the traffic class that is being monitored
- Passive monitoring measures metrics of the traffic flow traversing the device in the data path
- By default all traffic classes are passively monitored using integrated NetFlow functionality and out-of-policy traffic classes are actively monitored using IP SLA functionality (learned probe)

### Apply Policy (MC)

- If multiple exits exist including existing one, use existing one, otherwise randomly pick exit
- OER compares the results with a set of configured low and high thresholds for each metric
- olicies define the criteria for determining an Out-Of-Profile event.
- Can be applied globally, per traffic (learned automatically or defined manually) class and per external link (overwrites previous)
- By default, OER runs in an observe mode during the profile, measure, and apply policy phases (no changes to network are made until OER is configured to control the traffic)
- Every rule has three attributes: scope (traffic class), action (insert a route), and condition that triggers the rule (acceptable thresholds)

### Enforce (BR)

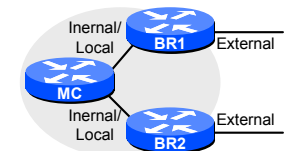
- Routing can be manipulated with artificially injected more-specific routes. Measured prefixes' parent route (the same or wider prefix) with a valid next hop must exist for prefix to be injected
- In control mode commands are sent back to the border routers to alter routing in the OER managed network to implement the policy decisions
- If an IGP is deployed in your network, static route redistribution must be configured
- OER initiates route changes when one of the following occurs: traffic class goes OOP, exit link goes OOP or periodic timer expires and the select exit mode is configured as select best mode

### Verify (MC)

After the controls are introduced, OER will verify that the optimized traffic is flowing through the preferred exit or entrance links at the network edge

## Interfaces

- Local interfaces – used for communication between MC and BRs. loopback interface should be configured if MC and BR are on the same router. Configured only on BR
- Internal interfaces - used only for passive performance monitoring with NetFlow. NetFlow configuration is not required. Internal interfaces do not forward traffic
- External interfaces - OER-managed exit links to forward traffic. At least two for OER-managed domain, at least one on each BR



## Authentication

#### key chain <name>

#### key <id>

#### key-string <text>

Authentication is required. MD5 key-chain must be configured between MC and BRs, even if they are configured on the same router. Key-ID and key-string must match on MC and BR

## Verify

- `show oer {master | border}`
- `show oer master traffic-class`
- `show oer master prefix <prefix> policy`
- `show oer border passive learn`
- `show ip cache verbose flow`
- `show oer border passive cache {learned | prefix} [applications]`

Loss – counters are incremented if retransmission takes place (repeated sequence number in TCP segment)

Delay – only for TCP flows (RTT between sending TCP segment and receipt of ACK)

Throughput – total number of packets sent (all types of traffic)

Reachability – tracks SYN without corresponding ACK

**oer master**

**mode monitor passive**

Enable measuring performance globally for all traffic flowing through device

**oer-map <name> <seq>**

**set mode passive**

Enable measuring performance metrics for particular prefixes

**oer master**

**mode monitor both**

Active and Passive enabled together (different than fast failover). Default mode.

**oer master**

**mode monitor fast**

fast failover - all exits are continuously probed using active monitoring and passive monitoring. Probe frequency can be set to a lower frequency than for other monitoring modes, to allow a faster failover capability. Failover within 3 sec.

### Mixed modes

After external interface is configured for BR, OER automatically monitors utilization of that link. BR reports link utilization to MC every 20 sec

**oer master**

**border <ip>**

**interface <if> external**

**max-xmit-utilization [receive] {absolute <kbps> | percentage <%>}**

Define maximum utilization on a single OER managed exit link (default 75%)

**oer master**

**max-range-utilization percent <max %>**

**max range receive percent <max %>**

Set maximum utilization range for all OER-managed exit links. OER keeps the links within utilization range, relative to each other. Ensures that the traffic load is distributed. If the range falls below threshold OER will attempt to move some traffic to use the other exit link to even the traffic load

### Link Utilization

**(MC) learn**

Enable automatic prefix learning on MC (OER Top Talker and Top Delay)

**delay**

Enables prefix based on the highest delay time. Top Delay prefixes are sorted from the highest to lowest delay time and sent to MC

**throughput**

Enable learning of top prefixes based on the highest outbound throughput

**monitor-period <minutes>**

Time period that MC learns traffic flows. Default 5 min

**periodic-interval <minutes>**

Time interval between prefix learning periods. Default 120 min

**prefixes <number>**

Number of prefixes (100) that MC will learn during monitoring period

**expire after {session <number> | time <minutes>}**

Prefixes in central DB can expire either after specified time or number of monitoring periods

**aggregation-type {bgp | non-bgp | prefix-length <bits>}**

Traffic flows are aggregated using a /24 prefix by default

**bgp** – aggregation based on entries in the BGP table (matching prefix for a flow is used as aggregation)

**non-bgp** – aggregation based on static routes (BGP is ignored)

**prefix-length** - aggregation based on the specified prefix length

**inside bgp**

Enable automatic prefix learning of the inside prefixes

**protocol {<#> | tcp | udp} [port <#> | gt <#> | lt <#> | range <lower> <upper>] [dst | src]**

Automatic learning based on a protocol or port number (application learning). Aggregate only flows matching specified criteria. There can be multiple protocol entries for automatic application learning.

## OE/PfR Measuring

### Automatic learning (learn)

## OER/PfR Learning

### Manual learning

Delay, Jitter, MOS are monitored using IP SLA probes

Reachability – tracks SYN without corresponding ACK

Learned probes (ICMP) are automatically generated when a traffic class is learned using the NetFlow

To test the reachability of the specified target, OER performs a route lookup in the BGP or static routing tables for the specified target and external interface

longest match assignment

**oer master**

**active-probe {echo <ip> | tcp-conn <ip> target-port <#> | udp-echo <ip> target-port <#>}**

A probe target is assigned to traffic class with the longest matching prefix in MTC list

Forced target assignment

**oer-map <name> <seq>**

**match ip address {access-list <name> | prefix-list <name>}**

**set active probe <type> <ip> [target-port <#>] [codec <name>]**

**set probe frequency <sec>**

Default frequency is 60 sec.

**ip sla monitor responder ...**

IP SLA responder must be configured on remote device

**oer master**

**mode monitor active [throughput]**

Uses integrated IP SLA. Active throughput uses SLA and NetFlow at the same time

**oer border**

**active-probe address source interface <if>**

By default active probes are sourced from an OER managed external interfaces

**show oer master active-probes [appl | forced]**

**oer-map <name> <seq>**

**match ip address {access-list <name> | prefix-list <name> [inside]}**

Only a single match clause (regardless of type) may be configured for each sequence. All sequence entries are permit, no deny.

**oer-map <name> <seq>**

**match oer learn {delay | inside | throughput | list <acl>}**

Match OER automatically learned prefix

**oer master**

**policy-rules <map-name>**

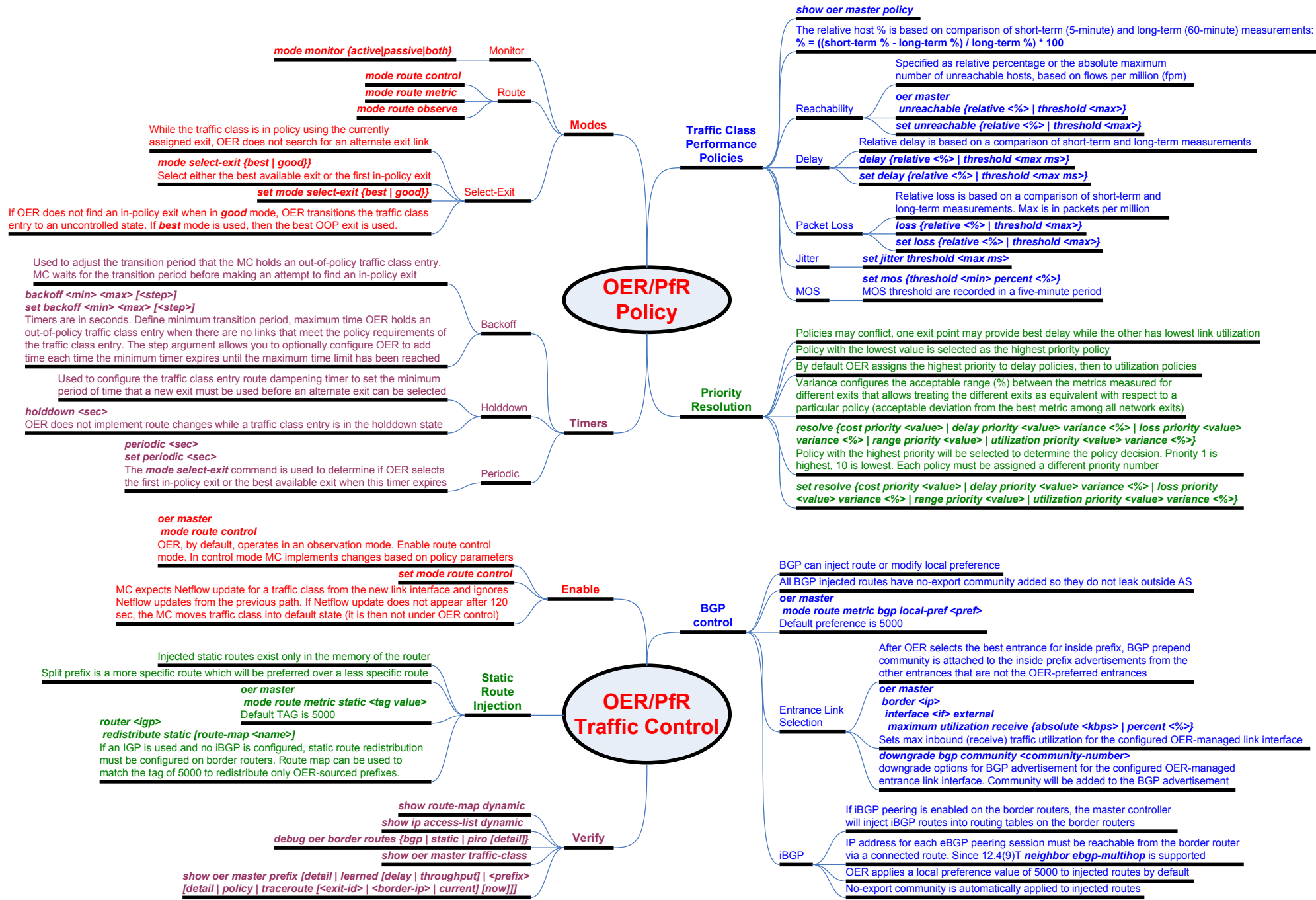
Associate OER map with MC configuration

OER will not control inside prefix unless there is exact match in BGP

RIB because OER does not advertise new prefix to the Internet

Prefix-list **ge** is not used and **le 32** is used to specify only inclusive prefix

Only named extended ACLs are supported



# Cisco HSRP

## Features

- (IF) standby version {1 | 2}** Cisco proprietary. UDP/1985  
V2 has different frame format (TLV), incompatible with V1. Default is V1
  - Version 1**  
Hello multicasted to 224.0.0.2  
Virtual MAC: 0000.0C07.ACxx, where xx – group #. Up to 255 groups per interface
  - Version 2**  
Hello multicasted to 224.0.0.102  
Virtual MAC: 0000.0C9F.Fxxx, where xxx – group #. Up to 4095 groups per interface, but platform-dependant, per-interface recommended limits still apply
- (IF) standby [<#>] ...** If group # is not defined, 0 is used
- (IF) standby name <name>**  
The HSRP group name must be unique on the router. It is assigned automatically (ex. Group name is "hsrp-Fa0/0-1"), but can be defined to be more informative
- (G) standby redirects [{enable | disable}]** **show standby [brief]**
- (IF) standby redirect [timers <adv> <hold>]**  
Real IP address of a router can be replaced with a virtual IP address in NH/GW field of the ICMP redirect packet. Default advertisement is 60 sec, holddown is 180 sec.
- (IF) no standby redirect unknown**  
Allows redirects only between routers configured for HSRP for particular group. If NH is a router for which real IP to virtual IP mapping is not defined, redirect is not ent.
- (IF) standby bfd**  
**(G) standby bfd all-interfaces**  
HSRP support for BFD is enabled by default
- Duplicate address rather indicates STP problem, than HSRP problem. Duplicate Hello packet is ignored, and does not affect HSRP operation. Duplicate messages are throttled at 30-sec intervals.

## Timers

- (IF) standby timers [msec] <hello> [msec] <hold>**  
Default Hello 3 sec, holdtime 10 sec. All routers in a group should use the same timers. If msec is used, timers are not propagated inside hellos.
- (IF) standby delay minimum <sec> reload <sec>**  
**Minimum** defines delay for HSRP initialization after an interface comes up. Default is 1 sec, recommended 30 sec. Delay after reload is 5 sec, recommended 60 sec. The delay will be cancelled if an HSRP packet is received on an interface

## Authentication

- (IF) standby authentication md5 key-string <pw> [timeout <sec>]**  
Timeout defines how long OLD key will be valid. Timeout is valid only for key-string, as key-chain can define own timeouts within key-chain context
- (IF) standby authentication md5 key-chain <name>**
- (IF) standby authentication text <pw>**  
Password is sent unencrypted in all HSRP messages

## Semi-Load balancing

- Load-balancing possible with different groups on the same interface. Some hosts use one default GW, other hosts use different GW (within the same segment)

<b>Router B:</b> interface fastethernet0/0 ip address 10.0.0.2/24 standby 1 ip 10.0.0.3 standby 1 priority 95 standby 2 ip 10.0.0.254 standby 2 priority 105	<b>Router A:</b> interface fastethernet0/0 ip address 10.0.0.1/24 standby 1 ip 10.0.0.3 standby 1 priority 105 standby 2 ip 10.0.0.254 standby 2 priority 95
--	--

## States

- Initial - not enabled yet, interface activated
- Learn - virtual IP is not known yet, and has not seen messages from active router
- Listen - router knows virtual IP, but is neither active, nor standby
- Speak - actively participate in election (must have virtual IP configured)
- Standby – monitoring the active router, ready to take over
- Active – router actively responding to ARPs
- One Active router (with highest priority), one Standby router, remaining routers in a group are in listen-state. Only Active and Standby routers generate messages. If standby router becomes active, other router (currently listening, and with highest priority) becomes standby router.
- (IF) standby priority <#>**  
Highest priority (0-255) wins (multicast), default is 100
- (IF) standby preempt [delay {minimum <sec> | reload <sec>}]**  
If local router has priority higher than the current active router, it should attempt to become active router. No preemption by default. If enabled, default delay is 0 – immediate.
- (IF) standby <#> follow <group-name>**  
HSRP group can become a redundancy client of another HSRP group. Client or slave groups must be on the same physical interface as the master group. Recursive following is not possible
- (IF) standby ip <ip> [secondary]**  
Secondary IP addresses/subnets can also run HSRP. There can be many secondary entries for the same group. Primary and secondary IPs can be used together.
- At least one router must have IP address in HSRP group. Other routers can learn via hello (Virtual IP is sent within HSRP messages)

## MAC

- (IF) standby 1 mac-address <MAC>**  
MAC address can be defined statically. When router becomes active, virtual IP is moved to different MAC. The router sends gratuitous ARP to update hosts
- (IF) standby use-bia [scope interface]**  
If router/switch has limitations for number of groups (MAC chip must support many programable MAC addresses), it can be solved with "standby use-bia" command. Without the scope, **use-bia** applies to all subinterfaces on the major interface
- Active router sources Hellos from configured real IP and virtual MAC. Standby router sources Hellos from configured real IP and BIA MAC address.
- When ARP is sent from PC to active router's virtual IP (default GW), virtual MAC is sent in reply
- When ARP is sent from PC to active router's real IP, router's BIA MAC is sent in reply
- When ARP is sent from PC to standby router's real IP, router's BIA MAC is sent in reply
- HSRP supports Proxy ARP. If request is received, active router responds with virtual MAC.
- (IF) standby arp gratuitous [count <#>] [interval <sec>]**  
HSRP sends one gratuitous ARP packet when a group becomes active, and then another packet after two and four seconds
- standby send arp [<i>] [<group-number>]**  
Send single gratuitous ARP packet for each active group. ARP cache is verified and re-built before sending gARP

## Tracking

- When tracking is used, the state change is reflected immediately, regardless of hello and hold timers
- Decrement priority for multiple interfaces is cumulative only if each intf is configured with priority value (different than 10). If no priority is defined only single total decrement by 10 is used, regardless of number interfaces in down state
- (IF) standby 1 track <interface> <decrement>**  
Only HSRP can track interface directly (physical state), without tracking objects
- (G) track 13 interface serial0/1 line-protocol**
- (IF) standby 1 track 13 decrement 20**

# Cisco GLBP

## Features

- (IF) glbp [<#>] ...**  
Hello multicasted to 224.0.0.102 UDP/3222  
Max 1024 GLBP groups per physical interface. Default group is 0 (not shown in config)
- AVG assigns unique MAC to each router: 0007.B400.xxyy, xx – group #, yy – router #**  
One primary AVG, one backup AVG, other members in a group are in listening state. If primary fails, one of AVF with highest priority/IP (backup AVG) is elected to be primary AVG. Other routers in listening state can become primary AVF
- Up to 4 primary forwarders in a group. They have MAC addresses assigned by AVG in a sequence. Other routers in a group are secondary forwarders in listening state – they learn virtual MACs via Hello
- (IF) glbp priority <1-255>**  
Higher priority is better (default 100). If priority is the same, higher IP address wins
- (IF) glbp ip [<ip> [secondary]]**  
IP has to be defined on AVG. GLBP can also run for secondary addresses
- (IF) glbp client-cache maximum <#> [timeout <sec>]**  
AVG keeps client cache containing which AVF is assigned to which host. Max 2000 hosts. If max is reached, oldest entries are removed. Timeout defined how long entries are kept in cache (without ARP query from a client). Recommended timeout – little longer than ARP cache timeout
- If AVF fails, other AVF awaiting in listening state, becomes primary AVF. The AVG starts two timers for failed AVF, redirect and timeout
- (IF) glbp preempt [delay minimum <sec>]**  
No AVG preemption by default. Delay can be defined before preemption takes place
- (IF) glbp forwarder preempt [delay minimum <sec>]**  
Backup AVF can become active AVF if weighting drops below low threshold for 30 sec. This feature is enabled by default
- show glbp [(brief | detail)]**

## Timers

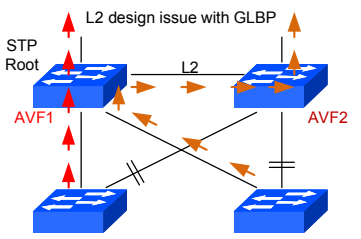
- (IF) glbp timers [msec] <hello> [msec] <hold>**  
Default Hello 3 sec. Holdtime 10 sec. Sub-second hello can be configured
- (IF) glbp timers redirect <redirect> <timeout>**  
redirect – during this time, AVG keeps redirecting hosts to that AVF  
timeout – after this time, AVF is removed from all gateways in a group

## Authentication

- Authentication schema is the same as for HSRP
- (IF) glbp authentication text <pw>**
- (IF) glbp authentication md5 key-chain <name>**
- (IF) glbp authentication md5 key-string <pw>**

## True Load balancing

- (IF) glbp weighting track <id> [decrement <value>]**  
**(IF) glbp weighting <max> [lower <lower>] [upper <upper>]**  
When two interfaces are tracked and both are down, the decrement is cumulative. If weight drops below lower mark AVF stops forwarding, when it reaches upper mark it re-enables forwarding
- (IF) glbp load-balancing {host-dependent | weighted | round-robin}**  
Define load-balancing method. AVG by default responds to hosts' ARP with virtual MAC requests in round-robin fashion
- Host-dependent load balancing is required by SNAT. Not recommended for small number of hosts. Given host is guaranteed to use the same MAC
- In weighted mode each router advertises weighting and assignments. Weighted load-balancing in ratio 2:1  
**RT1: glbp 1 weighting 20**  
**RT2: glbp 1 weighting 10**



# Other FHRP

## Features

- Hello sent to 224.0.0.18 (own protocol number: 112)
- Virtual MAC: 0000.3E00.01xx, xx – group #. MAC address cannot be changed manually. Max 255 groups per interface
- Semi-load balancing is possible with many groups and different default gateways set for hosts
- (IF) vrrp [<#>] ip [<ip> [secondary]]**  
All members must be configured with the same primary subnet, otherwise routers will not become members (they will act independently)
- (IF) vrrp priority <1-254>**  
Higher is better. Default 100. If priority is the same, higher IP address wins
- (IF) vrrp preempt [delay minimum <sec>]**  
Preemption enabled by default. Delay is 0 sec - immediate
- (IF) vrrp [<#>] shutdown**  
Disable VRRF for a certain group without removing configuration
- (IF) vrrp track <obj> [decrement <value>]**  
Uses IOS object tracking only
- show vrrp [(interface | brief)]**

## Timers

- (IF) vrrp timers advertise [msec] <sec>**  
Master advertises timers. Default Hello is 1 sec, Holdtime is 3 sec
- (IF) vrrp timers learn**  
Learn timers from master when acting as slave

## Authentication

- Authentication schema is the same as for HSRP
- (IF) vrrp authentication md5 key-string <pw> [timeout <sec>]**
- (IF) vrrp authentication md5 key-chain <name>**
- (IF) vrrp authentication [text] <pw>**

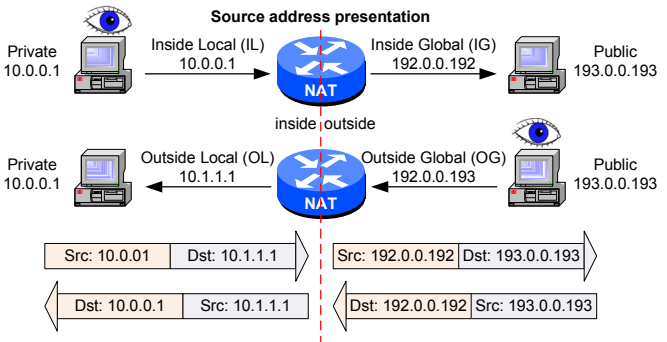
## DRP

- It enables the Cisco Distributed Director product to query routers (DRP agent) for BGP and IGP routing table metrics between distributed servers and clients
- Distributed Director is a standalone product that uses DRP to transparently redirect end user service requests to the topologically closest responsive server
- ip drp server**
- ip drp access-group <acl>** (limit source of DRP queries)
- ip drp authentication key-chain <key>**

## IRDP

- ICMP Router Discovery Protocol. Uses ICMP messages to advertise candidate default gateway. By default messages are broadcasted
- (IF) ip irdp address <ip> <preference>**  
Advertises IP address configured on interface as a gateway. Optionally, different IPs (many) can be advertised with different priorities (all defined IPs are advertised)
- Advertisements vary between **minadvertinterval** and **maxadvertinterval**
- (IF) ip irdp**
- (IF) ip irdp multicast** (enable multicasting to 224.0.0.1)
- (IF) ip irdp holdtime <sec>** (default is 30 min)
- (IF) ip irdp maxadvertinterval <sec>** (default is 450 sec)
- (IF) ip irdp minadvertinterval <sec>** (default is 600 sec)
- (IF) ip irdp preference <#>** (default is 0; higher is better)
- Server**
- (G) no ip routing**
- (G) ip gdp irdp**
- Client**

- Inside-to-Outside**
- if IPsec then check input access list
  - decryption
  - input access list (again, if IPsec)
  - input rate limits
  - input accounting
  - redirect to web cache
  - policy routing
  - routing
  - **NAT inside to outside**
  - crypto (mark for encryption)
  - output access list
  - inspect (CBAC)
  - TCP intercept
  - encryption
  - queuing



- Outside-to-Inside**
- If IPsec then check input access list
  - decryption
  - input access list
  - input rate limits
  - input accounting
  - redirect to web cache
  - **NAT outside to inside**
  - policy routing
  - routing
  - crypto (mark for encryption)
  - output access list
  - inspect (CBAC)
  - TCP intercept
  - encryption
  - queuing

# NAT part 1

## Features

- Inside local** – how inside address is seen locally (by inside hosts)
- Inside global** – how inside address is seen globally (by outside hosts)
- Outside local** – how outside address is seen locally (by inside hosts)
- Outside global** – how outside address is seen globally (by outside hosts)
- Not supported: Routing table updates, DNS zone transfers, BOOTP, SNMP
- (IF) ip nat {inside | outside}** - Define interface role for NAT
- If router does not have a route to destination, packet is unroutable, and does not use NAT. This can be also a case when **no ip classless** is configured
- If a translation entry already exists and matches traffic then it this entry will be used, and neither access lists nor route map will be consulted
- NAT keeps stateful information about fragments. If a first fragment is translated, information is kept so that subsequent fragments are translated the same way.
- If a fragment arrives before the first fragment, the NAT holds the fragment until the first fragment arrives

## FTP Pasive

- PORT and PASV commands carry IP addresses in ASCII form
- When the address is translated, the message size can change. If the size message remains the same, the Cisco NAT recalculates only the TCP checksum
- If the translation results in a smaller message, the NAT pads the message with ASCII zeros to make it the same size as the original message
- TCP SEQ and ACK numbers are based directly on the length of the TCP segments. NAT tracks changes in SEQ and ACK numbers. It takes place if translated message is larger than original one

## Static

- (G) ip nat inside source static <inside local> <inside global>**  
Static NAT (for 1:1 IP address) performs translations in both directions. Packets initiated from outside into inside are translated, but also packets initiated from inside to outside are translated.
- (G) ip nat inside source static network <local net> <global net> /24**  
Network translation assigns last octet one-to-one
- (G) ip nat inside source static tcp 192.168.1.1 21 192.1.1.3 21 extendable**
- (G) ip nat inside source static tcp 192.168.1.3 80 192.1.1.3 80 extendable**  
Statically mapping an IG address to more than one IL address is not allowed. To allow service distribution **extendable** keyword must be used. This is only for incoming traffic from outside. Outgoing traffic falls under dynamic NAT. If it's not configured, traffic is dropped
- (G) ip nat inside source static tcp <IL> <port> <IG> <port> [no-alias]**  
By default IG address is added to local IP aliases (**show ip alias**), so the router can terminate traffic (other than NATed) on itself, using this IP. If **no-alias** keyword is used, IG address is not added to aliases. Router will not terminate the traffic, but it will respond to ARP requests.
- (G) ip nat inside source static <IL> <IG> redundancy <name>**  
Redundancy with HRP. Active router is performing NAT translation

## Dynamic

- PAT**  
**(G) ip nat inside source list <acl> interface <if> overload**  
All inside sources are translated to single interface IP address. Up to 65535 IL addresses could theoretically be mapped to a single IG address (based on the 16-bit port number)  
Each NAT entry uses approximately 160 bytes of memory, so 65535 entries would consume more than 10 MB of memory and large amounts of CPU power
- (IF) ip nat pool <name> <start> <end> {netmask <mask> | prefix-length <prefix>} [type match-host]**  
Host portion of the IG address will match the host portion of the IL address if **match-host** is used. The **netmask** acts as a sanity check, ensuring that such addresses as 204.15.87.255 are not mapped
- (G) ip nat inside source list <acl> pool <name>**  
Translate dynamically source addresses of inside hosts  
When IG or OL addresses belong to directly attached interface, router created **ip aliases**, so it can answer ARP requests. If there is no NAT entry for such address, and router runs specific service, it can be attacked – router answers to packets (ICMP or UDP) not really destined for it

## Stateful

- (G) ip nat inside source list <acl> pool <name> mapping <mapping id>**  
**ip nat stateful id <id> redundancy <HSRP name> mapping-id <id>**  
Mapping-id identifies translations and must be the same on both routers. Stateful-id must be unique on each router
- With HSRP**  
R1: **ip nat stateful id <id> primary <R1 IP> peer <R2 IP> mapping-id <id>**  
R2: **ip nat stateful id <id> backup <R2 IP> peer <R1 IP> mapping-id <id>**
- Without HSRP**  
**show ip snat peer <ip>** - show translations on peer router  
**show ip snat distributed verbose**

## Virtual reassembly

- Router tracks fragments and delays them (holds) until all fragments are received or reassembly timeout expires (then incomplete packet is dropped). It is "virtual" reassembly, as packet is not put back into one, but only stored locally for NAT processing, after which, all fragments are sent to destination
- (IF) ip virtual-reassembly [max-reassemblies <#>] [max-fragments <#>] [timeout <sec>] [drop-fragments]**  
**max-reassemblies** – defines max simultaneous packets to be tracked. Drops packets if max is reached  
**max-fragments** – max number of fragments for single packet (exceeding will be dropped)  
**timeout** – how long router will wait for all fragments before dropping whole incomplete packet  
**drop-fragments** – drop all fragments arriving on interface

## Verify

- show ip nat translation**
- show ip nat statistics**
- clear ip nat translation \***
- NAT translation failure codes (**debug ip nat**)  
A = Inside to outside fails after routing  
B = Outside to inside fails before routing  
C = Outside to inside fails after routing  
D = Helped fails  
L = Internally generated packet fails  
E = Inside to outside fails after routing

If inside host opens route-map (only) based dynamic translation, outside host can be also able to initiate connection to inside host (bi-directional traffic initiation is allowed for specific one-to-one mapping, which is created in addition to extendable mapping)  
**ip nat inside source route-map ISP2\_MAP pool ISP2 reversible**

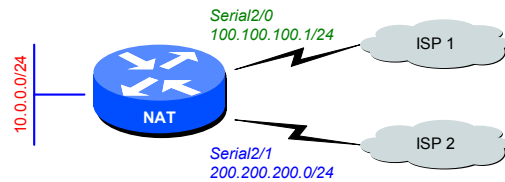
```
ip nat pool ISP1 100.100.100.10 100.100.100.50 prefix-length 24
ip nat inside source route-map ISP1_MAP pool ISP1
```

```
ip nat pool ISP2 200.200.200.10 200.200.200.50 prefix-length 24
ip nat inside source route-map ISP2_MAP pool ISP2
```

```
route-map ISP1_MAP permit 10
match ip address 1
match interface Serial2/0 ! outgoing interface
```

```
route-map ISP2_MAP permit 10
match ip address 1
match interface Serial2/1 ! outgoing interface
```

```
access-list 1 permit 10.0.0.0 0.0.0.255
```



### Multihoming to 2 ISPs

## NAT part 2

### Timeouts

```
timeout: 86400 sec
icmp-timeout: 60 sec
udp-timeout: 300 sec
tcp-timeout: 86400 sec
dns-timeout: 60 sec
syn-timeout: 60 sec
finrst-timeout: 60 sec
max-entries:
```

### Load balancing

In NAT TCP load balancing, non-TCP packets pass through the NAT untranslated

1. Define local servers LL addresses:  
**ip nat pool <name> <start> <end> prefix-length <bits> type rotary**  
 or using more flexible way:  
**ip nat pool <name> prefix-length <bits> type rotary**  
**address <start1> <end1>**  
**address <start2> <end2>**

2. Associate global IP (single IPs), by which local servers are seen from outside  
**ip nat inside destination list <acl> pool <name>**  
**access-list <acl> permit <global IP>**

**ip alias <global IP> <port>**

It may be required to create an IP alias for global IP, so the router accepts traffic for that IP if extended ACL is used with specific port numbers. The IP alias is not automatically created by the NAT

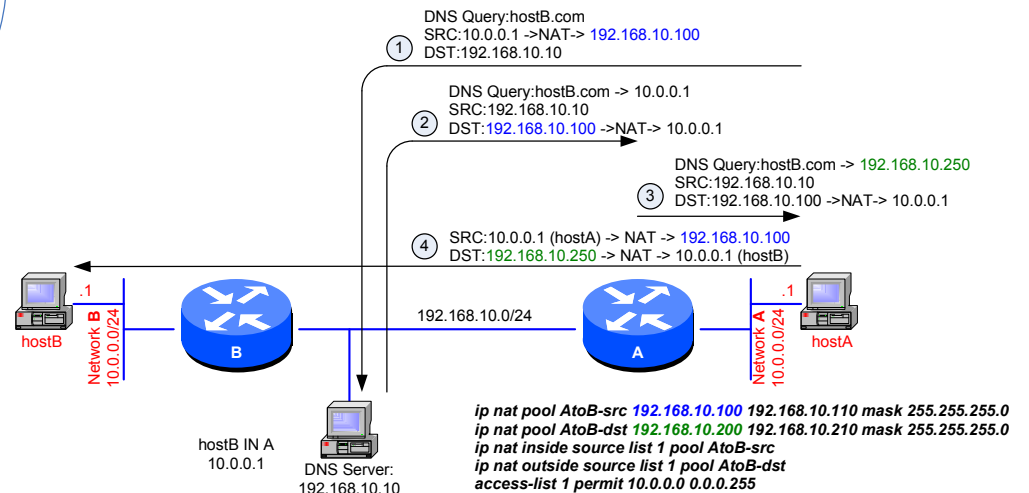
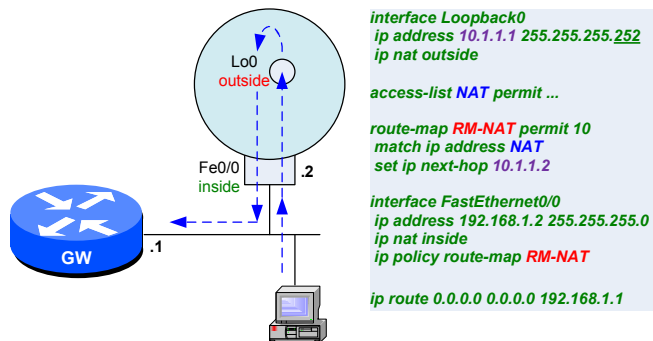
### Overlapping networks

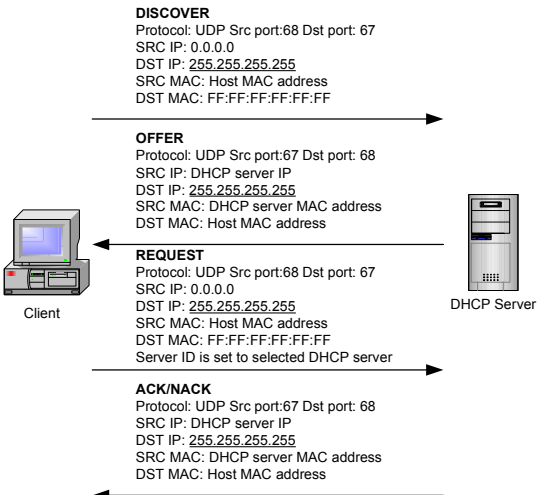
DNS can be used to allow overlapping networks to communicate. Returning reply from DNS server is translated (DNS payload information) with **ip nat outside source static**

If DNS is not used then static translation has to be used (**ip nat outside source static**), but it is more difficult to manage

Hosts must point a default GW to NAT router, not internet gateway

### NAT on a stick





Oper. Code	HW Type	HW Len	Hop count
Transaction ID (32b)			
Seconds (16b)		Flags (16b)	
Client IP Address (CIADDR) (32b)			
Your IP Address (YIADDR) (32b)			
Server IP Address (SIADDR) (32b)			
Gateway IP Address (GIADDR) (32b)			
Client HW Address (CHADDR) (16B)			
Server name (SNAME) (64B)			
Boot filename (128B)			
Vendor-specific options (64B)			

**service dhcp** (enabled by default)  
UDP/67 server; UDP/68 client; Payload is 300 bytes

Client has fixed UDP/68 port as reply is broadcasted to the segment and if random port was used other hosts would receive „unknown“ packets. Here, they know it is a BOOTP reply.

Server responding to client's Discover and Request messages also uses broadcast to inform other possible DHCP server on a LAN, that the request has been served

Address is assigned with lease time. Client can extend lease time dynamically sending DHCPREQUEST, usually at 50% of time. If server sends DHCPACK, lease is extended. If server sends DHCPNACK, client restarts the full lease. If no response is received, client uses an address until lease expires

Transaction ID (random) field is used to distinguish different queries. „Seconds“ field can be used by secondary server not to respond until this time expires and reply is not heard from primary server

When server replies, it places static arp entry in local cache for a client's MAC and assigned IP, so ARP request does not have to be generated, otherwise client could not respond to that ARP request as it doesn't know own IP yet (chicken and egg)

### Features

### Relay

**ip helper address <ip> [redundancy <HSRP name>]**  
Broadcast is changed to directed unicast with router's LAN interface's IP address as a source. This feature is used if DHCP server is not on the same segment as clients (broadcast is not propagated through a router). If redundancy is used, only active router will forward queries to the server

If a client is in local network *giaddr* in HDPC DISCOVER message is set to 0 (zero), and a pool is chosen from interface on which the message was received. If **ip helper address** is used, *giaddr* is set to forwarding router interface's IP, and a pool is chosen from this particular IP regardless of interface on which unicast request was received.

**(G) ip dhcp smart-relay**  
Relay agent attempts to forward the primary address as the gateway address three times. If no response is received then secondary addresses on relay agent's interface are used

### Proxy

When a dialing client requests an IP address via IPCP, the dialed router can request this IP on client's behalf from remote DHCP server, acting as a proxy. The dialed router uses own IP from PPP interface to set *giaddr* in the request

**interface <iif>**  
**ip address <ip> <mask>**  
**encapsulation ppp**  
**peer default ip address dhcp**  
**ip address-pool dhcp-proxy-client**  
**ip dhcp-server <ip>**

## DHCP

R2 PE:  
**interface <iif>**  
**encapsulation ppp**  
**ip address <ip> <mask>**  
**peer default ip address <peer-ip>**  
**ppp ipcp mask <mask>**  
**ppp ipcp dns <dns1> <dns2>**  
**no peer neighbor-route**

This feature is usefull when WAN links get's all IP information dynamically assigned, and DHCP options (DNS, domain, etc) need to be passed to clients behind a router.

### On-demand pool

**ip dhcp pool <name>**  
**import all**  
**origin ipcp**

R1 CPE:  
**interface <iif>**  
**encapsulation ppp**  
**ip address negotiated**  
**ppp ipcp netmask request**  
**ppp ipcp dns request**

### Client

**(IF) ip address dhcp**  
Assign IP address from DHCP

**(IF) ip dhcp client request ...**  
Request additional parameters (options)

**(IF) ip dhcp client lease <days> [<hours>]**  
Request specific lease time for an address

**(IF) ip address dhcp client-id <iif>**  
Specify Client-ID to identify specific profile on DHCP server

**release dhcp <iif>**  
Force interface to release and renew IP address

### Server

**ip dhcp pool <name>**  
**network <net> <mask>**  
**dns-server <ip>**  
**domain-name <name>**  
**lease <days> [<hours>]**  
**option <id> <type> <value>** (additional options – ex. 150 TFTP server, etc)  
**netbios-node-type <type>** (h-node: Hybrid node recommended)

**(G) ip dhcp exclude-address <start> <end>**  
Multiple lines defining which addresses in a network range will not be assigned to clients

**(G) ip dhcp database flash:/bindings [timeout <sec>] [write-delay <sec>]**  
Configure database agent for storing bindings, and conflict logging

**(G) no ip dhcp conflict-logging**  
Must be disabled if database agent is not configured (conflicts logging is possible if there is a place to store them)

**ip dhcp pool PC1**  
**host <ip> /24**  
**hardware-address <MAC>**  
Host pools inherit entire configuration from the main pool (IP is matched against network in the pool). When creating per-host pool, 01 must be added in the front of MAC defined as client-id (01 means ethernet media type). Ex. 0100.0c12.213e.23

**(G) ip dhcp ping {packets <#> | timeout <msec>}**  
DHCP server pings IP before it is leased

**(G) ip dhcp bootp ignore**  
Ignore BOOTP requests sent to this DHCP server

**show ip dhcp pool**  
**show ip dhcp binding**

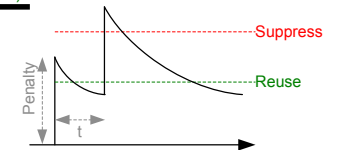
# MGMT part 1

## Accounting

- (IF) ip accounting output-packets**  
Only transit IP traffic is measured and only on an outbound basis
- (IF) ip accounting access-violation**  
Access-violation requires ACL to be applied on the interface. It cannot me a named ACL. Only process switched packets generate accurate statistics (fast switching or CEF do not)
- (G) ip accounting-threshold <threshold>**  
The default value is 512 source/destination pairs. This default results in a maximum of 12,928 bytes of memory usage
- (IF) ip accounting mac-address {input | output}**  
To display the MAC accounting information, use **show interface mac**
- (IF) ip accounting precedence {input | output}**  
To display IPP accounting, use **show interface precedence**
- (G) ip accounting-list <net> <mask>**  
Define hosts for which IP accounting information is kept
- (G) ip accounting-transits <count>**  
Define number of transit records (default is 0) stored in IP accounting database. Transit entries are those that do not match any of the filters specified by ip accounting-list. If no filters are defined, no transit entries are possible

## IP Event Dampening

- (IF) dampening [<half-life> <reuse> <suppress> <max> [restart]]**  
Reduce the effect of routing table instability. Mainly focused on IGP. Penatly is added (1000) every time interface flaps. Primary interface configuration is applied to all subinterfaces by default.
- Half-life:** Time, after which a penalty is decreased by half (default 5sec)
- Reuse:** When penalty decreases below this value, route is unsuppressed (default 1000)
- Suppress:** Suppress route when penalty is exceeded (default 2000)
- Max:** Maximum time a route can be suppressed (default 20 sec)
- Restart:** Penalty applied to interface when it comes up for the first time after reload (default 2000)



## Core dump / crashinfo

- (G) exception dump <ip>**  
Dump exception file to remote server
- (G) exception protocol {ftp | tftp}**  
If you use TFTP to dump the core file to a server, the router will only dump the first 16 MB of the core file. If FTP is used, **ip ftp username** and **ip ftp password** must be defined
- (G) exception core-file <name>**  
Specify the name of the core dump file
- (G) exception crashinfo file <device:filename>**  
Enable the creation of a diagnostic file at the time of unexpected system shutdown. The file name can be up to 38 characters. The filename will be **filename\_yyyyymmdd-hhmmss**
- (G) exception crashinfo buffersize <KB>**  
Change the size (default 32K) of the buffer used for crash info files
- (G) exception crashinfo dump command <cli>**  
Specify output to be written to the crashinfo file
- (G) exception crashinfo maximum files <#>**  
Define max number of crashinfo files. Old files are deleted automatically. If set to 0, all crashinfo files are deleted.

## KRON

- kron policy-list <policy-name>**  
**cli <command>**  
Define policy with commands to be executed. You CANNOT use configuration commands, only global exec
- kron occurrence <name> {in | at} <time> {oneshot | recurring | system-startup}**
- policy-list <policy-name>**  
There can be many policies assigned to the same schedule
- show kron schedule**

## TCLSH

- ```
foreach VAR {
  10.0.0.1
  10.0.0.2
} puts [exec „ping $VAR”]
```

# MGMT part 2

## Misc Services

- (G) service nagle**  
Buffer keystrokes and send them in one packet (useful for telnet)
- (G) no service prompt config**  
No prompt in config mode
- (G) warm-reboot [count <#> | uptime <min>]**  
When device is reloaded uncompresses IOS from DRAM is used, not compressed on Flash. Default count is 5, after which cold reboot is required. Default timeout is 5 min, after which next cold-reboot can be performed
- copy running-config startup-config [all]**  
If all is used, all default values, which are not shown in running config, are stored in startup config
- (G) ip options {drop | ignore}**  
Drop or ignore IP options packets that are sent to the router
- (IF) no ip unreachable**  
By default enabled. Affects all types of ICMP unreachable messages (traceroute, etc)
- (IF) no ip redirects**  
By default enabled. Enable sending of ICMP redirect messages if routing for destination points through the same interface on which packet was received
- (G) service timestamps {debug | log} {uptime | datetime [localtime | show-timezone] | msec | year}**  
Define timestamp for log and debug messages to either device uptime or real time (with timezone, milliseconds, etc)
- debug condition <condition>**  
Limit debugging output to specific condition. It is debug command independent – works for all debugs, as long as condition is met

## RMON

- The RMON engine on a router polls the SNMP MIB variables locally, no need to waste resources on SNMP queries
- When the value of the MIB variable crosses a raising threshold RMON creates a log entry and sends an SNMP trap. No more events are generated for that threshold until the opposite falling threshold is crossed
- (G) rmon alarm <number> <MIB OID> <interval> {delta | absolute} rising-threshold <value> [<event-number>] falling-threshold <value> [<event-number>] [owner <string>]**  
The MIB OID argument must be in the form entry.integer.instance
- (IF) rmon collection history <index> [buckets <number>] [interval <seconds>] [owner <name>]**  
Enable RMON history gathering on an interface
- (G) rmon event <number> [log] [trap <community>] [description <string>] [owner <string>]**  
Add an event (in RMON event table) that is associated with an RMON event number

## CPU threshold

- (G) process cpu threshold type {total | process | interrupt} rising <%> interval <sec> [falling <%> interval <sec>]**  
Interval defines duration of the CPU threshold violation that must be met to trigger a CPU thresholding notification. If falling threshold is not set it is the same as rising
- (G) process cpu statistics limit entry-percentage <%> [size <sec>]**  
Set the entry limit and size of CPU utilization statistics. Entry-percentage indicates the percentage of CPU utilization that a process must use to become part of the history table. Size is a duration of time (default 600 sec) which CPU statistics are stored in the history table
- (G) snmp-server enable traps cpu [threshold]**  
Enables CPU thresholding violation traps
- (G) snmp-server host <ip> traps <community> cpu**  
Sends CPU traps to the specified SNMP server

## DNS

- Authoritative server**
  - ip dns server**
  - ip dns primary <domain> soa <ns> <email> <timers ...>**
  - ip host <domain> ns <ip>**
  - ip host <domain> mx <priority> <ip>**
  - ip host <fqdn> <ip1> ... <ip6>**
  - show ip dns primary**
- Client**
  - ip domain {timeout <sec> | retry <#>}**
  - ip domain round-robin**
- Caching server**
  - ip dns server**  
Router acts as server, but forwards queries to authoritative server
  - ip name-server <ip>**
  - (G) ip domain lookup**  
Enable name lookup (enabled by default)
- Spoofing**
  - (G) ip dns spoofing [<ip>]**  
If upstream DNS server is up, router will proxy and forward queries. If upstream is down, router will respond to all queries with pre-configured IP only if query is not for router's own interface, if so, then it replies with interface IP on which query was received.
  - (G) ip domain lookup**
  - ip name-server <ip>**
  - (G) ip dns server**

## IP Traffic Export

- Export IP packets that are received on multiple, simultaneous WAN or LAN interfaces. It's like SPAN on switches
- ip traffic-export profile <profile-name>**
- interface <intf>** (outgoing interface)
- bidirectional** (By default, only incoming traffic is exported)
- mac-address <H.H.H>** (destination host which will receive exported traffic)
- incoming {access-list <acl>} | sample one-in-every <packet-#>}**
- outgoing {access-list <acl>} | sample one-in-every <packet-#>}**
- (IF) ip traffic-export apply <profile-name>**

# MGMT part 3

## IP SLA

- (G) ip sla responder**  
Control message asks Responder to open specific UDP or TCP port. After ACK is received, Sender sends a probe
- (G) ip sla <id>**  
Enable IP SLA. After type is defined, you cannot change it
- timeout <msec>**  
Amount of time IPSLA operation waits for a response. This value should be based on RTT
- frequency <sec>**  
Define a rate at which a IPSLA operation repeats
- threshold <msec>**  
Define threshold for calculating statistics (only). The value specified for this command must not exceed the value specified for the timeout command.
- request-data-size <bytes>**  
Set the request data size for the IP SLA operation
- ip sla monitor schedule <#> [life <sec>] [start-time {pending | now | <hh:mm> [<month> <day>]}] [ageout <sec>]**  
To stop a probe use **no ip sla monitor schedule <#>**
- show ip sla statistics [<id>]**

## RMON

- The RMON engine on a router polls the SNMP MIB variables locally, no need to waste resources on SNMP queries
- When the value of the MIB variable crosses a raising threshold RMON creates a log entry and sends an SNMP trap. No more events are generated for that threshold until the opposite falling threshold is crossed
- (G) rmon alarm <number> <MIB OID> <interval> {delta | absolute} rising-threshold <value> [-event-number] falling-threshold <value> [-event-number] [owner <string>]**  
The MIB OID argument must be in the form entry.instance
- (IF) rmon collection history <index> [buckets <number>] [interval <seconds>] [owner <name>]**  
Enable RMON history gathering on an interface
- (G) rmon event <number> [log] [trap <community>] [description <string>] [owner <string>]**  
Add an event (in RMON event table) that is associated with an RMON event number

## SNMPv3

- Extends security of SNMP with authentication and encryption
- (G) snmp-server view <name> <MIBs> {included | excluded}**
- (G) snmp-server group <name> v3 {auth | noauth | priv} [{read | write | notify} <view>] [access <acl>]**  
Define SNMP group policy for accessing specific MIBs (view). Auth (authNoPriv), noauth (noAuthNoPriv), and priv (authPriv) define if messages are authenticated and/or encrypted (privacy)
- (G) snmp-server user <name> <group> v3 [encrypted] [auth {sha | md5}] <password> [priv {des | 3des | aes} <password>] [access <acl>]**  
Define user, assigned to specific group. Define authentication and encryption methods. If **encrypted** is used, all passwords must be provided in encrypted form, not plain-text
- RFC does not allow storing SNMPv3 users/passwords in accessible configurations, so they are not shown in running config (stored in private NVRAM area). Users are not backed up with running-config, so you must store this information in some repository in case you need to restore configuration
- show snmp group**
- show snmp user**

## SNMPv2

- Community strings are passed as clear-text. ACLs and views should be used to protect from unauthorised SNMP access
- (G) snmp-server community <string> [-acl] [{ro | rw}] [view <name>]**  
Define community to access MIBs. ACL can be define to limit source hosts. View can be defined to limit MIBs available for querying
- (G) snmp-server enable traps <list>**  
Define list of traps (globally for all hosts)
- (G) snmp-server {location | contact} <string>**  
Define free text describing contact person, responsible for this device and location of this device
- (G) snmp-server system-shutdown**  
Allow device reload with SNMP write command
- (G) snmp-server ifindex persist**  
**(IF) snmp-server ifindex persist**  
Keep interfaces' indexes (or single interface index) after reload, so management systems do not have to re-learn indexes
- (G) snmp-server host <ip> [version {1 | 2c | 3} <community>] [<trap list>]**  
Define host, trap version and list of traps which will be sent to remote management system
- (G) snmp-server ip dscp <dscp>**  
Define DSCP used for SNMP packets
- (G) snmp-server trap-source <intf>**  
Define source interface for SNMP packets
- (G) snmp-server tftp-server-list <acl>**  
Define ACL with hosts allowed to receive config via TFTP when backup is initiated via SNMP
- (G) snmp-server view <name> <MIB list> {included | excluded}**  
Define list of accessible MIBs for specific view. It can be assigned to a community
- (IF) no snmp trap link-status**  
Disable traps for link up/down (especially for user interfaces)
- show snmp mib ifmib ifindex**
- show snmp {community | host}**
- show snmp view**

# Log & Archive

**archive**  
**log config**  
**hidekeys** (hide passwords, communities, etc when they are sent to syslog)  
**logging enable**  
**notify syslog** (send executed commands to syslog)

**show archive log config ...**

**path ...**  
 You can use \$t for current time and \$h for hostname

**maximum <#>**  
 Maximum configs to be archived (max 14)

**time-period <min>**  
 Snapshot config regularly every # of min

**write-memory**  
 Snapshot config when **write memory** (or **copy run start**) is executed

**(G) archive**

**archive config**  
 Backup configuration on request

**show archive config differences <config1> <config2>**  
 Displays differences in DIFF style

**show archive config incremental-diffs <config>**  
 Displays configuration made in IOS style

**Config backup**

**configure revert {now | timer {<minutes> | idle <minutes>}}**  
 Cancel timed rollback and trigger the rollback immediately (**now**) or change (extend) timers. Configuration Archive functionality must be enable first. Idle defines time for which to wait before rollback

**configure replace <target-url> [nolock] [list] [force] [ignorecase] [revert trigger <error> [timer <min>] | time <min>]**  
 Overwrite running-config with stored config. Classical copy startup to running merges both configs and overwrites only entries which can exist as single lines. **List** displays command lines applied. The **time** defines after how many minutes rollback will be performed if not confirmed. It is the same as **revert trigger timer**

**configure terminal revert timer <min>**  
 Configure from terminal and rollback after specified time if not confirmed. Rollback to last active config, unlike in **configure replace**, where file can be specified

**configure confirm**  
 Confirm configuration changes. It is used only if the **revert trigger** is used

## Archiving

## Logging

**(G) access-list <id> <... ...> log <tag>**  
 ACL entries can be marked with cookie (tag). Works for numbered and named ACLs. Logged messages will have that tag appended in square brackets [ <tag> ]

**event manager applet <name>**  
**event syslog patter <tag>**  
**action <id> <action>**  
 EEM applet can be created to match that tag from ACL

## ACL & Syslog & EEM corelation

## Syslog

**(G) logging on**  
 Enable logging (enabled by default) to destinations other than console. If logging is disabled, no messages will be sent to buffer or syslog. Messages will be sent only to console

**(G) logging console <level>**  
 It affects not only console, but also all TTY lines. If logging to console is disabled, logging to telnet session using **terminal monitor** will not work

**(G) logging buffered <size> <level>**  
 Messages are logges into local memory buffer. If max size is reached, old messages are overwritten (round-robin)

**(G) logging monitor <level>**  
 Define logging level for terminal lines. By default all messages are logged if **terminal monitor** is used

**(G) logging userinfo**  
 Generate log message when user enters privilege mode by executing **enable** or **disable** command. If privilege is automaticaly assigned to user (by AAA server or via line configuration), message is not shown

**(G) logging count**  
 Count all types of logging (per facility, message type, severity, etc) (**show logging count**)

**(G) logging rate-limit {<#> | console <#>} [except <severity>]**  
 Default limit is 10 messages per sec.

**(G) service sequence-numbers**  
 Sequence numbers are added in the front of messages

**(LINE) logging synchronous**  
 Refresh existing exec line if log message overwrites it (automatic Ctrl-R)

**(G) logging file flash:<path> <size> <level>**  
 Logging to flash is available only on switches

**(IF) logging event {link-status | subif-link-status [ignore-bulk]}**  
 Log physical or subinterface interface status changes. If **ignore-bulk** is used, subinterfaces do not generate logs if main interface is down

**show logging [count]**

Syslog messages are sent using UDP/514 (some servers and IOSes support TCP)

Every message contains: Facility, Severity, Hostname, Timestamp, Message

If timezone is sent then syslog message is marked with „\*” (asterisk)

**(G) logging host <ip> [transport {udp | tcp} port <port>] [session-id {hostname | ipv4 | ipv6 | string <string>}] [discriminator <name>]**  
 Logging to remote syslog server. All messages can be tagged with hostname, IP address or custom string. Filtering can be applied with discriminator

**(G) logging trap <severity>**  
 Specify severity level for logging to all hosts

**(G) logging facility <facility-type>**  
 Default facility is Local7 (Local4 for FW). Syslog server can send logs to specific file based on facility

**(G) logging discriminator <name> [[facility] [mnemonics] [msg-body] [drops <string> | includes <string>]] [severity {drops <sev> | includes <sev>}] [rate-limit <#>]**  
 Create a syslog message discriminator. It can be used to define filering for messages. It can be applied to syslog server to limit specific messages sent out. Console messages CANNOT be filtered

**(G) logging origin-id {hostname | ip | ipv6 | string <string>}**  
 Origin identifier is added to the beginning of all syslog messages

**(G) logging queue-limit <size> | trap <size>**  
 Default size is platform-depandent. Usualy 100 messages

**(G) logging source-interface <if>**  
 By default, interface, through which message is sent is used as source IP

**(G) snmp-server enable traps syslog**  
 Send syslog messages as SNMP traps

# Device Access

## Banners

**(G) banner {mtd | login | exec | incoming} % message %**  
 The % is just a sample delimiter (% is very rarely used inside banner text, so it is good choice)

mtd – message of the day display as a very first banner  
 login – banner shown just before login prompt, but after mtd  
 exec – shown after user is logged in  
 incoming – when reverse-telnet is executed to a device

SSH does not show mtd and login banners before login prompt. They are shown after user is logged in.  
 Dynamic tokens: \$(hostname), \$(domain), \$(line)

## Telnet

**(G) busy-message <hostname> <message>**  
 Displayed if telnet to that host is performed, and host is not reachable

**(G) ip telnet hidden {addresses | hostnames}**  
 Do not display IP address or hostname when telnetting to remote system. Both options can be defined separately

**(G) service telnet-zero-idle**  
 Router with idle session will advertise window=0 to remote device which will stop processing buffered data until session is resumed

**(G) service hide-telnet-address**  
 IP is not shown when it's resolved while telnetting to remote host. It is an alias command for **ip telnet hidden addresses** =<= this is what is shown in running config

**(G) ip telnet quiet**  
 Do not display any messages when telnet session is being established to remote system

**(G) ip telnet tos <hex tos>**  
 Define TOS value for telnet performed from the router. Default is 0xC0 (192) = CS6

**(G) service linenummer**  
 Display VTY line number when telnetting to that device  
 Break signal when using telnet: Ctrl + ]  
 Break signal when using AUX: Ctrl + Shift + 6, then B

**(G) hostname <name>**  
**(G) ip domain-name <name>**  
 Hostname (other than Router) and domain name is required to generate RSA key

**(G) crypto key zeroize rsa**  
 Delete the RSA key-pair. If new key is generated, old one is overwritten

**(G) crypto key generate rsa [modulus <bits>]**  
 If RSA key pair is generated then it automatically enables SSH. To use SSHv2 the key must be at least 768 bits

**(G) ip ssh {timeout <sec> | authentication-retries <#>}**  
 Default session negotiation timeout is 120 sec. and 3 retries

**(LINE) transport input ssh**  
 Limit access to VTY lines only via SSH

**(G) ip ssh version [1 | 2]**  
 Both SSH ver 1 and 2 are enabled by default. If any version is defined, only this version is supported

**(LINE) rotary <#>**  
**(G) ip ssh port <port> rotary <#>**  
 Connect the port with rotary group, which is associated with group of lines. Then you can ssh to specific VTY lines using non-standard port

**ssh [-v {1 | 2}] [-l <user>[:<#>]] [<ip>]**  
 By default local user will be used (the one which is currently logged in on a source device)

**(G) ip ssh source-interface <intf>**  
 Source interface for initiating ssh sessions

**(G) ip scp server enable**  
 Enables SCP server

**(G) ip ssh dscp <dscp>**  
 Define DSCP for SSH traffic initiated to or from the router

**(G) ip ssh break-string <string>**  
 Define Break control characters by prefixing them with ^V (Ctrl+V) or using the \xxx (hex) notation. Reverse telnet can be accomplished using SSH. For example control-B character is ASCII 2 (002)

## VTY & CON lines

**(LINE) session-timeout <min> [output]**  
 Define idle timeout for outbound sessions (to other device)

**(LINE) exec-timeout <min> [<sec>]**  
 Define inactivity timeout for inbound session

**(LINE) absolute-timeout <min>**  
 Define absolute session timeout (for in and out traffic is **output** is used)

**(LINE) refuse-message <text>**  
 Message displayed to remote device when line is busy

**(LINE) vacant-message <text>**  
 Message displayed, when line is vacant (console)

**(LINE) ip netmask-format {bit-count | decimal | hexadecimal}**  
 Define netmask format for all show commands

**(LINE) access-class <acl> {in | out} [vrf-also]**  
 Define ACL for limiting source addresses. If you have VRFs, from which you administer, add **vrf-also**

**(LINE) length <#>**  
 Define number of lines displayed. If you set to 0 (zero), no pausing is used

**(LINE) transport input {<list of protocols> | all}**  
 Define available protocols which can be used to access VTY remotely (default is **all**)

**(LINE) transport preferred {<protocol> | none}**  
 Default protocol used for outbound connection when only hostname is typed in exec prompt. Default is telnet. If you use **none**, misspelled commands do not cause outbound telnet

**(LINE) lockable**  
 Session can be locked by a user. To unlock, password is required (password is defined when **lock** command is executed)

**(LINE) no {mtd-banner | exec-banner}**  
 Disable banners on specific lines (ex. console)

**(LINE) logout-warning <sec>**  
 Display message before logging user out (ex. timing out an idle console). Disabled by default

## HTTP

**(G) ip http {server | secure-server}**  
 Enable HTTP (80) or HTTPS (443) server

**(G) ip http {port | secure-port} <port>**  
 Define non-default ports for HTTP or HTTPS

**(G) ip http authentication local**  
 By default enable secret is used to access web pages. Local users must be defined with privilege 15

**(G) ip http access-class <acl>**  
 Define networks from which web server is accessible

**(G) ip http max-connections <#>**  
 How many consecutive sessions can be established

**(G) ip http path <path>**  
 Set base path for web server (ex. for accessing IOS or other files from flash)

**(G) ip http secure-ciphersuit {3des-ede-cbc-sha | des-cbc-sha | rc4-128-md5 | rc4-128-sha}**  
 Define security algorithms for accessing secure web server

**(G) ip http client {username <user> | password <password>}**  
 Define username and password for accessing remote web pages (which require authentication)

**(G) ip http client source-interface <intf>**  
 Define source interface for HTTP and HTTPS traffic originated from router

**show ip http server all**

# NetFlow

## Features

Original version 1 is the default. Most common version is 5. Aggregation is possible in version 8 (11 schemas). All versions until 9 had fixed format, not compatible with each other. Flexible NetFlow is version 9

Traditional NetFlow exports 7 key fields: Source IP, Destination IP, Source Port, Destination Port, L3 Protocol, TOS Byte (DSCP), Input interface. Provides packet and byte count

```
show ip flow export
show ip cache [verbose] flow

ip flow-top-talkers
top <#>
sort by {packets | bytes}
match ...
```

## Version 5

- (IF) ip flow {ingress | egress}**  
NetFlow will capture flows entering or leaving the router, but NOT to the router or from the router itself – only transiting traffic. Ingress flow is applied before rate limiting and decryption, egress flow is applied after rate limiting and encryption
- ip flow-export version 5 [origin-as | peer-as | bgp-next-hop]**
- ip flow-export destination <ip> <udp-port>**
- ip flow-cache entries <#>**
- ip flow-export source <if>**
- ip flow-cache timeout inactive <sec>**  
How long inactive flow will remain in cache before expiration (default 15 sec)
- ip flow-cache timeout active <sec>**  
How long active flow will remain in cache before expiration (default 30 min)
- (G) ip flow-capture {fragment-offset | icmp | ip-id | mac-addresses | packet-length | ttl | vlan-id | nbar}**  
Capture values from Layer 2 or additional Layer 3 fields
- (G) ip flow-export interface-names**  
Sends both: ifIndex and ifName in option data record

## Version 8

- ip flow-aggregation cache <predefined-aggregation-type>**
- cache entries <#>**
- export destination <ip> <udp port>**
- mask destination minimum <bit mask>**
- mask source minimum <bit mask>**
- enabled**
- Version 8 aggregation cache. Flows are aggregated for entries in routing table. All bytes and packets are summarized. Number of flows is also exported.

## Version 9 Flexible NetFlow

Version 9 defines exporting process with new aggregations. Flexible Netflow is an extension

Flexible NetFlow uses two structures: Template FlowSet and Data FlowSet. Template is composed of Type and Length, sent periodically

- (G) ip flow-export template options export-stats**  
Enable sending export statistics (total flows and packets exported) as options data
- (G) ip flow-export template [options] timeout-rate <#>**  
Templates and options sent every # of minutes
- (G) ip flow-export template [options] refresh-rate <#>**  
Templates and options sent every # of packets

Two parameters: match and collect define what will be caught and included in the flow cache

- 1) Configure Template  
**(G) flow record <name>**
- 2) Configure Exporter  
**(G) flow exporter <name>**

```
transport udp <port>
export-protocol {netflow-v5 | netflow-v9}
destination <ip>
```

- 3) Configure Monitor  
**(G) flow monitor <name>**

- exporter <name>**
- record <name>**
- cache entries <#>**
- cache timeout {active | inactive | update} <sec>**
- cache type {normal | immediate | permanent}**  
Normal cache is like traditional, with active and inactive timers. Immediate accounts for single packet, good for real-time or DDoS detection – may result in large amount of data exported. Permanent does not expire flows from cache. Entire cache is periodically exported. When cache is full, new flows will not be monitored

- 4) Configure interface  
**(IF) ip flow monitor <name> {input | output}**

# EEM

**Features**

- Embedded Event Manager reacts to Event Detectors and performs policy defined by TCL Script or EEM Applet
- (G) **event manager applet <name> authorization bypass**  
Allow applet to run without AAA authorization (useful for debugging)

**EEM Policy**

- Setup environment variable (optional)  
(G) **event manager environment <variable> <value>**  
Variables can be set with CLI (no \$ is prepended to variable name. They can be access by actions using \$<name>)
- Register applet policy  
(G) **event manager applet <name>**  
Event trigger and actions are defined within applet's context
- Define event trigger  
**event <ED> <ED specific parameters>**  
Define event of set of events which trigger policy
- Define actions  
**action <seq> cli command ,..."**  
Define actions (ex. CLI commands – show or configuration)

**TCL Policy**

- Register user directories  
(G) **event manager directory user policy <path>**  
(G) **event manager directory user library <path>**  
Path can be local directory on Flash disk
- Write TCL policy offline and upload it (TFTP, FTP, etc)  
**copy tftp flash://eem**
- Enable auto update for TCL scripts (optional)  
(G) **event manager update user policy group „\*.tcl” repository <network path>**
- Setup environment variable (optional)  
(G) **event manager environment <variable> <value>**
- Register policy  
(G) **event manager policy <TCL script name> type user**

**Multi Event Correlation**

- event tag <id> <ED> <ED parameters>**  
Define up to 6 events with unique tags
- trigger occurs 1**
- correlate event <id1> or event <id2> ...**
- attribute tag <id1> occurs <#>**
- attribute tag <id2> occurs <#>**
- Correlation can be „and” and „or”

**Verify**

- show event manager environment**
- show event manager policy registered**
- show event manager directory user policy**
- show event manager history events**

## Event Detectors

Each ED has own set of variables, which are set when event is triggered. Variable names starting with underscore (\_) are reserved for Cisco global variables

- show event manager detector all detailed**  
Show TCL variables for registering events, along with all available variables
- event none**  
Define empty event, so applet can be started from CLI (for testing: **event manager run <policy>**)
- event syslog pattern „<regexp>” occurs <#>**  
Triggers when matches syslog messages with regular expression
- event snmp oid <numerical oid> get-type exact entry-op ge entry-val <val> pool-interval <sec>**  
Triggers when SNMP OID crosses defined threshold
- event interface name <if> parameter receive\_throttle entry-op ge entry-val <val> entry-val-is-increment true pool-intervale <sec>**  
Triggers when interface counters cross threshold. Supports 22 counters (input error, interface reset, transmit rate, etc)
- event timer cron cron-entry „<cron time pattern>”**  
**event timer watchdog time <sec>**  
Triggers on watchdog, count down, cron or absolute timer
- event snmp-notification oid <oid> oid-val <val> op eq src-ip-address <ip> direction incoming**  
Triggers when incoming or outgoing trap is intercepted
- event cli pattern „<regexp>” sync {yes | no}**  
Triggers synchronous or asynchronous events when CLI matching defined pattern is executed. Synchronous events hold CLI command and must return \$ \_exit\_status. If it is 1 then command is executed, if 0, command is dropped. Asynchronous events are executed independently, allowing CLI command to proceed
- event neighbor-discovery interface <if> cdp add**  
Triggers when CDP or LLDP message is detected.Interface can be .\* (all). Specific messages can be checked:  
**action 1 if \$ \_nd\_cdp\_platform eq „Cisco IP Phone”**
- event ipsla operation-id <#> reaction-type jitterAvg**  
Triggers when IPSLA test result crosses defined threshold:  
**action 1 if \$ \_ipsla\_measured\_threshold\_value > \$ \_ipsla\_threshold\_rising**

## Other actions

- action <id> set \$ \_exit\_status {0 | 1}**  
Return exit status after policy is executed
- action <id> puts {„<string>” | \$ \_cli\_result}**  
Displays text on terminal screen
- action <id> syslog msg „<text>”**  
Send message to syslog engine
- action 1 gets response**  
**action 2 if \$response eq yes goto 5**  
Interaction with user (must be run from CLI)
- action <id> foreach \_var \$ \_listvar**  
... <manipulate \$ \_var> ...  
**action <id> end**
- action <id> regexp „<regexp>” \$ \_var**  
**action <id> if \$ \_regexp\_result eq „1”**  
**action <id> ...**  
**action <id> else**  
**action <id> continue**  
**action <id> end**
- action <seq> mail server „\$ \_email\_server” to „\$ \_email\_to” from „\$ \_email\_from” subject „<subject>” body „\$ \_cli\_result”**  
Send email with output from CLI commands (variable \$ \_cli\_result). Email variables can be set with **event manager environment** option