

CISCO CCNA

STUDY NOTES

 **Study-ccna.com**

Contents

Introduction	6
Recommended Training Course	6
Recommended Practice Exams	7
Book 1	8
Part I – Introduction to Networking	8
Chapter 1 – Introduction to TCP/IP Networking	8
Chapter 2 – Fundamentals of Ethernet LANs	9
Chapter 3 – Fundamentals of WANs and IP Routing	10
Part II – Implementing Ethernet LANs	11
Chapter 4 – Using the Command-Line Interface	11
Chapter 5 – Analyzing Ethernet LAN Switching	12
Chapter 6 – Configuring Basic Switch Management	13
Shared password	13
Local usernames and passwords	13
SSH	14
Switch IP settings	14
Command line miscellanea	15
Chapter 7 – Configuring and Verifying Switch Interfaces	16
Part III – Implementing VLANs and STP	18
Chapter 8 – Implementing Ethernet Virtual LANs	18
Trunking	19
Voice VLAN	20
VLAN Trunking Protocol	21
Chapter 9 – Spanning Tree Protocol Concepts	23
Classic STP	23
The Spanning Tree Process	24
RSTP	29
Chapter 10 – RSTP and EtherChannel Configuration	32
Per-VLAN spanning tree	32
Configuring L2 EtherChannel	35
Troubleshooting EtherChannel	36
Part IV – IPv4 Addressing	38
Chapter 11 – Perspectives on IPv4 Subnetting	38
Chapter 12 – Analyzing Classful IPv4 Networks	38
Chapter 13 – Analyzing Subnet Masks	38
Chapter 14 – Analyzing Existing Subnets	39

Part V – IPv4 Routing	44
Chapter 15 – Operating Cisco Routers	44
Chapter 16 – Configuring IPv4 addresses and Static Routes	45
Static Routes	45
Route Selection – 1 st Consideration: Longest Prefix Match	46
Route Selection – 2 nd Consideration: Administrative Distance	47
Route Selection – 3 rd Consideration: Metric	48
Route Verification	49
Chapter 17 – IP Routing in the LAN	50
Router with separate interfaces	50
ROAS Router-On-A-Stick	52
Layer 3 switch	53
Layer 3 Etherchannel	55
Chapter 17 – IPv4 troubleshooting	56
Part VI – OSPF	57
Chapter 19 – Understanding OSPF concepts	57
Routing protocols	57
OSPF fundamentals	59
IPv6 OSPFv3	60
Multi-area OSPF design	61
Chapter 20 – Implementing OSPF	64
Chapter 21 – OSPF Network Types and Neighbors	68
Part VII – IPv6	70
Chapter 22 – Fundamentals of IP Version 6	70
Chapter 23 – IPv6 Addressing and Subnetting	70
Chapter 24 – Implementing IPv6 Addressing on routers	71
Static addresses	72
Dynamic addresses	72
Link-local address	73
Multicast addresses	73
Neighbour Discovery Protocol	74
Loopback and unspecified	74
Anycast	74
Chapter 25 – Implementing IPv6 Routing	75
Static routes	75
Neighbour Discovery Protocols	75
Part VIII – Wireless LANs	77

Chapter 26 – Fundamentals of Wireless Networks	77
Chapter 27 – Analysing Cisco Wireless Architectures	82
FlexConnect and FlexConnect ACLs	85
Chapter 28 – Securing Wireless Networks	86
Chapter 29 – Building a Wireless LAN	88
WLC CLI interface	89
Book 2	90
Part I – Access Control Lists	90
Chapter 1 – Introduction to TCP/IP Transport and Applications	90
Chapter 2 – Basic IPv4 Access Control Lists	91
Chapter 3 – Advanced IPv4 Access Control	95
Part II – Security Services	96
Chapter 4 – Security Architectures	96
AAA – Authentication, Authorisation, Accounting	96
Chapter 5 – Securing Network Devices	97
Local passwords	97
Firewalls and Intrusion Prevention Systems	98
Chapter 6 – Implementing Switch Port Security	102
Chapter 7 – Implementing DHCP	104
Setting up a DHCP server	106
Chapter 8 – DHCP Snooping and ARP Inspection	106
DHCP Snooping	106
Dynamic ARP Inspection (DAI)	108
Part III – IP Services	110
Chapter 9 – Device Management Protocols	110
Syslog	110
Network Time Protocol (NTP)	111
CDP and LLDP	113
Chapter 10 – Network Address Translation	115
Chapter 11 – Quality of Service (QoS)	119
Classification and marking	119
Queueing	122
Shaping and policing	124
Congestion avoidance	126
Wireless QoS levels	126
Chapter 12 – Miscellaneous IP Services	127
First Hop Redundancy Protocols	127

Simple Network Management Protocol (SNMP)	129
FTP, FTPS, TFTP, SFTP and IOS filesystem	131
Part IV Network Architectures	133
Chapter 13 – LAN architectures	133
Power over Ethernet (PoE)	135
Chapter 14 – WAN architecture	137
Metro Ethernet (MetroE) Layer 2 VPNs (Virtual Private Networks)	137
Multiprotocol Label Switching (MPLS) Layer 3 VPNs	138
Internet VPNs	139
Chapter 15 Cloud Architecture	139
Part V Network Automation	142
Chapter 16 – Introduction to controller-based networking	142
Open SDN, OpenFlow, OpenDaylight	143
Cisco Application Centric Infrastructure (ACI)	144
Cisco APIC Enterprise Model	144
Chapter 17 – Cisco Software-Defined Access	144
Chapter 18 – Understanding REST and JSON	146
Chapter 19 – Understanding Ansible, Puppet and Chef	147
Configuration management tools	147
Topics beyond OCG scope	149
Enhanced Interior Gateway Routing Protocol	149
RIP Routing Information Protocol	151
IOS image	151

Disclaimer and Copyright

The information contained in this guide is for informational purposes only. Any advice that I give is my opinion based on my own experience.

This guide contains affiliate links for products which will help you pass the CCNA exam. I can earn commissions on their sales at no additional cost to you. I only recommend products which I truly believe are the best in their category and will help your career.

Any Third Party materials in this guide comprise of the products and opinions expressed by their owners. I do not assume responsibility or liability for any Third Party material or opinions. You are responsible for complying with any legal requirements such as licensing of Third Party software.

You may share the link on study-ccna.com to download this guide but no part shall be directly reproduced, transmitted, or sold in whole or in part in any form, without the prior written consent of study-ccna.com. All trademarks and registered trademarks appearing in this guide are the property of their respective owners.

Introduction

CCNA Study Notes by [Andrzej Mendel-Nykorowycz](#)

Thanks for downloading these CCNA 200-301 Study Notes! We hope you find them useful on your path to passing the CCNA.

The exam topics are laid out in the same order as in Wendell Odom's excellent OCG Official Cert Guide books [Volume 1](#) and [Volume 2](#) available from Cisco Press. It's not required to own the OCG books to use these notes though, they'll work fine with whatever study material you're using.

The notes are designed to help with your revision for the exam after your main study – we recommend that you use a comprehensive training course to gain a full understanding of the topics on the CCNA exam first, and then use these notes along with practice exams for your final preparation. That's the proven method to pass the exam.

Recommended Training Course

We recommend Neil Anderson's [Cisco CCNA Gold Bootcamp](#) course as your main study material. The videos explain all the exam topics in an easy to understand way, and the lab exercises give you loads of hands on experience to get ready for a real world job as well as pass the exam. It's the highest rated Cisco course online with an average rating of 4.8 from over 30,000 public reviews and is *the* gold standard in CCNA training.



Recommended Practice Exams

Practice tests give you the best chance of passing the exam on your first try. You can have all the knowledge but still fail if you're not used to the type of questions you'll get on the exam.

[Boson's ExSim-Max](#) is universally recognized as the best set of practice exams for the CCNA. The questions exactly mimic the difficulty and style of the real exam, and include in-depth explanations for each answer. You'll understand where you went right or wrong and know when you're ready for the real exam.



Book 1

Part I – Introduction to Networking

Chapter 1 – Introduction to TCP/IP Networking

OSI model	4 Layer RFC 1122 TCP/IP model	5 Layer TCP/IP model	Protocol data unit (PDU)	Example	Equipment operating on given layer	Domain
Application	Application	Application		HTTP		
Presentation						
Session						
Transport	Transport	Transport	Segment	TCP, UDP	Router w/NAT	
Network	Internet	Network	Packet	IP	Router	Subnet
Data-link	Network Access	Data-link	Frame	Ethernet, 802.11	Switch	Broadcast domain
Physical		Physical	Bit/symbol	UTP, single- mode fiber, multimode fiber	Hub	Collision domain

The five layer TCP/IP model splits the four layer model's Network Access layer into separate Physical and Data-Link layers.

The CCNA 200-301 exam uses the RFC 1122 four layer TCP/IP model.

Chapter 2 – Fundamentals of Ethernet LANs

Physical layer standard	10BASE-S	10BASE-LX	10BASE-LR	10BASE-E	10BASE-T	100BASE-T	1000BASE-T
Max speed	10Gb/s	10Gb/s	10Gb/s	10Gb/s	10Mb/s	100Mb/s	1Gb/s
Max range	300m	400m	10km	30km	100m	100m	100m
Medium	Multimode fibre	Multimode fibre	Singlemode fibre	Singlemode fibre	UTP Cat 3	UTP Cat 5	UTP Cat 5e

MAC address: OU:IO:UI:VE:ND:OR

OUI – Organisationally Unique Identifier – ID assigned to a particular manufacturer

VENDOR – managed and assigned by the manufacturer

CSMA/CD (Carrier sense multiple access with collision detection) – if the sender detects a collision, it sends a jamming signal. All nodes cease transmitting and resume it after a random time.

Wires used in 10BASE-T and 100BASE-T: 1,2,3,6.

Crossover cable: 1→3, 2→6, 3→1, 6→2

Wires used in 1000BASE-T: 1-8.

Ethernet II frame:

Field	Length in bytes	Description
Preamble	7	Synchronisation
SFD – start frame delimiter AKA SOF – start of frame	1	Signals the next byte begins the destination MAC field
Destination MAC	6	
Sender MAC	6	
Type	2	Defines the type of packet inside the frame. Can be replaced by a 4-byte 801.1Q header.
Data	46-1500	Actual data. Padded to 46 bytes if shorter
FCS – frame check sequence	4	Checksum. Receiver discards frame if incorrect

Multicast MAC addresses: 01:00:5E:00:00:00-01:00:5E:7F:FF:FF

The last 23 bits of a multicast MAC address are used to map the relevant multicast IP address. Since multicast addresses have 28 variable bits, this is a one-to-many mapping, i.e. one MAC address is used for several (32 to be precise) IP addresses.

Chapter 3 – Fundamentals of WANs and IP Routing

High-Level Data Link Control (HDLC) – Point-to-point layer 2 protocol

HDLC field	Ethernet equivalent	Notes
Flag	Preamble + SFD	
Address	Destination address	De facto unused
N/A	Sender address	
Control	N/A	Mostly legacy
Type	Type	Cisco proprietary in HDLC
Data	Data	
FCS	FCS	

Part II – Implementing Ethernet LANs

Chapter 4 – Using the Command-Line Interface

Simple password on console:

- (config)# config line console 0
- (config-line)# password *pass* – create console password and save it cleartext

Enable password:

- (config)# enable secret *pass*

Save running config to NVRAM:

- # write (old style command) **OR**
- # copy running-config startup-config (use this as the up-to-date command)

Erase NVRAM (factory reset)

- # write erase
- # erase startup-config
- # erase nvram

Chapter 5 – Analyzing Ethernet LAN Switching

Commands:

- # show mac address-table dynamic – show all learned MAC addresses
- # show mac address-table dynamic address AAAA.BBBB.CCCC – show source port for a given MAC address (note format used to display MAC! Three dot-separated hex quartets, not the 6 colon-separated hex pairs typically shown in other operating systems)
- # show mac address-table dynamic interface Gi0/0 – show all MAC addresses learned on a particular switch port
- # show mac address-table dynamic vlan 1 – show all MAC addresses learned in VLAN 1
- # show mac address-table aging-time [vlan n] – show the global (or VLAN specific) ageing timer setting
- # show mac address-table count – show used and available space in MAC address table
- # show interfaces [*ifname*] status – shows status (connected/notconnect), VLAN, duplex, speed and type of all interfaces. If *ifname* is given, limit output to this interface.
- # show interfaces Gi0/0 – detailed status of Gi0/0 interface

Chapter 6 – Configuring Basic Switch Management

“line vty 0 15” – VTY virtual terminal lines are used for both SSH and telnet access to the CLI

Shared password

Set per-line passwords with no username:

- (config)# line console 0/line vty 0 15 – set configuration context to console/VTY
- (config-line)# login – enable password security
- (config-line)# password **pass** – set password

Enable mode password:

- (config)# enable secret **pass**

Local usernames and passwords

Set all lines to use local usernames and passwords

- (config)# line console 0/line vty 0 15 – set configuration context to console/VTY
- (config-line)# login local – enable password security
- (config-line)# no password – remove existing shared passwords

Define username/password pairs

- (config)# – username *user* secret *pass*

SSH

Set hostname and domain name (needed for FQDN)

- (config)# hostname *hostname*
- (config)# ip domain-name *example.com*

Generate RSA keys - this enables SSH

- (config)# crypto key generate rsa [modulus *len*] – use modulus parameter to set (large) key size

Set local usernames – see above

Restrict SSH to version 2:

- (config)# ip ssh version 2

Enable SSH, disable telnet:

- (config-line)# transport input ssh – from “line vty 0 15” configuration context

Switch IP settings

Switches use layer 2 Switch Virtual Interfaces (SVI) instead of layer 3 port interfaces (unless specifically configured on a Layer 3 switch). SVI interfaces are associated with a VLAN.

A layer 2 switch can have only 1 IP address, to be used for administrative access. A layer 3 switch can have multiple IP addresses (on its SVI or physical interfaces) and route packets between them.

Set IP and default gateway

- (config)# interface vlan *n* – enter interface context for SVI for VLAN *n*
- (config-if)# ip address *address netmask* – set static address **OR**
- (config-if)# ip address dhcp – use DHCP to obtain address
- (config-if)# no shutdown – enable the interface
- (config)# ip default-gateway *address* – set default gateway
- (config)# ip name server *server address* [*secondary-address*] - set one or more DNS servers

```
Sw1(config)# interface vlan 1
```

```
Sw1(config-if)# ip address 10.10.10.10 255.255.255.0
```


Show IP status:

- #show dhcp lease – show DHCP leases on all SVIs
- #show interfaces vlan 1 – details for VLAN 1 SVI
- #show ip default-gateway – show default gateway

Configure multiple interfaces:

- (config)# configure Gi0/0 - 24 – configure a range of interfaces at once

Command line miscellanea

- #show history – show exec command history
- #terminal history size *x* – set history size to *x* commands for this session only
- (config)# history size *x* – set default history size to *x* commands
- (config)# no logging console – don't display syslog messages on console
- (config)# logging console – display syslog messages on console
- (config-line)# logging synchronous – synchronise syslog messages with other output
- (config-line)# exec-timeout *minutes seconds* – log out users after set time of inactivity (5 minutes default, 0 means do not time out)
- (config)# no ip domain-lookup – do not resolve what looks to be a domain name in the command line

Chapter 7 – Configuring and Verifying Switch Interfaces

Set duplex and speed settings:

- (config-if)# speed {10 | 100 | 1000 | auto}
- (config-if)# duplex {full | half | auto}

Auto-negotiation is disabled if **both** speed and duplex settings are set manually.

If the other side has auto-negotiation disabled, IEEE standard requires using the slowest supported speed and, if the speed is 10 or 100, half-duplex (full duplex otherwise). Since the slowest supported speed will typically be 10, the result is 10 Mbps half-duplex. Cisco instead senses speed and uses full duplex at 1000 Mbps and higher.

Both sides of a link should have the same configuration to avoid interface errors.

LAN switch interface status:

Line status (Layer 1)	Protocol status (Layer 2)	<i>show interfaces status</i>	Description
Administratively down	down	disabled	Interface configured with shutdown
down	down	notconnect	Cable not connected or faulty
down	Down (err-disabled)	err-disabled	Port security, DHCP Snooping, Dynamic ARP Inspection or other security mechanism has disabled the interface
up	up	Connected	Interface is working

Ethernet input errors:

- Runts – frames smaller than 64 bytes, can be caused by collisions
- Giants – frames larger than the maximum frame size, i.e. 1518 bytes. Note: in some situations (802.1Q-in-802.1Q, MPLS) the frame can be larger than 1518 bytes but smaller than 1600 – these are valid frames and are called **baby giants**. Additionally, some installations deliberately use larger maximum frame size, up to 9000 bytes, these are called **jumbo frames**.
- CRC Cyclical Redundancy Check – Frames that did not pass FCS Frame Check Sequence, can be caused by collisions.

Ethernet output errors:

- Collisions – all collisions that happened when transmitting a frame
- Late collisions – collisions that happened after 64th byte of a frame – sign of duplex mismatch (in half-duplex mode, collisions should be detected before the 64th byte.)

Part III – Implementing VLANs and STP

Chapter 8 – Implementing Ethernet Virtual LANs

Usable VLAN IDs: 1-1001 (standard), 1006-4094 (extended). Extended range is enabled only on newer Cisco devices.

Creating VLAN and assigning interfaces:

- (config)# vlan *n* – Create a VLAN with ID *n* and enter its configuration context
- (config-vlan)# name *name* – Assign a human readable name to a give VLAN
- (config-if)# switchport access vlan *n* – associate the port with VLAN *n*
- (config-if)# switchport mode access – force the port to access mode (i.e. do not autonegotiate and do not accept trunking)

```
Sw1(config)# vlan 10
```

```
Sw1(config-vlan)# name Sales
```

```
Sw1(config)#Interface GigabitEthernet0/2
```

```
Sw1(config-if)# switchport mode access
```

```
Sw1(config-if)# switchport access vlan 10
```

Show VLAN config:

- # show vlan brief – show all enabled VLANs and a list of interfaces assigned to each
- # show vlan – as above + brief information on spanning tree protocols

Disable VLAN:

- (config-vlan)# shutdown – disable VLAN **OR**
- (config)# shutdown vlan *n*

Trunking

There are two trunking protocols: 802.1Q (IEEE standard) and ISL (Inter-Switch Link, Cisco proprietary and deprecated). You can set the protocol with:

- (config-if)# switchport trunk encapsulation {dot1q | isl | negotiate}

Dot1q is the only supported option on newer switch models, where entering the command will give an error message because it is non-configurable.

Controlling whether a port is trunking:

- (config-if)# switchport mode access – never trunk
- (config-if)# switchport mode trunk – force trunking
- (config-if)# switchport mode dynamic auto – default mode on newer switches. Trunk if the other side is set to **trunk** or to **dynamic desirable**, access otherwise. Result will be access if both sides are set to switchport mode dynamic auto.
- (config-if)# switchport mode dynamic desirable – default mode on older switches. Initiate trunking with the other side, trunk if the other side is set to **dynamic auto**, **dynamic desirable** or **trunk**, access otherwise. Result will be trunk if both sides are set to switchport mode dynamic desirable.

Dynamic trunk negotiation is done using Dynamic Trunking Protocol (DTP). DTP frames travel over the native VLAN (i.e. untagged), and native VLAN settings must be identical on both devices. Link will fail if one interface is set to **trunk** and the other to **access**. One caveat: if VTP is enabled (even in transparent mode), VTP domain on both switches must match for DTP to work.

Set allowed VLANs:

- (config-if)# switchport trunk allowed vlan *range* (can be specified multiple times)

```
Sw1(config)#Interface GigabitEthernet0/1
```

```
Sw1(config-if)# switchport trunk encapsulation dot1q
```

```
Sw1(config-if)# switchport mode trunk
```

```
Sw1(config-if)# switchport trunk allowed vlan 10,20
```

Review config

- # show interfaces *ifname* switchport – show settings of a given port
- # show interfaces trunk – show all trunking ports, including information on VLANs allowed on a given port and VLANs pruned by STP

Voice VLAN

Cisco IP phones can have a PC plugged directly in to the back of them. The phone is then plugged in to a network switch. This allows the IP phone and PC on a desk to both be connected into the same single network switch port.

The voice traffic from the phone is placed in a dedicated Voice VLAN, and the data traffic from the attached PC is placed in an Access VLAN.

The phones have a built-in internal switch that trunks the voice VLAN traffic from the phone and the access VLAN traffic from the attached PC towards the rest of the network. The switch port which the IP phone is plugged into is configured as an access port but actually functions as a trunk for the access VLAN and voice VLAN.

Recommended configuration:

- (config-if)# switchport mode access
- (config-if)# switchport access vlan *x*
- (config-if)# switchport voice vlan *y*

```
Sw1(config)# vlan 10
Sw1(config-vlan)# name Sales
Sw1(config)# vlan 20
Sw1(config-vlan)# name Phones
Sw1(config)# Interface GigabitEthernet0/3
Sw1(config-if)# switchport mode access
Sw1(config-if)# switchport access vlan 10
Sw1(config-if)# switchport voice vlan 20
```


VLAN Trunking Protocol

VTP (VLAN Trunking Protocol) synchronises VLAN information between the switches on a campus. It allows a switch to advertise VLAN info (IDs enabled and names) and other switches to receive it and update their configuration accordingly. VLAN info can be removed as well as added or updated. It is recommended to disable VTP, as it can misconfigure or remove VLAN config in the whole network if an error is made:

- (config)# vtp mode transparent **OR**
- (config)# vtp mode off

However, VTP configuration does appear on the exam, so it's best to describe the protocol and its configuration. There are 3, mutually incompatible VTP versions: VTPv1, VTPv2, VTPv3. VTPv3, in particular, addresses many of the protocol's shortcomings and makes it much less dangerous to use. VTP works on trunk links only.

VTP can operate in three modes:

- Server (default)– creates, modifies and deletes VLANs, stores VLAN information locally in NVRAM, originates and forwards VTP advertisements and synchronises VTP information.
- Client – Listens for VTP advertisements and also forwards them. A client's VLAN database is completely dependent on VTP – a client cannot add, modify or delete VLANs locally – and is not stored in NVRAM.
- Transparent – Does not participate in VTP, i.e. does not originate VTP advertisements nor does it use them to populate its VLAN database, but it will forward VTP advertisements. So if a VTP Server and Client are on either side of a VTP Transparent switch it will forward VTP packets between them.

In VTPv1, a transparent switch will only forward VTPv1 advertisements in its own domain or if its own domain is empty. In VTPv2, a transparent switch will forward VTP advertisements regardless of domain.

A transparent switch maintains its own VLAN database and stores it in NVRAM.

Transparent is the recommended mode. Manually add the same VLAN information on all switches to configure layer 2 connectivity throughout the campus network.

To set mode:

- (config)# vtp mode {server | client | transparent | off}

To set domain:

- (config)# vtp domain *vtp_domain*

To set version:

- (config)# vtp version *n*

Check VTP status:

- # show vtp status

If there are several VTP servers on the network, the one with the highest configuration revision number will originate VTP advertisements. In theory, this allows decentralised configuration: switch A makes changes, increments its configuration revision number and propagates new configuration to other switches. Switch B synchronises its VLAN information and configuration revision number when it receives the information. Switch B can then do the same and propagate its changes, with incremented configuration revision to other switches, including switch A.

In practice, however, this may lead to a VTP bomb – if an external switch with a higher configuration revision number is attached to a network (this can happen if a switch from another office is repurposed for example), it will overwrite VLAN configuration on all switches and destroy connectivity throughout the campus.

VTPv3 solves that by making one VTP server primary. The Primary VTP server announces its status to others, which precludes them from making configuration changes. In effect, non-primary VTP servers act as VTP clients for all intents and purposes, with the only difference between VTP non-primary servers and VTP clients being that VTP clients are explicitly precluded from becoming primary. Primary mode can be password protected – if another switch wants to become primary, it will have to input the password.

- # vtp primary – note this is in EXEC mode
- (config)# vtp password *password* [hidden] – specify VTP password. Encrypt password if **hidden** argument is specified.

Additionally, VTPv3 supports extended VLANs (1006-4094), while VTPv1 and VTPv2 support only standard VLANs. Furthermore, VTPv3 propagates MST Multiple Spanning Tree instance information.

Chapter 9 – Spanning Tree Protocol Concepts

NOTE: “switch” and “bridge” used interchangeably as far as STP is concerned.

Classic STP

Routers do not forward broadcast traffic between their interfaces by default, and they use routing protocol loop prevention mechanisms and the IP header’s TTL Time to Live field to prevent traffic looping at Layer 3.

When switches receive broadcast traffic however, they forward it out all ports apart from the one it was received on. This causes the potential for ‘broadcast storms’ if a physical loop is present in the layer 2 campus. There is no TTL value in a layer 2 header, so if a loop exists then broadcast frames will circle round it endlessly. The network will quickly become unusable as an increasing number of looping frames eat up all the available switch CPU and bandwidth.

Spanning Tree exists to prevent this. STP (and its descendants) was standardised in 802.1D, now included in 802.1Q. Switches exchange BPDU Bridge Protocol Data Units to detect loops and then block switch ports to close them. This has the negative effect of reducing the number of active paths and bandwidth available in the campus, but is necessary to avoid broadcast storms.

If the topology changes (e.g. a cable is disconnected), the switches converge on a new set of links to use.

Port can be in two permanent states:

- Forwarding
- Blocking (named Discarding in RSTP) – do not send or receive any frames on this interface (apart from receiving BPDUs).

There are also two transient states:

- Listening (not in RSTP) – Do not forward frames, listen to incoming frames to remove stale MAC entries that might cause loops
- Learning – Do not forward frames, listen to incoming frames to learn MAC addresses

When switch ports first come online they are in the Blocking state. If there is no loop the port will transition to Forwarding.

Ports that are administratively disabled or that are failed are said to be in a Disabled state.

The Spanning Tree Process

A Root Bridge is elected and all switches transition the links in their best path to this switch to the forwarding state. Alternative paths which would cause a loop are blocked.

Spanning Tree is enabled by default on switches and they will automatically elect a Root Bridge and go through the loop blocking process. Because a forwarding path is always selected towards the Root Bridge, it is best practice to manually select it as a higher performance switch in a central location in the campus rather than leaving it to chance.

BPDUs contain the sending switch's 8-byte Bridge ID (BID) which uniquely identifies the switch on the LAN. The BID consists of 2-byte Bridge Priority field (which can be configured) and 6-byte System ID (based on switch's MAC address).

The Bridge Priority can be from 0 – 65535, with 32768 being the default.

Root bridge selection: lowest priority wins. If there is a draw, lowest MAC address wins.

When they start up, all switches send BPDUs with their BIDs as root. When they see a better BID, they start announcing it as the new root BID. Over time, every bridge will converge on the best root BID.

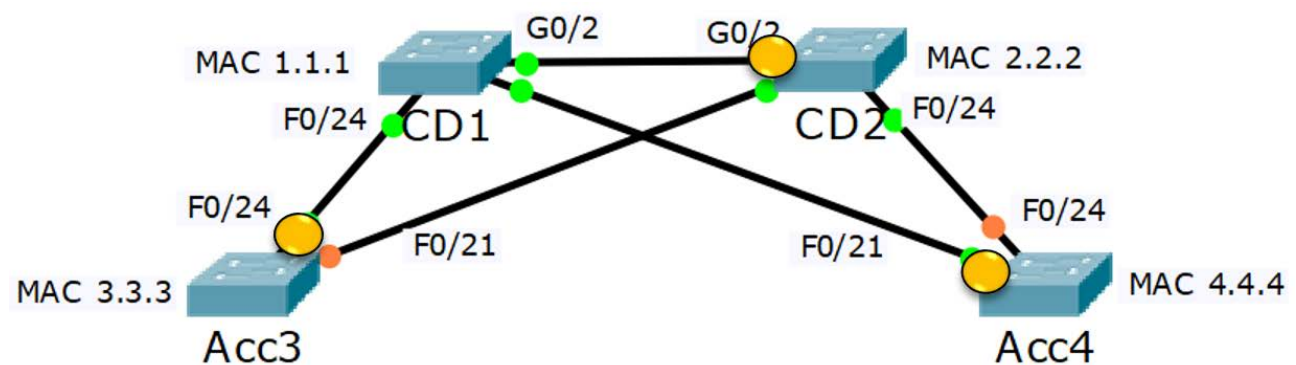
Each switch sends a Hello BPDU with

- Root bridge ID – ID of a bridge the sender believes to be root
- Sender bridge ID – ID of the sender itself
- Sender root cost – the cost of path from sender to root
- Timer values – set by Root Bridge, adopted and forwarded by other bridges
 - Hello – intervals between hello message, defaults to 2 seconds
 - MaxAge – How long a switch should wait when not receiving Hellos BPDUs before trying to change topology, defaults to 10 times Hello (20 seconds)
 - Forwarding delay – When changing topology, how long a switch should stay in each transient state: listening and learning, in that sequence. Defaults to 15 seconds for each state.

In the example below the Bridge Priority has not been manually set on any switches. CD1 is elected as the Root Bridge because it has the lowest MAC address.

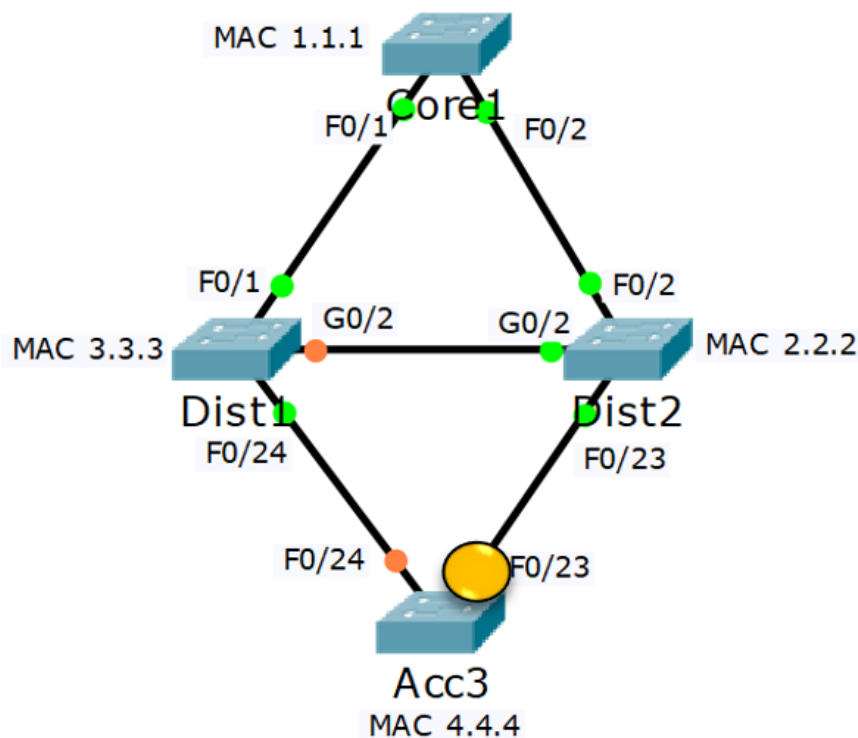
The other switches detect their lowest cost path to the Root Bridge (higher bandwidth links are preferred by default but Port Priority can be manually set). These paths transition to a forwarding state.

Each switch's exit interface on the lowest cost path to the Root Bridge is selected as its Root Port

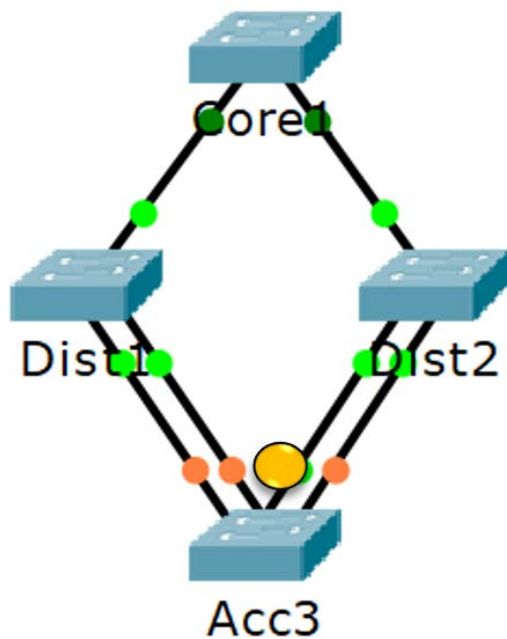



A Spanning Tree instance does not do load balancing. If a switch has multiple equal cost paths towards the Root Bridge, it will select the neighbour switch with the lowest Bridge ID.

In the example below, Acc3 selects the path to the Core1 Root Bridge via Dist2 as it has a lower Bridge ID. It does not also select the link to Dist1 as that would potentially form a loop.

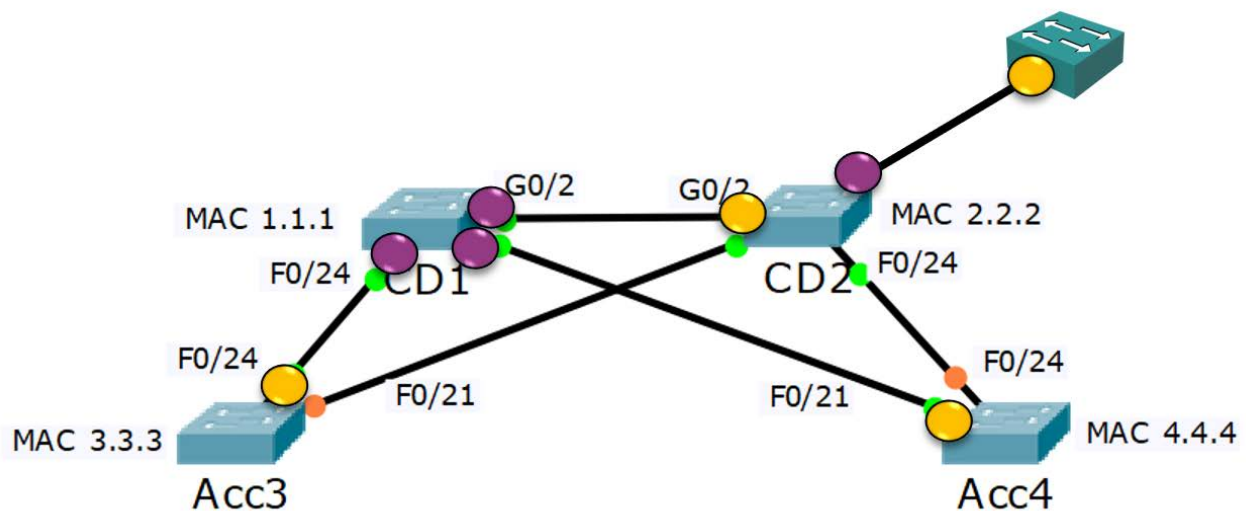


In the example below, Acc3 selects the path to the Core1 Root Bridge via the left hand link to Dist1 as it is the port with the lowest Port ID going to the lowest Bridge ID. It does not also select any of the other 3 links as that would potentially form a loop back to itself from the upstream switch.



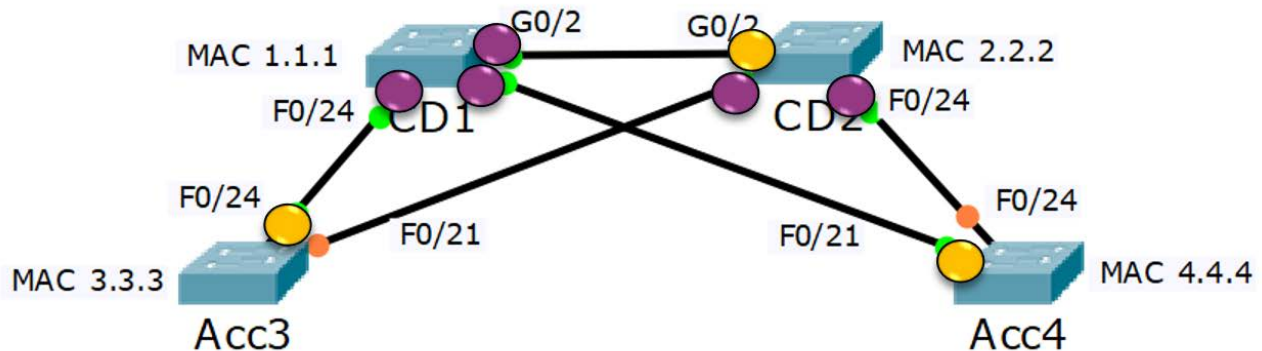
Ports on the neighbour switch opposite the Root Port are Designated Ports 

Root Ports point towards the Root Bridge, Designated Ports point away from it. All ports on the Root Bridge are always Designated Ports.

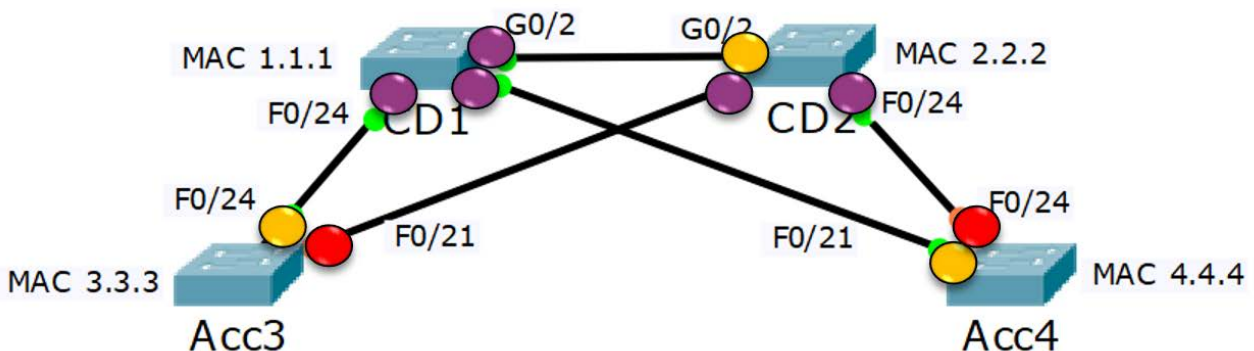


Root Ports and Designated Ports are the most direct paths to and from the Root Bridge and transition to a forwarding state.

On the remaining links, the switches determine which of them has the least-cost path to the Root. If they have equal cost paths then the Bridge ID is used as a tiebreaker. The port connecting this switch to the link is selected as a Designated Port.



Any ports which have not been selected as a Root Port or Designated Port pair would potentially form a loop. These are selected as Blocking Ports 



Spanning Tree only blocks ports on one side of the blocked link. BPDUs continue to be sent over the link but other traffic is dropped.

The easy way to figure out which ports are Root, Designated and Blocking:

1. Determine the Root Bridge first (best Bridge ID)
2. All ports on the Root Bridge are Designated Ports
3. Determine the Root Ports on the other switches (lowest cost to Root Bridge)
4. The ports on the other side of those links are Designated Ports
5. On the links which are left, one port will be Blocking
6. Determine the Blocking Port (highest cost path to Root Bridge or highest Bridge ID)
7. The ports on the other side of those links are Designated Ports

Convergence – the process by which switches react to changes in topology and agree on how to change how frames are forwarded. Happens when a switch stops receiving Hello BPDUs on its root port for a period determined by MaxAge timer or when the link on root port fails.

Port roles vs port states:

1. port roles (root port, designated port) relate to how STP views the network's topology;
2. port states (forwarding, blocking, listening, learning) determine whether a switch receives and sends frames on a given port.

When convergence happens, switches decide new port roles, which then determine port state.

RSTP

Rapid Spanning Tree Protocol – introduced in 1998, standardised in 2001 as 802.1w amendment to 802.1D, revised in 2004, now included in 802.1Q.

Similarities with classic STP:

- Election of Root Bridge
- Selection of Root Port
- Election of Designated Port
- Forwarding and Blocking (named ‘Discarding’ in RSTP) states

Differences (mainly so convergence happens quicker):

- **Alternate port** can quickly replace a root port without waiting for the port to enter a forwarding state
- **Backup port** can quickly replace a designated port without waiting for the port to enter a forwarding state
- Shortened timers – MaxAge set as 3 times Hello (6 seconds)
- Mechanism to ask a neighbouring switch about a possible link failure instead of waiting for MaxAge to expire
- Hello BPDUs are generated by each switch – this means each switch needs to have the same timer settings
- STP blocking and disabled states are lumped together into a discarding state
- Even when a state transition requires waiting, RSTP does not use a listening state and jumps right into the learning state.

An **alternate port** is the one with the second-lowest root cost, right after the root port. When a switch stops receiving Hello BPDUs on its root port, it quickly (either after MaxAge, i.e. 6 seconds, or after determining the link has failed):

1. informs the switch on the other side of its alternate port that a topology change has occurred and asks it to flush relevant entries in MAC table to avoid loops
2. flushes relevant entries in its own MAC table to avoid loops
3. designates its alternate port the new root port and moves it into forwarding state
4. moves the old root port into a discarding state
5. selects a new alternate port

This process bypasses the listening and learning states.

RSTP port/link types:

- point-to-point – full-duplex link between two switches. Default if PortFast is disabled. Convergence on these links requires entering the learning state.
- point-to-point edge – full-duplex link between a switch and a single edge device (e.g. a PC). Since the switch doesn't know what is on the other side, this link type is set when PortFast is set on a port. Ports on these links bypass learning state when converging.
- shared – half-duplex links are assumed to be connected to a hub, necessitating slower convergence

EtherChannel – bundles up to 8 L2 or L3 links together into a single link. Provides load-balancing and quick failover. More on that to follow.

BDPU Guard – If enabled on a port, disable the port if it receives a BPDU. This prevents rogue switch attacks (i.e. attacker becoming a root switch to listen to traffic) and incidents (a non-STP/RSTP aware switch plugged into network causing loops). Should be enabled on all access ports:

- (config)# spanning-tree bdpuguard default – enable BDPU Guard on all access links by default
- (config)# no spanning-tree bdpuguard default – disable BDPU Guard on all access links by default
- (config-if)# spanning-tree bdpuguard enable – enable BDPU Guard on the specified interface
- (config-if)# spanning-tree bdpuguard disable – disable BDPU Guard on the specified interface

BDPU Guard disables the interface if it receives a BPDU. To manually restore it:

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

To enable automatic recovery:

- (config)# errdisable recovery cause bdpuguard
- (config)# errdisable recovery interval *seconds* – Set interval before automatic recovery.

NOTE: This applies to automatic recovery from all types of errors.

PortFast allows the interface to transition directly from blocking to forwarding state without going through listening and learning states. It should be enabled only on ports connected to end devices, not other switches. (End devices cannot possibly form a loop but switches can.) Additionally, PortFast-enabled ports should also be configured with BDPU Guard.

- (config)# spanning-tree portfast default – enable PortFast on all access links by default
- (config)# no spanning-tree portfast default – disable PortFast on all access links by default
- (config-if)# spanning-tree portfast – enable PortFast on a specific interface (must be access port)
- (config-if)# spanning-tree portfast network – force-enable PortFast on a specific interface
- (config-if)# spanning-tree portfast disable – disable PortFast on a specific interface

RootGuard is used to prevent a device connected to a given port from becoming the root switch. Use this if you have manually set the Root Bridge near the center of the network and you want to make sure any new switches added to the network don't take over the role.

LoopGuard is used to prevent loops that might arise from connectivity failures, for example a unidirectional link where frames can only travel in one direction due to a broken wire inside the cable. Incompatible with RootGuard, will be automatically disabled if RootGuard is enabled on a port.

- (config-if)# spanning-tree guard root – Enable RootGuard on a port
- (config-if)# spanning-tree guard loop – Enable LoopGuard on a port
- (config-if)# spanning-tree guard none – Disable RootGuard and LoopGuard on a port
- (config)# spanning-tree loopguard default – Enable LoopGuard on all interfaces by default.

Chapter 10 – RSTP and EtherChannel Configuration

Per-VLAN spanning tree

Original STP and RSTP do not optimize for multiple VLANs. Without any extensions to the protocol, the BDPUs travel in the native VLAN and there is just one spanning tree (CST Common Spanning Tree) for all VLANs.

Per-VLAN spanning trees make load balancing possible. An administrator can configure different spanning trees to take different paths through the campus network by configuring different Root Bridges (and Port Priorities if desired) for different trees. All trees will still take the same paths without manual configuration.

There are three multiple spanning tree protocols available:

- PVST+ - Per-VLAN Spanning Tree, a Cisco-proprietary extension to classic STP. One separate spanning tree for each enabled VLAN. If used the administrator will typically configure one Root Bridge for half the VLANs and a different Root Bridge for the other half (multiple VLANs can be configured to use the same Root Bridge with a single command). This results in half the VLANs taking one path through the campus and the other half taking a different path.
- RPVST+ - Rapid Per-VLAN Spanning Tree, a Cisco-proprietary extension to RSTP. One separate spanning tree for each enabled VLAN. Works similarly to PVST+. This is the default mode since IOS version 15.2
- MST – Multiple Spanning Tree Protocol, IEEE standard, defined in 802.1Q amendment 802.1s. Administrator can create separate spanning trees (usually 2) and allocate multiple VLANs to each of them. Less CPU overhead than PVST+ and RPVST+ because there are less overall spanning trees for the switches to calculate.

If there are 200 VLANs in use then with PVST+ and RPVST+ there will be 2 Root Bridges used, 2 paths and 200 spanning trees. 100 use one path and the other 100 use the other path, but they are each calculated separately. With MST there will be 2 Root Bridges used, 2 paths and 2 spanning trees.

All three protocols embed the VLAN number in the priority part of Bridge ID. 2 bytes originally reserved for priority are divided into 4-bits of priority and 12 bits System ID Extension. As a result:

1. $\text{newpriority} = \text{floor}(\text{oldpriority}/4096)$
2. $\text{System ID Extension} = \text{oldpriority} \% 4096$

They also VLAN-tag the BDPUs, whereas the bare STP/RSTP sends them in native VLAN.

Selecting the protocol in use:

- (config)# spanning-tree mode mst – use MSTP
- (config)# spanning-tree mode rapid-pvst – use RPVST+
- (config)# spanning-tree mode pvst – use PVST+

Set priority:

- (config)# spanning-tree vlan *n* priority *m* – where *m* is 0 or a multiple of 4096
- (config)# spanning-tree vlan *n* root primary – set priority to 24576 or to a value 4096 less than the current lowest priority setting in the network.
- (config)# spanning-tree vlan *n* root secondary – set priority to 28672, which in ordinary circumstances would be higher than root but lower than other switches.

Set port cost:

- (config-if) spanning-tree [vlan *n*] cost *x* – manually set port cost for a given port. vlan parameter is optional. If given, set the cost only for a particular VLAN's spanning tree, if not, for all spanning trees.

RPVST+ configuration example:

```
Sw1(config)# spanning-tree mode pvst
```

Make Sw1 the Root Bridge for VLANs 10 – 13:

```
Sw1(config)#spanning-tree vlan 10-13 priority 4096
```

```
Sw2(config)# spanning-tree mode pvst
```

Make Sw2 the Root Bridge for VLANs 20 – 23:

```
Sw2(config)#spanning-tree vlan 20-23 priority 4096
```

MST configuration example:

```
Sw1(config)#spanning-tree mode mst
Sw1(config)#spanning-tree mst configuration
Sw1(config-mst)#name study-ccna
Sw1(config-mst)#revision 1
Sw1(config-mst)#instance 2 vlan 10,11,12,13
Sw1(config-mst)#instance 3 vlan 20,21,22,23
    Make Sw1 the Root Bridge for instance 2 VLANs 10 – 13:
Sw1(config)#spanning-tree mst 2 priority 4096
```

```
Sw2(config)#spanning-tree mode mst
Sw2(config)#spanning-tree mst configuration
Sw2(config-mst)#name study-ccna
Sw2(config-mst)#revision 1
Sw2(config-mst)#instance 2 vlan 10,11,12,13
Sw2(config-mst)#instance 3 vlan 20,21,22,23
    Make Sw1 the Root Bridge for instance 3 VLANs 20 – 23:
Sw2(config)#spanning-tree mst 3 priority 4096
```

Configuring L2 EtherChannel

An EtherChannel is a bundle of physical links that, for all intents and purposes, act as a single virtual link from the point of view of STP so each link in the bundle can forward traffic. It also provides redundancy – should one of the bundled physical links fail, the remaining links will remain available and the EtherChannel as a whole will continue functioning.

Manual configuration:

- (config-if)# channel-group *n* mode on – assign a port to the EtherChannel with ID *n*

Show EtherChannel status:

- # show etherchannel *n* summary – show summary status – state, ports bundled – of the EtherChannel with ID *n*
- # show etherchannel *n* port-channel – show detailed info – including auto-negotiation – on the EtherChannel with ID *n*

EtherChannel virtual interfaces are named **Port-channel***n* (usually shortened to **Pon**), where *n* is the ID, eg. Po1 for ID 1.

Auto-negotiation can be achieved with two protocols: Cisco-proprietary PAgP (Port Aggregation Protocol) and IEEE standard LACP (Link Aggregation Control Protocol) which is in more common use. They are mostly analogous. At least one side of the link must be set to initiate auto-negotiation, the other can be set to wait for the other to begin. This is done with:

- (config-if)# channel-group *n* mode active – Use LACP, initiate auto-negotiation
- (config-if)# channel-group *n* mode passive – Use LACP, wait for the other side to initiate auto-negotiation
- (config-if)# channel-group *n* mode desirable – Use PAgP, initiate auto-negotiation
- (config-if)# channel-group *n* mode auto – Use PAgP, wait for the other side to initiate auto-negotiation

Manual configuration (**mode on**) disables EtherChannel auto-negotiation and cannot be used with auto-negotiation enabled on the other side of the link. **Active-active** or **desirable-desirable** will work, **passive-passive** and **auto-auto** will not.

LACP configuration example (also configure the same settings on the switch on the other side of the link):

```
Sw1(config)#interface range f0/23 - 24
Sw1(config-if-range)#channel-group 1 mode active
Sw1(config-if-range)#exit
Sw1(config)#interface port-channel 1
Sw1(config-if)#switchport mode trunk
Sw1(config-if)#switchport trunk native vlan 199
```

Troubleshooting EtherChannel

Regardless of whether auto-negotiation was used or not, the following parameters must match across all bundled ports:

- Ports are not shut down
- Speed
- Duplex setting
- VLAN mode: all must be trunk or all must be access
- Access VLAN, if port set as an access port
- Allowed VLANs and native VLAN, if port set as trunk port
- STP settings (port cost)

If there is a mismatch, the PortChannel virtual interface will be put in an *err-disabled* state. To recover, fix the underlying mismatch and then do:

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

Sidenote: shutdown and no shutdown command automatically apply to the associated physical interfaces when the command is run at the PortChannel level.

Load distribution: EtherChannel can distribute its load between all links based on a number of criteria. This is set by:

- (config)# port-channel load-balance *method* – where method is one of the following:
 - src-mac (default on most switches)
 - dst-mac
 - src-dst-mac
 - src-ip
 - dst-ip
 - src-dst-ip
 - src-port
 - dst-port
 - src-dst-port
- # show etherchannel load-balance – shows the enabled method
- # test etherchannel load-balance interface PoN pol {mac|ip|port} *src dst* – simulate which link would be used to carry traffic between given source and destination

Part IV – IPv4 Addressing

Chapter 11 – Perspectives on IPv4 Subnetting

All concepts that were discussed in this chapter in the book are covered in the notes below.

Chapter 12 – Analyzing Classful IPv4 Networks

Classful addressing: 5 classes, including 3 (A, B and C) for conventional usage:

- Class A: 1.1.1.1-126.255.255.255, unicast, 8 network bits default subnet mask
- Class B: 127.0.0.0-191.255.255.255, unicast, 16 network bits default subnet mask
- Class C: 192.0.0.0-223.255.255.255, unicast, 24 network bits default subnet mask
- Class D: 224.0.0.0-239.255.255.255, multicast
- Class E: 240.0.0.0-255.255.255.255, reserved

Addresses beginning with 0 and 127 are reserved (the latter for loopback)

Chapter 13 – Analyzing Subnet Masks

Subnet mask notations:

- Binary – e.g. 11111111 11111111 11111111 00000000
- Dotted-decimal notation (DDN) – e.g. 255.255.255.0
- Prefix, also called CIDR (Classless interdomain routing) slash notation – e.g. /24

Chapter 14 – Analyzing Existing Subnets

It's possible to split a larger network into several smaller subnets through the process of 'subnetting'. Having multiple subnets means you are dividing your network into smaller, contained layer 3 domains which give better security (a router or firewall can limit traffic allowed between subnets) and performance (broadcast traffic is contained within the subnet). Subnets are usually allocated by the network designer based on physical location, corporate department (different subnets for sales and accounts), device type (different subnets for servers) etc.

There are four 8 bit octets within a 32 bit IPv4 address.

For example in 192.168.0.10:

The 1st octet is written as '192' in dotted decimal notation.

The 2nd octet is written as '168' in dotted decimal notation.

The 3rd octet is written as '0' in dotted decimal notation.

The 4th octet is written as '10' in dotted decimal notation.

When working in decimal format, reading from right to left the columns of the number go up in powers of 10 because each has 10 possible values, 0 – 9.

10,000 1000 100 10 1

192 is written as '192' in decimal because it is one 100's, nine 10's and two 1's.

$100 + 90 + 2$

When subnetting it's easier to work in binary format rather than decimal.

With binary format, reading from right to left the columns of the number go up in powers of 2 because each has 2 possible values, 0 or 1.

Each of the 8 bits in an IPv4 address's octet therefore has this defined set of binary values:

128 64 32 16 8 4 2 1

When all eight binary bits in an octet are set to 1 (11111111), the decimal value is 255 because $128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$.

255 in decimal is 11111111 in binary, they're just two different ways of writing the same value.

The IPv4 address 192.168.0.10 is written like this in binary:

11000000 . 10101000 . 00000000 . 00001010

$192 = 128 + 64$

11000000

$168 = 128 + 32 + 8$

10101000

$0 = 00000000$

$10 = 8 + 2$

00001010

Let's say we want to subnet the 172.16.0.0/16 network into smaller 172.16.0.0/19 subnets.

Subnet masks always consist of a solid contiguous block of 1s followed by a solid contiguous block of 0s. (111000.11000000.00100101.00001111 is not a valid subnet mask.)

The /19 indicates that the subnet mask has 19 "1's" followed by all 0's in binary:

11111111.11111111.11100000.00000000

Because we might be asked to on the exam, let's translate the subnet mask from binary to decimal before we get into the subnetting:

All eight bits are set to 1 in the first two octets. The decimal value for them is 255 because:

$$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255.$$

In the third octet, the first 3 bits are set to 1:

$$128 + 64 + 32 + 0 + 0 + 0 + 0 + 0 = 224.$$

The subnet mask is therefore 255.255.224.0 in dotted decimal notation.

Okay, let's start subnetting. The third octet is the interesting octet for subnetting because the final "1" in the binary subnet mask is there: 11111111.11111111.11**1**00000.00000000

The number value in the third octet will change for each subnet.

We note the binary value of the final subnet bit by looking at the position of the final "1". In our example, the final "1" has a value of 32:

11**1**00000

$$128 - 64 - \mathbf{32} - 0 - 0 - 0 - 0 - 0$$

This means that the block size is 32. The subnets will go up in increments of 32 (starting from 0), on the 'interesting' third octet.

Our subnets are:

172.16.**0**.0/19

172.16.**32**.0/19

172.16.**64**.0/19

172.16.**96**.0/19

Etc. up to our final subnet:

172.16.**224**.0/19

For the 1st subnet:

172.16.0.0/19 is the network address

172.16.0.1 to 172.16.31.254 are the usable IP addresses which can be allocated to hosts.

172.16.31.255 is the broadcast address

For the 2nd subnet:

172.16.32.0/19 is the network address

172.16.32.1 to 172.16.63.254 are the usable IP addresses which can be allocated to hosts.

172.16.63.255 is the broadcast address

For the 3rd subnet:

172.16.64.0/19 is the network address

172.16.64.1 to 172.16.95.254 are the usable IP addresses which can be allocated to hosts.

172.16.95.255 is the broadcast address

Etc.

Up to our final subnet:

172.16.224.0/19 is the network address

172.16.224.1 to 172.16.255.254 are the usable IP addresses which can be allocated to hosts.

172.16.255.255 is the broadcast address

When subnetting a larger network into smaller subnets, the number of usable available subnets is equal to 2^N , where N is the number of 'borrowed' network bits.

In each of those subnets, the number of usable IP addresses which we can allocate to hosts is equal to $2^H - 2$, where H is the number of remaining host bits. We subtract 2 from 2^H because the lowest IP address in a subnet is reserved for the Subnet ID, and the highest for the broadcast address.

In our example we subnetted the 172.16.0.0/16 network into smaller 172.16.0.0/19 subnets.

172.16.0.0 is a Class B address with a default subnet mask size of /16

We are using /19, so we are 'borrowing' 3 bits (19 minus 16) which would normally be used as host bits to use them as network bits instead.

$$2^3 = 8$$

We are subnetting the one 172.16.0.0/16 network into 8 smaller /19 subnets.

There are 13 host bits (32 bits in an IPv4 address minus the 19 network bits).

$$2^{13} - 2 = 8190$$

There are 8190 available host addresses in each subnet.

In the real world you can use an online subnet calculator, but you need to be able to do subnetting manually for the CCNA exam. Learning how to do it is time well spent because it gives you a much deeper understanding of how IP networking works.

Part V – IPv4 Routing

Chapter 15 – Operating Cisco Routers

Differences between routers and switches:

1. Routers forward traffic between IP networks based on the layer 3 destination IP address, switches don't (L3 switches are an exception). Switches forward traffic based on the layer 2 MAC address.
2. On routers, IP addresses are configured per physical interface (and possibly virtual subinterfaces), not per VLAN.
3. Routers can have an auxiliary Aux port (configured in **line aux 0**) that can be connected to an external modem or phone line for emergency access (faster interfaces are typically used in the real world now).
4. Routers have no MAC address table to show via **# show mac address-table**

Routers do not support the **show interfaces status** command. To list router interfaces status:

- **# show ip interface brief** – a brief summary
- **# show protocols [ifname]** – a somewhat longer summary, optional argument limits output to one interface
- **# show interfaces [ifname]** - longer summary, optional argument limits output to one interface
- **# show ip interface [ifname]** – full status, optional argument limits output to one interface

Chapter 16 – Configuring IPv4 addresses and Static Routes

Route types:

- Local – route to the IP address of a configured interface
- Connected – route to the subnet defined by IP address and subnet mask of a configured interface
- Static – manually defined
- Dynamic – learned via routing protocols
- Default gateway (AKA gateway of last resort) – used when destination doesn't match any other route

Static Routes

Adding static routes:

- (config)# ip route *subnetID netmask nexthop* – specify next-hop IP address on next router in path.
- Example - ip route 192.168.10.0 255.255.255.0 10.10.0.2
- **OR**
- (config)# ip route *subnetID netmask interface* – specify outbound interface on this router. This will show up as *directly connected*
- Example - ip route 192.168.10.0 255.255.255.0 FastEthernet0/1

The 1st option is preferred but the 2nd option is useful if the next hop IP address might change (ie it is allocated via DHCP)

The netmask must be written in Dotted Decimal Notation format when entering the command.

When adding a static route, IOS checks whether the outbound interface is up (when specifying an outbound interface), or if it has a route to the next hop router's IP address (when specifying next-hop IP address). If the check fails, the route is not added at all (and will not come up when the error is fixed). To override that, you can add ***permanent*** to the configuration command.

Delete a static route:

- (config) no ip route *subnetID netmask routerIP/interface*

Adding a static default route on a router:

- (config)# ip route 0.0.0.0 0.0.0.0 *nexthop/interface*

Routers use a default route as the ‘last resort’ when a more specific route is not available in their routing table. They are most commonly used to send traffic going out of the site’s internal network to the Internet. It’s obviously unfeasible to add separate routes to every possible destination on the Internet, so a default route acts as a catch-all for that traffic.

Route Selection – 1st Consideration: Longest Prefix Match

When two or more routes match a given destination IP address, the one with the longest (most specific) prefix is selected.

For example if a router receives a packet with destination address 192.168.10.10 and it has these 2 matching routes:

```
S 192.168.0.0/16 [1/0] via 10.10.10.2  
S 192.168.10.0/24 [1/0] via 172.16.0.2
```

It will forward the packet according to the 192.168.10.0/**24** route because it has a longer prefix match than the 192.168.0.0/**16** route.

Note that the 2 routes in this example both matched but are not exactly the same, one has a longer prefix.

Route Selection – 2nd Consideration: Administrative Distance

When a router has multiple routes **to the exact same network and prefix**, the one with the lowest *administrative distance* (0-255) is selected and installed in the routing table.

Route type	Administrative distance
Directly connected	0
Static	1
EIGRP summary route	5
eBGP	20
Internal EIGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
External EIGRP	170
iBGP	200
DHCP default gateway	254
Unknown	255

The others will be kept in their respective routing protocols' databases until they are needed (i.e. when the routes with lower administrative distances are removed).

For example if a router learns routes to 192.168.10.0/24 via both RIP and OSPF, the OSPF route will be installed in the routing table because it has a better Administrative Distance.

If a router learns routes to 192.168.0.0/16 via RIP and to 192.168.10.0/24 via OSPF, they will both be installed in the routing table because they are not to exactly the same network and prefix.

By default static routes have priority over dynamic routes learned via a routing protocol, but an administrator can override that by setting the static route *administrative distance* to a higher value:

- (config)# ip route *subnetID netmask nexthop/interface* **admdistance**

This is known as a **floating static route** and is most commonly used to add a backup static route across an alternate (usually lower bandwidth) link in case the dynamic route is removed from the routing table because the main link went down. A higher Administrative Distance must be set for the static route or it would be preferred.

Route Selection – 3rd Consideration: Metric

When a router has multiple routes **to the exact same network and prefix** and **with the same Administrative Distance** (ie learned via the same routing protocol) the one with the best metric is selected and installed in the routing table. The metric is the routing protocol's way of determining which is the best path to a destination network.

RIP uses 'hop count' as its metric. The path which goes through the least amount of routers to get to the destination network is preferred. This takes no account of bandwidth on the links which can lead to low bandwidth paths being preferred. This and its maximum hop count of 15 makes RIP unsuitable for most modern production networks.

EIGRP's metric is called 'metric' (kind of like how an orange is called an orange and it's also orange in color.) OSPF uses 'cost' as its metric.

Both OSPF and EIGRP's metrics automatically take the bandwidth of the available links to the destination network into consideration. They are also scalable and fast to converge, making them suitable to use as the IGP Interior Gateway Protocol for modern networks.

Organisations will typically choose either OSPF or EIGRP as their internal routing protocol.

EIGRP was originally Cisco proprietary but is now an open standard. There continues to be limited support for it in other vendor's devices though.

OSPF is the main routing protocol focused on in the CCNA exam, but you can still be asked questions about the other protocols.

The higher the bandwidth of a link, the lower the OSPF cost of that link by default. Lower cost links are preferred over higher cost links when calculating routes to install in the routing table. This usually results in packets taking the paths you want them to, but you can manually set the OSPF cost of links if you want to manipulate the paths that packets take over the network.

If a router learns multiple paths **to the exact same network and prefix** and with **the same Administrative Distance** and **same metric** it will install them all in its routing table and perform ECMP Equal Cost Multi Path load balancing of traffic over them. Load balancing only occurs with those exact conditions in place (otherwise only the best route will be preferred and installed in the routing table), with the exception of EIGRP which is capable of performing Unequal Cost Multi Path load balancing if configured by the administrator.

Route Verification

Show routes:

- # show ip route *[type]* – shows all routes, optionally filtered by type (local, connected, static, ospf etc.)
- # show ip route *address* – shows all routes to a given address
- # show ip route *address [mask/prefix]* – shows all routes to a given subnet/address. Mask can be given either as a CIDR prefix or in DDN notation.

Example line of output:

```
O 10.2.2.0/30 [110/128] via 10.2.2.5, 14:31:52, Serial0/0/1
```

This is an OSPF-learned route to subnet 10.2.2.0/30 with administrative distance of 110 and metric of 128. Next-hop router is 10.2.2.5, route was first learned at 14:31:52, packet will go out via Serial0/0/1

Chapter 17 – IP Routing in the LAN

There is typically a one-to-one relationship between an IP subnet and a VLAN in the LAN campus. For example Engineering hosts are in IP subnet 10.10.10.0/24 and VLAN 10, and Sales hosts are in IP subnet 10.10.20.0/24 and VLAN 20

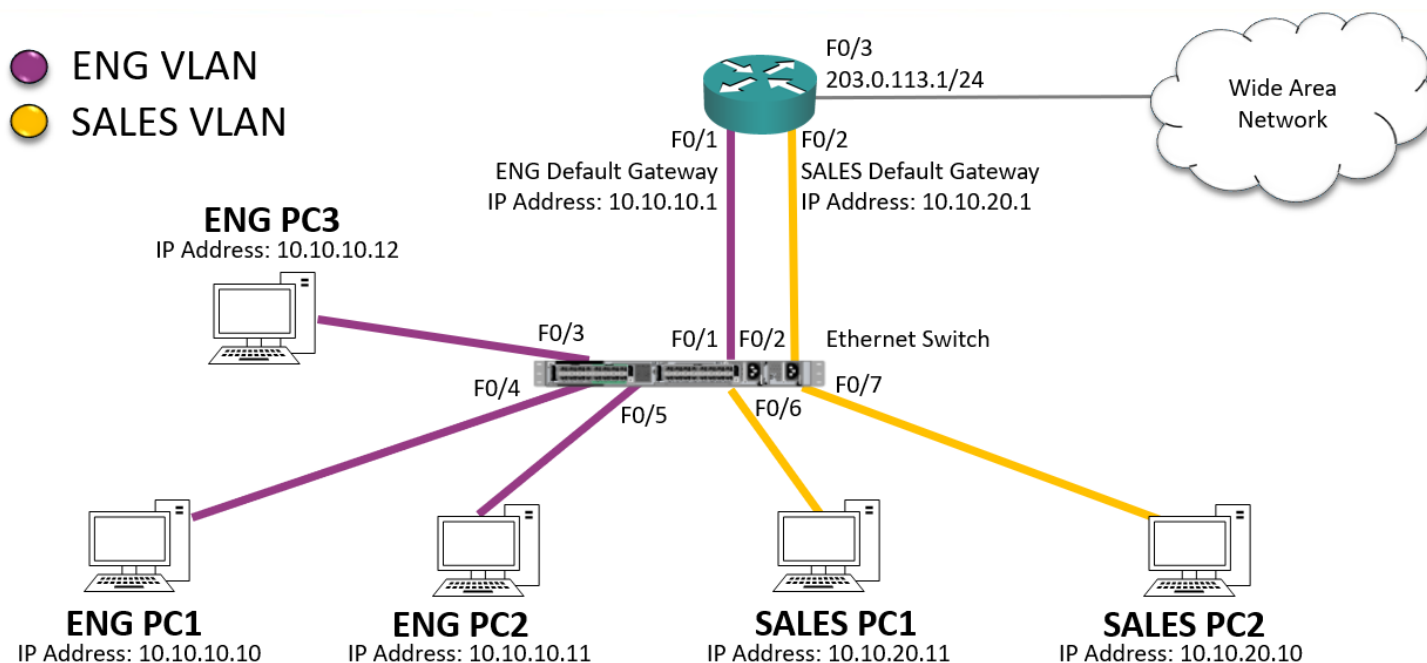
Hosts are segregated at Layer 3 by being in different IP subnets, and at Layer 2 by being in different VLANs.

Hosts in different IP subnets need to send traffic via a router to communicate with each other.

There are 3 options for routing in the LAN:

- Router with separate interfaces. Each VLAN / IP subnet has a separate dedicated physical interface on the router.
- Router On A Stick. Multiple VLAN / IP subnets are trunked to the same physical interface on the router. Each VLAN / IP subnet has a separate dedicated virtual subinterface under the physical interface.
- Layer 3 switch

Router with separate interfaces



The router interfaces are configured with IP addresses for their associated IP subnet and VLAN.

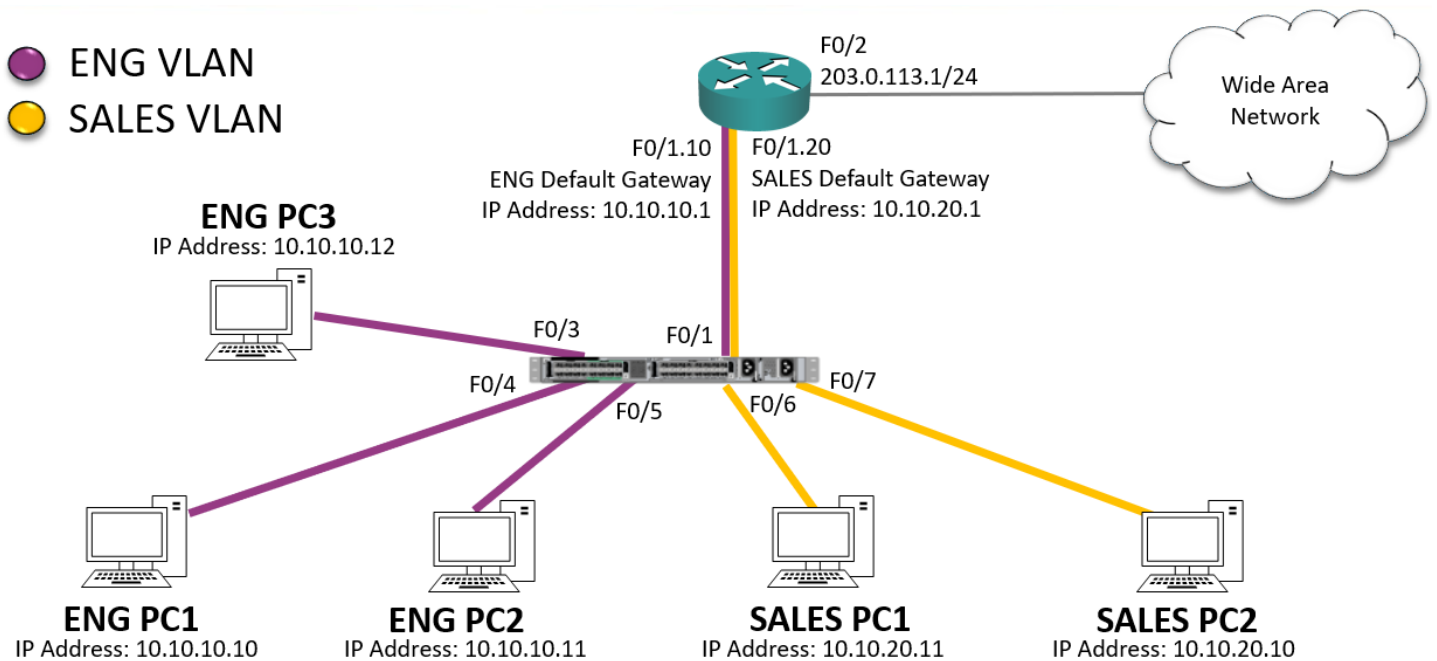
```
R1(config)#interface FastEthernet 0/1
R1(config-interface)#ip address 10.10.10.1 255.255.255.0
R1(config)#interface FastEthernet 0/2
R1(config-interface)#ip address 10.10.20.1 255.255.255.0
R1(config)#ip route 0.0.0.0 0.0.0.0 203.0.113.2
```

The switch ports connected to the router are configured as access ports in the relevant VLAN.

```
SW1(config)#interface FastEthernet 0/1
SW1(config-if)#switchport mode access
SW1(config-if)#switchport access vlan 10
SW1(config)#interface FastEthernet 0/2
SW1(config-if)#switchport mode access
SW1(config-if)#switchport access vlan 20
```

This topology is rarely used in real networks. Using separate interfaces on the router is not scalable because you need a separate physical interface for every VLAN – you are liable to run out of interfaces. Also, traffic being routed within the campus has to go up and down physical Ethernet cables to the router which is not efficient for performance.

ROAS Router-On-A-Stick



Subinterface names use the following format: Gi0/0.10, where Gi0/0 is the physical interface and 10 is the subinterface number (the same as VLAN ID unless the administrator is a maniac, but this is not a requirement!)

VLANs on a trunk need to be specified manually on each subinterface.

```
R1(config)#interface FastEthernet 0/1
R1(config-interface)#no ip address
R1(config-interface)#no shutdown
R1(config)#interface FastEthernet 0/1.10
R1(config-interface)#encapsulation dot1q 10
R1(config-interface)#ip address 10.10.10.1 255.255.255.0
R1(config)#interface FastEthernet 0/1.20
R1(config-interface)#encapsulation dot1q 20
R1(config-interface)#ip address 10.10.20.1 255.255.255.0
R1(config)#ip route 0.0.0.0 0.0.0.0 203.0.113.2
```

```
SW1(config)#interface FastEthernet 0/1
SW1(config-if)#switchport mode trunk
```

This topology is also not typically used. You do not need a separate physical interface for every VLAN so you are less likely to run out of interfaces, but traffic being routed within the campus still has to go up and down the same physical Ethernet cable to the router, and there is more contention for bandwidth than when using separate interfaces.

Native VLAN can be configured either on the physical interface or on a subinterface. If using the physical interface directly:

- (config) interface Gi0/0 – enter physical interface configuration context
- (config-if) ip address *ipaddress netmask*

If using a subinterface:

- (config) interface Gi0/0/10 – enter subinterface configuration context
- (config-if) encapsulation dot1q *n* **native**
- (config-if) ip address *ipaddress netmask*

Show VLAN configuration:

- show vlans – show all configured VLANs, associated (sub)interfaces (note that the analogous command on a switch would be **show vlan [brief]**)

Layer 3 switch

A layer 3 switch is a device that supports both layer 2 frame forwarding and layer 3 IP routing.

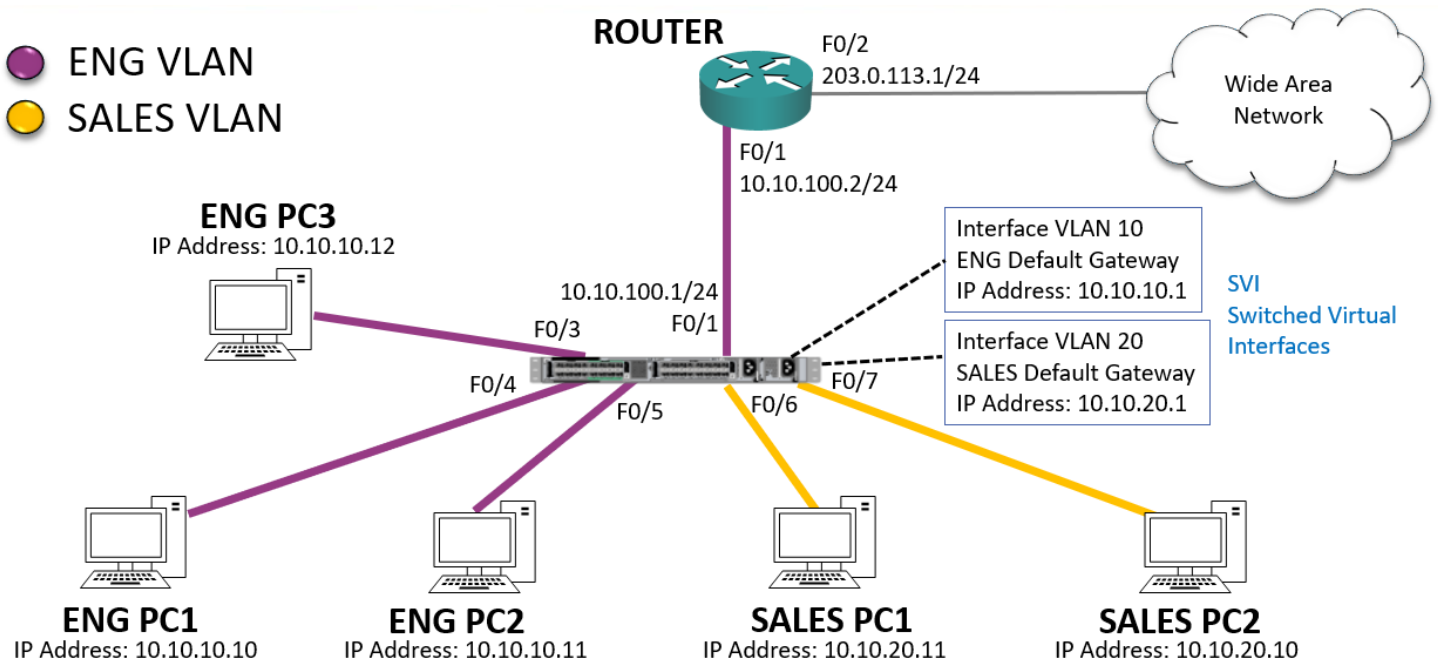
A physical port on a layer 3 switch is a layer 2 ‘switchport’ by default which supports access VLANs and trunks, but it can alternatively be set as a routed port which you can configure an IP address on. Ports are one or the other, either layer 2 or layer 3.

IP addresses can be configured in two ways (both can be configured on the same switch):

- SVI based (under VLAN interface)
- Switch routed port (under physical port interface)

SVIs based routing creates a virtual interface for each VLAN and routes packets between these interfaces. Initial configuration (save and reload to take effect):

- (config)# sdm prefer lanbase-routing – reconfigure switch ASIC to make routing possible
- (config)# ip routing – enable layer 3 routing



SVIs:

```
SW1(config)#interface vlan 10
SW1(config-if)#ip address 10.10.10.1 255.255.255.0
SW1(config)#interface vlan 20
SW1(config-if)#ip address 10.10.20.1 255.255.255.0
```

Layer 3 port:

```
SW1(config)#interface FastEthernet 0/1
SW1(config-if)#no switchport
SW1(config-if)#ip address 10.10.100.1 255.255.255.0
```

Allow connectivity between subnets and add default route to WAN:

```
SW1(config)#ip routing
SW1(config)#ip route 0.0.0.0 0.0.0.0 10.10.100.2
```

```
R1(config)#interface FastEthernet 0/1
R1(config-interface)#ip address 10.10.100.2 255.255.255.0
R1(config)#interface FastEthernet 0/2
R1(config-interface)#ip address 203.0.113.1 255.255.255.0
R1(config)#ip route 0.0.0.0 0.0.0.0 203.0.113.2
R1(config)#ip route 10.10.0.0 255.255.0.0 10.10.100.1
```

This is the topology most commonly used in modern networks. Traffic being routed within the campus is routed across the switch backplane, it does not need to travel over physical cables to an external router. You may still need an external router for WAN connectivity as shown in the example above (layer 3 switches only have Ethernet ports) or for services only supported on particular router models.

Troubleshooting

- has **sdm prefer** global config command been executed and the switch rebooted?
- is the VLAN created on the switch with *vlan x* command?
- is at least one up/up, STP forwarding, not VTP-pruned switch port assigned to the VLAN?
(this is the most common error – make sure the VLAN is associated with at least one port which is up ie. connected to a live port on the other side of the link)
- has the VLAN been shutdown?
- has the VLAN interface been shutdown?

Layer 3 Etherchannel

Two or more point-to-point L3 links can be bundled together in the same way as L2 Etherchannel. Configuration is similar:

Assign interface to Etherchannel manually:

- (config-if)# channel-group *n* mode on – assign a physical port to the EtherChannel with ID *n*
- (config-if)# no switchport – make the port a routed one

or configure auto-negotiation:

- (config-if)# channel-group *n* mode active – Use LACP, initiate auto-negotiation
- (config-if)# channel-group *n* mode passive – Use LACP, wait for the other side to initiate auto-negotiation
- (config-if)# channel-group *n* mode desirable – Use PAgP, initiate auto-negotiation
- (config-if)# channel-group *n* mode passive – Use PAgP, wait for the other side to initiate auto-negotiation
- (config-if)# no switchport

Configure PortChannel interface (configure IP address on interface after this):

- (config)# interface Port-channel *n* – configure the new port-channel interface
- (config-if)# no switchport – possibly redundant on port-channel interface but better safe than sorry

Troubleshooting:

- Are both the physical interfaces and port-channel interface configured with **no shutdown**?
- Are **speed** and **duplex** settings identical between all physical ports involved?

Chapter 17 – IPv4 troubleshooting

Ping command – self-explanatory.

Keep in mind that for a ping to succeed you need both a route to the pinged host and the pinged host needs a route back to you.

Entering just '**ping**' without parameters will ask you about all parameters, including source IP address (useful for reverse route testing).

Ping can be used with hostnames if DNS server address has been configured.

tracert – show all hops on the way to destination. Works by manipulating TTL value of a packet (IP packets have a TTL field that is decreased by one at each hop. When it reaches zero, the packet is discarded and the sender is informed by an ICMP message. This prevents loops).

Tracert sends a test packet with a TTL of 1, then another test packet with a TTL of 2, then another test packet with a TTL of 3 etc until it reaches the final destination.

tracert without parameters works analogously without parameters.

IOS tracert uses IP packets encapsulating UDP datagram, while most other operating systems use ICMP echo requests. Because of this they can be differently impacted by traffic security rules encountered on the way.

To SSH from a router: **ssh -l username address**

Part VI – OSPF

Chapter 19 – Understanding OSPF concepts

Routing protocols

Routing protocols allow routers to exchange information on networks, pick best routes, notify each other of changes in network topology, and automatically change routes (reconverge) if required due to changes in topology (ie link comes up or goes down).

Routing protocols can be split into two main types:

- Interior gateway protocols (IGPs)
- Exterior gateway protocols (EGPs)

Interior gateway protocols are used for routing within an organisation. Exterior gateway protocols are used for routing between organisations over the Internet. The only EGP in use today is BGP (Border Gateway Protocol).

Interior gateway protocols can be split into two main types:

- Distance Vector routing protocols
- Link State routing protocols

In Distance Vector protocols, each router sends its directly connected neighbours a list of all its known networks along with its own distance to each of those networks. Distance vector routing protocols do not advertise the entire network topology - a router only knows its directly connected neighbours and the lists of networks those neighbours have advertised. It doesn't have detailed topology information beyond its directly connected neighbours. Because of this, Distance Vector routing protocols are sometimes called 'Routing by rumour'.

In Link State routing protocols, each router describes itself and its interfaces to its directly connected neighbours (information is directly exchanged only between directly connected neighbours in both Distance Vector and Link State protocols). This information is passed unchanged from one router to another. Every router learns the full picture of the network area including every router, its interfaces and what they connect to.

- AS – Autonomous System – network under control of a single organisation
- ASN – Autonomous System Number
- IGP – Internal Gateway Protocol – class of protocols designed to work within a single AS
- RIP – Routing Information Protocol. Distance Vector IGP. Exists in two versions: RIPv1 and RIPv2. RIPv2 supports VLSM, RIPv1 does not. Uses hop count as metric. Maximum hop count of 15. Rarely used outside very small or lab networks.
- IGRP – Interior Gateway Routing Protocol. IGP, Cisco-proprietary. Distance Vector IGP. Calculates a composite metric which automatically takes interface bandwidth into account, and can also consider delay, load, reliability, and maximum transmission unit (MTU). Does not support VLSM, replaced by EIGRP.
- EIGRP – Enhanced Interior Gateway Routing Protocol. Originally Cisco-proprietary, but specification has now been published. Advanced Distance Vector IGP. Calculates a composite metric similarly to IGRP.
- OSPF – Open Shortest Path First. Link State IGP. OSPF refers to OSPFv2 if not stated otherwise. Uses cost as metric, which is automatically derived from interface bandwidth by default.
- OSPFv3 – Has support for IPv6
- IS-IS – Intermediate System to Intermediate System. Link State IGP. Used mainly in Service Provider (rather than normal enterprise) internal networks. Metric does not automatically take interface bandwidth into account, it acts like hop count unless manually configured.
- EGP – External Gateway Protocol – class of protocols designed to work between AS. Used to route on the Internet
- BGP – Border Gateway Protocol – the only EGP in use today
- Metric – weight attached to each route to decide which is best

Enterprises typically choose between EIGRP and OSPF as their IGP. EIGRP is simple to configure, scalable and fast to converge, but there is limited support in non-Cisco devices.

OSPF is scalable, fast to converge and universally supported in all vendor routers.

Service providers typically choose between OSPF and IS-IS as their IGP.

OSPF fundamentals

As it is a Link State routing protocol, routers using OSPF exchange *link-state announcements* (LSAs) between each other in order to build a complete map of the network's topology. Each router stores the information in their *link-state database* (LSDB). Routers send LSAs to their neighbours who forward them to their neighbours until every router in the network area has received them – this process is called *flooding*. Each router ends up with the same information in its network area LSDB. Based on that, each router uses the Dijkstra algorithm to calculate which routes are best for itself and insert these routes into its routing table.

Each router has an RID Router ID that should be unique and is selected as below (first match applies):

- Manually: (config-router)# router-id a.b.c.d
- Highest IP of router's loopback interfaces
- Highest IP of router's regular interfaces

First, routers have to become **aware of their neighbours**, i.e. establish OSPF relationship with other routers on the connected and OSPF-enabled subnets. A router sends OSPF Hello packets to a multicast address at 224.0.0.5, to which all OSPF routers should subscribe (if the network is non-broadcast, neighbours need to be manually configured. See Chapter 21). OSPF Hello packet includes the sender RID and a list of all RIDs the sender has already seen (i.e. mentioned in any OSPF Hello packets it has received so far). After initial Hello, the routers are in **Init** state. Once two routers have exchanged their seen RIDs (and thus have the same set of seen RIDs), they become **2-way neighbours**.

In the next step, routers **exchange their LSAs** via *link-state updates* (LSUs). First, they exchange DBD database description (lists of which LSA each router has) and then they exchange any LSAs that one of them has and the other doesn't. When both have the same set of LSAs, they become **full neighbours** (AKA **adjacent neighbours** or **fully adjacent neighbours**)

Routers exchange Hello messages continually at set intervals. Should no Hello arrive from a given router in a specified time (specified as multiples of Hello interval), the router should be considered dead. Then, the router or routers that noticed this should flood LSAs reflecting this change to let others know, update their respective LSDBs and recalculate routes if necessary.

Routers also flood LSAs regularly (every 30 minutes by default) regardless of link changes. NOTE: the timer is separate for each LSA to avoid network surges that would happen when reflooding each LSA at the same time.

Not all 2-way neighbours become full neighbours. On a broadcast network (meaning a network segment / IP subnet that can have multiple routers attached to it, such as Ethernet) all routers elect one of them to become a *designated router* (DR) and another to become a *backup designated router* (BDR). The rest become DROther. The DR and BDR act as hubs of information, which is more efficient and causes less LSA traffic than every router sharing information directly with each other. All routers become full neighbours with the DR and BDR but not with each other. BDR takes over should DR fail, in which case another BDR is elected.

DR and BDR are only used on broadcast segments where multiple routers may be connected to the same IP subnet. The concept is not applicable to point-to-point links.

DR and BDR are elected for each broadcast network, not at the router level. If a router has 3 interfaces connected to 3 broadcast networks, each of those networks will have a DR and BRD elected, which could be the same or different routers.

Routers **calculate best routes** using the data gathered in their LSDBs. They run the Dijkstra algorithm against that data: for each destination subnet they select the route with the lowest sum of all outgoing interface cost and input it into their routing table. Interface cost is by default determined by its bandwidth, but can be set manually.

When OSPF inserts a route into the routing table, it uses one of its neighbours as the next-hop neighbour. Should a route to the target subnet go from R1 via R2 via R3, with R3 being directly connected to the target subnet, R1's routing table will point to R2, not R3. R2's routing table will point to R3.

Progression of neighbour states as they synchronise their LSDBs: Down → Init → 2-Way → Exstart → Exchange → Loading → Full

IPv6 OSPFv3

OSPFv2 supports IPv4.

OSPFv3 supports IPv4 and IPv6.

OSPFv2 is commonly used in IPv4 only networks, OSPFv3 is required for IPv6.

Multi-area OSPF design

Every OSPF router learns the full picture of the network including every router, its interfaces and what they connect to.

This can cause issues in large networks:

- Too many routes can use up too much router memory
- Network changes cause all routers to reconverge which takes time and CPU resources

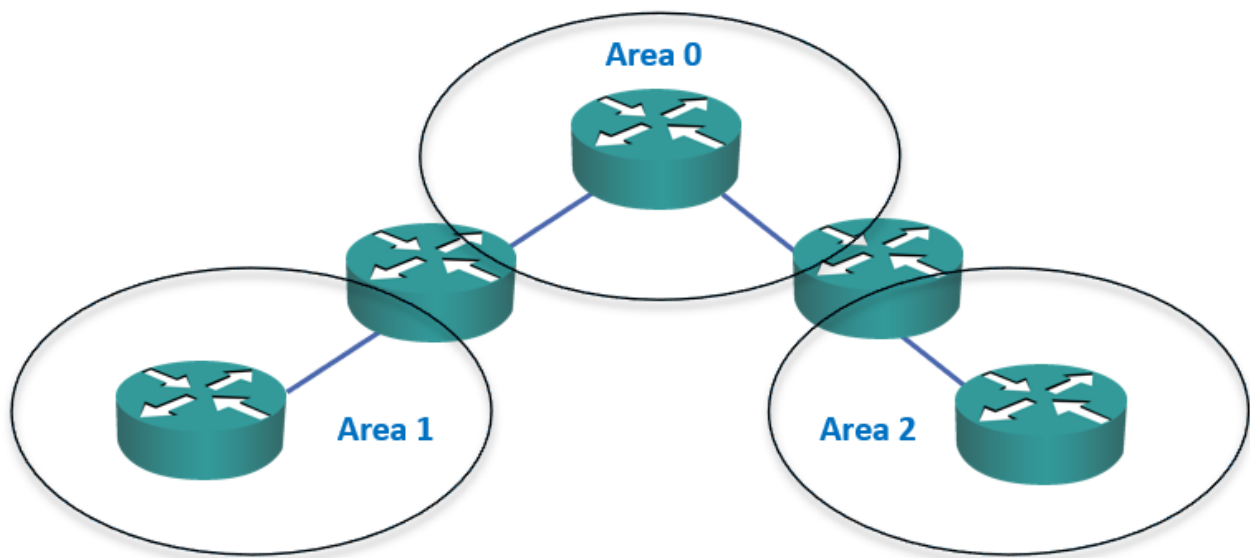
OSPF supports a hierarchical design which segments large networks into smaller areas to solve this problem. Each router maintains full information about its own area, but only summary information about other areas.

A two level hierarchy is used:

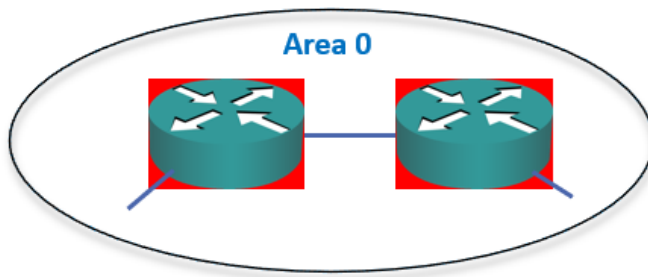
- Transit area (backbone or area 0). Does not generally contain end users.
- Regular areas (nonbackbone areas). Used to connect end users to the Transit area. By default, all transit traffic goes through the Transit area.

Routers maintain a full LSDB of other routers and links in their own area

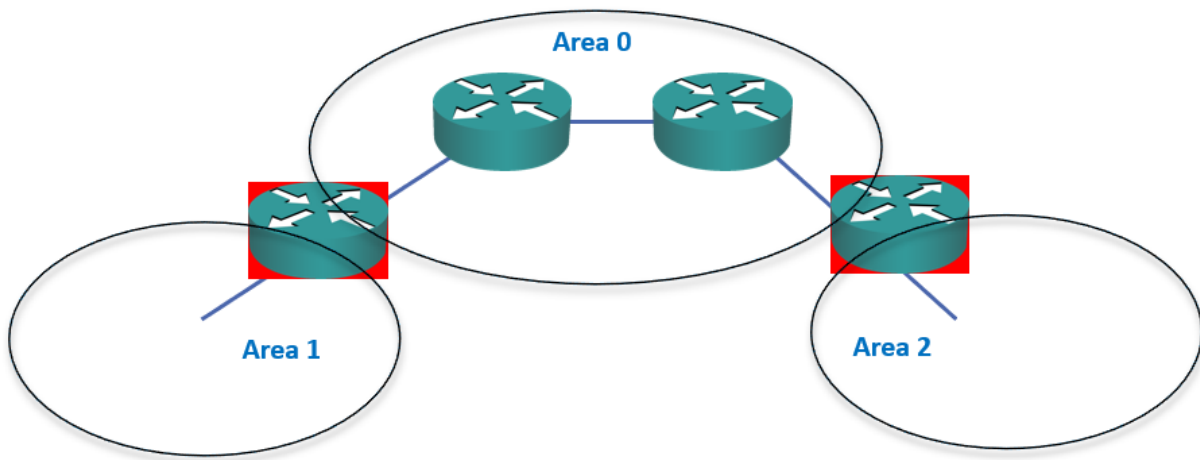
Small networks do not require a hierarchical design and all routers can be in Area 0. The recommendation is up to 50 routers in an area.



Routers which have all their OSPF interfaces in Area 0 are Backbone Routers.



Routers which have interfaces in multiple areas are Area Border Routers (ABRs).

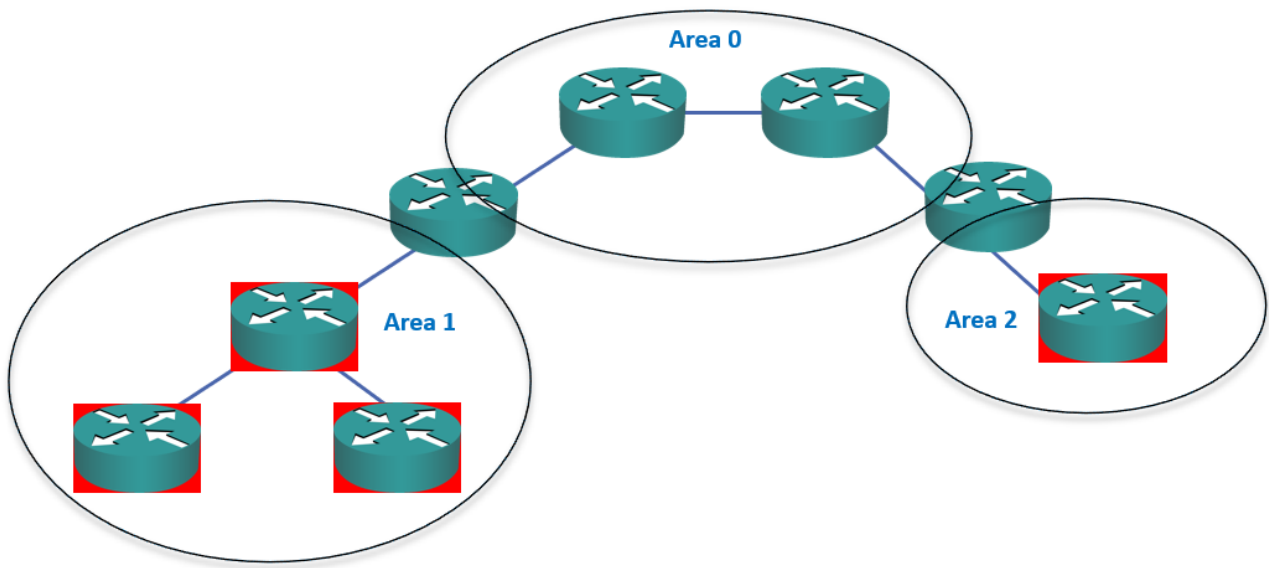


An ABR has the following characteristics:

- It separates LSA flooding zones.
- It becomes the primary point for area address summarization (this must be configured, it does not occur automatically.)
- It functions regularly as the source for default routes.
- It maintains the LSDB for each area with which it is connected.

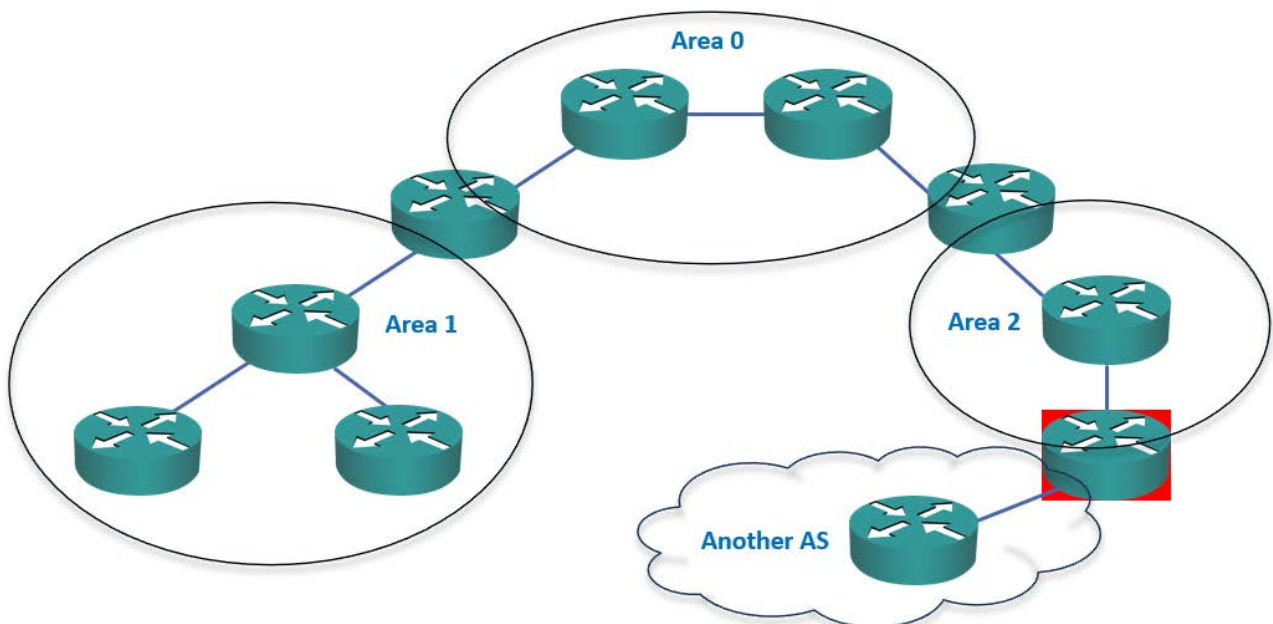
The ideal design is to have each ABR connected to two areas only, the backbone and another area, with three areas being the upper limit.

Routers which have all their OSPF interfaces in a normal area are normal internal routers.



Routers maintain a full LSDB of other routers and links in their own area. Internal area routers learn Inter Area routes to other areas from their ABRs.

Routers which redistribute routes from another routing protocol or static routes into OSPF are Autonomous System Boundary Routers (ASBRs).



Routes which are redistributed into OSPF appear as External Routes.

Chapter 20 – Implementing OSPF

To configure OSPF, it is enabled first globally on the router, and then interfaces are enabled for OSPF. When OSPF is enabled on a router's interfaces, the router will send out and listen for OSPF hello messages to peer with other OSPF routers connected to the same links, and then advertise the network and mask which is configured on those interfaces.

- (config)# router ospf *process-id* – enables OSPF globally on the router and enters OSPF router configuration mode. process-id can be any number from 1 to 65535 (this is to allow multiple OSPF processes, which is very rarely used.)
- (config-router)# network *ip-address wildcard-mask* area *area-id* – enable OSPF in the specified area for all networks within the range.

The network command uses a wildcard mask which is the inverse of a subnet mask. Subtract each octet in the subnet mask from 255 to calculate the wildcard mask.

A subnet mask of 255.255.0.0 equals a wildcard mask of 0.0.255.255

A subnet mask of 255.255.255.252 equals a wildcard mask of 0.0.0.3

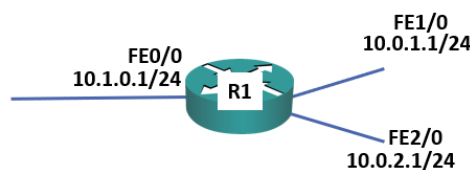
The network command means:

- Look for interfaces with an IP address which falls within this range.
- Enable OSPF on those interfaces – send out and listen for OSPF hello messages, and peer with adjacent OSPF routers.
- Advertise the network and mask which is configured on those interfaces.

```
R1(config)#router ospf 1
```

```
R1(config-router)# network 10.0.0.0 0.255.255.255 area 0
```

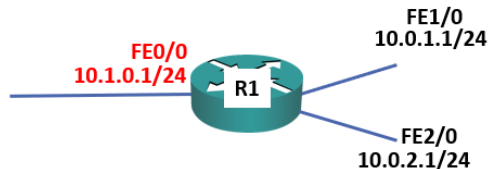
- All interfaces fall within this range
- OSPF will be enabled on all interfaces and the router will peer with adjacent OSPF routers
- Networks advertised:
 - 10.1.0.0/24
 - 10.0.1.0/24
 - 10.0.2.0/24
 - 10.0.0.0/8 is NOT advertised



Example 2:

```
R1(config)#router ospf 1
R1(config-router)# network 10.0.0.0 0.0.255.255 area 0
```

- Interface FE1/0 and FE2/0 fall within this range, FE0/0 does not
- OSPF will be enabled on FE1/0 and FE2/0 and the router will peer with adjacent OSPF routers
- Networks advertised:
 - 10.0.1.0/24
 - 10.0.2.0/24
 - 10.1.0.0/24 is NOT advertised
 - 10.0.0.0/16 is NOT advertised



There are always multiple ways you can configure network commands to achieve the same result. Best practice is to use the most simple and logical way.

- Three different configurations, same result:

```
R1(config-router)# network 10.0.0.0 0.255.255.255 area 0
OR
```

```
R1(config-router)# network 10.1.0.0 0.0.0.255 area 0
```

```
R1(config-router)# network 10.0.1.0 0.0.0.255 area 0
```

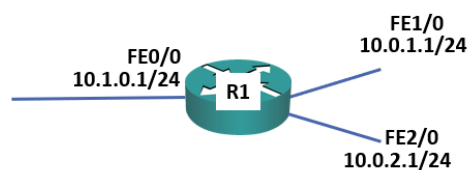
```
R1(config-router)# network 10.0.2.0 0.0.0.255 area 0
```

OR

```
R1(config-router)# network 10.1.0.1 0.0.0.0 area 0
```

```
R1(config-router)# network 10.0.1.1 0.0.0.0 area 0
```

```
R1(config-router)# network 10.0.2.1 0.0.0.0 area 0
```



Show ospf status:

- # show ip protocols – show configuration (works in user mode too!)
- # show ip ospf interface [*ifname*] – show OSPF operations — including number of neighbours and link type — on all interfaces, or on a given interface if optional argument given.
- # show ip ospf neighbour [*ifname*] – show OSPF neighbours on all interfaces, or on a given interface if optional argument given
- #show ip ospf database – show the LSDB
- #show ip ospf rib – show Routing Information Base (RIB)
- #show ip route – show routes
- #show ip route ospf – show only OSPF-learned routes

Passive interface – router advertises routes to the connected subnet but does not send OSPF messages on this interface. Useful when there is no other router on the connected subnet, or when there is an OSPF router belonging to another organisation (eg your ISP) on the link, and you want to advertise the network to your own internal routers but you do not want to advertise your internal networks to the other organisation. To enable:

- (config-router)# passive-interface *ifname*

Alternatively, make all interfaces passive by default and explicitly make some active again:

- (config-router)# passive-interface default
- (config-router)# no passive-interface *ifname* – make ifname active interface again

Default route advertisement. This injects a static default route on the router into OSPF, making it an Autonomous System Boundary Router (ASBR):

- (config-router)# default-information originate – advertise router's default route when it works
- (config-router)# default-information originate always – advertise router's default route whether it works or not

The OSPF cost is automatically derived from the interface bandwidth.

$\text{Cost} = \text{Reference Bandwidth} / \text{Interface Bandwidth}$

The default reference bandwidth is 100 Mbps

T1 link cost defaults to 64 ($100 / 1.544$)

FastEthernet link cost defaults to 1 ($100 / 100$)

GigabitEthernet link cost also defaults to 1 ($100 / 1000$ – rounded up to the lowest possible value 1)

OSPF treats all interfaces of 100 Mbps or faster as equal. FastEthernet, Gigabit Ethernet, 10 Gigabit Ethernet etc. all default to a cost of 1. This can cause undesirable routing in modern networks with high speed links so the default reference bandwidth should be changed on all routers to allow higher speed links to have a better cost.

- (config-router)# auto-config reference-bandwidth *speed* – set the reference speed of the interface to *speed* in **Mbits** to change calculated cost

OSPF metric of a link can be changed in either of the following ways

- (config-if)# ip ospf cost *n* – set the cost of this interface to *n*. Recommended method as it does not affect any other configuration on the router.
- (config-if)# bandwidth *speed* – set the notional speed of the interface to *speed* in **Kbits** to change calculated cost

OSPF Load balancing can happen if multiple routes have the same and lowest metric. It can be enabled (if it isn't by default, depends on hardware) by:

- (config-router) maximum-paths *n* – balance load across at most *n* paths

OSPF does not support unequal-metric load-balancing. For this EIGRP would be needed.

Chapter 21 – OSPF Network Types and Neighbors

5 OSPF Network types:

Name	DR/BDR elections	Manual neighbours	Hello/ Dead timers	Command	Default for
Broadcast	Yes	No	10/40	(config-if)# ip ospf network broadcast	Ethernet, FDDI
Non-broadcast	Yes	Yes	30/120	(config-if)# ip ospf network non-broadcast	X.25, Frame Relay
Point-to-point	No	No	10/40	(config-if)# ip ospf network point-to-point	PPP, HDLC
Point-to-multipoint broadcast	No	No	30/120	(config-if)# ip ospf network point-to-multipoint	
Point-to-multipoint non-broadcast	No	Yes	30/120	(config-if)# ip ospf network point-to-multipoint non-broadcast	

If the network requires manual neighbours, they need to be specified with a **neighbor** command:

- (config-**router**)# neighbor *address*

DR/BDR election: Highest OSPF interface priority wins, second highest becomes the BDR. In case of a tie, highest RID wins. To set OSPF interface priority:

- (config-if) ip ospf priority *n* – set interface OSPF priority to *n* (0-255)

Once a router is elected a DR/BDR, it will stay a DR/BDR until it fails. There is no pre-emption, a router joining the group (or a router changing its priority) will not trigger an election. If a network does not use DR/BDR elections, DRs/BDRs are simply not used and **show ip ospf neighbour** will not show DR/BDR/DROTHER in “State” column.

For neighbour relationship to work:

- Interfaces must be up/up
- ACL must not filter routing protocol messages
- Interfaces must be in same IP subnet
- Interfaces must be authenticated, if authentication is used
- Hello/Dead timers must match
- RIDs must be unique
- Interfaces must be in the same area
- OSPF process must not be shutdown
- **Matching MTU setting**
- **Matching network type**

If the last two requirements are not satisfied routers **will** see each other as neighbours, but OSPF will not work (i.e. LSAs are not exchanged)

OSPF process shutdown:

- (config-router) shutdown

This retains configuration but ceases OSPF activities, clears LSDB, brings down any neighbour relationships and removes OSPF-learned routes from the routing table.

Part VII – IPv6

Chapter 22 – Fundamentals of IP Version 6

Protocols changed for IPv6

- ARP → Neighbour Discovery Protocol
- ICMP → ICMPv6
- OSPFv2 → OSPFv3
- RIP → RIPng
- EIGRP → EIGRPv6
- BGP → MP BGP-4

IPv6 addresses and abbreviation: IPv6 address is 128-bit long (compared to just 32-bit long IPv4 addresses) and is written down as 8 sets of four hex digits called quartets. In order to simplify, addresses are abbreviated. Within each quartet, leading 0s are removed. Quarter 0000 will leave a single 0. If there is a string of two or more all-zero quartets, they can be abbreviated to “::”. Only one such string can be abbreviated.

Chapter 23 – IPv6 Addressing and Subnetting

Address type	First hex digits
Global unicast	Any not otherwise reserved, usually 2 or 3
Unique local	FD
Multicast	FF
Link-local	FE80/10

Global unicast addresses – public IPv6 addresses, publicly routable. Blocks are assigned by IANA to Regional Internet Registries, who in turn dole them out to ISPs which then assign them to companies/subscribers. Global Routing Prefix, decided by the RIR and ISP, determine how large the network (i.e. how many bits are available for subnet and host parts).

Inside a network, hosts can be assigned addresses:

- Statically
- via DHCPv6
- via Stateless Address Autoconfiguration (SLAAC)

Unique local addresses – private IPv6 addresses

Format: **FD**nn:nnnn:nnnn:ssss:hhhh:hhhh:hhhh:hhhh

FD – first two hex digits

nn – pseudo-random Global ID

ss – subnet

hh – interface ID

Global ID can be set to any value, but it is recommended to keep it a pseudo-random number to avoid address conflicts should two networks be merged in the future.

Chapter 24 – Implementing IPv6 Addressing on routers

Enable IPv6 routing:

```
(config)# ipv6 unicast-routing
```

Verifying IPv6 configuration (mimic IPv4 commands):

- # show ipv6 interface brief – show brief IPv6 address info (sans prefix length) for all interfaces
- # show ipv6 interface *ifname* – show fuller IPv6 address info for interface *ifname*
- # show ipv6 route [connected|static|local|...] - show IPv6 routes, optionally filtered by route type

Static addresses

Static unicast (both global unicast of unique local) address configuration:

- (config-if)# ipv6 address *address/prefix-length* – set full address manually
- (config-if)# ipv6 address *prefix/64 eui-64* – generate address from MAC address using EUI-64. Specify just the prefix, leaving the right part empty (::).

EUI-64 – automatically generates the local, 64-bit long, part of IPv6 address from 48-bit MAC address. Given MAC address:

xx:xx:xx:xx:xx:xx

it will generate:

pppp:pppp:pppp:xixx:xxFF:FExx:xxxx

Where: *p* is the prefix, *x* is taken straight from MAC address, *i* is taken from the MAC address with one bit flipped. FFEE is always inserted in the middle.

We flip the 7th bit of MAC address, which changes the second hex digit. One way to do it is to divide hex digits into four groups of four: 0-3, 4-7, 8-B, C-F. Within each group, the lower odd number gets flipped into the upper odd number, the lower even number gets flipped into the upper even number and *vice versa*: the upper odd number into the lower odd number, the upper even number into the lower even number.

To set MAC address:

- (config-if)# mac-address xxxx.xxxx.xxxx

Dynamic addresses

Can use either DHCP:

- (config-if)# ipv6 address dhcp

or Stateless Address Autoconfiguration (SLAAC)

- (config-if)# ipv6 address autoconfig

Link-local address

Link-local addresses are automatically generated for each interface on a router that has at least one other IPv6 unicast address configured. They are used for unicast traffic that is only relevant on the link it is directly connected to and should not be routed (e.g. NDP which is the IPv6 equivalent of ARP). They are also typically used when specifying the next-hop router address in static IPv6 routes.

Link-local address are formed by adding EUI-64 Interface ID to 64 bit prefix of FE80::.

Alternatively, Interface ID can be randomly generated or configured manually.

To configure a link-local address manually:

- (config-if)# ipv6 address *address* link-local

otherwise IOS will use EUI-64 to generate the address.

IOS creates a link-local address for each interface that has at least one other IPv6 unicast address configured. Alternatively, you can use:

- (config-if)# ipv6 enable – enable IPv6 on the interface and create a link-local address, but no unicast addresses (unless separately configured).

Multicast addresses

Always begin with FF. Examples (these mostly mirror the standardised IPv4 multicast addresses in the 224.0.0.0/24 range):

- FF02::1 – All nodes.
- FF02::2 – All IPv6 routers
- FF02::5 – All OSPF routers
- FF02::6 – All Designated Routers
- FF02::9 – All RIPng routers
- FF02:A – All EIGRPv6 routers
- FF02::1:1 – All DHCP Relay Agents

IPv6 does not have broadcast traffic but FF02::1 has basically the same effect.

show ipv6 interface will list multicast groups the address has joined.

There are several address scopes, each with a different first quartet:

Name	First quartet	
Interface-local	FF01	Packets remain within the device
Link-local	FF02	Packets won't be routed across router interfaces
Site-local	FF05	Broader than link-local. Limits of site defined by routers, but should not cross WAN links.
Organisation-local	FF08	Broader than site-local. Limits defined by routers, can encompass a whole organisation/company
Global	FF0E	No limits

Neighbour Discovery Protocol

Replaces IPv4 ARP. Uses a multicast address – solicited node address – to which only the correct host should be subscribed. The address is generated with prefix of **FF02::1:FF00:0/104** (prefix in bold) and the remaining 6 hex digits are taken from the last 6 hex digits of the IPv6 address being discovered.

Loopback and unspecified

Unknown address – all 0s, ::

Loopback - ::1

Anycast

Two (or more) hosts can be configured with the same IPv6 address and provide a service. Routers will then route packets to the nearest host with that address.

For example two routers supplying DNS services are both configured with the IPv6 address 2001:0db8::10, which is advertised in their routing protocol. Other routers will learn paths to both routers and send traffic to the nearest one. If either router fails the network will converge and the service will remain available on the other router.

Chapter 25 – Implementing IPv6 Routing

Static routes

Local routes have /128 prefix length, i.e. they match just the interface's address.

Connected and local routes are **not** created for link-local addresses. They are created only for global unicast and unique local unicast addresses. Routes are removed when an interface fails and are added again when the interface is brought back up.

Routing commands are analogous to ones used with IPv4

Show IPv6 routes:

- # show ipv6 route [static|local|connected|...] - show IPv6 routes, filtered by type if optional argument specified
- # show ipv6 route *address* – show IPv6 routes matching the provided address

Adding static IPv6 routes using global unicast or unique local unicast address for next-hop router on Ethernet link:

- (config)# ipv6 route *address/prefixlen nexthop* – simply specify the next-hop router

Adding static IPv6 routes using link-local address for next-hop router on Ethernet link:

- (config)# ipv6 route *address/prefixlen ifname nexthop* – we need to specify both the next-hop router address **and** the outgoing interface, so that the host know which interface to use to reach the next-hop address.

Adding static IPv6 routes on a serial (point-to-point) link:

- (config)# ipv6 route *address/prefixlen ifname*

In order to add a static default route, use one of the commands above as appropriate for the link type and next-hop address, with ::/0 as the *address/prefixlen*.

For static IPv6 host route, use *address/128* with one of the commands above.

For a floating route (static route acting as a fallback for dynamically discovered routes), set administrative distance to a value higher than that of dynamic routes. To do this, append the administrative distance (e.g. 130) to one of the commands above.

Neighbour Discovery Protocols

Neighbour Discovery Protocol (NDP) performs the tasks of ARP and then some:

- IPv6 to MAC discovery
- detecting routers in the same subnet
- used by SLAAC to detect the subnet and prefix length
- used by Duplicate Address Detection (DAD) to make sure no other host uses the same IPv6 address

A host sends a multicast **Neighbour Solicitation (NS)** message for the target's IPv6 address. The target host responds with **Neighbour Advertisement (NA)** message to the unicast address of the host that sent the NS message. NA messages can also be sent unsolicited to a multicast address FF02::1 (all-nodes) to announce the host's IPv6 and MAC address to everyone.

To show all hosts discovered by NDP:

- # show ipv6 neighbours – analogous to **show mac address-table dynamic**

To discover routers, a host sends a **Router Solicitation (RS)** message to the FF02::2 (all-routers) multicast address. All routers should respond with **Router Advertisement (RA)** message to the soliciting host's unicast address or to FF02::1 (all-hosts) multicast address. The contents of the RA include the router's subnet ID and prefix length as well as its link-local address.

To use SLAAC, the host first uses NDP RS/RA to discover the subnet ID and prefix length and then generates Interface ID (either using EUI-64 or by random) to fill in the rest of the address. It then uses DAD to make sure no other host is using the chosen IPv6 address.

To use DAD, a host sends a NS with its own IPv6 address. If any other host responds, the address is in use and the host needs to select another address.

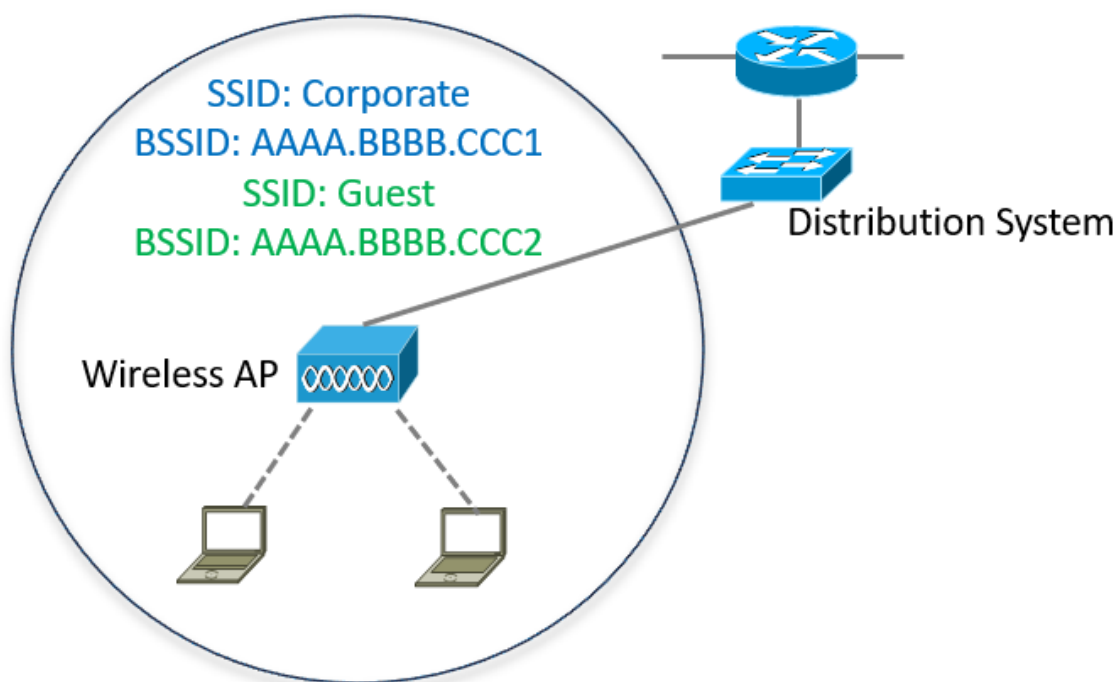
Part VIII – Wireless LANs

Chapter 26 – Fundamentals of Wireless Networks

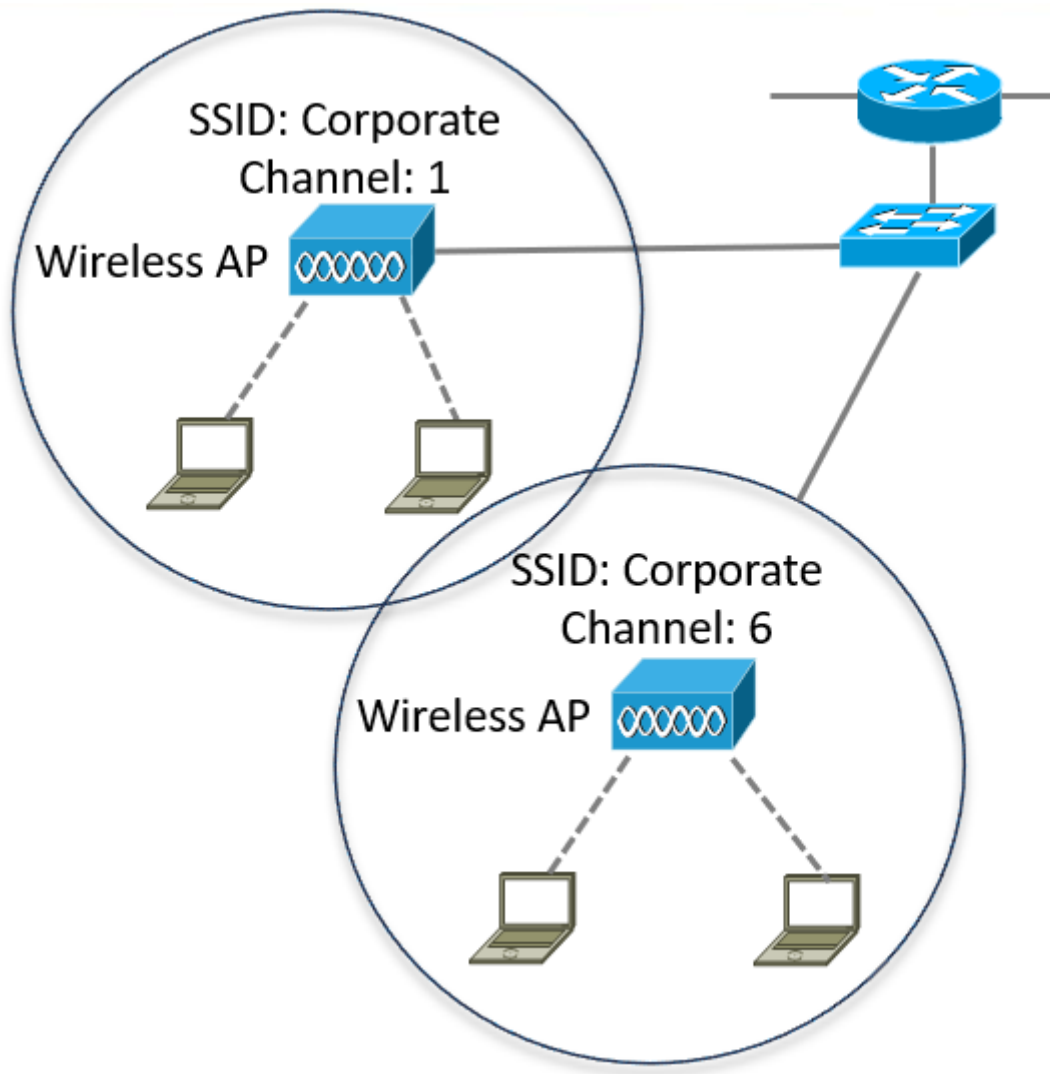
A wireless **access point (AP)** provides a **Basic Service Set (BSS)**, advertising its capabilities and managing who can join. BSS is identified by **BSSID**, based on the access point radio's MAC address.

The **Service Set Identifier (SSID)** is a unique identifier that names the wireless network, such as 'Corporate'. A single Access Point can support multiple SSIDs, for example 'Corporate' and 'Guest'. Different SSIDs can have different security settings and be mapped to different VLANs.

A BSS uses a **Distribution System (DS)**, usually an Ethernet switch, to allow communication with the rest of the network. The access point acts as a layer 2 bridge, forwarding frames from wireless network to wired and the other way round.



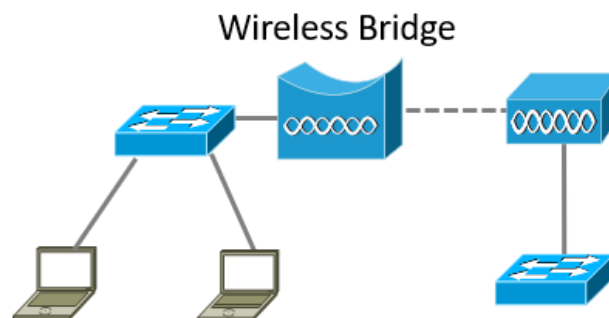
The same **SSID** can be supported across multiple access points in an **Extended Service Set (ESSID)** to give a larger coverage area. Clients can seamlessly **roam** between different **BSS** in the same **ESSID**, associating with one AP after another as they change their physical location.



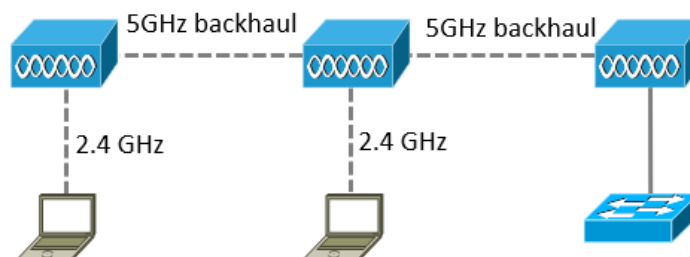
Alternatively, *ad-hoc* wireless networks can operate without a Distribution System. Such a network is called an **Independent Basic Service Set**.

Other wireless terminology includes:

- Repeater – a repeater repeats the signal it receives (preferably using a separate radio on a different channel) to extend the originating AP signal coverage.
- Workgroup bridge (WGB) – acts as a wireless client (STA) and bridges wireless network with wired network. This allows devices (one or more, depending on technology) without a wireless adapter to connect to an AP via the workgroup bridge. Wireless Bridges can be used to connect areas which are not reachable via cable to the network.

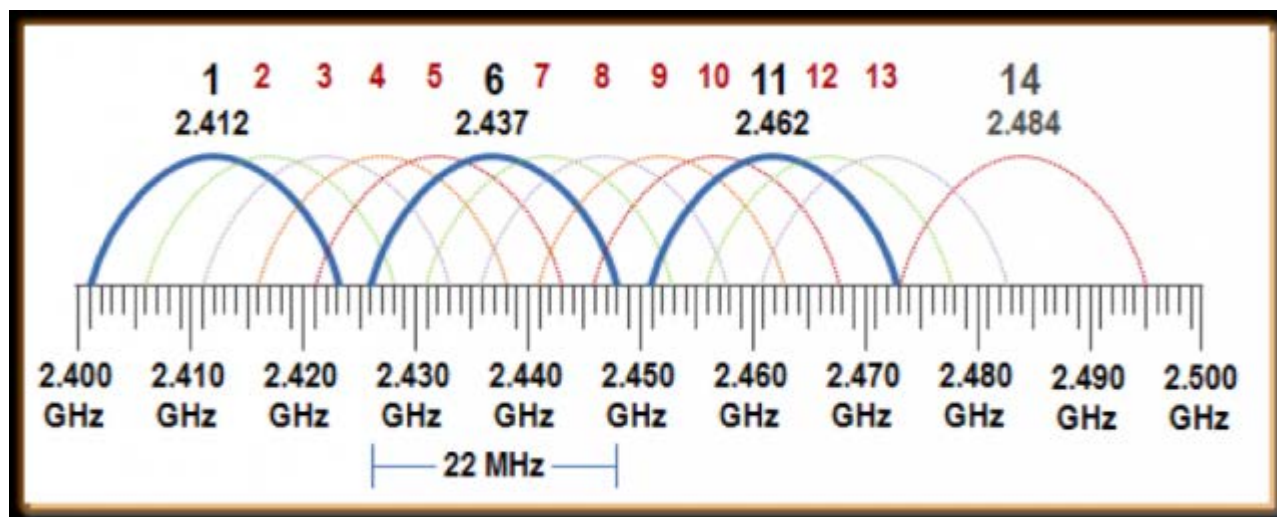


- Outdoor bridge – point-to-point or point-to-multipoint wireless bridge between distant wired networks.
- Mesh network – multiple access points bridged in a daisy-chain. One AP radio is used to serve clients, the other radio connects to a backhaul network.



Wireless uses 2.4 GHz and 5 GHz frequency bands.

2.4 GHz band supports 11-14 channels (depending on location, regulated by governments. 1-11 in the US, 1-13 in most of the rest of the world, 1-14 in Japan, but channel 14 is reserved for 802.11b, which is an old standard), but only 3 of these channels are non-overlapping (each channel is 20 MHz or 22 MHz wide, but there is only 5 MHz between each channel). Access Points with overlapping physical service areas should use non-overlapping channels.



5 GHz can support many more non-overlapping channels, the exact number depends on the set channel width (from 20 MHz to 160 MHz).

Signal on 2.4 GHz band propagates further and penetrates obstacles better, but 5 GHz band is less crowded, resulting in less interference and better speeds.

For roaming, a 10-15% overlap in area served by access points is recommended. With smaller overlap, one risks losing connection to the first access point before successfully roaming to the second. Larger overlap makes interference worse, since with greater AP density it is harder to find non-overlapping channels.

Wireless frame structure – note that the meaning of ADD1, ADD2 and ADD3 depends on the type of the frame, the below is just one example.

Name	FC	DUR	ADD1	ADD2	ADD3	SEQ	ADD4	DATA	FCS
Length (bytes)	2	2	6	6	6	2	6	Variable	4
Desc	Frame Control: wireless protocol, frame type, frame subtype etc.	Transmission timers; Association Identity of client in Power Save Poll control frame	Receiver	Transmitter	BSSID – used for filtering purposes	Fragment number and sequence number of frame	Optional, used only when passing frame between devices (incl. APs) in the distribution system		Frame Check Sequence

Wireless uses half-duplex communication. Wireless clients can transmit and receive, but only one or the other at a time (wired devices on a full-duplex switched network can both transmit and receive at the same time.)

There are three types of wireless frames:

- Data – carries data through the wireless network.
- Control – controls access to wireless medium: Ready-to-Send (RTS), Clear-to-Send (CTS), Acknowledge (ACK), Power Save (PS)
- Management – manages connection between AP and a client: beacons, probe requests and responses, (re-)association requests and responses, authentication requests and responses, de-authentications and announcement traffic

Chapter 27 – Analysing Cisco Wireless Architectures

There are several wireless network architectures available.

In an **autonomous AP architecture** each AP is managed individually. This architecture does not scale well for networks with a lot of APs.

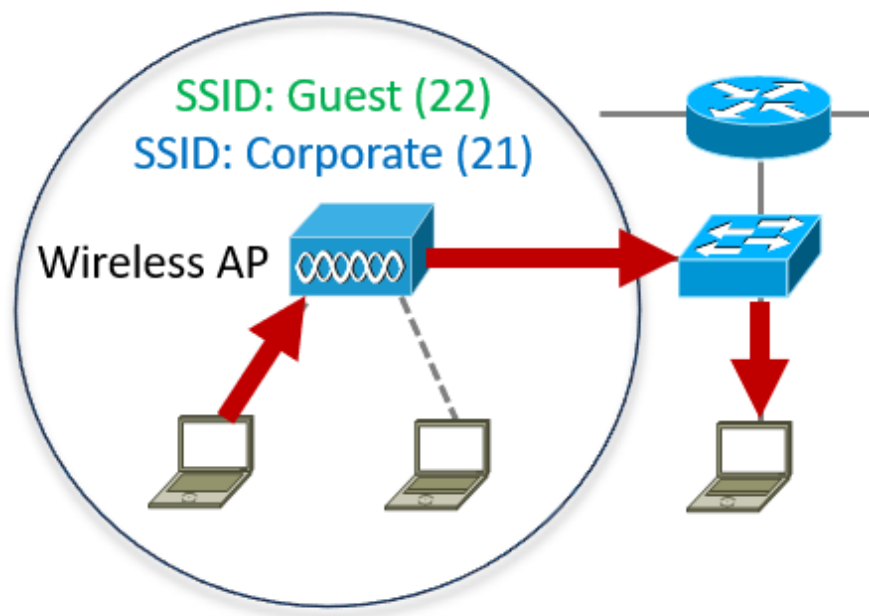
With a **Cloud-based AP architecture** a central management platform (e.g. Cisco Meraki or the deprecated CiscoWorks WLSE) takes care of configuring, monitoring and controlling the APs.

A **split-MAC architecture** is designed to overcome the limitations of **autonomous AP architecture** by splitting the functions of an AP into two devices: **lightweight AP (LAP)** and **Wireless Controller (WLC)**. Lightweight APs take care only of real-time functions – such as transmission, encryption, prioritising packets, responding to beacons and probe requests – with the rest offloaded to a (one or more) central WLC.

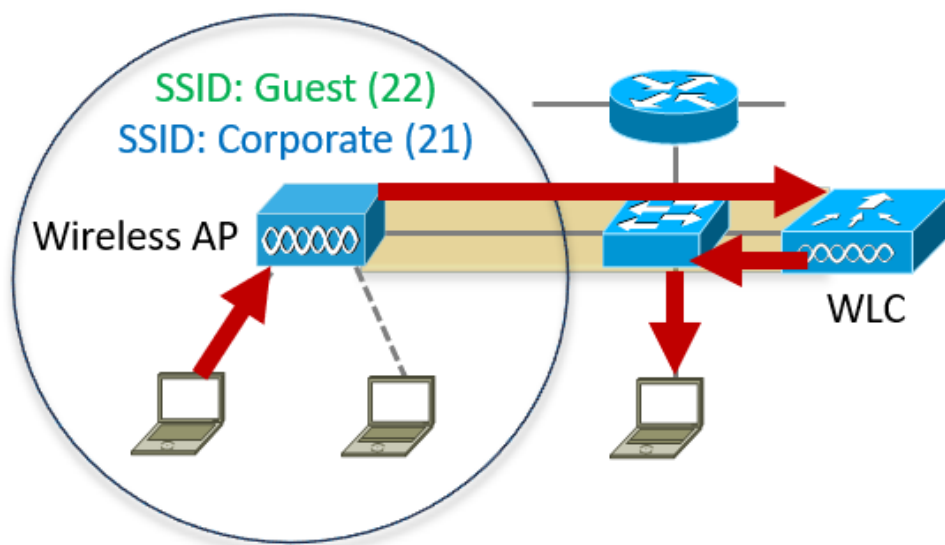
WLCs provide centralised management of the APs and can push their configuration to them, and monitor channels, manage association and roaming, authenticate clients, manage security and QoS.

Traffic from wireless clients flows via the AP through a **Control and Provisioning of Wireless Access Points (CAPWAP)** Layer 3 tunnel to the WLC and only from the WLC does it enter the wired network.

Traffic flow to wired network with Autonomous AP (AP is managed as standalone device and no WLC is used):



Traffic flow to wired network with Lightweight AP (WLC is used). CAPWAP tunnel shown with orange highlight:



The LAPs do not need to be in the same subnet as the WLC. The switch port they are connected to is configured as an access port in a LAP management VLAN. The switch port the WLC is connected to is configured as a trunk port for management VLANs and all the client wireless VLANs.

A CAPWAP tunnel is actually a set of two tunnels: control message tunnel on port 5246 and data tunnel at port 5247. Both run on UDP (since TLS is unsuitable for tunnelling), control messages are encrypted mandatorily, data is encrypted optionally. Datagram Transport Layer Security (DTLS) is used for encryption. AP-WLC authentication is done via X.509 certificates.

The WLC's central management allows for real-time monitoring of the network and RF spectrum and adjusting AP operation for maximum efficiency. This includes dynamic channel assignment, transmit power optimisation, self-healing wireless coverage (increasing transmit power of nearby APs to make up for a loss of an AP), faster client roaming, client load balancing, RF monitoring (e.g. detection of rogue APs), security management and wireless intrusion protection system.

WLC can be deployed in several ways:

Unified deployment involves locating the WLC in a central location, near the core of the network, and can support up to 6000 Aps per WLC.

WLCs can be also run as virtual machines in **cloud-based** deployment. A virtualised WLC typically supports up to 3000 APs.

In smaller networks it makes sense to distribute WLCs at the access layer and embed them in switch stacks there (**embedded** deployment) – an embedded WLC can support up to 200 APs.

The last option is to run WLC on an AP at branch location – other APs will connect to the WLC-AP over CAPWAP. This is called **Mobility Express** and can support up to 100 APs.

LAPs operate in several modes:

- Local – default, offers standard wireless services to clients, scans channels and monitors traffic in spare time
- Bridge – acts as a point-to-point or point-to-multipoint bridge
- FlexConnect – see below.
- Flex+Bridge – FlexConnect+mesh network

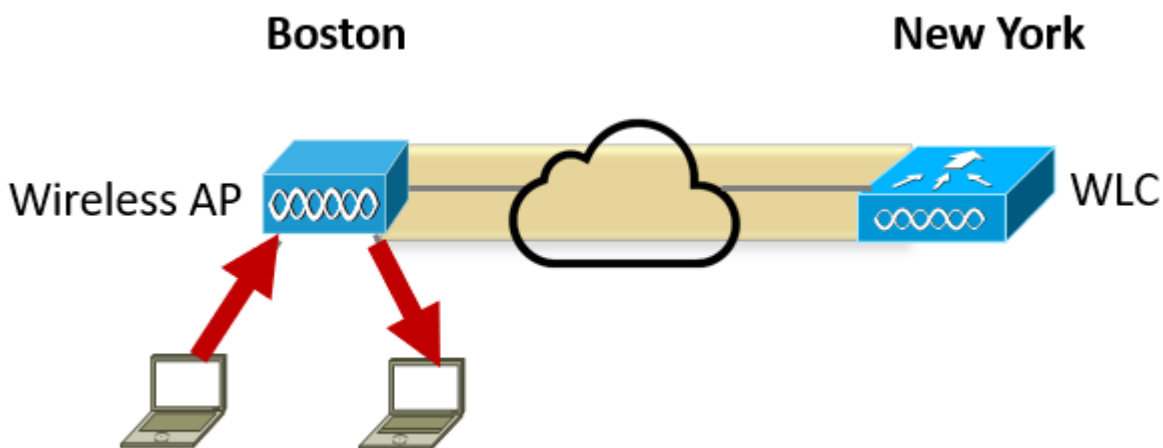
Special modes:

- Monitor – does not transmit, acts as a sensor: scans channels and monitors traffic
- Sniffer – Sniffs wireless network traffic and forwards it to a traffic analyser (such as Wireshark)
- Rogue detector – detects rogue devices
- SE-Connect – Spectrum analysis to detect sources of interference

FlexConnect and FlexConnect ACLs

FlexConnect is a mode that allows LAPs to transition from a Split-MAC (connected mode) to autonomous (stand-alone mode) architecture and back, depending on the status of connection to the WLC and administrator actions.

It is used for APs in remote offices to allow them to have centralised management with a WLC, but wireless client traffic flows to stay in the remote office. Local traffic in the remote office does not have to go over a CAPWAP tunnel to the WLC and back again.



A FlexConnect AP can be in 3 permanent states:

- Authentication-Central/Switch-Central – AP in connected mode, acts as a regular local LAP. Clients authenticate via WLC, traffic is sent via CAPWAP to WLC and switched from there. AP will failover to standalone if the CAPWAP tunnel is lost (ie because the WAN link to the WLC went down).
- Authentication-Central/Switch-Local – AP in connected mode. Clients authenticate via WLC, but traffic is switched locally by the FlexConnect AP
- Authentication-Local/Switch-Local – AP in stand-alone mode, clients authenticate via the AP, traffic is switched locally. AP configuration is controlled by WLC.

FlexConnect ACLs are used to manage access control of locally switched data. They are applied per-AP and per-VLAN. Like traditional ACLs, they have an implicit **deny any** at the end.

Chapter 28 – Securing Wireless Networks

Several basic concepts relate to wireless security. A client needs to be **authenticated** in order to associate with an access point. **Encryption** is separate from authentication – a client can be authenticated but unable to exchange messages because of lack of proper encryption keys.

In fact, WPA 1/2/3 Personal does just that: this setup uses **open authentication**, so that every client can authenticate with the AP, but in order to communicate it needs to negotiate encryption keys using a **pre-shared key** (equivalent to a password).

A third measure is related to **message integrity** – wireless frames may contain **message integrity check (MIC)**, a form of checksum that allows detection of changes in the message. If MIC computed by the sender does not match one computed by the receiver (indicating it has been corrupted or maliciously altered), the frame is dropped.

Authentication methods:

- Open authentication – any client can authenticate.
- Shared key authentication/WEP – used with deprecated and no longer secure **Wired Equivalent Privacy (WEP)**. AP sends a challenge that the client needs to respond to using a matching pre-shared key (a ‘password’ which is configured on the AP and the client). If this client passes, it is authenticated. This authentication method is even less secure than open authentication with WEP because it generates traffic that can be easily used for cryptanalysis.
- 802.1X/EAP – Extensible Authentication Protocol. AP uses open authentication to begin the connection, but the client (in this context known as **supplicant**) then needs to authenticate itself with an **Authentication Server (AS)** before being able to reach the rest of the network.

EAP-based authentication methods:

- LEAP – early Cisco proprietary method, used in tandem with WEP. Client authenticated with a username and password, dynamic (i.e. frequently changed) WEP key was used to overcome WEP weaknesses. However, it soon became clear that WEP is fundamentally broken and no amount of alteration will fix it. LEAP is now as deprecated as WEP is.

- EAP-FAST – Cisco proprietary. The AS Authentication Server generates a PAC Protected Access Credential that needs to be shared (automatically over the air or manually installed) with the supplicant. When connecting, the AS and the supplicant authenticate each other using the PAC and then set up a TLS tunnel which is used to authenticate the end user e.g. via username and password.
- PEAP – AS presents a signed certificate. If the supplicant successfully verifies the certificate, they both set up a tunnel that will be used for authentication using MSCHAPv2 (MS Challenge Authentication Protocol ver. 2) or GTC (Generic Token Card, a token generating one-time passwords).
- EAP-TLS – Same idea as PEAP but the client also requires a certificate. Usually requires building Public Key Infrastructure to manage certificates on clients.
- EAP-TTLS
- EAP-SIM

Encryption and integrity:

- WPA – uses Temporary Key Integrity Protocol, a stopgap measure to enable stronger encryption on WEP-supporting hardware. TKIP is now deprecated and some attacks have been devised. Some WPA devices also support CCMP (see below)
- WPA2 – uses Counter/CBC-MAC protocol (CCMP). AES counter mode for encryption, CBC-MAC for MIC. Still secure, only theoretical attacks devised.
- WPA3 – uses Galois/Counter Mode Protocol (GCMP). AES counter mode for encryption, Galois Message Authentication Code for MIC. Relatively fresh, considered most secure.

Layer 3 security:

- None – Disables Layer 3 security
- IPSec – Allows clients to establish an IPSec tunnel
- VPN Passthrough – Allows clients to establish a VPN tunnel with a defined server(s)
- Web Authentication – Requires users to authenticate via a webpage – common for Guest WLANs
- Web Passthrough – Requires users to visit a webpage and agree to conditions (possibly provide email address) before accessing the Internet – common for Guest WLANs

CCKM: Cisco-proprietary fast-rekeying method that speeds up roaming between access points. Used with 801.1X, it obviates the need to reauthenticate with RADIUS server when roaming.

Chapter 29 – Building a Wireless LAN

To configure an AP, connect via a console cable to setup an IP address first (the same as you would do with a router or switch).

WLC physical ports:

- Console port: same as on router or switch - out-of-band management, initial boot and system recovery, serial connection (9600 baud, 8 data bits, 1 stop bit, by default)
- Service ports: Ethernet port used for out-of-band management, initial boot and system recovery
- Distribution system port (DSP) – normal traffic and management. Handles CAPWAP traffic with APs and forwards it to appropriate VLANs. Connected to a trunk link, multiple ports are usually aggregated in a Link Aggregation Group (LAG), which operates as an EtherChannel **without auto-negotiation**.
- Redundancy port: connected to a second WLC to achieve redundancy, failover and high availability (HA)

WLC virtual interfaces:

- Management IP address: used to access the WLC's web interface and CLI. For management traffic (RADIUS authentication, NTP, syslog). Lightweight APs connect to the management IP address via CAPWAP.
- AP-manager interface: Communication with LAPs via Layer 3 Lightweight Access Point Protocol.
- Redundancy management: Management IP for the backup WLC
- Virtual interface: IP address facing wireless clients when they interact with WLC, e.g. when relaying DHCP requests. For client mobility reasons, every WLC in the same mobility group should have the same Virtual IP address.
- Service Port IP Address: bound to service port
- Dynamic interface: connects VLAN with a WLAN. For example the Corporate WLAN is associated with Corporate VLAN 10

Cisco WLCs support up to 512 WLANs. Each AP supports up to 16 WLANs

Steps to create a WLAN

1. (if using WPA1/2/3-Enterprise) – Setup a RADIUS server
 1. Enter advanced settings
 2. Security→AAA→RADIUS→Authentication
 3. Click “Add...”
 4. Fill in the details (priority, IP address, shared secrets, possibly others)
2. Create a Dynamic Interface
 1. Advanced settings
 2. Controller→Interfaces
 3. Click “Add...”
 4. Enter name, VLAN ID, IP address, subnet mask, gateway and DHCP servers.
3. Create a WLAN
 1. Advanced settings
 2. WLANs→WLANs
 3. Select “Create New” and click “Go”
 4. Input ESSID and profile name
 5. In WLAN configuration setup enter all the familiar details.

WLC CLI interface

WLC also has a CLI interface, but the commands differ from those used on switches and routers.

- # show ap config general *lap-name* – show config of LAP named *lap-name*
- # show ap config global – show global configuration settings for all LAPs
- # show ap core-dump *lap-name* – show the core dump for LAP named *lap-name*
- # show ap crash-file – show a list of core dumps for all LAPs

Book 2

Part I – Access Control Lists

Chapter 1 – Introduction to TCP/IP Transport and Applications

TCP connection establishment (three-way handshake):

- Client sends a TCP segment with a SYN (synchronize) flag
- Server responds with a TCP segment with SYN and ACK (acknowledge) flags
- Client responds with a TCP segment with an ACK flag

TCP four-way termination sequence:

- Host A sends a TCP segment with ACK and FIN (finish) flags
- Host B responds with a TCP segment with an ACK flag
- Host B sends a TCP segment with ACK and FIN flags
- Host A responds with a TCP segment with an ACK flag

RST (reset) flag can also be used to tear down a connection.

For data reliability, TCP uses sequence and acknowledgment numbers. Segments are re-sent if the source does not receive an acknowledgement from the intended receiver.

UDP is unreliable and does not have acknowledgements, it sends segments ‘best effort’ and does not check they reach the destination. UDP is more suitable for real-time traffic such as voice, where re-sent segments would arrive too late for acceptable call quality.

In a three-way TCP handshake, hosts synchronise the starting sequence number (usually to an unguessable non-zero number to prevent TCP sequence prediction attacks). Then the sequence number is incremented by the number of bytes in each segment. Let's say the sequence number is n . Host A then sends a segment with 1200 bytes of data – this segment's sequence number will be $n+1200$.

The sender sends segments until it reaches *TCP window size*. At this point, it will wait for the receiver to reply with a segment with an ACK flag and an *acknowledgment number* equal to *sequence number* of the **next** expected segment.

If the receiver replies instead with an earlier *acknowledgment number* it means the receiver hasn't received a segment with the same *sequence number*. The sender resends the missing segment and wait for another ACK segment with *acknowledgment number* indicating which segment should go next (either another missing one or the one that was originally expected next).

TCP window tells the sender how much data it should send before waiting for a segment with an ACK flag and the appropriate acknowledgment number. *TCP window* size is set by the receiver in its ACK segments – if there are no problems with the transmission, the receiver will gradually increase the window, which reduces overhead and increases speed.

Conversely, if segments go missing, this indicates network problems or congestion that can be potentially solved by reducing the window (retransmissions are costly) and slowing down the transmission. This allows TCP to adapt to bandwidth, reliability and delay of the link used.

The segment size is set at the beginning of connection by setting the MSS (Maximum Segment Size) option field in a SYN segment. MSS is set to the MTU of the link minus 20 bytes (for TCP header). MTU can be auto-discovered.

Chapter 2 – Basic IPv4 Access Control Lists

An ACL identifies traffic based on characteristics of the packet such as source IP address, destination IP address, and port number. The router can take an action based on the result of the ACL.

The original use of ACLs was as a security feature to decide if traffic should be allowed to pass through the router. By default a router will allow all traffic to pass between its interfaces. When ACLs are applied the router identifies traffic and then decides if it will be allowed or not.

ACL's are also used in other software policies when traffic has to be identified, for example:

- Identify traffic to give better service to in a QoS Quality of Service policy
- Identify traffic to translate to a different IP address in a NAT Network Address Translation policy

Access Control Lists are made up of ACE Access Control Entries which are a series of permit or deny rules. Each ACE is written in a separate line.

There are two types of ACLs:

- Standard – match on source address only. Ranges 1-99, 1300-1999
- Extended – match source and destination address, transport protocol and port number.
Ranges 100-199, 2000-2699

Standard Access List example:

```
R1(config)# access-list 1 deny 10.10.10.10
```

```
R1(config)# access-list 1 permit 10.10.10.0 0.0.0.255
```

This will deny all traffic from the 10.10.10.10 host, and permit all traffic from all other hosts in the 10.10.10.0/24 subnet.

The default wildcard mask is 0.0.0.0, meaning an individual host. You need to enter the wildcard mask when specifying an IP subnet.

Remember that ACLs use a wildcard rather than subnet mask.

Extended Access List syntax:

```
R2(config)#  
access-list 100 deny tcp 10.10.30.0 0.0.0.255 gt 49151 10.10.20.1 0.0.0.0 eq 23
```

			Source				Destination			
No.	Action	Protocol	IP	Wildcard	Qual.	Port	IP	Wildcard	Qual.	Port

There is no default wildcard mask for Extended ACLs, you can either use the 'host' keyword or type the 0.0.0.0 wildcard mask if you want to specify an individual host.

Extended Access List example:

```
R1(config)# access-list 100 deny tcp host 10.10.10.10 gt 49151  
10.10.50.10 0.0.0.0 eq 23
```

```
R1(config)# access-list 100 permit tcp 10.10.10.0 0.0.0.255 gt  
49151 10.10.50.10 0.0.0.0 eq telnet
```

This will deny Telnet traffic from the 10.10.10.10 host to the 10.10.50.10 host, and permit Telnet traffic from all other hosts in the 10.10.10.0/24 subnet to the 10.10.50.10 host. (You can enter TCP port 23 or just type 'Telnet' to specify Telnet traffic.)

Entering the source port range 'gt 49151' was optional and would usually be omitted as it's not required, it was shown here for illustrative purposes.

You can reference ACLs by number or by a name. Named ACLs begin with the command 'ip access-list' instead of 'access-list'.

Named Standard ACL example:

```
R1(config)#ip access-list standard Example1
R1(config-std-nacl)#deny 10.10.10.10 0.0.0.0
R1(config-std-nacl)#permit 10.10.10.0 0.0.0.255
```

Named Extended ACL example:

```
R1(config)#ip access-list extended Example2
R1(config-ext-nacl)#deny tcp host 10.10.10.11 host 10.10.50.10 eq
telnet
```

ACLs are applied at the interface level with the Access-Group command (they have no effect until this is done). They can be applied in the inbound or outbound direction. You can have a maximum of one ACL per interface per direction.

You can have both an inbound and an outbound ACL on the same interface, but not 2 inbound or outbound ACL. An interface can have no ACL applied, an inbound ACL only, an outbound ACL only, or ACLs in both directions.

```
R1(config)# interface GigabitEthernet0/1
R1(config)# ip access-group 100 out
R1(config)# ip access-group 101 in
```

The ACL is read by the router from top to bottom. As soon as a rule matches the packet, the permit or deny action is applied and the ACL is not processed any further. The order of rules is important!

This will deny 10.10.10.10 but permit the rest of the 10.10.10.0/24 subnet:

```
R1(config)# access-list 1 deny host 10.10.10.10
R1(config)# access-list 1 permit 10.10.10.0 0.0.0.255
```

This will permit all of the 10.10.10.0/24 subnet including 10.10.10.10 and is a misconfiguration:

```
R1(config)# access-list 1 permit 10.10.10.0 0.0.0.255
R1(config)# access-list 1 deny host 10.10.10.10
```

There is an implicit 'deny any any' rule at the bottom of ACLs. If an ACL is not applied to an interface, all traffic is allowed. If an ACL is applied, all traffic is denied except what is explicitly allowed.

In this example, traffic from 10.10.10.0/24 will be permitted, everything else is denied:

```
R1(config)# access-list 1 permit 10.10.10.0 0.0.0.255
```

If you want to reverse this so that all traffic is permitted except what is explicitly denied, add a permit any statement to the end of the ACL.

In this example, traffic from 10.10.10.0/24 is denied, everything else is permitted.

```
R1(config)# access-list 1 deny 10.10.10.0 0.0.0.255
R1(config)# access-list 1 permit any
```

Chapter 3 – Advanced IPv4 Access Control

source_port and *destination_port* need to use an *operator* that specifies how to match the given port number:

- *eq* – equal to
- *ne* – not equal to
- *lt* – less than
- *gt* – greater than
- *range* – range of port (e.g. **range 1024-11100**)

Common port names (such as **www**) can be used instead of port numbers. Several, comma-separated port numbers can be used in a single ACL rule.

ACEs are automatically numbered in increments of 10

```
R1#sh access-lists 110
Extended IP access list 110
 10 deny tcp host 10.10.10.10 host 10.10.50.10 eq telnet
 20 permit tcp 10.10.10.0 0.0.0.255 host 10.10.50.10 eq telnet
 30 deny tcp host 10.10.20.10 host 10.10.50.10 eq telnet
 40 permit tcp 10.20.10.0 0.0.0.255 host 10.10.50.10 eq telnet
```

Support for injecting ACEs in an existing ACL started in Named ACLs but is also supported in Numbered ACLs now.

```
R1(config)#ip access-list extended 110
R1(config-ext-nacl)#15 deny tcp host 10.10.10.11 host 10.10.50.10 eq telnet

R1#sh access-lists 110
Extended IP access list 110
 10 deny tcp host 10.10.10.10 host 10.10.50.10 eq telnet
 15 deny tcp host 10.10.10.11 host 10.10.50.10 eq telnet
 20 permit tcp 10.10.10.0 0.0.0.255 host 10.10.50.10 eq telnet
 30 deny tcp host 10.10.20.10 host 10.10.50.10 eq telnet
 40 permit tcp 10.20.10.0 0.0.0.255 host 10.10.50.10 eq telnet
```

Part II – Security Services

Chapter 4 – Security Architectures

Vulnerability→Exploit→Threat→Attack

A vulnerability is a weakness (misconfiguration, software bug) in a system that could potentially be used to compromise it. An exploit is a way to use such a vulnerability. A threat is an exploit in the hands of someone who can use it effectively. An attack is a threat being realised.

Typology of attacks:

- Reconnaissance attack – discovering IP addresses and ranges in use, services available etc. This is often a prelude to a more intrusive attack.
- Spoofing attacks – e.g. IP spoofing and reflection attack, MAC spoofing.
- Denial-of-service – exhausting a host's resources to make it unable to handle legitimate requests. Example – TCP SYN flood.
- Reflection attack – e.g. spoofing source IP address to generate unexpected traffic.
- Amplification attack – leveraging protocol characteristics to generate unexpected traffic in volumes much larger than generated by attacker himself.
- Man-in-the-middle attack – acting as an intermediary to intercept and possibly modify communication between two hosts.
- Buffer overflow attacks – can allow running of malicious code with permissions of attacked service.
- Privilege elevation – access a target with a low privilege level user account and then escalate to admin access.
- Malware - software that is specifically designed to disrupt, damage, or gain unauthorized access to a computer system.
- Human attacks AKA social engineering, eg phoning a user and tricking them into revealing their password.
- Password attack – dictionary, brute-force, password reuse.

AAA – Authentication, Authorisation, Accounting

Authentication – verifying the identity of a user

Authorisation – deciding on user's permissions

Accounting – recording of user's actions

Two common AAA servers:

- TACACS+ - Cisco proprietary, TCP on port 49
- RADIUS – Standards-based, traffic optionally encrypted, UDP on ports 1812 and 1813

TACACS+ is often used to provide centralised management of access to the CLI on network devices, as it supports granular permissions. RADIUS is often used to provide centralised management of normal user access to internal services eg a remote access VPN.

Network devices can use AAA servers to authenticate users, decide on their permissions and record their actions.

Chapter 5 – Securing Network Devices

Local passwords

Authentication can be handled either locally on each device (please refer to Book 1, Chapter 6) or via a centralised AAA server, with the latter being the preferred method. With local authentication, passwords are stored in cleartext by default, which is insecure. It is possible to encrypt all user passwords on the device with a single command:

- (config)# service password-encryption

When encrypted, passwords will show up with “5” after the **password** command in show running-config.

Instead of encrypting passwords, you should always hash them. This is done for the enable password when you use the **enable secret** command. NOTE: if both **enable secret** and **enable password** are configured, the latter will be ignored. **enable password** does not hash the password, always use **enable secret**.

By default, Cisco IOS uses MD5 hashes, but these are becoming progressively weaker over time. As computational power increases and vulnerabilities are being discovered, other choices, both of which use SHA-256, are better:

- # enable algorithm-type sha256 secret *password* – use Type 8 PBKDF2 hash

- # enable algorithm-type script secret *password* – use Type 9 Script hash

Alternatively, it is possible to specify the hash value directly:

- # enable *n hash* – where *n* is the numerical ID of the hashing algorithm and *hash* is the hash value.

Table of available hash algorithms

Numerical ID	Name	Notes
0	NA	Cleartext
4	NA	PSIRT hash, Easily cracked, disabled
5	md5	MD5 hash, default, somewhat vulnerable
7	NA	Vigenere cipher, Old, easily cracked, disabled
8	sha256	PBKDF2 hash, recommended
9	script	Script hash, recommended

The same applies to local usernames, just replace **enable** with **username *user***.

Using access lists, it is possible to limit where a user can login from via Telnet/SSH.

- (config-line)# access-class *ID* in – use the access list numbered *ID* to permit/deny login attempts.

IOS supports privilege level from 0-15. To specify a password for a particular privilege level:

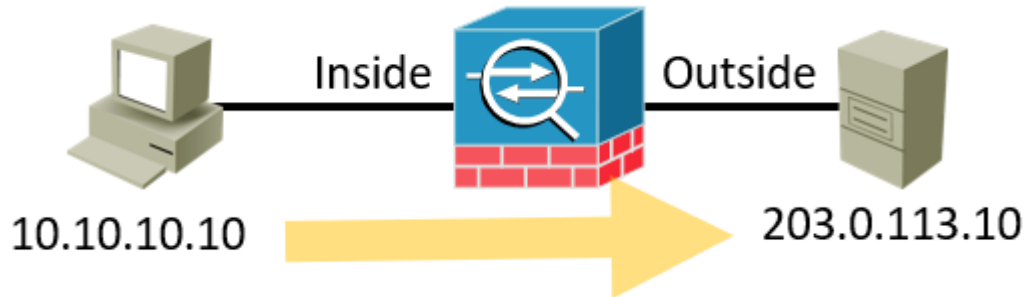
- # enable secret level *n password*

Firewalls and Intrusion Prevention Systems

Stateful firewalls maintain a connection table which tracks the two-way ‘state’ of traffic passing through the firewall. Return traffic is permitted by default.

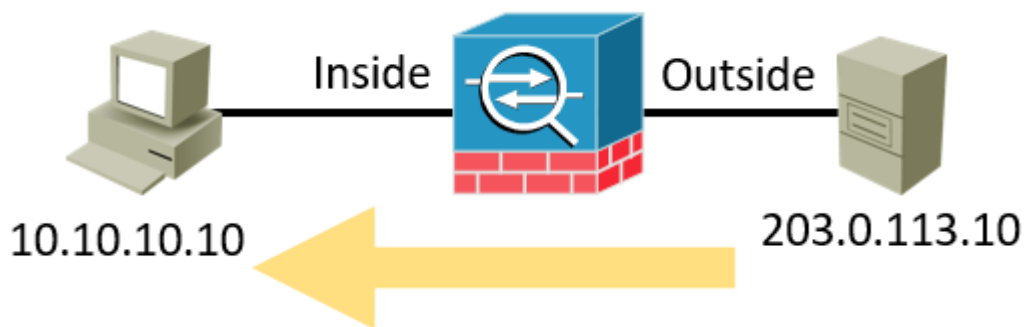
Firewall rules example:

- Deny all traffic from outside to inside
- Permit outbound web traffic from 10.10.10.0/24

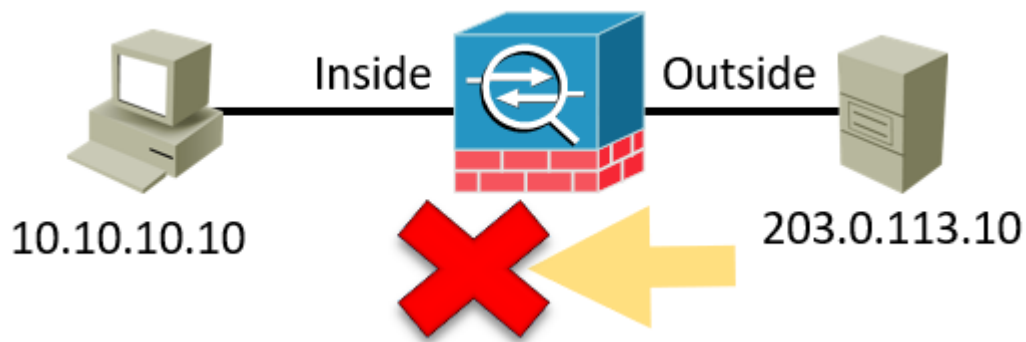


In the image above, the traffic is allowed by 'Permit outbound web traffic from 10.10.10.0/24' rule

- Connection table: 10.10.10.10:49160 > 203.0.113.10:80

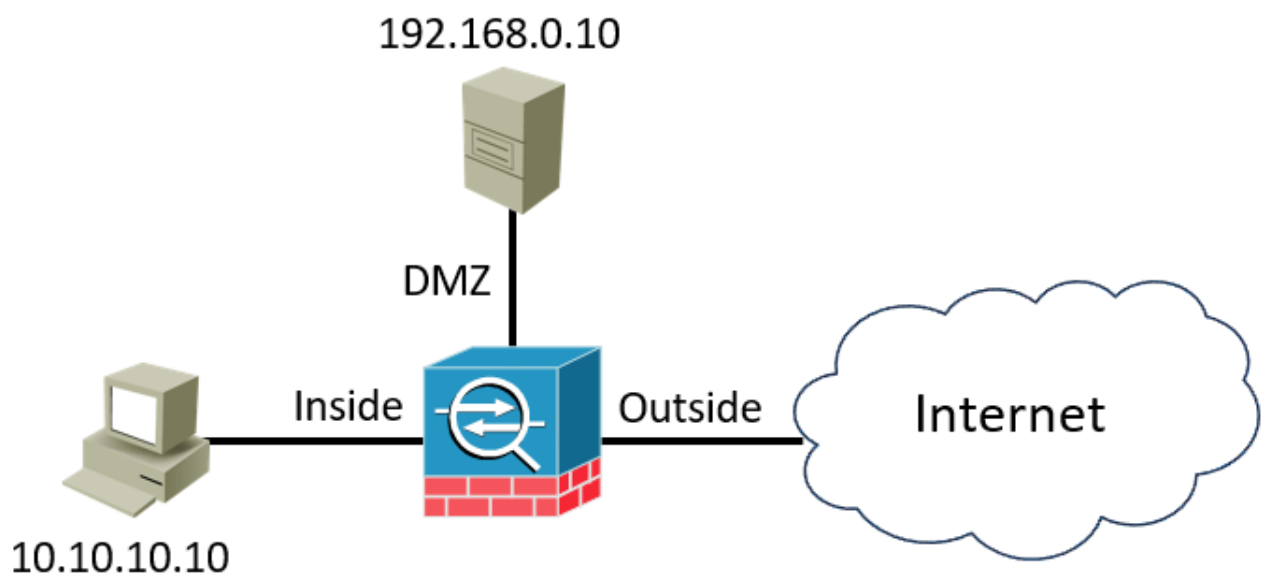


Traffic from 203.0.113.10:80 > 10.10.10.10:49160 is permitted because it is valid return traffic for a connection in the connection table. This overrides the 'Deny all traffic from outside to inside' rule.



In this example the connection has not been initiated from the host on the inside. Traffic from 203.0.113.10:80 > 10.10.10.10:49160 is dropped according to the 'deny all traffic from outside to inside' rule.

A zone is a concept used by firewalls to classify hosts, with each interface being assigned to a zone. At a minimum, there is an **inside zone** and **outside zone**, with traffic from the outside zone into the inside zone typically being blocked (apart from return traffic). A DMZ (demilitarized zone) is often used to give limited access to servers that need to be accessible from the public Internet.



Intrusion Prevention Systems (IPS) complement firewalls and match incoming traffic against a database of exploits in order to detect incoming attacks.

Next-generation firewalls (NGFW) can perform deep packet inspection, inspecting packets up to their layer 7 headers using **Application Visibility and Control** to classify traffic with more granularity than using port numbers. They can run several security services in tandem (Advanced Malware Protection), and filter URLs based on a database of domain scores.

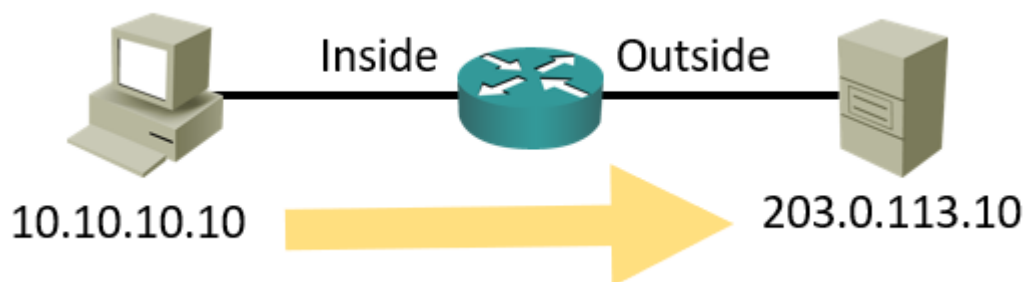
NGFW can also integrate Next-Generation IPS (NGIPS). NGIPS are better at filtering and prioritising events in their reports, using their awareness of hosts in the network (OS, services enabled, patterns of traffic) and reputation-based filtering. Cisco Talos Intelligence Group maintains databases used for reputation filtering by NGFW and NGIPS.

An ACL Access Control List on a router (as covered in Chapter 2) is a packet filter. Unlike stateful firewalls, packet filters **do not** maintain a connection table. They affect traffic in one direction only and do not track the state of two-way connections going through the router.

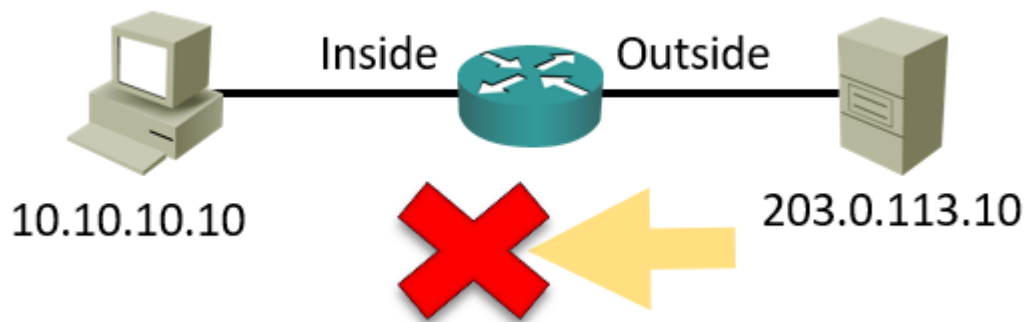
If you have an ACL applied on the way out only, the return traffic will be allowed because all traffic is allowed when an ACL is not applied. If you have ACLs applied in both directions, you will need explicit entries to allow both the outbound and the return traffic.

Access Control List example:

- Inbound ACL on outside interface: Deny all traffic
- Inbound ACL on inside interface: Permit web traffic from 10.10.10.0/24



Inbound ACL on inside interface: Permit web traffic from 10.10.10.0/24 allows traffic out to the web server. The connection is not tracked in a connection table.



Return traffic from 203.0.113.10:80 > 10.10.10.10:49160 is dropped because of Inbound ACL on outside interface: Deny all traffic.

To allow the return traffic you need to remove the 'deny all traffic from outside to inside' ACL on the outside interface, or add 'permit tcp any eq 80 10.10.10.0 0.0.0.255 range 49152 65535'.

Neither is a secure option for a router connected to the Internet.

ACL packet filters on routers can add to an overall 'defence in depth' strategy. Standard practice is to use firewalls on major security boundaries, and augment this with internal ACLs.

You can configure a router as a stateful firewall with the 'IOS Firewall' feature set. This uses different commands than ACLs.

Chapter 6 – Implementing Switch Port Security

Port security monitors and restricts Layer 2 traffic on a switch on a per-port basis. When enabled, it maintains a list of source MAC addresses authorized to send traffic through a given port, and limits the number of source MACs that can use the port. It can limit access to a specified set of MAC addresses or use a mixed solution: some addresses in the allowed pool are set statically, others learned dynamically.

Violations occur when a port encounters a source MAC address that is not in its list of learned or statically configured allowed addresses and when the port has reached its maximum of allowed addresses. It can then undertake a specified action.

To configure port security:

- (config-if)# switchport mode {access | trunk} – trunking/access mode has to be explicitly set (no DTP auto-negotiation)
- (config-if)# switchport port-security – enable port security on a given interface
- (config-if)# switchport port-security maximum *n* – allow up to *n* source MAC addresses on the interface (default: 1)
- (config-if)# switchport port-security violation {protect | restrict | shutdown} – set the action port security should undertake when a violation occurs (default: shutdown. See below for the 3 options)
- (config-if)# switchport port-security mac-address *macaddress* – Statically define an allowed source MAC address
- (config-if)# switchport port-security mac-address sticky – “sticky learn” allowed source MAC. In sticky learn mode, port security learns source MAC addresses and add them to the list of allowed MAC addresses until it reaches its limit. Sticky learned MAC addresses are stored in the **running-config**

Port Security configuration example (allow a maximum of two MAC addresses and manually add MAC address 0000.1111.1111 to the configuration):

```
SW1(config-if)#switchport port-security
SW1(config-if)#switchport port-security maximum 2
SW1(config-if)#switchport port-security mac-address 0000.1111.1111
```

NOTE: Port security can be also enabled for voice ports and EtherChannels. On a voice port, the maximum should typically be set to two to allow the IP phone and PC behind it. When applying port security to an EtherChannel, use the port-channel interface rather than the physical interface.

To show port security configuration and state:

- #show port-security interface *ifname* – show port security configuration and state of an interface, including the number of violations and source (MAC and VLAN) of last violation
- #show mac address-table secure – list MAC addresses associated with ports that have port-security enabled
- #show mac address-table static – the above plus other statically configured MAC addresses.

Port security violation modes defines how port security should act when it detects a violation.

- shutdown – put the interface in an err-disabled state (show interfaces, show interfaces status), set port security state to secure-down (show port-security). Possible to configure to recover automatically after a timeout
- restrict – drop the offending traffic, log the violation, increment the violation counter, keep the interface up
- protect – drop the offending traffic, do not log the violation, do not increment the violation counter, keep the interface up

To recover from an err-disabled state:

- (config-if) shutdown
- (config-if) no shutdown

To automatically recover from a shutdown (note that this is a global command):

- (config) errdisable recovery cause psecure-violation – enable automatic recovery
- (config) errdisable recovery interval *seconds* – set the wait period between violation and recovery.

Chapter 7 – Implementing DHCP

DHCP Dynamic Host Control Protocol allows you to automate allocation of IP addresses to the hosts in your network. This saves you the time of configuring all of them individually. DHCP is typically used to assign IP addresses to standard user PCs, while servers and network devices are manually configured.

DHCP basic messages (in order):

- Discover – from client to server, broadcast
- Offer – from server to client, unicast at L2, broadcast at L3
- Request – from client to server, broadcast
- Acknowledgment – from server to client, unicast at L2, broadcast at L3

DHCP is often centralised in large networks, there are not separate DHCP servers for every IP subnet. Instead, DHCP messages will be passed from clients to the server and back by a DHCP relay, usually the default gateway router on the subnet. This is required because routers do not forward broadcasts by default, the DHCP requests would never reach the DHCP server without configuring it.

DHCP relay sets the **giaddr** field to indicate which subnet the request is from when it passes the DHCP request on to the DHCP server.

For example if a client on the 10.10.10.0/24 subnet sends a DHCP request, its default gateway router acting as a DHCP relay will pass this information on to the DHCP server, including which IP subnet the request came from.

To set up a router to act as a DHCP relay:

- (config-if)# ip helper-address *server-ip* – listen on the given interface for DHCP messages and relay them to *server-ip*

Network devices like routers and switches are usually configured with manual IP addresses, but they can also be configured as a DHCP client. The most common reason for this is if they are connected to the Internet and receiving their address on that interface via DHCP from the ISP.

- (config-if)# ip address dhcp
- #show dhcp lease

When DHCP provides a default gateway, it will be inserted into the routing table as a static route with administrative metric of 254.

Verifying interface settings on Windows:

- ipconfig /all

Verifying interface settings on Linux:

- ifconfig – deprecated command
- ip addr

Verifying interface settings on MacOS:

- networksetup -getinfo *interface* – list IP settings
- networksetup -getdnsservers – list DNS servers used

Setting up a DHCP server

A router can act as a DHCP server:

- (config)# ip dhcp pool *pool-name* – enter DHCP server configuration context for a pool named *pool-name*
- (dhcp-config)# network *address subnet-mask* – give out address from the specified subnet – the address and mask need to match that of the interface on which the DHCP server is supposed to listen
- (config)# ip dhcp excluded-address *low-address [high-address]* – do not offer the specified address or range of addresses (if *high-address* specified)

Additionally, one should specify several DHCP options to let clients know more about the network:

- (dhcp-config)# default-router *address [secondary-address]* – define default gateway for clients to use. Multiple *secondary-address* can be specified
- (dhcp-config)# dns-server *address [secondary-address]* – define default DNS server for clients to use. Multiple *secondary-address* can be specified

DHCP Server configuration example:

```
R1(config)#ip dhcp excluded-address 10.10.10.1 10.10.10.10
R1(config)#ip dhcp pool Example
R1(dhcp-config)#default-router 10.10.10.1
R1(dhcp-config)#dns-server 10.10.20.10
R1(dhcp-config)#network 10.10.10.0 255.255.255.0
```

Chapter 8 – DHCP Snooping and ARP Inspection

DHCP Snooping

DHCP Spoofing can be used by rogue DHCP servers to provide clients with the attacker's machine as a bogus default gateway in order to route their traffic via the attacker (Man-in-the-middle attack, MITM). It can also be used as a Denial of Service attack to give invalid IP address settings to clients. This can be done maliciously by an attacker, or it can also easily happen by accident if any device running DHCP services is added to the network.

DHCP Snooping classifies ports as trusted or untrusted, and analyses traffic on untrusted ports to detect DHCP Spoofing attacks.

As part of the traffic analysis, DHCP snooping keeps a binding table database which tracks which MAC address was assigned which IP address by DHCP.

Ports connected to DHCP servers, and ports connecting switches to each other (to allow traffic to pass between clients and the DHCP server) should be configured as trusted.

- on trusted ports, DHCP messages are not filtered
- on untrusted ports messages from DHCP servers are always discarded
- on untrusted ports messages from DHCP clients are monitored for potential attacks:
 - DISCOVER and REQUEST messages are checked for MAC address consistency between Ethernet frame and DHCP message
 - RELEASE and DECLINE messages are checked against the DHCP Snooping binding table to see if the IP and interface match. If a host has requested an IP address via one interface and the switch later receives a RELEASE/DECLINE message for the same IP but on different interface, it will filter the latter message out, because it is likely from a malicious host that tries to trick the DHCP server into ending lease of IP address for a legitimate host.

To configure DHCP Snooping

- (config)# ip dhcp snooping – enable DHCP Snooping globally
- (config)# ip dhcp snooping vlan *n* – enable DHCP Snooping on VLAN *N*
- (config)# no ip dhcp snooping information option – This disables adding option 82 (Relay Agent Information) to DHCP messages passing through the switch. Option 82 should only be added when the switch acts as a relay agent, so we disable it with this command.
- (config-if)# ip dhcp snooping trust – trust this interface
- (config-if)# ip dhcp snooping untrust – do not trust this interface

DHCP Snooping configuration example:

```
Sw1(config)# ip dhcp snooping
```

```
Sw1(config)# ip dhcp snooping vlan 10
```

On ports connected to a DHCP server or another switch:

```
Sw1(config-if)# ip dhcp snooping trust
```

To show the binding table:

- `#show ip dhcp snooping binding`

DHCP Snooping can be relatively CPU-intensive and thus it is advisable to rate limit incoming DHCP messages per-interface to prevent DoS attacks:

- `(config-if)# ip dhcp snooping limit rate n` – limit the number of incoming DHCP messages to *n* per second, put the interface in err-disabled state if this limit is exceeded.
- `(config)# errdisable recovery cause dhcp-rate-limit` – enable automatic recovery from err-disabled state due to DHCP rate limit
- `(config)# errdisable recovery interval n` – automatically recover after *n* seconds (note: this covers all cases of err-disable, including port security related ones.)

Dynamic ARP Inspection (DAI)

With ARP Poisoning attacks, an attacker tricks hosts into believing an IP address is associated with the attacker's MAC address rather than the legitimate target's MAC address. This can be used as a Man-In-The-Middle or Denial of Service attack. The attacker sends gratuitous ARP messages with its own MAC address and the victim's IP address. If the attack succeeds, other hosts will send traffic destined to the victim to the attacker instead.

Dynamic ARP Inspection prevents ARP poisoning attacks from being successful. DAI utilises the DHCP binding table created by DHCP snooping, which has the list of which IP address was assigned to which MAC address by DHCP. Static entries can be added for hosts not using DHCP.

If a given host has sent DHCP messages on one port, it should remain there and all ARP messages should go via the same port. ARP messages going via other ports are dropped, unless these ports are trusted. Additionally and optionally, DAI can validate source and destination MAC and IP addresses.

There will not be an entry in the DHCP binding table for any hosts which have a static (non DHCP assigned) IP address. To prevent DAI from dropping traffic from these hosts, static entries must be added or the ports connected to them configured as trusted in DAI.

Configuration largely mirrors that of DHCP Snooping:

- (config) ip arp inspection vlan *n* – enable DAI on VLAN *n*
- (config-if) ip arp inspection trust – trust this port
- (config) ip arp inspection validate [dst-mac] [src-mac] [ip] – enable one or more extra validation steps.

DAI configuration example (configure DHCP Snooping first):

```
Sw1(config)#ip arp inspection vlan 10
```

Configure ports connected to hosts with static IP addresses as trusted:

```
Sw1(config-if)#ip arp inspection trust
```

Show DAI state and statistics:

- #show ip arp inspection statistics

DAI can be CPU intensive and thus can be rate limited, similar to DHCP Snooping:

- (config-if) ip arp inspection limit rate *n* [burst interval *sec*] – Limit the number of ARP messages to *n* per *sec* seconds (by default, rate is set to 15, burst interval is set to 1 second)
- (config-if) ip arp inspection limit rate none – disable rate limits
- (config) errdisable recovery cause arp-inspection – enable automatic recovery
- (config) errdisable recovery interval *n* – automatically recover after *n* seconds (note: this covers all cases of err-disable, including port security related ones.)

Part III – IP Services

Chapter 9 – Device Management Protocols

Syslog

There are 4 targets for logging messages on IOS:

- Console – displayed for users logged in via console port.
- Monitor – for users logged in via Telnet/SSH. Each user needs to execute **terminal monitor** EXEC command after logging in to actually receive these messages.
- Buffer – logged to RAM, can be viewed with **show logging** EXEC command.
- Syslog/host – sent to a remote syslog server.

To enable logging:

- (config)# logging {console | monitor | buffered} – enable one of the three targets
- (config)# logging host {*address / hostname*} – log messages to remote syslog server at *address* or *hostname*.

To filter message based on priority:

- (config)# logging {console | monitor | buffered | trap} {*level_name | level_number*} – log messages up to and including the specified level (see below for numbers and names) to a given target. ‘Debug’ displays everything. Note that **host** is replaced by **trap** in this command

Log messages severity levels:

- 1) Emergency
- 2) Alert
- 3) Critical
- 4) Error
- 5) Warning
- 6) Notification
- 7) Informational
- 8) Debug

Modifying log message format:

- (config)# [no] service timestamp – enable/disable timestamps
- (config)# [no] service sequence-numbers – enable/disable sequence numbers

Show logging configuration:

- #show logging

Debugging can be enabled for select services. When on, it will generate log messages of severity 7, which will be sent only to those facilities that have this priority enabled. Debugging is enabled with EXEC command **debug** followed by appropriate parameters, similar to those used by **show** command.

Network Time Protocol (NTP)

Time zone settings:

- (config)# clock timezone *timezone* *hour_offset* [*minutes_offset*] – *timezone* is just a label, the actual timezone is set as the offset from UTC
- (config)# clock summer-time *timezone* [recurring] – again, *timezone* is just a label. **recurring** parameter enables automatic daylight saving time adjustments

To manually set the time:

- # clock set *HH:MM:SS Day Month Year* – set clock to given time and date

NTP is used to synchronise clocks. NTP servers use very precise clocks to offer time to other hosts. These hosts can in turn synchronise with further hosts. NTP uses strata number to indicate accuracy of the host's clock – the lower the number, the more accurate the clock. Hosts using own clocks as sources set the strata number themselves, hosts getting the time from others set the strata number to one higher than their source.

When Cisco devices are set as NTP clients (getting time from NTP servers), they automatically can act as NTP servers (offering time to NTP clients). This mode is called static client mode.

- (config)# ntp server {*address* / *hostname*} – get time from NTP server at *address/hostname*

Note that the 'ntp server' command configures the router as an NTP *client*.

Alternatively, a device can act as a broadcast client, listening on an interface to NTP messages broadcast from any server:

- (config-if)# ntp broadcast client

A device can also act solely as NTP server, using its own clock to offer time to others:

- (config)# ntp master [*stratum*] – default stratum 8

ntp server and **ntp master** can be used simultaneously for redundancy, in which case the host will act as NTP client (as per **ntp server** command) and fall back to its own clock (as per **ntp master** command) if the connection fails. It will keep offering time to other clients no matter which source it uses.

Finally, a device can act in a symmetric active mode, mutually synchronising with its peer:

- (config)# ntp peer {*address / hostname*} – synchronise with NTP peer at *address/hostname*

In real world deployments, network devices are typically configured with only the ‘ntp server’ command to synchronize their clocks with an accurate NTP server which is often on the Internet.

To show time and settings

- # show clock
- # show ntp status
- # show ntp associations – show NTP servers to which the client is connected

NTP can be configured on a loopback interface. This is a virtual interface that is always up (unless manually shutdown). The network administrator should advertise the IP address of the loopback interface in the network’s routing protocol, ensuring that a path will be available to it when any physical interface on the device is up.

A loopback interface is preferable to using the IP address on a physical interface for services on the router, because if the physical interface goes down (for example because of link failure) its IP address is not reachable. If a physical interface is dead, the IP address on that interface is also dead - it does not failover to being reachable via another interface. Clients will fail to connect to the failed physical interface IP address, but will still be able to connect to the loopback IP via any available path.

To configure a loopback interface:

- (config) interface loopback 0 – enter loopback interface configuration context
- (config-if) ip address *address netmask* – configure loopback's IP address
- (config) ntp source loopback 0 – use loopback interface to send NTP packets (NTP will listen on all interfaces, physical and loopback, regardless)

NTP authentication using MD5 keys can be enabled to provide source verification. On NTP client:

- (config)# ntp authenticate – enable NTP authentication
- (config)# ntp authentication-key *key-number* md5 *key* – Define an MD5 key with a value of *key* (up to 32 characters) and ID of *key-number* (32-bit field)
- (config)# ntp trusted-key *key-number* – Treat key with ID *key-number* as trusted
- (config)# ntp server {*address/hostname*} key *key-number* – Get time from specified NTP server, provided it signs its messages with the key with ID *key-number*

On an NTP server, only the **ntp authenticate** and **ntp authentication-key** commands are needed.

CDP and LLDP

Cisco Discovery Protocol (CDP) and Link-layer Discovery Protocol (LLDP) are protocols which serve the same purpose – discovery of basic information on neighbouring network devices. CDP is Cisco-proprietary and for Cisco devices. It offers more granular information and is enabled by default on IOS, unlike LLDP.

LLDP was developed as an open standard by other vendors to offer similar capabilities to CDP.

CDP can identify the model (“platform”) and software version of discovered (Cisco) devices, while LLDP can only identify their software version. LLDP distinguishes between installed and enabled capabilities, while CDP does not. CDP conveys VTP information, LLDP does not. Both protocols operate on Layer 2 and discover only directly connected neighbours.

For CDP:

- # show cdp neighbors [*ifname*] – list basic information (without IP addresses or software versions shown) on all neighbours, filtered by interface is optional argument supplied.
- # show cdp neighbors detail – detailed information on all neighbours.
- # show cdp entry *name* – show detailed information on the selected neighbour.
- # show cdp – show CDP global status (enabled/disabled, timers etc.)
- # show cdp interface *ifname* – show CDP status on a given interface
- # show cdp traffic – show global CDP statistics
- (config)# [no] cdp run – enable/disable CDP globally
- (config-if)# [no] cdp enable – enable/disable CDP on a given interface.
- (config)# cdp timer *seconds* – set frequency of CDP message transmission (default: **60** seconds)
- (config)# cdp holdtime *seconds* – set time to wait since last message before considering a neighbour to have failed. (default: **180** seconds)

LLDP uses the same syntax, just replace **cdp** with **lldp**. The only difference is in enabling/disabling the protocol – it is disabled by default and it is possible to configure receiving and sending LLDP messages separately (so that the host can silently gather information on neighbours) on particular interfaces:

- (config)# [no] lldp run – enable/disable LLDP globally
- (config-if)# [no] lldp transmit – enable/disable LLDP message transmission on a given interface
- (config-if)# [no] lldp receive – enable/disable LLDP message receipt on a given interface

LLDP timers:

- (config)# lldp timer *seconds* – set frequency of CDP message transmission (default: **30** second)
- (config)# lldp holdtime *seconds* – set time to wait since last message before considering a neighbour to have failed. (default: **120** seconds)

When parsing LLDP/CDP output, note that “Local interface” and “Port ID” are the interfaces that form the link: “Local interface” refers to the interface on the host doing the query, while “Port ID” refers to the interface on the host being queried.

Chapter 10 – Network Address Translation

Network Address Translation (NAT) is a set of techniques that changes the source and/or destination IP address of a packet as it passes between interfaces on a router or firewall. It is most commonly used to translate the private source IP address of a host on an inside network to a public IP address on the outside network so it can communicate with hosts on the Internet. Internal hosts send Internet destined packets to the router which translates their source address to one of its own public IPs and saves a mapping, so that when it receives a reply it knows to forward it back to the internal host at the correct private address.

NAT was originally designed as a solution to the problem of public IPv4 addresses running out. It offers a scalable solution for Internet connectivity as many private IP addresses on the inside can be translated to a small number of (or one) shared public IP addresses on the outside.

For standard one-way NAT where a router is providing Internet connectivity to inside hosts using private IP addresses:

Inside local address —The IP address actually configured on the inside host's Operating System. The host's private IP address.

Inside global address — The NAT'd address of the inside host as it will be reached by the outside network. A public IP address reachable via the router's outside interface.

Outside local address —The IP address of the outside host as it appears to the inside network. The outside host's public IP address.

Outside global address —The IP address assigned to the host on the outside network by the host's owner. Router will report the same IP as the outside local address as it only has visibility of that one IP address for the outside host.

There are three main types of NAT:

- Static – maps inside local IP addresses to inside global IP addresses based on static configuration. Commonly used to give a fixed public IP address to internal servers which must be permanently available to outside hosts on the Internet.
- Dynamic – dynamically maps inside local IP addresses to inside global IP addresses from a preconfigured pool. A mapping is created when an internal host initiates a connection outside the private network.
- Dynamic with overloading (Port Address Translation/PAT) – dynamically maps inside local IP addresses and port numbers to the router's inside global IP address and port numbers. A mapping is created when an internal host initiates a connection outside the private network. Commonly used to allow normal inside hosts (which do not need to receive incoming connections) access to the Internet. Helps with IPv4 address space exhaustion.

Static NAT configuration:

- (config-if)# ip nat inside – declare the interface(s) on the internal network
- (config-if)# ip nat outside – declare the interface on the outside
- (config)# ip nat inside source static *inside_local inside_global* – create a static mapping.
This command needs to be issued separately for each mapping.

```
R1(config)#int f0/0
R1(config-if)#ip nat outside
R1(config)#int f1/0
R1(config-if)#ip nat inside
R1(config)#ip nat inside source static 10.0.1.10 203.0.113.3
```

Dynamic NAT configuration is a bit more complex. We need to create an ACL controlling which inside local addresses can be translated alongside a pool of inside global addresses

- (config-if)# ip nat inside – declare the interface on the internal network
- (config-if)# ip nat outside – declare the interface on the outside
- (config)# ip access-list {*acl_number* / *acl_name*} *rule* – create/populate an ACL to specify which inside local addresses can use NAT
- (config)# ip nat pool *pool-name* *first-address* *last-address* netmask *subnet-mask* – define a range of inside global addresses to use. Netmask is used only for verification: if the address range does not fit the subnet as determined by netmask, the command is rejected.
- (config)# ip nat inside source list {*acl_number* / *acl_name*} pool *pool-name* – enable dynamic NAT and allow mappings between inside local addresses permitted by the specified ACL and inside global addresses in the pool

```
R1(config)#int f0/0
R1(config-if)#ip nat outside
R1(config)#int f2/0
R1(config-if)#ip nat inside
```

Configure the pool of global addresses:

```
R1(config)#ip nat pool study-ccna 203.0.113.4 203.0.113.14 netmask
255.255.255.240
```

Create an access list which references the internal IP addresses we want to translate:

```
R1(config)#access-list 1 permit 10.0.2.0 0.0.0.255
```

Associate the access list with the NAT pool to complete the configuration:

```
R1(config)#ip nat inside source list 1 pool study-ccna
```

Dynamic NAT with overload configuration is similar to regular dynamic NAT, but instead of creating a pool of inside global addresses you specify just an interface to use and add the **overload** keyword.

- (config-if)# ip nat inside – declare the interface to be on the internal network
- (config-if)# ip nat outside – declare the interface to be on the outside
- (config)# ip access-list {*acl_number* / *acl_name*} *rule* – create/populate an ACL to specify which inside local addresses can use NAT
- (config)# ip nat inside source list {*acl_number* / *acl_name*} interface *ifname* overload – allow dynamic mappings between inside local addresses, permitted by the specified ACL, and ports and the inside global interface *ifname* and its ports.

```
R1(config)#int f0/0
```

```
R1(config-if)#ip nat outside
```

```
R1(config)#int f2/0
```

```
R1(config-if)#ip nat inside
```

Configure the pool of global addresses:

```
R1(config)#ip nat pool study-ccna 203.0.113.4 203.0.113.6 netmask  
255.255.255.240
```

Create an access list which references the internal IP addresses we want to translate:

```
R1(config)#access-list 1 permit 10.0.2.0 0.0.0.255
```

Associate the access list with the NAT pool to complete the configuration.

```
R1(config)#ip nat inside source list 1 pool study-ccna overload
```

Show and manage NAT status:

- #show ip nat translations – show the NAT mapping table
- #show ip nat statistics – show NAT statistics
- #debug ip nat – enable NAT debugging
- #clear ip nat translation – clear NAT mapping table

Chapter 11 – Quality of Service (QoS)

QoS is a set of tools used to give different applications their required quality of service during temporary periods of network congestion.

Different types of traffic have different requirements for:

- bandwidth
- delay – how long it takes packets to get from source to destination
- loss – proportion of packets lost on the way to destination
- jitter – variation in delay as sequential packets arrive at the destination

Data traffic, especially non-interactive batch traffic, is less sensitive to delay, loss and jitter, while real-time applications such as voice and video have strict requirements, even if they use less bandwidth.

It doesn't matter if an email takes a few extra seconds to reach its destination while the network is busy, but if this happens to a voice packet it will result in a bad quality call. QoS can be used to guarantee that the applications which need it get preferential treatment during periods of temporary congestion on the network.

Classification and marking

QoS needs to recognise the relevant packets in order to give them preferential treatment.

Classification is done by matching specific fields in packet headers.

Classification and marking on their own do not affect how packets are forwarded. They only set a marking which can be acting upon later by a policy. If a policy does not take an action on a marking then it has no effect.

It is best practice to do classification and marking as close to the source as possible – preferably on the switch which is connected to the host. This improves performance and simplifies configuration, as devices further down the line can rely on these markings rather than performing full classification themselves.

Cisco IP phones and Telepresence video units automatically mark their own packets.

Other types of hosts are typically not capable of marking their own traffic. For example if you want to guarantee bandwidth to traffic coming from an SAP server, you need to configure the classification and marking on the switch port connected to that server.

Where classification and marking must be manually configured, ACLs can be used to classify packets based on their layer 3 and 4 header information. For more complex classification, Network Based Application Recognition (NBAR) can be used, inspecting packets up to layer 7. NBAR has a built-in capability to recognise many popular applications, and you can also write your own custom signatures.

End devices other than IP phones or video units shouldn't be trusted to mark their own packets, as it would open up the potential of rogue users or applications to prioritise their own traffic. A trust boundary should be set at the access switch:

- (config-if)# mls qos trust extend cos *n*
OR (depending on switch model)
- (config-if)# switchport extend cos *n*
 - instruct the IP phone to mark packets incoming from the attached host with the specified COS value. Most often used to mark packets with the default 'best effort' value of 0, overriding anything set by the host.
- (config-if)# switchport extend priority trust – instruct a connected IP phone to trust COS header on packets coming from its PC port. Only required where the host is trusted to mark its own packets and rarely used.
- (config-if)# mls qos trust cos – extend trust to packets coming in from the port. Commonly used on ports connecting switches together to allow markings to pass across the campus network.

There are several schemes for packet marking. It is quite common for packets to have both a DSCP marking in their layer 3 header and a CoS marking in their layer 2 header. For example a Cisco IP phone will automatically mark its spoken voice packets with DSCP value EF and CoS value 5.

QoS policies can take an action based on any of the marking types.

- **Differentiated Service Code Point (DSCP)**, 6-bits in the Type of Service field of an IPv4 header or in the Traffic Class field of an IPv6 header (Layer 3). Backwards compatible with IP Precedence.
- IP Precedence (IPP), 3-bits in the Type of Service field of an IPv4 header (Layer 3), superseded by DSCP. 3 bits so possible values are 0-7
- Ethernet 802.1Q **Class of Service (CoS)** AKA **Priority Code Point (PCP)**, 3-bits in the 802.1Q field of an Ethernet frame (Layer 2). Can be only used when trunking is enabled. 3 bits so possible values are 0-7
- TIP, 3-bits in an 802.11 header of a wireless frame (Layer 2). 3 bits so possible values are 0-7.
- EXP, 3-bits in an MPLS label (between Layer 2 and Layer 3). 3 bits so possible values are 0-7.

DiffServ (differentiated services) defines several types of services – and their associated DSCP markings – to ensure consistency in QoS services:

- Expedited Forwarding (EF) – packets requiring low loss, delay and jitter, such as spoken voice. Also known as DSCP 46.
- Assured Forwarding (AF) – Assumes 4 queues of differing quality and 3 drop priority classes within each queue for a total of 12 DSCP values. Text name follows the format of AFXY, where X is the queue (1 – worst, 4 – best) and Y is drop priority (1 – best, 3 – worst). AF 41 indicates the best queue and the best drop priority of the AF values (which is below the EF value).
- Class Selector – 8 values (text value: CSX) used to map old IPP values to DSCP values.

Examples of recommended DSCP usage:

- EF (DSCP 46) – Voice data
- AF4y – Interactive video
- AF3y – Streaming video
- AF2y – High-priority data
- CS0 – Standard data ‘best effort’

Queueing

Queueing happens when a link is congested – packets arrive on the router or switch on an incoming interface faster than the device can send them out of the outgoing interface. This is most likely to occur where there is a speed mismatch, for example packets are arriving on a GigabitEthernet interface on the inside, and being sent out to the Internet via a 10 Mbps link.

Queueing only occurs when there is congestion. If the device can send packets out as soon as they arrive then there is no queue.

A standard queue with no QoS policy applied uses a FIFO First In First Out scheduler – the packet that arrives first in a given queue is sent first. If there is congestion and voice packets (for example) are stuck behind data packets waiting to get out, this can cause bad quality calls.

QoS queuing methods can be used to jump voice packets to the front of the queue, giving the required quality of service where required.

Class-Based Weighted Fair Queuing (CBWFQ) can be used to give a minimum bandwidth guarantee to specified traffic classes during periods of congestion on an interface. For example you can give call signalling traffic (control packets used to set up and tear down calls) a bandwidth guarantee.

Low Latency Queueing (LLQ) is a CBWFQ policy with a priority queue. Bandwidth guarantees can be given to specified traffic classes, and there is also a priority queue whose packets are always jumped straight to the front of the queue. For example you can put spoken voice packets to the front of the queue when they arrive, and give call signalling packets a bandwidth guarantee.

The amount of bandwidth available to an LLQ priority queue is limited to ensure it cannot completely starve out other traffic. (If it was using 100% of the available bandwidth then that would leave 0% for other traffic types.) The default maximum is 33% and it is not recommended to go above this.

The administrator should ensure that the priority queue has enough bandwidth to handle all of its packets. For example if an LLQ policy is configured on a 10 Mbps interface, with a 33% priority queue for spoken voice, then if more than 3.3 Mbps of spoken voice packets arrive there will not be enough bandwidth for them and they will all result in bad quality calls. Call Admission Control (CAC) can be used on Cisco voice servers to limit the number of concurrent calls going over the IP network.

LLQ configuration example (only takes effect when interface is congested):

Class-maps specify the packets which we'll be taking an action on later in the policy-map.

Here we're looking for spoken voice and call signalling packets, which have been automatically marked by IP phones.

```
class-map VOICE-PAYLOAD
match ip dscp ef
class-map CALL-SIGNALING
match ip dscp cs3
```

Policy-maps take an action on the traffic. We're guaranteeing spoken voice packets up to 33% of the available bandwidth (and limiting them to a maximum of 33%) and jumping them to the front of the queue. Voice signalling packets are guaranteed up to 5% of the available bandwidth (with no limit). 'Fair queuing' is applied to all other traffic (it's better than FIFO which can give large packets an unfair share).

```
policy-map WAN-EDGE
class VOICE-PAYLOAD
priority percent 33
class CALL-SIGNALING
bandwidth percent 5
class class-default
fair-queue
```

Finally we need to apply the policy to an interface. The earlier commands have no effect until this is done.

```
interface Serial0/0/0
bandwidth 768
service-policy out WAN-EDGE
```

Shaping and policing

Traffic Shaping and Policing can be used to control traffic rate. They both measure the rate of traffic through an interface and take an action if the rate is above a configured limit.

Classification can be used to allow different rates for different traffic types.

Traffic shaping buffers any excess traffic so the overall traffic stays within the desired rate limit.

Traffic policing drops or re-marks excess traffic to enforce the specified rate limit.

Traffic policing is often configured inbound by service providers on their customer facing interfaces to ensure that customers cannot send traffic at a faster rate than they have paid for. Traffic is dropped, or marked to get worse service during periods of congestion in the service provider's core network.

Traffic shaping is often configured outbound by customers on their service provider facing interfaces to smooth traffic to the rate they have paid for. If outbound traffic is queued up, they can prioritise the traffic types which need it.

Traffic shaping configuration example (customer has 100 Mbps physical interface to service provider but has only paid for 10 Mbps rate. Overall traffic is shaped to 10 Mbps rate, and important packets are given better service if traffic is queued up):

Class-maps specify the packets which we'll be taking an action on later in the policy-map.

Here we're looking for spoken voice, video, and call signalling packets, which have been automatically marked by IP phones and Telepresence units.

```
class-map VOICE
  match ip dscp ef
class-map VIDEO
  match ip dscp af41
class-map SIGNALLING
  match ip dscp cs3
```

Policy-maps take an action on the traffic. We're guaranteeing spoken voice packets up to 1 Mbps of the available bandwidth (and limiting them to that) and jumping them to the front of the queue. Video packets get up to 3 Mbps of the available bandwidth (and are limited to that) and are also jumped them to the front of the queue. Voice signalling packets are guaranteed up to 128 Kbps of the available bandwidth (with no limit). 'Fair queuing' is applied to all other traffic (it's better than FIFO which can give large packets an unfair share).

```
policy-map NESTED
  class VOICE
    priority 1024
  class VIDEO
    priority 3072
  class SIGNALING
    bandwidth 128
  class class-default
    fair-queue
```

We create a 2nd policy map to shape the overall traffic to 10 Mbps, and nest the 'NESTED' policy-map we just created inside this. This is the policy which will be applied to the interface to shape the overall traffic to 10 Mbps. When more than that is queued to be sent out, traffic will be prioritised as specified in the 'NESTED' policy-map.

```
policy-map WAN-EDGE
  class class-default
    shape average 10000000
  service-policy NESTED
```

The policy is applied to the interface:

```
Interface FastEthernet0/0
  service-policy out WAN-EDGE
```

Congestion avoidance

The outbound queue on an interface has a maximum depth which can fill up if packets arrive at a rate faster than they can be sent out. Tail Drop occurs when the queue is full - further incoming packets are dropped until there is space for them in the queue.

When packets are dropped, the TCP window shrinks and transmission slows down. This causes packets to then arrive at a slower rate which reduces the length of the queue.

The TCP window size and transmission rates will then increase again until the queue fills up, and then a cycle of increasing and decreasing transmission rates will keep repeating.

This is not efficient in terms of making the best utilisation of the bandwidth of the link.

The problem is that once the queue fills up, all flows are impacted and global synchronisation occurs – all senders reduce their transmission rate at the same time, leading to link underutilisation, after which they all increase transmission rate, potentially leading to another bout of congestion.

Random Early Detection (RED) and Weighted Random Early Detection (WRED) are congestion avoidance mechanisms designed to avoid global synchronisation. The fact that they improve performance is counter intuitive because they drop packets before the queue has filled up necessitating that.

RED randomly, at a rate depending on how much of the queue has filled, drops packets before congestion occurs, thus signalling to senders to slow down transmission. This way traffic is slowed down and then picks up gradually, because only individual traffic streams are impacted at a time. The overall utilisation of the link for all streams is improved.

WRED is an extension to RED that takes into account packet priority. It will drop packets with lower priority more often, as determined by their classification. This breaks global synchronisation *and* gives better service to more important traffic.

Wireless QoS levels

Cisco WLCs support 4 service levels:

- Platinum – VoIP, CAPWAP control tunnels
- Gold – Video, mission-critical interactive traffic streams
- Silver – Best-effort, default level
- Bronze – Guest services

Chapter 12 – Miscellaneous IP Services

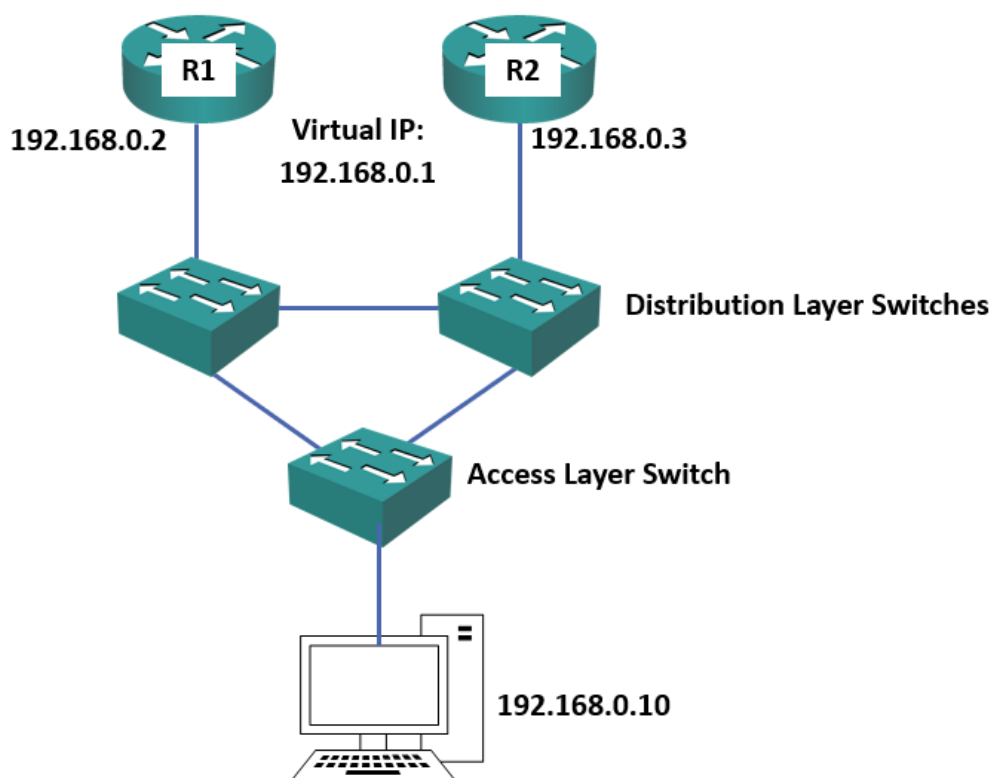
First Hop Redundancy Protocols

Hosts can have only one default gateway configured at a time, so when that router fails hosts will lose connectivity. First-hop redundancy protocols (FHRP) provide redundancy and failover for default gateways, eliminating the single point of failure.

There are several FHRP protocols, the CCNA exam focuses on Cisco-proprietary Hot Standby Router Protocol (HSRP). HSRP works by having two or more routers that can act as the default gateway – one of them selected as **active** and the rest selected as **standby** or **passive** (same thing, different name).

The physical interfaces on each router have their own IP address, but the **active** router also has a virtual IP address and an associated virtual MAC address. It is this virtual IP address that is used as the default gateway for hosts in the subnet.

Should the active router fail, standby routers will detect this and one of them will step in to become active, taking over the virtual IP address and corresponding virtual MAC. This way, failover is achieved with no changes to hosts' configuration.



In the example diagram above, HSRP has been configured for the 192.168.0.0/24 subnet. R1 has been configured with IP address 192.168.0.2 on its physical interface, and R2 has been configured with IP address 192.168.0.3 on its physical interface. Hosts in the subnet use the HSRP Virtual IP address 192.168.0.1 as their default gateway address which is active on either R1 or R2. The hosts are not aware of the HSRP process and nothing changes from their point of view if there is a failover.

HSRP allows per-subnet load balancing, where different subnets use different routers as their default gateway. For example 192.168.0.0/24 uses R1 as active and R2 as standby, and 10.10.0.0/24 uses R2 as active and R1 as standby.

HSRP configuration example:

```
R1(config)#interface g0/1
R1(config-if)#ip address 192.168.0.2 255.255.255.0
R1(config-if)#no shutdown
R1(config-if)#standby 1 ip 192.168.0.1
```

```
R2(config)#interface g0/1
R2(config-if)#ip address 192.168.0.3 255.255.255.0
R2(config-if)#no shutdown
R2(config-if)#standby 1 ip 192.168.0.1
```

The other FHRP protocols are GLBP Gateway Load Balancing Protocol and VRRP Virtual Router Redundancy Protocol.

GLB supports load balancing across multiple default gateway routers for traffic in the same subnet (HSRP and VRRP cannot do this). It uses an **Active Virtual Gateway** (AVG) and up to four **Active Virtual Forwarders** (AVF) – more routers can be in a group, but these are considered secondary and only act as backups.

There is one virtual IP used as the default gateway address, but the AVG and each AVF have their own associated virtual MAC. The AVG responds to ARP requests for the virtual IP: sometimes with its own virtual MAC, sometimes with the virtual MAC of another AVF. This way, some hosts route their packets via the AVG, others via one of the AVFs. This achieves per-host load balancing.

Virtual Router Redundancy Protocol is an open standard protocol (unlike HSRP and GLBP, which are Cisco-proprietary) that is very similar to HSRP in its operation and configuration. It supports one **master router** and multiple **backup routers**.

MAC addresses and multicast group addresses used by FHRP (XX is the hex representation of group ID):

- HSRPv1: 00:00:0c:07:ac:XX, 224.0.0.2 (all routers)
- HSRPv2, IPv4: 00:00:0c:9f:fx:XX, 224.0.0.102 (HSRP-specific)
- HSRPv2, IPv6: 00:05:73:a0:0X:XX, FF02::66
- GLBP: 00:07:B4:0X:XX:YY, 224.0.0.102, FF02::66 (YY==AVF number)
- VRRP: 00:00:5E: 00:01:XX, 224.0.0.18, FF02::12

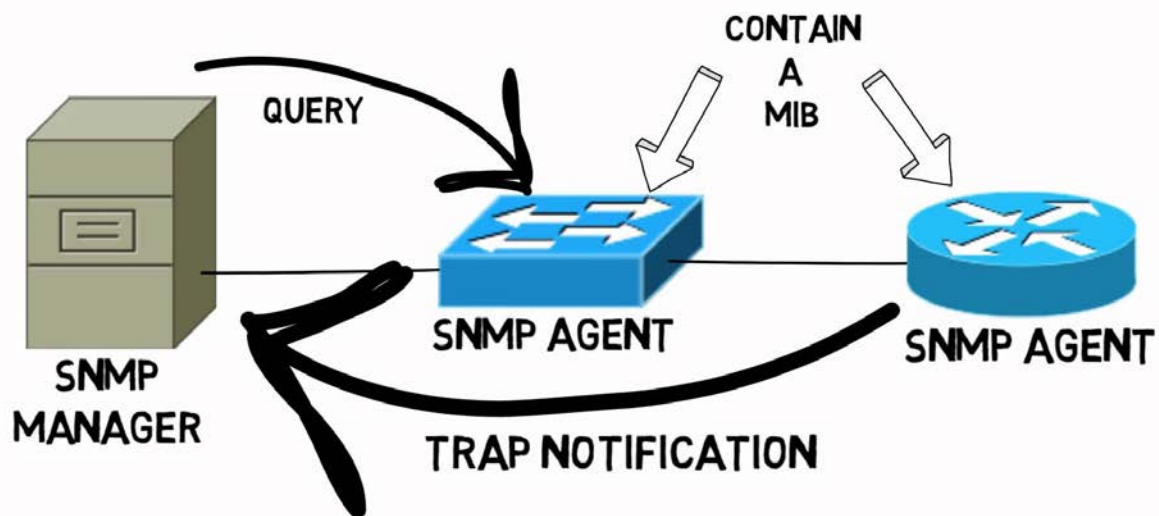
Simple Network Management Protocol (SNMP)

SNMP allows monitoring and configuration setting, status information, counters and other information in a centralised manner.

Regular network devices are **SNMP agents** and gather this information as variables in a semi-standardised **Management Information Base (MIB)** – some variables are common to all devices, others to some types of devices, and some are specific to a particular device.

Agents exchange information with **SNMP managers**, typically **Network Management Stations** (NMS servers), which poll agents for information, compile it and present it to users and network administrators.

Managers usually ask agents for information using **SNMP Get/GetNext/GetBulk Requests** and agents respond with **SNMP Get Responses**. Alternatively, agents can inform managers themselves using either **SNMP Traps** or **SNMP Informs** (the latter added in SNMPv2). Both messages use UDP, but while Informs have reliability added at the application layer (agent will ask the manager to acknowledge receipt of message and re-send it if such an acknowledgment is not forthcoming), Traps do not (if the message gets lost, it's lost).



MIBs use a tree like structure and its variables are identified by Object IDs (OID) that can be given as either numbers or names. For example:

iso.org.dod.internet.private.enterprise.cisco.ciscomgmt.ciscoflash.group

1.3.6.1.4.1.9.9.10

Part of each MIB is standardised in RFCs, the rest is defined by the vendor. NMS helps make sense of these variables and their values.

As for security, SNMP offers one of two methods, depending on version:

- SNMPv1 and SNMPv2c – Communities, i.e. pre-shared passwords exchanged in cleartext between agent and manager.
- SNMPv3 – Also supports authentication with username and hashed password, message integrity and optionally encryption.

FTP, FTPS, TFTP, SFTP and IOS filesystem

Paths begin with a file system prefix (e.g. flash:) followed by a slash-separated path.

Types of file system:

- Opaque – aliases
- Network
- Disk – Flash drives
- Usbflash – USB storage
- NVRAM – Non-volatile RAM, used for startup-config by default

File system navigation and management:

- # show file systems
- # dir *dirname* – show directory contents
- # show flash – show contents of the default flash file system
- # copy *source dest* – copy files
- # verify /md5 *filename md5hash* – verify MD5 hash of a file

Note that for copy command you can use keywords such as **tftp**, **ftp** or **scp** to copy files over the network using the selected protocol:

- # copy tftp *dest* – copy from TFTP server, will launch the client interactively
- # copy [ftp://user:pass@host/filename](#) *dest* – copy from FTP server, start client in batch mode
- (config)# ip ftp username *user* – default FTP username
- (config)# ip ftp password *pass* – default FTP password

FTP uses two TCP connections: one for control messages (default port: 21), the other for data.

In active mode, the client initiates the control connection using port 21 to the server, selects a random port for data connection, starts listening on it and then tells the server to open a data connection back to this port on the client using the PORT command.

In passive mode, after initiating the control connection, the client sends a PASV command to enable passive mode and then the server starts listening on a random port and tells the client the port's number using the PORT command. The client then initiates a data connection to this port.

A NAT or firewall device between the FTP client and server is liable to break the application, unless the device supports FTP application inspection.

FTP Secure (FTPS) is a protocol built upon FTP with TLS encryption. It has two modes:

- Explicit mode – only one port (21) for both control and data connection. Client creates a control connection to port 21, initiates TLS with AUTH command and then does the same for data connection. This mode retains backwards compatibility with regular FTP.
- Implicit mode – client connects to port 990 (control connection) and 989 (data connection) and goes straight into TLS without an explicit AUTH command.

SSH File Transfer Protocol (SFTP) is not directly related to FTP but serves the same purpose. It uses SSH for encryption and authentication. It uses SSH's port 22.

Trivial File Transfer Protocol (TFTP) is a stripped down protocol used for file transfer on UDP port 69. It handles error recovery at the application layer in a very basic way, resulting in slow transfer speeds.

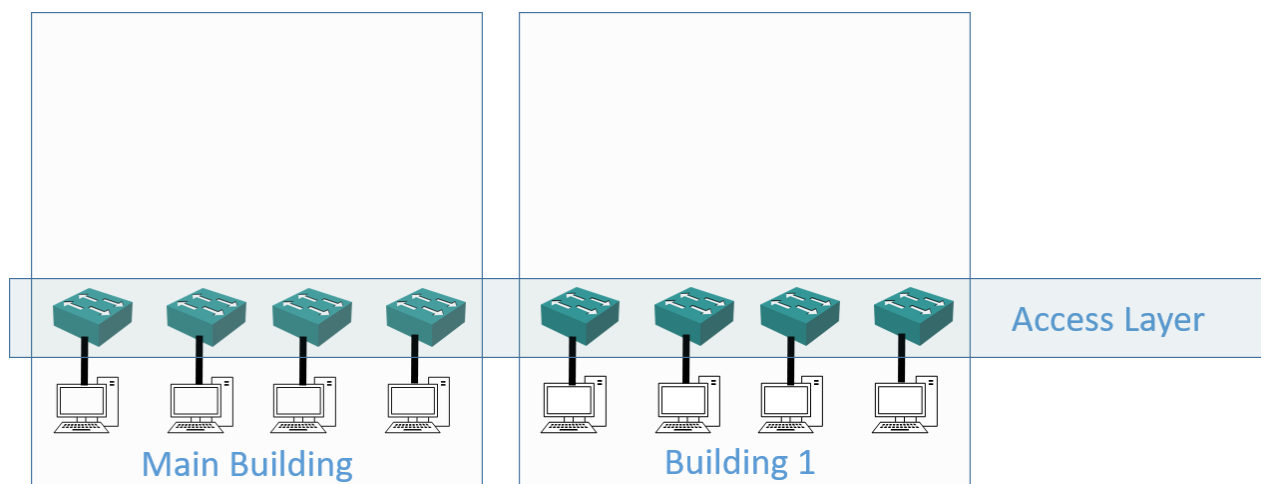
Part IV Network Architectures

Chapter 13 – LAN architectures

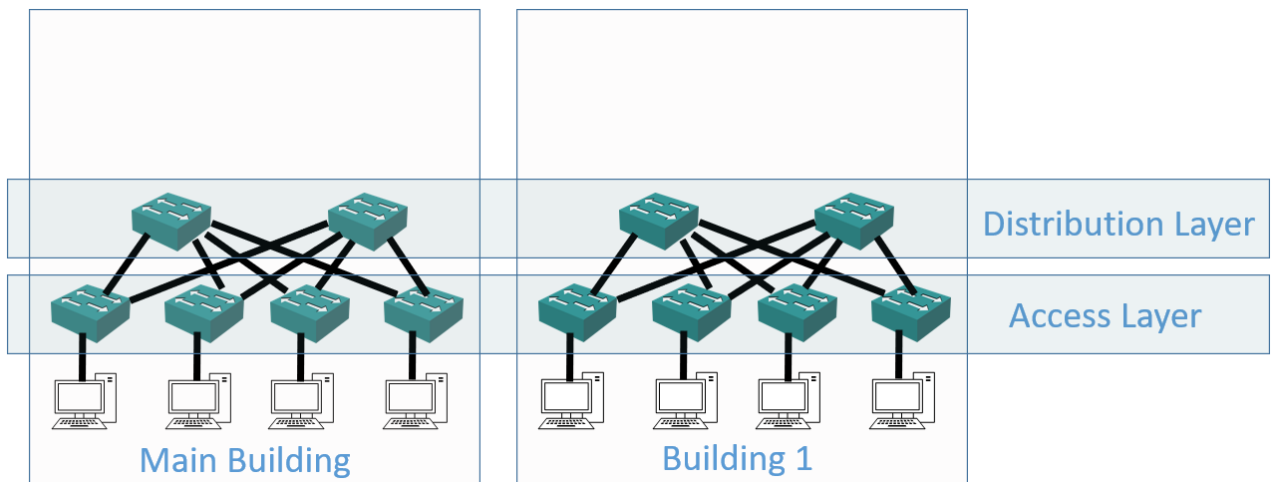
Campus LAN – Local Area Network LAN in a single building or several buildings in close proximity, for example a university campus.

The campus LAN should be designed for scalability, performance and security. To aid in a best practice design process, the network topology is split into access, distribution and core layers. The layers have their own design principles and characteristics.

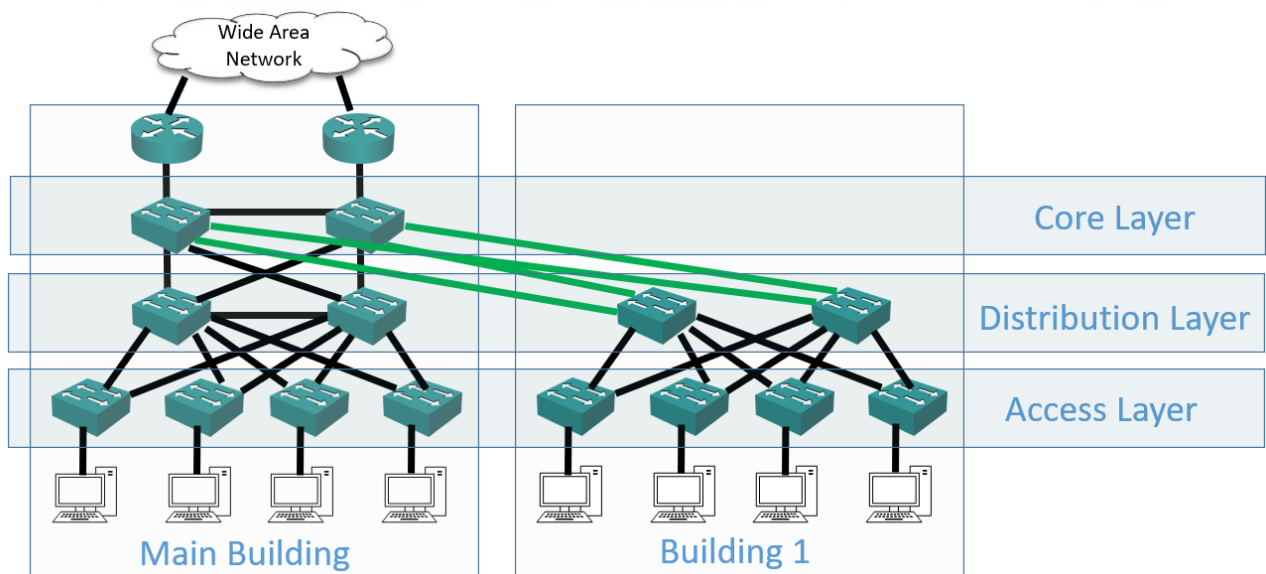
End hosts such as desktop computers, servers and IP phones connect into the network at the access layer. It is designed to have a high port count at an affordable cost. Desktops typically have only one Network Interface Card (NIC) so they connect into one switch or Wireless Access Point. Servers will often have dual NICs and connect to a pair of redundant switches. Client access security measures are enabled at the Access Layer.



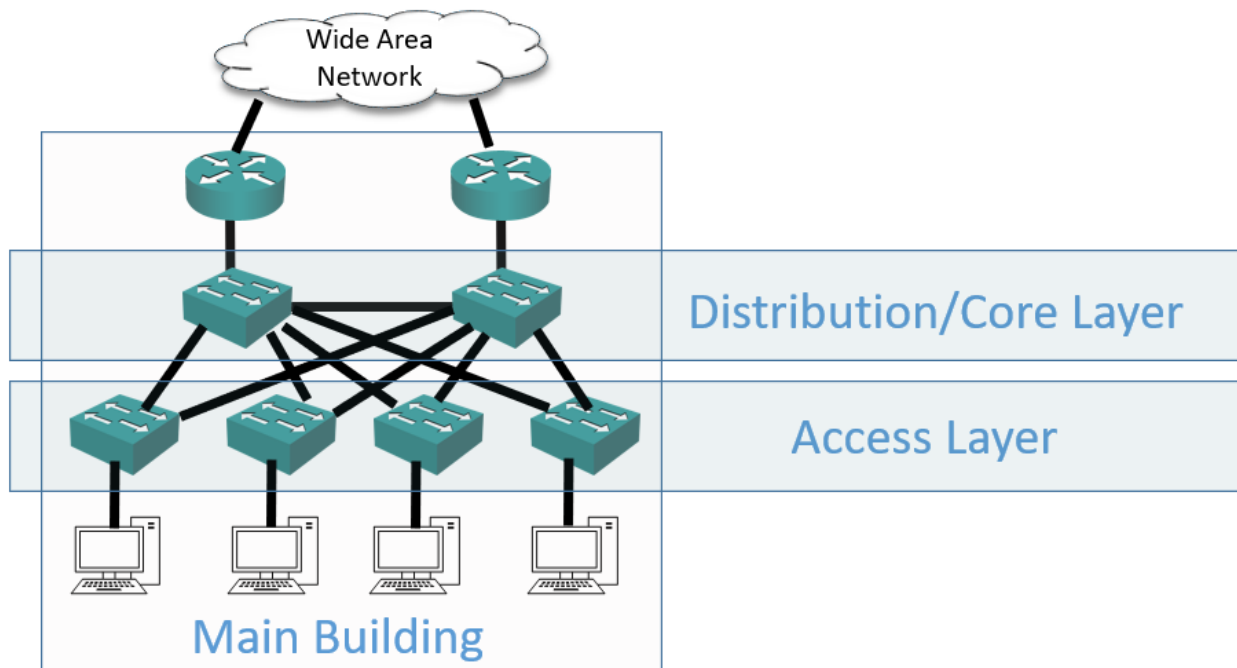
Access Layer switches uplink to Distribution Layer switches. The Distribution Layer switches serve as an aggregation point for the Access Layer and provide scalability. Distribution Layer switches are typically deployed in redundant pairs, with downstream Access Layer switches connected to both. End hosts are not typically connected here. Most software policy such as QoS is enabled at this layer.



Distribution Layer switches uplink to Core Layer switches. Core Layer switches are typically deployed in redundant pairs, with downstream Distribution Layer switches connected to both. Traffic between different parts of the campus travels through the core so it is designed for speed and resiliency. Software policy slows the switch down so should be avoided in the Core Layer.



Smaller campuses do not need the scalability of three separate layers. In these cases a Collapsed Distribution and Core layer is used, where the Distribution and Core layer functions are performed on the same hardware device.



Finally, in a Small Office Home Office SOHO environment, a very simple design can be used. A SOHO router, combining router, firewall, switch and wireless AP functions in one device is often the only network device in use. The network can be extended by adding switches and autonomous APs.

Power over Ethernet (PoE)

PoE allows one network device, called PSE Power Sourcing Equipment, to provide power to another device, called a PD Powered Device, via Ethernet cabling. The Power Sourcing Equipment is most commonly a switch, which is directly connected to the Powered Device such as an IP phone or wireless access point. PoE is useful because it negates the need to plug the Powered Device into a wall power port.

PoE provides for auto-negotiation to determine PoE power class, i.e. how much power to supply. Monitoring and power level adjustment are provided via the auto-negotiation and also CDP/LLDP messages. There are several PoE standards:

- Cisco Inline Power, Cisco-proprietary, early standard, 2 powered wire pairs, 7 Watts at PSE
- PoE, 802.3af, 2 powered wire pairs, 15 Watts at PSE
- PoE+, 802.3at, 2 powered wire pairs, 30 Watts at PSE
- UPoE, 802.3bt, 4 powered wire pairs, 60 Watts at PSE
- UPoE+, 802.3bt, 4 powered wire pairs, 100 Watts at PSE

Power Sourcing Equipment will typically provide auto-negotiated power by default to any device which requests it.

You can enable power policing to protect against a PD attempting to draw more power than it was assigned:

- (config-if)# power inline police

by default the interface in question will log an error, shutdown, enter **err-disable** state and need to be restarted. You can configure automatic recovery from such a state:

- (config)# errdisable recover cause inline-power

Alternatively, you can change the policy to log the event and restart the port immediately:

- (config-if)# power inline police action log

To show power policy:

- #show power inline police

Chapter 14 – WAN architecture

Metro Ethernet (MetroE) Layer 2 VPNs (Virtual Private Networks)

Metro Ethernet is a name for service provider WAN services that allow customers to connect their offices to each other using Ethernet links. From the customer's point of view it looks like their offices are connected to a large Ethernet switch.

Hosts in different offices can be in the same IP subnet because the connection between offices is in the same Layer 2 domain. This can be very useful for Disaster Recovery where hosts and services can be failed over to another site while keeping their same IP address.

The service provider (SP) needs to physically install a switch in a **Point of Presence (PoP)** near customers – close enough to allow Ethernet connection from customers.

Customer Edge (CE) and Provider Edge (PE): devices at the edge of the respective physical networks.

The links between customers and PoP are called (Ethernet) **access links**, which form **user network interfaces (UNI)**. The customer has no visibility of the SP's network – the SP promises to deliver Ethernet frames across the WAN, but its internals are irrelevant and hidden. Since Metro Ethernet provides Ethernet service, it is possible to connect to it via a switch instead of a router.

Long-range Ethernet standards for access links (note that given the range requirement, all use single mode fibre, even if the standard also supports multimode fibre e.g. 1000BASE-LX):

Physical layer standard	Max speed	Max range	Medium
100BASE-LX10	100Mb/s	10km	Single mode fibre
1000BASE-LX	1Gb/s	5km	Single mode fibre
1000BASE-LX10	1Gb/s	10km	Single mode fibre
1000BASE-ZX	1Gb/s	100km	Single mode fibre
10BASE-LR	10Gb/s	10km	Single mode fibre
10BASE-ER	10Gb/s	30km	Single mode fibre

Metro Ethernet can operate in one of three topologies, as defined by MEF Metro Ethernet Forum:

- Ethernet Line Service (E-Line) – point-to-point service between two sites. Multiple E-Lines can run on a single physical access link, so a central site can connect to several remote sites. Each E-Line should be in a different subnet.
- Ethernet LAN Service (E-LAN) – full mesh, acts as ordinary LAN, all sites can communicate directly with each other. All hosts can be in the same subnet.
- Ethernet Tree Service (E-Tree) – hub and spoke/partial mesh/point-to-multipoint, all sites can communicate with the central site but not with each other.

Multiprotocol Label Switching (MPLS) Layer 3 VPNs

MPLS Layer 3 VPNs are another service provider WAN service that allows a customer to connect their offices to each other. It is a Layer 3 service where the CE devices are Layer 2 adjacent with their connected PE device (rather than with each other as in Metro Ethernet). The CE devices have visibility of the PE devices and do not appear to be Layer 2 adjacent to each other. The CE devices are in different IP subnets from each other.

The service provider has a large, shared core IP network and connects all of its customers to it. PE routers at the edge insert an MPLS header into customer packets between their Layer 2 and Layer 3 headers that tells the rest of the MPLS network how to forward these packets. Customer traffic is kept strictly segregated from other customers for security.

CE routers are neighbours with their connected PE routers for routing. PE routers act as next-hop routers for their connected CE routers for traffic between the customer offices.

CEs and PEs exchange routing information using static routes or a routing protocol, while internally the MPLS network uses MP-BGP (Multiprotocol Border Gateway Protocol) which redistributes the learned routes into the MPLS network.

Since MPLS is a Layer 3 service, it can use a variety of Layer 2 access links, unlike MetroE which requires Ethernet.

MPLS enables the use of Layer 3 QoS tools: CE devices can mark outgoing IP packets to enable the MPLS network to recognise and prioritise them if this is supported by the Service Provider.

Internet VPNs

Private WANs are not always available or cost effective. Offices can use cheap consumer access links to the Internet instead and connect to the main office using VPNs. These VPNs need to be encrypted (to provide confidentiality), authenticate users (to prevent unauthorised access to the network), maintain data integrity (ensure it hasn't been changed in transit) and prevent replay attacks (an attacker re-sending sniffed packets to try to gain access).

VPNs can be used for site-to-site connections (permanently connecting various branches) and also for remote access (on-demand connections by remote users to the network).

For site-to-site VPNs the VPN tunnel endpoints are typically routers or firewalls.

When a user sends traffic to the other office on the other side of the VPN, the unencrypted packets arrive from the internal network to the VPN endpoint which then encrypts them and sends them across the Internet to the VPN endpoint on the other side, which decrypts the packets and sends them across its internal network to the final destination.

For site-to-site VPNs, IPSec with shared key encryption over GRE tunnel is often used. IPSec deals with data confidentiality and data integrity, GRE deals with tunnelling, including multicast and broadcast encapsulation.

For a remote access VPN, client software on a user's laptop connects to a VPN device such as a router or firewall in the corporate office.

Remote access VPNs are usually TLS based. Cisco's client software is called AnyConnect, but there are many others including OpenVPN (and derivatives) and Wireguard.

Chapter 15 Cloud Architecture

Bare metal server – operating system and applications are installed directly on the underlying hardware. This is commonly used where a single application requires the entire hardware power for itself.

Server virtualisation means services are not run on a bare metal server but instead on virtual machines (VMs) of which several can run on a single physical host. Each VM has its own operating system and applications. VMs are assigned a share of the host's resources (CPU threads, RAM, disk space, network bandwidth) and directly managed by a hypervisor which sits between the virtual machines and the underlying hardware.

A hypervisor runs a virtual switch (eg, Cisco Nexus 1000VE which is end-of-sale, or Cisco Application Virtual Switch) to control traffic flow inside the host and allow different virtual machines to be in different VLANs. The link between the host and its directly connected switch is configured as a trunk.

Hosts are typically mounted in racks with two switches (for redundancy) at the top of the rack (ToR) connected to the hosts in that rack. Racks are arranged in rows with additional switches at the end of each row (EoR) connected to the ToR switches in that row.

Traditionally, companies relied on their data centre staff to manage (add, edit, remove) VMs.

With public cloud computing, the data center is managed by a cloud provider. Cloud computing allows self-service provisioning by a customer – a customer can manage VMs, including ordering new ones or assigning more resources to existing ones, via APIs or a web GUI provided by the cloud provider and the request is handled automatically.

5 NIST (US National Institute for Science and Technology) requirements for cloud:

- On-demand self-service – see above
- Broad network access – available over many types of network (Internet, VPNs, private WAN etc.) from many devices
- Resource pooling – services are handled from a pool of resources that are shared between customers
- Rapid elasticity – Resource pool is large enough or can be expanded rapidly enough as to not be a constraint on customer's requests for new services
- Measured services – allows monitoring and reporting on usage of resources.

Cloud can be private or public. A private cloud is self-hosted by the organisation in question and they provide cloud services to their internal departments, such as self-service provisioning of a virtual machine without having to raise a ticket with the IT department.

A public cloud is a service provided by a third-party, such as Amazon Web Services or Google Cloud.

There are several “as a service” models of cloud:

- Software as a service – provides a working software to use, the underlying OS and (virtualised) hardware are oblique to the consumer.
- Platform as a service – provides a working OS along with development tools for developing and testing of software
- Infrastructure as a service – provides a VM on which the consumer can install an OS and further software components

Public clouds can be reached in several ways, each with its advantages and disadvantages:

- Internet – Easiest to set up, most flexible, but does not provide security and QoS.
- Internet VPN – Adds encryption.
- Private WAN – QoS and security, but not flexible, changing public cloud provider requires making changes to private WAN.
- Intercloud exchange – a special type of private WAN that connects to many public cloud providers, making switching providers easier.

Part V Network Automation

Chapter 16 – Introduction to controller-based networking

Router and switch planes:

- **Data (Forwarding) Plane:** Traffic which is forwarded through the device. This is the normal user traffic forwarded by the device such as web browsing or email packets.
- **Control Plane:** Makes decisions about how to forward traffic. Control plane packets such as routing protocol or spanning tree updates are destined to or locally originated on the device itself.
- **Management Plane:** The device is configured and monitored in the management plane. For example at the CLI through Telnet or SSH, via a GUI using HTTPS, or via SNMP or an API (Application Programming Interface).

In a traditional networking model, all functions on all these planes are decentralised, ie run independently on each device. Devices talk to each other using protocols such as STP or OSPF to create a coherent model of the network (“host A has this MAC and to reach it I should send messages out of that port”, “there are such and such subnets and I can reach them via this router”, etc.) and act on it when forwarding messages.

Data plane functions have to stay decentralised, as they need to be performed on network devices themselves, which have specialised hardware such as Application Specific Integrated Controllers for frame/packet forwarding and ternary content-addressable memory (TCAM) for quick MAC lookups.

Control functions, however, can be to a large extent centralised in a controller, which is the point of a network software-defined architecture (SDA) AKA software-defined networking (SDN).

In such an architecture, a controller talks to network devices using a southbound interface (SBI), usually a specialised API, that allows it to gather information on the devices, use it to create a model of the network, configure the devices and take control of many control plane functions.

It then communicates with an interface for network administrators or automation software via a northbound interface (NBI), which allows viewing of current network state and performing higher-level configuration of the network.

SDN introduces a fourth, **application plane**. These are the applications that talk to the controller using NBI. When a configuration change is requested, the controller decides how it should be implemented and communicates with network devices via SBI to change their state and configuration.

Examples of SBIs

- OpenPK – Cisco-proprietary API using Java, C or Python.
- OpenFlow – API using an Imperative SDN model, sending detailed instruction to network devices.
- OpFlex – API using a Declarative SDN model, sending instructions but leaving the implementations to the devices.
- NETCONF – Open standard using XML and Remote Procedure Calls (RPC), usually using SSH for transport.
- RESTCONF – Open standard using JSON or XML and REST with HTTP (usually HTTPS) for transport.

Examples of NBIs:

- REST – JSON with HTTP as transport (usually), see further chapters
- Open Society Gateway initiative (OSGi) - Java-based

There are many models of SDA and corresponding software packages:

Open SDN, OpenFlow, OpenDaylight

Open SDN is an SDA model developed by Open Networking Foundation. It defines an IP-based SBI called OpenFlow that centralises most control plane functions, while allowing for various APIs to be used for NBI.

OpenDaylight (ODL) is the most popular Open SDN controller. It is open-source, has been developed since mid-2010s and is now maintained by The Linux Foundation. It can support other SBIs beyond OpenFlow. As for NBI, it provides an OSGi Java API for apps running on the controller and a REST API for remote apps.

Cisco used to offer an Open SDN controller called Cisco Open SDN controller, but it was discontinued.

Cisco Application Centric Infrastructure (ACI)

ACI is an intent-based networking (IBN) model designed for data centres, build around application architecture. It integrates well with orchestration software for managing VMs to allow quick implementation of requests. It leaves most control plane functions decentralised.

ACI uses spine and leaf topology. In a single site, all spine switches connect to all leaf switches but no spine switch can connect to another spine switch and no leaf switch can connect to another leaf switch. All endpoints – routers, bare metal servers, virtual switches, etc. connect to leaf switches only. The APIC (see below) also connects to leaf switches.

This network is controlled by an Application Policy Infrastructure Controller (APIC): administrators or orchestration software define policies and intents for Endpoint Groups (EPG) (“these hosts should be able to talk to each other but not to these other hosts”) and APIC translates these into specific network states and configuration that it pushes via OpFlex SBI to network devices that implement them.

Cisco APIC Enterprise Model

APIC-EM is a now-discontinued solution designed to enable SDN for network devices that do not support specialised SBIs. It works around the problem by utilising existing tools – Telnet/SSH and SNMP – to monitor and configure network devices. It thus leaves the control plane untouched and focuses on management plane functions. APIC-EM controller offers inbuilt tools for network management and a REST API.

APIC-EM features have been mostly folded into Cisco DNA Center (DNAC), see next chapter.

Chapter 17 – Cisco Software-Defined Access

Cisco Software-Defined Access is a SDA model for enterprise campuses. It consists of a physical underlay – all the network devices, cables and wireless links – and a logical overlay where communication between endpoints happens. Taken together, underlay and overlay form the network’s fabric.

The underlay can be a completely new installation – greenfield deployment – or can utilise an existing infrastructure, in which case care must be taken to ensure device compatibility and avoid misconfiguration.

On top of the underlay, VXLAN tunnels – an open protocol for encapsulating Layer 2 frames in Layer 4 UDP datagrams – are created and these carry traffic between endpoints.

The overlay model changes some fundamental assumptions and solutions used in a network. In greenfield deployments, traffic between switches is fully routed (Layer 3), not switched (Layer 2), so STP is no longer needed and a routing protocol (IS-IS) is used instead. This, among other benefits, avoids loops without blocking any links.

Cisco DNA Centre (DNAC) serves as the controller for the network. In greenfield deployments it can control both the underlay and the overlay, but when using existing infrastructure it must limit itself to the overlay with the underlay configured manually – otherwise it would misconfigure the underlay as far as pre-existing requirements are concerned. Furthermore, only specific legacy devices can support SDA and some of them only in some roles, but not others (see bolded parts below).

Endpoint devices are connected to **SDA Edge Nodes**, Layer 3 switches that replace traditional access switches and are used by endpoint devices as default gateways, making First-Hop Redundancy Protocol redundant.

When an endpoint device sends a packet, it travels to the device's SDA Edge Node. This node – which will serve as the ingress node for the connection – looks up the Endpoint Identifier – i.e. the prefix of the destination address of the packet – at a LISP map server, AKA an **SDA Control Node**, that returns the matching Routing Locators (RLOCs) – i.e. the SDA Edge Nodes that connect to destination subnet. The ingress node will then check with the DNAC if the connection is permitted (see below) and, if so, it will create a VXLAN tunnel to the egress node, i.e. the switch indicated by the RLOC given by LISP and send the traffic in question via this tunnel.

Switches that connect to endpoints beyond SDA's control, usually WAN routers, are called **SDA Border Nodes**.

DNA Center uses a REST API for its NBI and a combination of SSH/Telnet (legacy devices) and RESTCONF/NETCONF (newer, SDA devices). As mentioned above, it enables a new security model, replacing traditional ACLs, that is based on policies and Scalable Group Access.

With a DNAC, administrators define groups of endpoints, marked by a Scalable Group Tag (SGT) and set policies on which group can communicate with which other groups. This greatly simplifies access security management, which can get very unwieldy in large deployments using traditional ACLs.

When it comes to management, DNAC covers many of the areas covered by traditional management systems, such as Cisco Prime Infrastructure (acts as a separate controller) or Cisco Network Assistant (running on workstation). Its advantages lie in easier QoS configuration, analysis of encrypted traffic, more information on client and device health, traffic monitoring over time and real-time path tracing.

Chapter 18 – Understanding REST and JSON

Many APIs mentioned in the previous chapter were said to be REST APIs – REpresentational State Transfer – also known as RESTful APIs. They are defined by six attributes:

- Client/server architecture – one process offers the API, the other makes the call
- Stateless operation – each API call should be processed independently, previous history of calls should not be considered
- Clear statement of cacheable/uncacheable – resources requested by an API call should indicate clearly if they should be cached and, if so, for how long.
- Uniform interface
- Layered
- Code-on-demand

REST APIs support CRUD actions: Create, Read, Update and Delete. They usually use HTTP, whose verbs (actions) map neatly onto CRUD:

- Create – POST
- Read – GET
- Update – PATCH, PUT
- Delete – DELETE

APIs need a data model in order to transform a server's internal representation of variables into something that can be sent to the client and interpreted. This process is called data serialisation and there are many data serialisation languages that serve this purpose, the primary being JavaScript Object Notation (JSON) as far as the exam is concerned. It is easy to read for both humans and machines.

JSON consists primarily of objects (an unordered set of key-value pairs) and arrays (ordered list of zero or more values). Objects can be nested, i.e. the value of a key can be another object. JSON objects and arrays correspond to Python dictionaries and lists, respectively.

The other two commonly used data serialisation languages are YAML (which is technically a superset of JSON, i.e. JSON data will be interpreted as correct YAML data) and XML.

Chapter 19 – Understanding Ansible, Puppet and Chef

Maintaining consistent configuration of network devices becomes harder as they are added to a network and then time passes. Small changes are hard to track and lead to configuration drift, i.e. differences between individual devices and the approved, reference configuration. Centralised configuration files, version control and configuration management tools can be used to solve these problems. They offer:

- Configuration provisioning – uploading configuration from a central repository to devices, allowing generation of standard configuration from templates, validating configuration before upload, automating configuration upload
- Configuration monitoring and enforcement – monitoring changes to configuration over time, accounting (who changed what), identifying and reacting to divergence between standard centralised and on-device configuration

For configuration provisioning, templates offer a way to automate configuration changes on many devices. Basically, a configuration management tool is fed a template together with sets of variables for each managed devices and spits out ready-made configuration files to deploy (upload)

Configuration management tools

Ansible: Ansible is a push-model, agentless tool written in Python. This means it does not rely on a piece of software running on target devices to call back home and report changes or ask for new configuration, but rather automatically logs into the target device using SSH or NETCONF and performs appropriate actions. Ansible configuration consists of:

- Playbooks – files defining actions to be undertaken on a device
- Inventory – Lists devices, provides information on them and groups them. Also allows setting variables per-group or per-host
- Templates – Templates for configuration files
- Variables – variables that are fed into templates to generate actual configuration files to provision

Puppet: Pull-model, agent-based tool written in Ruby. Agent programme on the device connects to the Puppet Master to report changes and check for configuration updates.

If a given device does not support running a Puppet agent, it is possible to use a proxy agent that connects to the device using SSH. Agent communicates with Puppet Master using HTTPS over port 8140.

Puppet configuration centres on Manifest files, which consists of declarative language code. Manifests are backed by Resources, Classes and Modules files.

Chef: similar to Puppet in that it is a pull-model and agent-based tool written in Ruby.

Agent communicates with the server using HTTPS over standard port 443. Chef can also work standalone, using cookbooks stored locally or in a .tar.gz archive on the Internet.

Its chief components are called Recipes – gathered in Cookbooks – while Runlists determine which recipe applies to which host.

Topics beyond OCG scope

Enhanced Interior Gateway Routing Protocol

EIGRP is no longer explicitly listed in the CCNA exam topics, but you can still be tested on it in the exam.

EIGRP uses a composite metric that can take into account the following: delay, bandwidth, load and reliability. By default only delay and bandwidth are considered.

After establishing initial adjacency, the router floods information on its routes and later sends only updates.

EIGRP uses three tables for route calculation:

- Neighbour table – lists all directly connected routers (which were discovered via Hello packets sent to multicast address 224.0.0.10).
- Topology table – contains all routes learned via EIGRP (not just the best routes which make it into the routing table). Routes are calculated using a Diffusing Update Algorithm (DUAL) that ensures loop-free paths. Each route in the topology table contains two metrics:
 - Advertised metric/advertised distance/reported distance – distance to destination route reported by the next-hop router.
 - Computed distance – distance to the destination subnet from this router, used as metric. Basically reported distance + distance to the router that advertises this route. The shortest (most preferred) computed distance to each network is called the **feasible distance**.
- Routing table – contains only the best route (or multiple routes, if equal- or unequal-cost load-balancing is in effect), i.e. route with the lowest metric in the topology table for each destination. These routes are called **successors**.

The topology table also contains **feasible successors**. They have the second best metric and are the fallback routes for successors. A feasible successor must have a reported distance lower than the feasible distance of the successor – this is called the **feasibility condition** and is enforced to ensure a loop-free feasible successor path. If the successor route fails and there is no feasible successor, the router must query neighbours for a new successor.

By default, Hello interval and Hold timer are set to 5 and 15 seconds respectively. EIGRP messages are sent using a specialised Reliable Transport Protocol at Layer 4 and can use many Layer 3 protocols – EIGRP is protocol independent and can work on other L3 protocols than IP.

Configure EIGRP:

- (config)# router eigrp *ASN* – enter EIGRP configuration context for Autonomous System *ASN*
- (config-router)# network *address wildcard* – enable EIGRP for interfaces with these addresses (same as for OSPF)
- (config-router)# eigrp router-id – manually set Router ID
- (config-if)# ip hello-interval eigrp *ASN seconds* – set Hello interval for *ASN* to *seconds*
- (config-if)# ip hold-timer eigrp *ASN seconds* – set Hold timer for *ASN* to *seconds*
- (config-if)# delay *ms* – set interface delay to *ms* milliseconds manually
- (config-if)# bandwidth *kbps* – set interface bandwidth to *kbps* kilobits per seconds manually
- (config-if)# [no] auto-summary – enable/disable automatic route summarisation

Basic EIGRP configuration example:

```
R1(config)#router eigrp 100
```

```
R1(config-router)# network 10.0.0.0 0.0.255.255
```

Display EIGRP topology:

- #show ip eigrp topology

EIGRP supports equal-cost load-balancing. By default, it will add up to 4 routes with equal metric to its routing table and balance traffic between them. To change the maximum number of routes used for load-balancing:

- (config-router)# maximum-paths <1-32>

Unlike OSPF, EIGRP also supports unequal-cost load-balancing, i.e. balancing traffic between several routes with different metric. To determine which paths can be used, it uses the **variance** parameter – all paths whose computed distance is shorter than feasible distance times variance and which satisfy the feasibility condition will be used. For variance value of 1 (default), only equal-cost balancing will be performed. To set variance:

- (config-router)# variance <1-255>

RIP Routing Information Protocol

The Routing Information Protocol (RIP) is a Distance Vector routing protocol. It uses hop count as its metric. The maximum hop count is 15. It will perform Equal Cost Multi Path, for up to 4 paths by default.

RIPv1 is a legacy protocol which is not typically used anymore (although it is still supported on Cisco routers). RIPv1 does not send subnet mask information with routing updates so Variable Length Subnet Masking (VLSM) is not supported. You can use any subnet mask you want for your different subnets, but they must all use the same mask (for example all your networks could use a /26 mask, but you can't have /24 subnets and /30 subnets). RIPv2 does support VLSM.

RIPv1 updates are sent every 30 seconds as broadcast traffic. RIPv2 uses multicast address 224.0.0.9. RIPv2 supports authentication, RIPv1 does not.

RIPng (RIP next generation) supports IPv6 networks. It is not covered on the CCNA exam.

RIP configuration example (the 'network' command should reference a classful network. No subnet mask is specified):

```
R1(config)#router rip
R1(config-router)#version 2
R1(config-router)#network 10.0.0.0
```

IOS image

There are several types of IOS images with different functionality sets:

1. K9 – includes SSH and IPsec
2. WAN – includes WAN technologies such as Ethernet over Multiprotocol Label Switching (EoMPLS), Virtual Private LAN Service (VPLS) and Hierarchical Quality of Service (HQoS)
3. NPE – export-restricted, does not support SSH, IPsec and other cryptography features
4. IP Services – full routing protocol functionality
5. Advanced IP Services – advanced IPv4 and IPv6, Layer 3 VPNs, MPLS
6. Advanced Enterprise Services – AppleTalk and ATM support

Recommended Training Course

We recommend Neil Anderson's [Cisco CCNA Gold Bootcamp](#) course as your main study material. The videos explain all the exam topics in an easy to understand way, and the lab exercises give you loads of hands on experience to get ready for a real world job as well as pass the exam. It's the highest rated Cisco course online with an average rating of 4.8 from over 30,000 public reviews and is *the* gold standard in CCNA training.



Recommended Practice Exams

Practice tests give you the best chance of passing the exam on your first try. You can have all the knowledge but still fail if you're not used to the type of questions you'll get on the exam.

[Boson's ExSim-Max](#) is universally recognized as the best set of practice exams for the CCNA. The questions exactly mimic the difficulty and style of the real exam, and include in-depth explanations for each answer. You'll understand where you went right or wrong and know when you're ready for the real exam.

