CCNA 200-301 study notes

Based on Odom Wendell's Official Cert Guide

March 12, 2021

Index

Book 1, Part I – Introduction to Networking

Chapter 1 – Introduction to TCP/IP Networking

OSI model	TCP/IP model	Protocol data unit (PDU)	Example	Equipment operating on given layer	Domain
Application	Application		HTTP	Firewall	
Presentation					
Session					
Transport	Transport	segment	TCP, UDP	Router w/NAT, firewall	
Network	Network	packet	IP	router	Subnet
Data-link	Data-link	frame	Ethernet, 802.11	switch	Broadcast domain
Physical	Physical	bit/symbol	UTP, single-mode fiber, multimode fiber	hub	Collision domain

RFC 1122, which defines a four-layer TCP/IP model, conflates physical and data-link layers into a single link layer, but it makes a lot of sense to separate the two. E.g. Ethernet can run on many types of physical medium, network layer interacts with Ethernet but does not need to know if there is copper or fibre underneath. Most authors use a five-layer TCP/IP model.

Chapter 2 – Fundamentals of Ethernet LANs

Physical layer standard	10BASE-S	10BASE-LX	10GBASE-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.

Cables used in 10BASE-T and 100BASE-T: 1,2,3,6. Crossover cable: 1→3, 2→6, 3→1, 6→2
Cables 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	
Source 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	

Ethernet over MPLS (EoMPLS) AKA Ethernet emulation – creates a separate broadcast domain!

MPLS – Multi-protocol label switching

Part II – Implementing Ethernet LANs

Chapter 4 – Using the Command-Line Interface

Simple password on console:

- (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 **OR**
- # copy running-config startup-config

Erase NVRAM (factory reset) – either command works

- # 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: Cisco always outputs MAC in three dot-separated hex quartets, not the usual 6 colon-separated hex pairs, but this command accepts both formats.
- # 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” – VTYs are used for both SSH and telnet

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 Switch Virtual Interfaces (SVI) instead of per-port interfaces (unless specifically configured on a Layer 3 switch). Each enabled VLAN has an interface associated with it.

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

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)# interface range Gi0/0 - 24 – configure a range of interfaces at once. Several comma-separated ranges can be specified 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. This will prevent typos from unnecessarily invoking telnet, but is not considered best practice. For best practice, see below
- (config-line)# transport preferred none - do not try to telnet/SSH to what looks like a domain name in the command line. This prevents typos from invoking telnet but does not disable domain lookups when they are actually needed.

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}

Autonegotiation is disabled if **both** speed and duplex settings are set manually.

If the other side has autonegotiation 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 almost (?) always be 10, the result is 10Mbits half-duplex. Cisco, instead, senses speed and uses full duplex at 1000 and higher.

LAN switch interface status:

Line status (Layer 1)	Protocol status (Layer 2)	*show interfaces status*	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, MLPS) 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 – Frames that did not pass FCS, 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)

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). One can set the protocol with:

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

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 – trunk if the other side is set to **trunk** or to **dynamic desirable**, access otherwise
- (config-if)# switchport mode dynamic desirable – initiate trunking with the other side, trunk if the other side is set to **dynamic auto, dynamic desirable** or **trunk**, access otherwise

Dynamic trunk negotiation is done using Dynamic Trunking Protocol (DTP). DTP frames travel over 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), the 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)

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

Cisco IP phones contain a small switch that *de facto* trunks towards the rest of the network and provides an access port for a PC. The voice VLAN can be set with:

- (config-if)# switchport voice vlan *n*

VLAN Trunking Protocol

VTP (VLAN Trunking Protocol) – allows a device to advertise VLAN info (IDs enabled and names) and others to receive it and update their configuration accordingly. OCG recommends disabling it, as VTP can mess up VLAN config in the whole network if misconfigured:

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

However, VTP configuration does appear on the exam, so 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 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. 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.

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 will originate VTP advertisements. In theory, this allows decentralised configuration: switch A makes changes, ups the configuration revision and propagates new configuration to other switches. 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 a switch with configuration revision higher than others (e.g. an old switch being repurposed) is attached to a network, it will overwrite VLAN configuration on all switches, And That's Terrible.

VTPv3 solves that by making one VTP server primary. 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, if set.

- # 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 Multiple Spanning Tree instance information, which is a very interesting topic well explained in the following Reddit post, on which the whole subchapter is based:
https://www.reddit.com/r/ccna/comments/6ohnqo/the_forgotten_a_vtp_mst_post/

Chapter 9 – Spanning Tree Protocol Concepts

NOTE: "switch" and "bridge" used interchangeably

Classic STP

STP (and its descendants), standardised as 802.1D, now included in 802.1Q. Allows switches to exchange information on network physical topology and agree which links to disable to ensure there are no loops. If 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 (Discarding in RSTP) – do not send or receive any frames (apart from BPDUs?) on this interface.

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

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

Each switch has a 8-byte Bridge ID (BID), consisting of a 2-byte priority field and 6-byte System ID, based on the switch's MAC address.

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

Root cost is the sum of all port costs on the way from the root bridge to the port in question. Port costs are determined by the actual link speed by default, but this can be tweaked.

All ports on the root bridge are set to a forwarding state. On non-root bridges, ports with lowest root cost (closest to root switch, in a way) are set to a forwarding state.

There are also designated ports: on a given Ethernet segment (collision domain) the one with lowest root cost (lowest BID as a tiebreaker) is set to a forwarding state. Others are blocked. In practice, since hubs are no longer used, every port facing “away” from the root switch is a designated port.

In short, frames will successfully go via a link that has a designated port on the side closer to the root switch and a root port on the other. If hubs are not used – and they are not used any more – the port closer to the root switch is always the designated port, it is the other side that determines whether a link is active or not.

Each bridge/switch sends a Hello BPDU (bridge protocol data unit) with

- Root bridge ID – ID of a bridge the sender believes to be root
- Sender bridge ID – ID of the sender
- 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.

At the beginning, 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.

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 the 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 switch
- Selection of root port
- Election of designated port
- Forwarding and blocking (discarding in RSTP) states

Differences:

- **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 ask it to flush relevant entries in MAC table to avoid loops
2. flushes relevant entries in its own MAC table to avoid loop
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 the 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 the network causing loops). Should be enabled on all access ports:

- (config)# spanning-tree bpduguard default – enable BPDU Guard on all access links by default
- (config)# no spanning-tree bpduguard default – disable BDPU Guard on all access links by default
- (config-if)# spanning-tree bpduguard enable – enable BPDU Guard on the specified interface
- (config-if)# spanning-tree bpduguard 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 bpduguard
- (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. Additionally, PortFast-enabled ports should also be configured with BPDU 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. This is most often used when adding new switches to the network. LoopGuard is used to prevent loops that might arise from connectivity failures. 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 did not take VLANs into account. Without any extensions to the protocol, the BDPUs would travel in a native VLAN and there would be just one spanning tree (CST, common spanning tree) for all VLANs. Per-VLAN spanning trees allow for better load balancing.

There are three multiple spanning tree protocols available:

- PVST+ - Per-VLAN Spanning Tree, a Cisco-proprietary extension to classic STP. One spanning tree for each enabled VLAN.
- RPVST+ - Rapid Per-VLAN Spanning Tree, a Cisco-proprietary extension to RSTP. One spanning tree for each enabled VLAN.

- MST – Multiple Spanning Tree Protocol, IEEE standard, can define an arbitrary number of spanning trees, defined in 802.1Q amendment 802.1s.

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. newpriority = floor(oldpriority/4096)
2. System ID Extension = oldpriority%4096

They also VLAN-tag the BPDUs, 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.

Configuring L2 EtherChannel

An EtherChannel is a bundle of physical links that, for all intents and purposes, act as a single link from the point of view of STP/RSTP. It also provides redundancy – should one of the bundled physical links fail, 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 autonegotiation – on the EtherChannel with ID *n*

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

Autonegotiation can be achieved with two protocols: Cisco-proprietary PAgP (Port Aggregation Protocol) and IEEE standard LACP (Link Aggregation Control Protocol). They are mostly analogous. At least one side of the link must be set to initiate autonegotiation, 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 autonegotiation
- (config-if)# channel-group *n* mode passive – Use LACP, wait for the other side to initiate autonegotiation
- (config-if)# channel-group *n* mode desirable – Use PAgP, initiate autonegotiation
- (config-if)# channel-group *n* mode auto – Use PAgP, wait for the other side to initiate autonegotiation

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

Troubleshooting EtherChannel:

Regardless whether autonegotiation was used or not, the following parameters must match across all bundled ports:

- 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 physical interfaces bundled in a given PortChannel

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 (what is the default???):
    - src-mac
    - 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 Po*N* 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

Skipped, all that needs introducing will be discussed in further notes.

Chapter 12 – Analyzing Classful IPv4 Networks

Classful addressing: 5 classes, including 3 for conventional usage:

- Class A: 1.1.1.1-126.255.255.255, unicast, 8 network bits
- Class B: 127.0.0.0-191.255.255.255, unicast, 16 network bits
- Class C: 192.0.0.0-223.255.255.255, unicast, 24 network bits
- 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)

In every subnet, the number of usable IP addresses is equal to 2^H-2, where H is the number of host bits. The lowest IP address is reserved for Subnet ID, the highest for broadcast, hence we subtract 2 from 2^H.

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) – e.g /24

Chapter 14 – Analyzing Existing Subnets

TODO: Subnet calculation mnemonics: https://www.reddit.com/r/ccna/comments/amboz5/subnetting_doesnt_have_to_be_difficult/

Part V – IPv4 Routing

Chapter 15 – Operating Cisco Routers

Differences between Cisco routers and switches:

1. On routers, IP addresses are configured per physical interface (and possibly subinterfaces), not per VLAN.
2. Routers can have an Aux port that can be connected to an external modem or phone line for emergency access.
3. Since a router does not do switching, it has no MAC table to show via **# show mac address-table**
4. Routers do routing, switches don't (L3 switches are an exception)
5. Switches use **show interfaces status**, routers use **show ip interface brief**

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

Auxiliary port:

Connected to an external modem, access can be configured in **line aux 0** configuration context.

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 – added by routing protocols
- Default gateway (AKA gateway of last resort) – used when destination doesn't match any other route

Adding static routes:

- (config)# ip route *subnetID netmask interface* – this will show up as *directly connected*

OR

- (config)# ip route *subnetID netmask nexthop* – specify next-hop IP address

Netmask given in DDN format

When two or more routes match a given destination IP address, the one with the longest prefix is selected. When two routes with the same prefix and subnet ID, the one with the lowest *administrative distance* (0-255) is selected. By default, static routes have priority over dynamic ones, but one can override it with by setting the static route *administrative distance* to a higher value using:

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

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

Of routes to the same network, only the one with the lowest administrative distance will be installed in the routing table, 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).

Adding a default route on a router:

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

When adding a static route, IOS checks whether the interface is up, when specifying an interface; or if a route to the router IP exists, 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 it, one can add ***permanent*** to the configuration command.

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.

Directly attached static routes are created when only the interface is specified. A **recursive static route** is created when only the nexthop is specified. A **fully specified static route** has both specified.

Show routes:

- # show ip route [*type*] – shows all routes, filtered by type (local, connected, static, ospf etc.), if optional argument specified
- # 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: 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

Delete a route:

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

Chapter 17 – IP Routing in the LAN

L2 switches use SVIs (Switched Virtual Interfaces) for each VLAN, routers use subinterfaces, one subinterface per VLAN on the trunk connected to the physical interface. L3 switches can use both SVIs and routed (sub)interfaces.

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

Router-on-a-stick

Routers do not autonegotiate trunking! VLANs on a trunk need to be specified manually on each subinterface:

- (config) interface Gi0/0.10 – enter subinterface configuration context
- (config-if) encapsulation dot1q *n* – where *n* is the VLAN this subinterface is associated with this subinterface
- (config-if) ip address *ipaddress netmask*

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

- (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*

In the former case, we do not specify which VLAN is the native VLAN, but I believe this does not matter, since we do not forward frames anyway (?). Not sure what the rationale for specifying the native VLAN ID in the latter case, though.

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 SVIs

A layer 3 switch is a device that does both layer 2 frame forwarding and layer 3 IP routing. It can be configured in two ways:

- SVI based routing
- Switch routed port

SVIs based routing creates a virtual interface for each VLAN and routes packets between these interfaces. To enable:

- (config)# sdm prefer lanbase-routing – reconfigure switch ASIC to make routing possible
- # write – save configuration
- # reload – reboot to allow the command above to take effect
- (config)# ip routing – enable layer 3 routing

The virtual interfaces can be then configured as described in Switch IP settings.

Troubleshooting

- has **sdm prefer** global config command been executed and the switch rebooted?
- is the VLAN enabled on the switch?
- is at least one up/up, STP forwarding, not VTP-pruned switch port assigned to the VLAN
- has the VLAN been shutdown?
- has the VLAN interface been shutdown?

Layer 3 switch routed port

A layer 3 switch port can be configured as a routed port. In such a state, it will not perform layer 2 forwarding and only route packets. Only one such port can be connected to one subnet (unless using L3 Etherchannel?). To configure a routed port:

- (config-if) no switchport – disable L2 logic on the port
- configure IP as usual

Routed ports work well for point-to-point links between devices.

Layer 3 Etherchannel

Two or more point-to-point 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 autonegotiation:

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

Configure PortChannel interface:

- (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
- configure IP as usual

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 both you need a route to the pinged host and the pinged host needs a route to you.

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

traceroute – 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).

traceroute without parameters works analogously without parameters.

IOS traceroute uses IP packets encapsulating UDP datagram, while most OS use ICMP echo requests. They may be thus differently impacted by traffic 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 routes, pick best routes and notify each other of changes in network topology

- 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
- EGP – Exterior Gateway Protocol – class of protocols designed to work between AS
- RIP – Routing Information Protocol. IGP. Exists in two versions: RIPv1 and RIPv2. RIPv2 supports VLSM, RIPv1 does not.
- IGRP – Interior Gateway Routing Protocol. IGP, Cisco-proprietary
- OSPF – Open Shortest Path First. IGP. OSPF means OSFPv2 if not stated otherwise
- OSFPv3 – OSPF for IPv6
- EIGRP – Enhanced Interior Gateway Routing Protocol. IGP, Cisco-proprietary, but specification has been published
- IS-IS – Intermediate System to Intermediate System. IGP
- Metric – weight attached to each route to decide which is best

Administrative distance – When comparing routes to the same subnet learned from different sources (both dynamic vs. static routes and routes learned from different routing protocols), the lowest administrative distance wins. Used because it is impossible to compare metrics between routes learned from different routing protocols. To set administrative distance for a static route (by default 1, preferred to all dynamic routes):

- (config)# ip route *network netmask routerIP administrative_distance*

To set administrative distance for routes learned from a particular routing protocol:

- (config-router)# distance *adm_distance*

IGP types:

Algorithm	Metric	How it works?	Examples	Notes
Distance-vector	Number of hops	Routers exchange their routing tables	RIPv1, RIPv2, IGRP	Slow convergence, possibility of loops
Advanced	Number of hops	Routers	EIGRP	Solves problems

distance-vector	+ latency + bandwidth	exchange their routing tables		of distance-vector protocols
Link-state	Sum of costs of links in a route	Routers exchange information on all their links	IS-IS, OSPF	

OSPF fundamentals

Routers using OSPF exchange *link-state announcements* (LSA) between each other in order to build a complete map of the network's topology. LSAs are held in *link-state databases* (LSDB). Routers send LSAs to their neighbours who forward them to their neighbours until every router in the network has received them – this process is called *flooding*. Based on that, each router uses the Dijkstra algorithm to calculate which routes are best and insert these routes into its routing table.

Each router has a 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 relationships 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 RID 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 database descriptions (lists of which LSA each router has) and then they exchange those 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 (e.g. Ethernet) network all routers elect one of them to become a *designated router* (DR) and another to become a *backup designated router* (BDR). The rest are called DROther. All routers become full neighbours with the DR but not with each other. BDR takes over should DR fail, in which case another BDR is elected.

Routers **calculate the 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 speed, but can be set manually.

When OSPF insert 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.

Progression of neighbour states: Down → Init → 2-Way → Exstart → Exchange → Loading → Full

IPv6 OSPFv3

OSPFv3, an IPv6-enabled version of OSPF, differs in one important detail. While OSPF on IPv4 uses a multicast address to establish neighbour adjacency, OSPF on IPv6 uses link-local addresses (FE80::/10). – RFC 5340 begs to differ?

Multi-area OSPF design

Larger networks are split into several OSPF areas to reduce network chatter, convergence times and processing time and memory usage of SPF calculations. Several rules:

- Areas need to be contiguous
- Interfaces facing the same subnet need to be in the same area.
- When a router has all its interfaces in a single area, it's an internal router.
- Routers that span areas are called *Area Border Routers* (ABR).
- All areas other than the backbone (area 0) need to be connected by an ARB to the backbone (not through another area).

In a multi-area OSPF design routers have to calculate routes for their own area only. Beyond that, they need to know which subnets are in which area, which ARBs in their own area lead to which area (if in non-backbone area, routers only need to know which ARBs lead to area 0) and how to route to these ARBs. The router knows the route to an ARB and the ARB then routes packets to destination in its own area or to an ARB leading to the destination area.

This ties with LSA types:

1. Router LSA – Describes router ID, interfaces (including status), IP addresses and masks
2. Network LSA – Describes DR and BDR IP addresses, subnet ID and mask
3. Summary LSA – Describes subnet ID and mask as well as RID of ABR that advertises this LSA

Router and network LSAs put together fully describe an area. This description is then distilled by an ABR into Summary LSAs that is advertised in the neighbouring area and propagated to all areas eventually.

Chapter 20 – Implementing OSPF

OSPF can be enabled in two ways. First, it can be enabled for a given range of networks, so that all interfaces connected to one of these networks will automatically have OSPF enabled. Second, it can be enabled per-interface.

To enable for a range of networks:

- (config)# router ospf *process-id* – enter ospf router configuration mode. *process-id* can be any number from 1 to 65635 (this is to enable multiple OSPF processes)
- (config-router)# network *ip-address wildcard-mask* area *area-id* – enable OSPF for all networks within the range. A wildcard is basically a reverse of a subnet mask, but it does not need to be continuous. *area-id* should equal 0 in a single-area setup.

To disable the above:

- (config)# router ospf *process-id* – enter ospf router configuration mode.
- (config-router)# no network *ip-address wildcard-mark* area *area-id* – remove previous **network** commands, disabling OSPF for these networks.

To use a per-interface OSPF setup:

- (config-if)# ip ospf *process-id* area *area-id*

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 we know there is no other router on the connected subnet. 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 makes router 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

OSPF metric can be changed in one of the following ways

- (config-if)# ip ospf cost *n* – set the cost of this interface to *n*
- (config-if)# bandwidth *speed* – set the notional speed of the interface to *speed* in **Kbits** to change calculated cost
- (config-router)# auto-config reference-bandwidth *speed* – set the reference speed of the interface to *speed* in **Mbits** to change calculated cost

Unless cost has been set manually, it is calculated as *reference-bandwidth/bandwidth*

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, the 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. 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 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 (most common) and FC
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) is.

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 – 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 addres, *i* is taken from the MAC address with one bit flipped. FFFE 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.

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 used for unicast traffic that should not be routed (e.g. NDP, IPv6 equivalent of ARP) and are automatically generated. They are also used for the next-hop router.

Link-local addresses are formed by adding EUI-64 Interface ID to 64 bit prefix of FE80::. While the link-local address range is FE80::/10; the address can begin with FE8, FE9, FEA or FEB, in practice these addresses always start with 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 all interfaces that have at least one other IPv6 unicast address configured. Alternatively, one can use:

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

Links with only link-local addresses will work for routing purposes!

Multicast addresses

Always begin with FF and are used extensively in place of broadcast addresses, which were used by IPv4. 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

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 – as in loopback?

Link-local	FF02	Packets won't be routed
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 the prefix of **FF02::1:FF**00:0/104 (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 configure the same IPv6 address and provide a service. Routers will then route packets to the nearest host with the address.

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 up again.

Routing command 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* – since connected and local routes are not created for link-local addresses, 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 a 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 **Neighbour Solicitation (NS)** message to a specific multicast address (see Chapter 24) to which the host with the target IPv6 (matching the last 6 hex digits) should be subscribed. The target host responds with a **Neighbour Advertisment (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 FF02::2 (all-routers) multicast address. All routers should respond with a **Router Advertisment (RA)** message to the soliciting host's unicast address or to FF02::1 (all-hosts) multicast address (why the second option in response to RS?). Routers can also announce themselves to FF02::1 (all-hosts) multicast address. RA contains, among others, 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

An **access point (AP)** provides **Basic Service Set (BSS)**, advertising its capabilities and managing who can join. BSS is identified by **BSSID**, which is equal to the access point's radio's MAC address. A BSS is known under its **Service Set Identifier (SSID)**, which is a human-readable name for the wireless network. A single AP can support multiple **BSS** each with a different BSSID and under a different **SSID**.

A BSS uses a **Distribution System (DS)**, usually Ethernet, to allow communication with the rest of the network. The access point acts as a layer 2 bridge, forwarding between frames from wireless network to wired and the other way round. Multiple SSIDs can be mapped to multiple VLANs. Multiple **BSS** can operate with the same **Extended Service Set** (**ESSID)** if they have the same configuration and are ultimately connected to the same infrastructure. 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 topologies include:

- 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.
- 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.

Wireless uses 2.4 GHz and 5 GHz frequency bands. The 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). 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 the 2.4 GHz band propagates further and penetrates obstacles better, but the 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

There are three types of wireless frames:

- Data – carries data through the wireless network. There are two main types, of which contention-based service frames is the dominant:
    - Contention-based service frames: Uses Distributed Coordination Function to coordinate access to wireless medium, relies on RTS and CTS control frames to avoid interference and hidden node problem.
    - Contention-free service frames: use Point Coordinator Function to manage access to medium, provides QoS. Rarely used in practice
- 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. The standard one would be **autonomous AP architecture**, where APs bridge BSSs with VLANs. Each AP is managed individually. This architecture does not scale well, as VLANs need to span as far as the SSIDs, which might be much more than desirable for a single subnet.

Cloud-based AP architecture is an extension of the **autonomous AP architecture** with a central management platform (e.g. Cisco Meraki or the deprecated CiscoWorks WLSE) taking 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 function of an autonomous 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 monitor channels, manage association and roaming, authenticate clients, manage security and do 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. Therefore, the LAPs do not need to be in the same subnet as WLC and do not need to be connected to a trunk link with all the VLANs its BSSs are bridged with – only the WLC needs that. 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 TCP 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.

Beyond uncoupling physical and logical network structure, 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. Beyond that, one would need more WLCs. WLCs can also be 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 AP. 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 BSSs to wireless clients, scan channels and monitors traffic in spare time
- Monitor – does not transmit, act as a sensor: scan channels and monitors traffic
- FlexConnect – local but will fall back to Autonomous AP mode if its CAPWAP is brought down, see below.

- Sniffer – Sniffs wireless network traffic and forwards it to a traffic analyser (such as Wireshark)
- Rogue detector – detect rogue devices (those whose MACs appear both on wired and wireless networks – why?)
- Bridge – acts as a point-to-point or point-to-multipoint bridge
- Flex+Bridge – FlexConnect+mesh network
- SE-Connect – Spectrum analysis to detect sources of interference

FlexConnect and FlexConnect ACLs

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

FlexConnect ACLs are used to manage access control of locally switched data. They are applied per-AP and per-VLAN and can be applied to native VLAN, unless configured via FlexConnect Group. Unlike traditional ACLs, it is not possible to set the direction for ACL, they are applied at interface's ingress or egress. Like traditional ACLs, they have an implicit **deny any** at the end.

Chapter 28 – Securing Wireless Networks

Several basic concepts related 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**. 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, the frame is dropped.

Authentication methods:

- Open authentication – any client can authenticate.

- Shared key authentication/WEP – used with deprecated and absolutely broken **Wired Equivalent Privacy (WEP).** AP sends a challenge that the client needs to respond to using a pre-shared key. If this client passes, it is authenticated. In fact, 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 an 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 fudge can get around it. LEAP is now as deprecated as WEP is.
- EAP-FAST – Cisco proprietary. The AS generates a 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 then used to authenticate the end user via, e.g. 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). Possibly other methods?
- 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 (requires?) 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 IP.

WLC Ports:

- Service ports: used for out-of-band management, initial boot and system recovery, connected to an access link
- 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 DSPs can be aggregated in a Link Aggregation Group (LAG), which operates as an EtherChannel **without autonegotiation**.
- Console port: out-of-band management, initial boot and system recover, serial connection (9600 baud, 8 data bits, 1 stop bit, by default)
- Redundancy port: connected to a second WLC to achieve redundancy, failover and high availability (HA)

WLC interfaces:

- Management IP address: used to access the WLC's web interface and SSH and to handle 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 (handling roaming in the same network???) should have the same Virtual IP address.
- Service Port IP Address: bound to service port
- Dynamic interface: connects VLAN with a WLAN, Used for client data and are user-defined. Will also be used for DHCP relay

Cisco WLC 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 uses a CLI interface, even if 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 flag
- Server responds with a TCP segment with SYN and ACK 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 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

For data reliability, TCP uses sequence and acknowledgment numbers. In a three-way 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 is supposed to resend 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 autodiscovered, but this sometimes fails, potentially

leading to hard-to-debug connection problems (e.g. the initial handshake succeeds, but the connection fails when actual data is sent).

Chapter 2 – Basic IPv4 Access Control Lists

There are two types of ACLs:

- Standard – match just the source IP. Ranges 1-99, 1300-1999
- Extended – match source and destination IP, transport protocol and port number. Ranges 100-199, 2000-2699

ACLs can be either named or numbered. To add a rule to a numbered standard ACL (this will create the ACL if it is empty):

- (config) access-list *ID* {*permit* | *deny*} *source_ip* [*source_wildcard*]

For a standard ACL, the matching parameter will be just source IP or source subnet. Wildcards are used instead of subnet masks in the latter case. ACL rules are evaluated one-by-one until a match is found, the rule (permit or denied) is then applied and processing stops. d*eny any* is the default rule if no match is found.

ACLs need to be then applied to an interface. Note, ACLs can be applied to incoming or outgoing traffic - a maximum of one ACL in one direction.

- (config-if) ip access-group *ID* {*in* | *out*}

If multiple ACLs for the same protocol applied to an interface in the same direction, only the last one will be actually used.

Chapter 3 – Advanced IPv4 Access Control

Extended ACLs can match protocol (IP/TCP/UDP/ICMP etc.), source IP and destination IP, port numbers. They have a slightly different syntax to accommodate the new matching criteria:

- (config) access-list *ID* {*permit* | *deny*} *protocol source* [*operator source_port*] *destination* [*operator destination_port*] [log | log-input] [established] – **log** and **log-input** parameters enable logging of matches against the entry, the latter option adds the input interface to the message. **established** argument matches established connections.

Unlike standard IPs, extended ACLs require the use of **host** when matching a specific IP address. Keyword **any** can be used to match all IP addresses. ***Source*** and ***destination*** can be either ***address wildcard*** or **host *address***

source_port and *destination_port* need to use an *operator* that tells 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.

Extended ACLs are applied to interfaces identically to standard ACLs:

- (config-if) ip access-group *ID {in* | *out}*

New-style configuration, editing and named ACLs

In order to edit ACLs, one needs to use new-style configuration. This also allows named ACLs.

Instead of:

- (config) ip access-list 1 *rule*

one can use:

- (config) ip access-list {extended | standard} {*name* | *number*}
- (config-ext-nacl) *rule* – assuming we have selected “extended” in the command above

Rule syntax is identical. When applying a named ACL to an interface, just swap out ID for name:

- (config-if) ip access-group {*name* | *number} {in* | *out}*

Rule can be removed from an ACL using the **no** keyword:

- (config-ext-nacl) no *rule*

ACLs also use sequence numbers to identify individual rules. Rules are checked in the order of their sequence numbers, so it’s possible to insert a new rule early in the ACL.

- # show ip access-list [{standard | extended} {*name* | *number*}] – display new-style ACL content
- (config) ip access-list {standard | extended} {*name* | *number*} – enter ACL configuration context
- (config-ext-nacl) no *seqn* – remove rule with sequence number *seqn*
- (config-ext-nacl) *seqn rule* – insert rule with sequence number *seqn*

Part II – Security Services

Chapter 4 – Security Architectures

Vulnerability→Exploit→Threat→Attack

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 realised.

Typology of attacks:

- 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 host.
- Reconnaissance attack – discovering IP addresses and ranges in use, services available etc.
- ` overflow attacks – can allow running of malicious code with permissions of attacked service
- Privilege elevation
- Malware
- Human attacks
- 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

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 (please refer to Book 1, Chapter 6) or via an AAA server, with the latter being the preferred method. With local authentication, password are stored in cleartext by default, which is obviously wrong. It is possible to encrypt password:

- (config)# service password-encryption

but this is *de facto* only obfuscation, as it is perfectly possible to decrypt such a password.

- (config)# no service password-encryption

disables encryption but does not decrypt the password into cleartext (not sure if it is usable in such a case). When encrypted, passwords will show up with “5” after the **password** command and before the crypttext in show running-config.

Instead of encrypting passwords, one should always hash them. This is done automatically with the **enable secret** command. NOTE: is both **enable secret** and **enable password** are configured, the latter will be ignored. **enable password** does not hash the password!

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, might be better:

- # enable algorithm-type sha256 secret *password* – use Type 8 PBKDF2 hash
- # enable algorithm-type scrypt secret *password* – use Type 9 Scrypt 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	scrypt	Scrypt hash, recommended

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

It is possible to limit where a user can login from via Telnet/SSH using access lists.

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

It is also possibly to limit outbound connections by swapping **in** for **out**, but this mostly is just a curiosity.

Finally, IOS support privilege level from 0-15. To specify a password for a particular privilege level:

- # enable secret level *n password*

Firewalls and Intrusion Prevention Systems

Traditional firewalls combine stateless (ACLs, URIs) and stateful inspection. Firewalls are stateful in that retain memory of previous packets and make decisions about future ones based on that – this makes it possible, for example, to detect a TCP SYN flood and filter traffic coming from the attackers. Intrusion Prevention Systems (IPS) complement firewalls and match incoming traffic against a database of exploits in order to detect incoming attacks.

Zone is a concept used by firewall 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 being heavily filtered. A DMZ (demilitarized zone) is used for hosts that need to be accessible from the public Internet.

Next-generation firewalls (NGFW) additionally perform deep packet inspection using **Application Visibility and Control** to classify traffic without relying on port numbers. They can run several security services in tandem (Advanced Malware Protection), filter URLs based on a database of domain scores. Finally, NGFW can 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.

This subchapter is very cursory and reads like a half-baked marketing material.

Chapter 6 – Implementing Switch Port Security

Port security monitors and restricts Layer 2 traffic on a per-port basis. When enabled, it maintains a list of source MACs sending traffic through a given port and limits the number of source MACs that can use the port. It can also 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 autonegotiation)
- (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 learn 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 **running-config**

NOTE: Port security can be also enabled for voice ports and EtherChannels. On voice port, the maximum should be set to two (possibly more). When applying port security to an EtherChannel, please 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 ,including the number of violations and source (MAC and VLAN) of last violation, of an interface (is *ifname* optional?)
- #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 define 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.

TODO: Violation counter

Chapter 7 – Implementing DHCP

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. In this case, most subnets will not have a DHCP server. Instead, DHCP messages will be passed from clients to the server and back by a DHCP relay, usually one of the routers on the subnet. While ordinary DHCP messages use 0.0.0.0 and 255.255.255.255 as source and destination addresses (respectively), DHCP relay will pass the message with the IP address of the interface on the relevant subnet as the source address and set **giaddr** field to the same value to indicate which subnet the request is 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*

Configuring router or switch as a DHCP client:

- (config-if)# ip address dhcp
- (config-if)# no shutdown – may or may not be used depending on the device
- #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.

TODO: Windows and Linux commands

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

Chapter 8 – DHCP Snooping and ARP Inspection

DHCP Snooping

DHCP Spoofing can be used by rogue DHCP servers to provide clients with wrong default gateway and route their traffic via themselves (Man-in-the-middle attack, MITM). DHCP Snooping divides ports into trusted and untrusted and analyses traffic on untrusted ports to detect DHCP Spoofing attacks:

- 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 on untrusted port 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

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)

DAI is used to prevent ARP Poisoning attacks, in which an attacker sends gratuitous ARP Replies with its own MAC and victim's IP address. If the attack succeeds, other hosts will send traffic destined to the victim to the attacker.

DAI utilises the DHCP binding table created by DHCP snooping with static entries added to cover hosts not using DHCP. The idea is that 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 (check if the ones in Ethernet header those in ARP message) and IP addresses.

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.

Show DAI state and statistics:

- #show ip arp inspection statistics

DAI can be CPU intensive and thus can be rate limited, not unlike 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 – for users logged in via console.
- 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. Note that **host** is replaced by **trap** in this command

Log messages severity levels (Level is actually 0-7):

0 - Emergency

1 - Alert

2 - Critical

3 - Error

4 - Warning

5 - Notification

6 - Informational

7 - 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)

NTP is used to synchronise clocks between different hosts. Some hosts use very precise clocks to offer time to other hosts. These hosts in turn synchronise with further hosts. NTP uses stratum 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 stratum number themselves, host getting the time from others set the stratum number to one higher than their source.

Cisco devices, when set as NTP clients (getting time from NTP servers), automatically 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*

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*

To set time:

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

To show time and settings

- # show clock – 'nuff said

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

NTP can be configured on a loopback device – please note that this is not the usual loopback address 127.0.0.1 (and others) but rather a virtual interface that is always up (unless manually shutdown) and handles requests at the assigned IP address. This is helpful in case a physical interface on which a host listens to NTP fails but the host itself is still reachable. In this case, clients will fail to connect to the failed physical interface IP address, but will still be able to connect to the loopback IP, provided they know the route.

To configure 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 an 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 a pair of two protocols serving the same purpose – discovery of basic information on neighbouring network devices. Their logic is similar, as is configuration syntax, but there are differences in capabilities and terminology. CDP is Cisco-proprietary, offers a bit more information than LLDP, at least when using Cisco equipment, and is enabled by default on IOS, unlike LLDP.

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, discover only immediate neighbours – switches and routers – and are not forwarded further.

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)# cdptimer *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

TODO: Source NAT vs Destination NAT, masquerade

Network Address Translation (NAT) is a set of techniques that allow hosts with private IP addresses, called inside local IP addresses, to connect outside their network. In essence, these “internal” hosts send the packets to the router which resends them with its one of its own public IPs, called inside global addresses, as the source IP and saves a mapping, so

that when it receives a reply, it knows to forwards it back to the internal host at the inside local address. There are three main types of NAT:

- Static – maps inside local IP addresses to inside global IP addresses based on static configuration.
- 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. It is the most commonly used type and helps with IPv4 address space exhaustion.

Static NAT configuration:

- (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 nat inside source static *inside_local inside_global* – create a static mapping. This command needs to be issued separately for each mapping.

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 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 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

Dynamic NAT with overload configuration is similar to regular dynamic NAT, but instead of creating a pool of inside global addresses one specifies just an interface to use and adds an **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.

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)

PHB – per-hop behaviour

Diffserv – Differentiated service

QoS is a set of tools and protocols used to manage:

- bandwidth – including reserving bandwidth for various types of traffic
- delay – one-way or round-trip
    - one-way - Amt. of time it takes traffic to go from source to destination
    - round-trip - Amt. of time it takes traffic to go from source to destination and return
- loss – proportion of packets lost on the way to target
- jitter – variations in one-way delay between subsequent packets of same application

The requirements for QoS differ between different types of traffic. 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.

For interactive audio quality it is recommended:

- One-way delay is 150 ms or less
- Jitter is 30ms or less
- Loss is 1% or less

Classification and marking

Classification of packet is done by matching specific fields in packets processed by router/switch in order to take specified actions on them. QoS tools can be applied to incoming or outgoing traffic – the latter after forwarding decision has been made.

It is perfectly possible to apply ACL logic to packet classification. For more complex classifications, Network Based Application Recognition (NBAR) is used by Cisco, inspecting packets more deeply in order to classify them.

The best practice is to do classification early in packet's life and mark it by setting an appropriate header. This greatly simplifies configuration, as devices further down the line can rely on these markings rather that performing full classification themselves. There are several schemes for packet marking, working on Layer 2 and Layer 3:

- **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).
- IP Precedence (IPP), 3-bits in the Type of Service field of an IPv4 header (Layer 3), superseded by DSCP.
- 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
- TIP, 3-bits in an 802.11 header of a wireless frame (Layer 2)
- EXP, 3-bits in an MPLS label (between Layer 2 and Layer 3)

End devices shouldn't be trusted with marking packets, as it would allow them to unduly prioritise their own traffic. The trust boundary should be set at the access switch (exception: trust boundary should be extended to IP phones) – markings on packets incoming from beyond the trust boundary will be ignored.

To extend trust boundary to messages coming from a given port:

- (config-if)# switchport extend priority trust – instruct the IP phone to trust COS header on packets coming from its PC port
- (config-if)# switchport extend cos *n* – instruct the IP phone to mark packets incoming from the attached host with the specified COS value.
- (config-if)# mls qos trust cos – extend trust to packets coming in from the port
- (config-if)# mls qos trust extend cos *n* – instruct the IP phone to mark packets incoming from the attached host with the specified COS value. NOTE: only available on some devices (can't find good summary of this vs. **switchport extend priority trust**).

DiffServ (differentiated services) define 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 voice transmission.
- 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 follow the format of AF*XY*, where *X* is the queue (1 – worst, 4 – best) and *Y* is drop priority (1 – best, 3 – worst). AF 41 would mean the best queue and the best drop priority.
- Class Selector – 8 values (text value: CS*X*) used to map old IPP values to DSCP values.

Examples of DSCP values:

- EF – Voice data
- AF4y – Interactive video
- AF3y – Streaming video
- AF2y – High-priority data
- CS0 – Standard data

Queueing

Classification and marking on their own do not affect how packets are forwarded. They are, however, used to assign packets to queues which in turn impact how and when they are forwarded. A standard queue uses a FIFO scheduler – the packet that arrives first in a given queue is sent first. Weighted round-robin scheduler – e.g. Class-Based Weighted Fair Queuing (CBWFQ) is commonly used to prioritise particular queues – in a given time period, each queue is guaranteed a minimum share of bandwidth.

This can be overridden by a low-latency queue, which will have priority over others regardless of assigned minimum bandwidth share. Such a configuration guarantees low-latency to high-priority data while still allowing other traffic to come through. The only problem is that if there is enough high-priority traffic, it can completely pre-empt low-priority traffic – to prevent this one can set a maximum share of bandwidth available to low-latency queue using **policing** mechanism (see below) or more sophisticated tools such as Call Admission Control (CAC), which will prevent calls from being made if they would oversaturate the link.

Shaping and policing

Shaping and policing are two mechanisms which can be used to monitor and manage the amount of traffic. Both operate with a configured bit rate and, when it is exceeded, act to bring the amount of traffic down. Policing does this by simply discarding packets exceeding the limit or marking them so that they are de-prioritised by devices down the line – this is used e.g. by ISPs on the ingress of to their networks to make sure subscribers do not exceed contracted rate (discarding packets) or that one subscriber does not starve others of bandwidth (marking – this allows a subscriber to exceed set rate if conditions allow). Shaping, on the other hand, limits the rate at the sender and queues packets, prioritising traffic as configured (see **Queuing** above)

Congestion avoidance

Congestion avoidance, instead, work by dropping packets to signal to the sender to slow down – when packets are dropped, TCP window shrinks and transmission slows down. This can be done in various ways. The easiest one is tail drop: when queue is full, further incoming packets are dropped until queue is no longer full. The problem is that once the queue fills up, all flows will be impacted and global synchronisation will occur – all senders will reduce their transmission rate at the same time, leading to link underutilisation, after which they will 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. RED randomly, at a rate depending on how much of the queue has filled, drops packet 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. WRED is an extension to RED that takes into account packet priority. It will more often drop packets with lower priority, as determined by DSCP or IPP field in IP header or possibly by other means, such as Resource Reservation Protocol (RSVP), thus leaving more bandwidth for high priority 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 protocol

First-hop redundancy protocols (FHRP) serve to eliminate the single point of failure created by the use of default gateways and provide redundancy and failover. Since each host can have only one default gateway configured at a time, when that router fails, hosts will lose connectivity. There are several such protocols, the exam requires you to know Cisco-proprietary Hot Standby Router Protocol (HSRP). In broad strokes, HSRP works by having two or routers that can act as a default gateway – one of them selected as **active** and the rest selected as **standby** or **passive** (same thing, different name). Each router will have its own IP address, but the **active** router will also use a virtual IP address and an associated virtual MAC address. It is this virtual IP address that will be used as the default gateway for hosts in the network. 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!

Additionally, HSRP allows per-subnet load balancing. Multiple virtual IP addresses are used, one for each subnet, and assigned to routers in order to balance load between them. Each subnet uses a different address as its default gateway.

The other FHRP protocols used are Gateway Load Balancing Protocol and Virtual Router Redundancy Protocol. Gateway Load Balancing Protocol uses one **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 act as mere backups. There is one virtual IP used for routing, but AVG and each AVF have its own virtual MAC. AVG responds to ARP

requests for the virtual IP: sometimes with its own virtual MAC, sometimes with the virtual MAC of one of 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 conceptually to HSRP: one **master router** and multiple **backup routers**.

MAC addresses and multicast group addresses (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 setting 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, some are specific to a particular device. Agents exchange information with SNMP **managers**, typically **Network Management Stations** (NMS), 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** and 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 ID (OID) that can be given as either numbers or names. E.g.

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

1.3.6.1.4.1.9.9.10

Part of 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:

- SNMPv1 and SNMPv2c – Communities, i.e. pre-shared passwords exchanged in cleartext between agent and manager.

- SNMPv3 – Authentication with username and hashed password, message integrity and (optional) encryption.

FTP, FTPS, TFTP, SFTP and IOS filesystem

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

Types of filesystems:

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

Filesystem navigation and management:

- # show file systems – show filesystem
- # 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 one 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, selects a random port for data connection, starts listening on it and then tells the server to open a data connection to this port using the PORT command. This will not work if the client is behind NAT (at least not without extra configuration and packet inspection), as is most often the case.

Passive mode solves this: after initiating 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 PORT command. The client then initiates a data connection to this port. This still requires NAT/firewall configuration on the server's side, but this is easier to achieve than on the client's side.

Please note that while port 20 is assigned for FTP data connection, it is only used in active mode: FTP server initiates data connection **from** port 20 to the port indicated by the client. An additional, but almost always disabled, feature of FTP is that a client can open control connections to two different FTP servers and tell one server to open a data connection to

the other (this is because the PORT command indicates both the port number and the IP address for such connection). This would allow the client to transfer files between two servers without using its own bandwidth, but the feature is disabled by nearly all FTP servers as it can be used for FTP bounce attacks (e.g. scanning the network from the point of view of one of the servers).

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 and 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 and is almost always handled by an SSH server. Nice, elegant, but possibly slower than FTP due to protocol differences and increased complexity.

Trivial File Transfer Protocol (TFTP) is an extremely stripped down protocol allowing for file retrieval and upload and nothing beyond it. For simplicity, it uses UDP (port 69) and handles error recovery at the application layer in a very basic way, resulting in slow transfer speeds. It is used for embedded systems and bootstrapping.

Part IV Network Architectures

Chapter 13 – LAN architectures

Campus LAN – LAN in a single building or several buildings in proximity.

Two-tier campus design is also known as collapsed core (since the function of core layer is taken over by distribution layer) consists of distribution layer (forwarding traffic between network devices) and access layer (forwarding traffic between user devices and rest of the network).

The distribution layer usually creates a partial mesh – not all pairs of devices are connected to each other, but there are more connected pairs than in star topology. Partial mesh provides some redundancy, unlike star topology, while being more scalable than full mesh. Access switches can and often do connect to two or more distribution switches for redundancy.

A three-tier campus design includes a core layer at the centre of design. Core layer switches forward traffic between different parts of the distribution layer which, in this design, do not need to be connected to each other directly. This kind of design requires fewer cables and switch ports and, in particular, reduces the number of required links between buildings, which can be particularly expensive. Core layer switches need to be able to handle large volumes of traffic and have to be extra reliable.

A spine-leaf architecture is a data center network topology that consists of two switching layers—a spine and leaf. The leaf layer consists of access switches that aggregate traffic from servers and connect directly into the spine or network core. Spine switches interconnect all leaf switches in a full-mesh topology.

Finally, in a SOHO environment, a very simple design is 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 in star topology and autonomous APs.

Power over Ethernet (PoE)

PoE allows one network device, called Power Sourcing Equipment, to provide power to another device, called a Powered Device, via UTP cabling. PoE provides for autonegotiation to determine PoE power class, i.e. how much power to supply. Monitoring and power level adjustment via aforementioned autonegotiation and 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

https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/12-2/53SG/configuration/config/PoE.html#wp1087217

To protect against a PD attempting to draw more power than it was assigned, one should enable power policing:

- (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. One can configure automatic recovery from such a state:

- (config)# errdisable recover cause inline-power

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

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

To show power policy:

- #show power inline police

Chapter 14 – WAN architecture

Metro Ethernet (MetroE)

Metro Ethernet is a label for WAN services that use Ethernet links to connect to the consumer. From a consumer's point of view, such a WAN service looks like a large Ethernet switch. The service provider (SP) needs to physically install a switch in a **Point of Presence (PoP)** near customers – near enough to allow Ethernet connection from customers to the switch. The links between customers and PoP are called (Ethernet) **access links**, which form **user network interfaces (UNI)**. From user/customer perspective, SP's network beyond UNI is opaque – it promises to deliver Ethernet frames across the WAN, but its internals are irrelevant and hidden. Since Metro Ethernet provide Ethernet service, it is possible to connect to it via a switch instead of a router.

Some 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 WAN can operate in one of three topologies, as defined by MEF:

- 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)

MPLS provides a Layer 3 service (in contrast to MetroE Layer 2 service). A service provider creates a large IP network and connects all of its customers to it. Routers at the edge insert an MLPS header between Layer 2 and Layer 3 headers that tells the rest of MLPS network how to forward these packets, creating in effect many separate MLPS VPNs (unencrypted) to keep each customer traffic separate.

Customer edge (CE) and provider edge (PE) – devices sitting at the edge of the respective physical networks.

The advantage of MPLS is that, since it is a Layer 3 service, it can use a variety of Level 2 access links, unlike MetroE which requires Ethernet at access links. Furthermore, MPLS enables the use of QoS: customer edge can mark outgoing IP packets to enable MPLS network to recognise and prioritise them.

Looking at Layer 3, CE routers become neighbours with their respective PE routers but not with each other. CEs and PEs can exchange routing information using e.g. OSPF, while internally MPLS network uses MP-BGP (Multiprotocol Border Gateway Protocol) which redistributes OSPF learned routes into the MPLS network. PE routers will become next-hop routers for their respective CE routers.

Internet VPNs

Private WANs are not always available or cost effective. Offices can use cheap consumer access links 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 and prevent replay attacks. VPN client encrypts packets, adds VPN header and encapsulates the bundle in an IP packet that is sent over public Internet to VPN server that de-encapsulates, decrypts the VPN packet and forwards the encapsulated packet to its network – the process of course also happens in reverse, i.e. from server to client.

VPNs can be used for site-to-site connection (permanently connecting various branches) or for remote access (on-demand connections by remote users to the network). For site-to-site IPSec with shared key encryption over the GRE tunnel is often used. IPSec deals with data confidentiality and data integrity, GRE deals with tunneling, including multicast and broadcast encapsulation. Remote access VPNs are usually TLS based, Cisco's solution is called AnyConnect, but there are many others, including OpenVPN (and derivatives) and Wireguard.

Chapter 15 Cloud Architecture

Server virtualisation means services are not run on bare metal server but instead on virtual machines (VMs) of which several can run on a single physical hosts. VMs are assigned a share of hosts' resources (CPU threads, RAM, disk space, network bandwidth) and directly managed by a hypervisor. A hypervisor will run a virtual switch (eg, Cisco Nexus 1000VE, which is end-of-sales, or Cisco Application Virtual Switch). Physically, hosts are typically mounted in racks with two switches (for redundancy) at the top of the rack (ToR) connected to hosts in a given rack. Racks are arranged with rows with switches at the end of row (EoR) connected to ToR switches at other racks.

Traditionally, customers relied on data centre staff to manage (add, edit, remove) VMs. Cloud computing takes one step further by allowing 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 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 (cloud consumer == cloud provider), a public cloud is a service provided by a third-party (cloud consumer =/= cloud provider). 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

In order to describe controller-based networking and network automation it makes sense to divide networking functions into 3 planes:

- Data plane – receiving, processing and forwarding messages, including (de-)encapsulation, MAC address lookup, IP routing, NAT etc.
- Control plane – overhead activities that impact frame/packet forwarding, such as STP, OSPF, ARP/NDP etc.
- Management plane – managing network devices, changing configuration, monitoring status and logs.

In a traditional networking model, all functions on all these planes are decentralised. 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 then via this router", etc.) and act on it when forwarding messages. This is a marvel of engineering, but is also difficult to configure and fragile. Data plan 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 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 even take control of many control plane functions. It then communicates with network administrators or other 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 of control plane functions, while allowing for various APIs to be used for NBI. OpenDaylight (ODL) is the most successful 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 and 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 of 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 other spine switch and no leaf switch can connect to other 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): engineers 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 stopgap 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 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 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.

Overlay 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 us 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 return 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 engress 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 SDAs control, usually WAN routers, are called **SDA Border Node**.

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 rest on CRUD actions: Create, Read, Update and Delete. It is usually accessible via HTTP, whose verbs (actions) map neatly unto CRUD:

- Create – POST
- Read – GET
- Update – PATCH, PUT
- Delete – DELETE

APIs need a data model in order to transform the 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 human- and machine-readable. JSON consists primarily of objects (an unordered set of key-value pairs) and arrays (ordered list of zero or more values). NOTE: according to RFC 7158, JSON text could be a simple value instead, but this is rarely encountered. 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 as time passes. Small changes are hard to track and lead to configuration drift, i.e. divergences between individual devices and reference configuration. Centralised configuration files, version control and configuration management tools are used to solve these problems. They offer:

- Configuration provisioning – uploading configuration from a central repository, allowing generation of 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 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 device 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 on the device. 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 uses a composite metric that can take into account the following: delay, bandwidth, load and reliability. By default, it uses only delay and bandwidth, more specifically, a sum of segment delays and the lowest segment bandwidth.

EIGRP uses three tables for route calculation:

- Neighbour table – lists all directly connected routers discovered via Hello packets (sent to multicast address 224.0.0.10).
- Topology table – contains metrics of all routes to any destination within the AS. It is filled with information received from neighbours. When establishing adjacency, the router floods information on its routes and later sends only updates. 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, used as metric. Basically reported distance + distance to the router that advertises this route. Shortest computed distance is called **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**.

Topology table also contains **feasible successors**. These are the fallback routes for successors with the second-lowest metric. A feasible successor must have a reported distance lower than feasible distance of the successor – this is called **feasibility condition** and is enforced to ensure loop-free feasible successor. If the successor fails and there is no feasible successor, the router must query neighbours for a new successor.

By default, the 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

Display EIGRP topology:

- #show ip eigrp topology

EIGRP supports equal-cost load-balancing, just like OSPF. 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 cost. 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>

IOS image

There are several types of IOS images with different functionality sets:

1. K9 – include SSH and IPSec
2. WAN – include 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, do 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