**Jeremy’s IT Lab CCNA Course**

**---------------------------------------------------**

***Section 3 – Interfaces + Cables (1.3):***

**8 bits = 1 byte**. Speed is measured in bits per second (Kbps, Mbps, Gbps, etc.) not bytes per second.

**Routers transmit** data on pins 1/2 and **receive** data on pins 3/6, **Switches** **are the opposite** **=** receive on 1/2 and transmit on 3/6.

**Auto MDI-X –** Allows for devices to automatically detect which pins a neighboring device transmits/receives data on (**removes** the **need to worry about straight-through/cross-over cables**).

**Fiber-Optic Connections –** Uses a Small Form-Factor Pluggable(**SFP**)Transceiver to plug into a switch then allow a fiber cable to be connected.

**MultiMode Fiber –** Core diameter is wider than Single-Mode Fiber. Allows multiple angles of light waves to enter the fiberglass core. Allows longer cables than UTP, but shorter cables than single-mode fiber.

**Single-Mode Fiber –** Core diameter is narrower than multimode fiber. Light enters at a single angle from a laser-based transmitter. Allows longer cables than both UTP and MultiMode fiber. **More** **expensive** than MultiMode.

**UTP cables support distances** up to 100 meters and **MultiMode/Single-Mode Fiber cables both support** over 150-meter distances but Single-Mode can support the longest distances.

**----------------------------------------------------------------------------**

***Section 4 – OSI (Open Systems Interconnection) Model:***

|  |  |
| --- | --- |
| Functions | Layers |
| **A**pplication | **7** |
| **P**resentation | **6** |
| **S**ession | **5** |
| **T**ransport | **4** |
| **N**etwork | **3** |
| **D**ata Link | **2** |
| **P**hysical | **1** |

* **Application Layer** (**7**) **–** Closest to the end user. Interacts with software apps (**doesn’t include** the **apps themselves** **but** the **protocols that** **interact with them**). **Functions of Layer 7:** Identifying communication partners/Synchronizing communication
  + - * **Presentation Layer** (**6**) **–** Translates data between application format to network formats, also translates between different Application-Layer formats. **EX:** Encryption of data as its sent, and decryption of data that’s received.
      * **Session Layer** (**5**) **–** Controls dialogues (sessions) between communicating hosts. **Establishes, manages, and terminates connections** between local application (your web browser) and the remote application (YouTube, Twitter, etc.)

***Network engineers don’t usually work with******these top 3 layers****, App Developers work with these to connect apps over networks. Once data passes through the top 3 layers, the Transport Layer adds a Layer 4 Header on top of the data.*

* **Transport Layer** (**4**) **–** Segments and reassembles data for communication between end hosts, making the data easier to be sent over the network and less likely to cause transmission problems. **Provides host-to-host communication**.

*Data is prepared by the top 3 layers; a Layer 4 Header is then added on and this unit of data with a L4 Header is called a ‘****segment****’. If the data being sent is large enough, the Transport Layer will break it up into segments and each one will receive a L4 Header. Next, that segment passes on to L3 and a L3 Header is added on.*

* **Network Layer** (**3**) **–** Provides connectivity between end hosts on different networks and provides logical addressing (IPs) of source and destination. Routers operate at Layer 3.

*Segment of data that has both L4/L3 headers is then called a ‘****packet****’. The packet then passes to L2, and a L2 Header is added on.*

* **Data Link Layer** (**2**) **–** Provides node-to-node connectivity and data transfer (**PC to switch, switch to router, router to router**). Defines how data is formatted for transmission over a physical medium (copper UTP cables). Detects and (possibly) corrects Physical Layer errors. Uses L2 addressing (MAC), separate from L3 addressing. Switches operate at L2.

*A packet that has passed through Layer 2 is given a L2 Header + Trailer and is then called a ‘****frame****’. A frame isn’t further encapsulated at Layer 1 but is instead passed through that layer, whether through physical or wireless means.*

* **Physical Layer** (**1**) **–** Defines physical characteristics of how to transfer data between devices (**voltage levels, max transmission distances, cable specs, etc.**). Digital bits are converted into electrical (**wired** **connections**) or radio (**wireless** **connections**) signals.

*After this, data is considered a complete frame and is sent from the local device to a remote device. It then begins the same process in reverse which is called ‘****de-encapsulation****’. The Data Link layer of the remote device translates the raw physical data into a complete frame and the L2 Header/Trailer are removed, leaving the Layer 3 packet. The Network Layer then removes the L3 Header, leaving the packet as a segment and that is passed to the Transport Layer. This L4 Layer then removes the L4 Header and the original data sent by the local device is passed to the upper layers of the remote device, completing the de-encapsulation process.*

Data = Data **}**

Data>**L4** Header **=** **Segment** **=** Protocol Data Units (**PDUs**)

Data>**L4** Header>**L3**Header **=** **Packet**

**L2** Trailer>Data>**L4** Header>**L3**Header>**L2** Header **=** **Frame** **}**

**Layer 1** **PDU =** Bit

*CCNA uses a combination of OSI and TCP/IP. Network engineers refer to OSI’s layers when specifying a problem. EX:* ***A Layer 4 problem is a Transport Layer issue not Application Layer****.*

**TCP/IP Suite:**

|  |  |
| --- | --- |
| TCP/IP Suite |  |
| **A**pplication | **4** |
| **T**ransport | **3** |
| **I**nternet | **2** |
| **L**ink | **1** |

* **Application Layer** (**4**) **of TCP/IP =** App, Presentation, and Session Layers of OSI
* **Transport** (**3**) **=** Transport (**OSI 4**)
* **Internet** (**2**) **=** Network (**OSI 3**)
* **Link** (**1**) **=** Data Link (**OSI 2**) **/** Physical (**OSI 1**)

**-----------------------------**

***Section 5 - Cisco CLI:***

**Password Encryption:** **(config)#**service password encryption **=** Command can be used to encrypt all device passwords, ensuring they can’t be seen in plain text under the running configuration.

**(config)#**enable secret *example* **=** Command can be used to implement a more secure form of password encryption (**MD5**) than the ‘*service password encryption’* provides.

***enable secret*** always takes precedence over ***enable password*** when both are configured*.*

**Save Configuration:** **#**write**/** **#**write memory **/** **#**copy running-config startup-config

**--------------------------------------------------**

***Section 6 – Ethernet LAN Switching:***

|  |  |  |
| --- | --- | --- |
| Eth. Header | Packet | Eth. Trailer |

**= Ethernet Frame** *(26 bytes total)*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Preamble | SFD | Destination | Source | Type (or length) |

**= Ethernet Header**

* ***Preamble =*** *(7 bytes)*
* ***SFD*** *(Start Frame Delimiter)* ***=*** *(1 byte)*
* ***Destination =*** *(6 bytes)*
* ***Source =*** *(6 bytes)*
* ***Type*** *(****Indicates L3 Protocol*** *used in the packet)* ***=*** *(2 bytes)*

**Preamble =** Allows devices to synchronize their receiver clocks.

**SFD =** Marks the end of the Preamble and the beginning of the rest of the frame.

**Destination/Source =** Indicate the MACs of the sending/receiving devices.

**Eth. Trailer = FCS (Frame Check Sequence)** *(****4 bytes****)* **–** Used by the receiving device to check for any possible errors that occurred during transmission. **Uses CRC (Cyclic Redundancy Check)** to do this.

**Unicast Frame –** A frame destined for a single target.

**MAC Address –** **6-byte (48-bit)** physical address assigned to a device when it is made. The first 3 bytes are the OUI(**Organizationally Unique Identifier**), which is assigned to the company manufacturing the device. The last 3 bytes are unique to the device itself.

**Hexadecimal –** In hexadecimal, the number in the ten’s place (10, 20, 30) is equal to decimal 16.

**EX:** Decimal 16 = Hexadecimal 10, Decimal 17 = Hexadecimal 11, Decimal 18 = Hexadecimal 12, etc.

***Section 7 – Ethernet LAN Switching Pt.2:***

The **minimum size for an ethernet frame is** 64 bytes. 64 bytes – 18 bytes (header & trailer size) = 46 bytes. **If a packet is less than 46 bytes,** padding bytes are added.

**ARP (Address Resolution Protocol) –** Used to discover the Layer 2 address (**MAC**) of a known Layer 3 address (**IP**). **IPv4 Ethernet Header Value =** 0x0806. Consists of 2 messages: ARP Request & ARP Reply:

* **ARP Request =** Is broadcast, sent to all hosts on the network
* **ARP Reply =** Is unicast, sent only to one host (host that sent the request)

**Broadcast MAC Address =** FFFF.FFFF.FFFF

**MAC Learning & Aging –** MAC Address Tables can learn MAC’s either through Static (manually entered) or Dynamic (learned through ARP) types. These Address Tables will clear any MAC’s that are dormant (**or aged**) for more than 5 minutes. **Tables can** also **be cleared manually with** the **command:** **#**clear mac address-table dynamic. You can clear a specified MAC or the entire table.

**-------------------------------------------**

***Section 8/9 – IPv4 Addressing:***

**IPv4 Address –** Decimal: 192.168.10.2 = 32-bit binary number (**2 parts-**Network ID & Host ID)

**Subnet Mask –** Decimal: 255.255.255.0 = 32-bit binary number (Identifies the Network ID portion of the address)

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **First Octet** | **First Octet Numeric Range** | **Prefix Length** |
| A | 0xxxxxxx | 0 – 127 | /8 |
| B | 10xxxxxx | 128 – 191 | /16 |
| C | 110xxxxx | 192 – 223 | /24 |

**IPv4 Classes:**

* **Class A** **=** 1.0.0.1 – 127.255.255.254

Default Subnet = 255.0.0.0

**Number of Networks =** 128

**Number of Hosts per Network =** 16,777,214

Binary = 00000001

* **Class B** **=** 128.0.0.1 – 191.255.255.254

Default Subnet = 255.255.0.0

**Number of Networks =** 65,636

**Number of Hosts per Network =** 65,534

Binary = 10000000

* **Class C** **=** 192.0.0.1 – 223.255.255.254

Default Subnet = 255.255.255.0

**Number of Networks =** 16,777,216

**Number of Hosts per Network =** 254

Binary = 11000000

**Loopback Addresses –** **Address range =** 127.0.0.0-127.255.255.255 / Used to test the ‘network stack’ (**OSI, TCP/IP model**) on local devices.

**Formula for Max Hosts Per Network =** 2n -2

**Formula for Max Subnets Per Network =** 2n

**--------------------------------------------------**

***Section 10 – Switch Interfaces (1.4):***

**Configure Interface Ranges:** **(config)#**int range *f0/5 – 12* OR **(config)#**int range *f0/5 -6 , f0/9 -12*

**Half Duplex –** Device **cannot** send and receive data at the same time. If receiving a frame, it must wait before sending a frame. (**Devices attached to a hub must operate at half duplex**)

**Full Duplex –** Device **can** send and receive data at the same time. (Devices attached to a switch can operate at full duplex)

**Hubs –** All devices connected to a hub are part of the **same collision domain**, the frames they send could collide with frames sent from the other devices.

**CSMA/CD = (Carrier Sense Multiple Access w/ Collision Detection)** describes how devices avoid collisions in a half-duplex situation and how they react to occurred collisions. Devices ‘listen’ to the collision domain until they detect that other devices aren’t sending. If a collision occurs, device will send a jamming signal to inform other devices.

**Speed/Duplex Autonegotiation Disabled:**

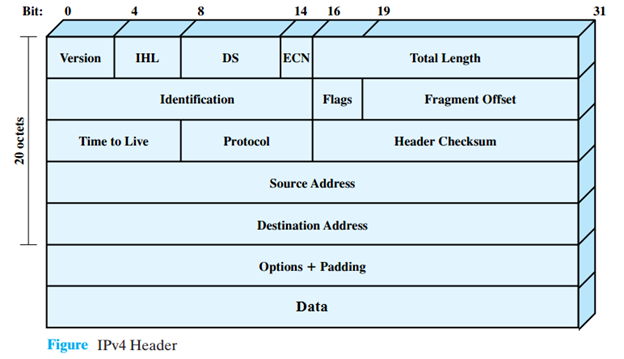
* **Speed =** Switch will try to sense the speed that the other device is operating at. If it fails, it will use the slowest supported speed.
* **Duplex =** If speed Is 10/100 Mbps, the switch will use half duplex. If the speed is 1000 Mbps or greater, it will use full duplex.

**Interface Errors:**

* **Runts =** Frames that are smaller than the minimum size (**64 bytes**)
* **Giants =** Frames that are larger than the max frame size (**1518 bytes**)
* **CRC =** Frames that failed the CRC check (in the Ethernet FCS trailer)
* **Frame =** Frames that have an incorrect format (due to an error)
* **Input Errors =** Total of various counters, such as the above four
* **Output Errors =** Frames the switch tried to send but failed due to an error

**------------------------------------------------**

***Section 11 – IPv4 Header* (32 bits)*:***

****

**Version** field(4 bits) **–** Identifies the version of IP used (IPv4, IPv6)

**(IHL) Internet Header Length** field (4 bits) **–** Identifies the length of the header in 4-byte increments (A value of 5 = 5 x 4-btyes = 20 bytes.

* **Minimum value** is 5 (= 20 bytes)
* **Max value** is 15 (15 x 4-btyes = 60 bytes)

**(DS) Differentiated Services /(ToS) Type of Service** (8 bits)**:**

* **(DSCP) Differentiated Services Code Point** field (6 bits) **–** Used for QoS, prioritizing delay-sensitive data (streaming voice, video, etc.)
* **(ECN) Explicit Congestion Notification** field (2 bits) **–** Provides end-to-end (between 2 endpoints) notification of network congestion without dropping packets. Requires both endpoints and network infrastructure to support it.

**Total Length** field (16 bits) **–** Indicates total length of packet (L3 header+ L4 segment). Minimum value = 20. Max value = 65,535.

**Identification** field (16 bits) **–** If a packet is fragmented due to being too large (> MTU), this field is used to identify which packet the fragments belong to.

**Flags** field (3 bits) **–** Used to control/identify fragments:

* **Bit 0 =** Reserved, always set to 0.
* **Bit 1 =** Don’t Fragment (DF bit), used to indicate a packet that shouldn’t be fragmented (**1=don’t fragment**).
* **Bit 2 =** More Fragments (MF bit), set to 1 if there are more fragments in the packet, set to 0 for the last fragment.

**Fragment Offset** field (13 bits) **–** Used to indicate the position of the fragment within the original, unfragmented packet. Allows fragmented packets to be reassembled even if the fragments arrive out of order.

**(TTL) Time to Live** field (8 bits) **–** Used to prevent infinite loops. In practice, indicates a “hop count”: each time a packet arrives at a router, the TTL decreases by 1. A **router will drop a packet with a TTL of 0**.

**Protocol field** (8 bits) **–** Indicates the protocol of the encapsulated Layer 4 PDU**:**

* Value of 6 **= TCP**
* 17 **= UDP**
* 1 **= ICMP**
* 89 **= OSPF**

**Header Checksum** field (16 bits) **–** Used to **check for errors only in the IPv4 header**. When a router receives a packet, it calculates the checksum of the header and compares it to the one in this field of the header. If they don’t match the router drops the packet.

**--------------------------------------------------------------------------------**

***Section 12 – Routing Fundamentals & Static Routing (3.3):***

**Local Routes and Connected Routes** are automatically added to a router’s routing table when you configure an IP on an interface.

**Local Route –** Provide a route to a router’s own IP address (/32 netmask).

**Connected Route –** Provide a route to the network an interface is connected to.

**Default Route –** A route to **0.0.0.0/0** = the least specific route possible, includes every possible destination IP. If a router doesn’t have a specific route that matches a packet’s destination IP, it will forward the packet using the default route (often used to direct traffic to the internet).

**Configure Static Route:**

* **(config)#**ip route *ip address netmask next-hop*
* **(config)#**ip route *ip address netmask exit-interface* (**Appears as directly connected in routing table**)
* **(config)#**ip route *ip address netmask exit-interface next-hop*

**-----------------------------------------**

***Section 13 – Life of a Packet:***

ARP Request = Broadcast & ARP Reply = Unicast

**Packets sent** **to devices** **across LAN’s** will first use the sending LAN’s default gateway as the initial destination MAC, then the receiving LAN’s router as destination MAC, and once the packet arrives to the target’s device LAN, it’s router will use that device as the destination MAC.

**Packets sent between devices within the same LAN** will immediately use the target device’s MAC as the destination MAC, even if a switch sits between them. The destination MAC will not be the switch’s MAC.

**------------------------------------------**

***Section 13 – Subnetting (1.6):***

**Point-to-Point Network –** A network (link) that directly connects two APs together, providing a connection between different LAN’s. **EX:** A connection between two office networks that are in different cities.

**/31 Subnetwork –** Only has 2 available IPs, network & broadcast addresses, but can be uses for a point-to-point network since only 2 IPs are required for each AP.

/31 = 11111111.11111111.11111111.11111110 = 2 host IPs

**2x =** Number of Subnets (x = Number of borrowed network bits)

**2n – 2 =** Number of Hosts (n = Number of host bits)

**----------------------------------------**

***Section 17-19 – VLANs (2.2):***

**VLANs are configured on switches** on a per-interface basis, and **logically separate end hosts at Layer 2**. A switch won’t forward traffic between VLANs, including broadcast/unknown unicast traffic.

Switches will ‘**tag**’ all frames they send over a trunk link. This tells the receiving switch which VLAN the frame belongs to.

**Trunk Ports =** ‘Tagged’ Ports & **Access Ports =** ‘Untagged’ Ports

**802.1Q –** An **industry standard** trunking protocol used to identify VLAN frames and to which VLAN they belong to.

**802.1Q Tag –** Inserted in the Ethernet header between the Source and Type/Length fields. The tag consists of 2 main fields:

* **TPID** (Tag Protocol Identifier) **= Always** set to a value of **0x8100** (indicates an 802.1Q-tagged frame).
* **TCI =** Tag Control Information (**consists of 3 stub fields**):

**PCP (Priority Code Point) =** Used for CoS (Class of Service), which prioritizes important traffic in congested networks.

**Drop Eligible Indicator (DEI) =** Indicates frames that can be dropped if network congestion occurs.

**VID (VLAN ID) =** Identifies the VLAN a frame belongs to.

VLAN Ranges – Range of VLANs (1-4094) is divided into 2 sections:

* **Normal VLANs =** 1-1005
* **Extended VLANs =** 1006-4094

**Native VLAN –** Is **1 by default** on all trunk ports but can be manually configured (**must match on all configured switches**). A switch does not add an 802.1Q tag to frames in the native VLAN. For **security reasons**, its best to **change the native VLAN to an unused VLAN**.

**Router on a Stick (ROAS) -** A router with a single interface (link) that connects to a switch (switch only must be configured as a trunk interface) and can handle multiple VLAN traffic on sub-interfaces. **EX:** G 0/0.10, G 0/0.20, G 0/0.30 (These are all one physical interface - G 0/0).

**Configuring ROAS:**

* **(config)#**int g0/0
* **(config-if)#**no shut
* **(config-if)#**int g0/0.10
* **(config-if)#**encapsulation dot1q 10
* **(config-if)#**ip address 192.1.1.10 255.255.255.192 (repeat process for remaining sub-interfaces but replace ‘g0/0.20’ and ‘encapsulation dot1q 20’ etc.)

**Layer 3 (Multilayer Switches) –** Switch that’s capable of both switching and routing. You can assign IP addresses to its interfaces, like a router. You can create virtual interfaces (SVIs) for each VLAN and assign IPs to each interface (**the actual VLAN must exist on the switch for the SVI to be up/up**). Can **configure routes on the switch and can be used for inter-VLAN routing**.

**----------------------------------**

***Section 20 – DTP & VTP:***

**Dynamic Trunking Protocol (DTP) –** Cisco proprietary protocol that allows switches to dynamically determine their interface status (*access* or *trunk*). A switchport in ***‘dynamic desirable’*** mode **will actively** try to **form a trunk** with other switches if they are in the following switchport modes:

* *switchport mode trunk*
* *switchport mode dynamic desirable*
* *switchport mode dynamic auto*

A switchport in ***‘dynamic auto’*** mode will **NOT** actively try to **form a trunk** with other switches but it will form a trunk if the switch connected to it is actively trying to form a trunk. It will form a trunk with a switchport in the following modes:

* *switchport mode trunk*
* *switchport mode dynamic desirable*

You can **disable DTP negotiation** on an interface **with the command:** *switchport nonegotiate*.

**VLAN Trunking Protocol (VTP) –** Allows you to configure VLANs on a central VTP server switch, and other switches (**VTP clients**) will synchronize their VLAN databases to the server. Designed for large networks with many VLANs. Rarely used, **recommended not to be used for security reasons**. There are **3 versions** (1, 2, and 3) and **3 modes:** Server, Client, and Transparent.

If a switch with no VTP domain receives a VTP advertisement that contains a VTP domain name, it will automatically join that domain.

**--------------------------------**

***Section 22 – STP (2.5):***

**Layer 2 Loops:**

* **Broadcast Storm –** The Ethernet header doesn’t have a **TTL field (Layer 3 loop prevention)**. **Switch broadcast frames (Layer 2)** will loop around the network indefinitely. If enough of these looped broadcasts accumulate, a network will be too congested for legitimate traffic to use the network.
* **MAC Address Flapping –** When frames with the same source MAC repeatedly arrive on different interfaces, a switch is continuously updating the interface in its MAC address table.

**Spanning Tree Protocol (STP aka ’IEEE 802.1D’) –** **Prevents Layer 2 loops** by placing redundant ports in a blocking state, disabling the interface.These interfaces act as backups that can enter a forwarding state if an active interface fails.

STP creates a single path to/from each point in the network which prevents Layer 2 loops. There’s a set process that STP uses to determine which ports should be forwarding and which should be blocking:

Switches send/receive **Hello BPDUs** out of all interfaces **every 2 seconds** by default. If a switch receives a Hello BPDU on an interface, it knows that interface is connected to another switch.

**Root Bridge –** Switches use one field in the STP BPDU, the **Bridge ID field**, to elect a root bridge for the network. The switch with the **lowest Bridge ID(\*)** becomes the root bridge.

**(\*) Bridge ID = Bridge Priority field + MAC Address**

**Bridge Priority field = Bridge Priority + Extended System ID (VLAN ID)**

**Default Bridge Priority –** **32,768** is the default priority on all switches, by default the MAC is used as the tiebreaker (**lowest MAC = Root Bridge**). The default priority for switches in different VLANs is, **32,768 + the VLAN ID of the VLAN the switch resides in**.

**EX:** VLAN 1 + 32,768 = Priority of 32,769.

**Once STP has converged** in a network topology and all switches agree on the root bridge, **only the root** **bridge sends BPDUs**.

**STP BPDU Protocol Version Identifier = 0**

**Root Port –** After the Root Bridge selection, each remaining switch will select one of its interfaces to be its root port. The **interface with the lowest root cost** will be the root port. **Ports across from** **the root port** will **always** be **Designated** ports.

**Root Port Selection:**

1. Lowest Root Cost
2. Lowest Neighbor Bridge ID
3. Lowest Neighbor Port ID

**STP Collision Domains =** **Each remaining collision domain** will **select one interface** to be **Designated** and the other port will be Non-Designated (Blocking).

**Designated Port Selection:**

1. Interface with lowest root cost
2. Interface with lowest Bridge ID

**STP Port States –** Blocking/Forwarding (Stable) & Listening/Learning (Transitional)

* **Blocking State =** Does not send/receive regular network traffic. Does receive BPDUs but does not forward them. Does not learn MACs.
* **Listening State =** Only Designated/Root ports enter this state. **15 seconds by default**, this is determined by the Forward Delay timer. Interfaces in the Listening State only forwards/receives STP BPDUs, they do not send/receive regular traffic and do **not learn MACs**.
* **Learning State =** After the Listening state, a Designated/Root port will enter this state. Identical to the Listening State except that it only sends/receives STP BPDUs and **does learn MACs**.

**STP Timers:**

|  |  |  |
| --- | --- | --- |
| **STP Timer** | **Purpose** | **Duration** |
| Hello | How often the Root Bridge sends BPDUs. | 2 secs |
| Forward Delay | How long a switch will stay in Listening/Learning states (each is 15 secs = 30 secs). | 15 secs |
| Max Age | How long an interface will wait to change STP topology after not receiving Hello BPDUs. Timer is reset after every received BPDU. | 20 secs (10 hello) |

**STP Destination MAC =** 0180.c200.0000

**PVST+ Destination MAC =** 0100.0ccc.cccd

**BPDU Guard –** An interface with this enabled will shut down if it receives a BPDU from another switch, to prevent a L2 loop from forming.

**Root Guard –** Interface will not accept a new switch as the Root Bridge even if it receives a superior BPDU (lower Bridge ID). Interface will be disabled.

**Configuring a Primary Root Bridge:** **(config)#**spanning-tree vlan *vlan-#* root primary

(This command sets the STP priority to **24,576**. If another switch already has a lower priority, it sets this switch’s priority to 4096 less than the other switch’s priority.)

**Configuring a Secondary Root Bridge:** **(config)#**spanning-tree vlan *vlan-#* root secondary

(This command sets the STP priority to **28,672**.)

**-----------------------------------------------------------**

***Section 23 - Rapid STP (Rapid PVST+) (2.5):***

**Rapid STP (802.1w) –** Faster at converging network changes than STP. All VLANs share one instance therefore, cannot load balance.

**Rapid PVST+ - Cisco’s upgrade to 802.1w**, Each VLAN has its own STP instance. Can load balance by blocking different ports in each VLAN.

**Multiple Spanning Tree Protocol (802.1s) –** Can group multiple VLANs into different STP instances to perform load balancing.

**RSTP Port States –** Discarding, Learning, Forwarding

**RSTP Port Roles –** Root, Designated, Alternate, Backup

In RSTP, the Non-Designated Port role is split into 2 roles – Alternate port & Backup port:

* **Alternate Port Role –** A discarding port that receives a superior BPDU from **another** switch, the same as a Blocking port role in STP. Functions as a **backup to the root port**.
* **Backup Port Role –** Discarding port that receives a superior BPDU from another interface on the **same** switch, this **only happens when 2 interfaces** are **connected to the same collision domain** (a hub). Functions as a **backup** port for a **designated port**. The interface with the lowest Port ID will be the Designated port and the other will be the Backup.

**UplinkFast & BackboneFast –** Both built into RSTP and operate by default, help blocking/discarding ports rapidly move to forwarding.

**Rapid STP is compatible with STP**. The interface(s) on the **Rapid STP-enabled switch** connected to the Classic STP-enables switch **will operate in Classic STP mode**.

**STP/RSTP BPDU Protocol Version Identifier –** **STP** has a protocol version of **0**, while **Rapid STP** has a protocol version of **2**.

In **STP, only the root bridge** originated BPDUs but in **Rapid STP, all switches** originate and **send their own BPDUs** from their designated ports.

**RSTP Link Types:**

* **Edge (RSTP’s PortFast) =** Port that’s connected to an end host, moves directly to forwarding without negotiation. Functions like a classic STP port with PortFast enabled.
* **Point-to-point =** Direct connection between 2 switches, functions in full-duplex.
* **Shared =** Connection to a hub, must operate in half-duplex mode.

**STP vs RSTP Costs:**

|  |  |  |
| --- | --- | --- |
| **Link Speed** | **STP Cost** | **RSTP Cost** |
| 10 Mbps | 100 | 2,000,000 |
| 100 Mbps | 19 | 200,000 |
| 1 Gbps | 4 | 20,000 |
| 2 Gbps | 3 | 10,000 |
| 10 Gbps | 2 | 2,000 |

**---------------------------------------------**

***Section 24 – EtherChannel (2.4):***

EtherChannel groups multiple interfaces together to act as a single interface, **STP will treat this group as a single interface**. Traffic using the EtherChannel will be load balanced among the physical interfaces in the group.

**Other names for EtherChannel =** Port Channel, **LAG** (Link Aggregation Group)

**Load-Balancing –** EtherChannel load balances based on ‘flows’, a **flow** is a communication between 2 nodes in a network. Frames in the same flow will be forwarded using the same physical interface.

**Configuring Load Balancing: (config)#**port-channel load-balance *method*

**Check Load Balance Being Used:** **#**show etherchannel load-balance

EtherChannel Configuration – **3 Methods:** **(config-if)#**channel-group *number* mode *mode*

* **PAgP (Port Aggregation Protocol) =** Cisco proprietary protocol, Dynamically negotiates the creation/maintenance of EtherChannel (**like DTP does for trunks**). Uses **Auto** and **Desirable** modes.
* **LACP (Link Aggregation Control Protocol) =** Industry standard protocol (**802.3ad**), Dynamically negotiates the creation/maintenance of EtherChannel (like DTP does for trunks). Uses **Active** and **Passive** modes.
* **Static EtherChannel =** A protocol isn’t used to determine if an EtherChannel should be formed, interfaces are instead statically configured to form an EtherChannel.

Up to 8 interfaces can be formed into a single EtherChannel (**LACP allows up to 16** but only 8 will be active, the other 8 will be in standby mode waiting for an active interface to fail).

Interfaces that are **members of an EtherChannel must have matching configurations:** Same duplex, speed, switchport mode, and same allowed VLANs/native VLAN (for trunk interfaces).

**---------------------------------------------------------**

***Section 25 – Dynamic Routing (3.0 - 3.4):***

**Network Route –** A route to a network or subnet (Mask length less than /32)

**Host Route –** A route to a specific host (/32)

**Dynamic Routing –** Routers can use dynamic routing protocols to advertise info about the routes they know to other routers. They form ‘adjacencies’/’neighbor relationships’ with adjacent routers to exchange this info.

**Dynamic Routing Protocol Types:**

* **IGP (Interior Gateway Protocol) =** Used to share routes within a single autonomous system, or a single organization/LAN. Uses either **Distance Vector** or **Link State** algorithms.
* **EGP (Exterior Gateway Protocol) =** Used to share routes between AS’s. Uses the **ESS** algorithm. **BGP (Border Gateway Protocol)** is the only EGP modernly used.

**Algorithm Type =** The processes each protocol uses to share routing info and choose the best route to destinations.

IGP Algorithm Types:

* **Distance Vector =** RIP (Routing Info Protocol) & EIGRP (Enhanced Interior Gateway Routing Protocol)

Distance Vector protocols operate by sending their known destination networks and their metric to reach those networks directly to their connected neighbors. This method of sharing route info is called **‘routing by rumor’** because routers don’t know about the network beyond their neighbor, they only know the info that neighbors tell them. Called ‘distance vector’ because routers only learn the **‘distance’ (metric)** and **‘vector’ (direction, the next-hop router)** of each route.

* **Link State =** OSPF (Open Shortest Path First) & IS-IS (Intermediate System to Intermediate System)

When using a Link State routing protocol every router creates a **‘connectivity map’** of the network. Each **router advertises** info about its interfaces (**connected networks**) **to** its **neighbors**. These ads are passed along to other routers until each one in the network develops the same map of the network. Each router then uses this map to calculate the best routes to each destination. This protocol uses more resources (CPU) on the router because more info is shared but tend to be faster in reacting to network changes than distance vector protocols.

**Metric –** Dynamic routing protocols use a route’s metric (**cost**) to determine which routes will be stored in its routing table (**lower Metric is chosen**).

**ECMP (Equal Cost Multi-Path) –** When 2 or more routes via the same routing protocol to the same destination, with the same metric, are both added to a routing table. Traffic is then load-balanced over both/all routes.

|  |  |  |
| --- | --- | --- |
| **IGP** | **Metric** | **Explanation** |
| **RIP** | Hop Count | Each router in the path counts as one ‘hop’. Total metric is the total hops to a destination. **Links of all speeds are equal**. |
| **EIGRP** | Bandwidth & Delay | By default, the bandwidth of the **slowest link in the route** and the total delay of all links in the route are used. |
| **OSPF** | Cost | Cost of each link is calculated based on bandwidth. The total metric is the total cost of each link in the route. |
| **IS-IS** | Cost | The total metric is the total cost of each link in the route. **All links have a cost of 10 by default**. |

**Administrative Distance (AD) –** Used to **determine which routing protocol** is preferred. A **lower AD** is preferred and indicates that the routing protocol is considered more ‘trustworthy’.

*In a* ***routing table****, the numbers in square brackets next to an IP are the* ***AD followed by*** *the* ***metric****.* EX: **[110/3]**

|  |  |
| --- | --- |
| **Route Protocol/Type** | **AD** |
| Directly Connected | 0 |
| Static | 1 |
| External BGP (eBGP) | 20 |
| EIGRP | 90 |
| IGRP | 100 |
| OSPF | 110 |
| IS-IS | 115 |
| RIP | 120 |
| EIGRP (External) | 170 |
| Unusable Route | 255 |

**Floating Static Route –** Changing the AD of a static route makes it less preferred than routes learned by a dynamic routing protocol to the same destination (**AD must be higher than the routing protocol’s**). The route will be inactive unless the route learned by the dynamic routing protocol is removed/fails.

**------------------------------------**

***Section 26 – RIP & EIGRP:***

**RIP (Routing Information Protocol) –** Distance vector IGP, uses routing by rumor logic to learn/share routes. Uses hop count as its metric, **max hop count is 15** anything more is unreachable. Uses **Request & Response message types** to share routing tables. RIP-enabled routers will share their routing table every 30 seconds.

**RIPv1 –** Only advertises classful addresses (**Class A, B, C**) and doesn’t support VLSM or CIDR. Doesn’t include subnet mask info in ads (Response messages). **Messages** are **broadcast to 255.255.255.255**.

**RIPv2 –** Supports VLSM, CIDR and includes subnets mask info in ads. **Messages** are **multicast** to **224.0.0.9**.

RIP Configuration:

* **(config)#**router rip
* **(config-router)#**version 2
* **(config-router)#**no auto-summary
* **(config-router)#**network 10.0.0.0
* **(config-router)#**network 172.16.0.0

The ***‘network’* Command –** **Tells the router to:** look for interfaces that fall in the range, activate RIP on those interfaces, form adjacencies with connected RIP neighbors, advertise the network prefix on the interface (**NOT the prefix in the *network* command**). OSPF and EIGRP *network* commands operate in the same way.

**Passive-Interface Command:** **(config-router)#**passive-interface g2/0 = Tells the router to stop sending RIP ads out of the specified interface. You should **always use this command on interfaces that don’t have any RIP neighbors**. EIGRP and OSPF have the same passive interface functionality.

**Default-Info Originate Command:** **(config-router)#**default-information originate = Advertises a default route into RIP neighbors.

**EIGRP (Enhanced Interior Gateway Routing Protocol) –** Was and mainly still is Cisco proprietary. An advanced/hybrid distance vector routing protocol. Much faster than RIP in reacting to changes in the network. Doesn’t have the 15 hop-count limit of RIP. Sends messages using **multicast** address **224.0.0.10**. The **only IGP that can perform unequal-cost load-balancing** (**by default, performs ECMP load-balancing over 4 paths**).

EIGRP Configuration:

* **(config)#**router eigrp 1 (1=**AS number, must match between routers** or they won’t form an adjacency/share route info)
* **(config-router)#**no auto-summary
* **(config-router)#**passive-interface ‘int’
* **(config-router)#**network 10.0.0.0 (*network* command will assume classful address w/out a WC)
* **(config-router)#**network 172.16.1.0 0.0.0.15

**EIGRP uses a wildcard mask** instead of a subnet mask.

**Wildcard Masks –** Inverted subnet masks, all 1’s in the subnet mask are 0 in the wildcard mask and all 0’s in the subnet are 1’s in the wildcard. A shortcut is to subtract each octet of the subnet mask from 255.

**EX:** 255.255.248.0: 255-255 = 0. 255-255 = 0. 255-248 = 7. 255-0 = .255: 0.0.7.255

**Router ID –** **Order of priority:**

1. Manual configuration
2. Highest IP on a loopback interface
3. Highest IP on a physical interface

In a routing table, **EIGRP routes** are **indicated with** the letter ‘**D**’ not ‘E’.

**EIGRP Metric –** Metric = Bandwidth + Delay (**Bandwidth of the slowest link + the Delay of all links**)

**(config-router)#**do show ip eigrp topology

**EX:** 192.168.4.0/24, 1 successors, FD is 28672

via 10.0.12.2 (28672/28416), G0/0

via 10.0.13.2 (30976/28416), F1/0

*^ (The numbers in parentheses above are the Feasible Distance and Reported Distance of each route.)*

**EIGRP Terminology:**

**Feasible Distance =** Router’s metric value to the route’s destination (**includes** the **metric to a neighbor router +** that **neighbor’s Advertised Distance** **to** the **destination**)

**Reported Distance (Advertised Distance) =** Neighbor’s metric value to the route’s destination

**Successor =** Route with the lowest metric to the destination (**best route**)

**Feasible Successor =** An alternate/backup route to the destination (not the best route) which meets the feasibility condition

**Feasible Condition =** A route is considered a feasible successor if its Reported Distance is lower than the successor route’s Feasible Distance.

The **Feasible Successor is a loop prevention system**, if a route meets the feasibility requirement (condition) its guaranteed not to be a looped route.

**EIGRP Unequal-Cost Load-Balancing:** **#**sh ip protocols **=** Will output “EIGRP max metric variance”

**Variance 1 =** Only ECMP (equal cost) load-balancing will be performed.

**(config-router)#**variance 2

**Variance 2 =** Feasible Successor routes with an **FD up to 2x** the **successor route’s FD** can be used to load-balance (**unequal cost**).

**--------------------------------------**

***Section 27-29 – OSPF (3.4):***

Uses the Shortest Path First algorithm (**aka Dijkstra’s algorithm**). **3 versions:**

* OSPFv1 (Not in use anymore)
* OSPFv2 (Used for IPv4)
* OSPFv3 (Used for IPv6, can be used for IPv4 but usually v2 is used)

Routers store info about the network in **LSAs** (**Link State Advertisements**), which are organized in a structure called the **LSDB** (**Link State Database**). Routers will flood LSAs until all routers in the *OSPF area* develop the same map of the network (LSDB).

**In** **OSPF**, there’s **3 main steps in** the **process of sharing LSAs and determining the best route** to each destination in a network**:**

1. Become neighbors with other routers connected to the same segment.
2. Exchange LSAs with neighbor routers.
3. Calculate the best routes to each destination and insert them into the routing table.

**OSPF Area –** A set of routers and links that share the same LSDB. All OSPF areas should be contiguous (**not separated**). All areas must have at least one ABR connected to the backbone area. OSPF interfaces in the same subnet must be in the same area.

* **Backbone Area (Area 0)** = An area that all other areas must connect to. Routers connected to the backbone area are called **Backbone Routers**.
* **Internal Routers =** Routers with all interfaces in the same area.
* **Area Border Routers (ABRs)** = Routers with all interfaces in multiple areas. ABRs maintain a separate LSDB for each area they are connected to. It’s recommended that you connect an **ABR** to a **max of 2 areas** (3+ areas can overburden the router).
* **Intra-Area Route** **=** Route to a destination inside the **same** OSPF area.
* **Inter-Area Route** **=** Route to a destination in a **different** OSPF area.

**OSPF Process ID – Locally significant**, routers with different process IDs can become OSPF neighbors.

**Router ID – Order of priority:**

1. Manual configuration
2. Highest IP on a loopback interface
3. Highest IP on a physical interface

Configure Router ID: **(config-router)#**router-id *A.B.C.D*

**Autonomous System Boundary Router (ASBR) –** An OSPF router that connects the OSPF network to an external network. By using the *‘****default-information originate****’* command, a router becomes an ASBR.

**OSPF Cost –** OSPF’s metric is called cost and is automatically calculated based on the bandwidth of an interface. It’s calculated by dividing a reference bandwidth value by the interface’s bandwidth. The **default reference bandwidth is 100 mbps**.

* Reference: 100 mbps / 10 mbps Interface = cost of 10
* Reference: 100 mbps / 100 mbps Interface = cost of 1
* Reference: 100 mbps / 1000 mbps Interface = cost of 1 (\*)
* Reference: 100 mbps / 10000 mbps Interface = cost of 1 (\*)

\* (**All values less than 1 will be converted to 1**) Therefore, FastEth,GigEth, 10GigEth, etc. are equal and all have a cost of 1 by default.

**Configuring OSPF Reference Bandwidth:** **(config-router)#**auto-cost reference bandwidth *mbps*

You **should configure** a **reference bandwidth greater than** the **fastest links in** your **network**. You should also configure the same reference bandwidth **on all OSPF routers** in a network.

The **OSPF** **cost** to a destination **is** the **total cost of the outgoing/exit interfaces**. Loopback interfaces have a cost of 1 (plus the cost of the outgoing interfaces to reach the specified loopback).

**Configuring OSPF Cost:** **(config-if)#**ip ospf cost *cost-amount*

**OSPF Neighbors –** When OSPF is activated on an interface, the router starts sending OSPF *hello* messages out of the interface at regular intervals. These are used to introduce the router to potential OSPF neighbors. The **default *hello* timer is 10 seconds** on an Ethernet connection. *Hello* messages are multicast to **224.0.0.5** (**multicast address for all OSPF neighbors**). OSPF messages are encapsulated in an **IP header** with a value of **89 *(0x59)*** in the **Protocol field**.

**OSPF Neighbor States:**

* **Down State –** OSPF is activated on an interface, it sends an OSPF *Hello* message to 224.0.0.5 but it doesn’t know about any neighbors yet (Its **Neighbor RID field** has a value of 0.0.0.0).
* **Init State –** A OSPF *Hello* packet is received by a router, but its own router ID isn’t in the *Hello* packet yet (**Neighbor RID field still 0.0.0.0**).
* **2-Way State –** Means a router has received, sent, and received a OSPF *Hello* packet (**Both own Router ID &** **Neighbor RID field have valid values**). **DR/BDR election** takes place/completed at this state.

*(Once participating routers reach this state, all* ***conditions*** *have been* ***met******for*** *them to become* ***OSPF neighbors****.)*

*(****ExStart to Full States*** *is when LSA Exchanges begin/take place.)*

* **Exstart State –** The 2 routers will now prepare to exchange info about their LSDB but first must choose which one will start the exchange. The router with the **higher RID** will become the **Master** and start the exchange while the **lower RID** will become the **Slave**. To decide the Master/Slave, they exchange DBD (**Database Description**) packets.
* **Exchange State –** Routers exchange **DBDs** which contain a list of their LSA’s (**only the** **basic info**) in their LSDB. The routers compare the info in the DBD they received to the info in their own LSDB to determine which LSAs they must receive from their neighbor.
* **Loading State –** Routers send Link State Request (**LSR**) messages to request that their neighbors send them any LSAs they don’t have. The requested LSAs are sent in Link State Update (**LSU**) messages. The routers then send **LSAck** messages to acknowledge that they received the LSAs.
* **Full State –** Routers have a full OSPF adjacency and identical LSDBs. They continue to send/listen for OSPF *Hello* (**every 10 secs**)to maintain neighbor adjacency. Every time a *Hello* packet is received, the **Dead Timer** (**40 secs**) is reset. If the **Dead Timer reaches 0** and no *Hello* is received, the **neighbor** **is removed**.

**Summary of OSPF Message Types:**

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Purpose** |
| 1 | **Hello** | Neighbor discovery and maintenance. |
| 2 | **DBD** (Database Description) | Summary of the LSDB of the router, used to check if LSDBs of each router match. |
| 3 | **LSR** (Link-State Request) | Requests specific LSAs from neighbor. |
| 4 | **LSU** (Link-State Update) | Sends specific LSAs to neighbor. |
| 5 | **LSAck** (Link-State Acknowledgement) | Used to acknowledge that a router received requested LSAs. |

**OSPF Configuration:**

* Directly activate OSPF on an interface**:** **(config-if)#**ip ospf *1* area *0*
* Make all interfaces OSPF passive: **(config-router)#**passive-interface default

Then make specific interfaces active: **(config-router)#**no passive-interface *int-id*

**OSPF Network Types –** Refers to the type of connection between OSPF neighbors. **3 main OSPF network types:**

* **Broadcast –** Enabled by **default on Ethernet** and FDDI (**Fiber Distributed Data Interfaces**) interfaces.
* **Point-to-Point –** Enabled by **default** on PPP (**Point-to-Point Protocol**) and HDLC(**High-Level Data Link Control**) interfaces.
* **Non-Broadcast –** Enabled by default on Frame Relay and X.25 interfaces. Does not allow multicast packets. The ‘***neighbor***’ **command** is needed to establish neighbor adjacencies.

**OSPF Broadcast Network Type –** A DR **(Designated Router)** and BDR **(Backup Designated Router)** must be enabled on each subnet/interface. (**only 1 DR if** there are **no OSPF neighbors**). Routers which aren’t the DR or BDR become a **DROther**.

In this network type, routers will only form a full OSPF adjacency with the DR and BDR of the segment (**Routers** will **only exchange LSAs with** the **DR/BDR**).

**DR/BDR Election Order of Priority:**

1. Highest OSPF interface priority
2. Highest OSPF Router ID

‘First place’ becomes the DR for the subnet and ‘second place’ becomes the BDR. The **default OSPF interface priority** is 1 on all interfaces.

**OSPF Priority 0 =** IF you set the OSPF interface priority to 0, the router **CANNOT** be the DR/BDR for the subnet/interface.

The **DR/BDR election** is ‘**non-preemptive**’. Once they’re selected, they’’ll keep their role until OSPF is reset, the interface fails/is shut down, etc.

**When** a **DR goes down** the BDR becomes the new DR, then an election is held for the next BDR. **DROthers** will **only** move to the **FULL state** with the **DR and BDR**. The neighbor state between DROthers will be **2-Way** (Won’t exchange LSAs).

**Full Adjacent OSPF Neighbor =** DR/BDR will form a FULL adjacency with **all** routers in a subnet. **DROthers** will **only form** a **FULL** **adjacency** with the DR/BDR, **not each other**.

Messages to the **DR/BDR** are **multicast** using address **224.0.0.6**.

**OSPF Point-to-Point Network Type –** Routers dynamically discover neighbors by multicasting OSPF *Hello* messages using 224.0.0.5. A **DR/BDR are not elected**, these encapsulations are used for P2P connections therefore there’s no reason to elect a DR/BDR.

**Serial Interfaces –** **Default encapsulation** is **HDLC**. To configure PPP encapsulation instead use the command: **(config-if)#**encapsulation ppp

One side of a serial connection is DCE(**Data Communications Equipment**) and the other is DTE (**Data Terminal Equipment**). To check use: **#**show controllers *interface-id*

You **must configure** the **clock rate** **on** the **DCE** side: **(config-if)#**clock rate *bits-per-second*

Ethernet interfaces use the ‘***speed’* command** to configure the interface’s operating speed. Serial interfaces use the ‘***clock rate’* command**.

**Serial Interface Encapsulation:** **(config-if)#**encapsulation *type*

If you change the **encapsulation**, it **must match** on both ends or interfaces will go down.

**Configure OSPF Network Type –** Change the network type on an interface: **(config-if)#**ip ospf network *type*. Not all network types work on all link types (a serial link can’t use the broadcast network type).

|  |  |
| --- | --- |
| **Broadcast** | **Point-to-point** |
| Default on Ethernet/FDDI interfaces | Default on HDLC/PPP (serial) interfaces |
| DR/BDR elected | **No** DR/BDR |
| Neighbors dynamically discovered | Neighbors dynamically discovered |
| Default **Timers:** Hello=10, Dead=40 | Default **Timers:** Hello=10, Dead=40 |

(**Non-Broadcast** network type default **timers:** Hello=30, Dead=120)

**OSPF Neighbor Requirements:**

1. Area # must match
2. Interfaces must be in the same subnet
3. OSPF process must not be shutdown
4. OSPF Router IDs must be unique (not match)
5. Hello and Dead timers must match:

**(config-if)#**ip ospf hello-interval *seconds* & **(config-if)#**ip ospf dead-interval *seconds*

1. Authentication settings must match:

**(config-if)#**ip ospf authentication-key *key* &**(config-if)#**ip ospf authentication

1. IP MTU settings must match:

**(config-if)#**ip mtu *bytes*

1. OSPF Network Type must match

*(Can still become neighbors if 7/8 don’t match but won’t reach FULL state)*

**OSPF LSA Types:**

* **Type 1: Router LSA =** Identifies Router (By using Router ID). Also lists networks attached to the router’s OSPF-activated interfaces.
* **Type 2: Network LSA =** Generated **only** by the **DR** of each ‘multi-access’ (broadcast) network and lists the routers which are attached to the multi-access network.
* **Type 5: AS External LSA =** Generated by ASBRs to describe routes to destinations outside of the AS (OSPF domain).

**----------------------------------------------------------------------**

***Section 30 – First Hop Redundancy Protocols (3.5):***

**FHRP** – Protocol that protects the default gateway used on a subnetwork by allowing 2 or more routers to provide backup **for that address**. If the active router fails, the backup router will take over the address.

**Virtual IP =** Configured on 2 (or more) routers and a **virtual MAC** is generated for the virtual IP (**each FHRP uses a different format for the virtual MAC**). An active router and a standby router are elected and end hosts in the network are configured to use the virtual IP as their default gateway. The active router replies to ARP requests using the virtual MAC so traffic destined for other networks will be sent to it.

If the **active router fails**, the standby becomes the active router. The new active router will send gratuitous ARP messages so that switches will update their MAC tables, it now functions as the default gateway.

If an **old active router comes back online**, by default it won’t take back its role as the active router, it will become the standby. You can configure **‘preemption’** so that the old active router does take back its role.

**HSRP (Hot Standby Router Protocol) –** A **Cisco proprietary** protocol where an **active** and **standby** router are elected. **Two versions:** Version 1 & Version 2 (V2 = Adds IPv6 and increases the number of groups that can be configured).

* **Multicast IPv4:** v1 = **224.0.0.2**

v2 = **224.0.0.102**

* **Virtual MAC:** v1 = **0000.0c07.ac*XX*** (***XX* =** HSRP group number)

v2 = **0000.0c9f.f*XXX*** (***XXX* =** HSRP group number)

In a scenario with **multiple subnets/VLANs**, you can configure a different active router in each one to load balance.

**VRRP (Virtual Router Redundancy Protocol) –** **Open standard** protocol where a **master** and **backup** routers are elected.

* **Multicast IPv4 =** **224.0.0.18**
* **Virtual MAC =** **0000.5e00.01*XX*** (***XX* =** VRRP group number).

In a scenario with multiple subnets/VLANs, you can configure a different active router in each one to load balance.

**GLBP (Gateway Load Balancing Protocol) –** **Cisco proprietary**, load balances among multiple routers within a single subnet.

An AVG (**Active Virtual Gateway**) is elected, then up to 4 AVFs (**Active Virtual Forwarders**) are assigned by the AVG (the **AVG** itself **can also be an AVF**). Each AVF acts as the default gateway for a portion of the hosts in the subnet.

* **Multicast IPv4 = 224.0.0.102**
* **Virtual MAC = 0007.b400.XXYY** (**XX =** GLBP group number, **YY =** AVF number)

**FHRPs Summary:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FHRP** | **Terminology** | **Multicast IP** | **Virtual MAC** | **Cisco Proprietary?** |
| HSRP | Active/Standby | v1: 224.0.0.2  v2: 224.0.0.102 | v1: 0000.0c07.acXX  v2: 0000.0c9f.fXXX | Yes |
| VRRP | Master/Backup | 224.0.0.18 | 0000.5e00.01XX | No |
| GLBP | AVG/AVF | 224.0.0.102 | 0007.b400.XXYY | Yes |

**Configuring HSRP:**

* **(config-if)#**standby *group-number* ip *virtual-ip*
* **(config-if)#**standby 1 priority *number*
* **(config-if)#**standby 1 preempt

The **group number must match between configured routers.**

The **‘priority’ command** is used to **determine** the **active router:**

1. Highest priority (**default 100**)
2. Highest IP

The **‘preempt’ command** causes a router to become the active router even if another router already has the role (**only necessary on** the **desired active router**).

Configuring Standby HSRP:

* **(config-if)#**standby version *1 OR 2*
* **(config-if)#**standby 1 ip *virtual-ip*

**HSRP version 1/2 aren’t compatible**, both routers must be configured under the same version.

**----------------------------------------**

***Section 31 – TCP/UDP (1.5):***

**Transport Layer (Layer 4) –** Provides transparent transfer of data between end hosts & various services to apps:

* Reliable Data Transfer
* Error Recovery
* Data Sequencing
* Flow Control

Provides Layer 4 addressing (**port numbers**) = Identifies the App Layer Protocol & Provides session Multiplexing.

**IANA Port Numbers:**

|  |  |
| --- | --- |
| **Well-Known:** | 0 – 1023 |
| **Registered:** | 1024 – 49151 |
| **Ephemeral/Private/Dynamic:** | 49152 – 65535 |

**TCP Flags:**

* TCP **Establishing Connection –** 3-Way Handshake = SYN flag>>SYN flag, ACK flag>>ACK Flag
* TCP **Terminating Connection –** 4-Way Handshake = FIN Flag>>ACK Flag/FIN Flag>>ACK Flag

**TCP Sequencing/Acknowledgement:**

**Hosts set** a **random initial sequence number** (EX: 10). **Forward Acknowledgement** is then used to indicate the sequence number of the next segment that the receiving host expects to receive (EX: 11). This is repeated with each sent segment:

**SYN** [Seq: **10**] >> **SYN-ACK** [Seq: **50**, ACK: **11]** >> **ACK** [Seq: **11**, ACK: **51**] This **continues adding one to each ‘Seq’/’Ack’** that’s sent between hosts which allows them to put segments in the correct order even if they arrive out of order. **If a** **segment isn’t acknowledged** it’s sent again (**TCP Retransmission**).

**TCP Flow Control:**

Destination host can tell the source host to increase/decrease the rate that data is sent **(Window Size) =** TCP header field that allows more data to be sent before an ack is required. **‘Sliding Window’** can be used to dynamically adjust how large the window size is.

**UDP** **(User Datagram Protocol) –** Connectionless and does not provide reliable communication, Sequencing, or Flow Control. **Used for** VoIP, live video, etc.

**TCP/UDP Port Numbers:**

|  |  |  |
| --- | --- | --- |
| **TCP** | **UDP** | **TCP/UDP** |
| **FTP Data** (20) | **DHCP** **Server** (67) | **DNS** (53) |
| FTP **Control** (21) | DHCP **Client** (68) |
| 22 | 69 |
| 23 | **SNMP Agent** (161) |
| 25 | SNMP **Manager** (162) |
| 80 | **Syslog** (514) |
| 110 |
| 443 |

**--------------------------------------------**

***Section 32 – IPv6 (1.8, 1.9, 3.3):***

* **Binary** (Base 2) – Can be **identified by:** **0b(X)** (**X**) = a number value
* **Decimal** (Base 10) – Can be **identified by:** **0d**
* **Hexadecimal** (Base 16) – Can be **identified by:** **0x(XX)** (**XX**) = number value

**Binary >> Hexadecimal:**

1. Split the number into 4-bit groups.
2. Convert each 4-bit group to decimal.
3. Convert each decimal number to hexadecimal.

**Hexadecimal >> Binary:**

1. Split up the hexadecimal digits.
2. Convert each hexadecimal digit to decimal.
3. Convert each decimal number to binary.

*An* ***IPv6*** *address is* ***128 bits*** *(****16 bytes****). It’s* ***represented by 8 groups*** *of* ***4 hexadecimal digits****, each group representing 16 bits.*

**Shortening IPv6 –** **Leading 0s** in each hexadecimal bit **can be removed**. **Consecutive quartets** of **all 0s** can be **replaced** with a double colon **(::)** but only **once** in a single IPv6 address.

**EX:** Full IPv6 = 2000:AB78:0020:01BF:ED89:0000:0000:0001

Shortened IPv6 = 2000:AB78:20:1BF:ED89::1

Full IPv6 = FE80:0000:0000:0000:0002:0000:0000:FBE8

Shortened IPv6 = FE80::2:0:0:FBE8

**Expanding IPv6:**

1. Put leading 0’s where needed (all quartets should have 4 hexadecimal characters).
2. If a double colon is used, replace it with all 0 quartets, ensuring there are 8 quartets in total.

**EX:** Shortened IPv6 = FE80::1010:2FC:0:9

Full IPv6 = FE80:0000:0000:0000:1010:02FC:0000:0009

Shortened IPv6 = FF02::2

Full IPv6 = FF02:0000:0000:0000:0000:0000:0000:0002

**Finding the IPv6 (Network) Prefix (Global Unicast Addresses) –** Typically, an enterprise requesting IPv6 addresses from an ISP will receive a /48 block. IPv6 subnets usually use a /64 prefix length, which means an enterprise has 16 bits to use to make subnets (48+16=64).

2001:0DB8:8B00: 0001: 0000:0000:0000:0001/64

* **48-bit** “**global routing prefix**” assigned by ISP.
* **16-bit** “**subnet identifier**”, used by the enterprise to make various subnets.
* **64-bit** “**interface identifier**”, the host portion of the address.

**Identify where** the **network bits end** **in** an **IPv6 address**, the **amount of network bits** present in an address are labeled by the prefix (**/x**). **If prefix** is a **multiple of 4**, simply shorten address up until the last network bit. **Otherwise**, **break quartet where** the **last network bit lies** into binary and convert all values after that bit to 0’s.

**EX:** Host Address = FE80:0000:0000:0000:4c2c:e2ed:6a89:2a27/9

Prefix = FE80::/9

Host Address = 2001:0DB8:001:0B23:BA89:0020:0000:00C1/64

Prefix = 2001:DB8:1:B23::/64

**Configuring IPv6:**

* **(config)#**ipv6 unicast routing *(Allows the router to perform IPv6 routing)*
* **(config)#**int *g0/0*
* **(config-if)#**ipv6 address *2001:db8:0:0::1/64*
* **(config-if)#**no shutdown

**---------------------------------**

***Section 33 – IPv6 (1.9):***

**IPv6 Modified EUI-64 (1.9d) –** **EUI** stands for Extended Unique Identifier. **Modified EUI-64** is a method of converting a MAC (48 bits) into a 64-bit interface identifier. The interface identifier can then become the ‘host portion’ of a /64 IPv6 address.

**How to convert the MAC:**

1. Divide the MAC in half = 1234 5678 90AB **>>** 1234 56 **|** 78 90AB
2. Insert FFFE in the middle = 1234 56FF FE778 90AB
3. Invert the 7th bit (**Each hexadecimal** digit is **4 bits**): 1234 56FF FE778 90AB **>>** (2 = 0010 **>>** 0000) **>>** 1034 56FF FE778 90AB

**EX:** MAC = 782B CABC 0876

EUI-64 Interface Identifier = 782B CAFF FEBC 0876 **>>** 8 = 1000 **>>** 1010 **>>** 7A2B CABC 0876

MAC = 0200 4C4F 4F50

EUI-64 Interface Identifier = 0200 4CFF FE4F 4F50 **>>** 2 = 0010 **>>** 0000 **>>** 0000 4C4F 4F50

**Configuring IPv6 (EUI-64):**

* **(config)#**int *g0/0*
* **(config-if)#**ipv6 address *2001:db8::/64* eui-64
* **(config-if)#**no shutdown

**Global Unicast (1.9a) –** **Public IPv6** address which can be used over the internet, must register to use them. Because they’re public addresses, it’s expected that they’re **globally unique**. Originally defined as the 2000::/3 block (2000:: to 3FFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF). Now defined as all addresses which aren’t reserved for other purposes.

**Unique Local Addresses (1.9a) –** **Private IPv6** addresses which **can’t** be used over the internet, don’t need to register to use them. They can be used freely within internal networks and don’t need to be globally unique (but should be) and can’t be routed over the internet. Used the address block FC00::/7 (FC00:: to FDFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF), however is now required for the 8th bit to be set to 1, so the **first two digits must be FD (FD00::/7)**.

EX: FD45:93AC:8A8F:0001:0000:0000:0000:00001/64

**Indicates** a **unique local** address. **40-bit** “**global ID**”, which should be randomly generated. **16-bit** “**subnet identifier**”, used by the enterprise to make various subnets. **64-bit** “**interface identifier**”, the **host portion** of the address.

**Link Local Addresses (1.9a) –** IPv6 addresses that are automatically enabled on IPv6-enabled interfaces. **Use command:** **(config-if)#**ipv6enable – on an interface to enable IPv6.

Uses the address block **FE80::/10** (FE80:: to FEBF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF). You won’t see link local addresses beginning with FE9, FEA, or FEB. Only FE8. The interface ID is generated using EUI-64 rules. These addresses are used for communication within a single link (subnet), **routers won’t route packets with** a **link-local destination IPv6** **address**.

**Multicast Addresses (1.9c) –** These addresses are one-to-many/one source to multiple destinations (that have joined the specific multicast group). IPv6 uses range **FF00::/8** for multicast. (FF00 to FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF). **IPv6 doesn’t use broadcast addresses**.

|  |  |  |
| --- | --- | --- |
| **Purpose** | **IPv6 Address** | **IPv4 Address** |
| All **Nodes/Hosts** (like broadcast) | FF02::1 | 224.0.0.1 |
| All **Routers** | FF02::2 | 224.0.0.2 |
| All **OSPF Routers** | FF02::5 | 224.0.0.5 |
| All OSPF **DRs/BDRs** | FF02::6 | 224.0.0.6 |
| All **RIP Routers** | FF02::9 | 224.0.0.9 |
| All **EIGRP Routers** | FF02::A | 224.0.0.10 |

**Multicast Address Scopes –** IPv6 defines multiple multicast “scopes” which indicate how far the packet should be forwarded. **IPv6 multicast scopes:**

* **Interface-local (FF01) =** **Packet doesn’t leave the device**, can be used to send traffic to a service within a local device.
* **Link-local (FF02) =** Packet remains in the local subnet; routers won’t route it between subnets
* **Site-local (FF05)**, **Organization-local (FF08)**, **Global (FF0E)**

**Multicast MACs –** The first 24 bits of a 48-bit multicast MAC are always set to **01-00-5E**, and the 25th bit is always set to 0. The remaining 23 bits are created from the last 23 bits of the multicast IP.

**Anycast Addresses (1.9b) –** IPv6 address that is **one-to-multiple**. **Multiple routers** are **configured with** the **same IPv6 address**. They use a routing protocol to advertise the address, when hosts send packets to that destination address, routers will forward it to the nearest router configured with that IP (**based on routing metric**). There’s no specific address range for anycast, **use** a **regular unicast address** (global unicast, unique local) and specify it as an anycast address.

**Anycast Address Configuration:**

* **(config-if)#**ipv6 address *2001:db8:1:1::99/128* anycast

Other IPv6 Addresses:

* **Unspecified IPv6 Address – :: =** Can be used when a device doesn’t yet know its IPv6 address. IPv6 default routes are configured to ::/0 (**Equivalent to IPv4 0.0.0.0**)
* **Loopback Address – ::1** **=** Messages sent to this address are processed within the local device but not sent to others. **IPv4 equivalent:** 127.0.0.0/8

**---------------------------------**

***Section 34 – IPv6 (3.3):***

**IPv6 Header Fields:**

* **Traffic Class =** Used for QoS, to indicate high priority traffic.
* **Flow Label =** Used to identify specific traffic ‘flows’ (communications between a specific source/destination).
* **Payload Length =** Indicates the length of the payload (encapsulated Layer 4 segment) in bytes.
* **Next Header =** Indicates the type of the ‘next header’ (header of the encapsulated segment), for example TCP/UDP. **Same function as** the IPv4 header’s ‘Protocol’ field.
* **Hop Limit =** Functions the same as the IPv4’s header ‘TTL’ field.

**Solicited-Node Multicast Address –** An IPv6 address that’s calculated from a unicast address: **ff02::1:ff + Last 6 digits of a unicast address**

**EX:** 2001:0db8:0000:0001:0f2a:4fff:fe[*a3:00b1] + ff02::1:ff* = ffo2::1:ffa3:b1

**NDP (Neighbor Discovery Protocol) –** IPv6 protocol that replaced ARP, its ARP-like function uses ICMPv6 and solicited-node multicast addresses to learn the MACs of other hosts. **2 message types** are **used:**

1. **Neighbor Solicitation (NS) =** ICMPv6 Type **135** (NDP **equivalent of** an **ARP Request**)
2. **Neighbor Advertisement (NA) =** ICMPv6 Type **136** (NDP **equivalent** of an **ARP Response**)

Since **IPv6 doesn’t use ARP**, it **also doesn’t use** an **ARP Table**, instead it **uses a Neighbor Table**.

Another function of NDP allows hosts to automatically discover routers on the local network. **2 messages** are **used for this process:**

1. **Router Solicitation (RS) =** ICMPv6 Type **133**

* Sent to multicast address **ff02::2** (all routers)
* Asks all routers on the local link to identify themselves.
* Sent when an interface is enabled/host is connected to the network.

1. **Router Advertisement (RA) =** ICMPv6 Type **134**

* Sent to multicast address **ff02::1** (all nodes)
* The router announces its presence, as well as other info about the link.
* These messages are sent in response to RS messages, they’re also sent periodically even if the router hasn’t received an RS.

**SLAAC (Stateless Address Auto-Configuration) –** Hosts use the RS/RA messages to learn the IPv6 prefix of the local link and then automatically generate an IPv6 address.

Using: **ipv6 address** *prefix/prefix-length* **eui-64** command, you need to manually enter the prefix.

Using: **ipv6 address autoconfig** command, you don’t need to enter the prefix. The device uses NDP to learn the prefix used on the local link. The device will use EUI-64 to generate the interface ID, or it will be randomly generated.

**DAD (Duplicate Address Detection) –** Allows hosts to check if other devices on the local link are using the same IPv6 address.

Any time an **IPv6-enabled interface** initializes, **or** an **IPv6 address** is **configured on** an **interface**, it performs DAD using 2 messages: NS + NA. The host will send an NS to its own IPv6 address, if it doesn’t get a reply, it knows the address is unique. If it does get a reply, it means another host on the network is using the same address.

**IPv6 Static Routing**: **(config)#**ipv6 route *destination/prefix-length next-hop exit-interface next-hop ad*

* **Directly Attached** Static Route: **Only** the **exit interface** is specified. **In IPv6**, this type of static route **won’t work on Ethernet interfaces**.

*ipv6 route destination/prefix-length exit-interface*

**EX: (config)#**ipv6 route *2001:db8:0:3::/64 s0/0*

* **Recursive** Static Route: **Only** the **next hop** is specified.

*ipv6 route destination/prefix-length next-hop*

**EX:****(config)#**ipv6 route *2001:db8:0:3::/64 2001:db8:0:12::2*

* **Fully Specified** Static Route: **Both** the **exit interface** and **next hop** are specified.

*ipv6 route destination/prefix-length exit-interface next-hop*

**EX:****(config)#**ipv6 route *2001:db8:0:3::/64 g0/0 2001:db8:0:12::2*

**Network Route:** **(config)#**ipv6 route *2001:db8:0:3::/64 2001:db8:0:12::2*

**Host Route:** **(config)#**ipv6 route *2001:db8:0:1::100/128 2001:db8:0:12::1*

**(config)#**ipv6 route *2001:db8:0:3::100/128 2001:db8:0:23::2*

**Default Route:** **(config)#**ipv6 route *::/0 2001:db8:0:23::1*

**Floating Static:** **(config)#**ipv6 route *destination/prefix-length next-hop exit-interface next-hop* ***ad***

Must specify a route’s AD and configure it higher than the main destination route’s AD to configure a **floating static IPv6 route**.

**------------------------------------------------------------------**

***Section 35 – Standard Access Control Lists (5.6):***

ACLs function as a packet filter, instructing routers to permit/discard specific traffic. They can filter traffic based on source/destination IPs, source/destination Layer 4 ports, etc. ACLs are configured in global config mode but **will not take effect unless applied to an interface**.

ACLs are applied either inbound/outbound and in an ordered sequence of **ACEs** (Access Control Entries), **meaning they must be applied in a specific desired order**. When a router checks a packet against an ACL, it processes the ACEs in order from top to bottom. If a packet matches one of the ACEs in an ACL, the router acts and stops processing the ACL, all entries below the matching entry will be ignored.

A **maximum of one ACL can be applied to a single interface per direction** (1 Inbound/1 Outbound).

**Implicit Deny –** Tells a router to deny all traffic that doesn’t match any of the configured entries in an ACL. **There’s an ‘implicit deny’ at the end of all ACLS**.

**Standard ACLs –** Match based on Source IPs only: Standard Numbered ACLs/Standard Named ACLs

**Extended ACLs –** Match based on Source/Destination IP, Source/Destination port, etc.: Extended Numbered and/or Named ACLs

**Standard Numbered ACLs** are identified with a number that can **range from 1-99 & 1300-1999**. Should be applied as close to the destination as possible.

Configure Standard Numbered ACL: **(config)#**access-list *number* {deny/permit} *ip wildcard-mask*

**Apply ACL to An Interface:** **(config-if)#**ip access-group *number* {in/out}

**Standard Named ACLs –** Still a Standard ACL, can only match traffic based on source IP of a packet. Identified with a name, once configured with that name-will enter ‘Standard Named ACL config mode’, can configure each entry within that config mode**:**

* **(config)#**ip access-list standard *acl-name*
* **(config-std-nacl)#**[entry-number] {deny/permit} *ip wildcard-mask*
* **(config-std-nacl)#**interface *int*
* **(config-if)#**ip access-group *acl-name* in/out

**----------------------------------------**

***Section 35 – Extended ACLs:***

**Advantages of Named ACL Config Mode -** You can delete specific individual ACEs by entry number from both Numbered/Named ACLs. **Command is: no** *sequence-number*. With ACLs that are configured from global config mode, you can’t do this, you can only delete the entire ACL. You **can also insert new entries** in **between other entries** by specifying the sequence number.

**Resequencing ACLs:** **(config)#**ip access-list resequence *acl-id* *starting-seq-num increment*

**EX: (config)#**ip access-list resequence 1 10 10

Change the sequence number of the first entry to 10. Add 10 for every entry after that.

**Extended ACLs –** Can be numbered/named like Standard ACLs. Numbered ACLs **range from 100-199, 2000-2699**. They’re processed from top to bottom like Standard ACLs but can match traffic based on more parameters so they’re more precise/complex than Standard ACLs.

**Configuring Extended Numbered ACLs:**

* **(config)#**access-list *number* [permit/deny] *protocol src-ip dest-ip*

**Configuring Extended Named/Numbered ACLs:**

* **(config)#**ip access-list extended *{name/number}*
* **(config-ext-nacl)#***[seq-num]* [permit/deny] *protocol src-ip dest-ip*

**Matching Protocols –** You can specify which protocol by the IP header protocol number:

* 1 **= ICMP**
* 6 **= TCP**
* 17 **= UDP**
* 88 **= EIGRP**
* 89 **= OSPF**

Can **specify ‘ip’** when wanting to set an ACL for all traffic.

**Matching TCP/UDP Port Numbers –** When matching TCP/UDP, you can optionally specify the source and/or destination port numbers to match.

**EX: (config-ext-nacl)#**deny tcp *src-ip* eq-range *src-port-num* *dest-ip* eq-range *dst-port-num*

* **eq** # = equal to port #
* **gt** # = greater than port #
* **lt** # = less than port #
* **neq** # = NOT port #
* **range** # # = from port # to port #

**Extended ACLs should be applied as close to the SOURCE as possible** to limit how far the packets travel in the network before being denied.

**-------------------------------------------**

***Section 37 – CDP & LLDP (2.3):***

**CDP/LLDP** are Layer 2 discovery protocols that share info with and discover info about connected devices. The shared info includes hostnames, IPs, device type, etc.

**CDP (Cisco Discovery Protocol) –** **Cisco proprietary**, periodically sends messages to **multicast MAC** address **0100.0CCC.CCCC**. When a device receives a CDP message, it processes and discards the message and does not forward it to other devices. CDP messages are sent **every 60 seconds**. **CDPv2** is used by default. **Does** **share** **VTP info**.

**CDP Holdtime –** By default, is **180 seconds**. If a message isn’t received from a neighbor within those 180 seconds, the neighbor is removed from the **CDP neighbor table**.

**CDP Config Commands:**

* Enable/Disable CDP Globally = **(config)#**(no) cdp run
* Enable/Disable CDP on an Interface = **(config)#**(no) cdp enable
* Configure CDP Timer = **(config)#**cdp timer *seconds*
* Configure CDP Holdtime = **(config)#**cdp holdtime *seconds*
* Enable/Disable CDPv2 = **(config)#**(no) cdp-advertise-v2

**LLDP (Link Layer Discovery Protocol) –** **Industry Standard protocol** (IEEE 802.1AB). LLDP messages are sent periodically to **multicast MAC** address **0180.C200.000E**. When a device receives an LLDP message, it processes and discards the message and does not forward it to other devices. LLDP messages are sent every **30 seconds** and the **hold time** **= 120 seconds**.

**LLDP Config Commands:**

* Enable/Disable LLDP Globally = **(config)#**(no) lldp run
* Enable/Disable LLDP on Specific Interface (**tx**) = **(config)#**(no) lldp transmit
* Enable/Disable LLDP on Specific Interface (**rx**) = **(config)#**(no) lldp receive
* Configure LLDP Timer = **(config)#**lldp timer *seconds*
* Configure LLDP Holdtime = **(config)#**lldp holdtime *seconds*

**---------------------------------**

***Section 38 – NTP (4.2):***

View the time and time source on a Cisco device’s internal clock with the command: **#**show clock detail

The **default time zone** is UTC(Coordinated Universal Time), **and** the **default time source** is the hardware calendar. The **most important reason** to **have accurate time on a device** is to ensure accurate logs for troubleshooting.

**Hardware Clock Configuration: #**calendar set

The hardware **calendar** **is** the **default** **time** **source**, but the hardware and software clocks are separate and can be configured separately.

**Manual (Software) Time Configuration:** **#**clock set

You **should synch the** “**clock**” **and** “**calendar**” by using the command: **#**clock update-calendar (**Matches hardware** clock **to software clock**) OR **#**clock read-calendar(**Matches software** clock **to hardware clock**)

**Configure Time Zone:** **(config)#**clock timezone *name hours-offset*

**Configure Daylight Savings: (config)#**clock summer-time *time-zone* recurring *start-week day month hh:mm end-week day month hh:mm hours-offset*

**Network Time Protocol –** Manually configuring time on devices is not scalable, each clock will drift. NTP allows automatic syncing of time over a network. **NTP clients** request the time from **NTP servers**. NTP allows accuracy of time within ~ 1 millisecond (If server/client are in the same LAN), or ~50 milliseconds if connecting to the NTP server over a WAN/the internet. The distance of an NTP server from the original reference clock is **called stratum**. **NTP uses UDP port 123** to communicate.

**Reference Clocks –** Usually a very accurate time device like an atomic clock/GPS clock and are ‘stratum 0’ within the NTP hierarchy. **NTP servers directly connected to reference clocks** are ‘stratum 1’.

An NTP client can synch to multiple NTP servers, and a device can operate in either one/all 3 NTP modes: **Server, Client, Symmetric Active**.

**NTP Hierarchy:**

* **Stratum 0 =** Reference Clocks
* Stratum **1 =** (**Primary Servers**) NTP servers that get their time directly from reference clocks
* Stratum **2 =** (**Secondary Servers**) NTP servers that get their time from stratum 1 NTP servers
* Stratum **3 =** (**Secondary** Servers) NTP servers that get their time from stratum 2 NTP servers
* Stratum **15 =** The max, anything above is considered unreliable

**NTP servers with lower stratum levels** are **preferred** because they’re closer to the source so they’re considered more accurate.

**Basic NTP Config Commands:**

* **(config)#**ntp server *ip-address* (prefer)
* **(config)#**ntp update-calendar
* **(config)#**ntp source *interface*

**Configure NTP Server Mode:** **(config)#**ntp master

(The **default stratum of** the **NTP master command** is 8.)

Devices can ‘**peer**’ with other devices at the same stratum to provide more accurate time.

**Configure NTP** **Symmetric Active (Peer) Mode:** **(config)#**ntp peer *address*

**NTP Authentication:**

* **(config)#**ntp authenticate
* **(config)#**ntp authentication-key *key-number* md5 *password*
* **(config)#**ntp trusted-key *key-number*
* **config)#**ntp server *ip-address* key *key-number*

**NTP Show Commands:**

* **#**show ntp associations
* **#**show ntp status

**---------------------------------------------------------**

***Section 39 – Domain Name System (4.3):***

**Standard DNS queries/responses** typically use UDP, TCP is used for DNS messages greater than 512 bytes. In either case, **port 53** is used.

**DNS ‘A’ records** map host names to IPv4 addresses. **DNS ‘AAAA’ records** map host names to IPv6 addresses.

**DNS Cache –** Devices will save the DNS server’s responses to a local DNS cache. This means they don’t have to query the server every single time they want to access a destination.

**Windows DNS Commands:**

* **C:\Users\user>**nslookup *domain-name* (Lookup info about a domain name)
* **C:\Users\user>**ipconfig /displaydns (Shows the DNS cache)
* **C:\Users\user>**ipconfig /flushdns (Clears the DNS cache)
* **C:\Users\user>**ping *ip-address* -n *number*

**Configure DNS Server in Cisco IOS:**

* **(config)#**ip dns server (Configures a router to act as a DNS server)
* **(config)#**ip host *hostname ip-address* (Configures a list of hostname/IP address mappings)
* **(config)#**ip name-server *ip-address* (Configure an external DNS server that the router will query if a requested record isn’t in its host table)
* **(config)#**ip domain lookup (Enable router to perform DNS queries, enabled by **default**. Can also be **ip domain-lookup**)
* **(config)#**ip domain name *name* (Configure the default domain name)
* **#**show hosts

**-------------------------------------------------------------------------------------**

***Section 40 –* Dynamic Host Configuration Protocol *(4.3 & 4.6):***

**DHCP** **servers use** UDP port 67 & **DHCP clients use** UDP port 68.

**DHCP Message Types:** (**D-O-R-A**)

|  |  |  |
| --- | --- | --- |
| **D**iscover | Client **>>** Server | Broadcast |
| **O**ffer | Server **>>** Client | Broadcast **OR** Unicast |
| **R**equest | Client **>>** Server | Broadcast |
| **A**ck | Server **>>** Client | Broadcast **OR** Unicast |
| Release | Client **>>** Server | Unicast |

**DHCP Relay –** Some networks are configured to have each router act as the DHCP server for its connected LANs. However, large enterprises often choose to use a centralized DHCP server. If the server is centralized, it won’t receive the DHCP clients’ broadcast DHCP messages (**broadcast messages don’t leave the local subnet**). To fix this, a router can be configured to act as a **DHCP relay agent**. The router will forward the clients’ broadcast DHCP messages to the remote DHCP server as unicast messages.

**DHCP Server Configuration:**

* **(config)#**ip dhcp excluded-address *ip-addresses* (Specify a range of addresses you may want to be reserved for network devices)
* **(config)#**ip dhcp pool *POOL\_NAME* (Create a DHCP pool)
* **(dhcp-config)#**network *ip-address subnet-mask* (OR) *CIDR* (Specify subnet of addresses to be assigned to clients)
* **(dhcp-config)#**dns-server *ip-address* (Specify the DNS server that DHCP clients should use)
* **(dhcp-config)#**domain-name *name* (Specify domain name of a network)
* **(dhcp-config)#**default-router *ip-address*
* **(dhcp-config)#**lease *days hours minutes* (OR) *infinite* (Specify the lease time)

**#**show ip dhcp binding **–** Lists all DHCP clients that are currently assigned an IP

**DHCP Relay Agent Configuration:**

* **(config)#**int *interface* (Configure the interface connected to the subnet of the client devices)
* **(config-if)#**ip helper-address *ip-address* (Configure the IP of the DHCP server)
* **(config-if)#**do show ip int *interface*

**-------------------------------------------------------------------------------**

***Section 41 – Simple Network Management Protocol (4.4):***

**SNMP** can be used to monitor the status of devices, make configuration changes, etc. There are **2 main types of devices in SNMP:**

1. **Managed Devices –** Devices being managed using SNMP (Network devices, routers, switches)
2. **Network Management Station (NMS) –** The device/devices managing the managed devices, this is the SNMP “server”

**3 main operations** used in **SNMP:**

1. Managed devices can notify the NMS of events
2. The NMS can ask managed devices for info about their status
3. The NMS can tell the managed devices to change aspects of their configuration

SNMP **Server Components:**

* **SNMP Manager =** Software on an NMS that interacts with managed devices. Receives notifications, sends requests for info/config changes, etc.
* **SNMP Application =** Provides an interface for the network admin to interact with: Displays alerts, stats, charts, etc.

SNMP **Managed Devices Components:**

* **SNMP Agent =** SNMP software running on managed devices that interacts with the SNMP Manager on the NMS
* **Management Information Base (MIB) =** The structure that contains the variables that are managed by SNMP: Each variable is identified with an Object ID **(OID)**. **Example variables =** Interface status, CPU usage, temperature, etc.

SNMP Versions:

* **SNMPv2c =** Allows the NMS to retrieve large amounts of info in a single request, more efficient. **‘c’** refers to the **‘community strings’** used as passwords in SNMPv1 and this version.
* **SNMPv3 =** Much more secure version that supports strong encryption/authentication.

**SNMP Messages:**

* **Read =** **(Get/GetNext/GetBulk) –** Request sent from a manager to an agent to retrieve the value of a variable (OID) or multiple variables. The agent will send a Response message with the current value of each variable.
* **Write =** **(Set) –** Messages sent by the NMS to change info/value of one or more variables on managed devices. EX: (Change an IP). Agents will send a *Response* message with the new values.
* **Notification =** Messages sent by managed devices to alert the NMS of an event. **2 types:**

**(Trap) –** Notification sent from agent to manager that **does not** receive a *Response* to acknowledge that it was received, these messages are unreliable.

**(Inform) –** Notification message that **is** **acknowledged** with a *Response* message.

* **Response =** Messages sent in response to a previous message/request.

**SNMP Agent uses UDP port 161.**

**SNMP Manager (NMS) uses UDP port 162.**

**------------------------------------**

***Section 42 – Syslog (4.5):***

Syslog can be used on network devices to log events such as interface status, changes in OSPF neighbors, system restarts, etc. These messages can be displayed in the CLI, saved in a device’s RAM, or sent to an external Syslog server.

**Syslog Message Format –** seq:time stamp: %facility-severity-MNEMONIC:description

* **seq =** A sequence number indicating the order of messages.
* **time stamp =** Indicates the time the message was generated.

*(The above 2 fields may or may not be displayed depending on a device’s configuration.)*

* **facility =** Value that indicates which process on a device generated the message.
* **severity =** Indicates the severity of a logged event (**8 total severity levels**).
* **MNEMONIC =** A short code for the message indicating what happened.
* **description =** Detailed info about the event being reported.

**Syslog Severity Levels:** (**E**very **A**wesome **C**isco **E**ngineer **W**ill **N**eed **I**ce-cream **D**aily)

|  |  |  |
| --- | --- | --- |
| **Level** | **Keyword** | **Description** |
| **0** | **Emergency** | *System is unusable* |
| **1** | **Alert** | *Action must be taken immediately* |
| **2** | **Critical** | *Critical conditions* |
| **3** | **Error** | *Error conditions* |
| **4** | **Warning** | *Warning conditions* |
| **5** | **Notice** | *Normal but significant condition (****Notification****)* |
| **6** | **Informational** | *Informational messages* |
| **7** | **Debugging** | *Debug-level messgages* |

Syslog Logging Locations:

* **Console Line –**  Syslog messages will be displayed in the CLI when connected to a device via the console port (**All messages level 0-7** are displayed by default).
* **VTY Lines –** Syslog messages displayed in the CLI when connected to a device via Telnet/SSH (**Disabled by default**).
* **Buffer –** Syslog messages will be saved to RAM. All messages (level 0-7) are displayed by default. View the messages with the command: **#**show logging
* **External Server –** A device can be configured to send Syslog messages to an external server. **Syslog servers will listen for messages on UDP port 514.**

**--------------------------------**

***Section 43 – SSH (4.8):***

**Layer 2 Switch Management IP –** Layer 2 switches can be assigned a Switch Virtual Interface (SVI) to allow remote connections to its CLI (via SSH/Telnet). Must configure the SVI and the switch’s default gateway as well.

**SSH Device Support –** IOS images that support SSH will have ‘K9’ in their name.

**To enable and use SSH**, RSA public and private keys must be generated**:**

* **(config)#**ip domain name *fqdn*
* **(config)#**crypto key generate rsa
* **(config)#**do sh ip ssh

**SSH Configuration – VTY Lines:**

* **(config)#**enable secret *password*
* **(config)#**username *user* secret *password*
* **(config)#**access-list 1 permit host *ip-address*
* **(config)#**ip ssh version 2 (Optional)
* **(config)#**line vty 0 15 (Configure all 16 VTY lines)
* **(config-line)#**login local (Enable local user authentication)
* **(config-line)#**exec-timeout *seconds* (Optional)
* **(config-line)#**transport input ssh (Best practice to limit VTY line connections to SSH only)
* **(config-line)#**access-class 1 in (Apply the ACL to restrict VTY line connections)

**SSH Configuration:**

1. Configure hostname
2. Configure DNS domain name
3. Generate RSA key pair
4. Configure enable PW, local username/PW
5. Enable SSHv2 (only)
6. Configure VTY lines (‘**transport input ssh**’)

**------------------------------------------**

***Section 44 – FTP & TFTP (4.9):***

**FTP** and **Trivial FTP** are protocols used to transfer files over a network. **Network engineers most commonly use** them for the process of upgrading the OS of a network device. You can download the newer version of IOS from a server and then reboot the device with the new IOS image.

**TFTP –** Named “Trivial” because it’s simple and has only basic features compared to FTP, it only allows a client to copy a file to/from a server. **Not** a **replacement for FTP**, used when lightweight simplicity is more important than functionality. Provides no authentication or encryption, best used in a controlled environment to transfer small files quickly. **TFTP servers listen on UDP port 69.**

**TFTP Reliability –** Every TFTP data message is acknowledged using “**lock-step**” communication: **If a client is transferring** a **file to a server**, the server will send Ack messages (and vice-versa). Timers are used and if an expected message isn’t received in time, the waiting device will resend its previous message.

TFTP Connections – **TFTP file transfers have 3 phases:**

1. **Connection =** TFTP client sends a request to the server and the server responds back, initializing the connection.
2. **Data Transfer =** The client and server exchange TFTP messages, one sends data and the other sends acks.
3. **Connection Termination =** After the last data message is sent, a final ack is sent to terminate the connection.

**FTP –** **Uses TCP ports 20 + 21**. Usernames/PWs are used for authentication but there’s no encryption. **For better security**, FTPS (**FTP over SSL/TLS**) can be used. SSH FTP (**SFTP**) can be used for even greater security.

FTP allows file transfers, navigation of file directories, listing of files, etc. A client sends FTP commands to a server to perform these functions.

**FTP Control Connections** (**TCP 21**) **–** Established and used to send FTP commands/replies. When files/data are to be transferred, separate **FTP Data** (**TCP 20**) **Connections** are established and terminated as needed.

**Active Mode Connection –** Default method of establishing FTP data connections in which the **server initiates** the TCP **connection**.

**Passive Mode Connection –** The **client initiates** the data **connection**; this is often necessary when the client is behind a firewall that could block the incoming connection from the server.

**Copying Files (TFTP):**

* **#**copy *source destination* / Then enter the TFPT server IP, enter the file name, and enter the name you want to save the file as on flash.
* **#**show flash **–** Shows the content of a device’s flash.
* **Upgrading Cisco IOS:** **(config)#**boot system *filepath* / If you don’t use this command, the device will use the first IOS file it finds in flash to boot.

(**Save** the **running config to** the **device before rebooting**.)

* **Check** current **running version of IOS:** **#**show version

**Copying Files (FTP):**

* **(config)#**ip ftp username *user*
* **(config)#**ip ftppassword *password*

*^ (Credentials the device will use to connect to FTP server, must be configured on server too.)*

* *Rest of the process is the same as TFTP.*

**---------------------------------**

***Section 45 – NAT (4.1):***

**RFC 1918 Private IPv4 Addresses:**

* 10.0.0.0/8 (10.0.0.0 – 100.255.255.255)
* 172.16.0.0/12 (172.16.0.0 – 172.31.255.255)
* 192.168.0.0/16 (192.168.0.0 – 192.168.255.255)

**Network Address Translation –** Used to modify the source/destination IP addresses of packets, allowing hosts with private IPs to communicate over the internet.

**Static NAT –** Involves statically configuring one-to-one mappings of private IPs to public IPs (these are **permanent until deleted manually**). Because it **requires one IP to be mapped to one device**, it doesn’t help preserve IPs. An **inside local** IP is mapped to an **inside global** IP:

* **Inside Local =** The IP of an inside host, from the perspective of the local network. Usually a private IP.
* **Inside Global =** The IP of the inside host, from the perspective of outside hosts (**IP of** the inside **host after NAT**). Usually a public IP.

**Static NAT Configuration:**

* **(config-if)#**ip nat inside
* **(config-if)#**int *interface*
* **(config-if)#**ip nat outside

*^**(Define which interfaces on a device are connected to either internal/external networks.)*

* **(config)#**ip nat inside source static *inside-local-ip inside-global-ip*

**Outside Local =** The IP of an outside host, from the perspective of a local network.

**Outside Global =** The IP of an outside host, from the perspective of external networks.

**-------------------------------------**

***Section 46 – NAT (Part 2):***

**Dynamic NAT –** Router dynamically maps *inside local* addresses to *inside global* addresses as needed. An **ACL** is used to identify which traffic should be translated: **If source IP** is **permitted by the ACL**, it will be translated. **If** the **source IP** is **denied by** the **ACL**, it won’t be translated (the **traffic WON’T be dropped**).

**NAT Pool =** Used to define the range of available inside global IPs.

**NAT Pool Exhaustion =** Although **NAT mappings** are **dynamically assigned**, they are **still one-to-one** (one *inside local* IP per one *inside global* IP). If there aren’t enough *inside global* IPs available (all are currently being used), the NAT pool has been exhausted. **Hosts that seek NAT translation during this** will have their packets dropped until an *inside global* IP becomes available.

Unlike Static NAT, **Dynamic NAT mappings** are not permanent and eventually time out if not used by a host.

**Dynamic NAT Configuration:**

* **(config-if)#**ip nat inside
* **(config-if)#**int *interface*
* **(config-if)#**ip nat outside

*^**(Define which interfaces on a device are connected to either internal/external networks.)*

* **(config-if)#**exit
* **(config)#**access-list *1* permit *ip-address wildcard-mask*
* **(config)#**ip nat pool *pool-name start-ip end-ip* prefix-length *#*
* **(config)#**ip nat inside source list *1* pool *pool-name*

**Port Address Translation (PAT)** AKA (**NAT Overload**) – Translates both the IP and port number (if necessary). Uses a unique port number for each communication flow, so a single public IP can be used by many different internal hosts.

**PAT Configuration:**

* **(config-if)#**ip nat inside
* **(config-if)#**int *interface*
* **(config-if)#**ip nat outside

*^**(Define which interfaces on a device are connected to either internal/external networks.)*

* **(config-if)#**exit
* **(config)#**access-list *1* permit *ip-address wildcard-mask*
* **(config)#**ip nat inside source list *1* interface *int* overload

**--------------------------------------**

***Section 47 – QoS Pt.1 (4.7):***

**Quality of Service –** A set of tools used by network devices to manage characteristics of network traffic such as:

1. **Bandwidth =** The overall capacity of the link, measured in bits per second. QoS tools allow you to reserve a certain amount of a link’s bandwidth for specific kinds of traffic. **EX:** 20% voice traffic, 30% for specific data traffic, 50% for all other traffic.
2. **Delay =** The amount of time it takes traffic to go from source to destination **= one-way delay**. The amount of time it takes traffic to go from source to destination and return **= two-way delay**.
3. **Jitter =** The variation in one-way delay between packets sent by the same app. **IP phones have** a ‘**jitter buffer**’ to provide a fixed delay to audio packets.
4. **Loss =** The % of packets sent that don’t reach their destination. **Can be caused by** faulty cables and by a device’s packet queues that get full which cause the device to start discarding packets.

**Minimum Standards for Acceptable Interactive Audio Quality:**

* **Delay =** 150 ms or less
* **Jitter =** 30 ms or less
* **Loss =** 1% or less

**QoS Queuing –** If a network device receives messages faster than it can forward them out of the appropriate interface, the messages are placed in a queue. **By default, queued messages will be forwarded in a First In First Out** (FIFO) **manner**.

**Tail Drop –** If a queue is full, new packets will be dropped, this is **called a tail drop**. Harmful because itcan lead to **TCP global synchronization**. **When** the **queue fills up and tail drop occurs**, all TCP hosts sending traffic will slow down the rate at which they send traffic. They will all then increase the rate at which they send traffic, which rapidly leads to more congestion, dropped packets, and the process repeats again.

**Random Early Detection (RED**) **–** Solution to prevent **tail drop** and **TCP global synchronization**. When the traffic in a queue reaches a certain point, the device will start randomly dropping packets from certain TCP flows. Those flows that dropped packets will reduce the rate at which traffic is sent but you will avoid TCP synchronization. **In standard RED all** **kinds of traffic are treated the same**.

**Weighted Random Early Detection (WRED)** – **Improved version of RED** that allows you to control which packets are dropped depending on the traffic class.

**-----------------------------------------------**

***Access Ports Data & Voice (2.1a) –*** IP phones use VoIP tech to enable phone calls over an IP network, they’re connected to a switch just like any other end host. **IP phones have an internal 3-port switch:**

* 1 port is the “**uplink**” to the external switch
* 1 port is the “**downlink**” to the PC
* 1 port connects internally to the phone itself

**This allows** a **PC and IP phone to share** a **single switch port**. Traffic from the PC passes through the IP phone to the switch. It’s recommended to separate “**voice**” traffic (from the IP phone) and “**data**” traffic (from the PC) by placing them in separate VLANs. This can be done by using a voice VLAN. Traffic from the PC will be untagged but traffic from the phone will be tagged with a VLAN ID.

**Configure Access and Voice Ports:**

* **(config)#**int g0/0
* **(config-if)#**switchport mode access
* **(config-if)#**switchport access vlan 10
* **(config-if)#**switchport voice vlan 11

(PC will send traffic untagged as normal. The switch will use CDP to tell PH to tag its traffic in VLAN 11).

**Although** this **interface sends/receives traffic from 2 VLANs**, its considered an access port, not a trunk.

Recommended standards for acceptable interactive audio quality are a Delay of 150 ms or less, Jitter of 30 ms or less, and Loss of 1% or less.

**------------------**

***PoE (1.1h) –*** Allows Power Sourcing Equipment (**PSE**) to provide power to Powered Devices (**PD**) over an ethernet cable. Typically, the PSE is a switch, and the PDs are IP phones, IP cameras, WAPs, etc. The **PSE** receives AC power from the outlet, converts it to DC power, and supplies that to the **PDs**.

**When** a **device is connected to** a **PoE-enabled port**, the PSE (switch) sends low power signals, monitors the response, and determines how much power the PD needs. **If** the **device needs power**, the PSE supplies the power to allow the PD to boot. The PSE continues to monitor the PD and supply the required amount of power.

**Power Policing –** Can be configured to prevents PDs from drawing too much power:

* **#**power inline police **=** Configures power policing with the default settings (**disable** the **port and send** a **Syslog message if** a **PD draws too much power**). The interface will be put in an ‘error-disabled’ state and can be re-enabled with ‘**shutdown**’ then ‘**no shutdown**’.
* **#**power inline police action log **= Doesn’t** shut down the interface if PD draws too much power, will restart the interface and send a Syslog message.

**---------------------------------------**

***Section 48 – QoS Pt. 2 (4.7):***

**Classification –** Organizes network traffic (**packets**) into traffic classes (**categories**). To give priority to certain types of traffic, you must identify which types of traffic to give priority to. Some **methods of classifying traffic are:**

* An **ACL –** Traffic that’s permitted by an ACL will be given certain treatment, other traffic won’t.
* **NBAR (Network Based Application Recognition) –** Performs a deep packet inspection, looking beyond Layer 3/4 info up to Layer 7 to identify specific kinds of traffic.
* *In* ***Layer 2/3 headers*** *there are* ***specific fields used for classification:***
* **PCP** aka **CoS** (Layer 2)
* **DSCP** (Layer 3)

**Marking =** Refers to setting the values of certain fields in Layer 2/3 headers for use in Classification.

**Layer 2 Marking for QoS Classification:**

**PCP (Priority Code Point)** *aka* **CoS (Class of Service)** – Field of the 802.1Q tag (in Ethernet Header) that can be used to identify high/low priority traffic (**Only when there is a dot1q tag**). 3 bits in length and contains **8 possible values** but **only 3** are **CCNA relevant:**

* **0 = Best Effort** – Means there’s no guarantee that data is delivered or that it meets any QoS standard, this is regular traffic.
* **4 = Video**
* **3 = Critical Apps**
* **5 = Voice** – IP phones mark call signaling traffic (**used to establish calls**) as PCP3, but they mark actual voice traffic as PCP5.

**Since PCP resides in** the **dot1q header,** it **can only be used over** the **following connections:** Trunk links, Access links **with** a voice VLAN.

**IP ToS Byte** **– IP Header field used in Layer 3 Marking for QoS Classification:**

**DSCP (Differentiated Services Code Point) –** Field of the IP header that can also be used to identify high/low priority traffic:

* **DF (Default Forwarding)** = Used for best-effort traffic and has a DSCP marking of 0.
* **EF (Expedited Forwarding)** = Used for traffic that requires low loss/latency/jitter(usually voice) and has a DSCP marking of 46.
* **AF (Assured Forwarding)** = Defines **4 traffic classes**, all packets in a class have the same priority. Within each class there’s **3 levels of drop precedence:**
* **Higher Class =** Higher priority.
* **Higher Drop Precedence =** More likely to drop packet during congestion.

(32) (16) (8) (4) (2) (1)

4 2 1 **|** 2 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0/1 | 0/1 | 0/1 | 0/1 | 0/1 | 0 |  |  |

**^** *Class* **^^** *Drop Precedence* **^ ^** *Always zero*

***Written as =*** *AFXY****:***

* ***X =*** *The decimal value of the Class.*
* ***Y =*** *The decimal value of the Drop Precedence.*

**Formula to convert AF** value **to** decimal **DSCP** value**:** AF(XY) = 8(X) + 2(Y) = DSCP(Z)

* **CS (Class Selector)** = Defines 8 DSCP values for backward compatibility with IPP. The 3 added bits for DSCP are set to 0 and the original IPP bits are used to make 8 values.

(32) (16) (8) (4) (2)

4 2 1 **|** 2 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0/1 | 0/1 | 0/1 | 0 | 0 | 0 |  |  |

IPP: 0 1 2 3 4 5 6 7

CS: CS0 CS1 CS2 CS3 CS4 CS5 CS6 CS7

DSCP: 0 8 16 24 32 40 48 56

**Formula to convert CS to** decimal **DSCP** Value**:** CS(X)8 = DSCP(Y)

**RFC 4954 =** Developed to bring all the layer 2/3 QoS classification markings together and standardize their use. Specific recommendations:

* **Voice Traffic =** EF
* **Interactive Video =** AF4x
* **Streaming Video =** AF 3x
* **High Priority Data =** AF2x
* **Best Effort =** DF

**Trust Boundaries –** Defines where devices trust/don’t trust the QoS markings of received messages:

* **If** the **markings are trusted** = Device will forward the message without changing markings.
* **If** the **markings aren’t trusted** = Device will change the markings according to the configured policy.

*(****If*** *an* ***IP phone*** *is* ***connected to a switch port****, it’s recommended to move the trust boundary to the IP phones.)*

**Queueing/Congestion Management –** An essential part of QoS is the use of multiple queues, this is where classification helps devices match traffic based on various factors (**EX:** DSCP markings in IP header) and then place it in the appropriate queue.

**A device** is **only able to forward one frame out of an interface at a time**, so a **scheduler** is used to decide which queue traffic is forwarded from next. **Prioritization** allows the scheduler to give certain queues more priority than others.

**CBWFQ** **(Class-Based Weighted Fair Queuing) –** Method of scheduling that uses a weighted round-robin scheduler while guaranteeing each queue a certain percentage of the interface’s bandwidth during congestion.

* **Round-Robin =** Packets are taken from each queue in order, cyclically.
* **Weighted =** More data is taken from high priority queues each time the scheduler reaches that queue.

*(Round-robin scheduling isn’t ideal for voice/video traffic because it can delay and jitter since even the high priority queues must wait their turn in the scheduler.)*

**LLQ** **(Low Latency Queuing) –** Designates one (or more) queues as strict priority queues. This means that if there’s traffic, the scheduler will always take the next packet from the strict priority queue, until the strict priority queue is empty. This is **effective for reducing** the **delay and jitter of voice/video traffic but** has the **downside** of potentially starving other queues if there’s always traffic in the designated strict priority queue.

*(Policing can control the amount of traffic allowed in the strict priority queue so that it can’t take all of a link’s bandwidth.)*

**QoS Process:** Classification **>>** Queuing **>>** Scheduling **>>** Actual Data Transmission

**Shaping/Policing =** Both used to control the rate in which traffic is sent/received:

* **Shaping –** **Buffers traffic** in a queue if the traffic rate goes over the configured rate.
* **Policing** **–** **Drops traffic** (can also re-mark traffic instead) if the traffic rate goes over the configured rate.
* ‘**Burst**’ **traffic** **over the configured rate** is allowed for a short period of time, this accommodates data apps that are ‘bursty’ in nature. The amount of burst traffic allowed is configurable. In both cases, classification can be used to allow for different rates for different kinds of traffic.

**----------------------------------------------------------------------**

***Section 49 – Security Fundamentals (5.1, .2, .4, .8):***

**Vulnerability –** Any potential weakness that can compromise the CIA (Confidentiality, Integrity, Availability) of a system/info.

**Exploit –** Something that can potentially be used to exploit the vulnerability.

**Threat –** The potential of a vulnerability to be exploited.

**Reflector/Amplification Attacks –** A **Reflection attack** is when an attacker sends traffic to a reflector and spoofs the source address of its packets using the target’s IP. The reflector (**EX: A DNS server**) sends the reply to the target’s IP, if the amount of traffic sent is large enough this can result in a DoS.

A Reflection attack becomes an **Amplification attack** when the amount of traffic sent by the attacker is small but triggers a large amount of traffic to be sent from the reflector to the target.

***AAA (5.8):***

* **Authentication –** The process of verifying a user’s identity (**logging in**).
* **Authorization –** The process of granting/restricting the user’s appropriate access/permissions.
* **Accounting –** The process of recording a user’s activities on a system (**logging changes**).

Enterprises usually use a AAA server to provide AAA services, ISE (Identity Services Engine) is Cisco’s AAA server. **AAA servers usually support 2 protocols:**

* **RADIUS** is open standard (**UDP ports 1812/1813**)
* **TACACS+** is Cisco proprietary (**TCP port 49**)

**--------------------------------------------**

***Section 50 – Port Security (5.7):***

**Port Security –** Allows you to control which source MAC(s) are allowed to enter a switchport, if an unauthorized MAC enters the port an action will be taken. The **default action is to** place the interface in an “err-disabled” state.

**When** you **enable port security on an interface**, the default settings are to only allow one MAC. You can configure this manually, if you don’t the switch will allow the first source MAC that enters the interface. The **max number of MACs allowed** can be changed.

**Port Security can be enabled on** an access port or trunk port, but they must be statically configured as either access/trunk (**can’t be dynamic**).

**#**show port-security int *interface***:****Shows** some **port security info** for the **specified interface**.

**#**show errdisable recovery: **Shows all possible causes of err-disabled** **and** the **current states of configured recovery actions for each**.

**Err-Disable Recovery** is **useless if** you don’t remove the device that caused the interface to enter the err-disabled state first.

There are **3 different violation modes that determine** what the switch will do if an unauthorized frame enters an interface configured with port security:

* **Shutdown =** **Shuts down** the **port** by placing it in an err-disabled state. **Generates** a Syslog and/or SNMP message when the interface is disabled.
* **Restrict =** **Switch discards traffic** from unauthorized MACs, the interface is **not disabled**. **Generates** a Syslog and/or SNMP message each time an unauthorized MAC is detected, the **violation counter** is incremented by 1 for each unauthorized frame.
* **Protect =** **Switch discards traffic** from unauthorized MAC, the interface is **not disabled**. It **does not generate** a Syslog/SNMP message for unauthorized traffic and **does not increment** the violation counter.

**Secure MAC Aging –** By default secure MACs won’t age out, this **can be configured:** **(config-if)#**switchport port-security aging time *minutes*

**Aging Type: Absolute** (**Default**) **=** After the secure MAC is learned, the aging timer starts and the MAC is removed after the time expires, **even if** the **switch continues receiving frames** from that source MAC.

**Inactivity =** After the secure MAC is learned, the aging timer starts but is reset every time a frame from that source MAC is received on the interface. This is **configured with:** **(config-if)#**switchport port-security aging type absolute (*OR*) inactivity

**Secure Static MAC Aging:**

* **Addresses configured with:** **(config-if)#**switchport port-security mac-address *x.x.x* =Disabled by default (MAC won’t age out).
* **Can be enabled** (configured to age out) **with:** **(config-if)#**switchport port-security aging static

**Sticky Secure MAC Addresses –** Can be enabled with: **(config-if)#**switchport port-security mac-address sticky

**^** (**When enabled**, dynamically learned secure MACs will be added to the running config) **^**

A **Sticky secure MAC will never age out**, but you need to save the running-config to the startup-config to make them permanent. **When you enter the command:** **(config-if)#**switchport port-security mac-address sticky, all current sticky secure MACs will be converted to regular dynamically learned secure MACs.

**------------------------------------------------**

***Section 51 – DHCP Snooping (5.7):***

**DHCP Snooping** – Security feature of switches that’s used to filter DHCP messages received on untrusted ports. **Only filters** DHCP messages, non-DHCP messages aren’t affected. All ports are untrusted by default, usually **uplink ports** (ports that point **toward** network devices) are configured as trusted and **downlink ports** remain untrusted.

**DHCP Starvation –** An attacker uses spoofed MACs to flood DHCP Discover messages. The target server’s DHCP pool becomes full, resulting in a DoS to other devices.

**DHCP Poisoning –** Can be used to perform a MIM attack, an illegitimate DHCP server replies to clients’ DHCP Discover messages and assigns them IPs but makes the clients use the illegitimate server’s IP as the default gateway. Clients then send traffic to the attacker instead of the legitimate default gateway, and the attacker can examine/modify the traffic before forwarding it the legitimate gateway.

**DHCP Messages =** DHCP Snooping filters messages, it differentiates between DHCP Server and DHCP Client messages.

**DHCP Server Messages:**

* OFFER
* ACK
* NAK = Opposite of ACK, used to decline a client’s REQUEST

**DHCP Client Messages:**

* DISCOVER
* REQUEST
* RELEASE
* DECLINE

**DHCP Snooping Operations =** If a DHCP message is received on an untrusted port, **inspect** it **and act as follows:**

* **If** it’s a **DHCP Server message =** Discard it.
* **If** it’s a **DHCP Client message =** Perform the following checks**:**
* **DISCOVER/REQUEST messages –** Check if the frame’s source MAC and the DHCP message’s CHADDR (**Client Hardware Address**) fields match. **Match =** forward, **mismatch =** discard.
* **RELEASE/DECLINE messages =** Check if the packet’s source IP and the receiving interface match the entry in the DHCP Snooping Binding Table. **Match =** forward, **mismatch =** discard.
* When a client successfully leases an IP from a server, it creates a new entry in the **DHCP Snooping Binding Table**.

**DHCP Snooping Configuration:**

* **(config)#**ip dhcp snooping
* **(config)#**ip dhcp snooping vlan *number (****Enable for each desired VLAN****)*
* **(config)#**no ip dhcp snooping information option *(Removes Option 82 from DHCP messages)*
* **(config)#**int *interface*
* **(config-if)#**ip dhcp snooping trust

***^*** *(By* ***default****, all ports will be untrusted. Must manually specify which ports to trust.)*

* **#**show ip dhcp snooping binding

**DHCP Option 82 =** ‘**DHCP relay agent info**’ provides additional info about which DHCP relay agent received the client’s message. **By default**, **with DHCP snooping enabled**, Cisco switches will add Option 82 to DHCP messages they receive from clients. This **causes problems because Cisco switches** will drop DHCP messages with Option 82 that are received on an untrusted port.

**DHCP Snooping Rate-Limiting –** DHCP Snooping can limit the rate at which DHCP messages are allowed to enter an interface (**Useful to protect against DHCP exhaustion** attacks). IF the rate of DHCP messages crosses the configured limit, the interface is err-disabled. **Like with Port Security**, the interface can be manually re-enabled, or automatically re-enabled with err-disable recovery.

**DHCP Snooping Rate-Limiting Configuration:**

* **(config)#**int *interface*
* **(config-if)#**ip dhcp snooping limit rate *#*
* **(config)#**errdisable recovery cause dhcp-rate-limit

***^*** *(Enable err-disable recovery for DHCP Rate-Limiting)*

* **#**show errdisable recovery

**------------------------------------------------------------**

***Section 52 – Dynamic ARP Inspection (5.7):***

**DAI –** A security feature of switches that’s used to filter ARP messages received on untrusted ports; **all ports** are **untrusted by default**. Usually, all **ports connected to other network devices** should be configured as trusted and **interfaces connected to end hosts** should remain untrusted.

**ARP Poisoning (MITM) –** Similar to DHCP poisoning, ARP poisoning involves an attacker manipulating targets’ ARP tables, so traffic is sent to the attacker. **To do this**, an attacker can send gratuitous ARP messages using another device’s IP **or** respond to a legitimate ARP request.

**DAI inspects** the **sender MAC/IP fields of ARP messages received on untrusted ports** and checks that there’s a matching entry in the DHCP snooping binding table. **If there’s a matching entry,** the ARP message is forwarded, **if not** its discarded.

**ARP ACLs –** Can be configured to map IPs/MACs for DAI to check. DAI can also be configured to conduct more in-depth checks. **DAI also** **supports rate-limiting** to prevent attackers from overwhelming a switch with ARP messages.

**DAI Configuration:**

* **(config)#**ip arp insepection vlan *number (Enable for each desired VLAN)*
* **(config)#**interface range *interfaces*
* **(config-if-range)#**ip arp inspection trust
* **#**show ip arp inspection interfaces

**DAI Rate Limiting:**

* **(config)#**interface range *interfaces*
* **(config-if-range)#**ip arp inspection limit rate *packets* burst interval *seconds*

*(****If ARP messages*** *are* ***received faster than*** *the* ***specified rate****, the interface will be err-disabled.)*

It can be **re-enabled in 2 ways:**

* **(config)#**errdisable recovery cause arp-inspection

*OR*

* **(config-if)#**shutdown *(AND)* **(config-if)#**no shut

**DAI Optional Checks:**

* **(config)#**ip arp inspection validate (*dst-mac* **|** *ip* **|** *src-mac)*

*(****Must enter all*** *the* ***desired validation checks*** *in a single command, not one by one.* ***Otherwise,******only*** *the* ***last entered validate command*** *will be saved.)*

**--------------------------------------------------------------**

***Section 53 – LAN Architectures (1.2a, b, c, e):***

**Star =** Several devices all connect to one central network device called a ‘**star topology**’.

**Full Mesh =** When each device is connected to each other device.

**Partial Mesh =** Some devices are connected to each other but not all.

**Two-Tier** (**Collapsed Core**) **–** Consists of 2 hierarchical layers: Access + Distribution Layer (**Aggregation Layer**).

* **Access Layer –** Layer that end hosts connect to, typically they have lots of ports for those connections. **QoS marking, DAI, port security and other security services** are typically performed here.
* **Distribution Layer** (**Core-Distribution Layer**) **–** Aggregates connections from the Access Layer switches. Typically, is the border between Layer 2 and Layer 3. **Connects to services like** Internet, WAN, etc.

**Three-Tier Campus LAN Design =** In large LAN networks with many Distribution Layer switches, the number of connections required between Distribution Layer switches grows rapidly. **Cisco recommends adding a Core Layer** if there are more than 3 Distribution Layers in a single location.

**Three-Tier –** Access, Distribution (**Aggregation Layer**), and Core Layers.

* **Core Layer –** Connects Distribution Layers together in large LAN networks. The focus is speed (‘**fast transport**’). CPU-intensive operations such as security, QoS, etc. **should be avoided at this layer**. Should maintain connectivity throughout the LAN **even if devices fail**. Connections are all Layer 3, **no STP**.

**Spine-Leaf –** Designed for data centers. **Every Leaf switch** is connected to **every Spine switch** and vice-versa.

* **Leaf switches = Don’t connect** to other Leaf switches.
* **Spine switches = Don’t connect** to other Spine switches.
* **End hosts** **=** Connect only to Leaf switches.

**The path taken by traffic** is randomly chosen to balance the traffic load among the Spine switches. **Each server** is separated by the same number of hops (**except those connected to the same Leaf**), providing consistent latency for East-West traffic.

**SOHO Networks –** All network functions are typically provided by a single device (**router**). That one device can serve as a Router, Switch, Firewall, WAP, Modem, etc.

**------------------------------------------------------------**

***Section 54 – WAN Architecture (1.2.d, 5.5):***

**Hub-and-Spoke Topology =** Network design where a central device, **the hub** (**EX:** Data center), is connected to multiple other devices/end hosts (**the spokes**).

**Leased Lines (Serial) –** A dedicated physical link that typically connects 2 sites and uses serial connections (**PPP** & **HDLC** encapsulations). Various **standards with different speeds:** North America=**T1:** 1.544 Mbps, T2, T3 / Europe=E1, E2, E3.

**MPLS (Multi-Protocol Label Switching) –** Service providers’ shared infrastructure that multiple enterprises connect to and share to make WAN connections. The **label switching in MPLS** allows VPNs to be created over the MPLS infrastructure by using labels. MPLS uses those labels to forward traffic, not destination IPs. **MPLS Infrastructure terms:**

* **CE Router =** Customer Edge Router (Sit at edge of customer’s network & connect to PEs)
* **PE Router =** Provider Edge Router (Sit at edge of provider’s network & connect to CEs)
* **P Router =** Provider Core Router (**Don’t connect to CEs**)

*(****When PE routers receive frames from CE routers****, they add a label to the frame. These labels sit between layers 2/3 and are used to make forwarding decisions within the service provider network.* ***Only*** *the* ***PE/P routers use MPLS****,* ***not CE routers****.)*

**Layer 3 MPLS VPN –** **CE and PE routers peer** using a routing protocol to share routing info. **CEs from different locations** will learn about each other’s network through this peering.

**Layer 2 MPLS VPN –** **CE and PE routers don’t form peering’s**; The service provider network is transparent to CE routers (acting like a switch). The **CE routers will have** their **WAN interfaces in the same subnet**, making it like they’re directly connected. **If** a **routing protocol is used** the CE routers will peer directly with each other.

**DSL (Digital Subscriber Line) –** Provides internet connectivity over phone lines and can share the same phone lines that’s already installed in most homes**.** A **DSL Modem** is **required** (sometimes built into router)to convert data into a format that can be sent over the phone lines.

**CATV (Cable Television**) – Provides internet via the same cable TV lines used for TV service. A **cable modem** (sometimes built into router) is **required** to convert data into a format that can be sent over CATV cables.

**Redundant Internet Connections:**

* **Single Homed =** 1 connection to 1 ISP (**No redundancy**)
* **Dual Homed =** 2 connections to 1 ISP
* **Multihomed =** 1 connection **each** to separate ISPs
* **Dual Multihomed =** 2 connections **each** to 2 ISPs

**Internet VPNs = Private WAN services** **(leased lines** **and MPLS)** provide security because each customer’s traffic is separated by using dedicated physical connections or by MPLS tags. However, **when using the internet as a WAN to connect sites**, there’s no built-in security by default. **For this, VPNs are used:**

**IPSec (Site-to-Site) –** A VPN between 2 devices and is used to connect 2 sites together over the internet. A **VPN “tunnel”** is created between the 2 devices by encapsulating the original IP packet with a VPN header and a new IP header. **Devices/end hosts connected to these now encrypted endpoints**, send unencrypted data to their site’s designated router, which will then encrypt and forward it In the IPSec tunnel to the destination site.

**IPSec Transport Mode –** Does not encrypt the IP header, only the IP packet’s payload is encrypted.

**IPSec doesn’t support** broadcast/multicast traffic, only unicast. This means that **routing protocols can’t be used over the tunnels** because they rely on multicast (can be solved with GRE over IPSec).

**GRE (Generic Routing Encapsulation) Over IPSec –** GRE creates tunnels like IPSec but doesn’t encrypt the original packet. However, it has the advantage of being able to encapsulate a variety of Layer 3 protocols as well as **broadcast/multicast messages**. **GRE over IPSec can be used to** get the flexibility of GRE with the security of IPSec.

***^*** *(An* ***original packet*** *is encapsulated by a GRE header and a new IP header,* ***then*** *the* ***GRE packet*** *will be encrypted and encapsulated within an IPSec VPN header and new IP header.)*

**DMVPN (Dynamic Multipoint VPN) –** A Cisco-developed solution that allows routers to dynamically create a full mesh of IPSec tunnels without having to manually configure every single tunnel.

**Remote Access VPNs –** Used to allow end devices to access a company’s internal resources securely over the internet. **Typically uses TLS** (Transport Layer Security), formerly known as SSL. **VPN client software is installed on end devices**, these end devices then form secure tunnels to one of the company’s routers/firewalls acting as a TLS server. This allows end users to securely access resources on a company’s internal network without being directly connected to the company network.

**Site-to-Site vs Remote-Access VPN:**

* **Site-to-Site** VPNS typically use IPSec. **|** **Remote-Access** VPNs typically use TLS.
* **Site-to-Site** VPNs provide service to many devices within the sites they’re connecting.
* **Remote-Access** VPNs provide service to the one end device the VPN client software is installed on.
* **Site-to-Site** VPNs are typically used to permanently connect 2 sites over the internet.
* **Remote-Access** VPNs are typically used to provide on-demand access for end devices that want to securely access company resources while connected to a network which is not secure.

**802.11k =** IEEE standard that provides assisted roaming in a wireless network. A **roaming wireless client can request** a list of known neighboring APs that might be potential candidates for transition. **The returned list by an AP** is optimized for the client and contains only APs in the same wireless band that the client is associated with. This **enables the client to** find a new AP without needing to probe.

**------------------------------------------------------------------**

***Section 55 – Virtualization & Cloud (1.2.f, 1.12):***

**5 Essential Cloud Characteristics:**

* **On-Demand Self-Service =** Customer can use/stop a service freely without direct communication to a service provider.
* **Broad Network Access =** Service is available through standard network connections and can be accessed through many kinds of devices.
* **Resource Pooling =** A pool of resources is provided by a service provider, and a customer can request a service, the resources to fulfill that request are allocated from the shared pool.
* **Rapid Elasticity =** Customers can quickly expand the services they use in the cloud from a pool of resources that appear infinite. Can also quickly reduce their services when not needed.
* **Measured Service =** Cloud provider measures a customer’s usage of cloud resources and charges them accordingly; the customer can measure their own use as well.

**3 Service Models of Cloud –** In cloud computing, everything is provided on a ‘service’ model. 3 varieties of service:

* SaaS (Software as a Service)
* PaaS (Platform as a Service)
* IaaS (Infrastructure as a Service)

**4 Deployment Models of Cloud:**

* Private Cloud
* Community Cloud
* Public Cloud
* Hybrid Cloud

**--------------------------------------**

***Virtualization & Cloud:***

**Containers –** Software packages that contain an App and all its dependencies (bins/libs) for the contained App to run (can run multiple apps in a single container but usually only one). Containers run on a **Container Engine** (Docker); The container engine is run on a host OS. Containers are lightweight and include only the dependencies required to run the specific App.

**Container Orchestrator –** Software platform for automating the deployment, management, scaling, etc. of containers (Kubernetes is the most popular). In small numbers, manual operation is possible but **large-scale systems can require thousands of containers**.

**VMs vs Containers:**

* **VMs** can take minutes to boot up. **|** **Containers** can boot up in milliseconds.
* **VMs** take up more disk space (**gigabytes**), **| Containers** take up very little disk space (**megabytes**).
* **VMs** use more CPU/RAM (**each VM** has **own OS**). **|** **Containers** use much fewer resources (**shared OS**).
* **VMs** are portable and can move between physical systems running the same hypervisor. **Containers** are **more** portable; smaller, faster to boot, and Docker containers can be run on nearly any container service.
* **VMs** are more isolated because each VM runs its own OS. **| Containers** are less isolated because they all run on the same OS, if the OS crashes, all containers running are affected.

**VRF (Virtual Routing & Forwarding) –** Used to divide a single router into multiple virtual routers, like VLANs are used to divide one switch (LAN) into multiple virtual switches (VLANs). It does this by allowing a router to build multiple separate routing tables. Interfaces (**Layer 3 only**) and routes are configured to be in a specific VRF. Router interfaces, SVIs, and routed ports on multilayer switches can be configured in a VRF.

**VRF Leaking =** Usually, **traffic in one VRF** can’t be forwarded out of an interface in another VRF. V**RF leaking** can be configured to allow traffic to pass between VRFs.

**VRF-lite =** VRF is commonly used to facilitate MPLS. However. VRF-lite is VRF without MPLS.

**VRF** is **commonly used by service providers to** allow one device to carry traffic from multiple customers:

* **Each customer’s traffic** is isolated from the others.
* **Customer IPs** can overlap without issues.

**--------------------------------------------------------------------------**

***Section 56 – Wireless Fundamentals (1.1.d, 1.11.a-c):***

**Wi-Fi networks act like a hub** when transmitting frames, all devices within range receive all frames.

**CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) –** Used to facilitate half-duplex communications. **When CSMA/CA is used**, a device will wait for other devices to stop transmitting before it transmits data itself.

* **Absorption =** Wireless signal passes through material and is converted into heat, weakening original signal.
* **Refraction =** When a wave is bent while entering a medium where the signal travels at a different speed (**glass, water**).
* **Reflection =** Signal bounces off material (**like metal**).
* **Diffraction =** Wave encounters an obstacle and travels around, **results in** “blind spot”.
* **Scattering =** Material causes a signal to scatter in all directions (**dust, smog, uneven surfaces**).

**RF (Radio Frequency) –** **To send wireless signals**, the **sender applies an alternating current to an antenna**, this creates electromagnetic fields which propagate out as waves. Electromagnetic waves can be measured in multiple ways**:**

* **Amplitude =** Max strength of the electric and magnetic fields. **Best described as the** height of a wave from trough to crest.
* **Frequency =** Measures the number of up/down cycles per a given unit of time. **Most common measurement** of frequency is hertz (Hz, kHz, MHz, GHz, THz).
* **Period =** The amount of time of one cycle. **EX:** If Frequency = 4 Hz, Period = 0.25 seconds

**Radio Frequency Bands = Wi-Fi uses 2** main bands (**frequency ranges**)**:**

* **2.4 GHz =** Typically provides further reach in open space and better penetration of obstacles such as walls.
* **5 GHz =** Less problems dealing with interference since more devices tend to use the 2.4 GHz band.

**Channels –** Each frequency band is divided into multiple channels. Devices are configured to transmit/receive traffic on one (or more) of these channels. **Larger WLANs with multiple APs**, it’s important that adjacent APs don’t use overlapping channels.

* **2.4 GHz = Channels 1, 6, 11** are recommended. **Using these channels**, you can place APs in a ‘honeycomb’ pattern to provide complete coverage of an area without interference between channels.
* **5 GHz =** Consists of non-overlapping channels, so it’s much easier to avoid APs interference.

**802.11 Standards:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Standard** | **Frequencies** | **Max Data Rate (Theoretical)** | **Alternate Name** |
| 802.11 | 2.4 GHz | 2 Mbps |  |
| 802.11b | 2.4 GHz | 11 Mbps |  |
| 802.11a | 5 GHz | 54 Mbps |  |
| 802.11g | 2.4 GHz | 54 Mbps |  |
| 802.11n | 2.4/5 GHz | 600 Mbps | ‘Wi-Fi 4’ |
| 802.11ac | 5 GHz | 6.93 Gbps | ‘Wi-Fi 5’ |
| 802.11ax | 2.4/5/6 GHz | 4\*802.11ac | ‘Wi-Fi 6’ |

**Service Sets –** Groups of wireless network devices. **3 main types:** Independent, Infrastructure, Mesh

All devices in a service set share the same **SSID (Service Set Identifier)**.

**IBSS (Independent Basic Service Set) –** A wireless network in which 2 or more wireless devices connect directly without using an AP. **Also called** an ad hoc network, **not scalable** beyond a few devices.

**BSS (Basic Service Set) –** Kind of infrastructure Service Set in which clients connect to each other via an AP, but not directly to each other. Wireless devices request to associate with the BSS, devices that have associated, are **called ‘clients’ or ‘stations’**.

**BSSID –** Used to uniquely identify an AP. Other APs can use the same SSID, but not the same BSSID. BSSID is the MAC address of an AP’s radio.

**BSA (Basic Service Area) –** Area around an AP where its signal is usable.

* **BSS =** Group of devices
* **BSA =** A physical area

**ESS (Extended Service Set) –** Used to create larger wireless LANs beyond the range of a single AP. **APs with** their **own BSS’s** are **connected by** a **wired network:**

* **Each BSS uses** the same SSID.
* **Each BSS has** a unique BSSID.
* **Each BSS uses** a different channel to avoid interference.
* **Roaming =** Clients can pass between APs without having to reconnect, providing a seamless Wi-Fi experience when moving between APs.
* **BSAs should overlap** about 10-15%.

**MBSS (Mesh Base Service Set) –** Can be used in situations where it’s difficult to run an Ethernet connection to every AP.

* **Mesh APs use 2 radios:** 1 to provide a BSS to wireless clients and 1 to form a ‘**backhaul network**’, used to bridge traffic from AP to AP.
* At least one AP is connected to the wired network, it’s called the **Root Access Point (RAP)**.
* The other APs are called **MAPs (Mesh Access Points)**. A protocol is used to determine the best path through the mesh.

**Distribution System –** Most **wireless networks aren’t standalone**, **rather they’re a way for wireless clients to connect to** a **wired network infrastructure**. In 802.11, the upstream wired network is the **DS**. Each BSS or ESS is mapped to a VLAN in the wired network.

* **An AP can provide multiple wireless LANs**, each with a unique SSID. Each WLAN is mapped to a separate VLAN and connected to the wired network via a trunk.

**AP Operational Modes:**

* **Repeater –** Can be used to extend the range of a BSS, a repeater will simply retransmit any signal it receives from an AP.
* A **repeater with** a **single radio =** Must operate on the same channel as an AP (**reduces overall throughput**).
* **Repeater with 2 radios =** Can receive on one channel and retransmit on another channel.
* **WGB (Workgroup Bridge) –** Operates as a wireless client of another AP and can be used to connect wired devices to a wireless network. 2 kinds of WGBs:
* **uWGB (Universal WGB) =** 802.11 standard that allows **one device** to be bridged to a wireless network.
* WGB = **Cisco-proprietary** version that allows **multiple wired clients** to be bridged to a wireless network.
* **Outdoor Bridge –** Can be used to connect networks over long distances without a physical cable connecting them. **APs use specialized antennas** that focus signal power in one direction, which allows the wireless connection to be made over longer distances than normal.

**----------------------------------------------------------------------**

***Section 57 - Wireless Architectures (1.1.d-e, 2.6-8):***

**802.11 Frame Format:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Frame Ctrl  *(2 bts)* | Duratn/ID  *(2 bts)* | Addr.1  *(6 bts)* | Addr.2  *(6 bts)* | Addr.3  *(6 bts)* | Sqnce Ctrl  *(2 bts)* | Addr.4  *(6 bts)* | QoS Ctrl  *(2 bts)* | HT Ctrl  *(4 bts)* | Frame Body (Packt)  *(Variable*  *Size)* | FCS  *(4 bts)* |

* **Frame Control =** Provides info such as the message type and subtype. **Can be used to** indicate whether a frame is a management frame.
* **Duration/ID =** Depending on the message type, **this field can indicate:**
* Time (in microseconds) a channel will be dedicated for transmission of a frame.
* Identifier for the association (connection).
* **Addresses =** Up to 4 can be present in an 802.11 frame, depending on the message type:
* **Destination Address (DA) =** Final recipient of the frame.
* **Source Address (SA) =** Original sender of the frame.
* **Receiver Address (RA) =** Immediate recipient of the frame.
* **Transmitter Address (TA) =** Immediate sender of the frame.
* **Sequence Control =** Used to reassemble fragments and eliminate duplicate frames.
* **QoS Control =** Used in QoS to prioritize traffic.
* **HT (High Throughput) Control =** Added in 802.11an (AKA ‘High Throughout Wi-Fi’). 802.11ac is AKA ‘Very High Throughput’ Wi-Fi.
* **FCS (Frame Check Sequence) =** Same as in an Ethernet frame, used to check for errors.

**802.11 Association Process =** APs bridge traffic between wireless stations and other devices. **For a station to send traffic through an AP**, it must be associated with it. There are **three 802.11 connection states:**

* Not Authenticated, Not Associated
* Authenticated, Not Associated
* Authenticated and Associated

There’s **2 ways a station can scan for a BSS:**

* **Active Scanning =** Station sends probe requests and listens for a probe response from an AP.
* **Passive Scanning =** Station listens for beacon messages from an AP, beacon messages are sent periodically by APs to advertise the BSS.

There are **three 802.11 message types:**

* **Management =** Used to manage a BSS (Beacon, Probe, Authentication, Association).
* **Control =** Used to control access to the medium (radio frequency) and data frames (**RTS, CTS, ACK**).
* **Data =** Used to send actual data packets.

**3 Wireless AP Deployment Methods:** *(Autonomous, LW, Cloud-Based)*

* **Autonomous APs –** Self-contained systems that don’t rely on a WLC, configured individually by console cable (CLI), Telnet/SSH (CLI), or HTTP/HTTPS web connection (GUI). An IP for remote management should be configured. Security policies and QoS rules are configured individually on each AP. There’s no central monitoring or management of APs. Autonomous APs connect to the wired network with a trunk link. Can be used in small networks but not viable in larger networks.
* **Lightweight APs (Split-MAC Architecture) –** The functions of an AP can be split between the AP and a WLC. **Lightweight APs handle** “real-time” operations like RF traffic, encryption/decryption, sending out beacons/probes, etc. **Other functions (security management, client authentication/association requests, key exchange, etc.) are** carried out by a WLC. The WLC is also used to centrally configure the Lightweight APs.

The WLC and the Lightweight APs use a protocol called **CAPWAP** to communicate. **2 tunnels are created between each AP and the WLC:**

* **Control Tunnel (UDP Port 5246) –** Used to configure APs, control/manage the operations. All traffic in this tunnel is encrypted by default.
* **Data Tunnel (UDP Port 5247) –** All traffic from wireless clients is sent through this tunnel to the WLC, it **does not** go directly to the wired network. Traffic is not encrypted by default but can be configured with DTLS (**Datagram Transport Layer Security**).

**Because all traffic from wireless clients is tunneled to** the **WLC with CAPWAP**, APs connect to switch access ports, **not** trunk.

**Lightweight AP Operational Modes:**

* **Local =** Default mode where the AP offers a BSS (or multiple) for clients to associate with.
* **FlexConnect =** Also offers one or more BSSs for clients to associate with. However, allows an AP to locally switch traffic between the wired and wireless networks if the tunnels to the WLC go down.
* **Sniffer =** AP **doesn’t** offer a BSS for clients. It’s dedicated to capturing 802.11 frames and sending them to a device running software such as Wireshark.
* **Monitor =** AP **doesn’t** offer a BSS for clients. It’s dedicated to receiving 802.11 frames to detect rogue devices. **If a client is found to be** a **rogue device**, it can send de-authentication messages to disassociate them from their AP.
* **Rogue Detector =** AP doesn’t even use its radio. It listens to traffic on the wired network only, but it receives a list of suspended rogue clients and AP MACs from the WLC. **By listening to ARP messages on the wired network and cross-referencing** it **with info it receives from the WLC**, it can detect rogue devices.
* **SE-Connect (Spectrum Expert Connect) =** AP doesn’t offer a BSS for clients. It’s dedicated to RF spectrum analysis on all channels. It can send info to software such as Cisco Spectrum Expert on a PC to collect/analyze the data.
* **Bridge/Mesh =** Lightweight AP can be a dedicated bridge between sites, for example over long distances. A mesh can be made between the APs.
* **Flex Plus Bridge =** Adds FlexConnect functionality to the Bridge/Mesh mode. **Allows WAPs to locally forward traffic** even if connectivity to the WLC is lost.
* **Cloud-Based APs –** In **between Autonomous AP and Split-MAC architecture (in terms of function)**, Autonomous APs that are centrally managed in the cloud. **Cisco Meraki** is a popular cloud-based Wi-Fi solution. **Only management/control traffic** (AP configuration, generated performance reports, etc.) is sent to the cloud. **Data traffic is sent to** the wired network.

**In a Split-MAC Architecture, 4 Main WLC Deployments:**

* **Unified =** WLC is a hardware appliance in a central location of the network. **Supports ~** 6000 APs.
* **Cloud-Based =** WLC is a VM running on a server, usually in a private cloud in a data center. **Supports ~** 3000 APs.
* **Embedded =** WLC is integrated within a switch. **Supports ~** 200 APs.
* **Mobility Express =** WLC is integrated within an AP. **Supports ~** 100 APs.

**------------------------------------------------------------**

***Section 58 – Wireless Security (1.11.d, 5.9):***

**Encryption –** All devices on a WLAN will use the same protocol to encrypt traffic, **however each client will use** **a unique encryption/decryption key** so that other devices can’t read their traffic.

**Integrity –** **MIC (Message Integrity Check)** is added to messages to help protect their integrity. The AP and client calculate a MIC using the same protocol and compare to verify integrity.

**Authentication Methods:**

* **Open Authentication =** Client sends an authentication request, and the AP accepts it. **Not a secure authentication method**. After a client is authenticated and associated with an AP, it’s possible to require authentication via other methods before access to the network is granted.
* **WEP (Wired Equivalent Privacy) =** Used to provide both authentication/encryption of wireless traffic, encrypts using the RC4 algorithm. ‘**Shared-key**’ **protocol**, requiring sender/receiver to have the same key. **Not secure**, can be easily cracked.
* **EAP (Extensible Authentication Protocol) =** An authentication framework, defines a standard set of authentication functions used by various EAP Methods. **EAP is integrated with 802.1x**, which provides port-based network access control. **802.1x is used to** limit network access for clients connected to a LAN or WLAN until they authenticate. **3 main entities in 802.1x:**
* **Supplicant:** Device that wants to connect to a network.
* **Authenticator:** Device that provides access to a network.
* **Authentication Server (AS):** Device that receives client credentials and permits/denies access.

**4 EAP Methods:** *(LEAP, EAP-FAST, PEAP, EAP-TLS)*

* **LEAP (Lightweight EAP) =** Clients must provide a username/password to authenticate. **In addition,** mutual authentication is provided by both the client and server sending a challenge phrase to each other. **Dynamic WEP keys are used**, meaning that the WEP keys are changed frequently. Like WEP, LEAP is **considered vulnerable and should not be used anymore**.
* **EAP-FAST (EAP Flexible Authentication via Secure Tunneling) =** Consists of 3 phases:

1. **PAC Provisioning –** A PAC (Protected Access Credential) is generated and passed from the server to the client.
2. **Encrypted TLS Tunnel –** A secure TLS tunnel is established between the client and authentication server.
3. **Authentication –** Inside of the secure (encrypted) TLS tunnel, the client and server communicate further to authenticate/authorize the client.

* **PEAP (Protected EAP) =** Also involves a secure TLS tunnel between client and server, **instead of a PAC,** the server has a digital certificate. The client uses the certificate to authenticate the server, the cert is also used to establish the TLS tunnel.
* **EAP-TLS (EAP Transport Layer Security) =** Requires a certificate on the authentication server and on every single client. The **most secure wireless authentication method**, but it is more difficult to implement than PEAP because every client device needs a certificate. **TLS Tunnel isn’t needed for authentication** since client/server authenticate each other with the certs, TLS is still used to exchange encryption key info.

**Encryption and Integrity Methods:**

* **TKIP (Temporal Key Integrity Protocol) =** WEP was found to be vulnerable, but wireless hardware at the time was built to use it so a temporary solution (TKIP) was needed until a new standard was created and new hardware was built. **Used in WPA version 1**.
* **CCMP (Counter/CBC-MAC Protocol) =** Developed after TKIP and is more secure, **used in WPA2**. CCMP must be supported by a device’s hardware to be used, older hardware usually cannot use CCMP. Consists of 2 different algorithms to provide encryption and MIC**:**

1. **AES (Advanced Encryption Standard)** **counter mode** encryption – Most secure encryption protocol currently, multiple modes of operation for AES, CCMP uses counter mode.
2. **CBC-MAC (Cipher Block Chaining Message Authentication Code) –** Used as a MIC to ensure integrity of messages.

* **GCMP (Galois/Counter Mode Protocol) =** More secure/efficient than CCMP, its increased efficiency allows higher data throughput than CCMP. **Used in WPA3**. Consists of 2 algorithms:

1. **AES** **counter mode** encryption.
2. **GMAC** **(Galois Message Authentication Code) –** Used as a MIC.

Wi-Fi Protected Access – The **Wi-Fi Alliance** has **developed 3 WPA certs for wireless devices =** WPA, WPA2, WPA3. **All support 2 authentication modes:**

* **Personal Mode =** A pre-shared key (**PSK**) is used for authentication. Connect to a network, enter password for authentication, that’s Personal Mode. Common in small networks. **PSK itself isn’t sent over the air**. A 4-way handshake is used for authentication and the PSK is used to generate encryption keys.
* **Enterprise Mode =** 802.1x is used with an authentication server (**RADIUS**). No specific EAP method is specified, so all are supported.

**WPA –** Certification was developed after WEP was found to be vulnerable. Includes TKIP for encryption/MIC and 802.1x authentication (**Enterprise** Mode) or PSK (**Personal** Mode).

**WPA2 –** Includes the protocols CCMP for encryption/MIC and 802.1x authentication (Enterprise Mode) or PSK (Personal Mode).

**WPA3 –** Includes GCMP for encryption/MIC and 802.1x authentication (Enterprise Mode) or PSK (Personal Mode). Provides **additional security features:**

* **PMF (Protected Management Frames) =** Protects 802.11 management frames from eavesdropping/forging.
* **SAE (Simultaneous Authentication of Equals) =** Protects the 4-way handshake when using Personal Mode authentication.
* **Forward Secrecy =** Prevents data from being decrypted after it’s been transmitted, so an attacker can’t capture wireless frames and try to decrypt them later.

**----------------------------------------------------------------------**

***Section 59 – Wireless Configuration (2.7-2.9, 5.10):***

**WLC –** Can connect to switches via a LAG (**Link Aggregation Group) aka EtherChannel**. WLCs only support static LAG, no PAgP or LACP.

* **(dhcp-config)#**option 43 ip *ip-address* **=** Can be used to tell APs the IP of their WLC.

**WLC Ports –** The **physical** ports that cables connect to.

**WLC Interfaces –** The **logical** interfaces within the WLC (**EX:** SVIs on a switch).

**WLC Ports:**

* **Service Port =** Dedicated management port used for out-of-band management. Must connect to a switch access port since it **supports only 1 VLAN**. This **port can be used to connect to a device while** its booting, performing system recovery, etc.
* **Distribution System Port =** Standard network ports that connect to **DS** (wired network) and are used for data traffic. Usually connect to trunk ports, and **if multiple distribution ports are used**, they can form a LAG.
* **Console Port =** Standard console port (**RJ45 or USB**).
* **Redundancy Port =** Used to connect to another WLC to form a high availability (**HA**) pair.

**WLC Interfaces:**

* **Management Interface =** Used for management traffic (Telnet, SSH, HTTP(S), RADIUS, NTP, Syslog, etc.) CAPWAP tunnels are formed to/from the WLC’s management interface.
* **Redundancy Management Interface =** **When** **2 WCLs are connected by their redundancy ports,** one WLC is ‘**active**’ and the other is ‘**standby**’. This interface can be used to connect to/manage the ‘standby’ WLC.
* **Virtual Interface =** Used when communicating with wireless clients to **relay DHCP requests, perform client web authentication, etc.**
* **Service Port Interface =** If the service port is used, this interface is bound to it and used for out-of-band management.
* **Dynamic Interface =** User-defined and typically used for client data. The interface used to map a WLAN to a VLAN (dynamic interface can segment traffic on a WLC, **like** a **VLAN** does **on a switch**). For example, traffic from the ‘Internal’ WLAN will be sent to the wired network from the WLC’s ‘Internal’ dynamic interface.

**WLC Configuration (Layer 3 Security):**

* **Web Authentication =** **After** a **wireless client gets an IP and tries to access a web page**, they’ll have to enter a username/password to authenticate.
* **Web Passthrough =** Like the above but **no** username/password are required. A warning statement is displayed, and the client simply must agree to gain access to the internet.
* There are **Conditional** and **Splash Page** web redirect options which are like the above, but additionally require 802.1x layer 2 authentication.

**WLC QOS Settings:**

* **Platinum =** Voice
* **Gold =** Video
* **Silver =** Best Effort
* **Bronze =** Background

**----------------------------------------------------------**

***Section 60 – Network Automation (6.1-3):***

**Logical “Planes” –** Various functions of network devices can be logically divided (**categorized**) into planes:

* **Data Plane (Forwarding Plane) =** Contains all tasks involved in forwarding user data/traffic from one interface to another. **Routers receive messages**, look for the most specific matching route in the routing table, and forward it out of the appropriate interface. A **switch receives a message**, looks at the destination MAC, and forwards it (or floods it) out of the appropriate interface. **NAT** is part of the Data Plane. **Deciding to forward/discard messages due to ACLs**, **port security, etc.** is also part of the Data Plane.
* **Control Plane =** Functions that build tables (**Routing, MAC, ARP tables, STP, etc.**), and other functions that influence the Data Plane, are part of the Control Plane. This plane controls what the Data Plane does by performing **overhead work**. **EX:** OSPF itself doesn’t forward user data packets, but it informs the Data Plane about how packets should be forwarded. Same concept with STP and ARP.

*(****In traditional networking****, the Data and Control planes are both distributed. Each device has its own Data Plane and its own Control Plane. The planes are “distributed” throughout the network.)*

* **Management Plane =** Also performs overhead work but doesn’t directly affect the forwarding of messages in the Data Plane. This plane consists of protocols that are used to manage devices: **SSH/Telnet, Syslog, SNMP, NTP, etc.**

The **operations of the Management/Control Planes are usually managed by the CPU**; However, this is not desirable for the Data Plane. Instead, a specialized hardware ASIC (**App-Specific Integrated Circuit**) is used. **ASICs are** chips built for specific purposes. **Using a switch as an example:**

1. **When a frame is received**, the ASIC (**not CPU**) is responsible for the switching logic.
2. The MAC table is stored in a kind of memory called TCAM (**Ternary Content-Addressable Memory**). A common name for the MAC address table is CAM table.
3. **The ASIC feeds the destination MAC of the frame into the TCAM**, which returns the matching MAC address table entry.
4. The frame is then forwarded out of the appropriate interface.

**SDN (Software-Defined Networking) –** An approach to networking that centralizes the control plane into an app called a controller. **SDN is aka** ‘SDA (Software-Defined Architecture)’ **or** ‘Controller-Based Networking’. **Traditional control planes use** a **distributed architecture**, each router in the network might run OSPF and the routers share routing info to calculate preferred routes to each destination. **An SDN controller** centralizes control plane functions like calculating routes. The controller can interact programmatically with network devices using **APIs**.

**SBI (Southbound Interface) –** Used for communications between the controller and the network devices it controls. Typically consists of a communication protocol and API. APIs facilitate data exchanges between programs. **An API on the network devices** allows the controller to access info on the devices, control their data plane tables, etc. Some **examples of southbound APIs:** OpenFlow, Cisco OpFlex, Cisco onePK, NETCONF.

**NBI (Northbound Interface) –** **Allows us to interact with the controller**, access the data it gathers about a network, program it, and make changes in the network via the SBI.

A **REST API** is used on the controller as an interface for apps to interact with it. Data is sent in a structured format (**JSON** or **XML**). This makes it easier for programs to use data. Some **examples of northbound APIs:** OSGi

**Automation in Traditional Networks vs SDN:**

* Networking tasks can be automated **in traditional networks =** Scripts can be written to push commands to many devices at once. **Python with good use of** ‘**regex**’ (Regular Expressions) can parse through ‘show’ commands to gather info about network devices.
* **However**, the **robust and centralized data collected by SDN controllers greatly facilitates these functions =** The controller collects info about all devices in a network. **Northbound APIs** allow apps to access info in a format that is easy for programs to understand (JSON/XML). **The centralized data** facilitates network wide analytics.

***Although SDN and automation aren’t the same*** *thing, the SDN architecture greatly facilitates the automation of various tasks in a network via the SDN controller and APIs.*

**------------------------------------------------------**

***Section 61 – JSON, XML, & YAML (6.7):***

**Data Serialization –** The process of converting data into a standardized format that can be stored (**in a file**) or transmitted (**over a network**) and reconstructed later (i.e., by a different app). **Data serialization languages** allow us to represent variables with text.

**JSON (JavaScript Object Notation) –** An open standard file format and data interchange format that uses human-readable text to store and transmit data objects. REST APIs often use JSON. **Whitespace** (**blank space**) is insignificant. JSON **can represent 4 ‘primitive’ data types:**

* **String =** A text value (**surrounded by double quotes**).
* **Number =** Numeric value not surrounded by quotes.
* **Boolean =** A data type that has only 2 possible values (**true or false**), not surrounded by quotes.
* **Null =** Value represents the intentional absences of any object value, not surrounded by quotes.

JSON also has **2 ‘structured’ data types:**

* **Object** (‘**Dictionary**’) **=** Unordered list of key-value pairs (**variables**). Surrounded by curly brackets **{}**. **The key** is a string. **The value is** any valid JSON data type. The **key and value are separated by** a colon. **If** there’s **multiple key value pairs**, each pair is separated by a comma. Objects within objects are called “**nested objects**”.
* **Array =** A series of values (not key-value pairs) separated by commas and surrounded by square brackets **[]**; the values don’t have to be the same data type.

**XML (Extensible Markup Language) –** Was developed as a markup language (used to format HTML text) but is now used as a general data serialization language. Often used by REST APIs but is less human-readable than JSON. **Whitespace** is insignificant. **Key & values are written as:** *<key>value</key>*

**YAML (YAML Ain’t Markup Language) –** Used by the network automation tool Ansible. **Whitespace** is significant, **indentation is** very important. **YAML files start with** ---. (**-**) is used to indicate a list. **Key & values are represented as:** *key:value*

**----------------------------------------**

***Section 62 – REST APIs (6.5):***

**API (Application Programming Interface) –** Software interface that allows 2 apps to communicate with each other. **In SDN architecture**, APIs are used to communicate between apps and the SDN controller (**via the NBI**), and between the SDN controller and the network devices (**via the SBI**).

**CRUD (Create, Read, Update, Delete) –** Refers to the operations performed using REST APIs:

* **Create =** Operations are used to create new variables and set their initial values.
* **Read =** Operations are used to retrieve the value of a variable.
* **Update =** Operations are used to change the value of a variable.
* **Delete =** Operations are used to delete variables.

**HTTP uses verbs (aka methods)** that map to these CRUD operations**:**

|  |  |  |
| --- | --- | --- |
| ***Purpose*** | ***CRUD Operation*** | ***HTTP Verb*** |
| Create new variable | **Create** | POST |
| Retrieve value of variable | **Read** | GET |
| Change the value of variable | **Update** | PUT, PATCH |
| Delete variable | **Delete** | DELETE |

**HTTP Request –** When an HTTP client sends a request to an HTTP server, the **HTTP header includes info like this:** An HTTP Verb (**EX:** GET). A URI (**Uniform Resource Identifier**) indicating the resource it’s trying to access. HTTP request can include additional headers which pass additional info to the server.

**When a REST client makes an API call (request) to a REST server**, it will send an HTTP request. REST APIs don’t have to use HTTP for communication, although HTTP is the most common choice.

**HTTP Response –** The server’s response will include a status code indicating if the request succeeded or failed and other details (**EX:** 404). **The first digit indicates** the **class of the response:**

* **1xx (Informational) =** Request was received, continuing process
* **2xx (Successful) =** Request was successfully received, understood, and accepted
* **3xx (Redirection) =** Further action needed to complete the request
* **4xx (Client Error) =** Request contains bad syntax or cannot be fulfilled
* **5xx (Server Error) =** The server failed to fulfill an apparently valid request

**REST (Representational State Transfer) APIs –** Isn’t a specific API, instead it describes a set of rules about how the API should work. There are **6 constraints of RESTful architecture:**

* Uniform Interface
* Client-server
* Stateless
* Cacheable or non-cacheable
* Layered System
* Code-on-demand (optional)

**REST: Client-Server =** A client uses API calls (**HTTP Requests**) to access the resources on a server. The separation between the client and server means they both can change and evolve independently of each other.

**REST: Stateless =** This means that each API exchange is a separate event, independent of all past exchanges between a client and server (**server doesn’t store info about previous requests to determine how it should respond** to new requests). **If authentication is required**, this means that the client must authenticate with the server for each request it makes.

**REST: Cacheable or Non-Cacheable =** REST APIs must support caching of data. Caching refers to storing data for future use. Not all resources have to be cacheable, but any cacheable resources MUST be declared as cacheable.

**--------------------------------------**

***Section 63 – SDN (6.3-6.4):***

**SDN Architecture:**

|  |
| --- |
| **Application Layer** |
| Contains scripts/apps that tell the SDN controller what network behaviors are desired. |
| **Control Layer (REST API, Controller, Controller Plane)** |
| Contains the SDN controller that receives and processes instructions from the App Layer. |
| **Infrastructure Layer (Data Plane)** |
| Contains the network devices that are responsible for forwarding messages across the network. |

**Cisco SD Access –** Cisco’s SDN solution for automating campus LANs.

**ACI (Application Centric Infrastructure) –** Cisco’s SDN solution for automating data center networks.

**Cisco DNA (Digital Network Architecture) Center –** The controller at the center of SD-Access. **Can be a network manager** in a traditional network. Is an app installed on Cisco UCS server hardware. It has a REST API which can be used to interact with DNA center. **DNA Center** enables Intent-Based Networking (**IBN**), this allows for the communication of intent of network behavior to the DNA Center, which will take care of the details of the actual configurations and policies on devices.

**Underlay –** The underlying **physical** network of devices and connections (wired and wireless) which provide IP connectivity (**EX:** Multilayer switches and their connections). **Underlay’s purpose is** to support the VXLAN tunnels of the overlay. There are **3 different roles for switches in SD-Access:**

* **Edge Nodes =** Connect to end hosts
* **Border Nodes =** Connect to devices outside of the SD-Access domain
* **Control Nodes =** Use LISP (**Locator ID Separation Protocol**) to perform various control plane functions

**Brownfield Deployment =** You can add SD-Access on top of an existing network if network hardware/software supports it. In this case DNA Center won’t configure the underlay.

**Greenfield Deployment =** A new deployment configured by DNA Center to use the **optimal SD-Access underlay:**

* **All switches** are Layer 3 and use IS-IS as their routing protocol.
* **All links between switches** are routed/Layer3 (No STP needed).
* **Edge nodes** (**access switches**) act as the default gateway of end hosts (routed access layer).

**Overlay –** The **virtual** network built on top of the physical underlay network. **LISP provides** the **control plane of SD-Access:**

* A list of mappings of EIDs (**endpoint identifiers**) to RLOCs (**routing locators**) is kept.
* **EIDs identify** end hosts connected to edge switches, and **RLOCs identify** the edge switch which can be used to reach the end host.
* **Cisco TrustSec** (**CTS**) **=** Provides policy control (**QoS, security policy, etc.**)
* **VXLAN =** Provides the data plane of SD-Access by creating virtual tunnels.

**Fabric –** The combination of the overlay and underlay; the physical and virtual network as a whole.

**DNA Center vs. Traditional Network Management:**

**Traditional Network Management =**

* Devices are configured one-by-one via SSH or console connection.
* Configurations and policies are managed per-device (**distributed**).
* New network deployments can take a long time due to the manual labor required and errors/failures are more likely.

**DNA Center-Based Network Management =**

* Devices are centrally managed/monitored from the DNA Center GUI or other apps using its REST API.
* **Admins communicate their intended network behavior to DNA Center**, which changes those intentions into configurations on the managed network devices.
* Configurations, policies, software versions are centrally managed. DNA Center can monitor cloud servers for new versions and update managed devices.
* New network deployments are much quicker, new devices can automatically receive their configurations from DNA Center.

**--------------------------------------------------------**

***Section 64 – Ansible, Puppet, Chef (6.6):***

**Configuration Drift =** When individual changes made over time cause a device’s configuration to deviate from the standard/correct configurations as defined by the company.

**Configuration Provisioning =** Refers to how configuration changes are applied to devices. Traditionally, configuration provisioning is done by connecting to devices one-by-one via SSH. **Configuration management tools** like **Ansible, Puppet, and Chef** allow us to make changes to devices on a mass scale with a fraction of the time/effort. **2 essential components:** templates and variables.

**Configuration Management Tools =** Network automation tools that facilitate the centralized control of large numbers of network devices. Some options are: Ansible, Puppet, and Chef. These tools can be used to perform tasks such as generate configurations for new devices on a large scale or perform config changes on devices (all or certain subsets). Check configs for compliance and compare configs between devices.

**Ansible –** Configuration management tool written in Python. Is agentless, it doesn’t require any special software to run on managed devices. Uses SSH to connect to devices, make config changes, extract info, etc. Uses a push model, the **Ansible server** (**Control Node**) uses SSH to connect to managed devices and push config changes to them. **After installing Ansible you must create several text files:**

* **Playbooks =** These files are “**blueprints of automation tasks**” they outline the logic and actions of the tasks that Ansible should do, written in YAML.
* **Inventory =** List the devices that will be managed as well as characteristics of each device such as their device role (access switch, core switch, WAN router, etc.) Written in INI, YAML, or other formats.
* **Templates =** Represent a device’s config file, but specific values for variables are not provided. Written in Jinja2 format.
* **Variables =** List variables and their values. These values are substituted into the templates to create complete config files. Written in YAML.

**Puppet –** Configuration management tool written in Ruby. Is typically agent-based, software must be installed on the managed devices. It **can be run agentless**, in which a proxy agent runs on an external host, and the proxy agent uses SSH to connect to the managed devices and communicate with them. The Puppet server is called the “**Puppet Master**”. Uses a pull model (**clients “pull” configs from the Puppet master**). **Clients use** TCP port 8140 to communicate with the Puppet master. Uses a proprietary language for files. **Text files required on the Puppet master include:**

* **Manifest =** File defines the desired config state of a network device.
* **Templates =** Similar to Ansible templates, used to generate Manifests.

**Chef –** (Least popular) Configuration management tool written in Ruby. Agent-based, requires specific software installed on managed devices. Uses a pull model. The server uses **TCP port 10,002** to send configs to clients. Files use a DSL (**Domain-Specific Language**) based on Ruby. **Text files used by Chef:**

* **Resources =** ‘Ingredients’ in a recipe. Config objects managed by Chef.
* **Recipes =** ‘Recipes’ in a cookbook. Outline the logic and actions of the tasks performed on resources.
* **Cookbooks =** A set of related recipes grouped together.
* **Run-List =** An ordered list of recipes that are run to bring a device to the desired config state.

**Ansible, Puppet, Chef Comparison:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Ansible** | **Puppet** | **Chef** |
| **Language** | Python | Ruby | Ruby |
| **Key Files Defining Actions** | Playbook | Manifest | Recipe, Run-List |
| **Communication Protocol** | SSH | HTTPS (via REST API) | HTTPS (via REST API) |
| **Key Port** | 22 | 8,140 | 10,002 |
| **Agent-Based/Agentless** | Agentless | Agent-Based (OR Agentless) | Agent-Based |
| **Push/Pull** | Push | Pull | Pull |