ENCOR Study Guide

Alexander

September 19, 2024

OCG

Layer 2

VLANs are defined in the IEEE 802.1Q standard, which states that 32 bits are added to the packet header in between the Source MAC and the EtherType/Size fileds. 16 bits for the Tag protocol identifier (TPID) and 16 for the Tag control information (TCI). The TPID is a 16-bit field set to a value of 0x8100 in order to identify the frame as an IEEE 802.1Q-tagged frame. This field is located at the same position as the EtherType field in untagged frames, and is thus used to distinguish the frame from untagged frames. The TCI can be broken into the following subfields: Priority code point (PCP) which is 3 bits and refers to 802.1p CoS and maps to the frame priority level, Drop eligible indicator (DEI) which is a 1 bit and may be used in conjuction or separately from PCP to indicate if the frame is eligible to be dropped in the presence of congestion, VLAN identifier (VID) is a 12 bit field specifying the VLAN to which the frame belongs.

The VLAN identifier has only 12 bits, which provide 4094 unique VLANs. Catalyst switches use the following logic for VLAN identifiers:

- VLAN 0 is reserved for 802.1p traffic and cannot be modified or deleted.
- VLAN 1 is the default VLAN and cannot be modified or deleted.
- VLANs 2 to 1001 are in the normal VLAN range and can be added, deleted, or modified as necessary.
- VLANs 1002 to 1005 are reserved and cannot be deleted.
- VLANs 1006 to 4094 are in the extended VLAN range and can be added, deleted, or modified as necessary.

A routed subinterface is required when there are multiple VLANs on a switch that require routing, and it is not desirable to use a dedicated physical routed interface per VLAN, or there are not enough physical router interfaces to accommodate all the VLANs. To overcome this issue, it is possible to create a trunk port on the switch and create a logical subinterface on the router. A subinterface is created by appending a period and a numeric value after the period. Then the VLAN needs to be associated with the subinterface with the command encapsulation dot1q vlan-id.

With Catalyst switches, it is possible to assign an IP address to a switched virtual interface (SVI), also known as a VLAN interface. An SVI is configured by defining the VLAN on the switch and then defining the VLAN interface with the command interface vlan vlan-id.The switch must have an interface associated to that VLAN in an up state for the SVI to be in an up state. If the switch is a multilayer switch, the SVIs can be used for routing packets between VLANs without the need of an external router.

Some network designs include a point-to-point link between switches for routing. For example, when a switch needs to connect to a router, some network engineers would build out a transitVLAN (for example, VLAN 2001), associate the port connecting to the router to VLAN 2001 could exist elsewhere in the Layer 2 realm, or that a spanning tree could impact the topology. Instead, the multilater switch port can be converted from a Layer 2 switch port to a routed switch port with the interface configuration command no switchport. Then the IP address can be assigned to it.

Access Port: An access port is a type of switch port that is configured to carry traffic for only one VLAN (Virtual Local Area Network). It does not tag the Ethernet frames with VLAN identifiers and is used to connect end devices such as computers and printers to the network.

ARP (Address Resolution Protocol): ARP is a network protocol used to map an IP address (Layer 3) to a MAC address (Layer 2) within a local network. When a device needs to communicate with another device on the same local network and knows its IP address, ARP is used to discover the corresponding MAC address to facilitate frame delivery.

Broadcast Domain: A broadcast domain is a logical segment of a network in which any broadcast packet sent by a device is received by all other devices within the same domain. Broadcast domains are typically bounded by routers or layer 3 devices, as these devices prevent broadcast packets from crossing into other segments of the network.

CEF (Cisco Express Forwarding): CEF is a high-performance, Layer 3 packet switching technique used in Cisco routers. It improves packet forwarding efficiency by maintaining a Forwarding Information Base (FIB) and an adjacency table to quickly route packets through the network without requiring extensive routing table lookups.

Collision Domain: A collision domain is a network segment where data packets can collide with one another when being sent over the same network medium. In Ethernet networks, collision domains are typically bounded by switches or bridges that separate segments to reduce packet collisions.

CAM (Content Addressable Memory): CAM is a type of memory used in network switches to store MAC addresses and their associated ports. It allows for rapid lookup of MAC addresses when forwarding frames by comparing the incoming MAC address against stored entries to determine the appropriate outgoing port.

Layer 2 Forwarding: Layer 2 forwarding refers to the process of switching Ethernet frames based on MAC addresses at the Data Link layer (Layer 2) of the OSI model. The switch uses the MAC address table to forward frames to the appropriate port.

Layer 3 Forwarding: Layer 3 forwarding involves routing packets based on IP addresses at the Network layer (Layer 3) of the OSI model. Routers use routing tables and algorithms to determine the best path for forwarding packets across different network segments.

FIB (Forwarding Information Base): The FIB is a data structure used by routers and switches to store routing or forwarding information for quick lookup and packet forwarding decisions. It is used by CEF in Cisco devices to determine the outgoing interface for packets based on their destination IP addresses.

MAC Address Table: The MAC address table, also known as the forwarding table or content addressable memory (CAM) table, is used by network switches to map MAC addresses to specific switch ports. This table enables efficient frame forwarding by directing traffic only to the port associated with the destination MAC address.

Native VLAN: The native VLAN is a VLAN that is associated with an 802.1Q trunk port where untagged frames are sent. The native VLAN's purpose is to ensure that untagged traffic is correctly associated with a VLAN, preventing VLAN tag miscommunication on trunk links.

Process Switching: Process switching is a method of packet forwarding in which the CPU of a router processes each packet individually. This method involves a high degree of CPU intervention, which can result in slower forwarding performance compared to hardware-based methods.

RIB (Routing Information Base): The RIB is a data structure used by routers to store routing information learned from various routing protocols. It contains details about network routes and their metrics, which are used by the router to make forwarding decisions.

Trunk Port: A trunk port is a type of switch port that can carry traffic for multiple VLANs by tagging frames with VLAN identifiers. It is used to connect switches or other networking devices to maintain VLAN information across network segments.

TCAM (Ternary Content Addressable Memory): TCAM is a specialized type of memory used in network devices to perform high-speed lookups of IP addresses and other data. Unlike traditional CAM, TCAM supports three states (0, 1, and "don't care"), enabling more complex matching operations used in features like access control lists (ACLs) and routing tables.

VLAN (Virtual Local Area Network): A VLAN is a logical network segment that groups together devices on different physical LANs into a single broadcast domain. VLANs are used to segment network traffic for security, performance, and organizational purposes, isolating traffic between different groups of devices even if they share the same physical network infrastructure.

Forwarding architectures refer to the methods and systems that network devices (like routers and switches) use to process and forward packets through a network. The main types of forwarding architectures: store-and-forward, cut-through, fragment-free, Virtual Output Queuing (VOQ), Application-Specific Integrated Circuits (ASICs), network processors, Software-Defined Networking (SDN). Store-and-forward is a packet-switching method in which a network device receives the entire packet before forwarding it to the next hop. This approach allows the device to perform error checking on the received packet, typically using Cyclic Redundancy Check (CRC) mechanisms to ensure data integrity. By buffering the entire packet, the device can also handle packets of varying sizes and manage them more effectively. While this method enhances reliability and can reduce the chances of forwarding corrupted packets, it introduces latency, as the device must wait until the complete packet is received before any forwarding action is taken. This delay can impact overall network performance, especially in high-throughput environments where speed is critical. Cut-through switching is a method that allows a network device to begin forwarding a packet as soon as it reads the destination address, without waiting for the entire packet to be received. This technique significantly reduces latency, making it faster than the store-and-forward method. As soon as the switch identifies the destination port, it starts transmitting the packet towards its destination, which is particularly beneficial in scenarios where low latency is paramount, such as in real-time applications. However, cut-through switching lacks error checking until the entire packet has been transmitted, which can lead to forwarding of corrupted packets if issues arise during transmission. Consequently, while cut-through offers speed advantages, it does so at the potential cost of data integrity. Fragment-free switching represents a compromise between the store-and-forward and cut-through methods. In this approach, the switch reads the first 64 bytes of a packet, which typically contain the header information and any collision data. If the initial segment appears valid, the switch begins forwarding the packet, thus avoiding the transmission of fragmented packets caused by collisions. This method reduces the likelihood of forwarding corrupted packets while still allowing for lower latency than pure store-and-forward systems. While fragment-free switching enhances performance in environments with high collision rates, it does not offer the same level of reliability as full store-and-forward, since it only checks a portion of the packet before forwarding. Virtual Output Queuing (VOQ) is a sophisticated switching architecture that addresses the issue of head-of-line blocking in traditional queuing systems. In VOQ, each input port maintains a separate queue for each output port, allowing packets destined for different outputs to be processed concurrently. This architecture enables more efficient use of bandwidth and minimizes latency by allowing multiple flows to be forwarded without being impeded by packets queued at the front of the line. VOQ is particularly advantageous in high-capacity switches and routers, as it optimizes resource utilization and enhances throughput. However, implementing VOQ can be complex, requiring additional memory and management resources, making it more challenging to deploy in simpler network environments. Application-Specific Integrated Circuits (ASICs) are specialized hardware components designed for high-speed packet processing in networking devices. ASICs are optimized for specific tasks, such as packet forwarding, routing, or encryption, enabling them to operate with high efficiency and low latency. By utilizing parallel processing and dedicated pathways for data, ASICs can handle vast amounts of traffic simultaneously, making them ideal for high-performance switches and routers. Their customization allows for tailored functionality to meet the specific needs of network applications. However, the design and manufacturing of ASICs can be costly and time-consuming, resulting in reduced flexibility compared to general-purpose processors, as any changes to functionality typically require redesigning the hardware. Network processors are programmable chips designed to handle complex networking tasks, such as packet inspection, classification, and traffic management. Unlike ASICs, which are hardwired for specific functions, network processors provide flexibility, allowing for the implementation of various protocols and features through software. This programmability enables network operators to adapt to changing requirements and to perform deep packet inspection for enhanced security measures. While network processors can provide significant versatility and support advanced functionalities, they generally operate at slower speeds compared to ASICs due to their more complex architectures. As such, they are often used in environments where flexibility and advanced processing capabilities are prioritized over raw performance. Software-Defined Networking (SDN) is an innovative network architecture that decouples the control plane from the data plane, allowing centralized control over network resources. In SDN, a central controller manages the network, directing traffic flows and implementing policies across multiple devices, which can be configured and monitored programmatically. This separation enables greater flexibility, as network administrators can quickly adapt to changing demands, implement dynamic traffic management, and automate network operations through software applications. SDN also facilitates more efficient resource utilization and simplifies network management, as changes can be made from a single point of control. However, this centralized approach can introduce challenges such as increased latency for control messages and potential risks associated with a single point of failure, necessitating robust security measures and high availability strategies.

In a centralized forwarding architecture within a single network device, the control plane and data plane functions are tightly integrated, with the control plane responsible for making all forwarding decisions. The device contains a central routing processor that manages the routing table, handles protocol updates, and calculates the best paths for incoming packets. When a packet arrives, the ingress line card sends it to the routing processor, which analyzes the packet's header and consults the routing table to determine the appropriate output port. This decision is then communicated back to the line card for forwarding. While this architecture simplifies processing and allows for a cohesive decision-making process, it may introduce latency, as packets must traverse the routing processor before forwarding. Additionally, if the routing processor becomes overwhelmed or fails, packet forwarding can be severely disrupted, highlighting the importance of robust design and redundancy within the device.

In a distributed forwarding architecture within a single network device, the control and data plane functions are more decentralized, allowing individual components to operate independently. Each ingress line card is equipped with its own local forwarding engine and routing information, enabling it to make forwarding decisions without having to rely on a central processor. When a packet arrives, the line card can immediately consult its local forwarding table to determine the appropriate output port, allowing for faster processing and reduced latency. This independence enhances performance, especially under high traffic loads, as each line card can handle packets concurrently without bottlenecking at a central control point. However, this decentralized approach can lead to challenges in maintaining consistent routing information across multiple line cards, as updates must be propagated and synchronized among them. Overall, distributed forwarding within a single device promotes efficiency and speed, making it well-suited for handling high volumes of traffic.

Ingress line cards are responsible for receiving incoming packets from the network. They perform initial processing, such as buffering and determining the destination of the packets. In a centralized forwarding architecture the ingress line cards may send packets to a centralized control plane for processing. After receiving forwarding decisions from the central controller, they then pass the packets to the switch fabric. For distributed forwarding architectures the ingress line cards can make forwarding decisions locally based on routing tables stored on the line cards themselves, reducing the need to communicate with a central controller.

Egress line cards handle outgoing packets, sending them to the appropriate destination on the network. For centralized forwarding, egress cards like ingress cards depend on the central controller's instructions for the proper routing to packets. In a distributed forwarding architecture would have the egress line cards making independent forwarding decisions based on local tables, helping to expedite the forwarding process.

The role of the switches fabric is to facilitate communication between ingress and egress line cards. It transfers packets from the input ports to the output ports. In a centralized model, the switch fabric is often controlled by the central processor, which directs traffic based on the global forwarding decisions. For distributed forwarding the fabric operates in a way that allows for fast, direct communication between line cards without needing to reference a centralized control point for each packet.

Route Processor Engines (RPEs) are responsible for maintaining routing tables and making routing decisions. It runs the control plane functions. In centralized architectures, the RPE is often part of a central controller that oversees the entire network. It processes incoming route information and disseminates updates to line cards. In a distributed scenario, each device has its own RPE that independently processes routing protocols and updates its local routing table, enabling localized decision-making.

Forwarding engines roll is to use the routing information from the route processor to make realtime forwarding decisions for packets. With a centralized model the forwarding engine will rely on direction form the centralized route processor, making forwarding decisions based on the information received from it. Imagine a distributed model where the forwarding engine works independently, utilizing local routing tables to make fast, efficient forwarding decisions without needing to consult a central point.

In centralized forwarding architectures, the route processor and forwarding decisions are made at a central point, while ingress and egress line cards primarily handle the input and output of packets with the switch fabric facilitating the movement of data. In contrast, distributed forwarding allows for localized decision-making, where each component (line cards, RPE, forwarding engine) operates independently, enhancing speed and scalability. Both architectures utilize the same components but in different configurations and operational dynamics, with centralized architectures relying on a singular control point and distributed architectures spreading the decision-making processes across multiple devices.

Software Cisco Express Forwarding (CEF) is a packet-switching method that leverages the device's CPU to manage routing and forwarding tasks. In this architecture, when a packet arrives at a network device, the software CEF examines the packet headers and makes forwarding decisions based on the routing table stored in the device's memory. This process involves several steps: the incoming packet is received, the routing processor performs a lookup in the forwarding information base (FIB), and then the packet is sent out through the appropriate interface. While software CEF provides flexibility and supports advanced features like policy-based routing and deep packet inspection, it can introduce latency due to reliance on the CPU for forwarding decisions. This architecture is typically utilized in smaller devices or scenarios where complex processing is required, but it may not scale well under high traffic conditions due to potential CPU overload.

Hardware CEF (hCEF) is a more efficient forwarding mechanism that offloads the packet processing tasks from the CPU to specialized hardware components, such as application-specific integrated circuits (ASICs) or dedicated forwarding engines. In hardware CEF, the control plane still manages routing protocols and updates the FIB, but the actual packet forwarding is performed by dedicated hardware, enabling high-speed processing and reduced latency. When a packet arrives, the hardware component quickly performs lookups in the FIB and forwards packets with minimal CPU intervention. This architecture is highly scalable and can handle larger volumes of traffic with greater efficiency, making it suitable for high-performance network devices. The use of hardware CEF allows for features such as load balancing and fast switching, while maintaining lower latency and higher throughput compared to software CEF.

Switch Database Management (SDM) templates are configurations applied to Cisco switches that define how system resources are allocated for various features, such as routing, VLANs, and access control lists (ACLs). SDM templates help optimize the switch's performance based on the specific needs of the network. By configuring SDM templates, administrators can allocate resources to match the expected traffic patterns, whether the focus is on IP routing, VLAN management, or other functionalities. For instance, a template might prioritize IP unicast routing if the switch is primarily used for Layer 3 routing, or it might optimize for bridging if the primary function is Layer 2 switching. Administrators can select from various predefined templates or create custom configurations to tailor the device's performance. Properly managing SDM templates is crucial, as improper configurations can lead to resource shortages or degraded performance, particularly in high-traffic scenarios.

Define a VLAN

- vlan vlan-id
- name vlan-name

Configure an interface as a trunk port

- switchport mode trunk

Configure an interface as an access port assigned to a specific VLAN

- switchport mode access
- switchport access vlan vlan-id | name name

Configure a static MAC address entry

- mac address-table static mac-address vlan vlan0id interface interface-id

Clear MAC address from the MAC address table

- clear mac address-table dynamic [address mac-address | interface interface-id | vlan vlan-id]

Assign an IPv4 address to an interface

- ip address ip-address subnet-mask

Assign a secondary IPv4 address to an interface

- ip address ip-address subnet-mask secondary

Assign an IPv6 address to an interface

- ipv6 address ipv6-address/prefix-length

Modify the SDM database

- sdm prefer vlan | advanced

Display the interface that are configured as a trunk port an all the VLANs that they permit

- show interfaces trunk

Display the list of VLANs and their associated ports

- show vlan [brief | id valn-id | name vlan-name | summary]

Display the MAC address table for a switch

- show mac address-table [address mac-address | dynamic | vlan vlan-id]

Display the current interface state, including duplex, speed, and link state

- show interfaces status

Display the Layer 2 configuration information for a specific switch port

- show interfaces interface-id switchport

Display the ARP table

- show ip arp [mac-address | ip-address | vlan vlan-id | interface-id]

Display the IP interface table

- show ip interface [brief | interface-id | vlan vlan-id]

Display the IPv6 interface table

- show ipv6 interface [brief | interface-id | vlan vlan-id]

BPDU (Bridge Protocol Data Unit): A type of network message exchanged between switches (bridges) in a network to maintain the Spanning Tree Protocol (STP). BPDUs help switches to discover and maintain the network topology, determine the root bridge, and calculate the shortest paths.

Configuration BPDU: A type of BPDU used by switches to exchange information about the network topology, including the root bridge identifier, path cost to the root bridge, and port roles. Configuration BPDUs are used during STP calculations to build and maintain the spanning tree.

Designated Port: In STP, a port on a switch that is selected to be the forwarding port for a particular network segment. It is the port that has the best path to the root bridge for that segment. Each segment has one designated port.

Forward Delay: A timer used in STP that defines how long a port stays in the Listening and Learning states before transitioning to the Forwarding state. This delay helps to ensure that all network topology changes are properly processed before the port starts forwarding traffic.

Hello Time: The interval at which BPDUs are sent out by the root bridge to maintain the STP topology. This timer helps in detecting changes in the network and ensures that all switches are synchronized with the current topology.

Local Bridge Identifier: A unique identifier assigned to each switch (bridge) in an STP network, usually consisting of the switch's Bridge Priority and MAC address. It helps to uniquely identify each switch in the network for STP operations.

Max Age: A timer that defines how long a switch should keep a BPDU before discarding it if no newer BPDU is received. This timer helps to ensure that outdated information about the network topology is eventually removed.

Root Bridge: The central switch in an STP network that serves as the reference point for all path calculations. The root bridge is selected based on the lowest Bridge ID, which is a combination of the bridge's priority and MAC address.

Root Bridge Identifier: A unique identifier for the root bridge in an STP network, consisting of the bridge's priority and MAC address. This identifier is used to distinguish the root bridge from other switches in the network.

Root Path Cost: The cumulative cost of the path from a switch to the root bridge. It helps in determining the best path to the root bridge. Each switch calculates this cost based on the cost of each link in the path.

Root Port: The port on a switch that has the lowest path cost to the root bridge. It is the port used to forward traffic towards the root bridge. Each non-root switch has one root port.

System Priority: A value used in combination with the MAC address to form the Bridge ID. The system priority helps in determining the root bridge and other switch roles in the STP process.

System ID Extension: A part of the Bridge ID that extends the MAC address space to uniquely identify switches. This extension allows for a larger number of unique bridge identifiers, especially in networks with many switches.

Topology Change Notification (TCN): A message sent by a switch to notify all other switches in the STP network of a topology change. TCNs trigger the recalculation of the spanning tree to adapt to the changes in the network topology.

Set the STP max age

- spanning-tree vlan vlan-id max-age

Set the STP hello interval

- spanning-tree vlan vlan-id hello-time hello-time

Set the STP forwarding delay

- spanning-tree vlan vlan-id forward-time forward-time

Display the STP root bridge cost

- show spanning-tree root

Display the STP information (root bridge, local bridge, and interfaces) for one or more VLANs

- show spanning-tree [vlan vlan-id]

Identify when the last TCN occurred and which port was the reason for it

- show spanning-tree [vlan vlan-id] detail

802.1D Port States

- Disabled: The port is in an administratively off position (that is, shut down).
- Blocking: The switch port is enabled, but the port is not forwarding any traffic to ensure that a loop is not created. The switch does not modify the MAC address table. It can only receive BPDUs from other switches.
- Listening: The switch port has tgransitioned from a blocking state and can now send or receive BPDUs. It cannot forward any other network traffic. The duration of the state correlates to the STP forwarding time. The next port state is learning.
- Learning: The switch port can now modify the MAC address table with any network traffic that it receives. The switch still does not forward any other network traffic besides BPDUs.; The duration of the state correlates to the STP forwarding time. The next port state is forwarding.
- Forwarding: The switch port can forward all network traffic an can update the MAC address table as expected. This is the final state for a switch port to forward network traffic.
- Broken: The switch has detected a configuration or an operational problem on a port that can have major effects. The port discards packets as long as the problem continues to exist.

802.1D Port Types

- Root port: A network port that connects to the root bridge or an upstream switch in the spanning-tree topology. There should be only one root port per VLAN on a switch.
- Designated port: A network port that receives and forwards BPDU frames to other swithces. Designated ports provide connectivity to downstream devices and switches. There should be only on active designated port on a link.
- Blocking port: A network port that is not forwarding traffic becuase of STP calulations.

The interface STP cost is an essential compnent for root path calulation because the root path is found based on the cumulative interface STP cost to reach the root bridge. The interface STP cost was originally stored as a 16-bit value with a reference value of 20 Gbps. As switches have developed with higher-speed interfaces, 10 Gbps might not be enough. Another method, called long mode, uses a 32-but value and uses a reference speed of 20 Tbps. The original method, known as short mode, has been the default for most switches, but has been transitioning to long mode based on specific platform and OS versions. Devices can be configured with the long-mode interface cost with thecommand spanning-tree pathcost method long. The entire Layer 2 topology should use the same setting for every device in the environment to ensure a consistent topology. Before you enable this setting in an environment, it is important to conduct an audit to ensure that the setting will work.

The root bridge election process is a critical component of the Spanning Tree Protocol (STP), which is used to prevent loops in network topologies with multiple switches. Here's a detailed explanation of how the root bridge election process works:

i. Bridge IDs: Each switch in an STP topology has a unique identifier called the Bridge ID, which consists of two parts: the Bridge Priority (a configurable value) and the MAC address of the switch. The default Bridge Priority is typically set to 32768.

ii. Initiation of Election: When STP begins, all switches consider themselves potential root bridges. Each switch starts by advertising its Bridge ID to its neighbors through Bridge Protocol Data Units (BPDUs).

iii. BPDU Tansmission: Each switch sends out BPDUs containing its own Bridge ID to all of its neighboring switches. This process occurs periodically, allowing switches to communicate and evaluate their IDs.

iv. Comparing Bridge IDs: Upon receiving BPDUs from other switches, each switch compares the Bridge IDs. The switch with the lowest Bridge ID is elected as the root bridge. The comparison process involves:

   - First comparing the Bridge Priority values. The switch with the lowest priority value is favored.
   - If the priorities are equal, the switch with the lowest MAC address wins.

v. Consensus on Root Bridge: As BPDUs are exchanged, all switches will eventually agree on a single root bridge, as they all will learn and propagate the same BPDU with the lowest Bridge ID.

vi. Convergence: Once the root bridge is elected, the other switches determine their roles (designated ports and blocked ports) in the network topology based on their distance from the root bridge, which is measured in terms of path cost. The switches will then update their forwarding tables to reflect the new topology.

vii. Stable State: The network converges to a stable state with one root bridge and a loop-free topology. This state is maintained until there is a change in the network, such as a switch failure or configuration change, prompting a reelection process.

Routing
Services
Overlay
Wireless
Architecture
Security
SDN

1.0 Architecture 15%

1.1 Explain the different design principles used in an enterprise network

- 1.1.a High-level enterprise network design such as 2-tier, 3-tier, fabric, and cloud
- 1.1.b High availability techniques such as redundancy, FHRP, and SSO

1.2 Describe wireless network design principles

- 1.2.a Wireless deployment models (centralized, distributed, controller-less, controller-based, cloud, remote branch)
- 1.2.b Location services in a WLAN design
- 1.2.c Client density

1.3 Explain the working principles of the Cisco SD-WAN solution

- 1.3.a SD-WAN control and data planes elements
- 1.3.b Benefits and limitations of SD-WAN solutions

1.4 Explain the working principles of the Cisco SD-Access solution

- 1.4.a SD-Access control and data planes elements
- 1.4.b Traditional campus interoperating with SD-Access

1.5 Interpret wired and wireless QoS configurations

- 1.5.a QoS components
- 1.5.b QoS policy

1.6 Describe hardware and software switching mechanisms such as CEF, CAM, TCAM, FIB, RIB, and adjacency tables

2.0 Virtualization 10%

2.1 Describe device virtualization technologies

- 2.1.a Hypervisor type 1 and 2
- 2.1.b Virtual machine
- 2.1.c Virtual switching

2.2 Configure and verify data path virtualization technologies

- 2.2.a VRF
- 2.2.b GRE and IPsec tunneling

2.3 Describe network virtualization concepts

- 2.3.a LISP
- 2.3.b VXLAN

3.0 Infrastructure 30%

3.1 Layer 2

- 3.1.a Troubleshoot static and dynamic 802.1q trunking protocols
- 3.1.b Troubleshoot static and dynamic EtherChannels
- 3.1.c Configure and verify common Spanning Tree Protocols (RSTP, MST) and Spanning Tree enhancements such as root guard and BPDU guard

3.2 Layer 3

- 3.2.a Compare routing concepts of EIGRP and OSPF (advanced distance vector vs. link state, load balancing, path selection, path operations, metrics, and area types)
- 3.2.b Configure simple OSPFv2/v3 environments, including multiple normal areas, summarization, and filtering (neighbor adjacency, point-to-point, and broadcast network types, and passive-interface)
- 3.2.c Configure and verify eBGP between directly connected neighbors (best path selection algorithm and neighbor relationships)
- 3.2.d Describe policy-based routing

3.3 Wireless

- 3.3.a Describe Layer 1 concepts, such as RF power, RSSI, SNR, interference, noise, bands, channels, and wireless client devices capabilities
- 3.3.b Describe AP modes and antenna types
- 3.3.c Describe access point discovery and join process (discovery algorithms, WLC selection process)
- 3.3.d Describe the main principles and use cases for Layer 2 and Layer 3 roaming
- 3.3.e Troubleshoot WLAN configuration and wireless client connectivity issues using GUI only
- 3.3.f Describe wireless segmentation with groups, profiles, and tags

3.4 IP Services

- 3.4.a Interpret network time protocol configurations such as NTP and PTP
- 3.4.b Configure NAT/PAT
- 3.4.c Configure first hop redundancy protocols, such as HSRP, VRRP
- 3.4.d Describe multicast protocols, such as RPF check, PIM and IGMP v2/v3

4.0 Network Assurance 10%

4.1 Diagnose network problems using tools such as debugs, conditional debugs, traceroute, ping, SNMP, and syslog
4.2 Configure and verify Flexible NetFlow
4.3 Configure SPAN/RSPAN/ERSPAN
4.4 Configure and verify IPSLA
4.5 Describe Cisco DNA Center workflows to apply network configuration, monitoring, and management
4.6 Configure and verify NETCONF and RESTCONF

5.0 Security 20%

5.1 Configure and verify device access control

- 5.1.a Lines and local user authentication
- 5.1.b Authentication and authorization using AAA

5.2 Configure and verify infrastructure security features

- 5.2.a ACLs
- 5.2.b CoPP

5.3 Describe REST API security
5.4 Configure and verify wireless security features

- 5.4.a 802.1X
- 5.4.b WebAuth
- 5.4.c PSK
- 5.4.d EAPOL (4-way handshake)

5.5 Describe the components of network security design

- 5.5.a Threat defense
- 5.5.b Endpoint security
- 5.5.c Next-generation firewall
- 5.5.d TrustSec and MACsec
- 5.5.e Network access control with 802.1X, MAB, and WebAuth

6.0 Automation 15%

6.1 Interpret basic Python components and scripts
6.2 Construct valid JSON-encoded files
6.3 Describe the high-level principles and benefits of a data modeling language, such as YANG
6.4 Describe APIs for Cisco DNA Center and vManage
6.5 Interpret REST API response codes and results in payload using Cisco DNA Center and RESTCONF
6.6 Construct an EEM applet to automate configuration, troubleshooting, or data collection

6.7 Compare agent vs. agentless orchestration tools, such as Chef, Puppet, Ansible, and SaltStack

In the context of IT infrastructure and network automation, orchestration tools help manage and automate configuration, deployment, and management of systems and services. These tools can generally be classified into two categories: agent-based and agentless. Agent-based tools require a software agent to be installed on the managed nodes (servers or devices). This agent communicates with a central management server to receive and execute configuration tasks. Agentless tools do not require an agent on the managed nodes. Instead, they use standard protocols (like SSH or WinRM) to connect to and configure the nodes directly. Chef uses a client-server model where the Chef client (agent) runs on each node. It communicates with the Chef server to retrieve and apply configurations. Chef uses ”recipes” and ”cookbooks” written in Ruby to define configurations. The Chef server stores these recipes and the node's state. Puppet also uses a client-server model with Puppet agents installed on managed nodes. These agents periodically check in with the Puppet master server to apply configurations. Puppet uses ”manifests” written in its own declarative language to define configurations. The Puppet master server maintains these manifests and the node's state. Ansible uses a push-based model where configurations are pushed from the Ansible control node to managed nodes over SSH (or WinRM for Windows). No agents are required on the nodes. Ansible uses YAML for defining playbooks which describe the desired state of the systems. It operates in an ad-hoc manner, executing commands as needed. SaltStack offers both agent-based and agentless options. In its agent-based mode, Salt uses Salt minions (agents) on managed nodes. In agentless mode, it can operate over SSH to manage systems. SaltStack uses YAML for defining configurations and states. It supports both push and pull models, and provides real-time command execution capabilities. Choosing between agent-based and agentless orchestration tools depends on factors like the size and complexity of your infrastructure, the level of control and state management required, and the existing toolchain and practices within your organization.