CCNP Route Master Study Guide

QUIZLET FLASHCARDS - [*https://quizlet.com/_4ckrl4*](https://quizlet.com/_4ckrl4)

1.0 Network Principles

1.1 Identify Cisco Express Forwarding concepts

- **GENERAL CONCEPT**
    - Considered as switching within routers
    - Enables faster packet transfer from one interface to another
    - This is due embedding relevant information in hardware
- **PROCESS-BASED SWITCHING**
    - Eats up CPU resources to switch packets
    - Known as ***"IP Input"*** in the CPU table
- **FAST SWITCHING**
    - Store data in ***"Fast Switching Cache"*** for faster lookups
    - CPU would have more breathing room
    - Packet would first need to be analyzed by the CPU before entering the cache
    - Does not track the routing table so the cached entry aging out may take a bit
- **CISCO EXPRESS FORWARDING**
    - Pre-populated cache, downloads information from arp/etc as soon as available
    - Enabled by default on routers and multilayer switches
    - CEF cannot be disabled on switches due to specialized hardware
    - Some packets on switches ***cannot*** be CEF-switched:
        - ARP Requests (Glean)
        - Packets requiring response from router CPU such as MTU too large
        - Routing Protocol Traffic
        - CDP or LLDP
        - Packets needing encryption

1.1.a FIB

- **A.K.A. *"Forwarding Information Base"***
    - Shadow copy of the routing table
    - AD, Tags, Metrics, and more are not included in the FIB table
- **VERIFICATION**
    - `show ip cef [detail]`
    - `show ip cef <ip address><mask> detail`

1.1.b Adjacency Table

- **GENERAL CONCEPT**

- Pre-Populated with L2 tables such as:
    - ARP Table
    - Frame-Relay Map Table
- If FIB does not have a specific IP address…
    - An ARP request is sent out so that it can be seen in the FIB as ‘attached’
    - This means there was a successful lookup

- **ADJACENCY TYPES**
    - **Glean**
        - Need to wait for an ARP, so the request is punted to the CPU
        - Not enough L2 information to forward the L3 packet
        - If reply does not come back, entry stays as Glean until it ages out
    - **Null**
        - Valid packet but needs to be dropped
        - Could happen because route has null0
        - Processed in hardware
    - **Drop**
        - Valid packet but needs to be dropped
        - Could happen because no route or something is wrong with packet
    - **Discard**
        - Valid packet but needs to be dropped
        - Could happen because we have a security policy, such as an ACL
    - **Punt**
        - Similar to Glean, packets will be forwarded to the CPU
        - Punt is for packets that are destined for our processor to begin with
- **VERIFICATION**
    - `show adjacency <intf type/number> [summary | detail]`
    - `show adjacency vlan <vlan-id> detail`

1.2 Explain general network challenges

1.2.a Unicast Flooding

- **GENERAL CONCEPT**
    - Occurs when no entry in L2 table (CAM)
        - Packet is flooded out all interfaces in the Vlan
        - Only exception is the received interface
    - To reduce flooding, make the ARP entry expire earlier than the CAM entry
        - The ARP entry would repopulate before CAM entry expires
    - To block unicast flooding on a per interface basis:
        - `Router(config-if)# switchport block unicast`

1.2.b Out-of-order packets

- **GENERAL CONCEPT**
    - Packets arriving in a different order than they were sent

        - Could be caused by asymmetric routing or packet loss
    - TCP attempts to alleviate this issue via the use of sequence numbers
        - Allows receiver to reorder the packets
    - If packets arrive out of order, the receiver may send *duplicate ACKs*
    - This could trigger the **"Fast Retransmit"** algorithm
        - The sender will assume packet loss, and retransmit sequences
        - The sender will reduce window size
        - This leads to reduced TCP throughput

1.2.c Asymmetric routing

- **GENERAL CONCEPT**
    - When the return path is different from the original path of a packet
    - Not normally a problem unless there are mechanisms to drop the data
        - Examples: NAT, Firewalls, Unicast RPF

1.3 Describe IP operations

1.3.a ICMP Unreachable and Redirects

- **ICMP UNREACHABLE (TYPE 3)**
    - Disabled by default for security reasons
    - Feedback mechanism to notify the host about an unreachable destination
    - Up to the sender to act upon that information from router
    - **Six failure feedback codes:**
        - **Code-0**: *"Network Unreachable"*, no route to the network in question
        - **Code-1**: *"Host Unreachable"*, no host exists on that subnet
        - **Code-2**: *"Protocol Unreachable"*, unknown protocol destined for router
        - **Code-3**: *"Port Unreachable"*
        - **Code-4**: *"Fragmentation required and DF bit set"*
        - **Code-5**: *"Source Route Failed"*
- **ICMP REDIRECTS (TYPE 5)**
    - A packet destined for a specific gateway which is not the preferred gateway is then redirected to the preferred gateway
    - The sender is then sent an ICMP Redirect message
        - Indicates that the message has been redirected
        - Suggests to use the preferred gateway going forward

1.3.b IPv4 and IPv6 fragmentation

- **GENERAL CONCEPT**
    - Payload of an IP packet is greater than the MTU of the link
        - Must be fragmented, otherwise it will be dropped
        - Routers will fragment packets in IPv4, unless the DF bit is set
    - Only source host fragments IPv6 packets using Extension Header

- Two choices for IPv6 packets to avoid fragmentation:
    - Use **Path MTU Discovery** (PMD) to find the lowest MTU in path
    - Use the minimum MTU size (1280 Bytes)

1.3.c TTL

- **GENERAL CONCEPT**
    - Originally intended for routers to decrement the seconds
        - Based upon how long each packet was queue
        - This feature was never implemented
    - Instead, TTL is used as a glorified hop count system
        - Each router decrements the TTL by 1
        - Upon reaching 0, the packet is dropped
    - **Traceroute** uses this concept to determine hops
        - Increments TTL for each packet until destination is reached
        - Each hop will report back ***"ICMP Time Exceeded"***
        - Final destination will report back with ***"ICMP Destination Unreachable"***

1.4 Explain TCP operations

1.4.a IPv4 and IPv6 (P)MTU

- **GENERAL CONCEPT**
    - MTU A.K.A. ***"Maximum Transmission Unit"***
    - The maximum amount of bytes supported in the payload of the transmission
    - Default MTU is set as 1500 bytes
    - Different technologies can add more bytes to headers requires MTU adjustment
- **CONFIGURATION STEPS**
    - `Router(config-if)# mtu <bytes>`
- **VERIFICATION**
    - `show system mtu`
- **PATH MTU DISCOVERY CONCEPT**
    - PMD can be used to avoid fragmentation
    - IPv6 source sends out probes to discover lowest MTU in path
    - ***Packet Too Big*** ICMPv6 message will be sent back to host, containing MTU
    - IPv4 will have DF A.K.A. ***"Don't Fragment"*** bit set
    - Any router that drops the packet in the path will send back *ICMP Type 3 Code 4*

1.4.b MSS

- **GENERAL CONCEPT**
    - MSS A.K.A. ***"Maximum Segment Size"***
    - The amount of data a host will accept in a single TCP/IP datagram
    - MSS > MTU + protocol overhead = fragmentation at an IP level
        - Should attempt to avoid fragmentation when using tunneling techniques

- Typically, MSS is calculated at 1460 bytes
    - Allocates 20 bytes respectively for TCP and IP header overhead
- The MSS of a host is sent in TCP SYN
    - Each host uses the lowest of the two values
- **CONFIGURATION STEPS**
    - `Router(config-if)# `**`ip tcp adjust-mss`**` <max-segment-size>`
    - `Router(config-if)# `**`ip mtu`**` <mtu-size>`
- **VERIFICATION**
    - `show ip interface <intf type/number>`

1.4.c Latency

- **GENERAL CONCEPT**
    - Often defined by RTT A.K.A ***"Round Trip Timer"***
    - Length of time it takes to receive a response back
    - TCP Latency has an inverse relationship with throughput
        - More latency, less throughput

1.4.d Windowing

- **GENERAL CONCEPT**
    - Allows single acknowledgment of multiple TCP segments
    - ***Window Size*** specifies how many bytes may be sent before an ACK is required
        - **Example**: Window Size 1000, ACK 1001 = successful, increase size
        - **Example**: Window Size 1000, ACK 900 = unsuccessful, decrease size
    - Can be adjusted based upon host requirements

1.4.e Bandwidth-delay product

- **GENERAL CONCEPT**
    - Amount of data that can be at transit at any given point of time
    - Calculated by multiplying the link's bandwidth in bits and Round Trip Delay Time
    - Networks with large bandwidth-delay product are known as LFN
        - LFN A.K.A. ***"Long Fat Network"***
        - **Example**: Satellite Link - high bandwidth but huge delays
    - TCP Window Scaling is used to alleviate the issue

1.4.f Global synchronization

- **GENERAL CONCEPT**
    - Refers to how TCP streams gradually increase window sizes until drops occur
    - This causes TCP streams to shrink at once, then repeat the process
    - Ends up with a sawtooth-like graph for bandwidth utilization on the link
    - One way to prevent this issue is by discarding random TCP streams
        - A.K.A. ***"Random Early Detection"*** queuing
        - Only some streams will back off, allowing more efficient link utilization

1.5 Describe UDP operations

1.5.a Starvation

- **GENERAL CONCEPT**
    - Occurs when TCP and UDP streams occupy the same queue
    - When congestion occurs…
        - TCP reacts by reducing Window Size, reducing bandwidth utilization
        - UDP reacts by eating up the newly available bandwidth
    - This results in ***UDP Dominance*** and ***TCP Starvation***
    - Solution is the create separate queues for TCP and UDP, or use QoS

1.5.b Latency

- **GENERAL CONCEPT**
    - UDP does not suffer from throughput related issues as TCP
        - UDP does not expect to receive ACKs for the data
        - UDP is not affected by packet loss or latency
    - UDP latency can affect application performance, such as VoIP
    - UDP is generally used for real time applications
        - These applications can be sensitive to latency and jitter
        - Jitter is the variation of latency over time
    - These applications may attempt to alleviate these issues by using buffers

1.6 Recognize proposed changes to the network

1.6.a Changes to routing protocol parameters

- **GENERAL CONCEPT**
    - Metrics can be changed to provide added flexibility to routing protocols
    - Other types of changes involve redistribution or added routes

1.6.b Migrate parts of the network to IPv6

- **GENERAL CONCEPT**
    - Migration techniques involve "**Dual Stack"** and "**Tunneling"**
    - Tunneling is encapsulating IPv6 data into an IPv4 packet
    - Dual Stack involves running both IPv4 and IPv6 on the device
- **TUNNELING CONCEPT**
    - Router or host encapsulates the IPv6 packet inside an IPv4 packet
    - Results in fewer routers needing any IPv6 configuration at all
    - Two main categories of tunnels are: point-to-point and multipoint
    - Protocol number of the IP Packet will be 41 (IPv6) which means encapsulation
- **POINT-TO-POINT IPv6 TUNNEL CONCEPT**

- Two devices sit at the ends of the tunnel
- These point-to-point tunnels work like virtual point-to-point serial links
- Each router configures a type of virtual interface called tunnel interface
- Point-to-Point tunnels work best when IPv6 occurs regularly

- **POINT-TO-MULTIPOINT IPv6 TUNNEL CONCEPT**
    - Allows router to use a tunnel interface to send packets to multiple destinations
    - Additional logic so that the sending router knows which remote router to send to
    - Point-to-Multipoint works best when IPv6 traffic occurs infrequently
    - Does not support IPv6 IGPs, thus requiring statics or BGP
- **OVERVIEW OF IPv6 TUNNELING OPTIONS**
    - **Manually Configured Tunnel**
        - Point-to-Point tunnel that is configured manually
        - Acts like a virtual point-to-point link, supporting IPv6 IGPs
        - Good for more permanent tunnels
        - Slightly less overhead than GRE
    - **GRE Tunnel**
        - Point-to-POint tunnel that is configured manually
        - Same advantages as manually configured tunnels
        - Can support other Layer 3 protocols over the same tunnel
    - **6to4 Tunnel**
        - Multipoint tunnel that is formed dynamically
        - Require less configuration than all other types when adding new site
        - Supports global unicasts, with some extra configuration
        - Uses second and third quartets to store IPv4 address
    - **ISATAP Tunnel**
        - Multipoint tunnel is dynamically formed
        - Easily supports global unicast addresses for all prefixes
        - Uses seventh and eighth quartets to store IPv4 address
    - **IPv4-Compatible Tunnel**
        - Does not scale well for large networks, not recommended
        - Address format is ::IPv4Address, automatically created
    - **IPv6 Rapid Deployment (6rd) Tunnel**
        - An extension of the 6to4 feature
        - Allows an ISP to offer IPv6 service to customers over its IPv4 network
    - **Teredo Tunnel**
        - Known as shipworm
        - Two dual-stacked devices can speak even with IPv4 NAT in between
        - Uses UDP port 3544 to communicate with Teredo servers
        - These servers are used as dispatchers between clients and relays
        - Although proposed by Microsoft, there is a version for Linux
        - Defined in RFC 4380
        - Uses 2001::/32 as the prefix, with the format being:
            - 32-bits: 2001::/32
            - 32-bits: Server Public IPv4 Address

        - 16-bits: Flags
        - 16-bits: Obfuscated Client UDP Port
        - 32-bits: Obfuscated Client Public IPv4 Address
- **MANUALLY CONFIGURED TUNNELS**
    - **Many similarities between GRE:**
        - Both create virtual point-to-point links between two IPv4 routers
        - IPv6 IGP routing protocols can be run over these virtual links
        - Difference is that MCT encapsulates packets without an additional header
    - **Configuration Steps:**
        - `Router(config)#` **`interface loopback`** `{number}`
        - `Router(config-if)#` **`ip address`** `{ipv4-address} {subnet}`
        - `Router(config)#` **`interface tunnel`** `{number}`
        - `Router(config-if)#` **`tunnel source`** `{interface | ipv4-address}`
        - `Router(config-if)#` **`tunnel destination`** `{ipv4-address}`
        - *Must match tunnel source on the other router!*
        - `Router(config-if)#` **`tunnel mode ipv6ip`**
        - `Router(config-if)#` **`ipv6 address`** `{ipv6-address/prefix}`
    - **Verification:**
        - **`show interface tunnel`**
        - **`show ipv6 interface brief`**
- **GENERIC ROUTING ENCAPSULATION TUNNELS**
    - Only one difference between MCT and GRE: tunnel mode
    - **Configuration Steps:**
        - `Router(config-if)#` **`tunnel mode gre ip`**
    - **IOS Default MTU Settings:**
        - **GRE**: 1476
        - **MCT**: 1480
- **IPv6 AUTOMATIC 6TO4 TUNNELS**
    - **Addressing Option-1:**
        - Use the 2002 prefix, reserved for 6to4 protocol
        - Example: 2002:0101:0101:xxxx::/64 -> 1.1.1.1
        - Embeds the IPv6 information within the IPv6 address
    - **Addressing Option-2:**
        - Provide global prefix to the IPv6 cloud segments
        - IPv4 address cannot be embedded in this scenario
        - This in return requires routing to be specific with next-hop
    - **Configuration Steps:**
        - `Router(config)#` **`interface tunnel`** `{number}`
        - `Router(config-if)#` **`tunnel source`** `{interface | ipv4-address}`
        - *Derive the /48 prefix used for allocating local IPv6 subnets*
        - `Router(config-if)#` **`ipv6 address`** `2002:0101:0101::1/64`
        - `Router(config-if)#` **`tunnel mode ipv6ip 6to4`**
        - `Router(config)#` **`ipv6 route`** `<remote global prefix> tunnel 0 <next-hop>`
        - *The next hop command is only required with option-2 addressing*
    - **Verification**:
        - **`show ipv6 route`**
        - **`show ipv6 interface tunnel`**
- **ISATAP TUNNELS**

- Enables reachability for IPv6 Hosts not directly connected to an IPv6 router
- ISATAP A.K.A. “Intra-Site Tunnel Addressing Protocol”
- Designed for transporting IPv6 packets *within* a site
- 64-Bit Interface Identifier (after the /64 prefix):
    - First 32 bits contain the value 0000:5EFE to indicate ISATAP address
    - Remaining 32 bits encode the IPv4 address
    - ipv6prefix:0000:5efe:ipv4address
- Allows hosts to tunnel IPv6 packets through an IPv4 domain
- This requires the use of a DNS server
- Configuration Steps:
    - `Router(config)#` **`ipv6 unicast-routing`**
    - `Router(config)#` **`interface tunnel`** `{number}`
    - `Router(config-if)#` **`ipv6 add`** `2001:1111:2222:aaaa::/64 eui-64`
    - `Router(config-if)#` **`no ipv6 nd suppress-ra`**
    - *`Allows a router to send router advertisements`*
    - `Router(config-if)#` **`tunnel source`** `{interface | ipv4-address}`
    - `Router(config-if)#` **`tunnel mode ipv6ip isatap`**

- **IPv6 RAPID DEPLOYMENT (6RD) TUNNELS**
    - **Main Differences Between 6rd and 6to4:**
        - Does not require 2002::/16 prefix and can be from service provider
        - Not all 32 bits of the IPv4 destination address need to be carried
        - The IPv4 destination address is obtained from a combination of:
            - Bits in the payload header
            - The information configured in the router
        - Defined in RFC 5569

1.6.c Routing protocol migration

- **GENERAL CONCEPT**
    - Routing protocol migration can be done by setting up boundaries
    - Modifying AD can ensure no routing issues

2.0 Layer 2 Technologies

2.1 Configure and verify PPP

- **GENERAL CONCEPT**
    - PPP A.K.A. ***“Point-to-Point Protocol”***
    - Operates in the LLC sub-layer of the data link layer in OSI
    - Server generally known as ***Access Concentrator***
- **PPP STAGES**
    - **Phase-1**: Active Discovery (only in PPPoE)
    - **Phase-2**: Link Control Protocol (LCP)
    - **Phase-3**: Authentication (CHAP or PAP)

- **Phase-4**: Network Control Protocol (NCP)
- **ACTIVE DISCOVERY (PPPoE)**
  - → **PADI** (PPPoE Active Discovery Initialization)
    - Broadcast from client to AC
    - *"Are there any PPPoE Servers out there? My unique Host-ID is xx-xx"*
  - ← **PADO** (PPPoE Active Discovery Offer)
    - Unicast from AC to client
    - *"Yes, I'm here xx-xx. My unique AC ID is yy.yy"*
  - → **PADR** (PPPoE Active Discovery Request)
    - Unicast from client to AC
    - *"Thanks for the info! Can I have a Session-ID please?"*
  - ← **PADS** (PPPoE Active Discovery Session-Confirmation)
    - Unicast from AC to client
    - *"Yes, let's use Session-ID 0x02"*
- **LINK CONTROL PROTOCOL (LCP) OPTIONS**
  - **Callback Option**
    - Reinitiate dial session on the lower rate side
    - Must be configured on both ends of the session
  - **Multilink Option**
    - If frames are larger than MTU, will initiate fragmentation
    - MRRU A.K.A. ***"Maximum Received Reconstructable Unit"***
      - Specifies maximum amount of bytes reconstructable
      - Example: MTU is 1500 bytes, MRRU is 3000 bytes
      - Once byte limit is reached, packet will be dropped
    - MRU A.K.A. ***"Maximum Receivable Unit"***
      - Specifies the MTU size, do not confuse with MRRU
  - **Authentication Option**
    - Specifies the type of authentication if any are to be done
  - **Magic Number Option**
    - Loop prevention system
    - If the same magic number is sent and received, then assumes loop
- **LINK CONTROL PROTOCOL (LCP) CONTROL MESSAGES**
  - **Configuration-Request**
    - Lists all link specific options a sender wishes to implement
  - **Configuration-Reject**
    - When a receiver does not support a particular feature and offers no alternatives
  - **Configuration-NAK** (Negative Acknowledgement)
    - Receiver does not support a particular feature and offers an alternative
  - **Configuration-Acknowledgement**
    - Acknowledging all LCP options in the most recent received Config-Req
- **LCP CONFIGURATION STEPS**
  - `Router(config-if)# encapsulation ppp`
  - `Router(config-if)# ppp multilink`
- **NETWORK CONTROL PROTOCOL (NCP)**

- Negotiate what Layer 3 Protocol to use:
    - **IP**: IPCP
    - **IPX**: IPXCP
    - **CDP**: CDPCP
- DHCP could also be used but not really necessary as IPCP is easier

- **NCP CONFIGURATION STEPS**
    - *Specifically, this is for IPCP in case DHCP is not preferred*
    - `Router(config-if)#` **`ip address negotiated`**
    - `Router(config-if)#` **`peer default ip address`** `<x.x.x.x |` **`dhcp`** `|` **`dhcp-pool`** `|` **`pool`**`>`

2.1.a Authentication (PAP, CHAP)

- **PAP GENERAL CONCEPT**
    - PAP A.K.A. ***"Password Authentication Protocol"***
    - Sends clear text username and password for authentication
    - Two-way handshake and by default sends hostname as username
    - Once PPP authentication is done, PPP session will stay up and established
- **PAP CONFIGURATION STEPS**
    - ***Client One Way***
    - `Router(config-if)#` **`encapsulation ppp`**
    - `Router(config-if)#` **`ppp authentication pap callin`**
    - `Router(config-if)#` **`ppp pap sent-username`** `Chris` **`password`** `Cisco`
    - ***Client Two Way***
    - `Router(config)#` **`username`** `Sally` **`password`** `Server`
    - `Router(config-if)#` **`encapsulation ppp`**
    - `Router(config-if)#` **`ppp authentication pap callin`**
    - `Router(config-if)#` **`ppp pap sent-username`** `Chris` **`password`** `Cisco`
- **CHAP GENERAL CONCEPT**
    - CHAP A.K.A. ***"Challenge Handshake Authentication Protocol"***
    - Three-way handshake and by default sends hostname as username
    - A CHAP challenge is sent from the server
        - Combination of random numbers
        - Password never sent across the link
        - ***Challenge + Username + Password*** = **MD5 HASH**
    - Client sends MD5 hash across link to server and server verifies validity
    - Once PPP authentication is done, PPP session will stay up and established
- **CHAP CONFIGURATION STEPS**
    - ***Client One Way***
    - `Router(config-if)#` **`encapsulation ppp`**
    - `Router(config-if)#` **`ppp authentication chap callin`**
    - `Router(config-if)#` **`ppp chap hostname`** `Chris`
    - `Router(config-if)#` **`ppp chap password`** `Cisco`
    - ***Alternative Client***
    - `Router(config)#` **`username`** `Sally` **`password`** `Cisco`
    - `Router(config-if)#` **`encapsulation ppp`**
    - `Router(config-if)#` **`ppp authentication chap callin`**
    - ***Client Two Way***
    - `Router(config)#` **`username`** `Sally` **`password`** `Cisco`
    - `Router(config-if)#` **`encapsulation ppp`**

- Router(config-if)# **ppp authentication chap**
- Router(config-if)# **ppp chap hostname** Chris
- Router(config-if)# **ppp chap password** Cisco

- **VERIFICATION**
  - **debug ppp negotiations**
  - **debug ppp authentication**
  - **show interface serial** <number>
  - **show users**

2.1.b PPPoE (client side only)

- **GENERAL CONCEPT**
  - PPPoE A.K.A. ***"PPP Over Ethernet"***
  - Allows single DSL connection to support an entire LAN of PPP clients
  - BBA A.K.A. ***"BroadBand Access"***
- **PPPoE CONFIGURATION STEPS**
  - ***Server Side***
  - Router(config)# **hostname** Router
  - Router(config)# **username** client **password** cisco
  - Router(config)# **bba-group pppoe** PPPOE-GROUP
  - Router(config-bba-group)# **virtual-template** 1
  - Router(config)# **interface virtual-template** 1
  - Router(config-if)# **ip unnumbered** loopback 0
  - Router(config-if)# **peer default ip address pool** MY-POOL
  - Router(config-if)# **ppp authentication chap**
  - Router(config)# **ip local pool** MY-POOL 1.2.1.2 1.2.1.254
  - Router(config)# **interface** FastEthernet0/0
  - Router(config-if)# **pppoe enable group** PPPOE-GROUP
  - ***Client Side***
  - Router(config)# **hostname** Router
  - Router(config)# **interface Dialer** 7
  - Router(config-if)# **ip address negotiated**
  - Router(config-if)# **encapsulation ppp**
  - Router(config-if)# **dialer pool** 1
  - Router(config-if)# **ppp chap password** cisco
  - Router(config)# **interface** FastEthernet0/0
  - Router(config-if)# **pppoe-client dial-pool-number** 1
- **VERIFICATION**
  - **show pppoe session**
  - **show pppoe summary**
  - **show interface virtual-access** <number>
- **PPPoE MTU CONSIDERATIONS**
  - PPP adds 8 bytes of overhead on Ethernet Frame
  - 1508 Bytes + 14 Bytes = 1522 Bytes
    - Results in fragmentation
    - CPU-intensive process
  - Server side not a huge issue
    - Virtual-Access interfaces defaults to 1492 MTU
  - Client side it is an issue

        - Dialer interface defaults to 1500 MTU
    - Adjust-MSS intercepts TCP packets and adjusts MSS to avoid fragmentation
        - Default for Windows is 1460 otherwise
- **PPPoE MTU CONFIGURATION STEPS**
    - ***Client Side***
    - `Router(config)#` **`interface Dialer`** `7`
    - `Router(config-if)#` **`ip mtu 1492`**
    - `Router(config)#` **`interface`** `FastEthernet0/0`
    - `Router(config-if)#` **`ip tcp adjust-mss 1452`**
- **PPPoE VPDN CONFIGURATION STEPS**
    - ***Client Side***
    - `Router(config)#` **`vpdn enable`**
    - `Router(config)#` **`vpdn group`** `CBTNuggets` **`request dialout`** `pppoe`
    - `Router(config)#` **`vpdn group`** `CBTNuggets` **`ppp authentication`** **`<pap/chap/mschap>`**
    - `Router(config)#` **`vpdn group`** `CBTNuggets` **`localname`** `ninja`
    - `Router(config)#` **`vpdn username`** `ninja password secretninja` **`store-local`**
    - `Router(config)#` **`interface`** `fa0/0`
    - `Router(config-if)#` **`ip address pppoe setroute`**
    - `Router(config-if)#` **`pppoe client vpdn group`** `CBTNuggets`

2.2 Explain Frame Relay

2.2.a Operations

- **GENERAL CONCEPT**
    - Multipoint layer 2 technology
    - Legacy technology typically used in service provider end
    - ISP Frame Relay Switches monitor data usage
        - This is done by monitoring transmit and receive buffers
        - If congestion occurs, various bits in Frame-Relay header are modified
- **FRAME RELAY ADVANCED CONCEPTS**
    - PVC A.K.A. ***"Private Virtual Circuit"***
        - Service provider carries multiple customer traffic in a single link
    - DLCI A.K.A. ***"Data-Link Connection Identifier"***
        - Works as a Layer 2 address in Frame Relay
        - 10-bit value (can be extended) that ranges from 0 to 1023
            - 0-15 and 1007-1023 are reserved
    - LMI A.K.A. ***"Local Management Interface"***
        - Works as a keepalive between Frame Relay switch and end device
        - Propagates DLCI information to hub and spoke
        - Standards of LMI: Cisco, Q933a, ANSI
        - Uses reserved DLCI number of 1023 or 0
    - CIR A.K.A. ***"Committed Information Rate"***
        - Amount of data rate that a service provider guarantees
        - Anything above CIR is considered as BURST
        - Closer CIR is to access rate (total bandwidth), the more expensive

- Inverse ARP
  - Dynamically maps the destination IP with corresponding local DLCI
  - Only one DLCI can be mapped with a Layer 3 address
  - Subinterfaces disable this functionality
- **FRAME RELAY CONGESTION**
  - DE A.K.A. ***"Discard Eligible"***
    - If customer goes above CIR, their packets are marked with DE bit
    - If there is congestion and DE bit is set, packets can be dropped WAN
  - FECN A.K.A. ***"Forward Explicit Congestion Notification"***
    - If the transmit buffer is near full, WAN switch will set the FECN bit
    - Just informational, notifies switch that there is congestion going towards it
  - BECN A.K.A. ***"Backward Explicit Congestion Notification"***
    - Sent to notify router about congestion if frames are sent toward DLCI
    - Can have the router respond to BECNs by slowing down transmission
- **LMI EXTENSION OPTIONS**
  - **Virtual Status Messages**
    - Provide communication between the network and user device
    - Prevents data from entering black holes when PVC fails
  - **Multicasting**
    - Allows a sender to transmit a single frame to multiple recipients
  - **Global Addressing**
    - Gives connection identifiers global rather than local significance
    - This allows the Frame Relay cloud to perform like one big LAN
  - **Simple Flow Control**
    - Provides flow control mechanisms that applies to the entire interface
    - Intended for devices that cannot use the congestion notification bits
- **VERIFICATION**
  - **`show frame-relay map`**

2.2.b Point-to-point

- **CONFIGURATION STEPS**
  - `Router(config-if)#` **`no ip address`**
  - `Router(config-if)#` **`encapsulation frame-relay`**
  - `Router(config-if)#` **`no shut`**
  - `Router(config-if)#` **`exit`**
  - `Router(config)#` **`interface serial`** `x/y.<subinterface number> point-to-point`
  - `Router(config-sub-if)#` **`ip address`** `<address> <subnet mask>`
  - `Router(config-sub-if)#` **`frame-relay interface-dlci`** `<dlci>`

2.2.c Multipoint

- **CONFIGURATION STEPS**
  - ***Physical Interface Configuration***
  - `Router(config-if)#` **`encapsulation frame-relay`**
  - `Router(config-if)#` **`ip address`** `<address> <subnet mask>`
  - ***Configure Static Mapping*** *(required with subinterfaces)*

- Router(config-if)# **frame-relay map ip** <destination address> <dlci> broadcast

3.0 Layer 3 Technologies

3.1 Identify, configure, and verify IPv4 addressing and subnetting

3.1.a Address types (Unicast, broadcast, multicast, and VLSM)

- **GENERAL CONCEPT**
  - **Unicast**
    - Assigned to a single network interface located on a specific subnet
    - Used for **one-to-one** communications
  - **Broadcast**
    - Assigned to all network interfaces located on a subnet on the network
    - Used for **one-to-everyone** communications
  - **Multicast**
    - Assigned to one or more network interfaces located on various subnets
    - Used for **one-to-many** communications
  - **VLSM**
    - VLSM A.K.A. ***"Variable-Length Subnet Mask"***
    - Allows divisions of an IP address space beyond classful system
    - Introduced alongside of **CIDR** to slow IPv4 address pool depletion

3.1.b ARP

- **ARP A.K.A. *"Address Resolution Protocol"***
  - Communication protocol
  - Used to discover link layer address associated with a given IPv4 address

3.1.c DHCP relay and server

- **DHCP A.K.A. *"Dynamic Host Configuration Protocol"***
  - See **Section 6.5** for expanded details
- **DHCP RELAY AGENT**
  - Router forwards DHCP discovery as unicast from client to DHCP server
  - This transaction is transparent to the client
- **DHCP RELAY CONFIGURATION**
  - Router(config-if)# **ip helper-address** 192.168.23.3
- **DHCP SERVER CONFIGURATION**
  - DHCP(config)# **ip dhcp pool** EXAMPLE
  - DHCP(dhcp-config)# **network** 192.168.12.0
  - DHCP(dhcp-config)# **dns-server** 208.67.222.222
  - DHCP(dhcp-config)# **default-router** 192.168.12.1
  - DHCP(config)# **ip dhcp excluded-address** 192.168.12.100
- **VERIFICATION**

- `show ip dhcp binding`

3.1.d DHCP protocol operations

- **D.O.R.A. A.K.A.** ***"Discover, Offer, Request, Acknowledge"***
    - See **Section 6.5** for expanded details
    - DHCP built on top of **bootstrap protocol (bootp)**

3.2 Identify IPv6 addressing and subnetting

3.2.a Unicast

- **GENERAL CONCEPT**
    - **IPv6 String Format:**
        - They are 128-bits in length and in hex
        - Represented in all lowercase
        - `x:x:x:x:x:x:x:x`
            - Each x is a 16-bit section
            - Each section referred to as a hextet
    - **Helpful Notation Rules:**
        - **Rule 1**: Omit Leading 0s
            - **Example**: 2001:0001:0010… -> 2001:1:10…
        - **Rule 2**: Omit All-0s Hextets
            - **Example**: fe80:0000:0000: **…** :0001 -> fe80::1
- **GUA A.K.A.** ***"Global Unicast Address"***
    - **Global Routing Prefix**: Prefix of the address assigned by provider
    - **Subnet ID**: Separate field for allocating subnets within customer site
    - **Interface ID**: Identifies the interface on a subnet, generally 64-bits
    - Several ways a device can be configured with GUA:
        - Manually configured
        - Stateless Address Autoconfiguration (SLAAC)
        - Stateful DHCPv6
- **LINK-LOCAL UNICAST ADDRESS**
    - All devices MUST have an IPv6 link-local address, fe80::/10
    - Not routable off the link
    - Only unique on the link
    - Only one link-local address per interface
    - Devices can use DAD A.K.A. ***"Duplicate Address Detection"***
- **LOOPBACK ADDRESSES**
    - An IPv6 loopback address is ::1
    - Equivalent to IPv4 127.0.0.0/8
- **UNSPECIFIED ADDRESSES**
    - An all-0s address (::)
    - Used as a source address to indicate the absence of an address

    - Cannot be assigned to an interface
    - Cannot be used as destination address
    - Router will never forward a packet that has an unspecified source address
- **ULA A.K.A.** ***"Unique Local Addresses"***
    - Counterpart of IPv4 private addresses, fc00::/7
    - Allow sites to be combined or privately interconnected
    - NAT should **not** be used to translate between ULA and GUA
    - **RFC 4193** defines a process whereby…
        - All locally assigned Global IDs can be generated dynamically
        - Pseudo random algorithm gives it a very high probability of being unique
    - **Site-Local Addresses** (the original ULA) have been deprecated (fec0::/10)
- **IPv4 EMBEDDED ADDRESS**
    - Used to aid the transition from IPv4 to IPv6
    - Features such as NAT64 are required to translate between the two address families
    - **Example Address**: ::ffff:192.168.10.10

3.2.b EUI-64

- **GENERAL CONCEPT**
    - Two options to generate the Interface ID of an IPv6 address:
        - EUI-64 process
        - Random 64-bit value (privacy extension)
    - The EUI-64 uses the Ethernet MAC address to generate the Interface ID
- **CONVERSION STEPS**
    - **Step-1:** Convert MAC to binary, then split in half
    - **Step-2:** FF:FE is inserted between the two halves
    - **Step-3:** Flip the seventh bit **->** Universally/Locally bit A.K.A. ***"Local/Global bit"***
    - **Result**: *aaaa.aaaa.aaaa* -> *fe80::a8aa:aaff:feaa:aaaa*

3.2.c ND, RS/RA

- Covered in depth in **Section 6.5.a**

3.2.d Autoconfig (SLAAC)

- Covered in depth in **Section 6.5.a**

3.2.e DHCP relay and server

- Covered in depth in **Section 6.5.a**

3.2.f DHCP protocol operations

- Covered in depth in **Section 6.5.a**

3.3 Configure and verify static routing

- **GENERAL CONCEPT**
  - ***Static Routes*** are manually configured routes
- **IPv4 CONFIGURATION STEPS**
  - `Router(config)# ip route <prefix> <mask> {interface | next-hop}`
- **IPv6 CONFIGURATION STEPS**
  - `Router(config)# ipv6 route <prefix>/<length> {interface | next-hop}`
- **VERIFICATION**
  - `show ip route`
  - `show ipv6 route`

3.4 Configure and verify default routing

- **GENERAL CONCEPT**
  - ***Default Route*** is a route a router would use if there is no specific route available
- **STATIC ROUTE CONFIGURATION STEPS**
  - `Router(config)# ip route 0.0.0.0 0.0.0.0 {interface | next-hop}`
- **EIGRP CONFIGURATION STEPS**
  - **Three Methods:**
    - **Option-1**: By advertising a static default route with EIGRP
    - **Option-2**: By configuring a default network
    - **Option-3**: By using the summary-address command
  - **Static Default Route Method:**
    - `Router(config)# ip route 0.0.0.0 0.0.0.0 null0`
    - `Router(config-router)# network 0.0.0.0` **or**
    - `Router(config-router)# redistribute static`
  - **Default Network Method:**
    - A **classful** network advertised into EIGRP and marked with a flag
    - On the router on which all traffic should be directed
      - Identify a classful network that can be advertised into EIGRP
      - Ensure the network is being advertised
    - `Router(config)# ip default-network {network-number}`
  - **Summary-Address Method:**
    - Identify specific interfaces for outgoing default route
    - ONLY the default route will be advertised
    - Default will be locally installed as a summary route
    - `Router(config-if)# ip summary-address eigrp <ASN> 0.0.0.0 0.0.0.0`
- **OSPF CONFIGURATION STEPS**
  - **Two Methods:**
    - **Option-1**: Using the default-information originate command
    - **Option-2**: Using stub areas
  - **Default Information Originate Method:**
    - Used on ASBRs to flood a default route into the entire OSPF domain
    - Only injects the default route as Type 2 using Type 5 LSA with metric 20
      - Default route must exist in the routing table already

- With the **always** parameter…
    - Default route is advertised no matter what
  - `Router(config-router)# default-information originate [always]`
- **Stub Areas Method:**
  - With a stub area, ABR injects a default route into an area
  - In parallel, ABR will not advertise external routes (5 LSA)
  - There are several features of stub areas:
    - ABRs create a default route via Type 3 LSA
    - ABRs do not flood Type 5 LSA into the stub area
    - ABRs may not flood other Type 3 LSAs into the area
    - The default route has a metric of 1 unless otherwise configured
    - `Router(config-router)# area <area-num> default-cost <cost>`
  - Cannot redistribute external routes into the stubby area
  - `Router(config-router)# area <area-num> stub`

- **BGP CONFIGURATION STEPS**
  - **Three Methods:**
    - **Option-1**: Advertise via network command
    - **Option-2**: Redistribution
    - **Option-3**: Default Originate
  - **Advertise via Network Command Method:**
    - Inject default route into BGP only if…
      - Default route is currently present in the routing table
    - `Router(config-router)# network 0.0.0.0`
  - **Redistribution:**
    - Injects default route into BGP only if…
      - Default route is currently present in the routing table
      - Learned by the specific protocol we are redistributing from
    - `Router(config-router)# redistribute <protocol>`
  - **Default Originate:**
    - Two ways, either globally or per neighbor
    - `Router(config-router)# default-information originate <- global`
    - `Router(config-router)# neighbor <ip-addr> default-originate <- neighbor`
- **VERIFICATION**
  - `show ip route`

3.5 Evaluate routing protocol types

3.5.a Distance vector

- **GENERAL CONCEPT**
  - The router has no idea or scope of the entire topology, just the relative costs

3.5.b Link state

- **GENERAL CONCEPT**

- The router has the entire scope of the topology
- The router constructs a map based off the info

3.5.c Path vector

- **GENERAL CONCEPT**
  - The path of the route is maintained, but nothing else
  - **Example**: Which Autonomous Systems did the route traverse through

3.6 Describe administrative distance

Routing Protocol	Administrative Distance
Connected	0
Static	1
EIGRP Summary	5
eBGP	20
EIGRP	90
OSPF	110
RIP	120
External EIGRP	170
iBGP	200
NHRP	250

- **AD A.K.A. *"Administrative Distance"***
  - A value used to rank routes from most preferred to least preferred
  - Used to prevent potential loops by installing a route based on preferred protocol
- **CONFIGURATION STEPS**
  - `Router(config-router)# distance <ad-num>`
- **VERIFICATION**
  - `show ip route`

3.7 Troubleshoot passive interfaces

- **GENERAL CONCEPT**
  - Passive interfaces do not send hello messages
    - Prevents adjacencies on the interface
    - Prevents send/receive routing updates
- **CONFIGURATION STEPS**
  - `Router(config-router)# passive-interface <interface>` **or**
  - `Router(config-router)# passive-interface default`

- **VERIFICATION**
    - **show ip ospf interface**
    - **show ip protocols**

3.8 Configure and verify VRF lite

- **VRF A.K.A.** ***"Virtual Routing and Forwarding"***
    - Multiple routing tables
    - Separation maintained via VRF-to-Interface Allocation
    - Originally designed only for MPLS VPNs with BGP
    - By default, Routing Protocols are only active on the Global Routing Table
- **VRF-LITE INITIAL CONFIGURATION STEPS**
    - `Router(config)# ip vrf Company-A`
    - `Router(config-if)# ip vrf forwarding Company-A`
- **GENERAL VERIFICATION**
    - `show ip vrf <name>`
    - `show ip vrf interfaces <name>`
    - `show ip interface <type/name>`
    - `show ip route vrf <name>`
    - `show ip protocols vrf <name>`
- **VRF-LITE STATIC ROUTE CONFIGURATION STEPS**
    - `Router(config)# ip route vrf <name> <subnet> <mask> {interface | next-hop}`
- **VRF-LITE RIP CONFIGURATION STEPS**
    - `Router(config)# router rip`
    - `Router(config-router)# address-family ipv4 vrf Customer-A`
    - `Router(config-router-af)# network 1.0.0.0`
- **RIP VRF VERIFICATION**
    - `show ip rip database vrf <name>`
- **VRF-LITE EIGRP CONFIGURATION STEPS**
    - `Router(config)# router eigrp 100`
    - `Router(config-router)# address-family ipv4 vrf Customer-A autonomous-system 2`
    - `Router(config-router-af)# network 1.1.1.0 0.0.0.255`
- **EIGRP VRF VERIFICATION**
    - `show ip eigrp vrf <name> topology`
    - `show ip eigrp vrf <name> neighbors`
    - `show ip eigrp vrf <name> interfaces`
- **VRF-LITE OSPF CONFIGURATION STEPS**
    - `Router(config)# router ospf 1 vrf Customer-A`
    - `Router(config-router)# network 1.1.1.0 0.0.0.255 area 1`
    - `Router(config-router)# capability vrf-lite`
    - `Router(config)# router ospf 2 vrf Customer-B ... Must use different IDs`
- **OSPF VRF VERIFICATION**
    - `show ip ospf 1 neighbors`
    - `show ip ospf 1 database`

3.9 Configure and verify filtering with any protocol

- **ACL A.K.A.** ***"Access Control Lists"***

- **General Match Steps:**
  - **Examine-1**: Prefix
  - **Examine-2**: Wildcard Mask
- **Description**:
  - Most commonly used for filtering, such as packet filtering or route filtering
  - Default "`deny any`" at the end of an ACL

- **IP PREFIX-LISTS**
  - **General Match Steps:**
    - **Examine-1:** Prefix and prefix-length (**x.x.x.x/yy**)
    - **Examine-2**: Range of prefixes or range of prefix lengths (**ge** and/or **le**)
  - **Description:**
    - Processing is faster than normal ACLs
    - Default "`deny any`" at the end of a prefix-list
    - **Permit Any:** `ip prefix-list <name> seq 10 permit 0.0.0.0/0 le 32`
- **COMMAND FORMAT**
  - **EIGRP Format:**
    - `distribute-list {ACL | prefix-list | route-map} {in | out} {interface}`
  - **OSPF Format:**
    - `distribute-list {ACL | prefix-list | route-map} {in | out}`
    - `area {number} filter-list prefix {name} {in | out}`
  - **BGP Format:**
    - `neighbor x.x.x.x distribute-list {ACL} {in | out}`
    - `neighbor x.x.x.x filter-list {AS-Path-ACL} {in | out}`
    - `neighbor x.x.x.x route-map {name} {in | out}`
    - `neighbor x.x.x.x prefix-list {name} {in | out}`
- **EIGRP ROUTE FILTERING**
  - **Distribution-List Filter Direction:**
    - **In**: Prevent incoming updates from entering EIGRP topology table
    - **Out**: Prevent routes in routing table from being advertised to neighbors
  - **Extended ACL Details:**
    - Match on prefix and neighbor sending the route
- **OSPF ROUTE FILTERING**
  - **Distribution-List Filter Direction:**
    - **In**: Prevent LSA from becoming a route in own local routing table
    - **Out**: ASBR; Prevents creation of protocols/routes into External LSAs
  - **Filter-List Filter Direction (ABR-only):**
    - **In**: Filter prefixes being created and flooded *into* the configured area
    - **Out**: Filter prefixes coming *out* of the configured area
  - **Extended ACL Details:**
    - Match on prefix and advertised router id of the LSA
- **BGP ROUTE FILTERING**
  - **General Filter Direction:**
    - **In**: Prevent updates from entering BGP table, applied against neighbor
    - **Out**: Prevent best BGP route from being advertised to neighbors
  - **Filter-List Filter Direction (beyond CCNP level):**

- In: Prevents matches of the AS_PATH attribute, applied against neighbor
- Out: Prevents matches of the AS_PATH attribute from being advertised

3.10 Configure and verify redistribution between any protocols or routing sources

- **GENERAL CONCEPT**
  - Redistribution takes the routes from the **routing table** to redistribute
  - Requires at least one working physical link within each routing domain
  - Redistribute *from* the protocol *into* the specified routing domain
- **COMMAND FORMAT**
  - **EIGRP Format:**
    - `redistribute protocol {process-id | as-number} …`
      - `{metric bw delay reliability load mtu} …`
      - `{match {internal | nssa-external | external 1 | external 2}} …`
      - `{tag tag-value} …`
      - `{route-map name}`
    - `default-metric bw delay reliability load mtu`
  - **OSPF Format:**
    - `redistribute protocol {process-id | as-number} …`
      - `{metric metric-value} (default 20 for IGP, 1 for BGP) …`
      - `{metric-type type-value} …`
      - `{tag tag-value} …`
      - `{route-map name} …`
      - `{subnets}`
  - **BGP Format:**
    - `redistribute protocol {process-id | as-number} …`
      - `{metric metric-value} …`
      - `{match {internal | nssa-external | external 1 | external 2}} …`
      - `{route-map name}`
- **COMMAND BREAKDOWN**
  - **protocol**: Source of the routing information
  - **metric**: If default is not set, required information by the protocol
  - **process-id, as-number:** Specify the AS/Process number
  - **match:** If redistributing from OSPF, allows match based on type of OSPF route
  - **tag**: Assigns unitless integer value, can later be matched via route-map
  - **route-map:** Apply logic in the referenced route-map
  - **metric-type**: Defines the external metric type (1 or 2) for the routes redistributed
  - **subnets**: Redistributes subnets of classful networks and must be enabled

3.11 Configure and verify manual and auto-summarization with any routing protocol

- **GENERAL CONCEPT**
  - Summarization is a technique used to reduce the size of a routing table
  - The specific routes are condensed into a larger route

- Manual summarization involves specifying the exact subnet to summarize
- Auto-summarization occurs automatically and only on classful boundaries
- Auto-summary only supports **contiguous** networks
- **Contiguous**: A single classful network end to end
- **Discontiguous**: Multiple classful networks end to end
- Summarization can also be used for path manipulation (more specifics preferred)

- **COMMAND FORMAT**
  - **EIGRP Format:**
    - `**ip summary-address eigrp** {asn} {prefix} {mask} (interface subcommand)`
    - `**auto-summary** (router subcommand)`
  - **OSPF Format:**
    - `**area** {number} **range** {prefix} {mask} {**cost** cost} (ABR only)`
    - `**summary-address** {prefix} {mask} (ASBR only)`
  - **BGP Format:**
    - `**aggregate-address** {prefix} {prefix-length} {**summary-only**}`
    - `**Auto-summary**`
- **EIGRP SUMMARIZATION**
  - Summarization can be performed on *any* router
  - With summarization, query scope is reduced
  - Summary router adds route to routing table with outgoing interface of null0
  - This is to prevent loops in case the summary route comes back around
  - Summary routes by default are given an AD of 5!
  - For auto summarization, network must be local and cannot be redistributed
- **OSPF SUMMARIZATION**
  - OSPF allows summarization at ABRs and ASBRs
  - This is because ALL routers must have the same LSDB
  - Summarization is done for LSAs, not routes
  - By default on ABR, the cost is the best cost among the subordinate subnets
- **BGP SUMMARIZATION**
  - With **auto-summary** and no **mask** statement, the network statement logic...
    - The router adds a route for that classful network to the BGP table
    - The classful route is added if:
      - The exact classful route is in the routing table
      - Any subset routes of that classful network are in the routing table
    - The first occurs regardless of the **auto-summary** settings
    - The second occurs ONLY IF **auto-summary** is configured
  - With aggregation…
    - The prefix/prefix-length is the summary route
    - The **summary-only** is used to not advertise the subordinate routes
    - You need to apply network/redistribute before being able to aggregate

3.12 Configure and verify policy-based routing

- **GENERAL CONCEPT**
  - Ingress packets typically routed via normal routing process

- PBR overrides the router's natural destination-based forwarding logic
- PBR feature is tied to use of Route-Maps
- **Route-Map Purpose:**
    - Define match criteria for PBR packets
    - Define forwarding action for these packets
- **Route-Map Packet Forwarding:**
    - Outgoing interface (should be point-to-point)
    - IP Next-Hop

- **COMMAND FORMAT**
    - `ip policy route-map {name} (interface subcommand)`
    - `ip local policy route-map {name} (global command)`
- **ROUTE-MAP DETAILS**
    - Packets can be matched using a route-map via two criteria:
        - `match ip address {EXTENDED-ACL | STANDARD-ACL}`
        - `match length {MIN-BYTES} {MAX-BYTES}`
    - There are four set command options:
        - `set ip next-hop ip-address [... ip-address]`
        - `set ip default next-hop ip-address [... ip-address]`
        - `set interface interface  [... interface]`
        - `set default interface interface [... interface]`
    - With **default** parameter, attempt to use the "default" routing table first, then PBR!
    - Route-Map can also be used to set **IP precedence** or **ToS bits**:
        - `set ip precedence {value}`
        - `set ip tos {value}`

3.13 Identify suboptimal routing

- **GENERAL CONCEPT**
    - AD is used to determine best route which can result in suboptimal paths
    - This occurs during redistribution, and if not properly designed, can result in loops
    - EIGRP automatically prevents loops by setting AD of external routes higher (170)
    - Routing loops can also be prevented using tags and route-maps
- **MODIFY ADMINISTRATIVE DISTANCE**
    - `distance distance ip-adv-router wc-mask [acl-number-or-name]`

3.14 Explain ROUTE maps

- **GENERAL CONCEPT**
    - Route-maps provide if/then/else logic like in programming languages
    - A route-map can call an ACL or prefix-list
    - Route-maps also has the concept of using sequence numbers
    - A single route-map contains one or more route-map commands (entries)
    - To match all packets, omit the **match** command
- **COMMAND FORMAT**
    - `route-map {name} {permit | deny} {seq-num}`
    - `match <match criteria>`

- **set** <set criteria>

3.15 Configure and verify loop prevention mechanisms

3.15.a Route tagging and filtering

- **GENERAL CONCEPT**
  - A route tag is a unitless 32-bit integer that most routing protocols can assign
  - Assigned by a route-map **set** reference or the distribute-list/redistribute command
  - By tagging redistributed routes, you can then prevent loops easily
- **LOOP PREVENTION EXAMPLE**
  - `Router(config-router)#` **`redistribute eigrp`** `100` **`tag`** `77` **`subnets`**
  - `Router(config)#` **`route-map`** `stop-loops` **`deny`** `10`
  - `Router(config-route-map)#` **`match tag`** `77`
  - `Router(config)#` **`route-map`** `stop-loops` **`permit`** `20`
  - `Router(config-router)#` **`distribute-list route-map`** `stop-loops` **`in`**

3.15.b Split-horizon

- **GENERAL CONCEPT**
  - Distance vector protocols are susceptible to routing loops
  - Prevents a route received on an interface to flood back out the same way
  - In NBMA networks with hub and spoke topologies, this will cause issues
- **DISABLING SPLIT HORIZON**
  - **`no ip split-horizon`** `{`**`eigrp`** `{AS}} (interface subcommand)`

3.15.c Route poisoning

- **GENERAL CONCEPT**
  - Advertises a route with a “bad” metric
  - This enables receiving router to remove the route from its routing table
- **SPECIFIC DETAILS**
  - EIGRP sends an infinite metric
  - OSPF sends an LSA age metric of 3600 seconds
  - RIP sends a metric of 16 hops

3.16 Configure and verify RIPv2

- **CONFIGURATION STEPS**
  - `Router(config)#` **`router rip`**
  - `Router(config-router)#` **`version 2`**
  - `Router(config-router)#` **`network`** `x.x.x.x`
- **VERIFICATION**
  - **show ip route**
  - **show ip rip database**

3.17 Describe RIPng

- **COMMONALITIES**
  - Uses UDP
  - Distance vector routing protocol
  - AD of 120
  - VLSM is supported
  - Uses split horizon rule
  - Uses poison reverse
  - 30-second periodic full update
  - Uses triggered updates
  - Uses hop count metric
  - Metric of 16 as infinity
  - Supports route tags
  - Multicast update destination
  - Does not form neighborships
- **DIFFERENCES**
  - UDP port of 521 for RIPng, RIPv2 uses 520
  - No auto-summarization for IPv6
  - Destination address of FF02::9
  - Link-Local Next-Hops
  - IPv6 uses IPv6 AH/ESP authentication
- **CONFIGURATION STEPS**
  - `Router(config)#` **`ipv6 unicast-routing`**
  - `Router(config)#` **`ipv6 router rip`** `name`
  - `Router(config-if)#` **`ipv6 rip`** `name` **`enable`**
- **VERIFICATION**
  - **`show ipv6 route`**
  - **`Show ipv6 protocol`**
  - **`show ipv6 rip next-hops`**

3.18 Describe EIGRP packet types

- **EIGRP A.K.A.** ***"Enhanced Interior Gateway Routing Protocol"***
  - **Uses DUAL A.K.A.** ***"Diffusing Update Algorithm"*** **to…**
    - Determine the best loop-free path
    - Determine the backup loop-free path
    - Provide FAST convergence
  - **Uses bandwidth and delay for metric calculation…**
    - `BW = (10^7/BW)*256, DELAY = (Delay In Microseconds) * 256`
    - `Metric = (K1*BW+((K2*BW)/(256-load))+K3*delay)*(K5/(reliability+K4))`
    - **`Real (Default) Metric = (Slowest_BW + All_Link_Delays)`**
  - **Uses hello packets to determine neighbors…**
    - Used to dynamically discover new neighbors
    - Used to maintain the neighbor relationship
  - **Uses three types of tables…**

        - **Topology Table**: Contains all successor and feasible successors
        - **Routing Table**: Contains only the successors
        - **Neighbor Table**: Contains neighborship information
- **SUCCESSOR AND FEASIBLE SUCCESSOR ROUTES**
    - **Feasible Distance**: Metric for a route to choose the best route for the prefix
    - **Reported Distance**: Metric for a route the neighbor is *reporting* for the prefix
    - The route with the smallest feasible distance is elected as primary route
    - **Successor Route:** The primary route
    - **Feasible Successor Routes:** The elected backup routes
    - **Feasibility Condition**: Route's RD < FD of the successor
    - The Feasibility Condition exists to prevent potential loops
- **HELLO PACKETS**
    - By default, multicast address used is 224.0.0.10
    - Unicast will be used for static neighbors, useful with NBMA topologies
    - **Neighborship Requirements:**
        - Interface primary IP address must be in the same subnet
        - AS number must be the same
        - Connected interface must not be passive
        - Authentication must pass
        - K-values used must match
    - **Default Hello And Dead Timers:**
        - **T1 or Slow Links**: 60 seconds by default
        - **LAN or Fast Links**: 5 seconds by default
        - **Modify**: `ip hello-interval eigrp {as} {interval} (interface subcommand)`
- **UPDATE PACKETS**
    - When neighbors come up, the routers exchange full topology tables
    - Once done, there is **NO** periodic reflooding of the information
    - If something changes, only a partial update is sent about the affected prefix
    - This change can be as small as a metric value change or about link failures
    - Uses RTP A.K.A. "Reliable Transport Protocol" to send updates/ACK messages
- **ACK PACKETS**
    - Used during the topology exchange alongside update packets
    - Generally ACK is sent in unicast by the neighbor
- **QUERY/REPLY PACKETS**
    - When a route has failed (active), a Query message is sent to all neighbors
    - Query messages need an **exact** prefix/length for the Reply
    - Queries are stopped on the routers that are one-hop from summarization
    - **If a neighbor receiving the Query message has a loop free route...**
        - Reply message is sent in response
        - Otherwise, the Query is forwarded on to its neighbors
    - **Once original router has received a Reply from all neighbors…**
        - If new loop-free routes were learned, install the best as successor
        - If no routes are learned, the active prefix is removed from topology table
    - **Active Timer = 3 minutes**

        - Once 3 minute timer was up, the router will cut neighbor relationships
        - This was an issue with neighbors sending late query replies
    - **By default, after 90 seconds, routers send SIA-Query to neighbor…**
        - SIA A.K.A. ***"Stuck In Active"***
        - Condition: Neighbor is still alive but waiting for a response to OWN query
        - The neighbor in SIA will send an SIA-Reply
        - If neighbor does not respond, active timer continues to decrement
    - **This timer can be configured using the following command:**
        - `timers active time {time} (router subcommand)`
    - **You can limit Query Scope:**
        - Stub Routers
        - Route Summarization

3.19 Configure and verify EIGRP neighbor relationship and authentication

- **NEIGHBOR CONFIGURATION EXAMPLE**
    - `Router(config)# interface Ethernet1/1`
    - `Router(config-if)# ip address 10.0.0.1 255.255.255.0`
    - `Router(config)# router eigrp 100`
    - `Router(config-router)# network 10.0.0.0 0.0.0.255`
- **NEIGHBOR VERIFICATION**
    - `show ip eigrp topology`
    - `show ip eigrp neighbors`
    - `show ip protocols`
- **AUTHENTICATION**
    - Must pass for neighbor to establish
    - Routers should use the same pre-shared key
    - EIGRP only supports MD5 type authentication, not clear text
    - These packets can still be intercepted by man-in-the-middle attacks
- **AUTHENTICATION CONFIGURATION EXAMPLE**
    - `Router(config)# key chain TEST`
    - `Router(config-keychain)# key 1`
    - `Router(config-keychain-key)# key-string CISCO`
    - `Router(config)# interface gi0/1`
    - `Router(config-if)# ip authentication mode eigrp 1 md5`
    - `Router(config-if)# ip authentication key-chain eigrp 1 TEST`
- **ADVANCED AUTHENTICATION OPTIONS**
    - Keychain can also be configured to use time-based logic
    - Multiple keys can be configured for different time periods
    - If there are multiple active/valid keys at the same time, lowest key number wins
    - Clocks should match on both routers, use NTP to update clocks if required
    - Clocks can also be set locally using the "clock set" command in exec mode
    - **Time Commands:**
        - `send-lifetime hh:mm:ss (in key configuration mode)`
        - `accept-lifetime hh:mm:ss (in key configuration mode)`

- **AUTHENTICATION VERIFICATION**
    - `show key chain name`
    - `show clock`
    - `debug eigrp packets`

3.20 Configure and verify EIGRP stubs

- **STUB ROUTER**
    - Should not forward traffic between two remote EIGRP-learned subnets
    - Does not advertise EIGRP-learned routes from one neighbor to another
    - Non-stub routers one-hop away from stub **do not** forward queries to stub
    - By default, EIGRP stub routers only advertises:
        - Connected
        - Summary
    - A stub router can be configured to advertise redistributed, static routes, or mix
    - The "`receive-only`" parameter means the stub will not advertise ***any*** prefix
- **CONFIGURATION STEPS**
    - `Router(config-router)# eigrp stub receive-only`
- **VERIFICATION**
    - `show ip protocols`

3.21 Configure and verify EIGRP load balancing

3.21.a Equal cost

- **GENERAL CONCEPT**
    - By default, equal cost/metric routes are installed in the routing table
    - To change the maximum number of equal cost routes that can be installed:
        - `maximum-paths {number} (router subcommand)`
    - As for whether load balancing is per packet or per session, CEF decides
    - Generally, per flow is enabled by default
- **VERIFICATION**
    - `show ip route`
    - `show ip eigrp topology`

3.21.b Unequal cost

- **GENERAL CONCEPT**
    - **Variance:** Allows EIGRP to consider additional paths for load sharing
    - These paths only need to be similar to the successor route
    - By default, variance is set to 1 which means only equal paths are load balanced
    - The variance multiplier is an integer between 1 and 128
    - **ONLY** feasible successor routes are considered in the unequal load balancing!
    - This means that if a route did not meet feasibility condition, it is skipped
    - **Feasible successor considered if…**

- `(Variance)*(Successor Route's FD) > Feasible Successors FD`

- **CONFIGURATION STEPS**
  - `variance {multiplier} (router subcommand)`
- **VERIFICATION**
  - `show ip protocols`

3.22 Describe and optimize EIGRP metrics

- **GENERAL CONCEPT**
  - EIGRP metric can be manipulated to prefer one path over the other
  - There are two ways to manipulate the EIGRP metric:
    - Changing the bandwidth and/or delay values
    - Using offset-list
  - Changing bandwidth value is **NOT** recommended
    - Other technologies are affected such as QoS
    - On WAN subinterfaces, bandwidth value should be chosen carefully
    - Bandwidth on these interfaces should be sum of all CIR
  - Recommended to change the delay value instead
  - EIGRP distance calculation can be changed by configuring the k-values
- **CONFIGURATION STEPS**
  - `Router(config-router)# metric weights {tos} k1 k2 k3 k4 k5`
  - `Router(config-if)# bandwidth {amount}`
  - `Router(config-if)# delay {amount}`
- **VERIFICATION**
  - `show ip protocols`
- **OFFSET-LIST CONCEPT**
  - Offset-list simply allow a value to be added to the EIGRP metric
  - Behind the scenes, offset-list modifies the delay metric for the specified routes
  - The offset value is added to both the FD and the RD
  - Direction of the update message is also specified
- **OFFSET-LIST CONFIGURATION STEPS**
  - `offset-list {ACL} {in | out} <offset value> {interface} (Router Subcommand)`

3.23 Configure and verify EIGRP for IPv6

- **COMMONALITIES**
  - Uses Layer 3 header protocol type of 88
  - Uses successor, feasible successor logic
  - Uses DUAL
  - Uses triggered updates
  - Uses composite metric
  - Metric meaning infinity is 2^32 - 1
- **DIFFERENCES**
  - Uses neighbor's link local address as next-hop IP
  - Destination address is FF02::A

- Authentication relies on IPv6 built-in authentication and privacy features
  - IPv6 has no concept of classful networks, so no auto-summary feature
  - Does not require neighbors to be in the same IPv6 subnet to become neighbors
- **CONFIGURATION STEPS**
  - `Router(config)#` **`ipv6 unicast routing`**
  - `Router(config)#` **`ipv6 router eigrp`** `100`
  - `Router(config-router)#` **`eigrp router-id`** `1.1.1.1`
  - `Router(config-if)#` **`ipv6 eigrp`** `100`
- **VERIFICATION**
  - **`show ipv6 route`**
  - **`show ipv6 protocols`**
  - **`show ipv6 eigrp neighbors`**
  - **`show ipv6 eigrp interfaces details`**
  - **`show ipv6 eigrp topology [all-links]`**
  - **`debug ipv6 eigrp notifications`**

3.24 Describe OSPF packet types

- **OSPF A.K.A.** ***"Open Shortest Path First"***
  - **Uses link state logic, which can be broken into three branches:**
    - Neighbor discovery
    - Topology database exchange
    - Route computation
  - **Uses concept of areas and a hierarchical design**
    - All areas must connect into the backbone (area 0)
    - ABR connect non-backbone areas to the backbone area
    - Detailed topology information not exchanged between areas
  - **All internal routers must have the same image of the network**
    - Link State Databases (topology database) must be the same
    - Shortest Path First (SPF) is run on the LSDB to find best path
  - **Has two neighborship classes:**
    - 2-Way Neighbors: Not exchanged topology information
    - Fully Adjacent Neighbors: Link state databases matches
- **ROUTER-ID ASSIGNMENT PROCESS**
  - *Same as EIGRP*
  - Manually configured router-id preferred over…
  - Highest IP address of any up/up loopback interface preferred over…
  - Highest IP address of any up/up non-loopback interface
  - Not preemptive
- **SAME OSPF ROUTER-ID**
  - **Same Router-ID in the same area:**
    - The routers will generate a message saying there's a duplicate router-id
  - **Same Router-ID in a different area:**
    - The routers will flush each other's LSAs and declare an OSPF Flood War
- **OSPF PACKETS/MESSAGES**
  - **Hello**: Used to form and maintain neighborship

- **Database Description (DD):** Used to exchange header information of LSAs
- **Link-State Request (LSR)**: Used to request for missing LSAs
- **Link-State Update (LSU)**: Used to send missing LSA information to neighbor
- **Link-State Acknowledgement (LSAck)**: Used to ack received updates

- **HELLO MESSAGES**
    - Messages are sent using the multicast address 224.0.0.5
    - **The following parameters must match:**
        - Hello Interval
        - Dead Interval
        - Area ID
        - Subnet Mask
        - Stub Area Flag
        - Authentication
    - **Additional parameters that may be present:**
        - OSPF Router ID
        - List of neighbors reachable on interface
        - Router Priority
        - Designated Router (DR) IP Address
        - Backup DR IP Address
    - **Change Hello/Dead Intervals:**
        - `ip ospf hello-interface {value} (interface subcommand)`
        - `ip ospf dead-interval {value (interface subcommand)`
    - **Verify Intervals:**
        - `show ip ospf interface {interface}`
    - **Subsecond Interval Modification:**
        - `Router(config-if)# ip ospf dead-interval minimal hello-multiplier {x}`
- **OSPF FLOODING**
    - Routers advertise their local and learned LSAs to all OSPF neighbors
    - Flooding prevents the looping of LSA as a side-effect of the DD exchange
    - Re-flooding occurs every 30 minutes based on each LSA's age variable
    - Router that creates the LSA sets this age to 0 seconds
    - When a router needs to flush LSA from LSDB, Max Age is set to 3600 seconds

3.25 Configure and verify OSPF neighbor relationship and authentication

- **DATABASE EXCHANGE WITHOUT DR**
    - **STEP-1_INIT**
        - Router sends a hello packet on its OSPF enabled link
    - **STEP-2_2-WAY**
        - Router receives a Hello packet
        - Sees its own RID as having been seen by the neighbor
        - Also sees all parameters passing
    - **STEP-3_EXSTART**

- First **Database Descriptor (DD)** received
- DD gives high level overview of the LSAs
- Also, an election is held for master and slave router
- The master is in charge of the initial sequence number of DD
- The router with the highest RID is elected as Master
- DD packets are sent using unicast

- STEP-4_EXCHANGE
    - Occurs after master/slave election
    - Neighbors continue to multicast DD packets to each other
    - This continues until they all have the same LSIDs known in the area
- STEP-5_LOADING
    - The routers have the same view of all LSIDs
    - For any missing LSA, router sends a **Link State Request (LSR)**
    - The router listening to LSR sends a **Link State Update (LSU)**
    - Routers sends LSAcks or implicit ack by returning same LSA
- STEP-6_FULL
    - All LSAs have been sent, received, and acknowledged
    - Database is fully populated
    - Routers run SPF to calculate the best paths for each subnet
- **Additional neighbor states (8 Total)**
    - **Down**: No neighborship with that router
    - **Attempt**: Layer 2 problem with statically defined neighbors

DATABASE EXCHANGE WITH DR

- Only slight differences
- All the other routers exchange their databases with DR/BDR only
- Communication to DR/BR occurs over 224.0.0.6
- Communication from DR/BDR still occurs over 224.0.0.5

NEIGHBOR CONFIGURATION DETAILS:

- **To advertise any particular subnet or enable any interface:**
    - `network {subnet} {wildcard-mask} area {area-number} (router subcommand)`
    - `ip ospf {process-id} area {area-number} (interface subcommand)`
- **To change the OSPF router-id:**
    - `router x.x.x.x (router subcommand)`

NEIGHBOR VERIFICATION

- `show ip ospf neighbor`
- `show ip ospf`
- `show ip ospf database`
- `show ip protocols`

OSPF AUTHENTICATION CONCEPT

- **Configuring authentication is a two-step process:**
    - Authentication and authentication type must be enabled and selected
    - Authentication key must be configured per interface
- **This can be done in one of two ways:**
    - Enable at the interface
    - Enable on all interfaces in an area by changing the area wide settings

        - Any authentication on an interface will override area wide settings
    - **Authentication Key**
        - Cannot be configured area wide
        - Must be configured individually on each interface
    - **Three types of authentication:**
        - **Type-0**: No Authentication
        - **Type-1**: Clear Text Authentication
        - **Type-2**: MD5 Authentication
    - **Number of keys**
        - Multiple keys can be configured on the same interface
        - OSPF does not support the use of a keychain
        - This can only be done if MD5 authentication is enabled
        - Useful for migration of keys
- **AUTHENTICATION CONFIGURATION DETAILS:**
    - **Enable Authentication - Area Wide Settings (router subcommand)**
        - `area {area-number} authentication (type 1)`
        - `area {area-number} authentication message-digest (type 2)`
    - **Enable Authentication - Interface Settings (interface subcommand)**
        - `ip ospf authentication null (type 0)`
        - `ip ospf authentication (type 1)`
        - `ip ospf authentication message-digest (type 2)`
    - **Configure Key (interface subcommand)**
        - `ip ospf authentication-key {key-value} (type 1)`
        - `ip ospf message-digest-key {key-number} md5 {key-value} (type 2)`
- **AUTHENTICATION VERIFICATION**
    - `show ip ospf interface {interface}`
    - `debug ip ospf hello`
    - `debug ip ospf adj`

3.26 Configure and verify network types, area types, and router types

3.26.a Point-to-point, multipoint, broadcast, nonbroadcast

Network Type	DR/BDR?	Hello Unicast/Multicast?	Hello/Dead Intervals?
Point-to-Point	No	Multicast	10/40
Point-to-Multipoint	No	Multicast	30/120
Point-to-Multipoint Non-Broadcast	No	Unicast	30/120
Broadcast	Yes	Multicast	10/40
Non-Broadcast	Yes	Unicast	30/120

Loopback	No	N/A	N/A

- **OSPF NETWORK TYPES**
  - Determines operational characteristics of OSPF on that interface
    - Whether the router will use multicast to discover neighbors
    - If at least two routers can exist in the subnet attached to interface
    - Whether the router should attempt to elect an OSPF DR on that interface
  - Determined automatically by Layer-2 encapsulation on the interface
- **NETWORK TYPE BROADCAST**
  - **Broadcast is default on LAN interfaces:**
    - Discovers neighbors dynamically
    - Supports use of DR/BDR
    - Intervals: 10 & 40
  - **Configuration Details:**
    - `ip ospf network broadcast` (interface subcommand)
- **NETWORK TYPE NON-BROADCAST**
  - **Default on Frame Relay / NBMA physical or multipoint interfaces:**
    - Does NOT discover neighbors dynamically
    - Intervals: 30 & 120
  - **Statically Configure Neighbors:**
    - `neighbor ip-address {priority priority}` (router subcommand)
  - **Configuration Details:**
    - `ip ospf network non-broadcast` (interface subcommand)
  - If network is not full mesh, make sure the DR/BDR can reach everyone
- **NETWORK TYPE POINT-TO-POINT**
  - **Default on any point-to-point interface:**
    - Does not elect DR/BDR
    - Discovers neighbors dynamically
    - Intervals: 10 & 40
  - **Configuration Details:**
    - `ip ospf network point-to-point` (interface subcommand)
- **NETWORK TYPE POINT-TO-MULTIPOINT**
  - **Not default on any kind of interface:**
    - Does not elect DR/BDR
    - Discovers neighbors dynamically
    - Intervals: 30 & 120
  - **How does this network type help with partial mesh topologies?**
    - Regardless of actual mask, router advertises /32 LSAs for connectivity
    - LSAs received on P-2-MP subinterface allowed to be flooded back out
    - Routers without direct PVC connectivity can still achieve L3 connectivity
  - **Configuration Details:**
    - `ip ospf network point-to-multipoint` (interface subcommand)
- **NETWORK TYPE POINT-TO-MULTIPOINT NON-BROADCAST**
  - **Not default on any kind of interface:**

            - Similar to point-to-multipoint
            - Neighbors will not be discovered dynamically
            - Not described in RFC 2328
        - **Advantages over other types?**
            - Cost to reach each neighbor can be changed per neighbor
            - By default, changing cost on an interface changes cost for all neighbors
        - **Configuration Details:**
            - `ip ospf network point-to-multipoint non-broadcast` (interface subcommand)
- **HOW TO REMEMBER IT ALL**
    - If the network type starts with "point" it does not use DR/BDR
    - If network type has "non-broadcast", it does not do dynamic neighbors
    - Only broadcast and point-to-point use faster times

3.26.b LSA types, area type: backbone, normal, transit, stub, NSSA, totally stub

- **LSA GENERAL CONCEPT**
    - Each router stores data, composed of individual **link-state advertisements**
    - Each router in an area will have the same link-state database information
    - Each router, individually, runs the Shortest Path First (SPF) algorithm
    - Each router considers itself to be the root of the tree
    - SPF process must examine the individual LSAs and see how they fit together
- **LSA TYPE OVERVIEW**
    - **Type-1**: Router LSA
    - **Type-2**: Network LSA
    - **Type-3**: Network Summary LSA
    - **Type-4**: ASBR Summary LSA
    - **Type-5**: AS External LSA
    - **Type-7**: NSSA External LSA
- **TYPE-1 LSA: ROUTER LSA**
    - Each router creates a Type-1 LSA for itself and floods it in the area
    - The LSA identifies an OSPF router based on its OSPF router-id
    - What does it accomplish? Helps to build SPF tree
    - OSPF identifies a Type-1 using a 32-bit **link-state identifier (LSID)**
    - Each router uses its own OSPF router-id as the LSID
    - ABRs create multiple Type-1 LSAs for themselves, one per area
    - **This LSA also includes information about its attached links.**
        - **No neighbors:** Subnet advertised; link to 'a stub network'
        - **With DR:** The IP of the DR; link to 'a transit network'
        - **No DR:** It lists the neighbor's RID; link to 'another router (point-to-point)'
    - **Router LSA Verification:**
        - `show ip ospf database router x.x.x.x`
- **TYPE-2 LSA: NETWORK LSA**
    - Generated on multi-access networks
    - Depends on the existence of a DR

- Required to map all connected routers on multi-access network, like LAN
  - Type-2 LSA also lists the RIDs of the OSPF neighbors connected to the interface
  - Network type tells the router's interface whether a DR should be elected
  - **Routers uses the following election rules to elect a DR:**
    - Choose a router with the highest priority value (default 1, max 255)
    - If priority is tied, choose the router with highest RID
    - Choose a BDR, based on next-best priority, if tied, next-best highest RID
  - **Change Priority:**
    - `ip ospf priority value (interface subcommand)`
  - When a DR fails, the current BDR becomes the new DR
  - Election is held only for the BDR, not the DR
  - **Network LSA Verification:**
    - `show ip ospf database network x.x.x.x`
- **TYPE-3 LSA: NETWORK SUMMARY LSA**
  - **OSPF uses concept of areas to reduce memory and compute resources**
    - Type-1 and 2 LSAs are not advertised across areas
    - This saves CPU and convergence time
  - ABR A.K.A. ***"Area Border Routers"***
    - Connect different OSPF areas together
    - Generate Type-3 LSAs for each subnet in an area
  - **Type-3 LSA does not contain all the detailed topology information**
    - Consists of each subnet and cost to reach that subnet
    - Summarizes the information from Type-1 and Type-2 LSAs
  - Type-3 LSAs appear to be another subnet connected to the ABR to other routers
  - ABR assigns an LSID of the subnet being advertised
  - ABR also adds its own RID in the LSA as there could be multiple ABRs
  - **Summary LSA Verification:**
    - `show ip ospf database summary x.x.x.x`
- **LIMITING NUMBER OF LSA**
  - By default, there is no upper limit for the number of LSAs that a router can learn
  - As the LSDB grows, memory and convergence becomes an issue
  - To limit the number of LSAs, use the following command:
    - `max-lsa number (router subcommand)`
  - Not recommended, as once LSA count goes over limit all relationships die
- **TYPE-4 LSA: ASBR SUMMARY LSA**
  - ASBR A.K.A. ***"Autonomous System Border Router"***
  - How do the other areas know how to get to the ASBR?
  - ABR generates a Type-4 LSA
  - This tells the area routers that to get to the ASBR, to go through the ABR
  - "I am your ABR and you can reach the ASBR through ME!"
  - A new Type-4 LSA must be generated by the ABR for every new area
- **TYPE-5: AS EXTERNAL LSA**
  - Generated by ASBRs
  - Type-5 LSA floods every area, no other LSA does this
  - Type-4 provides reachability to ASBR for the Type-5 external routes

- Please see the filtering section 3.9 for further details
- **TYPE-7: NSSA EXTERNAL LSA**
  - Generally, external routes cannot be injected to stub areas
  - This means routers within these areas cannot do redistribution
  - Not-so-stubby areas (NSSA) solves this problem with Type-7 LSA
  - Routers in NSSA areas can redistribute routes as Type-7 LSA
  - Type-5 LSAs are still not allowed
  - Type-7 converts to Type-5 LSA once it reaches the ABR
- **GENERAL AREA NOTES**
  - **Backbone Area**
    - All areas must connect into the backbone
    - ABRs connect non-backbone areas to the backbone
    - Virtual-links can get around this limit, though not recommended design
  - **Normal Area**
    - An area that operates independently
    - Able to receive Type-3 and Type-5 LSAs from other areas
  - **Transit Area**
    - An area that connects the backbone area to another area
    - Used for virtual-link setups
    - This area **MUST NOT** be stub
- **STUBBY AREA CONCEPT**
  - **Four types of OSPF stubby areas:**
    - Stub
    - Totally Stubby
    - Not-So-Stubby Areas (NSSA)
    - Totally NSSA
  - **Differences in types of Stubby Areas**
    - For all types of stubby areas, ABR always filters Type-5 LSAs
    - For totally stubby and totally NSSA areas, ABR also filters Type-3 LSAs
    - For stubby and NSSA, ABRs do not filter Type- LSAs
- **CONFIGURATION STEPS FOR STUBBY AREAS**
  - To configure an area as a stub area, use the following command:
    - `area area-id stub (router subcommand)`
  - All routers in area should be configured the same way!
  - The metric for the default route injected by ABRs can be changed from default 1:
    - `area area-id default-metric metric (router subcommand)`
  - To configure an area as totally stubby, use the following command:
    - `area area-id stub no-summary (router subcommand)`
  - The no-summary option is only required on the ABR
- **CONFIGURATION STEPS FOR NSSA AREAS**
  - ABRs do not advertise, by default, a default-route into NSSA areas:
    - `area area-id nssa default-information-originate (router subcommand)`
  - To configure an area as NSSA, use the following command:
    - `area area-id nssa (router subcommand)`
  - To configure an area as totally NSSA, use the following command:

- **area** area-id **nssa no-summary** (router subcommand)
- The no-summary option is only required on the ABR
- **STUB VERIFICATIONS**
  - **show ip ospf**
  - **show ip ospf database**
  - **show ip ospf database database-summary**

3.26.c Internal router, backbone router, ABR, ASBR

- **GENERAL NOTES**
  - **Internal Router**: All routers within a single area, not including ABR
  - **Backbone Router**: All routers within the backbone, including ABRs
  - **ABR**: Router that connects a normal area to the backbone area
  - **ASBR**: Router that connects an external autonomous system to the domain

3.26.d Virtual link

- **GENERAL CONCEPT**
  - OSPF requires all non-backbone areas connect to the backbone area
  - With some designs, this is not possible!
  - This introduces a reachability problem
  - Virtual Link allows two ABRs to form a transit area
    - This allows an area to connect *through* another non-backbone area
    - This makes the disjointed area believe it is connected to area 0 directly
  - ABRs connected over virtual links send all OSPF messages as unicast
  - ABRs connected over virtual links also mark the *Do Not Age (DNA)* bit in LSAs
  - The transit area **MUST NOT** be a stub area
- **CONFIGURING VIRTUAL LINKS WITHOUT AUTHENTICATION**
  - To configure a virtual link between ABRs, use the following command:
    - **area** area-number **virtual-link** remote-RID (router subcommand)
  - The area-number must refer to the transit area number
  - The remote-RID can be verified from the local router using the following:
    - **show ip ospf border-routers**
  - To verify OSPF virtual link:
    - **show ip ospf virtual-links**
- **CONFIGURING VIRTUAL LINKS WITH AUTHENTICATION**
  - Authentication key is not defined on the interface level but otherwise same
  - To configure authentication on virtual links, use the following:
    - **area** area-number **virtual-link** remote-RID **authentication...**
      - **null** OR
      - **authentication-key** key-value OR
      - **message-digest message-digest-key** key-number **md5** key-value

3.27 Configure and verify OSPF path preference

- **GENERAL CONCEPT**
  - **Multiple ways to do path manipulation:**

        - By changing metric
        - By using summary route (Section 3.11)
        - By filtering (Section 3.9)
        - By changing metric-type (Section 3.10 - can also be done with default)
    - Within an OSPF area, options are limited; the only way is by changing metric
    - **Metric can be changed by…**
        - Changing the `auto-cost reference bandwidth`
        - Changing the bandwidth of an interface
        - Changing OSPF cost directly at the interface level
- **PATH MANIPULATION CONCEPT**
    - With multiple areas, common network designs have at least two ABRs
    - Ideally, both ABRs advertise the same subnets from one area to another
    - To prefer one ABR over another, we can use summarization or filtering
    - Filtering is not recommended as there is no high availability
    - OSPF Route Hierarchy:
        - **O** > **O IA** > **E1** > **E2** > **N1** > **N2**
- **METRIC TUNING CONFIGURATION STEPS**
    - OSPF calculates the cost for an interface based on the following formula:
        - `reference-bandwidth / interface-bandwidth`
    - Default reference-bandwidth is 100 Mbps
    - **The reference bandwidth can be changed by using:**
        - `auto-cost reference-bandwidth bandwidth (router subcommand)`
    - This setting is local to the router, but it should be changed on all area routers
    - **Bandwidth can be changed directly with the following command:**
        - `bandwidth value (interface subcommand)`
    - Changing bandwidth is not recommended as it will affect other dependencies
    - **Cost can also be directly changed on the interface:**
        - `ip ospf cost value (interface subcommand)`
    - **To verify OSPF cost on any interface:**
        - `show ip ospf interface brief`
        - `show ip ospf interface`

3.28 Configure and verify OSPF operations

- **CONFIGURATION STEPS**
    - **STEP-1**: Start up OSPF process and define router-id
        - `Router(config)# router ospf 1`
        - `Router(config-router)# router-id 1.1.1.1`
    - **STEP-2**: Place interfaces in appropriate areas
        - `Router(config-router)# network 10.0.0.0 0.0.0.255 area 0 OR`
        - `Router(config-if)# ip ospf 1 area 0`
    - **STEP-3**: Configure any miscellaneous parameters
        - Make sure every interface is in right network type
        - Make sure to enable any edge scenarios such as virtual link
        - See the previous sections for any actual details
- **VERIFICATION**

- `show ip ospf neighbor`
- `show ip ospf database`
- `show ip ospf interface brief`
- `show ip protocols`
- `show ip route`

3.29 Configure and verify OSPF for IPv6

- **COMMONALITIES**
    - Use IP protocol type 89
    - Use link state logic
    - Supports VLSM
    - LSA flooding and aging concepts
    - Area structure concept
    - Packet types
    - 32-bit LSID
    - Uses interface cost metric
    - Age = 3600 as the infinity metric
    - DR/BDR election process
    - Periodic reflooding every 30 minutes
- **DIFFERENCES**
    - Multicast address of FF02::5, FF02::6
    - OSPFv3 uses IPv6 AH/ESP authentication
    - Neighbor checks are the same except that “same subnet” is not checked for IPv6
    - IPv6 OSPF supports multiple instances basically VRFs (not supported in IOS)
- **LSA TYPES**
    - The first two LSAs are only responsible for building SPF trees
    - Now, Type-9 contains the actual subnet information
    - Type-9 makes SPF recalculation not so disruptive anymore
    - New Type-8 link LSA goes to next-hop neighbors to provide link local info
    - The DR can send that info to connected neighbors and the area with Type-9

LSA Name	LS Type Code	Flooding Scope	LSA Function Code
Router LSA	0x2001	Area Scope	1
Network LSA	0x2002	Area Scope	2
Inter-Area-Prefix-LSA	0x2003	Area Scope	3
Inter-Area-Router-LSA	0x2004	Area Scope	4
AS-External-LSA	0x4005	AS Scope	5
Group-membership-LSA	0x2006	Area Scope	6
Type-7-LSA	0x2007	Area Scope	7

Link-LSA	0x0008	Link-local scope	8
Intra-Area-Prefix-LSA	0x2009	Area scope	9

- CONFIGURATION STEPS
  - **General Configuration:**
    - `Router(config)# ipv6 unicast-routing`
    - `Router(config)# ipv6 router ospf <process-id>`
    - `Router(config-if)# ipv6 ospf <process-id> area <#>`
  - The concept and commands related to OSPF stub areas are identical
  - **Summarize on ABR and ASBRs using the following:**
    - `area <#> range prefix/length (router subcommand)`
  - **Change Network Types:**
    - `ipv6 ospf network type (interface subcommand)`
- VERIFICATION
  - `show ipv6 route`
  - `show ipv6 ospf`
  - `show ipv6 protocols`
  - `show ipv6 ospf neighbors`
  - `show ipv6 ospf interfaces brief`
  - `show ipv6 ospf database`

3.30 Describe, configure, and verify BGP peer relationships and authentication

- COMPARISON BETWEEN IGP AND BGP
  - BGP needs to form neighborship like IGPs
  - BGP needs to advertise prefixes, just like IGPs
  - BGP also advertises Next Hops for those prefixes
  - Neighbor IP address may not be a common subnet for BGP
  - BGP uses TCP while IGP uses multicast
  - IGPs emphasize fast convergence for efficient routes, BGP scalability
  - BGP uses path vector logic which is similar to distance vector
  - **NLRI A.K.A.** ***"Network Layer Reachability Information"***
    - Advertised prefix and length
    - Known as NLRI as it can transport beyond just IPv4 addressing
    - BGP advertises different path attributes for path making decisions
- GENERAL BGP CONCEPT
  - **Two types of neighbors in BGP: internal and external**
    - Neighborship behavior is different between the two types
    - As everything is unicast, you can have neighbors not directly connected
  - **There are two kinds of AS numbers: public and private**
    - Public AS numbers can be advertised over the Internet
    - Private AS numbers should not be advertised over the Internet
- eBGP NEIGHBORSHIP OVERVIEW
  - **BGP must complete three steps to get best routes:**

        - Form neighborship
        - Exchange topology information
        - Run a best-path algorithm
    - BGP forms neighborship using TCP port 179 (unicast)
    - eBGP neighbors are assumed to be directly connected
- **eBGP Neighborship Requirements**
    - **Following requirements must be met:**
        - The ASN number must match (remote-as) for the local/remote router
        - Peer must be reachable via an IGP route
        - BGP router-ids must not be the same for the two routers
        - If configured, MD5 authentication must pass
        - Each router must complete a TCP connection with the BGP peer
        - The source IP address used to reach that peer must match
    - **The BGP router-id is elected as follows:**
        - Manually configure the router-id
        - Choose highest IP address of any up/up loopback interface
        - Choose the highest IP address of any up/up non-loopback interface
- **BGP UPDATE-SOURCE**
    - The local router finds the outgoing interface to be used to reach IP address
    - This default behavior can be changed by explicitly specifying the source address
    - Useful with two links or redundant Layer 3 paths between the two routers
- **iBGP Neighborship Requirements**
    - Neighborship requirements are the same as eBGP except for the asn value
    - The asn value should be the same AS number as the local AS number
    - The exception here is the TTL value for iBGP neighbors is 255 by default
    - iBGP also defaults to multihop
    - Full mesh is preferred for iBGP despite being able to communicate multihop
        - The routers in between will not have routes going back
        - When the destination and source want to talk, they will be blackholed!
- **BGP MULTIPATH**
    - BGP supports the **`maximum-paths`** `number-of-paths` router subcommand
    - The logic is significantly different to IGP multipaths
    - BGP uses the best path algorithm to pick only one end route as the best path
    - These factors must be considered before load balancing can occur:
        - If the attributes tie from the ‘N WLLA OMNI’ formula
        - If routes are iBGP learned, next-hop IP address should be different
        - If routes are eBGP, should be learned from the same neighboring AS

3.30.a Peer group

- **PEER GROUP CONCEPT**
    - A BGP router may have dozens of BGP neighbors
    - Many neighbors may require the same BGP policies
    - **Configuring per-neighbor policies are burdensome**

        - Takes more time to configure
        - Requires more of CPU when sending/receiving BGP updates
    - **Peer Groups are groups of peers which the same outbound policies apply**
        - Only inbound policies can be overridden on a per-neighbor basis
        - All configuration options for neighbors can be applied to peer groups
        - Updates generated once per group
- **INTERNAL PEER GROUP CONFIGURATION EXAMPLE**
    - `Router(config)# router bgp 300`
    - `Router(config-router)# neighbor my-company peer-group`
    - `Router(config-router)# neighbor my-company remote-as 300`
    - `Router(config-router)# neighbor my-company route-map Attribute out`
    - `Router(config-router)# neighbor my-company filter-list 2 out`
    - `Router(config-router)# neighbor 2.2.2.2 peer-group my-company`
    - `Router(config-router)# neighbor 3.3.3.3 peer-group my-company`
    - `Router(config-router)# neighbor 3.3.3.3 filter-list 4 in`
- **EXTERNAL PEER GROUP CONFIGURATION EXAMPLE**
    - `Router(config)# router bgp 300`
    - `Router(config-router)# neighbor Externals peer-group`
    - `Router(config-router)# neighbor Externals route-map set-metric out`
    - `Router(config-router)# neighbor 7.7.7.7 remote-as 150`
    - `Router(config-router)# neighbor 7.7.7.7 peer-group Externals`
    - `Router(config-router)# neighbor 9.9.9.9 remote-as 250`
    - `Router(config-router)# neighbor 9.9.9.9 peer-group Externals`

3.30.b Active, passive

- **GENERAL CONCEPT**
    - Neighbors with the lowest BGP router-id will establish the connection
    - The port used will be destination TCP port 179 and the source port random
    - This behavior can be modified with a few commands
- **CONFIGURATION EXAMPLE**
    - `Router(config)# router bgp 500`
    - `Router(config-router)# neighbor 10.0.0.1 transport connection-mode passive`
- **BEFORE AND AFTER SNAPSHOT**
    - **BEFORE**:
        - ```
          Router# show ip bgp neighbors | in host
          Local host: 10.0.0.2, Local port: 57717
          Foreign host: 10.0.0.1, Foreign port: 179
          ```
    - **AFTER:**
        - ```
          Router# show ip bgp neighbors | in host
          Local host: 10.0.0.2, Local port: 179
          Foreign host: 10.0.0.1, Foreign port: 34121
          ```

3.30.c States and timers

- **OVERALL PROCESS**
    - A router tries to establish a TCP connection using destination port 179
    - When the TCP connection completes, the router sends BGP Open message
    - The BGP Open message operates similarly to Hello messages

- Once all the parameters must the routers become neighbors

- **BGP NEIGHBOR STATES**
  - **Idle**
    - The BGP process is administratively down or…
    - The BGP process is awaiting next retry attempt
  - **Connect**
    - The BGP process is waiting for the TCP connection to be completed
  - **Active**
    - TCP connection failed, Connect-Retry timer running
    - Listening for incoming TCP connection
  - **Opensent**
    - TCP connection exists, BGP Open message has been sent to the peer
    - The matching Open message has to yet been received from the router
  - **Openconfirm**
    - An Open message has been sent and received by both sides
  - **Established**
    - All neighbor parameters match and the neighbor relationship works
    - The peers can now exchange Update messages
- **BGP MESSAGE TYPES**
  - **Open**
    - Used in neighbor establishment
    - BGP values and capabilities are exchanged
  - **Update**
    - Informs neighbors about withdrawn, exchanged, and new routes
    - Used to exchange PAs and associated NLRI that use those attributes
  - **Notification**
    - Used to signal a BGP error
    - Typically results in a reset to the neighbor relationship
  - **Keepalive**
    - Sent on a periodic basis to maintain the neighbor relationship
    - The lack of receipt within the Hold timer causes down neighbor

3.31 Configure and verify eBGP (IPv4 and IPv6 address families)

3.31.a eBGP

- **eBGP BASIC CONFIGURATION STEPS**
  - `Router(config)# router bgp asn-number`
  - `Router(config-router)# bgp router-id rid`
  - `Router(config-router)# neighbor ip-address remote-as remote-asn`
  - `Router(config-router)# neighbor ip-address password key`
  - *Please note that the MD5 key is in the TCP header of the packet!*
- **BGP UPDATE-SOURCE CONFIGURATION STEPS**
  - `Router(config)# interface loopback0`
  - `Router(config-if) ip address loopback-address`

- `Router(config)#` **`router bgp`** `asn-number`
- `Router(config-router)#` **`neighbor`** `neighbor-loopback` **`remote-as`** `remote-asn`
- `Router(config-router)#` **`neighbor update-source`** `loopback-address`
- `Router(config-router)#` **`neighbor ebgp-multihop`** `hops`

- **iBGP BASIC CONFIGURATION STEPS**
  - `Router(config)#` **`router bgp`** `asn`
  - `Router(config-router)#` **`neighbor`** `neighbor-ip` **`remote-as`** `asn`
  - `Router(config-router)#` **`neighbor`** `neighbor-ip` **`update-source`** `loopback-address`
  - `Router(config-router)#` **`neighbor`** `neighbor-ip` **`password`** `key`
- **VERIFICATION**
  - **`show ip bgp prefix`** `[subnet-mask]`
  - **`show ip bgp neighbors`** `ip-address` **`received-routes`**
  - **`show ip bgp neighbors`** `ip-address` **`routes`**
  - **`show ip bgp neighbors`** `ip-address` **`advertised-routes`**
  - **`show ip bgp summary`**

3.31.b 4-byte AS number

- **GENERAL CONCEPT**
  - 4-byte ASNs provide 4,294,967,296 autonomous system numbers
  - The original 2-byte ASNs only provide 65536
  - **Mappable ASNs** are legacy ASN values that are mapped in 4-byte
  - **ASPLAIN** is a simple decimal representation of the ASN
  - **ASDOT**+ breaks into ASN into 16-bit values separated by a dot
- **INTEROPERABILITY**
  - Legacy peer responds it does not support 4-byte ASN capability or…
  - Legacy peer does not respond at all with the capability
  - New BGP peer can then use ASN 23456 which is a transit value (*AS_TRANS*)
  - Interoperable peering is achieved as new peer adapts to the legacy peer

3.31.c Private AS

- **GENERAL CONCEPT**
  - **The range for public and private AS numbers are:**
    - Public AS numbers 1 - 64495
    - Private As numbers 65512 - 65534
  - **The following numbers and ranges are reserved:**
    - 0, 64496 - 65511, 65535

3.32 Explain BGP attributes and best-path selection

- **OVERVIEW OF BGP BEST PATH ALGORITHM**
  - The best path algorithm takes the following initial steps to determine best path:
    - **N**: Next-Hop, it should be reachable
    - **W**: Weight, bigger value is preferred
    - **L**: Local Preference, bigger is preferred
    - **L**: Locally Originated Routes

        - **A**: AS_PATH length, smaller length is preferred
        - **O**: Origin, Prefer i> e > ?
        - **M**: MED, smaller is preferred
        - **N**: Neighbor Type, prefer eBGP over iBGP
        - **I**: IGP cost to next hop, smaller value is preferred
    - When BGP is unable to find best path, the following three are used as tiebreaker:
        - Oldest (longest known) eBGP route
        - Lowest neighbor BGP rid
        - Lowest neighbor IP address
    - **BGP Path Attribute Classifications**
        - **Mandatory** (Next-Hop, AS_PATH, Origin Code)
        - **Optional - Transitive** - Can leave local AS)
        - **Optional - Non-Transitive** - Not allowed to leave local AS (Local Pref)
- **NEXT-HOP ATTRIBUTE**
    - With eBGP, the next-hop IP address is changed by the advertising router
    - With iBGP, the next-hop IP address is not changed, but this is configurable
    - iBGP peers must have IGP reachability to next-hops. If not:
        - iBGP-learned routes will not be installed in IP Routing Table
        - iBGP-learned routes will not be advertised to any other BGP peers
    - The most common ways to resolve this issue are:
        - Advertise links to eBGP peers to the internal network
        - Use the **`neighbor`** `x.x.x.x` **`next-hop-self`** command
    - Generally, to prevent potential black holes, iBGP should be full mesh
    - A route reflector system can also be used for scalability
    - **Synchronization** can also resolve this issue, but not recommended
        - Only BGP routes with IGP routes are preferred
        - This inflates the IGP routing table tremendously
        - Enable with **`synchronization`** (router subcommand)
- **WEIGHT ATTRIBUTE**
    - Not generally considered a path attribute; it is a Cisco proprietary feature
    - It is locally significant to a single router and is not advertised to any neighbors
    - It can be used to influence the choice of outbound routes
    - It is set on inbound routes
    - A bigger value is preferred for it
    - The default value of weight is 0 for learned routes and 32768 for locally injected
    - These default values cannot be modified
    - **Two Configuration Options:**
        - **`neighbor`** `x.x.x.x` **`route-map`** `name (use` **`set weight`**`)`
        - **`neighbor`** `x.x.x.x` **`weight`** `value (affects all prefixes)`
- **LOCAL PREFERENCE ATTRIBUTE**
    - Purpose is to identify the best exit point from the AS to reach a given prefix
    - This attribute is advertised to all iBGP neighbors but not to eBGP
    - Higher value is preferred
    - Default value is 100

- **Changing Default Value:**
    - **default local-preference** <0-4294967295> (router subcommand)
- **Changes to LOCAL_PREF are applied to an eBGP neighbor using:**
    - **neighbor** x.x.x.x **route-map** name **in** (router subcommand)
- Within the route-map, the **set local-preference** command is used

LOCALLY ORIGINATED ROUTES ATTRIBUTE

- A route that was injected locally by the router in question
- If you see a next-hop of 0.0.0.0, that means a locally originated route
- **Inject Route Locally:**
    - **network** x.x.x.x **mask** y.y.y.y (router subcommand)

AS_PATH ATTRIBUTE

- By default, if not BGP PA have been explicitly set, AS_PATH is used
- AS_SEQ is a component of the AS_PATH attribute
- Lists the ASNs that would be part of an end-to-end path
- **BGP uses the AS_PATH to perform two functions:**
    - Choose best route for a prefix on the shortest AS_PATH
    - Prevent potential routing loops
- When a route is advertised by BGP, the AS number is prepended in AS_PATH
- When a router receives an update that lists its own ASN, router ignores that route

AS_PATH PREPENDING

- WIth AS_PATH prepending, the length of AS_PATH PA can be made longer
- As mentioned previously, longer AS_PATH is less preferred
- Ideally, local AS number is prepended, otherwise you risk other routers dropping
- Configured using **set as-path prepend** under a route-map entry

ORIGIN ATTRIBUTE

- When BGP creates a route, one of the mandatory attributes is the *Origin Code*
- **Routes originated by:**
    - **network** command = igp
    - **redistributed** command = unknown (?)
- **Order of preference:**
    - igp > egp > ?
- The symbols will be located in **show ip bgp** command to the far right

MULTI-EXIT DISCRIMINATOR (MED)

- Optional non-transitive attribute
- Can be modified to influence the inbound traffic coming from the neighboring AS
- It is not advertised beyond the neighboring AS so it only goes one AS deep!
- A smaller value is preferred
- Default value is 0
- The MED value can be changed using the command set metric under route-map
- The direction will always be outbound
- If the same route came from two different external AS, MED cannot be compared
- You can get the router to compare the MED values if you want:
    - **bgp always-compare-med** (router subcommand)

4.0 VPN Technologies

4.1 Configure and verify GRE

- **GRE A.K.A. *"Generic Route Encapsulation"***
    - Tunneling mechanism, can contain many different protocols in header
    - Default encapsulation for Cisco IOS Tunnel Interfaces
- **CONFIGURATION STEPS**
    - `Router(config)# interface tunnel 0`
    - `Router(config-if)# ip address 10.1.3.1 255.255.255.0`
    - `Router(config-if)# tunnel source Loopback 0`
    - `Router(config-if)# tunnel destination 3.3.3.2`
- **VERIFICATION**
    - `show ip interface brief`

4.2 Describe DMVPN (single hub)

- **DMVPN A.K.A. *"Dynamic Multipoint Virtual Private Network"***
    - Point-to-multipoint Layer 3 Overlay VPN
    - Logical Hub and Spoke topology
    - Direct spoke to spoke traffic supported
- **DMVPN REQUIREMENTS**
    - Hub Router, reachable via static, public IP address
    - Spoke Routers, reachable via static, or dynamic, public IP address
    - Multipoint GRE Tunnels
    - IPSec, optional, but typically used
- **MULTIPOINT GRE CONFIGURATION (mGRE):**
    - `Router(config)# interface tunnel 0`
    - `Router(config-if)# ip address 10.1.3.1 255.255.255.0`
    - `Router(config-if)# tunnel source Loopback0`
    - `Router(config-if)# tunnel key 1`
    - `Router(config-if)# tunnel mode gre multipoint`
- **NHRP A.K.A. *"Next Hop Resolution Protocol"***
    - Used for a tunnel to answer the following questions:
        - What is the private tunnel IP address of my peer?
        - What is the public IP address of my peer?
    - Maps a peer's tunnel IP address to peer's public IP address
    - Can populate NHRP cache via static or dynamic entries
- **NHS A.K.A. *"Next Hop Server"* (Hub)**
    - Statically defined in spoke
    - Example: NHRP NHS = 10.1.3.2, NHRP Cache: 10.1.3.2 -> 3.3.3.2 (static)
- **NHRP MESSAGE TYPES**
    - **NHRP Registration Request**
        - Spokes register their NBMA and VPN IP to NHS
        - Required to build Spoke-to-Hub tunnels

- **NHRP Resolution Request**
    - Spoke queries for NBMA-to-VPN mappings of other spokes
    - Required to build Spoke-to-Spoke tunnels
- **NHRP Redirect**
    - NHS answer to Spoke-to-Spoke data-plane packet through it

- **DMVPN PHASES OVERVIEW**
  - **Phase-1**: Hub and Spoke only
    - If spoke wanted to send data to each other, they could not
    - Hub can become the bottleneck in CPU and Bandwidth
    - Summarization is possible
    - Now considered obsolete
  - **Phase-2**: Adds Spoke-to-Spoke Capability
    - All spoke routers must know all IP routes of all other spoke routers
    - NO summarization allowed
    - NO default route origination allowed from hub
    - Removes burden from hub in terms of CPU and Bandwidth
    - Spokes now have a mGRE interface like the hub
  - **Phase-3**: Modification of Routing Behavior
    - Hub allowed to summarize all routes from Spokes
    - Sets the next-hop of summarized routes to itself
    - Hub can send NHRP Redirect Messages to Spokes
    - Hub preemptively sends information to Spokes about other Spokes
    - Spokes no longer needs to ask via NHRP Resolution Request
    - Next-hop preservation no longer required
    - Default route from hub is now possible

- **DMVPN NEGOTIATION OVERVIEW**
  - Creates on-demand tunnels between nodes
    - Initial tunnel-mesh is hub-and-spoke (always on)
    - Traffic patterns trigger spoke-to-spoke tunnels
    - Solves management scalability problem
  - Maintains tunnels based on traffic patterns
    - Spoke-to-Spoke tunnel is on-demand
    - Spoke-to-Spoke tunnel lifetime is based on traffic
  - Requires two IGPs: Underlying and Overlay
    - IPv4/IPv6 supported for both passenger and transport
    - One routing protocol with service provider to exchange public addresses
    - Another routing protocol between corporate routers

- **DMVPN PHASE 1 AND 2 CONFIGURATION**
  - `Hub(config)#` **`interface tunnel`** `0`
  - `Hub(config-if)#` **`tunnel source`** `{ipv4-address | interface}`
  - `Hub(config-if)#` **`tunnel key`** `{#}`
  - `Hub(config-if)#` **`ip nhrp map multicast dynamic`**
  - `Hub(config-if)#` **`ip nhrp network-id`** `{#}`
  - `Hub(config-if)#` **`ip mtu`** `1400`
  - `Hub(config-if)#` **`ip tcp adjust-mss`** `1360`

- `Spoke(config)# interface tunnel 0`
- `Spoke(config-if)# tunnel destination {NBMA IP of Hub}`
- *This is the phase differentiator - remove for Phase 2*
- `Spoke(config-if)# tunnel key {#}`
- `Spoke(config-if)# ip nhrp map multicast {NBMA IP of HUB}`
- `Spoke(config-if)# ip nhrp map {private-ip} {NBMA IP}`
- `Spoke(config-if)# ip nhrp network-id {#}`
- `Spoke(config-if)# ip mtu 1400`
- `Spoke(config-if)# ip tcp adjust-mss 1360`

- **DMVPN PHASE 2 NHRP FLOW**
  - NHRP Resolution Request initiated by spoke:
  - Spoke looks up next-hop address (tunnel IP) for destination network
  - Spoke looks into local NHRP database for mapping
    - If it does, it sends the traffic
    - If not, sends the NHRP Resolution Request to its NHS
  - The hub forwards to owner of network we are trying to reach
  - Owner responds directly to requesting spoke
  - Requestor adds NHRP mapping to local database
  - Spoke-to-spoke communication has established
- **DMVPN PHASE 3 NHRP FLOW**
  - Spoke goes to send traffic to another spoke
  - Spokes can and should be using a default route!
  - Traffic routed to the hub
  - Hub sends NHRP Redirect to spoke (traffic continues to flow!)
  - Spoke receives redirect and generates NHRP Resolution Request
  - Hub receives NHRP Resolution Request and forwards to destination spoke
  - Destination spoke sends NHRP Resolution Reply to originating spoke
  - Viola! Spoke-to-spoke tunnel as initiated by the hub
- **DMVPN PHASE 3 CONFIGURATION**
  - `Hub(config-if)# ip nhrp redirect`
  - `Spoke(config-if)# ip nhrp shortcut`

4.3 Describe Easy Virtual Networking (EVN)

- **EVN A.K.A. *"Easy Virtual Networking"***
  - An easy method to provide complete isolation between network segments
    - No VRF / VRF-lite
    - No BGP
    - No Vlan Routing / ACLs
  - **Reality**: Automation tool for vrf-lite, relies on 802.1Q encapsulation
- **VRF-LITE CONFIGURATION STEPS**
  - `Router(config)# interface gi1/1`
  - `Router(config-if)# ip address 10.5.5.1 255.255.255.0`
  - `Router(config-if)# ip pim sparse-mode`
  - `Router(config)# interface gi1/1.51`
  - `Router(config-if)# description Subinterface for Group1`
  - `Router(config-if)# encapsulation dot1Q 51`
  - `Router(config-if)# ip vrf forwarding Group1`

  - Router(config-if)# **ip address** 10.5.5.1 255.255.255.0
  - Router(config-if)# **ip pim sparse-mode**
  - Router(config)# **interface** gi1/1.52
  - Router(config-if)# **description** Subinterface for Group2
  - Router(config-if)# **encapsulation dot1Q** 52
  - Router(config-if)# **ip vrf forwarding** Group2
  - Router(config-if)# **ip address** 10.5.5.1 255.255.255.0
  - Router(config-if)# **ip pim sparse-mode**
- **EVN CONFIGURATION STEPS**
  - Router(config)# **vrf definition** group1
  - Router(config-vrf)# **vnet tag** 51
  - Router(config)# **vrf definition** group2
  - Router(config-vrf)# **vnet tag** 52
  - Router(config)# **interface** gi1/1
  - Router(config-if)# **vnet trunk**
  - Router(config-if)# **ip address** 10.5.5.1 255.255.255.0
  - Router(config-if)# **pim sparse-mode**
  - Router(config)# **interface** gi1/2
  - Router(config-if)# **vrf forwarding** group1
  - Router(config-if)# **ip address** 10.75.1.2 255.255.255.0
- **EVN SUMMARY AND REALITY**
  - EVN generates subinterfaces with vnet trunk
  - EVN adds a vrf forwarding instance to each subinterface
- **VERIFICATION**
  - **show vrf**
  - **ping vrf** <VRF> <IP>

5.0 Infrastructure Security

5.1 Describe IOS AAA using local database

- **AAA A.K.A. *"Authentication, Authorization, and Accounting"***
  - **Authentication**: Verify the identity of the user.
  - **Authorization**: What is the user allowed to do?
  - **Accounting**: Used for billing and auditing
- **GENERAL CONCEPT**
  - **802.1X** is the port-based mechanism that will **block** or **unblock** the interface
  - EAPoL A.K.A. ***"Extensible Authentication Protocol over LAN"***
    - No traffic allowed beside EAPoL
    - EAPoL is the protocol that verifies the user
  - Once authenticated, network access is granted and traffic resumes
- **AUTHENTICATION SERVER TYPES**
  - RADIUS A.K.A. ***"Remote Authentication Dial-In User Service"***
  - TACACS+ A.K.A. ***"Terminal Access Controller Access-Control System"***
- **PRIVILEGE LEVELS**
  - **Level-0**: Only a few commands are available, most used command is 'enable'
  - **Level-1**: Default exec user level, can use some show commands
  - **Level-15**: Also known as 'enable mode' or 'privileged mode'; full access
- **CONFIGURATION OPTIONS**

- Assign some privilege level 15 commands to level 1
- Move some commands from level 1 to higher level
- Create new privilege level and assign some level 15 commands to it

- **CONFIGURATION STEPS**
  - ***Set Command Privilege***
  - Router(config)# **privilege exec level** 1 show running-config
  - Router(config)# **privilege exec level** 15 show ip arp
  - ***Enable AAA Locally***
  - Router(config)# **username** JUNIOR **privilege** 8 **password** CISCO
  - Router(config)# **aaa new-model**
  - Router(config)# **aaa authentication login default local**
  - ***New User Privilege***
  - Router(config)# **privilege exec level** 8 configure terminal
  - Router(config)# **privilege exec level** 8 debug ip routing
  - Router(config)# **privilege exec level** 8 undebug all
  - Router(config)# **privilege exec level** 8 show running-config
  - Router(config)# **privilege configure level** 8 interface
  - Router(config)# **privilege interface level** 8 shutdown
  - Router(config)# **privilege interface level** 8 no shutdown

5.2 Describe device security using IOS AAA with TACACS+ and Radius

5.2.a AAA with TACACS+ and RADIUS

- **RADIUS CONFIGURATION STEPS**
  - Router(config)# **aaa new-model**
  - Router(config)# **radius-server host** 192.168.1.200 **auth-port** 1812 **acct-port** 1813
  - Router(config)# **radius-server key** MY_KEY
  - Router(config)# **aaa authentication login default group radius**
- **TACACS+ CONFIGURATION STEPS**
  - Router(config)# **aaa new-model**
  - Router(config)# **tacacs-server host** 192.168.1.200 **port** 49 **key** MY_KEY
  - Router(config)# **aaa authentication login default group tacacs+**
- **PORT INFORMATION**
  - RADIUS uses UDP, reserved ports are 1812 and 1813
  - RADIUS legacy unofficial ports are 1645 and 1646
  - TACACS+ uses TCP, reserved port is 49
- **AUTHORIZATION CONFIGURATION STEPS**
  - Router(config)# **aaa authorization exec group tacacs+**
- **ACCOUNTING CONFIGURATION STEPS**
  - Router(config)# **aaa accounting network default stop-only group tacacs+**
- **VERIFICATION**
  - **test aaa group** [**radius** | **tacacs+**] <username> <password> **new-code**

5.2.b Local privilege authorization fallback

- **GENERAL CONCEPT**

- If TACACS+ or RADIUS becomes unavailable, need fallback option
    - `Router(config)#` **`aaa authentication login default group tacacs+ local`**
- The **`local`** keyword allows local database usage!

5.3 Configure and verify device access control

5.3.a Lines (VTY, AUX, console)

- **GENERAL CONCEPT**
    - The lines all provided access to the CLI, in one form or another.
    - **Line VTY**
        - Provided SSH or Telnet access to the device
    - **Line AUX**
        - Provided CLI access via aux cable
        - Only showed output during complete power up
    - **Line CON**
        - Provided CLI access through console cable
        - Showed output during power up process as well

5.3.b Management plane protection

- **MPP A.K.A. *"Management Plane Protection"***
    - Allows certain protocols on the management interface
        - **beep**: Beep Protocol
        - **ftp**: File Transfer Protocol
        - **http**: HTTP Protocol
        - **https**: HTTPS Protocol
        - **snmp**: Simple Network Management Protocol
        - **ssh**: Secure Shell Protocol
        - **telnet**: Telnet Protocol
        - **tftp**: Trivial File Transfer Protocol
        - **tl1**: Transaction Language Session Protocol
- **CONFIGURATION STEPS**
    - `Router(config)#` **`control-plane host`**
    - `Router(config-cp-host)#` **`management-interface`** `FastEthernet0/0` **`allow ssh`**
    - `Router(config-cp-host)#` **`management-interface`** `FastEthernet0/0` **`allow snmp`**

5.3.c Password encryption

- **GENERAL CONCEPT**
    - There are two primary ways to encrypt passwords on Cisco devices
        - `Router(config)#` **`enable secret`** `PASSWORD`
        - `Router(config)#` **`service password-encryption`**
    - Secret enables encryption on the enable password
    - The service command does it globally to all passwords

5.4 Configure and verify router security features

5.4.a IPv4 access control lists (standard, extended, time-based)

- **ACL A.K.A. *"Access Control List"***
  - Two flavours, standard and extended
  - PERMIT or DENY addresses and/or protocols
  - Explicit DENY at the end of every ACL
- **STANDARD ACL CONFIGURATION STEPS**
  - `Router(config)# access-list 1 permit 192.168.12.0 0.0.0.255`
  - `Router(config)# interface fastEthernet 0/0`
  - `Router(config-if)# ip access-group 1 in`
- **EXTENDED ACL CONFIGURATION STEPS**
  - `Router(config)# access-list 100 permit tcp 1.1.1.0 0.0.0.255 host 2.2.2.2 eq 80`
- **ACCESS-LIST EDITOR CONFIGURATION STEPS**
  - `Router(config)# ip access-list extended 100`
  - `Router(config-ext-nacl)# no 20`
- **NAMED ACL CONFIGURATION STEPS**
  - `Router(config)# ip access-list extended DROPICMP`
  - `Router(config-ext-nacl)# deny icmp host 192.168.12.2 1.1.1.0 0.0.0.255`
- **VERIFICATION**
  - `show access-lists`
- **TIME-BASED ACL CONFIGURATION STEPS**
  - `Router(config)# time-range WH`
  - `Router(config-time-range)# periodic weekdays 9:00 to 17:00`
  - `Router(config)# ip access-list extended NO_FACEBOOK`
  - `Router(config-ext-nacl)# deny tcp any host 192.168.23.3 eq 80 time-range WH`
  - `Router(config-ext-nacl)# permit ip any any`
  - *When verifying time based ACLs, check for **active** keyword*

5.4.b IPv6 traffic filter

- **GENERAL CONCEPT**
  - IPv6 standard ACL offers the following functions:
    - Filter traffic based on source and destination address
    - Filter traffic inbound or outbound on a specific interface
    - Add priority to the ACL
    - Implicit **`deny all`** at the end of the ACL
  - No concept of numbered IPv6 ACLs, just named ACLs
  - No concept of ***standard*** or ***extended***
- **IPv6 ACL CONFIGURATION STEPS**
  - `Router(config)# ipv6 access-list TEST`
  - `Router(config-ipv6-acl)# permit tcp any 2001:aaaa::/64 eq telnet`
  - `Router(config)# interface Ethernet0`
  - `Router(config-if)# ipv6 traffic-filter TEST out`
- **VERIFICATION**
  - `show ipv6 access-list`
  - `debug ipv6 packet`

5.4.c Unicast reverse path forwarding

- **GENERAL CONCEPT**
    - **Normal Scenario**: Router looks at destination address of packet before routing
    - uRPF A.K.A. ***"Unicast Reverse Path Forwarding"***
        - Checks source address as well
        - **Three modes**: Loose, Strict, VRF Mode
- **LOOSE MODE**
    - Checks to see if the source address route exists on the routing table
- **STRICT MODE**
    - Same as loose, except also checks if source interface is the same on router
    - This means asynchronous routing will be blocked
- **CONFIGURATION STEPS**
    - `Router(config-if)# ip verify unicast source reachable-via [any | rx]`

6.0 Infrastructure Services

6.1 Configure and verify device management

6.1.a Console and VTY

- **GENERAL CONSOLE CONFIGURATION STEPS**
    - `Router(config)# line con 0`
    - `Router(config-line)# login`
    - `Router(config-line)# password THE_PASS`
- **GENERAL VTY CONFIGURATION STEPS**
    - `Router(config)# line vty 0 4`
    - `Router(config-line)# login local`
    - `Router(config-line)# access-class 1 in`
    - `Router(config-line)# transport input ssh telnet`
    - `Router(config)# username cindy password cisco`
- **VERIFICATION**
    - `show line`
    - **Note**: You can also configure based on the line number as displayed in the above

6.1.b Telnet, HTTP, HTTPS, SSH, SCP

- **TELNET**
    - TCP Port 23
    - Clear text authentication protocol
    - **Configuration Example:**
        - `Router(config)# username cisco password 0 cisco`
        - `Router(config)# line vty 0 4`
        - `Router(config-line)# transport input telnet`

- **SSH**
    - TCP Port 22
    - Encrypted and secure
    - **Configuration Example**:
        - `Router(config)# hostname Router`
        - `Router(config)# aaa new-model`
        - `Router(config)# ip domain-name cisco.com`
        - `Router(config)# username cisco password 0 cisco`
        - `Router(config)# crypto key generate rsa`
        - `Router(config)# line vty 0 4`
        - `Router(config-line)# transport input ssh`
- **HTTP**
    - TCP Port 80
    - Clear text www protocol
    - Used for GUI access to devices
    - **Configuration Example:**
        - `Router(config)# ip http server`
        - `Router(config)# ip http authentication aaa`
- **HTTPS**
    - TCP Port 443
    - Encrypted www protocol
    - Used for GUI access to devices
    - **Configuration Example:**
        - `Router(config)# ip http secure-server`
        - `Router(config)# crypto ca trustpoint CA-trust-local`
        - `Router(config-ca)# enrollment url http://host1:80`
        - `Router(config)# crypto ca authenticate CA-trust-local`
        - `Router(config)# crypto ca enrollment CA-trust-local`
        - `Router(config)# ip http secure-trustpoint CA-trust-local`
- **SCP**
    - TCP Port 22
    - Secure file transfer, uses same system as SSH
    - Triple A and SSH must be configured in order for SCP to work
    - **Configuration Example:**
        - `Router(config)# aaa new-model`
        - `Router(config)# aaa authentication login default local`
        - `Router(config)# aaa authorization exec default local`
        - `Router(config)# username user1 privilege 15 password 0 lab`
        - `Router(config)# ip scp server enable`

6.1.c (T)FTP

- **GENERAL CONCEPT**
    - Another means of copying from one location to another
    - Port Information:
        - **TFTP**: UDP 69
        - **FTP**: TCP 21 for initial authentication, TCP 20 for data transfer
- **TFTP CONFIGURATION STEPS**

        - `Router1(config)# tftp-server flash:/c2500-js-1.122-10b`
        - `Router2# copy tftp flash`
- **FTP CLIENT CONFIGURATION STEPS**
        - `Router(config)# ip ftp username cisco`
        - `Router(config)# ip ftp password cisco123`
        - `Router# copy running-config ftp:`
- **FTP SERVER CONFIGURATION STEPS (DEPRECATED)**
        - `Router(config)# ftp-server enable`

6.2 Configure and verify SNMP

6.2.a v2

- **SNMP VERSIONS**
    - **Version-1:** Open standard and basic
    - **Version-2:** Requires community string to gain read/write access
    - **Version-3:** Adds encryption and real authentication
- **OID A.K.A. *"Object Identifier"***
    - Numeric string that can be assigned various properties of a router
    - Example: Interface Counters
- **MIB A.K.A. *"Management Information Base"***
    - Collection of OIDs
    - This can be imported to monitoring tool for it to understand router
- **SNMP GENERAL CONCEPT**
    - NMS A.K.A. ***"Network Management System"***
        - External server to store logging information
    - SNMP Agents
        - Runs on network devices to monitor it
- **COMMUNITY CONFIGURATION STEPS**
    - `Router(config)# snmp-server community TSHOOT ro`
- **OPTIONAL CONFIGURATION STEPS**
    - `Router(config)# snmp-server location Amsterdam`
    - `Router(config)# snmp-server contact info@networklessons.com`
    - Displays information about where the device is located and contact to NMS
- **FULL CONFIGURATION STEPS**
    - `Router(config)# snmp-server host 192.168.12.2 version 2c TSHOOT`
    - `Router(config)# snmp-server enable traps`

6.2.b v3

- **THREE SECURITY LEVELS**
    - **noAuthNoPriv** = *no authentication and no encryption*
    - **AuthNoPriv** = *authentication but no encryption*
    - **AuthPriv** = *authentication AND encryption*
    - *With **noAuthNoPriv**, the username will **replace the community-string***
- **THREE NEW ELEMENTS**

- **SNMP View:** Can restrict "view" of device, so only specific OIDs are visible
- **SNMP Group**: Can associate view with group, and can specify read/write access
- **SNMP User**: Exact user credentials required to access view based on group

- **GROUP CONFIGURATION STEPS**
  - `Router(config)#` **`snmp-server group`** `MYGROUP` **`v3 priv`** `<access | read | write>`
  - *If you do not specify, it will assume read-only*
- **USER CONFIGURATION STEPS**
  - `Router(config)#` **`snmp-server user`** `USER GROUP` **`v3 auth md5`** `PASS` **`priv aes 128`** `KEY`
- **VERIFICATION**
  - **`show snmp user`**
  - **`show snmp group`**
- **VIEW CONFIGURATION STEPS**
  - `Router(config)#` **`snmp-server view`** `ALL-ACCESS 1.3.6.1.2.1.2.2.11` **`included`**
  - `Router(config)#` **`snmp-server group`** `MYGROUP` **`v3 priv read`** `ALL-ACCESS`

6.3 Configure and verify logging

6.3.a Local logging, syslog, debugs, conditional debugs

- **LOGGING CONSOLE CONFIGURATION STEPS**
  - `Router(config)#` **`logging console`**
  - *This is enabled by default on IOS*
  - `Router#` **`terminal monitor`**
  - *Enables logging on screen to SSH/Telnet for that session*
- **LOCAL HISTORY CONCEPT**
  - Cisco IOS keeps history of syslog messages
  - Can be displayed via **`show logging`**
  - Messages stored in RAM by default
- **INCREASE BUFFER SIZE (RAM) CONFIGURATION STEPS**
  - `Router(config)#` **`logging buffered`** `16384`
- **SYSLOG SERVER CONFIGURATION STEPS**
  - Store logging messages remotely in a server as RAM data can be lost
  - `Router(config)#` **`logging`** `192.168.1.2`
- **DEBUG CONFIGURATION STEPS**
  - See debug information via console
    - `Router(config)#` **`logging console debugging`**
  - See debug information via VTY
    - `Router(config)#` **`logging monitor debugging`**
  - Filtering can be done based on severity levels, debug being severity 7
- **SEVERITY LEVELS**
  - *Every Alley Cat Eats Wet Noodles In Doors*
  - **Severity-0**: Emergencies
  - **Severity-1**: Alerts
  - **Severity-2**: Critical
  - **Severity-3**: Errors
  - **Severity-4**: Warnings

- **Severity-5**: Notifications
- **Severity-6**: Informational
- **Severity-7**: Debugging

- **CONDITIONAL DEBUGGING CONCEPT**
  - Filter based on specific interfaces and some other items
  - You need to turn on some debug item first, then you can turn on the filter
- **CONDITIONAL DEBUGGING CONFIGURATION STEPS**
  - `Router# debug ip rip`
  - `Router# debug condition interface serial 0`
- **VERIFICATION**
  - `show debug condition`
- **REMOVE CONDITIONAL DEBUGGING STEPS**
  - `Router# undebug condition interface serial 0`

6.3.b Timestamps

- **CONFIGURATION STEPS**
  - `Router(config)# service timestamps log`
  - `Router(config)# service timestamps log datetime msec`
  - `Router(config)# service sequence-numbers`

6.4 Configure and verify Network Time Protocol (NTP)

6.4.a NTP master, client, version 3, version 4

- **TIME TRACKING PURPOSE**
  - Logging output
  - Debugging output
  - User “show” commands
  - Network Management/Reporting Tools
- **CLOCK SOURCE**
  - Internal **System Clock**
  - Most are battery-driven and maintains the time/date across reloads
  - This information can be distributed via NTP
- **SYSTEM CLOCK INFORMATION SOURCE**
  - Manual Configuration
  - NTP A.K.A. ***“Network Time Protocol”***
  - SNTP A.K.A. ***“Simple Network Time Protocol”***
  - VINES A.K.A. ***“Virtual Integrated Network Service”***
- **VERSIONS**
  - **Version 3**: Only works with IPv4
  - **Version 4**: Introduced concept of NTP authentication
  - Uses UDP port 123 for source and destination
- **NTP CLIENTS AND SERVERS**

- **NTP Client**
    - Device that periodically polls a server for time/calendar information
- **NTP Server**
    - Provides time information to client
    - Considered “authoritative source” based on stratum:
        - **Stratum-1:** Device directly connected to atomic clock source
        - **Stratum-X:** Time Server that is X hops away from Stratum-1

- **MODES OF OPERATION**
    - **NTP Server**
        - Also called NTP Master
    - **NTP Client**
        - Receives NTP data from Server/Master
    - **NTP Peers**
        - Also called “Symmetric Mode”
        - Two-or-more NTP Servers exchange information
    - **NTP Broadcast/Multicast**
        - Unidirectional “push” from NTP server
- **NTP MASTER CONFIGURATION STEPS**
    - Router# **clock set** 07:01:30 28 Feb 2015
    - Router(config)# **ntp master** [stratum]
    - Router(config)# **ntp peer** x.x.x.x (optional)
    - Router(config-if)# **ntp broadcast** (optional)
- **NTP CLIENT CONFIGURATION STEPS**
    - Router(config)# **ntp server** x.x.x.x **version** 4 OR
    - Router(config-if)# **ntp broadcast client**
- **VERIFICATION**
    - **show ntp status**
    - **show ntp associations**

6.4.b NTP authentication

- **GENERAL CONCEPT**
    - **Two Protection Methods:**
        - Access-Lists
        - NTP Authentication
    - Uses an MD5 Hash of configured authentication keys
    - Multiple keys can be configured for use against different peers
- **MASTER CONFIGURATION STEPS**
    - Router(config)# **ntp authentication-key** <key-number> **md5** <key-string>
    - Router(config)# **ntp trusted-key** <key-number>
    - Router(config)# **ntp authenticate**
- **CLIENT CONFIGURATION STEPS**
    - Router(config)# **ntp authentication-key** <key-number> **md5** <key-string>
    - Router(config)# **ntp trusted-key** <key-number>
    - Router(config)# **ntp authenticate**
    - Router(config)# **ntp server** x.x.x.x **key** <key-number>

6.5 Configure and verify IPv4 and IPv6 DHCP

6.5.a DHCP client, IOS DHCP server, DHCP relay

- **DHCP A.K.A. *"Dynamic Host Configuration Protocol"***
- **DHCPv4 PROCESS (DORA)**
    - **DHCPv4 Discover**
        - Client sends a broadcast requesting DHCP services
        - May include last known IPv4 address or 0.0.0.0
    - **DHCPv4 Offer**
        - Server responds with an offer, which includes:
            - IPv4 Address
            - Subnet Mask
            - Lease Duration
            - Client's MAC address
        - Offer may include other options:
            - Default Gateway
            - DNS Server Addresses
    - **DHCPv4 Request**
        - Client responds with request to accept the offered address
        - The request is broadcasted to inform all DHCP servers on LAN
    - **DHCPv4 Acknowledgement**
        - Server sends ACK to client and closes out communication
- **DHCPv4 CONFIGURATION STEPS**
    - `Router(config)# ip dhcp pool MY_POOL`
    - `Router(dhcp-config)# network 192.168.12.0 255.255.255.0`
- **DHCPv4 EXCLUDING CONFIGURATION STEPS**
    - `Router(config)# ip dhcp excluded-address 192.168.12.100`
- **DHCPv6 THREE METHODS**
    - **Method-1:** Stateless Address Autoconfiguration (SLAAC)
    - **Method-2:** SLAAC + Stateless DHCPv6 Server
    - **Method-3:** Stateful DHCPv6 Server
- **COMMUNICATION OCCURS VIA ICMPv6**
    - RS A.K.A. ***"Router Solicitation"***
        - Clients send RS to obtain IPv6 addresses dynamically
        - Source IPv6 address for RS is either link local address or unspecified
    - RA A.K.A. ***"Router Advertisement"***
        - Automatically sent out interfaces if IPv6 unicast-routing is enabled
        - Configuring Router as IPv6 Router (`ipv6 unicast-routing`):
            - Can forward IPv6 packets transiting the router
            - Can configure with dynamic IPv6 routing protocol
            - IPv6 interfaces joins the all-IPv6 multicast group (FF02::2)
        - Used to advertise presence and link specific parameters:
            - Link Prefix
            - Prefix-length

- Default gateway
    - Link MTU
    - Suggested IPv6 address method
  - Sends advertisements:
    - Every 200 seconds automatically
    - In response to RS message
- **A, O, AND M FLAGS**
  - **Address Autoconfiguration Flag (A Flag)**
    - Suggests SLAAC
    - Allows host to generate own GUA A.K.A. ***"Global Unicast Address"***
    - Two options for creating Interface ID:
      - EUI-64 Process
      - Random 64-bit Value
  - **Other Configuration Flag (O Flag)**
    - Suggests Stateless DHCPv6 Server
    - May include DNS server address and domain name
  - **Managed Address Configuration Flag (M Flag)**
    - Suggests Stateful DHCPv6 Server
    - Similar to DHCP for IPv4
    - Only difference is, host uses RA packet source as default gateway
- **FOUR PRIMARY DHCPv6 MESSAGE TYPES**
  - **SOLICIT (1)**
    - Client message to locate servers
  - **ADVERTISE (2)**
    - Server message to clients to indicate available DHCPv6 service
  - **REQUEST (3)**
    - Client message to request configuration parameters including IPv6 addresses, from specific DHCPv6 server
  - **REPLY (7)**
    - Server message containing assigned addresses and configuration parameters
- **SLAAC A.K.A. *"Stateless Address Autoconfiguration"***
  - Host generates own GUA without any services based on provided prefix in RA
  - **Flags**: A = 1, O = 0, M = 0 (default behavior)
  - With **Privacy Extension**, host will generate random interface, otherwise EUI-64:
    - MAC address split into two sections, and FF:FE inserted in between
    - The **seventh bit of the first byte** is inverted
  - Hosts generate Temporary Addresses with **Privacy Extension**:
    - Generated using random 64-bit value
    - Used as source IPv6 address when device originates connection
  - Hosts will generate a Public Address:
    - Used as a permanent destination address by other hosts
    - Hosts may have multiple Temporary Addresses, but only one Public
  - **Client Configuration:**

    - Router(config-if)# **ipv6 address autoconfig**
- **STATELESS DHCPv6 WITH SLAAC**
    - Host generates own GUA with services provided by DHCP server
    - **Flags**: A = 1, O = 1, M = 0
    - **Server Configuration:**
        - Router(config)# **ipv6 dhcp pool** <pool-name>
        - Router(config-dhcpv6)# **dns-server** <ipv6-address>
        - Router(config-dhcpv6)# **domain-name** <domain>
        - Router(config-if)# **ipv6 nd other-config-flag**
        - Router(config-if)# **ipv6 dhcp server** <pool-name> [**rapid-commit**]
    - The rapid-commit option shortens messages from four to two:
        - SOLICIT
        - REPLY
    - **Relay Agent Configuration:**
        - Router(config-if)# **ipv6 dhcp relay destination** <ipv6-address>
- **STATEFUL DHCPv6**
    - Host receives prefix and options from DHCP server
    - **Flags**: A = 0, O = 0, M = 1
    - **Server Configuration:**
        - Router(config)# **ipv6 dhcp pool** <pool-name>
        - Router(config-dhcpv6)# **address prefix** <prefix/length>
        - Router(config-dhcpv6)# **dns-server** <ipv6-address>
        - Router(config-dhcpv6)# **domain-name** <domain>
        - Router(config-if)# **ipv6 nd managed-config-flag**
        - Router(config-if)# **ipv6 nd prefix** <prefix/length> **no-autoconfig**
        - Router(config-if)# **ipv6 dhcp server** <pool-name>

6.5.b DHCP options (describe)

- **GENERAL CONCEPT**
    - DHCP includes other configuration options other than just address allocation
    - DHCPv4 can hand out useful data such as default gateway, DNS server, etc.
    - DHCPv6 will not hand out default gateway as the RA does that already

6.6 Configure and verify IPv4 Network Address Translation (NAT)

6.6.a Static NAT, dynamic NAT, PAT

- **NAT A.K.A. *"Network Address Translation"***
    - Function by which IP addresses within a packet are replaced with another
    - NAT44 refers to IPv4 <> IPv4
    - NAT444 describes architecture rather than technology
        - Two levels of NAT44
        - Enables IPv4 private address space to be used over and over
        - Service providers use NAT444 to extend dwindling IPv4 space
    - Originally introduced to slow down the depletion of available IP address space
    - No guaranteed private address will not leak, requires use of filters

- **FOUR TYPES OF ADDRESSES**
    - **Inside Local (IL)**
        - Addresses assigned to inside devices
        - Not advertised to the outside
    - **Inside Global (IG)**
        - Addresses by which inside devices are known to the outside
    - **Outside Global (OG)**
        - Addresses assigned to outside devices
        - These addresses are not advertised
    - **Outside Local (OL)**
        - Addresses by which outside devices are known to the inside
- **VERIFICATION**
    - `show ip nat translations`
- **INSIDE/OUTSIDE CONFIGURATION STEPS**
    - `Router(config)# interface FastEthernet1/0`
    - `Router(config-if)# ip nat inside`
    - `Router(config)# interface Serial 2/0`
    - `Router(config-if)# ip nat outside`
- **STATIC NAT**
    - One-to-One Mappings
    - **Configuration Example:**
        - `Router(config)# ip nat inside source static 10.1.1.3 204.15.87.1`
        - `Router(config)# ip nat inside source static 10.1.2.2 204.15.87.2`
- **DYNAMIC NAT**
    - One-to-Many Mappings
    - Three Steps:
        - Define a pool of outside global addresses
        - Define an access list with the inside local addresses
        - Link the pool and access list together with a source list
    - **Configuration Example:**
        - `Router(config)# ip nat pool PoolOne 204.15.86.1 204.15.86.254 prefix 24`
        - `Router(config)# access-list 1 permit 10.1.1.0 0.0.0.255`
        - `Router(config)# ip nat inside source list 1 pool PoolOne`
    - **Alternate Pool Configurations:**
        - *Using netmask instead of prefix*
        - `ip nat pool PoolOne 204.15.86.1 204.15.86.254 netmask 255.255.255.0`
        - *Matching host address (10.1.1.23 -> 204.15.86.23)*
        - `ip nat pool PoolOne 204.15.86.1 204.15.86.254 prefix 24 type match-host`
- **PAT A.K.A. *"Port Address Translation"***
    - Many-to-Many Mappings
    - Allows you to map on a per port basis rather than per IP
    - **Configuration Example:**
        - *Using the overload keyword in the source list command enables PAT*
        - `Router(config)# ip nat inside source list 1 interface Serial0 overload`

6.7 Describe IPv6 NAT

6.7.a NAT64

- **GENERAL CONCEPT**
    - Commonly used to translate between private and public IPv4 address space
    - Replaced NAT-PT as it had tight coupling with DNS in order to work
    - **AFT A.K.A.** ***"Address Family Translation"***
        - Provides communication between IPv6-only and IPv4-only networks
        - Translation is not a long-term strategy, ultimate goal is native IPv6
    - Advantages over tunneling:
        - Translation provides a means of gradual and seamless migration to IPv6
        - Content providers can provide services transparently to IPv6 users
- **STATELESS NAT64**
    - No bindings or session state maintained
    - Not recommended and is generally considered useless
    - IPv4 address of destination embedded in IPv6 address
    - Specific IPv6 range represents IPv4 systems within IPv6 world
    - Allows bidirectional reachability of v6 to / from v4 hosts
- **STATEFUL NAT64**
    - **Three Components:**
        - **NAT64 Prefix**
            - Any /32 ~ /96 prefix used with a converted IPv4 address
            - NAT64 prefix can be network-specific (NSP) or well-known (WSP)
            - NSP is assigned by the organization and manually configured
            - WKP for NAT64 is 64:ff9b::/96
            - WKP is prepended to the converted IPv4 if NSP is not configured
        - **DNS64 Server**
            - Functions as a normal DNS server for IPv6 AAAA records
            - Attempts to locate an IPv4 record if AAAA record is not available
            - If an A record is located, DNS64 converts the IPv4 record to IPv6
            - This is done using the NAT64 prefix
            - Only required from 6 -> 4
        - **NAT64 Router**
            - NAT64 device advertises the NAT64 prefix into IPv6-only network
            - Perform the translation between IPv6-only and IPv4-only networks
- **CONFIGURATION STEPS**
    - `Router(config)# nat64 v6v4 static 2001:db8:feed:1::e 172.16.1.10`
    - `Router(config-if)# nat64 enable`

6.7.b NPTv6

- **NPTv6 A.K.A.** ***"Network Prefix Translation"***
    - Stateless technology alongside the deprecated NAT66
    - Concept of translation from ULA to GUA is the subject of ongoing debate

    - NAT does not provide security and with IPv6 address depletion is not an issue
    - Useful mechanism for implementing address independence in an IPv6 network
    - The translation occurs as a 1:1 relationship between the IPv6 addresses
        - Occurs between the inside and outside addresses
        - Only the prefix is replaced
    - Described in more depth in RFC 6296
- **CONFIGURATION STEPS**
    - *`Only works on ASRs and IOS XE`*
    - `Router(config)#` **`interface`** `GigabitEthernet1/1`
    - `Router(config-if)#` **`nat66 inside`**
    - `Router(config)#` **`interface`** `GigabitEthernet1/2`
    - `Router(config-if)#` **`nat66 outside`**
    - `Router(config)#` **`nat66 prefix inside`** `2002:AB01::/64` **`outside`** `2002:AB02::/64`

6.8 Describe SLA architecture

- **SLA A.K.A. *"Service-Level Agreements"***
    - Measures network's current performance
    - If performance does not meet threshold, PBR reacts appropriately
    - Measurement can be as simple as using ping to determine response
    - Or it can be sophisticated as measuring jitter (delay variation) of VoIP
- **SLA OPERATION SETUP**
    - When defining IP SLA operation, a receiver can be a router or a host
    - Receiver may need to be configured as IP "SLA Responder"
- **SUMMARY OF AVAILABLE OPERATION TYPES**
    - ICMP (echo, jitter)
    - RTP (VoIP)
    - TCP Connection (establishes TCP connections)
    - UDP (echo, jitter)
    - DNS
    - DHCP
    - HTTP
    - FTP

6.9 Configure and verify IP SLA

6.9.a ICMP

- **CONFIGURATION STEPS**
    - **`ip sla`** `sla-ops-number (global command)`
    - **`icmp-echo`** `{destination-ip-address | destination-hostname} ... (SLA subcommand)`
        - **`source-ip`** `{ip-address | hostname}` *`OR # optional`*
        - **`source-interface`** `interface` *`# optional`*
    - **`frequency`** `seconds (SLA subcommand)`
    - **`ip sla schedule`** `sla-ops-number ... (global subcommand)`
        - **`life`** `{`**`forever`** `| seconds}` *`OR`*

        - **start-time** {hh:mm[:ss] month day | **pending** | **now** | **after** hh:mm:ss}
        - **ageout** seconds # optional
        - **recurring** # optional
- **VERIFICATION**
    - **show ip sla configuration**
    - **show ip sla statistics**

6.10 Configure and verify tracking objects

6.10.a Tracking objects

- **CONFIGURATION STEPS**
    - **To configure track object, use the following command in global:**
        - **track** <object-name> **ip sla** <sla-ops-number> [**state** | **reachability**]
    - **To configure delay reaction, use the delay command:**
        - **delay** {**down** seconds | **up** seconds} (track subcommand)
- **VERIFICATION**
    - **show track**
    - **show ip sla statistics**

6.10.b Tracking different entities (for example, interfaces, IP SLA results)

- **CONFIGURATION STEPS**
    - **To configure a static route with a track object, use the following command:**
        - **ip route** prefix mask {**interface** | **next-hop**} **track** object-number (global)
    - **To configure PBR to use object tracking, use the following set command:**
        - **set ip next-hop verify-availability** next-hop seq **track** object-number
        - *When tracking object is down, PBR acts as if set command does not exist*

6.11 Configure and verify Cisco NetFlow

- **GENERAL CONCEPT**
    - Accounting feature on top of an existing switching path such as CEF
    - Provides statistics on packets flowing through router:
        - **INGRESS**: IP, IP to MPLS, Frame Relay, ATM
        - **EGRESS**: IP, MPLS to IP. Traffic sourced from the router not captured
    - Single flow is a combination of seven key-fields
- **FLOW PARAMETERS**
    - Source IP Address
    - Destination IP Address
    - Source Port Number
    - Destination Port Number
    - Layer 3 Protocol Type
    - Type of Service (ToS)
    - Logical Interface (ifIndex)
- **LIFE CYCLE**

- Router notices network traffic and creates flows in NetFlow cache locally
- Active flows expire
- Router exports flow-records to collector
- Expiration of Flow:
    - Inactivity Timer (15 Seconds Default)
    - Active Timer Expired (30 Minutes Default)
    - NetFlow Cache Full (FIFO)
    - TCP RST or FIN Flag Seen

- **FLOW PROCESSING**
    - **Pre Processing**
        - **Packet Sampling**
            - Statistical sampling of network traffic
            - Set up for traffic engineering or capacity planning
            - E.G. 1 in 100 packets
        - **Filtering**
            - Sets up specific subset of network traffic for class-based analysis
    - **Post Processing**
        - **Aggregation Schemes**
            - Sets up extra aggregation caches
            - Different combinations of fields determine which flows are grouped
        - **Export to multiple destinations**
            - Sets up identical streams of NetFlow data to be sent
- **GOOD TO KNOW TERMINOLOGY**
    - **Flow**: Actual traffic that occurred
    - **Flow-Record**: Info about that traffic
    - **Export-Packet**: Transport to the collector
    - **Packet Header**: Has version, sequence number, and more
    - **Schema**: Cognitive framework that helps organize and interpret info
    - **Template Record (Schema):** Defines fields and structure in Flow-Record
    - **Options Template Record (Schema):** Defines fields and structures in Options
    - **Options-Record (Data):** Configuration of the Netflow, such as sampling

6.11.a Netflow v5, v9

- **VERSION 5**
    - Fixed format, cannot be extended or added to
    - IPv4 only
    - Added BGP AS information and sequence numbers
    - Export data from main cache only
    - No real concept of ‘ingress’ and ‘egress’ flows
    - Collector engine reverses the info behind the scenes without additional config
- **VERSION 9**
    - Add additional information to flows and template based
    - Added support for MPLS, BGP next-hop, and IPv6 headers

- Exports data from main and aggregation cache

6.11.b Local retrieval

- **GENERAL CONFIGURATION**
  - `Router(config)#` **`snmp-server community public ro`**
  - `Router(config)#` **`ip flow-capture icmp`**
  - `Router(config)#` **`ip flow-capture`** `{capture-type}`
  - `Router(config)#` **`ip multicast netflow rpf-failure`**
  - `Router(config)#` **`ip flow-cache entries`** `1024`
  - `Router(config)#` **`ip flow-cache timeout active`** `15`
  - `Router(config)#` **`ip flow-cache timeout inactive`** `30`
  - `Router(config)#` **`interface`** `fa0/0`
  - `Router(config-if)#` **`ip flow ingress`**
  - `Router(config-if)#` **`ip flow egress`**
- **GENERAL VERIFICATION**
  - **`show ip flow interface`**
  - **`show ip cache flow`**
  - **`show ip cache verbose flow`**
- **SAMPLING CONFIGURATION**
  - `Router(config)#` **`flow-sampler-map`** `{sampler-name}`
  - `Router(config-sampler)#` **`mode random one-out-of`** `10`
  - `Router(config-if)#` **`flow-sampler`** `{sampler-name}` **`egress`**
- **TOP TALKERS CONFIGURATION**
  - `Router(config)#` **`ip flow-top-talkers`**
  - `Router(config-flow-top-talkers)#` **`top`** `10`
  - `Router(config-flow-top-talkers)#` **`sort-by packets`**
  - `Router(config-flow-top-talkers)#` **`match protocol`** `1`
- **TOP TALKERS VERIFICATION**
  - **`show ip flow top-talkers`**

6.11.c Export (configuration only)

- **GENERAL CONFIGURATION**
  - `Router(config)#` **`ip flow-export destination`** `{ip-address} {port}`
  - `Router(config)#` **`ip flow-export version`** `{5 | 9}`
- **AGGREGATION CONFIGURATION**
  - `Router(config)#`**`ip flow-aggregation cache destination-prefix`**
  - `Router(config-flow-cache)#` **`export destination`** `{ip-address} {port}`
  - `Router(config-flow-cache)#` **`export version 9`**
  - `Router(config-flow-cache)#` **`cache entries`** `1024`
  - `Router(config-flow-cache)#` **`mask destination minimum`** `26`
  - `Router(config-flow-cache)#` **`enabled`**
- **VERIFICATION**
  - **`show ip flow export`**
  - **`show ip flow export template`**
  - **`show ip cache flow aggregation destination-prefix`**