Q - Can DNS use TCP? In which cases DNS uses TCP?
A - DNS uses TCP for Zone Transfer over Port: 53 TCP is used if the size of the packet goes over 512 bytes. Zone Transfer is nothing but one of the many mechanisms available for administrators to replicate DNS databases across a set of DNS servers. It is necessary to maintain a **consistent** DNS database between DNS Servers. This is achieved by the TCP protocol. This communication happens between DNS Servers only. The connection is established between the DNS Server to transfer the zone data and Source and Destination DNS Servers will make sure that data is consistent by using TCP ACK bit.

DNS uses UDP for DNS Queries over Port: 53
A client computer will always send a DNS Query using UDP Protocol over Port 53. The query and response work can be done in one RTT. So this approach is fast as compared to TCP as in case of TCP, connection has to established first. If a client computer does not get response from a DNS Server, it must re-transmit the DNS Query using the TCP after 3-5 seconds of interval.

Q - How does BGP work?
A - BGP is an Exterior Gateway Routing Protocol and is used to do routing between ASs or ISPs. Has 4 message types: Open(Open a BGP session with the peer), Keeplalive(duh!), Update(Routing updates) and Notification(Bad!). Main reason to use BGP is influence inbound or outbound(mostly) routes. This means giving preference to one path over other depending on various factors and policies. Has a long metric list. N-W-L-Li-AS path-O-M-N-I. Has two flavors. eBGP and iBGP.

TSHOOT - Before Neighbors:
Loopback reachability, Neighbor command verification for right AS, update source

TSHOOT - After Neighbors (Routes not showing up in the RT):
Next hop self, BGP Synchronization

Q - Define bandwidth
A - bandwidth is technically the bit-rate of available data capacity of a network. The width of the band limits the data rate that can be carried on the medium. Bandwidth is measured in kilobits per second or “Kbps.”

Q - Define throughput
A - bit rate (bits/time unit) at which bits are transferred between sender/receiver. Only the data that is successfully delivered to the sender is accounted for. Throughput is measured over a given time

Q - Two switching techniques
A - **Store-and-Forward:** Store-and-Forward switching will wait until the entire frame has arrived prior to forwarding it. This method stores the entire frame in memory. So if there are 3 links between the source and the destination, the transmission delay Dtr will be: 3L/R (rate of transmission for 3 links = 3* * 1/R) Once the frame is in memory, the switch checks the destination address, source address, and the CRC. If no errors are present, the frame is forwarded to the appropriate port. This process ensures that the destination network is not affected by corrupted or truncated frames.

Cut-Through: Cut-Through switching will begin forwarding the frame as soon as the destination address is identified. The difference between this and Store-and-Forward is that Store-and-Forward receives the whole frame before forwarding.Since frame errors cannot be detected by reading only the destination address, Cut-Through may impact network performance by forwarding corrupted or truncated frames. These bad frames can create broadcast storms wherein several devices on the network respond to the corrupted frames simultaneously.

Q - DISADV of Packet Switching
A - Queuing of packets can lead to excessive congestion. Which leads to Packet Delay and Loss. We need congestion control mechanisms.

Q - Types of delay
A - D_Node = D_proc(check bit errors, determine output link) + D_queue(router buffer delay) + D_trans(L/R where L is the length of the packet in bits and R is the rate of transmission) + D_prop(length of physical link/medium propagation speed)

Q - why we need checksum in both layer-2 and layer-3 of OSI layer
A - Simply put, different layers of the OSI model have checksums so you can assign blame appropriately. Suppose there is a web server running on some system (assume TCP port 80, i.e. OSI Layer 4) Suppose there is a software error in the Operating System of that web server that corrupts certain IP payloads. (IPv4 OSI Layer 3)

If we only rely on Ethernet (i.e. OSI Layer 2) checksums, then that error would go un-noticed until something crashes or throws an error, because the Ethernet NIC would simply transmit the (already corrupted) data that it received from the Operating System IP stack. However, if TCP, IP and Ethernet all have checksums, we can isolate the layer where the error occurred, and notify the appropriate Operating System or application component of the error.

Q - What is IP Fragmentation? What are its disadvantages? How to avoid fragmentation?
A - The process of breaking the datagrams into smaller pieces so that they can be transmitted on a link with a smaller **MTU** (MTU of a layer is the size of the largest PDU that the layer/link can pass/transfer) than the original datagram size.

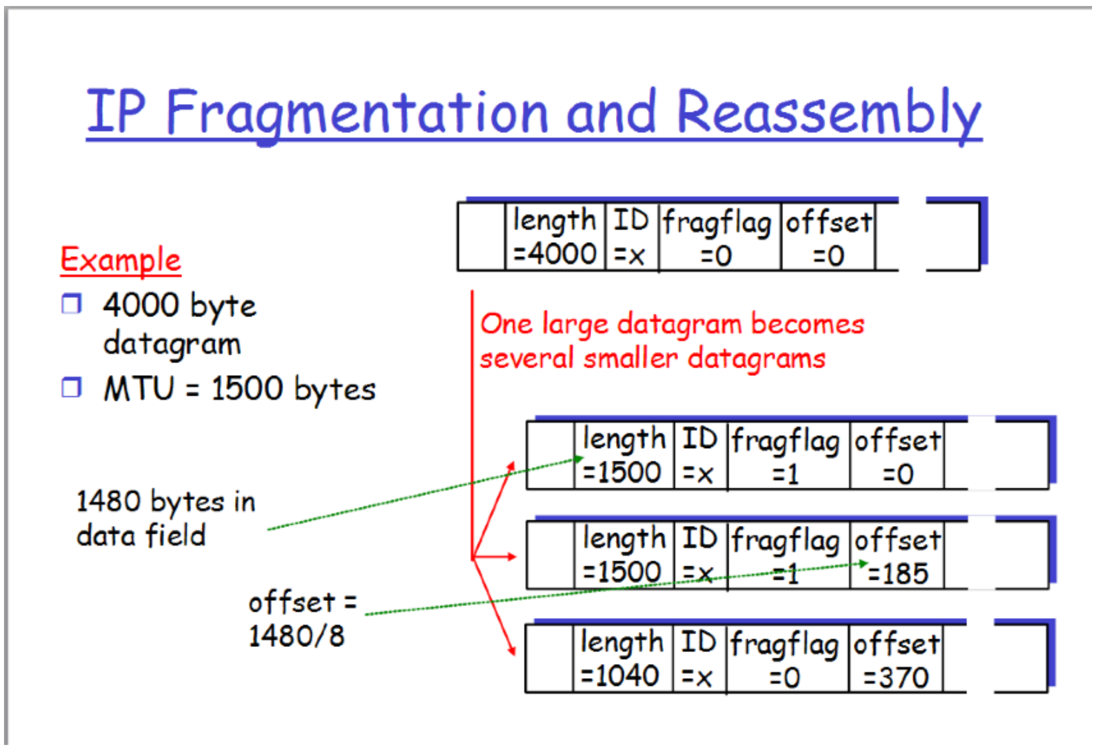
MSS is the largest amount of data that a communication device can receive in a single TCP segment. This does **NOT** count the TCP header and the IP header (20 bytes each).

Small MSS would mean reduction or elimination of the need of IP fragmentation. This is because a TCP segment will be packed inside an IP datagram. IP datagrams have their own size issues (MTU). If the segment size is large, the IP datagram will be large too which would increase the need of IP fragmentation.

Why divide by 8?

The length field is 16 bits (16 to 31) and the offset field is 13 bits. So to account for those 3 bits, 2^16/2^13 = 8

So that gives 2^13 = 8192 possible offset values. At the same time, the total length field is 16 bits, so technically the max payload size is 2^16 - 20 = 65516 bytes.
Of course the fragment offset field must be able to span the entire spectrum of possible sizes, so 65536 / 8192 = 8.


IP Fragmentation and Reassembly
Example
4000 byte datagram
MTU = 1500 bytes
length =4000 | ID =x | fragflag =0 | offset =0
One large datagram becomes several smaller datagrams
length =1500 | ID =x | fragflag =1 | offset =0
length =1500 | ID =x | fragflag =1 | offset =185
length =1040 | ID =x | fragflag =0 | offset =370
1480 bytes in data field
offset = 1480/8

Fragmentation Example walkthrough

MTU = 1500 bytes. Total datagram size = 4000 bytes. Now say max PDU size that can travel on the IP layer is 1500. This will include 20 byte IP header and 20 byte Transport header (this also means that MSS in this case is 1460). Each fragment needs to be fabricated in a way such that this 20 byte IP header is attached to all. So for one fragment, we have 1480 bytes (payload + transport header) and 20 bytes IP header. Next offset will be 1480/8. Why 8? (we already know by this point, if not GTFO)

Disadvantage of IP fragmentation:

1. There is a small increase in CPU and memory overhead to fragment an IP datagram. This holds true for the sender as well as for a router in the path between a sender and a receiver. Creating fragments simply involves creating fragment headers and copying the original datagram into the fragments. IP fragmentation can cause excessive retransmissions when fragments encounter packet loss and reliable protocols such as TCP must retransmit all of the fragments in order to recover from the loss of a single fragment. Fragmentation causes more overhead for the receiver when reassembling the fragments because the receiver must allocate

memory for the arriving fragments and coalesce them back into one datagram after all of the fragments are received.

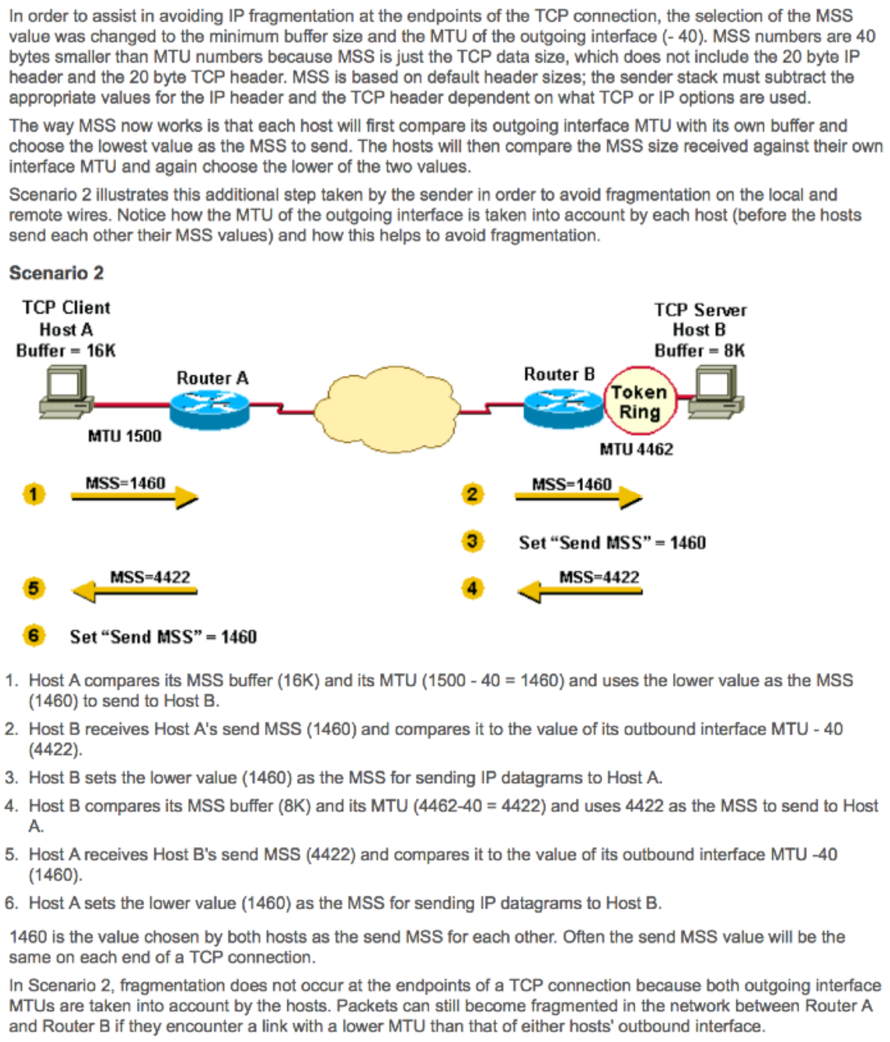
Reassembly on a host is not considered a problem because the host has the time and memory resources to devote to this task. But, reassembly is very inefficient on a router whose primary job is to forward packets as quickly as possible. A router is not designed to hold on to packets for any length of time. Also a router that does reassembly chooses the largest buffer available (18K) with which to work because it has no way to know the size of the original IP packet until the last fragment is received.

2. Another fragmentation issue involves how dropped fragments are handled. If one fragment of an IP datagram is dropped, then the entire original IP datagram must be resent,
and it will also be fragmented.

Senders typically use two approaches to decide the size of IP datagrams to send over the network.
a) The first is for the sending host to send an IP datagram of size equal to the MTU of the first hop of the source destination pair.
b) The second is to run the path **MTU discovery (PMTUD) algorithm:**

TCP MSS as described earlier takes care of fragmentation at the two endpoints of a TCP connection, but it does not handle the case where there is a smaller MTU link in the middle between these two endpoints. PMTUD was developed in order to avoid fragmentation in the path between the endpoints. It is used to dynamically determine the lowest MTU along the path from a packet's source to its destination.
For IPv4 packets, Path MTU Discovery works by setting the Don't Fragment (DF) flag bit in the IP headers of outgoing packets. Then, any device along the path whose MTU is smaller than the packet will drop it, and send back an Internet Control Message Protocol (ICMP) Fragmentation Needed (Type 3, Code 4) message containing its MTU, allowing the source host to reduce its Path MTU appropriately. The process is repeated until the MTU is small enough to traverse the entire path without fragmentation.

IPv6 routers do not support fragmentation or the Don't Fragment option. For IPv6, Path MTU Discovery works by initially assuming the path MTU is the same as the MTU on the link layer interface where the traffic originates. Then, similar to IPv4, any device along the path whose MTU is smaller than the packet will drop the packet and send back an ICMPv6 Packet Too Big (Type 2) message containing its MTU, allowing the source host to reduce its Path MTU appropriately. The process is repeated until the MTU is small enough to traverse the entire path without fragmentation.

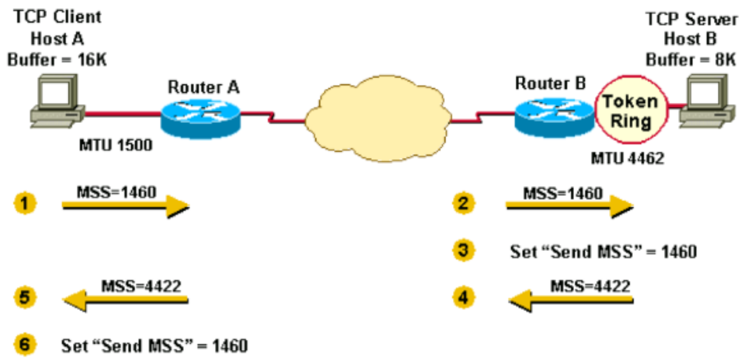


In order to assist in avoiding IP fragmentation at the endpoints of the TCP connection, the selection of the MSS value was changed to the minimum buffer size and the MTU of the outgoing interface (- 40). MSS numbers are 40 bytes smaller than MTU numbers because MSS is just the TCP data size, which does not include the 20 byte IP header and the 20 byte TCP header. MSS is based on default header sizes; the sender stack must subtract the appropriate values for the IP header and the TCP header dependent on what TCP or IP options are used.

The way MSS now works is that each host will first compare its outgoing interface MTU with its own buffer and choose the lowest value as the MSS to send. The hosts will then compare the MSS size received against their own interface MTU and again choose the lower of the two values.

Scenario 2 illustrates this additional step taken by the sender in order to avoid fragmentation on the local and remote wires. Notice how the MTU of the outgoing interface is taken into account by each host (before the hosts send each other their MSS values) and how this helps to avoid fragmentation.

Scenario 2



1. Host A compares its MSS buffer (16K) and its MTU (1500 - 40 = 1460) and uses the lower value as the MSS (1460) to send to Host B.
2. Host B receives Host A's send MSS (1460) and compares it to the value of its outbound interface MTU - 40 (4422).
3. Host B sets the lower value (1460) as the MSS for sending IP datagrams to Host A.
4. Host B compares its MSS buffer (8K) and its MTU (4462-40 = 4422) and uses 4422 as the MSS to send to Host A.
5. Host A receives Host B's send MSS (4422) and compares it to the value of its outbound interface MTU -40 (1460).
6. Host A sets the lower value (1460) as the MSS for sending IP datagrams to Host B.

1460 is the value chosen by both hosts as the send MSS for each other. Often the send MSS value will be the same on each end of a TCP connection.

In Scenario 2, fragmentation does not occur at the endpoints of a TCP connection because both outgoing interface MTUs are taken into account by the hosts. Packets can still become fragmented in the network between Router A and Router B if they encounter a link with a lower MTU than that of either hosts' outbound interface.

Fig: How end to end fragmentation is avoided

Large Segment Offload or (TSO - TCP segment offload)

When a system needs to send large chunks of data out over a computer network, the chunks first need breaking down into smaller segments that can pass through all the network elements like routers and switches between the source and destination computers. This process is referred to as segmentation. Often the TCP protocol in the host computer performs this segmentation. Offloading this work to the NIC is called TCP segmentation offload (TSO).

For example, a unit of 64kB (65,536 bytes) of data is usually segmented to 46 segments of 1448 bytes each before it is sent through the NIC and over the network. With some intelligence in the NIC, the host CPU can hand over the 64 KB of data to the NIC in a single transmit-request, the NIC can break that data down into smaller segments of 1448 bytes, add the TCP, IP, and data link layer protocol headers - according to a template provided by the host's TCP/IP stack - to each segment, and send the resulting frames over the network. This significantly reduces the work done by the CPU. As of 2014 many new NICs on the market support TSO.

What is the ROMMON Mode?

A - The ROM Monitor is a bootstrap program that initializes the hardware and boots the Cisco IOS XR software when you power on or reload a router. A version of the ROM Monitor software exists on each card. If the Cisco IOS XR software cannot boot on a card, the card startup ends in ROM Monitor mode

MPLS Maximum Receive Unit

Maximum receive unit (MRU) is a parameter that Cisco IOS uses. It informs the LSR how big a received labeled packet of a certain FEC can be that can still be forwarded out of this LSR without fragmenting it. This value is actually a value per FEC (or prefix) and not just per interface. The reason for this is that labels can be added to or removed from a packet on an LSR.

Think of the example of a router in which all the interfaces have an MTU of 1500 bytes. This means that the biggest IP packet that can be received and transmitted on all interfaces is 1500 bytes. Imagine that the packets can be labeled by adding a maximum of two labels. (Typically, MPLS VPN and AToM networks label the packets respectively the frames with two labels.) Also assume that the MPLS MTU is set to 1508 on all links to accommodate for the extra 8 bytes (2 times 4 bytes - size of the MPLS header) for the labels. A labeled packet that is transmitted on any of the links can now be 1508 bytes. The label operation plays a role in determining the MRU. Because the label operation is determined per FEC or prefix, the MRU can change per FEC or prefix.

```
lactometer#show mpls forwarding-table 10.200.254.2 detail
Local  Outgoing    Prefix            Bytes tag  Outgoing   Next 
Hop
tag    tag or VC   or Tunnel Id      switched   interface
21     Pop tag     10.200.254.2/32   0          Et0/0/0    
10.200.200.2
        MAC/Encaps=14/14, MRU=1512, Tag Stack{}
        00604700881D00024A4008008847
        No output feature configured
```

SDN Architecture

A- **SDN** - The separation of the control plane and the data plane of a network. The **Controller** in Control Plane is the brain of the Networking Device. Rather than each device having its own brain, the Controller is a single brain controlling hundreds of devices.

Benefits - Easier management(policies). The Controller can push centralized polices and configuration down to the data plane. An application programmer can create an application that commands the Controller through the REST or JAVA API(North Bound Communication using the North Bound API). The controller then uses OpenFlow(South Bound API) to update the flow tables in the switches giving them instruction of the flow.(South Bound Communication). OpenFlow only updates the flow tables. Does not install routing protocols.

OpenFlow: is an open source communication interface that allows access and manipulation of the data plane of a device (both physical and hypervisor) via a controller. It sits between the controller and the physical devices and install flow tables on to the devices thus orchestrating the flow of the traffic. The Flow Table contains Flow Entries. Controller to device communication is facilitated by Southbound APIs (OpenFlow protocol, NetConf, SNMP)

3 Layers of SDN Architecture:
1. Application Layer - Where the Application resides. This application talks to the Controller using the REST or JAVA API
2. Control Layer - The Control Plane where the Controller resides. Receives communication from the application(North) and sends instructions to the Infrastructure layer via the OpenFlow
3. Infrastructure Layer - The Data Plane. Performs forwarding.

East West Communication - Controller X talks to Controller Y in the Control Layer.

Basis for OpenFLow:

1. First, there must be a common logical architecture in all switches, routers, and other network devices to be managed by an SDN controller. This logical architecture may be implemented in different ways on different vendor equipment and in different types of network devices, so long as the SDN controller sees a uniform logical switch function.
2. Second, a standard, secure protocol is needed between the SDN controller and the network device.

Table types:

The OpenFlow specification defines three types of tables in the logical switch architecture. A **Flow Table** matches incoming packets to a particular flow and specifies the functions that are to be performed on the packets. There may be multiple flow tables that operate in a pipeline fashion, as explained subsequently.

A flow table may direct a flow to a **Group Table**, which may trigger a variety of actions that affect one or more flows.

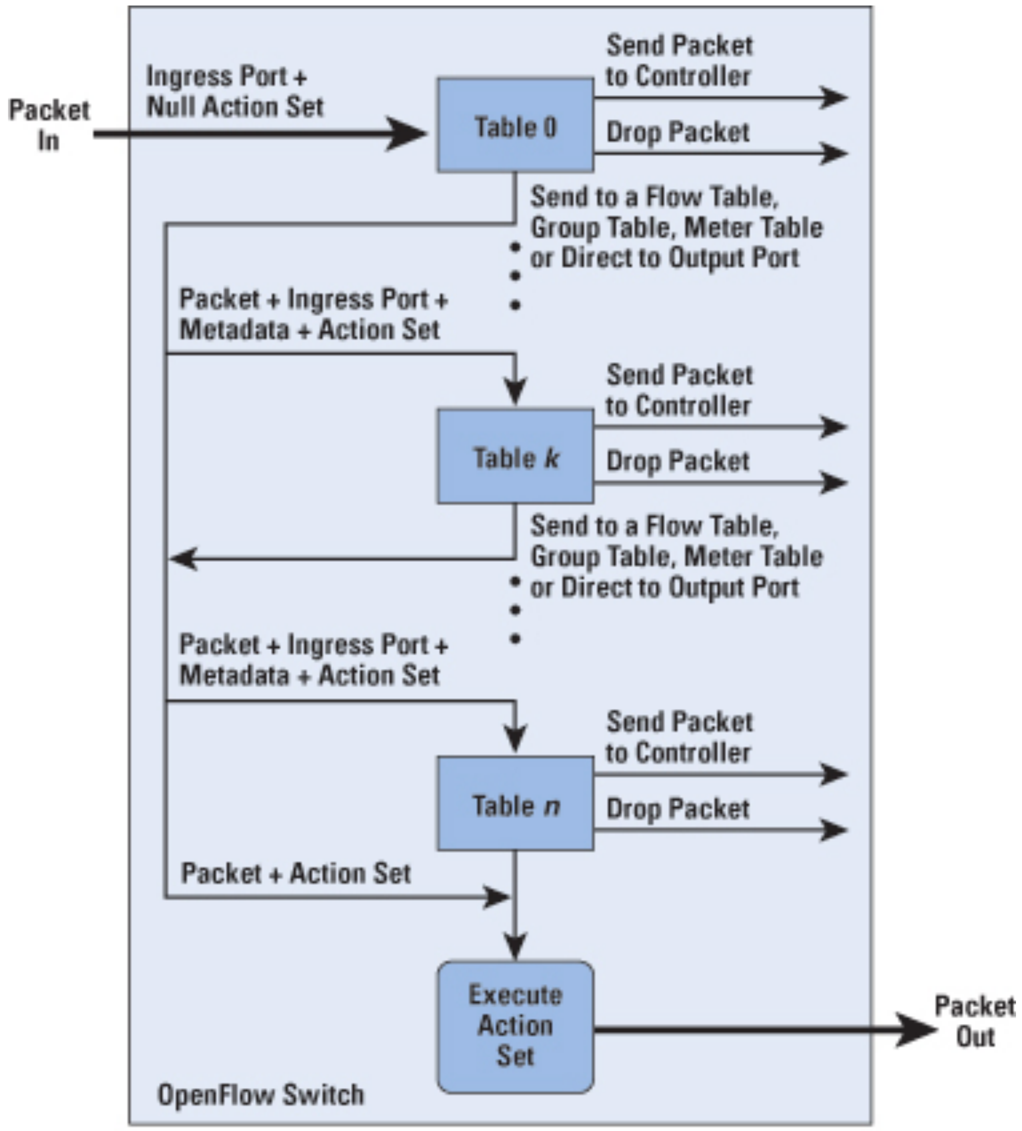
A **Meter Table** can trigger a variety of performance-related actions on a flow.

Components of a Flow-Table: (Highlighted are the important ones)

A **flow table** is the basic building block of the logical switch architecture. It has stuff like ***Match Fields*** (used to select packets that match the values in the fields), *Priority, Counters* (updated for matching packets), ***Instructions*** (contains actions/action sets to be performed on the matched packets), *Timeouts* (max time before a flow is rendered expired) etc.

The match fields can match stuff like *ingress port, Ethernet SA and DA, IPv4/v6 SA and DA, TCP/UDP source port/dest port.* It may also support matching of *VLAN ID, ICMP type and code, MPLS label etc.*
The **instructions** contain ***Action*** and ***Action set***. Actions describe packet forwarding, packet modification, and group table processing operations. The actions can be of type *Output, Set-Queue ID, Group (process the packet through specific group), push-tag/pop-tag, change-TTL etc.*

Action set is list of actions associated with a packet that are accumulated while the packet is processed by each table. See figure below:


Packet In
Ingress Port + Null Action Set
Table 0
Send Packet to Controller
Drop Packet
Send to a Flow Table, Group Table, Meter Table or Direct to Output Port
Packet + Ingress Port + Metadata + Action Set
Table k
Send Packet to Controller
Drop Packet
Send to a Flow Table, Group Table, Meter Table or Direct to Output Port
Packet + Ingress Port + Metadata + Action Set
Table n
Send Packet to Controller
Drop Packet
Packet + Action Set
Execute Action Set
Packet Out
OpenFlow Switch

OpenFlow Protocol

The OpenFlow protocol describes message exchanges that take place between an OpenFlow controller and an OpenFlow switch. Typically, the protocol is implemented on top of SSL or Transport Layer Security (TLS), providing a secure OpenFlow channel.

The OpenFlow protocol enables the controller to perform add, update, and delete actions to the flow entries in the flow tables. It supports three types of messages, as shown in Table 1.

- **Controller-to-Switch:**These messages are initiated by the controller and, in some cases, require a response from the switch. This class of messages enables the controller to manage the logical state of the switch, including its configuration and details of flow- and group-table entries. Also included in this class is the **Packet-out** message. This message is used when a switch sends a packet to the controller and the controller decides not to drop the packet but to direct it to a switch output port.
- **Asynchronous:** These types of messages are sent without solicitation from the controller. This class includes various status messages to the controller. Also included is the **Packet-in** message, which may be used by the switch to send a packet to the controller when there is no flow-table match.
- **Symmetric:** These messages are sent without solicitation from either the controller or the switch. They are simple yet helpful. **Hello** messages are typically sent back and forth between the controller and switch when the connection is first established. **Echo** request and reply messages can be used by either the switch or controller to measure the latency or bandwidth of a controller-switch connection or just verify that the device is operating. The Experimenter message is used to stage features to be built into future versions of OpenFlow.

OpenFlow Messages

Message	Description
	Controller-to-Switch
Features	Request the capabilities of a switch. Switch responds with a features r specifies its capabilities.
Configuration	Set and query configuration parameters. Switch responds with parame
Modify-State	Add, delete, and modify flow/group entries and set switch port properti
Read-State	Collect information from switch, such as current configuration, statistic capabilities.
Packet-out	Direct packet to a specified port on the switch.
Barrier	Barrier request/reply messages are used by the controller to ensure m dependencies have been met or to receive notifications for completed
Role-Request	Set or query role of the OpenFlow channel. Useful when switch conne controllers.
Asynchronous-Configuration	Set filter on asynchronous messages or query that filter. Useful when s to multiple controllers.
	Asynchronous
Packet-in	Transfer packet to controller in case there is no flow match.
Flow-Removed	Inform the controller about the removal of a flow entry from a flow tabl
Port-Status	Inform the controller of a change on a port.
Error	Notify controller of error or problem condition.
	Symmetric
Hello	Exchanged between the switch and controller upon connection startup
Echo	Echo request/reply messages can be sent from either the switch or the they must return an echo reply.
Experimenter	For additional functions.

SDN Domains

In a large enterprise network, the deployment of a single controller to manage all network devices would prove unwieldy or undesirable. A more likely scenario is that the operator of a large enterprise or carrier network divides the whole network into numerous non-overlapping SDN domains as shown in Figure:

Reasons for using SDN domains include the following:

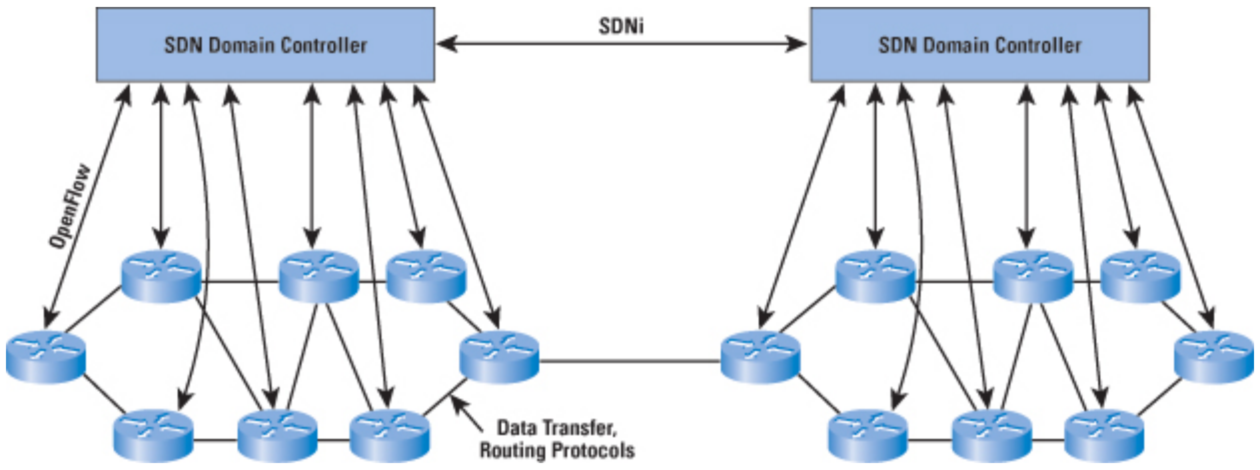
Scalability: The number of devices an SDN controller can feasibly manage is limited. Thus, a reasonably large network may need to deploy multiple SDN controllers.

Privacy: A carrier may choose to implement different privacy policies in different SDN domains. For example, an SDN domain may be dedicated to a set of customers who implement their own highly customized privacy policies, requiring that some networking information in this domain (for example, network topology) not be disclosed to an external entity.

Incremental deployment: A carrier's network may consist of portions of traditional and newer infrastructure. Dividing the network into multiple, individually manageable SDN domains allows for flexible incremental deployment.

The existence of multiple domains creates a requirement for individual controllers to communicate with each other via a standardized protocol to exchange routing information. This is done through **SDNi** (interfacing SDN domain controllers)

Further reading on SDNi



CONGESTION CONTROL AND FLOW CONTROL

Congestion Control: deals with measures that avoid overwhelming the network channel (congestion window, maintained by the sender)

Flow Control: deals with measures that help in avoiding overwhelming the receiver (receiver window). The sender maintains a variable called the receive window (RWND; advertised by the receiver to the sender) which gives sender the idea how much free buffer space is available at the receiver.

Logic for flow control:
- To prevent overwhelming at receiver: **LastByteRcvd - LmudastByteRead <= RcvBuffer**
- Receiver calculates RWND by: **RWND = RcvBuffer - [LastByteRcvd - LastByteRead]**
(RWND is dynamic and keeps on changing, Initially receiver advertises RcvBuffer = RWND)
- **Sender** makes sure that: **LastByteSent - LastByteAcked <= RWND**

Logic for congestion control:
- **Sender** makes sure that: **LastByteSent - LastByteAcked <= min{RWND, CWND}**

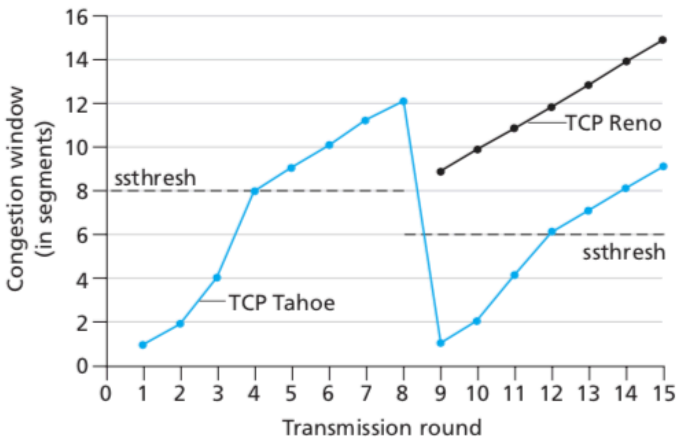
TCP slow start: When a TCP connection begins, the value of CWND is typically initialized to a small value of 1 MSS [RFC 3390], resulting in an initial sending rate of roughly MSS/RTT. This process results in a **doubling** of the sending rate every RTT. Thus, the TCP send rate starts slow but grows exponentially during the slow start phase.

When CWND >= ssthresh we are in Congestion Avoidance mode. In that phase increase the CWND 1 MSS per RTT. This goes on until a loss event occurs. When a loss event (triple duplicate ack or timeout) occurs, below happens (see table).

If a timeout event occurs, fast recovery transitions to the slow-start state after performing the same actions as in slow start and congestion avoidance: The value of cwnd is set to 1 MSS, and the value of ssthresh is set to half the value of cwnd when the loss event occurred.

Soon after a triple duplicate ack event, we enter fast Fast Recovery Retransmit (FRR) mode, the value of cwnd is increased by 1 MSS for every duplicate ACK received for the missing segment. With FRR, time is not lost waiting for a timeout in order for retransmission to begin.

Triple duplicate ACK (less severe)	**Timeout (more severe)**
Set ssthresh(SlowStart Threshold) to CWND/2	Set ssthresh to CWND/2
Set CWND to ssthresh. This is done only in Reno (TCP Fast recovery). Its set to 1 MSS in Tahoe (back to Slow Start phase)	Set CWND to 1 MSS



CONGESTION AVOIDANCE

Q - Fair Queuing and Weighted Fair Queuing (WFQ) and QoS and Congestion Topics

Both FQ and WFQ are a way (algorithms) packet scheduling takes place in networking devices like routers and switches.

Fair Queuing: The concept implies a separate data packet queue (or job queue) for each traffic flow (or for each program process) as opposed to the traditional approach with one FIFO queue for all packet flows (or for all process jobs).

WFQ: a natural generalization of fair queuing (FQ): whereas FQ shares the link's capacity in equal subparts, WFQ allows to specify, for each flow, which fraction of the capacity will be given.

WFQ can be utilized by controlling the QoS.

QoS: quality of service refers to traffic prioritization and resource reservation control mechanisms. It is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.

Protocols that are used to achieve QoS:

1. The type of service (ToS) field in the IP(v4) header (now superseded by DiffServ):
The modern redefinition of the ToS field is a six-bit Differentiated Services Code Point (DSCP) field and a two-bit Explicit Congestion Notification (ECN) field. While Differentiated Services is somewhat backwards compatible with ToS, ECN is not.

ECN: Conventionally, TCP/IP networks signal congestion by dropping packets. **But** when ECN is successfully negotiated, an ECN-aware router may set a mark in the IP header instead of dropping a packet in order to signal impending congestion. The receiver of the packet echoes the congestion indication to the sender, which reduces its transmission rate as if it detected a dropped packet.
Use of ECN on a TCP connection is optional; for ECN to be used, it must be negotiated at connection establishment by including suitable options in the SYN and SYN-ACK segments.
(the underlying network infrastructure should also support it.)

ECN is a both L3 and L4 topic which needs to be addressed at both layers. This is because congestion is only known to have happened once the packet leaves the sender and is on the network medium. So there must be an echo of the congestion indication by the receiver to the transmitter.

When both endpoints support ECN they mark their packets with ECT(0) or ECT(1). ECT stands for ECN Capable Transport. If the packet traverses an active queue management (AQM) queue (e.g., a queue that uses random early detection (RED)) that is experiencing congestion and the corresponding router supports ECN, it may change the code point to CE (Congestion Encountered) instead of dropping the packet. This act is referred to as “marking” and its purpose is to inform the receiving endpoint of impending congestion. At the receiving endpoint, this congestion indication is handled by the upper layer protocol (transport layer protocol) and needs to be echoed back to the transmitting node in order to signal it to reduce its transmission rate.
Because the CE indication can only be handled effectively by an upper layer protocol that supports it, ECN is only used in conjunction with upper layer protocols, such as TCP, that support congestion control and have a method for echoing the CE indication to the transmitting endpoint.

TCP supports ECN using two flags in the TCP header. The first, ECN-Echo (ECE) is used to echo back the congestion indication (i.e. signal the sender to reduce the amount of information it sends). The second, Congestion Window Reduced (CWR), to acknowledge that the congestion-indication echoing was received. Use of ECN on a TCP connection is optional; for ECN to be used, it must be negotiated at connection establishment by including suitable options in the SYN and SYN-ACK segments.

When ECN has been negotiated on a TCP connection, the sender indicates that IP packets that carry TCP segments of that connection are carrying traffic from an ECN Capable Transport by marking them with an ECT code point. This allows intermediate routers that support ECN to mark those IP packets with the CE code point instead of dropping them in order to signal impending congestion.

Upon receiving an IP packet with the Congestion Experienced code point, the TCP receiver echoes back this congestion indication using the ECE flag in the TCP header. When an endpoint receives a TCP segment with the ECE bit it reduces its congestion window as for a packet drop. It then acknowledges the congestion indication by sending a segment with the CWR bit set. A node keeps transmitting TCP segments with the ECE bit set until it receives a segment with the CWR bit set.

2. Differentiated services (DiffServ) (DSCP)
Differentiated services or DiffServ is a computer networking architecture that specifies a simple, scalable and **coarse-grained mechanism** for classifying and managing network traffic and providing quality of service (QoS) on modern IP networks.

DiffServ operates on the principle of **traffic classification**, where each data packet is placed into a limited number of traffic classes, rather than differentiating network traffic based on the requirements of an individual flow. **Each router on the network is configured to differentiate traffic based on its class.** Each traffic class can be managed differently, ensuring preferential treatment for higher-priority traffic on the network.

3. Integrated services (IntServ)
IntServ specifies a fine-grained QoS system, which is often contrasted with DiffServ's coarse-grained control system. There are two parts to a flow spec:

1. **What does the traffic look like?** Done in the Traffic SPECification part, also known as TSPEC.

TSPECs include **token bucket algorithm** parameters. The idea is that there is a token bucket which slowly fills up with tokens, arriving at a constant rate. Every packet which is sent requires a token, and if there are no tokens, then it cannot be sent. Thus, the rate at which tokens arrive dictates the average rate of traffic flow, while the depth of the bucket dictates how 'bursty' the traffic is allowed to be.

2. **What guarantees does it need?** Done in the service Request SPECification part, also known as RSPEC.
RSPECs specify what requirements there are for the flow: it can be normal internet 'best effort', in which case no reservation is needed. This setting is likely to be used for webpages, FTP, and similar applications.

4. Resource Reservation Protocol (RSVP) - an IntServ protocol
A protocol designed **to reserve resources across a network** for an integrated services Internet. RSVP operates over an IPv4 or IPv6 Internet Layer and **provides receiver-initiated setup of resource reservations** for multicast or unicast data flows with scaling and robustness. It does not transport application data but is similar to a control protocol, like Internet Control Message Protocol (ICMP)

5. RSVP-TE
Resource Reservation Protocol - Traffic Engineering **is an extension** of the resource reservation protocol (RSVP) for traffic engineering. It supports the reservation of resources across an IP network. Applications running on IP end systems can use RSVP to **indicate to other nodes the nature (bandwidth, jitter, maximum burst, and so forth) of the packet streams they want to receive.**
RSVP-TE generally **allows the establishment of MPLS label switched paths (LSPs)**, taking into consideration network constraint parameters such as available bandwidth and explicit hops.

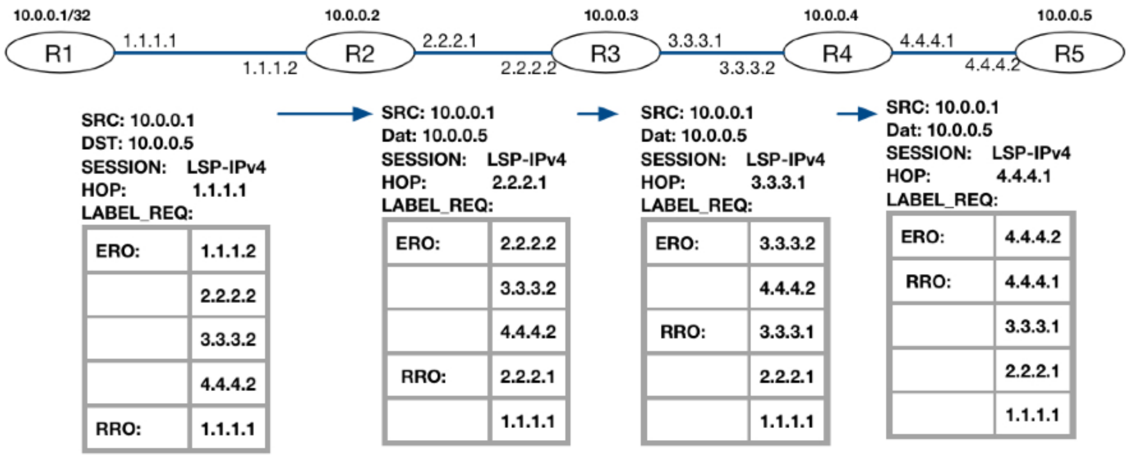
- Identification on LSP via Tunnel ID (a number which is same for prim and bkup), explicit tunnel ID (IP of the HE), LSP ID (this is different in case of backup). If the LSP-PATH uses MPLS FRR (one-to-one/Path Protection) backup, two sets of RSVP sessions belong to the same LSP-Path. One set belongs to the original protected LSP path and the other belongs to the Protection LSP. Both LSP-Paths will have the same Tunnel-ID but LSP-ID will be different

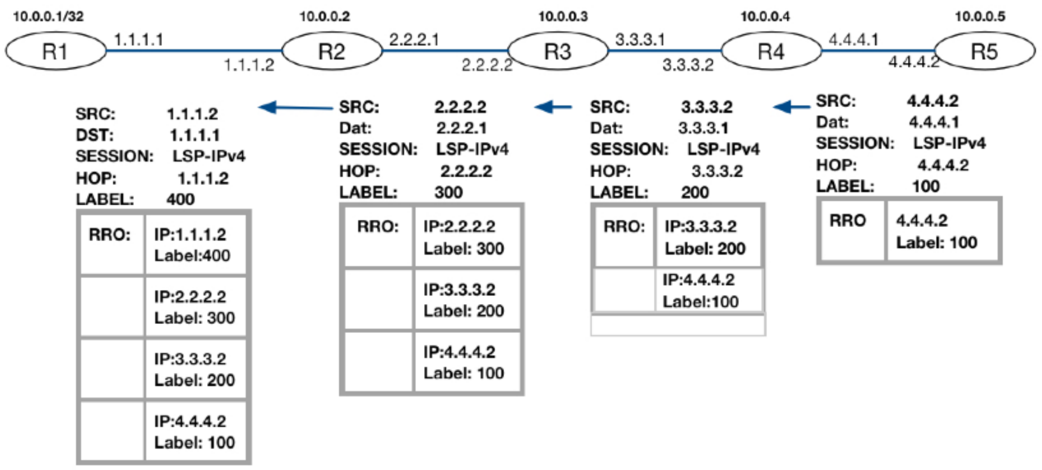
- EROs and RROs: explicit route objects and record route objects are the elements that constitute the PATH message. RRO is also in RESV. Explicit Route Objects (EROs) limit LSP routing to a specified list of LSRs. As the PATH message goes towards the downstream router, ERO list keeps on contracting and RRO list keeps on expanding (visited nodes). As the RESV messages goes from TAIL to HEAD, RRO list keeps on expanding with ingress IP and label allocated.

- **RA:** route alert: this means that the intermediate hops are allowed to inspect the PATH message

- **MESSAGE TYPES:**
  - PATH, RESV
  - PATH ERROR: (usually because of parameter problems in a path message), the router sends a unicast PathErr message to the sender that issued the path message.
  - RESV ERROR: used in case of unavailability of requested resources.
  - TEAR MESSAGES (PATH TEAR, RESV TEAR) are used to clear the PATH and RESV states from the network and hence tear down the LSP.

- Refresh Mechanism, RLK values. R: refresh timer (time the router waits before sending a refresh signal to the adjacent router) in the PATH and RESV. L: Lifetime Timer is the lifetime of RSVP sessions after which RSVP session is cleared. K: Keep Alive, is the timer for refresh





messages(number of consecutive refresh messages that can be missed before the RSVP session times out). PATH and RESV need to be flowing. RSB (RESV STATE BLOCK) and PSB are refreshed when new PATH and RESV are received. RSB maintains a relationship with the downstream node and PSB with the upstream node i.e. HE

- The path message contains information about the resources needed for the connection. Each router along the path begins to maintain information about a reservation. Each receiver host sends reservation request (RESV) messages upstream toward senders and sender applications. RESV messages must follow exactly the reverse path of path messages. RESV messages create and maintain a reservation state in each router along the way.

- When the Tail End router receives the PATH message, it allocates a label and distributes it to the upstream router via RESV message. Since RSVP-TE is unidirectional, the Tail end router doesn't need to reserve any resources. Each LSR along the path does not start resource reservation and label distribution toward the HE router until it receives a label from the downstream router (remember its Ordered Control). This process is performed by the LSR in every hop of the LSP toward the HE router. After the entire process finishes, the HE router has an egress label for this LSP-Path. Each LSR router along the route has one ingress label and one egress label in the RSVP session. The Tail router has one ingress label. All required resources (for example, bandwidth) are reserved along the path.

- After receiving the PATH message, Tail-end Router will signal RESV message back to the Headend including the label assigned. The RROs in the RESV messages contain both the IPv4 sub-object and the LABEL sub-object. The IPv4 sub object contains the router's local egress interface IP address for the message. The LABEL sub-object also contains the label generated and distributed by the local router.

- The established path remains open as long as the RSVP session is active. The session is maintained by the transmission of additional path and reservation messages that report the session state every 30 seconds.

6. Multiprotocol Label Switching (MPLS) provides eight QoS classes

DiffServ and IntServ Difference:

DiffServ is a coarse-grained, class-based mechanism for traffic management. In contrast, IntServ is a fine-grained, flow-based mechanism.

Congestion Avoidance Mechanisms in routers using queueing disciplines(Tail Drop, RED, WRED, Flow based WRED) aka Network Assisted Congestion Control:

Queuing principles: are queue management algorithms used by Internet routers, e.g. in the network schedulers, and network switches to decide when to drop packets

Tail Drop:
Tail Drop, or Drop Tail, is a very simple. In contrast to the more complex algorithms like RED and WRED, **in Tail Drop the traffic is not differentiated. Each packet is treated identically.** With tail drop, when the queue is filled to its maximum capacity, the newly arriving packets are dropped until the queue has enough room to accept incoming traffic.
PROBLEMS: Tail drop leads to the problem of **TCP Global Synchronization** - as all TCP connections "hold back" simultaneously, and then step forward simultaneously. Because each sender will reduce the transmission rate at the same time when packet loss occurs.

Random Early Detection (RED):
Random early detection (RED), also known as *random early discard* or *random early drop* addresses this problem. RED monitors the average queue size and drops (or marks when used in conjunction with ECN) packets based on statistical probabilities. If the buffer is almost empty, all incoming packets are accepted. **As the queue grows, the probability for dropping an incoming packet grows too**. When the buffer is full, the probability has reached 1 and all incoming packets are dropped.
RED is more fair than tail drop, in the sense that it does not possess a bias against bursty traffic that uses only a small portion of the bandwidth. The more a host transmits, the more likely it is that its packets are dropped as the probability of a host's packet being dropped is proportional to the amount of data it has in a queue. **Early detection helps avoid TCP global synchronization.**

PROBLEMS: RED does not accommodate quality of service (QoS) differentiation. Its variants, WRED and DWRED do.

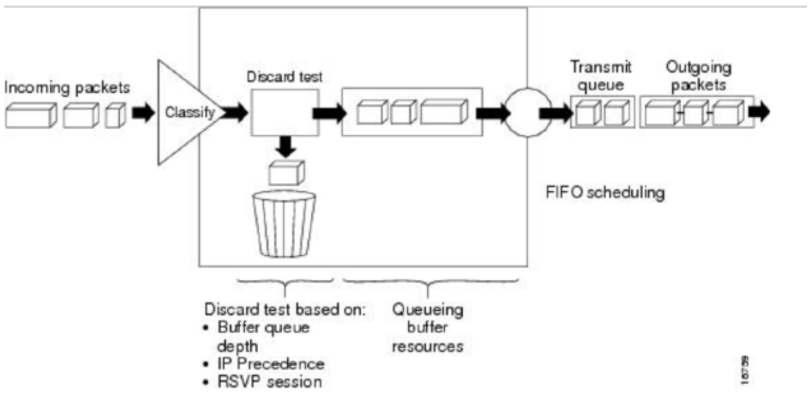
Weighted random early detection (WRED):
It is an extension to random early detection (RED) where a single queue may have several **different queue thresholds**. Each queue threshold is associated to a **particular traffic class.** WRED combines the capabilities of the RED algorithm with the IP Precedence feature to provide for preferential traffic handling of higher priority packets. WRED **can selectively discard lower priority traffic** when the interface begins to get congested and provide differentiated performance characteristics for different classes of service.
WRED **is also RSVP-aware**, and it can provide the controlled-load QoS service of integrated service.

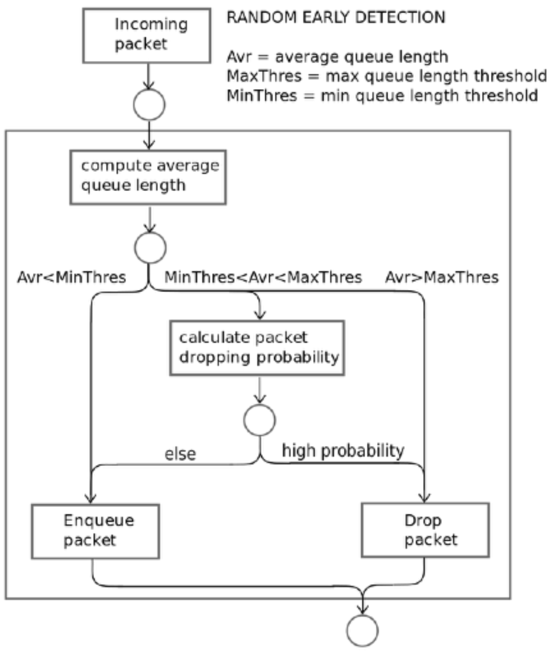
Flow Based WRED:

Flow-based WRED is a feature that forces WRED to afford **greater fairness** to all flows on an interface **in regard to how packets are dropped.** Flow-based WRED relies on the following two main approaches to remedy the problem of unfair packet drop:

•It **classifies incoming traffic into flows based on parameters** such as destination and source addresses and ports.

•It maintains state about active flows, which are flows that have packets in the output queues.




Incoming packet
RANDOM EARLY DETECTION
Avr = average queue length
MaxThres = max queue length threshold
MinThres = min queue length threshold
compute average queue length
Avr<MinThres
MinThres<Avr<MaxThres
Avr>MaxThres
calculate packet dropping probability
else
high probability
Enqueue packet
Drop packet

Some known problem terminologies with TCP

In VOQ, the physical buffer of each input port maintains a separate virtual queue for each output port. Therefore congestion on an egress port will block only the virtual queue for this particular egress port. Other packets in the same physical buffer destined to different (non-congested) output ports are in separate virtual queues and can therefore still be processed. It addresses a common problem known as **head-of-line blocking**

Buffer Bloating: Buffer bloat is the undesirable latency that comes from a router or other network equipment buffering too much data. It is a huge drag on Internet performance

TCP Head of line blocking
HTTP2 does however still suffer from another kind of HOL, namely on the TCP level. One lost packet in the TCP stream makes all streams wait until that package is re-transmitted and received. Simple example is one TCP connection trying to download 5 web images from a HTTP server and 2nd image gets lost. The rest 3 will wait until 2nd is retransmitted. This HOL is being addressed with the QUIC protocol...

QUIC is a "TCP-like" protocol implemented over UDP where each stream is independent so that a lost packet only halts the particular stream to which the lost packet belongs, while the other streams can go on.

CONGESTION AVOIDANCE IN UDP

UDP itself does not avoid congestion. Congestion control measures must be implemented at the application level.

RTP runs over UDP and RTCP(Real-time Transport Control Protocol) working with RTP provides measures for QoS(Quality of Service) like packet loss, delay, jitter, etc to report back to the sender so it knows when to slow down or change codecs.

DCCP - Datagram Congestion Control Protocol
DCCP provides a way to gain access to congestion control mechanisms without having to implement them at the application layer. It provides a congestion-controlled flow of unreliable datagrams. It allows for flow-based semantics like in Transmission Control Protocol (TCP), but does not provide reliable in-order delivery (this would mean overhead)
Challenge - TCP's congestion control is so tightly coupled to its reliable semantics that few TCP mechanisms are directly applicable without substantial change.

SCTP - Stream Control Transmission Protocol
Stream Control Transmission Protocol (SCTP) is a transport-layer protocol, serving in a similar role to the popular protocols TCP and UDP. SCTP provides some of the same service features of both: it is message-oriented like UDP and ensures reliable, in-sequence transport of messages with congestion control like TCP; it differs from these in providing multi-homing and redundant paths to increase resilience and reliability.

Multihoming support in which one or both endpoints of a connection can consist of more than one IP address, enabling transparent fail-over between redundant network paths

GBN, SR and TCP Comparison

GBN	**SR**
Simplicity and avoidance of buffering at the receiver end. Sender resends the packet once the retry timeout expires	Buffering at the receiver. Buffer all out of order packets that are received.
Cumulative ack. If packet n is received and packet n-1 was delivered to the upper layer, receiver will send an ack for n, indicating that everything till n has been received. Hence cumulative nature.	Acks are not technically cumulative as out of order packets if received without error are also Acked. So sender will receive out of order Acks. For eg. An Ack of 5 does not guarantee that packet 2 wasn't lost.
Sliding window operation	Sliding window operation
Since there is no buffering at the receiver's end, packets beginning from the sequence number of the lost packet need to be resent. This can cause a lot of packets to be resent. No out of order packet buffering. This means that if a lot of packets reside in the pipeline and a single packet error can cause all of them to be retransmitted. This happens particularly when window size and BxD product are both big. A network with huge BxD is called a Long Fat Network (LFN)	Only the packet that was lost needs to be resent as receiver buffers out of order packets. Once the lost packet is received again, all buffered packets along with this one are sent to the upper layer. Since only selective packets are retransmitted now, each packet has its own logical timer.
Window size can be one less than the sequence number space. So if sequence number space is 8 bit, it will be 2^8 - 1 = 255	Window size should be less than or half of the sequence number space. So if sequence number space is 8bit, it will be LT 128. This is to distinguish between a new packet and a retransmission.

TCP

- Selective ack in addition to the default cumulative ack
- Many TCP implementations will buffer correctly received but out-of-order segments. In a case where Ack gets lost for a packet n, GBN would re-transmit everything from n onwards (n+1, n+2, …). TCP won't do this and at most would re-transmit only n. And even not that if the ack for n+1 is received before n's timeout
- Another modification to TCP, called selective-acknowledgement lets a TCP receiver ack out-of-order segments selectively than cumulatively. Just like SR. Depends on the implementation though. Cumulative nature is by default.
- Thus TCP is a mix of GBN and SR

Next we have TCP client and server FSMs

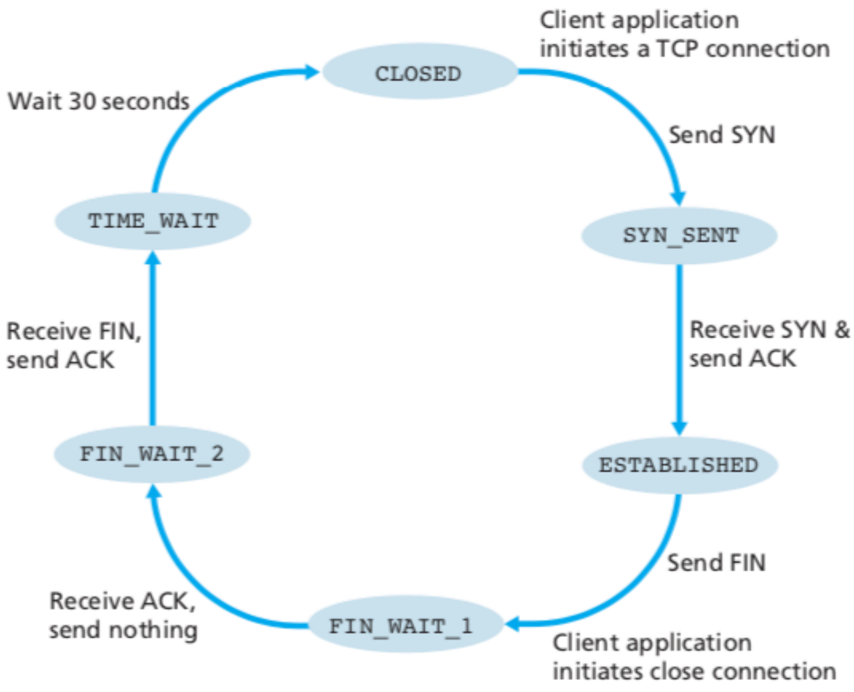

Client application initiates a TCP connection
CLOSED
Send SYN
SYN_SENT
Receive SYN & ACK, send ACK
ESTABLISHED
Send FIN
Client application initiates close connection
FIN_WAIT_1
Receive ACK, send nothing
FIN_WAIT_2
Receive FIN, send ACK
TIME_WAIT
Wait 30 seconds

Figure 3.41 ♦ A typical sequence of TCP states visited by a client TCP

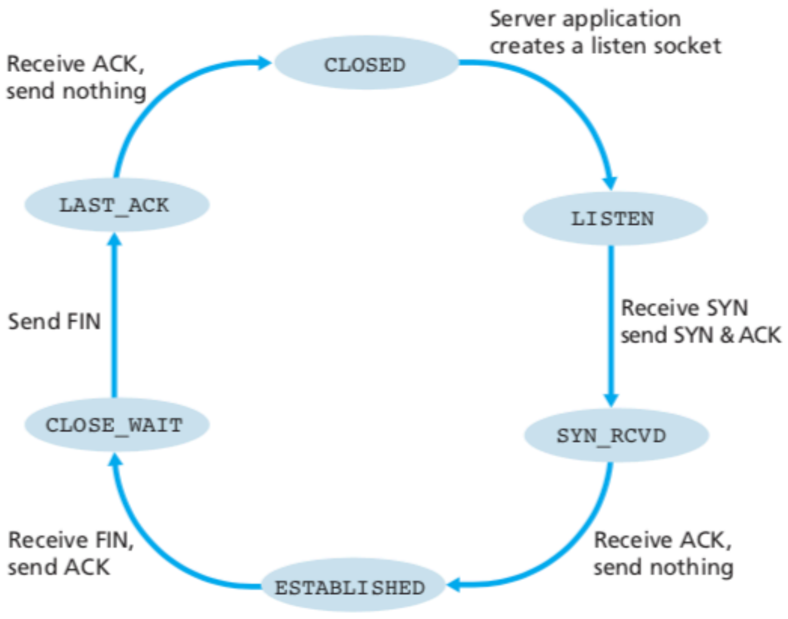

Server application creates a listen socket
CLOSED
LISTEN
Receive SYN send SYN & ACK
SYN_RCVD
Receive ACK, send nothing
ESTABLISHED
Receive FIN, send ACK
CLOSE_WAIT
Send FIN
LAST_ACK
Receive ACK, send nothing

Figure 3.42 ♦ A typical sequence of TCP states visited by a server-side TCP

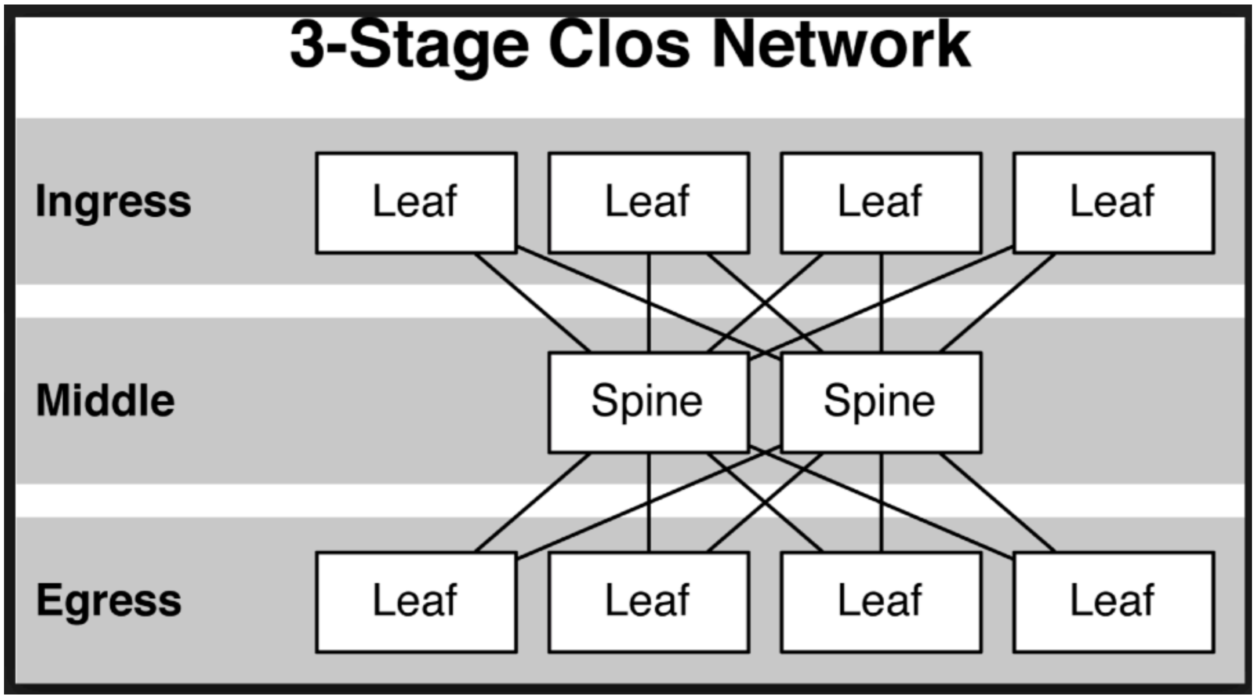
Network Topology Types

CLOS Network:

It is a type of multi-stage circuit switching network named after the inventor Charles Close. Clos networks are defined by three integers n, m, and r.

- n represents the number of sources which feed into each of r ingress stage crossbar switches.
- Each ingress stage crossbar switch has m outlets, and there are m middle stage crossbar switches. There is exactly one connection between each ingress stage switch and each middle stage switch.
- There are r egress stage switches, each with m inputs and n outputs. Each middle stage switch is connected exactly once to each egress stage switch.

Thus, the ingress stage has r switches, each of which has n inputs and m outputs. The middle stage has m switches, each of which has r inputs and r outputs. The egress stage has r switches, each of which has m inputs and n outputs.



Examples: TRILL, FabricPath, Brocade’s VCS
Benefits: The key advantage of Clos networks is that the number of crosspoint(s) (which make up each crossbar switch) required can be far fewer than would be the case if the entire switching system were implemented with one large crossbar switch.

OSPF

OSPF NEIGHBOR-SHIP STATES

1. **Down**: No hellos have been received from the neighbor for more than the dead interval
2. **Attempt:** After sending a hello but before receiving a hello from the other end
3. **Init:** Hello packet received. Also a permanent state when parameters in the hello message do not match and when one or more neighbor verification checks fail. Parameters that must match:
   - The devices must be in the same **area**
   - The devices must have the same **authentication** configuration
   - The devices must be on the same **subnet**
   - The devices **hello and dead intervals** must match
   - The devices must have matching **stub flags**

4. **Two-way:** own router ID found in received hello packet, all parameters are good, neighbor verification has passed.
5. **ExStart:** master and slave roles determined.
6. **Exchange:** database description packets (DBD) are sent.
7. **Loading:** DBDs are exchanged, now exchange of LSRs (Link state request) and LSUs (Link state update) packets will take place
8. **Full:** OSPF routers now have an adjacency and they believe that their LSDBs are identical.

OSPF LSA TYPES

- *Type 1 – Router LSA:* The Router LSA is generated by each router for each area it is located. In the LSID you will find the originating router's ID. This is to represent himself on the network and its links to other routers or networks in the same area, together with the metrics to them

- *Type 2 – Network LSA:* Network LSAs are generated by the DR. The LSID will be the router ID of the DR. Used in Broadcast and NBMA network types

- Type 3 – Summary LSA: The summary LSA is created by the ABR and flooded into other areas. Basically sending one area's Type 1 and Type 2 LSAs to other area. This summarization helps provide scalability by removing detailed topology information for other areas, because their routing information is summarized into just an address prefix and metric. LSID is the subnet being advertised

- Type 4 – Summary ASBR LSA: Other routers need to know where to find the ASBR. This is why the ABR will generate a summary ASBR LSA which will include the router ID of the ASBR in the LSID field.

- Type 5 – External LSA: also known as autonomous system external LSA: The external LSAs are generated by the ASBR. these LSAs contain information imported into OSPF from other routing processes. LSID is the subnet being advertised. These are of two types:
  - E1: metric sent by ASBR for external route (type 5) + metric to reach the ASBR (type 4 LSA)
  - E2: just the metric sent by ASBR as the total cost to get to the external destination including the cost to the ASBR

  E1 is preferred over E2

- Type 7 – NSSA External LSA: NSSA (not-so-stubby-area) doesn't allow external LSAs (type 5). To overcome this issue NSSA ASBR generates type 7 LSAs instead which remains within the NSSA. This type 7 LSA gets translated back into a type 5 by the NSSA ABR. LSID is the subnet being advertised

- SPF Calculations done only inside area when Type 1 or Type 2 LSAs change
- Prefer intra-area routes over inter-area. Use Type 1 and Type 2 LSAs to calculate intra-area routes. Use Type 1 and Type 2 to calculate the cost to reach ABR and Type 3 and Type 4 LSAs to calculate metric for inter-area routes.
- Filtering can be done at area level for Type 3 and Type 5 LSAs at ABRs and ASBRs respectively. For Type 3, IP prefix lists can be used in conjunction with area filter lists.
- To filter specific routes, we need distribute lists which can refer to IP prefix lists, ACLs or Route maps
- Similarly summarization is allowed at ABRs and ASBRs.

OSPF MESSAGE TYPES

- **Hello:** neighbor discovery, build neighbor adjacencies and maintain them.
- **DBD:** This packet is used to check if the LSDB between 2 routers is the same. The DBD is a summary of the LSDB. First master sends DBD, then slave sends DBD.
- **LSR:** Requests specific link-state records from an OSPF neighbor. Slave requests LSR
- **LSU:** Sends specific link-state records that were requested. This packet is like an envelope with multiple LSAs in it. Master sends LSA
- **LSAck:** OSPF is a reliable protocol so we have a packet to acknowledge the others.

OSPF NETWORK TYPES

TYPE	DR/BDR?	Hello Timer	Dynamic discovery of Neighbor	More than 2 routers in the subnet
Broadcast(Eth)	Yes	10	Yes	Yes
P2P	No	10	Yes	No
NBMA	Yes	30	No	Yes
P2MTP	No	30	Yes	Yes
P2MTPNB	No	30	No	Yes

OSPF AREA TYPES

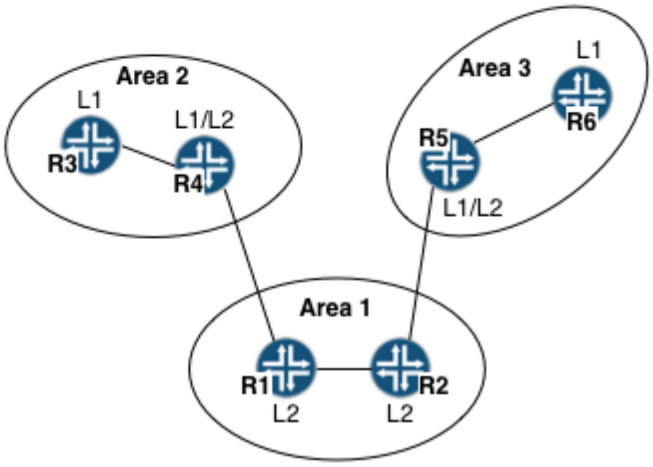
AREA TYPE	TYPE 5 ALLOWED INTO THE AREA	TYPE 3 ALLOWED INTO THE AREA	TYPE 7 ALLOWED INTO THE AREA
Stub	No	Yes	No
Totally Stubby	No	No	No
NSSA	No	Yes	Yes
Totally NSSA	No	No	Yes

OSPF PATH SELECTION PREFERENCE LIST

Intra-Area (O)
Inter-Area (O IA)
External Type 1 (E1) - includes internal cost in calculation
NSSA Type 1 (N1)
External Type 2 (E2) - does not include internal cost in calculation
NSSA Type 2 (N2)

IS-IS

Sample IS-IS topology with L1, L2 and L1/L2 routers.



IS-IS Areas

In OSPF protocol any of the router's interfaces can be assigned to a particular area, however the concept of area in IS-IS is different. Here in general, every single router belongs to an Area. The idea of this comes from the fact that IS-IS was initially created to route Connectionless Network Protocol (CLNP) where the address belongs to a device (Router), whereas in Internet Protocol (IP) the address belongs to the particular interface.
IS-IS protocol has two levels or hierarchy, Level 1 and Level 2. Level 1 corresponds OSPF intra-area routing whereas Level 2 corresponds with the OSPF backbone Area 0 routing. Level 2 areas join all the areas with the backbone area.

A Level 1 router can become adjacent with the Level 1 and Level 1-2 (L1/L2) router. A Level 2 router can become adjacent with Level 2 or Level 1-2 (L1/L2) router. There is no adjacency between L1 only and L2 only router.

IS-IS Level 1 (L1) Router

An IS-IS Level 1 router has the link state information of its own area for all the intra-area topology. In order to route packets to other areas it uses the closest Level 2 capable (L1/L2) router. Level 1 Area behaves pretty much as OSPF totally stubby area. L1 only router send L1 Hellos.

IS-IS Level 1-2 (L1/L2) Router

An IS-IS L1/L2 router maintains two link state database information. One is for Level 1 and the other for Level 2. Hence two distinct SPF calculations are run, one on Level 1 link state database and other on the Level 2 link state database. IS-IS Level 1-2 router behaves very close to OSPF Area Border Router (ABR). L1/L2 router sends both L1 and L2 hellos.

As default behavior L1/L2 router will only allow one way passage of prefixes from L1 Area to L2 Area, but not in reverse.
However if it is required to move prefixes from L2 Area to L1 Area then redistribute command under IS-IS configuration is required.

IS-IS Level 2 (L2) Router

An IS-IS Level 2 router has the link state information for the intra-area as well as inter-area routing. L2 router sends only L2 hellos. IS-IS Level 2 area can be compared with OSPF backbone area 0.

IS-IS Adjacency Table

Router Type	**L1**	**L1/L2**	**L2**
L1	L1 Adjacency if Area Id Matches,	L1 Adjacency if Area Id Matches, else no Adjacency	No Adjacency
L1/L2	L1 Adjacency if Area Id Matches,	L1 and L2 Adjacency if Area id Matches , else only L2	L2 Adjacency , Area Id doesn't
L2	No Adjacency	L2 Adjacency , Area Id doesn't matter	L2 Adjacency , Area Id doesn't

IS-IS Adjacency States

There are only three adjacency states in IS-IS.

Down: This is the initial state. Its means that no hellos have been received from the neighbor.
Initializing: This state means that the local router has successfully received hellos from the neighboring router, however it's not sure that the neighboring router has also successfully received local router's hellos.
Up: Now it's confirmed that neighboring router is receiving local router's hellos.

DNS

The Domain Name System (DNS) is a **hierarchical** and **decentralized** naming system for computers, services, or other resources connected to the Internet or a private network. The hierarchy goes like: Root > TLD/CCTLD > Authoritative.
In practice, the queries typically follow the pattern: The query from the requesting host to the local DNS server is recursive, and the remaining queries are iterative.

DNS RECORDS

- If **Type=A,** then Name is a hostname and Value is the IP address for the hostname. Thus, a Type A record provides the standard hostname-to-IP address map- ping. As an example, (relay1.bar.foo.com, 145.37.93.126, A) is a Type A record.
- If **Type=NS,** then Name is a domain (such as foo.com) and Value is the hostname of an authoritative DNS server that knows how to obtain the IP addresses for hosts in the domain. This record is used to route DNS queries further along in the query chain. As an example, (foo.com, dns.foo.com, NS) is a Type NS record.
- If **Type=CNAME,** then Value is a canonical hostname for the alias hostname Name. This record can provide querying hosts the canonical name for a host- name. As an example, (foo.com, relay1.bar.foo.com, CNAME) is a CNAME record.
- If **Type=MX,** then Value is the canonical name of a mail server that has an alias hostname Name. As an example, (foo.com, mail.bar.foo.com, MX) is an MX record. MX records allow the hostnames of mail servers to have simple aliases. Note that by using the MX record, a company can have the same aliased name for its mail server and for one of its other servers (such as its Web server). To obtain the canonical name for the mail server, a DNS client would query for an MX record; to obtain the canonical name for the other server, the DNS client would query for the CNAME record.
-

RSVP-TE Local repair techniques

In the **one-to-one backup** method, a label-switched path is established that intersects the original LSP somewhere downstream of the point of link or node failure. A separate backup LSP is established for each LSP that is backed up.

```
[R1]----[R2]----[R3]------[R4]------[R5]
   \       \         \      /   \     /
    [R6]----[R7]-----[R8]------[R9]

Protected LSP:  [R1->R2->R3->R4->R5]

R1's Backup:    [R1->R6->R7->R8->R3]
R2's Backup:    [R2->R7->R8->R4]
R3's Backup:    [R3->R8->R9->R5]
R4's Backup:    [R4->R9->R5]
```

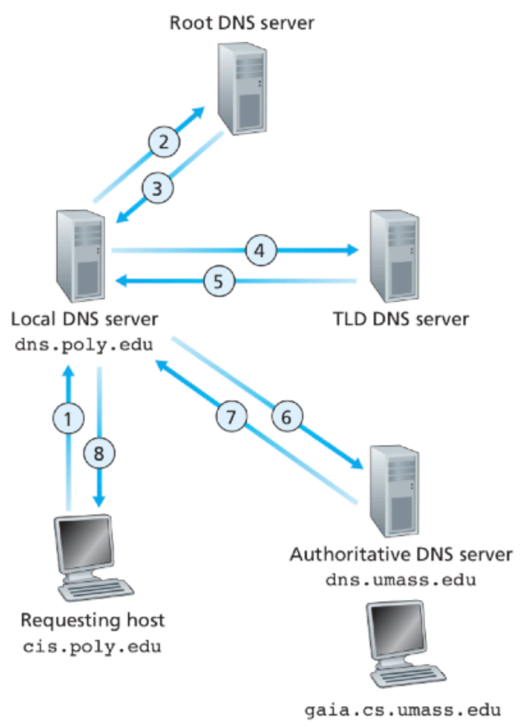
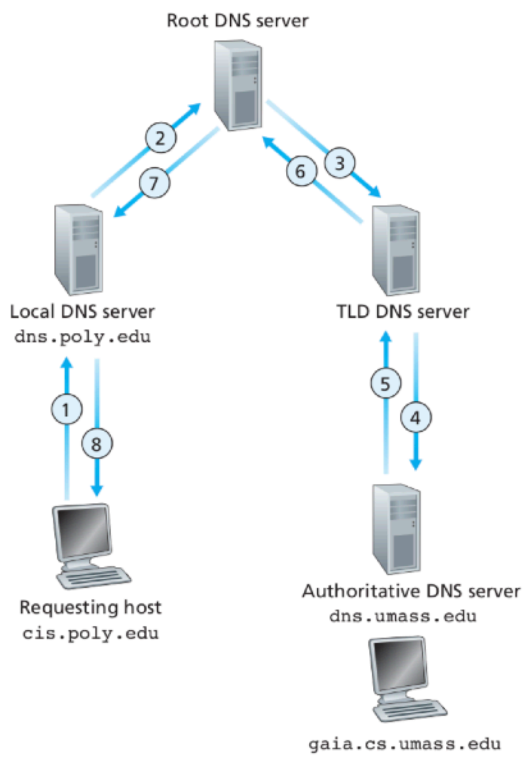


Figure 2.21 ♦ Interaction of the various DNS servers



_ 're **2.22** ♦ Recursive queries in DNS

Example 1. One-to-One Backup
Technique

In the simple topology shown in Example 1, the protected LSP runs from R1 to R5. R2 can provide user traffic protection by creating a partial backup LSP that merges with the protected LSP at R4. We refer to a partial one-to-one backup LSP [R2->R7->R8 >R4] as a detour.

To protect an LSP that traverses N nodes fully, there could be as many as (N - 1) detours. Example 1 shows the paths for the detours necessary to protect fully the LSP in the example. To minimize the number of LSPs in the network, it is desirable to merge a detour back to its protected LSP, when feasible. When a detour LSP intersects its protected LSP at an LSR with the same outgoing interface, it will be merged.

When a failure occurs along the protected LSP, the PLR redirects traffic onto the local detour. For instance, if the link [R2->R3] fails in Example 1, R2 will switch traffic received from R1 onto the protected LSP along link [R2->R7], using the label received when R2 created the detour. When R4 receives traffic with the label provided for R2's detour, R4 will switch that traffic onto link [R4-R5], using the label received from R5 for the protected LSP. At no point does the depth of the label stack increase as a result of the detour.

While R2 is using its detour, traffic will take the path
[R1->R2->R7->R8->R4->R5].

The facility backup method takes advantage of the MPLS label stack. Instead of creating a separate LSP for every backed-up LSP, a single LSP is created that serves to back up a set of LSPs. We call such an LSP tunnel a bypass tunnel.

The bypass tunnel must intersect the path of the original LSP(s) somewhere downstream of the PLR. Naturally, this constrains the set of LSPs being backed up via that bypass tunnel to those that pass through some common downstream node. All LSPs that pass through the point of local repair and through this common node that do not also use the facilities involved in the bypass tunnel are candidates for
this set of LSPs.

```
 [R8]
    \
[R1]---[R2]----[R3]-----[R4]---[R5]
          \            /        \
           [R6]===[R7]          [R9]

Protected LSP 1:   [R1->R2->R3->R4->R5]
Protected LSP 2:   [R8->R2->R3->R4]
Protected LSP 3:   [R2->R3->R4->R9]
Bypass LSP Tunnel: [R2->R6->R7->R4]
```

Example 2. Facility Backup Technique

In Example 2, R2 has built a bypass tunnel that protects against the failure of link [R2->R3] and node [R3]. The doubled lines represent this tunnel. This technique provides a scalability improvement, in that the same bypass tunnel can also be used to protect LSPs from any of R1, R2, or R8 to any of R4, R5, or R9. Example 2 describes three different protected LSPs that are using the same bypass tunnel for protection.

As with the one-to-one method, there could be as many as (N-1) bypass tunnels to fully protect an LSP that traverses N nodes. However, each of those bypass tunnels could protect a set of LSPs.

When a failure occurs along a protected LSP, the PLR redirects traffic into the appropriate bypass tunnel. For instance, if link [R2->R3] fails in Example 2, R2 will switch traffic received from R1 on the protected LSP onto link [R2->R6]. The label will be switched for one which will be understood by R4 to indicate the protected LSP, and the bypass tunnel's label will then be pushed onto the label-stack of the redirected packets. If penultimate-hop-popping is used, the merge point in Example 2, R4, will receive the redirected packet with a label indicating the protected LSP that the packet is to follow. If penultimate-hop-popping is not used, R4 will pop the bypass tunnel's label and examine the label underneath to determine the protected LSP that the packet is to follow. When R2 is using the bypass tunnel for protected LSP 1, the traffic takes the path [R1->R2->R6->R7->R4->R5]; the bypass tunnel is the connection between R2 and R4.

In one-to-one backup approach the PLR needs to maintain the state of separate backup paths for each LSP. This along with the periodic refresh messages is a burden for the PLR. This is where facility backup comes handy. Read more here

Read more here

NOTES ON RSVP AUTO BW

There are two ways to accomplish bandwidth measurement on a particular LSP:

1. Offline Calculation: Done outside of router, based on some type of bandwidth modeling often using a third party tool
2. Auto-Bandwidth: The BW value is calculated on the routers by periodically measuring how much traffic is actually forwarding over the LSPs. Because it runs directly on the routers, it can respond to changing traffic conditions much more rapidly, and with less overhead. Done by the head-end.

How does it work?
1. For every 'STATISTICS INTERVAL' BW over the LSP is measured (usually 60 secs)
2. For every 'ADJUST INTERVAL' the largest sample from above is used to calculate the new LSP BW. (For eg use 5 samples and adjust every 300 secs). Tricky number because we don't want to re-signal to often, also we want to react to changes rapidly.
3. If the change is larger than a user configured minimum, Adjust the the threshold and re-signal the new BW across RSVP (this operation is MBB)

Underflow condition - BW utilization has gone down significantly. But ADJUST INTERVAL is set to some high value. You are like, oh no I should have reacted and signaled the new BW more often.

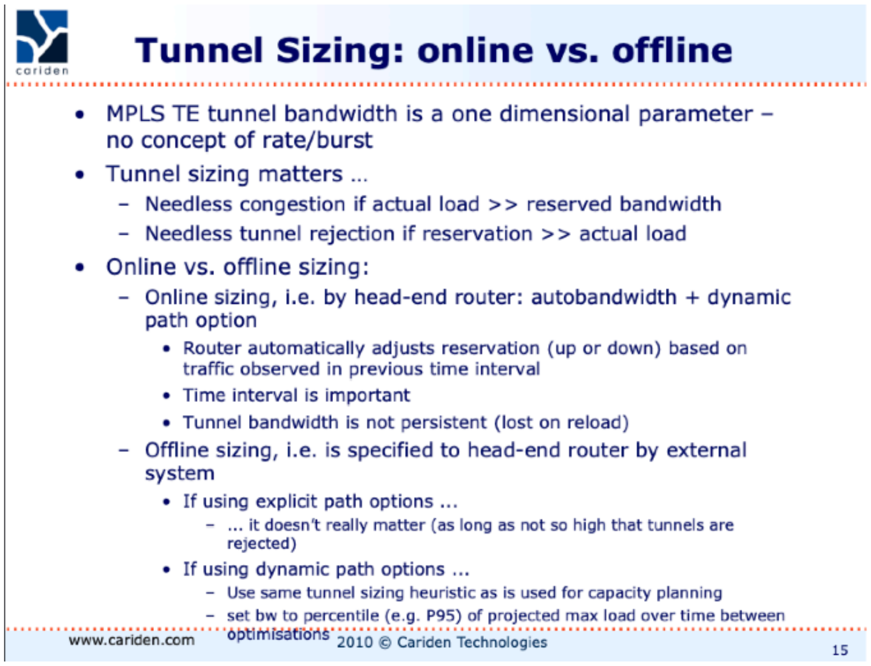
Overflow condition - BW utilization has increased significantly. But ADJUST INTERVAL is set to some high value. You are like, oh no I should have reacted and signaled the new BW more quickly/often.

RSVP is GOOD at adapting to changing network conditions such as failures of capacity changes. It's BAD in adapting to changing destinations.

- in IP changing destinations would mean change in the IGP cost to reach your BGP neighbor.
- In RSVP, it means tearing down the LSP (resizing down) and resizing up the new LSP. This takes ADJUST INTERVAL time to happen with potential congestion in the mean time.

WHY AUTO-BW is suboptimal? And sometimes needs manual intervention.

1. Consider a traffic destination shift, if your box does not support underflow, you can react to increasing LSP BW but cannot reclaim BW from the newly decreased LSP which means inefficient routing for the whole ADJUST INTERVAL period.
2. Auto-BW doesn't know about congestion. If a link becomes congested, packet loss causes TCP to throttle back and IP traffic goes down, auto-BW adapts to this new rate by the next adjust interval thinking everything is fine. This leads to sustained congestion. **[MOST IMPORTANT]**
3. Sometimes when a new BW reservation cannot be met, the LSP value is not updated yet (adjust interval not hit) even though traffic has increased, this leads to non-RSVP accounted traffic on the network which will cause silent congestion.
4. In parallel LSP scenario, if bW reservation cannot be met causes the existing LSP to be torn down completely. What happens? Traffic immediately shifts to remaining LSPs (Non accounted traffic is now going over these links). But worse is that these remaining LSPs have now become larger and are likely to fail to reserve the required bW. This results in failure of all parallel LSPs.
5. Tunnel Packing is common problem when it comes to online calculation or auto-bw. What this means is - Large LSPs can't fit small pipes. For instance 3x6 Gbps LSPs won't fit 2 x 10G circuits (even though 18 < 20). The third LSP will have to find another longer path if it exists at all. The two circuits will be left with 4 Gbps of unused capacity. A solution is to use multiple parallel LSPs where we land into the problem of max ECMP paths supported by the vendor. Explicit tunnels can address this problem.

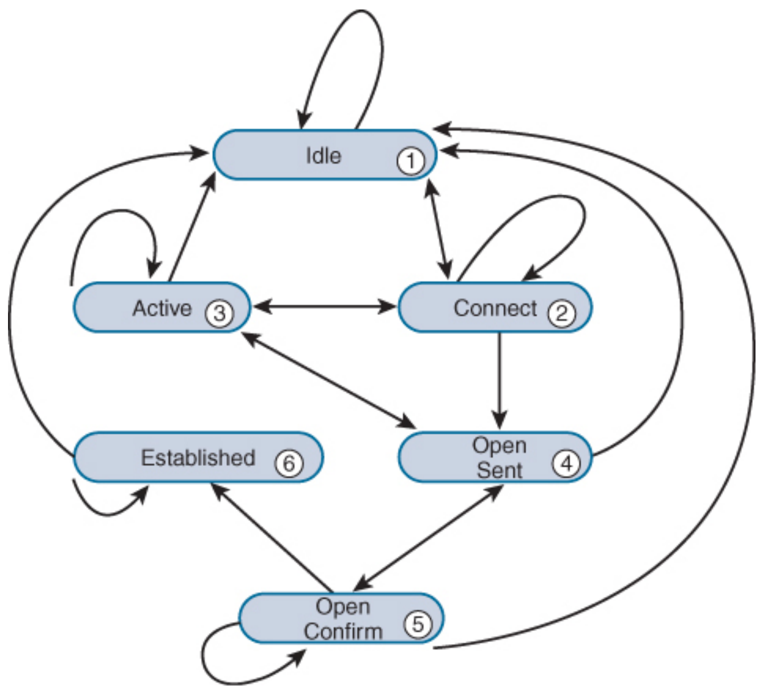


Tunnel Sizing: online vs. offline

- MPLS TE tunnel bandwidth is a one dimensional parameter – no concept of rate/burst
- Tunnel sizing matters ...
  - Needless congestion if actual load >> reserved bandwidth
  - Needless tunnel rejection if reservation >> actual load
- Online vs. offline sizing:
  - Online sizing, i.e. by head-end router: autobandwidth + dynamic path option
    - Router automatically adjusts reservation (up or down) based on traffic observed in previous time interval
    - Time interval is important
    - Tunnel bandwidth is not persistent (lost on reload)
  - Offline sizing, i.e. is specified to head-end router by external system
    - If using explicit path options ...
      - ... it doesn't really matter (as long as not so high that tunnels are rejected)
    - If using dynamic path options ...
      - Use same tunnel sizing heuristic as is used for capacity planning
      - set bw to percentile (e.g. P95) of projected max load over time between optimisations

www.cariden.com 2010 © Cariden Technologies 15

BGP FSM



Idle

This is the first stage of the BGP FSM. BGP detects a start event, tries to initiate a TCP connection to the BGP peer, and also listens for a new connect from a peer router.

If an error causes BGP to go back to the Idle state for a second time, the ConnectRetryTimer is set to 60 seconds and must decrement to zero before the connection is initiated again. Further failures to leave the Idle state result in the ConnectRetryTimer doubling in length from the previous time.

Connect

In this state, BGP initiates the TCP connection. If the 3-way TCP handshake completes, the established BGP Session, BGP process resets the ConnectRetryTimer and sends the Open message to the neighbor, and then changes to the **OpenSent** State.

If the ConnectRetry timer depletes before this stage is complete, a new TCP connection is attempted, the ConnectRetry timer is reset, and the state is moved to **Active**. If any other input is received, the state is changed to Idle.

During this stage, the neighbor with the higher IP address manages the connection. The router initiating the request uses a dynamic source port, but the destination port is always 179.

Active

In this state, BGP starts a new 3-way TCP handshake. If a connection is established, an Open message is sent, the Hold Timer is set to 4 minutes, and the state moves to OpenSent. If this attempt for TCP connection fails, the state moves back to the Connect state and resets the ConnectRetryTimer.

OpenSent

In this state, an Open message has been sent from the originating router and is awaiting an Open message from the other router. After the originating router receives the OPEN message from the other router, both OPEN messages are checked for errors. The following items are being compared:

BGP Versions must match, The source IP address of the OPEN message must match the IP address that is configured for the neighbor, The AS number in the OPEN message must match what is configured for the neighbor, BGP Identifiers (RID) must be unique. If a RID does not exist, this condition is not met, Security Parameters (Password, TTL, and the like).

If the Open messages do not have any errors, the Hold Time is negotiated (using the lower value), and a KEEPALIVE message is sent (assuming the value is not set to zero). The connection state is then moved to **OpenConfirm**. If an error is found in the OPEN message, a Notification message is sent, and the state is moved back to **Idle**.

If TCP receives a disconnect message, BGP closes the connection, resets the ConnectRetryTimer, and sets the state to **Active**. Any other input in this process results in the state moving to Idle.

OpenConfirm

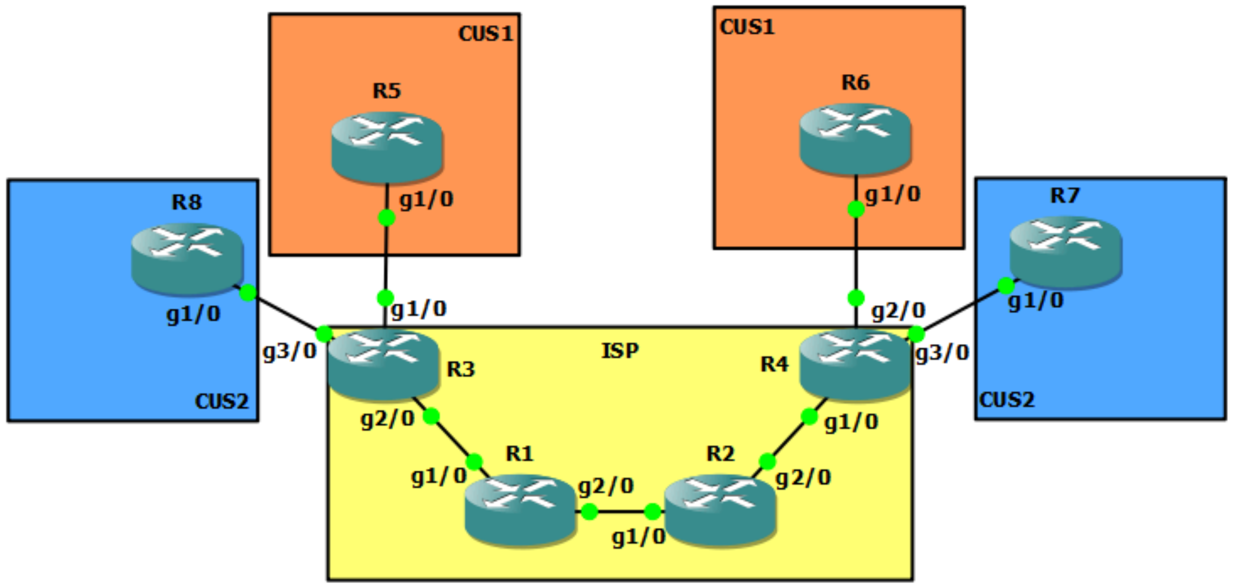
In this state, BGP waits for a Keepalive or Notification message. Upon receipt of a neighbor's Keepalive, the state is moved to Established. If the hold timer expires, a stop event occurs, or a Notification message is received, and the state is moved to Idle.

Established

In this state, the BGP session is **established**. BGP neighbors exchange routes via Update messages. As Update and Keepalive messages are received, the Hold Timer is reset. If the Hold Timer expires, an error is detected and BGP moves the neighbor back to the Idle state.

BGP path selection: valid NH > Higher Weight > Higher LP > Locally injected via nw or redistribute command > shortest AS_PATH > Origin (IGP > EGP > INCOMP ?) > lower MED > Neighbor type (eBGP > iBGP) > lower IGP metric > check if multi-path installation is required? > oldest known > lowest router ID > minimum cluster length > lowest neighbor address

L3VPN and flavors



Route Distinguisher:

The route distinguisher's sole job is to keep a route unique while the PE routers advertise NLRI (Network Layer Reachability Information) to each other. If R5 and R8 both advertise 5.5.5.5/32 to R3, how will R3 advertise both of those routes to R4 while keeping them unique. The VPNV4 family itself doesn't run in a VRF. It runs in the global routing instance and hence it needs something to distinguish a route.

Route Target:

The route target's job is to tell the PE routers what VPN a route actually belongs to. When R3 receives an advertisement from R5, not only does it change the route into a VPNv4 route with the RD to make it unique, it also adds a community value to that advertisement. This is an RT value. Once this NLRI gets to R4, R4 will ensure that only routes that have a certain RT, will be placed in their respective VRF (using the RT import under the VRF config for this particular customer). Say an extended community of 100:100 is encoded into this NLRI on R3. Say R4 has an import 100:100 configuration under its VRF, and hence matching the community of 100:100 on the received NLRI, the R4-PE router knows that the advertisement is meant for VRF CUS1. Note that the RD has nothing to do with this.

VPN LABEL:

So why do we need a VPN label now when we already know what VRF the route belongs to (and hence what customer it belongs to)? The RT is for the control plane, while the VPN label is for the data plane to do the actual forwarding. Let's expand on that idea a bit. When R3 advertises NLRI to R4, the RT is used to determine where a route actually belongs. When it comes to R5 actually sending a packet to R6, the VPN label is used. When a packet is sent, there is no field in the packet that the route-target is stored. Only the route advertisement contains the route-target as a community value.

The VPN label (also called the service label) needs to be carried in the core using an IGP label (also called the transport label). The transport label is popped at the PHP node (R2) and the packet is sent to the PE node (R4) with just the VPN label. This saves R4 from doing extra job of popping the IGP label. The PE can do a label lookup and find the prefix in the VRF table. More here

IMPLICIT NULL and EXPLICIT NULL in terms of MPLS

Implicit null is by **default** which means penultimate router should only send IP packet thus it pops the label to reduce the load on last hop router. The one disadvantage in implicit null approach is if the network is configured for QoS based on MPLS EXP bits, then QoS info is lost between penultimate router and last hop router.

In this case, we can make use of Explicit null which means penultimate hop router does not pop the label. It sends with label value of 0 but with other fields including EXP bits intact. This way QoS treatment is preserved between penultimate router and last hop router. Explicit null should be configured manually in last hop router.

Some other tricky stuff in MPLS L3 VPNs:

MPLS L3 VPN BGP Allow AS in: configured on CE routers to accept updates even if they see their own AS listed. Useful for scenarios where two customers needing L3VPN connectivity are in the same AS. Updates travel from CE1 to PE1 to P to PE2 to CE2. CE2 will dishonor this update as it will see it's own as (same originating AS)

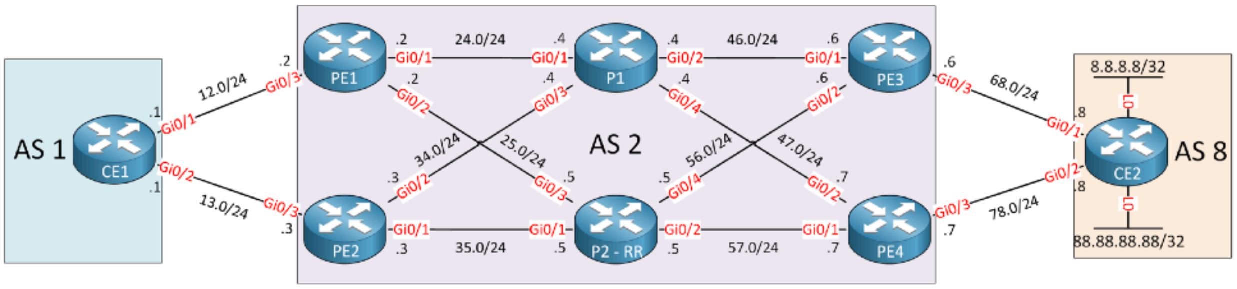
MLPS L3 VPN BGP AS Override: configured on PE routers to address the same problem as discussed above. Just that, in this, the PE removes the AS of the CE and puts its own AS instead, hence overrides it, preserving the AS_PATH length.

Connection designs: https://networklessons.com/cisco/ccna-routing-switching-icnd2-200-105/singledual-homed-and-multi-homed-designs

BGP PIC Edge and Core

Core: decreases convergence time when a core router fails and your IGP has to find a new best path to your PE router.

Edge: Some PE fails, BGP needs to switch to an alternative PE in such an event



PIC Core SCENARIO:

BGP: to get to 8.8.8.8/32 and 88.88.88.88/32 we use next hop 6.6.6.6 (PE3)
IGP: to get to 6.6.6.6/32 we use next hop 192.168.24.4 with interface GigabitEthernet0/1 (via P1)

FIB:
8.8.8.8/32 via 192.168.24.4 interface GigabitEthernet0/1
88.88.88.88/32 via 192.168.24.4 interface GigabitEthernet0/1

P1 fails, OSPF takes its sweet time to converge and choose P2 to reach 6.6.6.6 for which, PE1 now takes Gi0/2 as the outgoing interface. Convergence for 2 prefixes is easy for OSPF, but for a large number of routes, this is gonna take some time and effort. Solution? BGP PIC Core.

PIC Core involves a concept of indirection pointer (NH + outgoing interface). In case of a failure, we just update the pointer PE1 points to

```
8.8.8.8 -> 6.6.6.6 -> P1 -> Gi 0/1
         |
          -> 6.6.6.6 -> P2 -> Gi 0/2
```

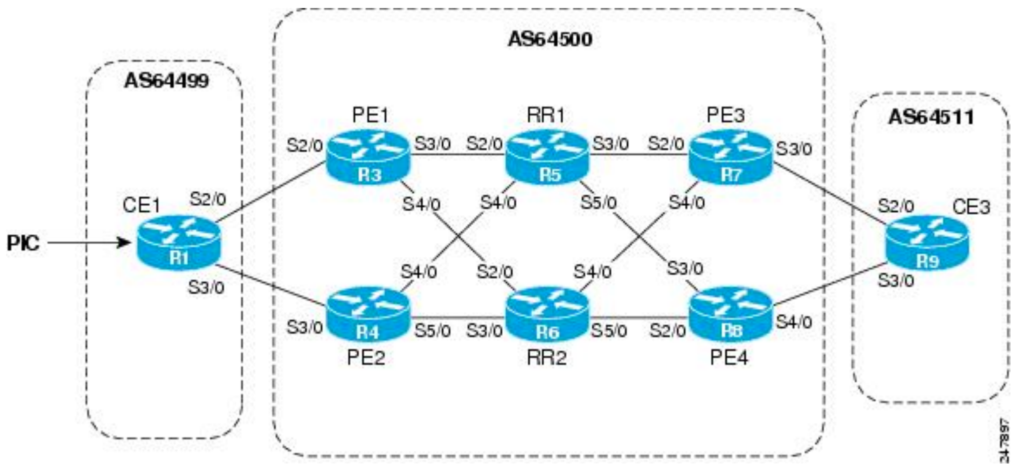
PIC Edge SCENARIO:

What if PE3 fails? This would mean iBGP neighbor-ship going down. And we all know how long does it take for BGP to be up. In this case we would want:

- to use PE4 as the alternative PE
- To pick the path to PE4 quickly without affecting convergence

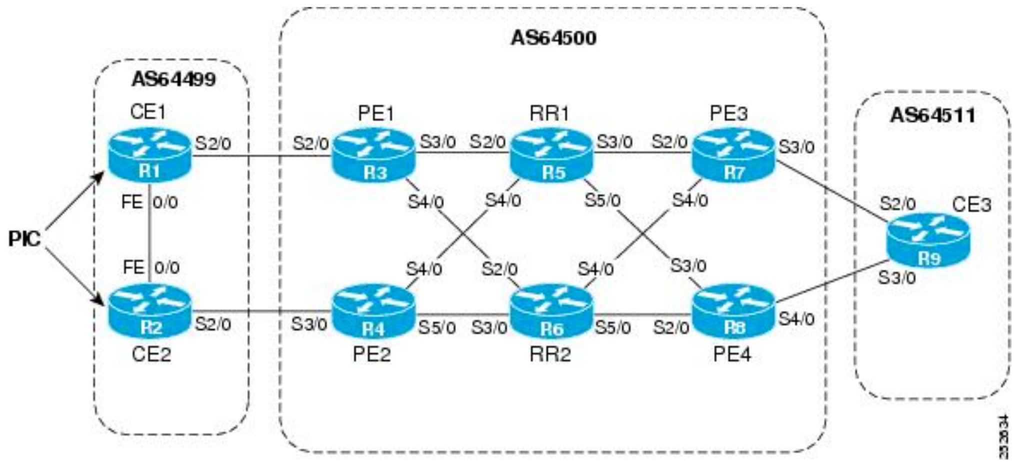
This is achieved using PIC Edge by telling the P2-RR to send and advertise all additional paths towards neighbor PE1. Also PE1 needs to be told to receive all additional paths and install them as backup. After this is done, PE1 starts seeing PE4 as a backup NH and uses it in case PE3 fails.

Other PIC Edge scenarios:



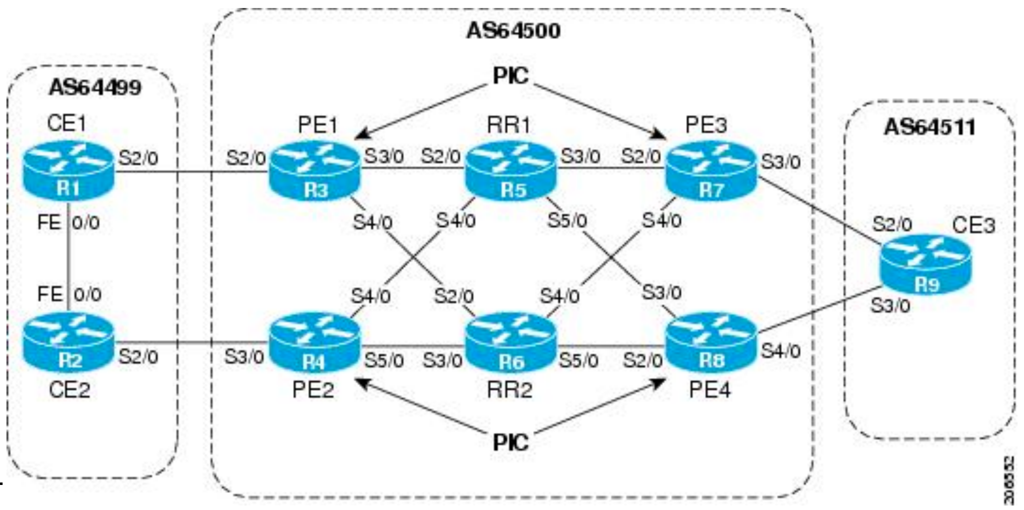
SCENARIO 1: CE1 is configured with the BGP PIC feature. BGP computes PE1 as the best path and PE2 as the backup/alternate path and installs both routes into the RIB and Cisco Express Forwarding plane. When the CE1-PE1 link goes down, Cisco Express Forwarding detects the link failure and points the forwarding object to the backup/alternate path. Traffic is quickly rerouted due to local fast convergence in Cisco Express Forwarding.

SCENARIO 2: In figure below, If the CE1-PE1 link or PE1 goes down and BGP PIC is enabled on CE1, BGP recomputes the best path, removing the next hop PE1 from RIB and reinstalling CE2 as the next hop into the RIB and Cisco Express Forwarding. CE1 automatically gets a backup/alternate repair path into Cisco Express Forwarding and the traffic loss during forwarding is now in sub-seconds, thereby achieving fast convergence.



SCENARIO 3: PE1 has two paths to reach CE3 from the reflect routers in the figure above:
PE3 is the primary path with the next hop as a PE3 address.
PE4 is the backup/alternate path with the next hop as a PE4 address.

In this example, all the PE devices can be configured with the BGP PIC feature under IPv4 or VPNv4 address families. For BGP PIC to work in BGP for PE-CE link protection, set the policies on PE3 and PE4 for prefixes received from CE3 so that one of the PE devices acts as the primary and the other as the backup/alternate. Usually, this is done using local preference and giving better local preference to PE3. When the PE3-CE3 link goes down, CEF detects the link failure, and PE3 recomputes the best path, selects PE4 as the best path, and sends a withdraw message for the PE3 prefix to the reflect routers. Some of the traffic goes through PE3-PE4 until BGP installs PE4 as the best path route into the RIB and CEF. PE1 receives the withdraw, recomputes the best path, selects PE4 as the best path, and installs the routes into the RIB and Cisco Express Forwarding plane.



SCENARIO 4: CE3 is dual-homed with PE3 and PE4.
PE1 has two paths to reach CE3 from the reflect routers:
PE3 is the primary path with the next hop as a PE3 address.

PE4 is the backup/alternate path with the next hop as a PE4 address. Same policies and local preference rules apply here as discussed in previous scenario. When PE3 goes down, PE1 knows about the removal of the host prefix by IGPs in sub-seconds, recomputes the best path, selects PE4 as the best path, and installs the routes into the RIB and CEF plane. Normal BGP convergence will happen while BGP PIC is redirecting the traffic through PE4, and packets are not lost.

LDP

The Label Distribution Protocol (LDP) is a protocol for distributing labels in non-traffic-engineered applications. LDP allows routers to establish label-switched paths (LSPs) through a network by mapping network-layer routing information directly to data link layer-switched paths. LDP populates LIB (parallel analogy to RIB) and hence the LFIB (parallel analogy to FIB). **Label Information Base** (**LIB**) is a table where prefix to label bindings are built and stored in the control plane. **Label Forwarding Information Base** (**LFIB**) is another MPLS table used to forward label packets throughout the MPLS network and build in the data plane.

Some other terms LDP uses are LSR, LER, LSP.

LDP Message types:

Discovery: Basic and Extended. Hello messages sent as UDP packets on Discovery listening ports. Help in forming adjacencies. Extended are used for T-LDP for establishing targeted LDP sessions (useful in rLFA). In this case, the dest address if of the T-LDP neighbor. Targeted router chooses to respond or ignore, if decides to respond, sends hellos periodically. The hello message contains the LDP transport address which is usually the LSR ID. Transport address is used to establish the TCP session between the two LDP speaking routers. Labels are exchanged over this reliable connection.

Session: Session messages establish, maintain, and terminate sessions between LDP peers. Session is established over the TCP connection that was established earlier post hello exchange.

Advertisements: Advertisement messages create, change, and delete label mappings for forwarding equivalence classes (FECs). Requesting a label or advertising a label mapping to a peer is a decision made by the local router. In general, the router requests a label mapping from a neighboring router when it needs one and advertises a label mapping to a neighboring router when it wants the neighbor to use a label.

Notifications: Just like BGP, they are bad and involve notifying errors to LDP peers.

SR-LDP inter-op: SRMS

WHY prefer SR-core over LDP-core:

Problems with LDP:

- It uses an additional protocol running on all devices just to generate labels. In order to provide stability and scalability, large networks should have as few protocols as possible running in the core

- It introduces the potential for black-holing of traffic, because IGP and LDP are not synchronized

- It does not employ global label space, meaning that adjacent routers may use a different label to reach the same router

- It's difficult to visualize the Label Switched Path (LSP) because of the issue above

- It may take time to recover after link failure unless session protection is utilized

Why SR is good considering these points?:

- Labels are carried by IGP so no separate protocol for label distribution and hence LSD and IGP always in sync.

- SR offers Global label space. In SR, every node has a unique identifier called the node SID. This identifier is globally unique and would normally be based on a loopback on the device. Adjacency SIDs can also exist for locally significant labels for a segment between two devices.

WHY SR-TE better than RSVP-TE:

- SR-TE is stateless and everything is only maintained at the HE which is in turn controlled by the controller. No state is maintained at the mid-points and the tail.

- Ease of use easily deployable

- Does not have to deal with problem of auto-bw, has to deal with the problem of micro-loops though, OTOH, RSVP-TE is immune to loops as path is hard-coded

- Can leverage MP-BGP and uses BGP NLRIs to advertise policies.

Challenges in SR:

- Centralized controller, so need to put all the intelligence there. Requires Network awareness and active network monitoring to implement policies and flex algorithms.

- Controller, data plane inter-op scalability. For e.g. restart of controller

- One controller is a SPOF. To improve reliability we need East-west scalability i.e. multiple controllers and inter-op

- It's not just Network config but more of software tasks are involved as OpenFlow and Rest APIs are used to program different vendor devices in the forwarding plane

- No BW reservation techniques in SR. it can react to BW changes but reserving something upfront is not achievable

- Depth of MPLS label stack. Since HE is responsible for pushing the label stack as told to it by the controller, LSP can be shorter as all devices don’t support large label stack.

BGP RR and Confederations sub-optimality

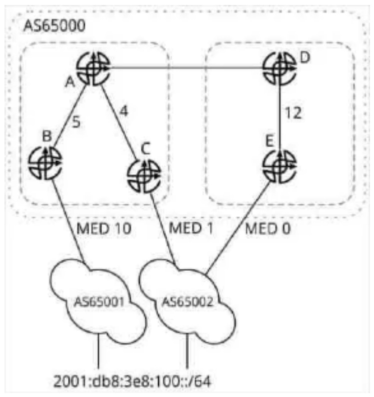
Challenges with RR:

- Increased route convergence time: Node delay + processing delay as RR runs SPF first before reflecting them to RR clients.

- Sub-optimal routing as RRs word taken for granted: if RR chooses one POP for a destination prefix based on low IGP cost and advertises that POP as the NH for the prefix to other clients sitting across the coast (even though they have their own POP which is near to them)

- Reduced path diversity when it comes to BGP FRR. Having iBGP would mean having greater visibility and hence more area for backup computation. Having RR in the picture, an iBGP neighbor loses the picture of other routes as only best route is advertised to them by the RR

- MED oscillations:



A and D are RRs

AT A: (MED not compared as different AS)

edge	MED	IGP Cost	
C	1	4	(BEST, lower IGP metric)
B	10	5	

AT D:(MED compared and E chosen)

edge	MED	IGP Cost	
E	0	12	(BEST, lower MED)
C	1	5	

E sends updates to peers

Now AT A: (Still path through B wins as MED not compared)

edge	MED	IGP Cost
E	0	13
C	1	5
B	10	5(BEST)

A sends updates to peers and D receives it

AT D: (disregards MED and choses lower IGP)

edge	MED	IGP Cost
E	0	12
B	10	6(BEST)

D sends this to A and A is back again where it started.

Solution to MED oscillations:

- The BGP Additional Paths feature is a BGP extension that allows the advertisement of multiple paths for the same prefix without the new paths implicitly replacing any previous paths (IMPLICIT WITHDRAW). This behavior promotes path diversity and reduces MED oscillations.

- Strip off MED: strip it off at EDGE, used in accordance with BGP Communities
- always-compare-MED: causes sub-optimal path selection though and is difficult to achieve as to have a reasonable comparison, AS-AS harmony needs to be there.
- Deterministic-MED: causes forced comparison of MED by reordering/regrouping the updates with same AS. Without this feature, the MED may not be compared because of the top-down processing the BGP algorithm uses to evaluate routes. Example

Loop prevention in RR deployment:

when RR receives update which CLUSTER_LIST contains router's own cluster ID, this update is discarded.

BGP ORIGINATOR ID: The origin of the route. Can be some iBGP router where the route originated (by network command) or can be the edge router in the same AS that received eBGP route update from another AS. This value is never modified. Used by iBGP speakers to detect loops (both client and RR)

BGP CLUSTER LIST: a list that shows what clusters the update has travelled. Used by RRs to prevent loop (clients or a non-client has no idea to what cluster they belong to). The list is modified along the path as the update travels. Some notes:

- It’s only created when ‘reflection’ takes place by the RR.

- If RR is the originator, it won't create a cluster list.
- If RR receives an eBGP update, RR does not create a cluster list (as this is not technically a reflection)
- The cluster list is stripped off when the eBGP update is sent to another AS. They don't need that shit.
- More about clusters here

RR Design best practices:

- A Session Between an RR and Its Client Should Not Traverse a Non-client

CONFEDERATIONS rules:

- Intra-confederation eBGP session: The member AS number is prepended to the AS_CONFED_SEQUENCE of the AS_PATH.
- Internal BGP session: The AS_PATH is not modified
- External BGP session: The member AS numbers are removed from the AS_PATH, and the confederation number is prepended to the AS_PATH
- The intra-confederation eBGP session follows iBGP rules for prefix advertisement in some regards and eBGP rules in others. For example, NEXT_HOP, MED, and LOCAL_PREF are preserved and propagated, **yet** AS_PATH is modified when sending the updates.

RR or CONFED?

- Route reflection requires a single IGP inside an AS, whereas con- federation supports single or separate IGPs. This is probably the most distinctive advantage of confederation over route reflection. If your IGP is reaching its scalability limit or it is just too big to handle administration, confederation can be used to reduce the size of the IGP tables.
- Migration complexity is very low for route reflection
- RR has more deployment history and is more explored. CONFED needs capability support on routers.

COMMUNITY DESIGN

Communities is a powerful tool that provides PBR in BGP. Some examples:

- **Prefix Origin Tracking:** An ISP network typically has three types of routes: transit, peering, and customer. This prevents the ISP from advertising peering routes over the transit connection and transit routes to peers. If the ISP does not filter prefix advertisements between the transit and peering connections, the ISP ends up providing transit service for its peers.

- **Dynamic Customer Policy:** A customer that is multi-homed to the same provider might want traffic to load-balance between the two connections or might want to use the connections in a primary and backup scenario. ISP can define a policy to let customers dynamically influence traffic patterns upstream.

- **Local Preference Manipulation:** to influence inbound traffic. ISP will define some communities and ask customer to tag their routes with them. ISP will take the appropriate action then.

- **QoS propagation:** The QoS Policy Propagation via BGP (QPPB) feature ensures that traffic in both directions is provided with the QoS level the customer has purchased. The ISP network in this example offers three levels of service: gold, silver, and bronze. These service levels have their associated communities.

BGP TE and Load Balancing

Inbound TE:

- MED (be careful, they are often filtered)

- AS-PATH prepend - Signal preferred path by growing AS_PATH on less preferred paths

- Communities and LP - ISPs can ask customer to tag their routes with certain communities for which they can set desired LP

- Selective advertisements (aka conditional route injection)

- Using NetFlow Analysis: NetFlow with BGP provides valuable information like:

    source AS
    destination AS
    peer ASs
    IP NextHop and BGP NextHop

This enables various use cases for network monitoring and peering analysis:

- Quickly notice AS path, peering, or traffic engineering anomalies

- Pick a specific peer, customer, or site and see a complete view of where the traffic is coming from, passing through, and exiting. See in a snapshot which countries/regions/cities traffic is going to or coming from

- View traffic on a single BGP path and see how it changed over time

- Determine least cost path routing depending on traffic volumes and paths

- This analysis can be used by network operators to answer fundamental questions about their network including: Who is my traffic going to? Which AS paths is it taking? Which country or region does it terminate in? Whom should I connect (peer) to? Which transit provider is the most cost effective? How much is traffic costing me for a particular server, customer, or peer? Should I add more circuit capacity to my network? What paths? Do I need new peering agreements to reduce traffic costs?

Outbound TE:

Some common known problems with Internet

- **LFN aka Bandwidth Delay Product:** In data communications, the bandwidth-delay product is the product of a data link's capacity (in bits per second) and its round-trip delay time (in seconds).[1] The result, an amount of data measured in bits (or bytes), is equivalent to the maximum amount of data on the network circuit at any given time, i.e., data that has been transmitted but not yet acknowledged. The bandwidth-delay product was originally proposed[2] as a rule of thumb for sizing router buffers in conjunction with congestion avoidance algorithm Random Early Detection (RED).

- **Buffer Bloat:** Bufferbloat is a cause of high latency in packet-switched networks caused by excess buffering of packets. Bufferbloat can also cause packet delay variation (also known as jitter), as well as reduce the overall network throughput. When a router or switch is configured to use excessively large buffers, even very high-speed networks can become practically unusable for many interactive applications like voice over IP (VoIP), online gaming, and even ordinary web surfing.

- **TCP Global Synchronization:** TCP global synchronization occurs when a sudden burst of traffic causes simultaneous packet loss across many TCP sessions using a single (congested) link. Each affected TCP session backs off its send rate at the same time, causing link utilization to go way down. When link utilization is down packet loss is low or non-existent and therefore each affected TCP session increases its send rate at the same time, causing link utilization to go way up and causing severe congestion again, and the cycle repeats. The result is a saw tooth shaped bandwidth utilization on the single link causing inefficient use of available bandwidth and inconsistent application throughput. The leading cause of TCP global synchronization is sudden congestion causing numerous packet drops and simultaneously affecting many TCP flows all at once, as would typically happen in classic tail-drop queue management. More intelligent queue management such as WRED (weighted random early discard) is designed to improve overall network performance by avoiding numerous and sudden packet drops that can cause TCP global synchronization.

- **Thundering Herd Problem**

- **Head of line Blocking:** Can be at L7(HTTP) or L4(TCP). Head-of-line blocking (HOL blocking) in computer networking is a performance-limiting phenomenon that occurs when a line of packets is held up by the first packet. Examples include input buffered network switches, out-of-order delivery and multiple requests in HTTP pipelining.

Some common WHYs related to protocols

- **Why a triple duplicate ack?**: Since TCP does not use negative acknowledgments, the receiver cannot send an explicit negative acknowledgment back to the sender. Instead, it simply re-acknowledges (that is, generates a duplicate ACK for) the last in-order byte of data it has received.

- **Why sequence numbers are needed?**:
  - they were introduced with an idea that even ACKs and NAKs can get corrupted. In that case sender has no way of knowing that Receiver has received the last piece of data correctly
  - for sequential numbering of packets of data flowing from sender to receiver (ordered delivery)
  - Gaps in the sequence numbers of received packets allow the receiver to detect a lost packet
  - Packets with duplicate sequence numbers allow the receiver to detect duplicate copies of a packet, basically differentiate between a retransmission and new segment

- **Why do we need areas in OSPF?:**
  When an OSPF domain grows large, the flooding and the resulting size of the link state database becomes a scaling problem. The problem is remedied by breaking the routing domain into areas

- **Why area 0?:**
  For loop prevention. Requiring all non-backbone routes to go through the backbone is a loop-prevention mechanism. OSPF is a LSRP inside an area but is DVRP when it comes to inter-area communication. The exchange of routing information between areas is essentially Distance Vector. With Distance Vector, the larger number of redundant paths you have the worse your convergence properties get. OSPF requires all areas to attach directly to the backbone so it limits the topology to a simple hub and spoke topology. This eliminates redundant paths and prevents it from being subjected to 'count to infinity' problems.

- **What other ways loops are prevented in OSPF?:**
  - Type 1 and Type 2 LSAs will be preferred over Type 3
  - Type 3 from backbone area would be preferred over Type 3 from Non-backbone

  The above can be put to use in case of two ABRs between 2 areas(one backbone and other non-backbone)

VxLAN

Bringing different virtual machines together into the same LAN segment when there's a whole network sitting between them. We wanna run a L2 LAN segment over the existing L3 underlay.

What problems does VXLAN solve?

- Scaled segmentation of the network: you can theoretically create as many as 16 million VXLANs in an administrative domain (as opposed to 4094 VLANs)
- VXLANs provide network segmentation at the scale required by cloud builders to support very large numbers of tenants.
- With traditional Layer 2 networks you are constrained by Layer 2 boundaries and forced to create large or geographically stretched Layer 2 domains. VXLAN's functionality allows you to dynamically allocate resources within or between data centers and enables migration of virtual machines between servers that exist in separate Layer 2 domains by tunneling the traffic over Layer 3 networks.

Design:

- No need for VLAN trunking or spanning tree as underlay is already routed. OSPF, EIGRP, IS-IS are good choices for underlay
- Flexibility: Underlay can be changed anytime without the need of redesigning the overlay
- VTEP provides the connection between the overlay and the underlay. VNI (nothing but 24 bit VXLAN ID) is the L2 segment. VTEP bridges VNIs to the L3 network.

Encapsulation:

Scope of Leaf and Spine architecture:

- Core, Distribution and Access layers is traditional design now. Follows North -> South traffic pattern. Problems:
  - restricts mobility. VM moving from one VXLAN to another would need to request new IP via DHCP. Another option is to expand VXLAN across the access layers.
  - But we cannot span VXLAN across the access layers as that increases the L2 failure domain and also increases the broadcast domain
  - In data center model we typically have hosts with static addresses. So this model doesn't work here.
- Spine - Leaf arch = CLOS
- Has only two layers. Spine and leaf. All spines connect to all leaves and everything is L3 routed. We have ECMP so a link failure is not a problem.
- Focuses on east-west traffic pattern which is what traffic in a DC like.
- Devices can move without changing the IP addresses once VXLAN is deployed over this L3 underlay.
- Scalable : need to add move hosts/VMs add leaf SW. need to add more BW? Add Spine SW. want to expand overall? Divide existing Leaf/Spine into clusters and another Spine layer.

Learning of host addresses:

- how does host A which is about to send traffic learn about the DMAC of host B? How does ingress VTEP learn about egress VTEP
- Here we use the control plane to learn MAC addresses instead of the traditional data plane learning using ARP/ND. In control plane learning, hosts learn MAC addresses way before then they actually plan to use it. This is shared along with BGP updates.

Routing and multi-tenancy:

-