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. 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:
Next hop self, BGP Synchronization

Q - How does a Juniper Router decides its path (BGP)?
A - Discuss BGP metric, ties and tie-breakers

Q - Types of Routing? Types of Dynamic Routing Protocols?
A - Static and Dynamic. LSRP and DVRP

Q - What's the difference between a router and switch?
A - **Router**: L3 device, High forwarding rates, Highly configurable, connects two LANs, WANs, PDU-Packet, each port has own broadcast domain, does forwarding on software logic (Routing Tables)
Switch: L2 device, less configurable, connects devices in a LAN, PDU-Frame, each port has own collision domain and whole switch is one broadcast domain, does forwarding on hardware logic (CAM and ASICs)

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

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) + 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 webserver 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 IP datagram size a host can receive. This does not contain the IP header and TCP header size. So, MSS = MTU - 40

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.

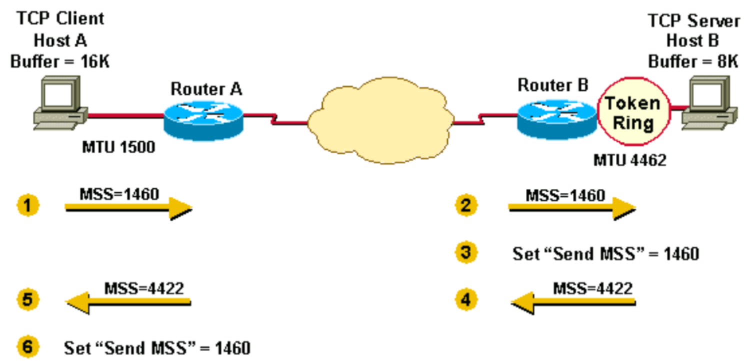
IPv6 routers do not support fragmentation or the Don't Fragment option. For IPv6, Path MTU Discovery works by initially assuming the path MTU is the same as the MTU on the link layer interface where the traffic originates. Then, similar to IPv4, any device along the path whose MTU is smaller than the 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

Q - 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

Q - 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.

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.