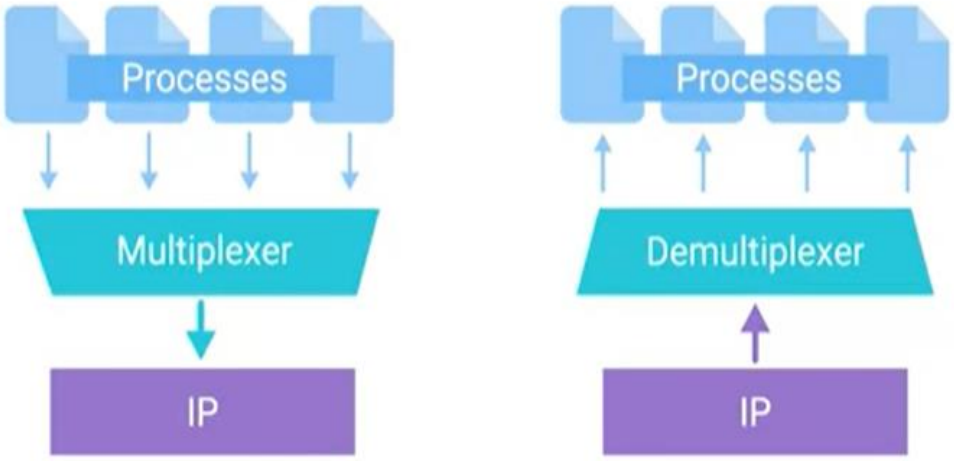
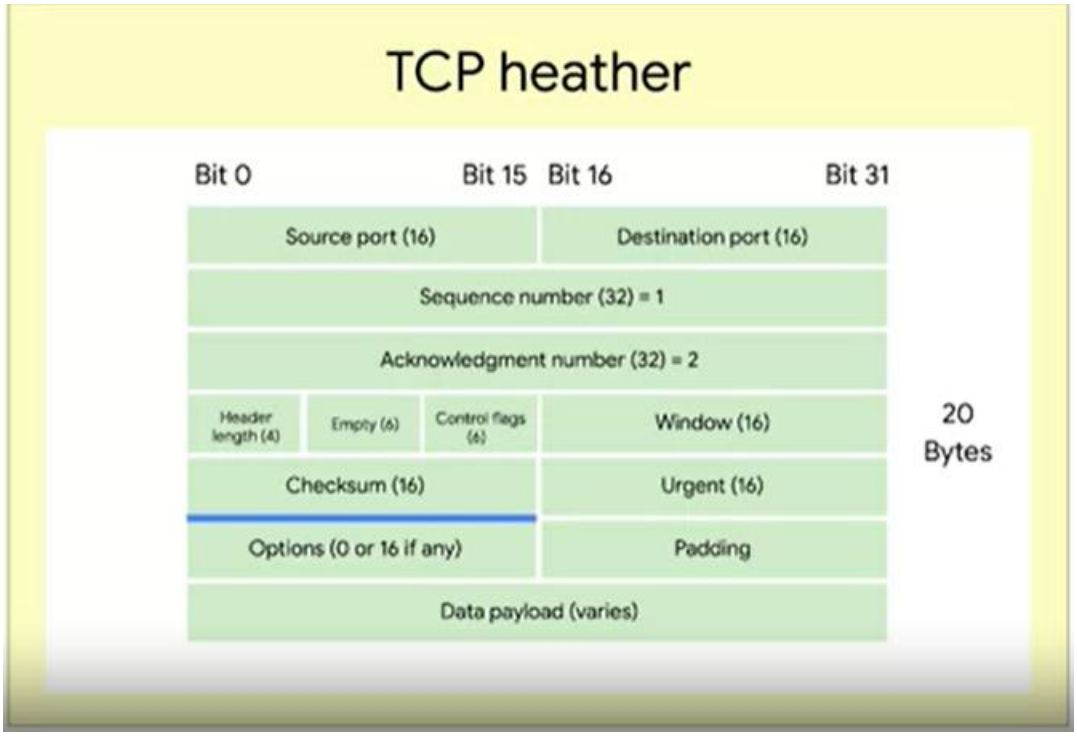
TCP Control Flags and the Three-way Handshake

-Aayush Tyagi

- The Transport layer allows traffic to be directed to specific network applications. And the Application layer allows these applications to communicate in a way they understand.
- The transport layer has the ability to multiplex and demultiplex, which sets this layer apart from all others.
- Multiplexing in the transport layer means that nodes on the network have the ability to direct traffic toward many different receiving services.
- Demultiplexing is the same concept, just at the receiving end, it's taking traffic that's all aimed at the same node and delivering it to the proper receiving service.



-
- A port is a 16-bit number that's used to direct traffic to specific services running on a networked computer.
- FTP is an older method used for transferring files from one computer to another, but you still see it in use today.
- A TCP segment is made up of a TCP header and a data section.

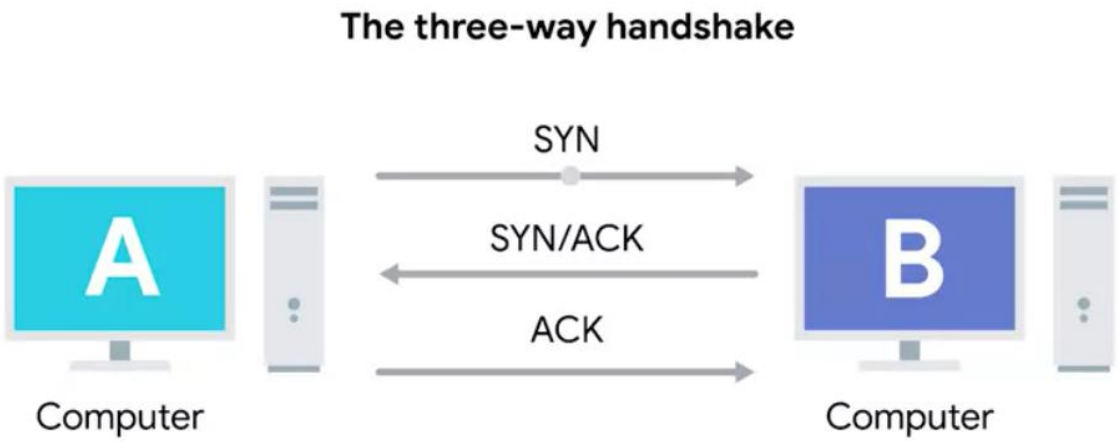


- A source port is a high numbered port chosen from a special section of ports known as ephemeral ports.
- Sequence Number: This is a 32-bit number that's used to keep track of where in a sequence of TCP segments this one is expected to be.
- The acknowledgment number is the number of the next expected segment.
- Data Offset Field: This field is a four-bit number that communicates how long the TCP header for this segment is.
- A TCP window specifies the range of sequence numbers that might be sent before an acknowledgement is required.
- TCP Checksum: It operates just like the checksum fields at the IP and Ethernet level.
- The Urgent pointer field is used in conjunction with one of the TCP control flags to point out particular segments that might be more important than others.
- Next up, we have the options field. Like the urgent pointer field, this is rarely used in the real world, but it's sometimes used for more complicated flow control protocols.
- Finally, we have some padding which is just a sequence of zeros to ensure that the data payload section begins at the expected location.

- TCP as a protocols establishes connections used to send long chains of segments of data.
- TCP establishes connections through the use of different TCP control flags used in a very specific order.

TCP control flags - order they appear in the TCP header

1. **URG** -> Urgent - A value of one here indicates that the segment is considered urgent and that the urgent pointer field has more data about this;
2. **ACK** -> Acknowledged - A value of one in this field means that the acknowledgment number field should be examined.
3. **PSH** -> push - The transmitting device wants the receiving device to push currently-buffered data to the application on the receiving end as soon as possible. (A buffer is a computing technique where a certain amount of data is held somewhere, before being sent elsewhere - used to send large chunks of data more efficiently);
4. **RST** - Reset - One of the sides in a TCP connection has not been able to properly recover from a series of missing or malformed segments;
5. **SYN** (synchronize) - It's used when first establishing a TCP connection and makes sure the receiving end knows to examine the sequence number field;
6. **FIN** (finish) - When this flag is set to one, it means the transmitting computer does not have any more data to send and the connection can be closed.



TCP Socket States

Socket - The intantiation of an end-point in a potential TCP connection;
Instantiation - The actual implementation of something defined elsewhere.

- The traffic can be sent to any port, but the response will be available only if the program has opened a socket on that port.
- TCP sockets can exist in a lot of states.
  1. LISTEN - A TCP socket is ready and listening for incoming connections - only on the server side;
  2. SYN_SENT - A synchronication request has been sent but the connection hasn't been established yet; - only on the client side;
  3. SYN-RECEIVED - A socket previously in a LISTEN state has received a synchronization request and sent a SYN/ACK back; - only on the server side;
  4. ESTABLISHED - The TCP connection is in working order and both sides are free to send to each other data; - state is available in both the client and server sides of the connection;
  5. FIN_WAIT - A FIN has been sent, but the corresponding ACK from the other end hasn't been received yet;
  6. CLOSE_WAIT - The connection has been closed at the TCP layer, but that the application that opened the socket hasn't released its hold on the socket yet;
  7. CLOSED - The connection has been fully terminated and that no further communication is possible;

- There are more socket states, TCP protocols is universal but the socket states names may vary from operating systems but their functionality is the same;

Connection-oriented and Connectionless Protocols

- Connection-oriented Protocols - Establishes a connection, and uses this to ensure that all data has been properly transmitted. (TCP);
- Every segment of data sent is acknowledged;
    - At the Ethernet and IP layer if the Checksum does not computer all the data is discarded, and it is up to TCP to determine when to resend the data since TCP expects an ACK for every bit of data it sends;
    - Sequence numbers enable data to be put back in the same order;

➔ There is a lot of overheade in regrad to the connection-oriented protocol

- Need to establish a connection;
- Need to send a stream of constant streams of acknowledgments;
- Tear the connection down at the end;

➔ Connectionless Protocol - User Datagram Protocol (UDP) - Here you set the destination port and send the packet - useful for messages that are not supper important - example - streaming video - bandwidth is used for actual data rather than establishing connection overhead;

System ports V/s Ephemeral ports

- Transportation layer protocols use the concept of ports and multiplexing and demultiplexing to deliever data to individual service listening on network nodes. These ports are represented by a single 16-bit number (0 - 65535);

- The IANA has split the range into independent sections :
    - Port 0 - It is not used for network traffic, but for communication between different programs on the same computer;
    - Ports 1 - 1023 - System ports / ports - used for network services;
    - HTTP - port 80;
    - FTP - port 21;
    - Most OS Administrative access is required to start a program that listens on a system port;
    - Ports 1024 - 49151 - registered ports - used for uncommon network services;
    - port 3306 - database listen on;
    - On most operating systems, any user of any access level can start a program listening on a registered port.
    - Ports 49152 - 65535 - private or ephemeral ports
- Ephemeral ports can't be registered with the IANA and are generally used for establishing outbound connections.
- All TCP traffic uses a destination port and a source port.
- When a client wants to communicate with a server, the client will be assigned an ephemeral port to be used for just that one connection, while the server listens on a static system or registered port.

Firewalls

- A device that blocks traffic that meets certain criteria;
- Provide secure network and can operate at lots of different layers of a network;
- Inspection of application layer traffic;
- Block ranges of IP addresses;
- They are most commonly used at the transportation layer;
- Firewalls will have configuration to enable data to flow through certain ports while be blocked on other ports;

- A firewall can be configured at the perimeter of the network enabling clients/IPs to view the network and preventing any other information or access;
- Firewalls can be independent networks but could also be programs that could be run anywhere;
- Many cases the functionality of the router and firewalls is performed by the same device;
- Firewalss can run on individual hosts instead of being a network device;
- All major modern os habe firewall functionality built-in - therefore blocking services to ports can be performed at the host level as well;

The Application Layer

- Physical Layer - Computers send electrical/optical signals to send communication across a cable;
- Datalink layer - Individual computers can address each other and send data using the ethernet at the data-link layer;
- Network Layer - Computers and routers communicate between diferent networks using IP.
- Transportation Layer - Data is received and sent by proper applications;

    - The TCP section has a generic data section, the payload section is the entire content of the data application - web pages from server to browser, video content, word-content a wordprocessor is sending to a printer;

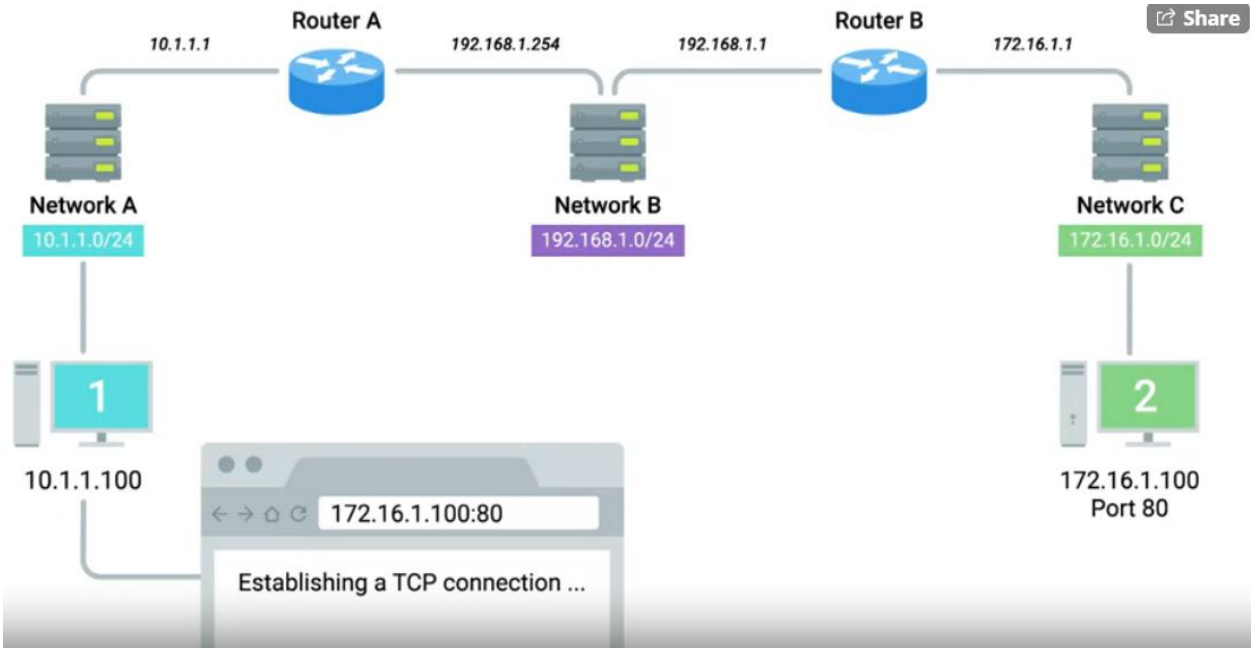
    - Multiple protocols operate at the application layer - HTTP, SMTP, etc - they are standardized across application types;

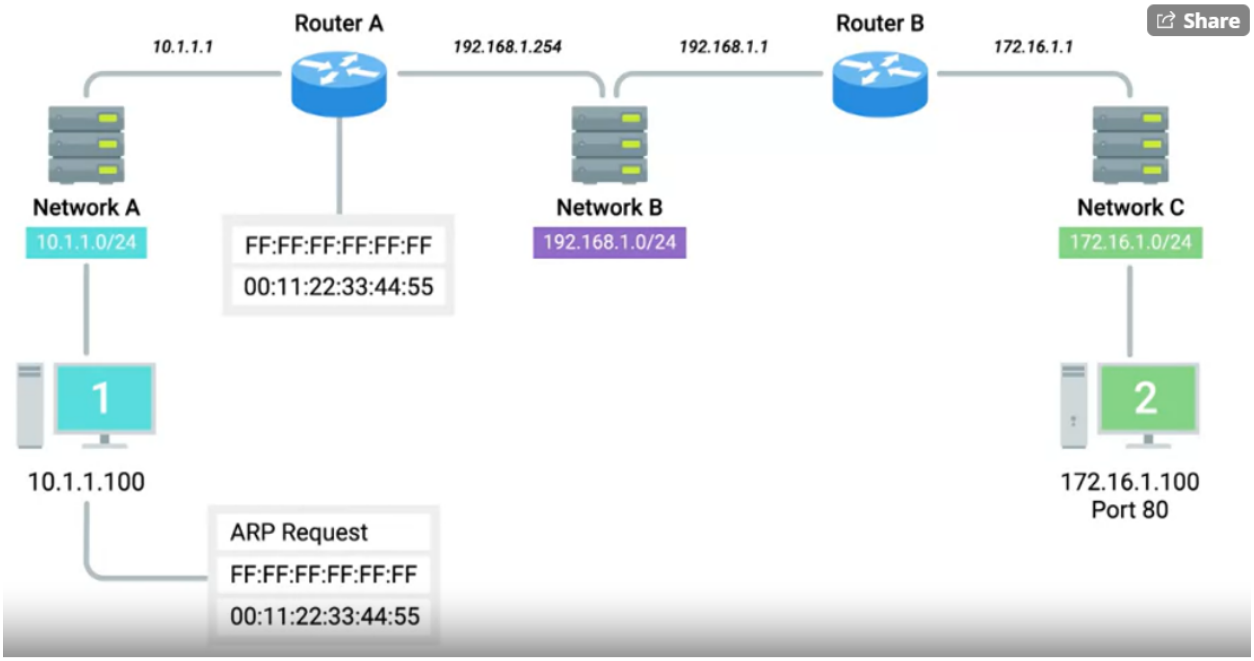
  - Most commom servers - Microsoft IIS, Apache, and nginx for web browsers they need to be compatible and communicate over the same protocol;
  - For Web traffic - application layer protocol - HTTP;

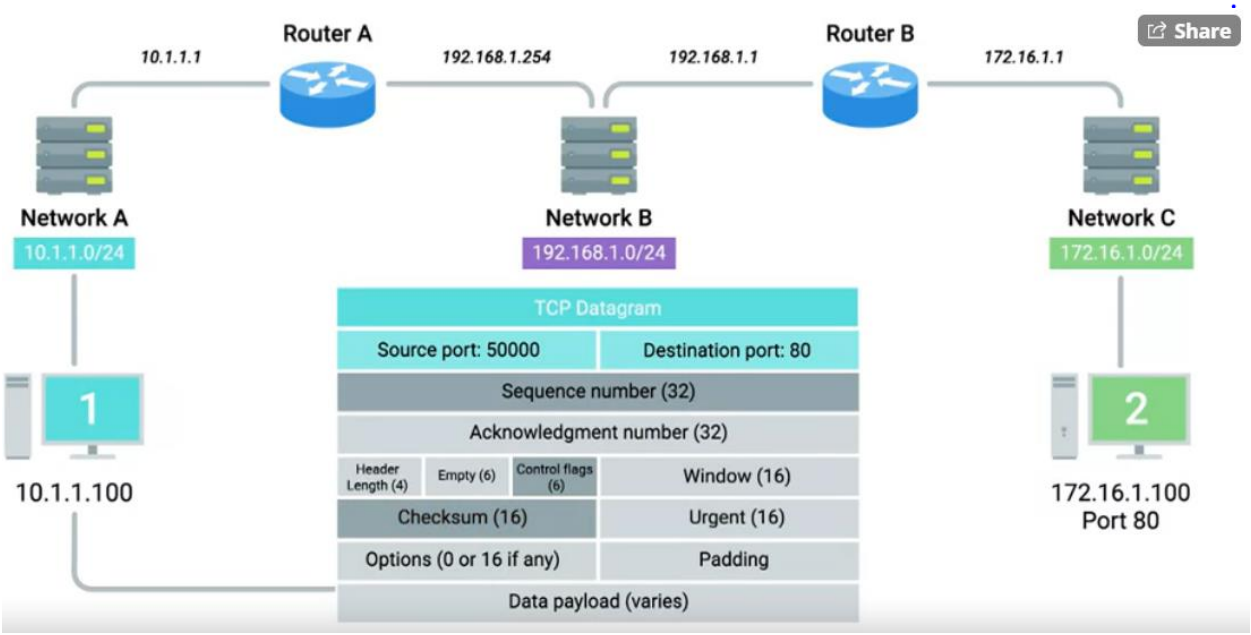
The Application Layer and the OSI Model

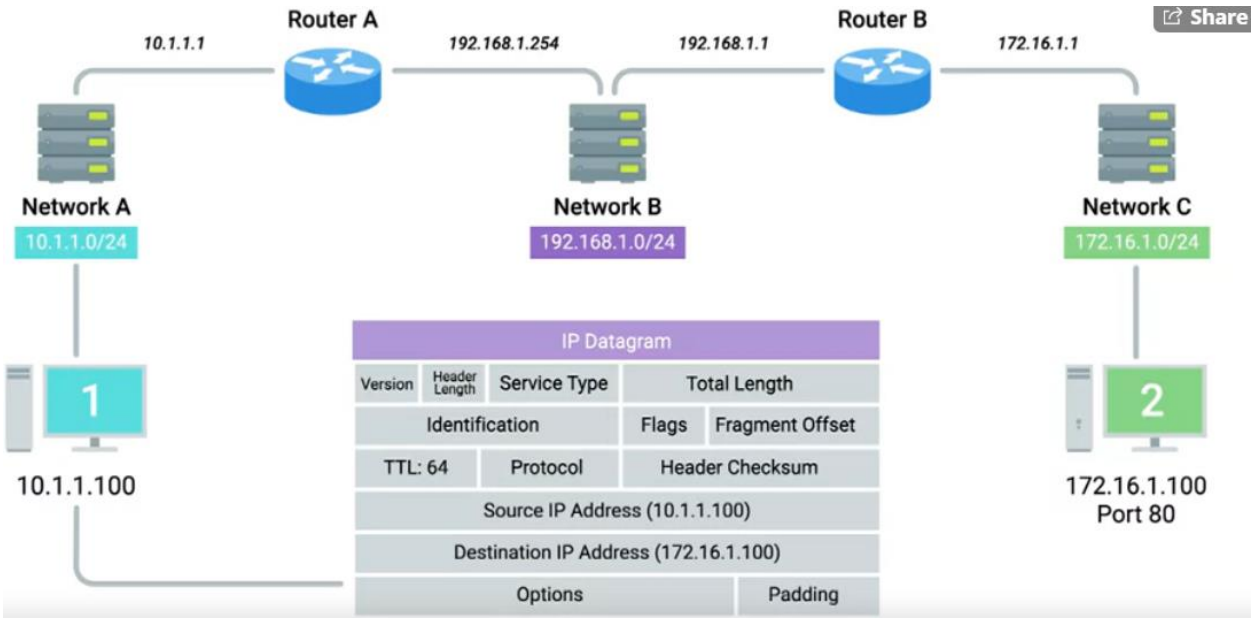
OSI - Open System Interconnection model;
7 layer
---

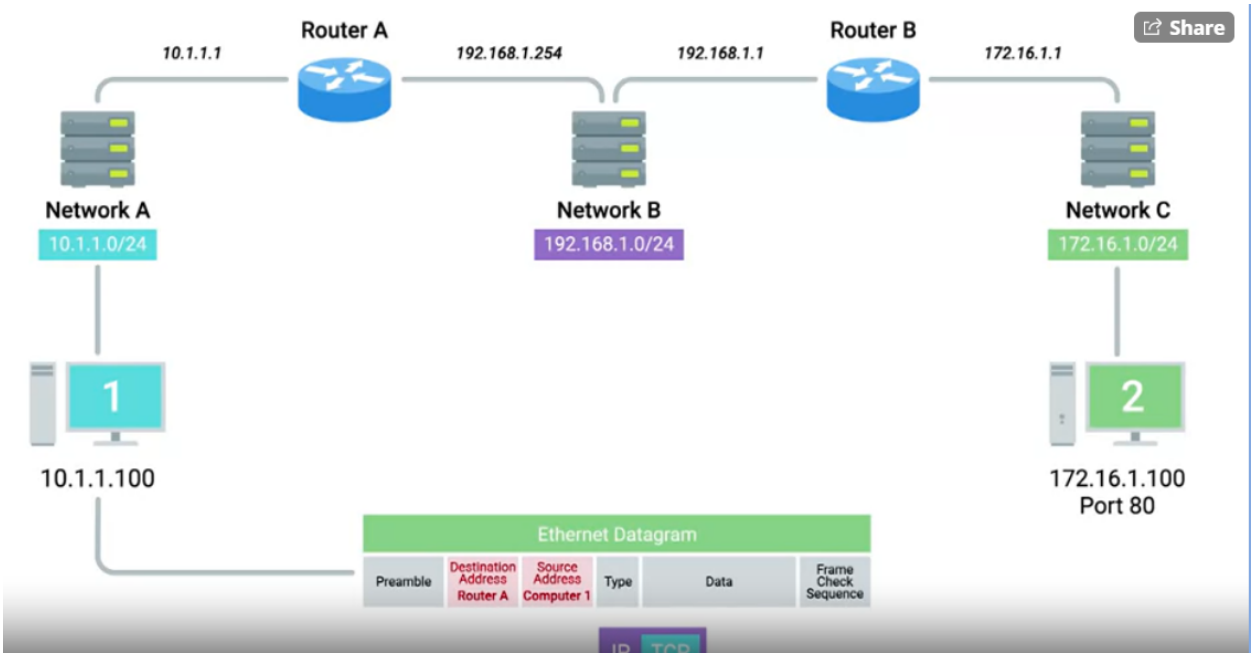
1. Application
2. Presentation - Responsible for making sure that the unencapsulated application layer data is understood by the application. It handles encryption and compression of data;
3. Session - Failitating communication between actual applications and the transport layer, takes the application layer data and sends it to the presentation layer;
4. Transportation
5. Network
6. Data link
7. Physical

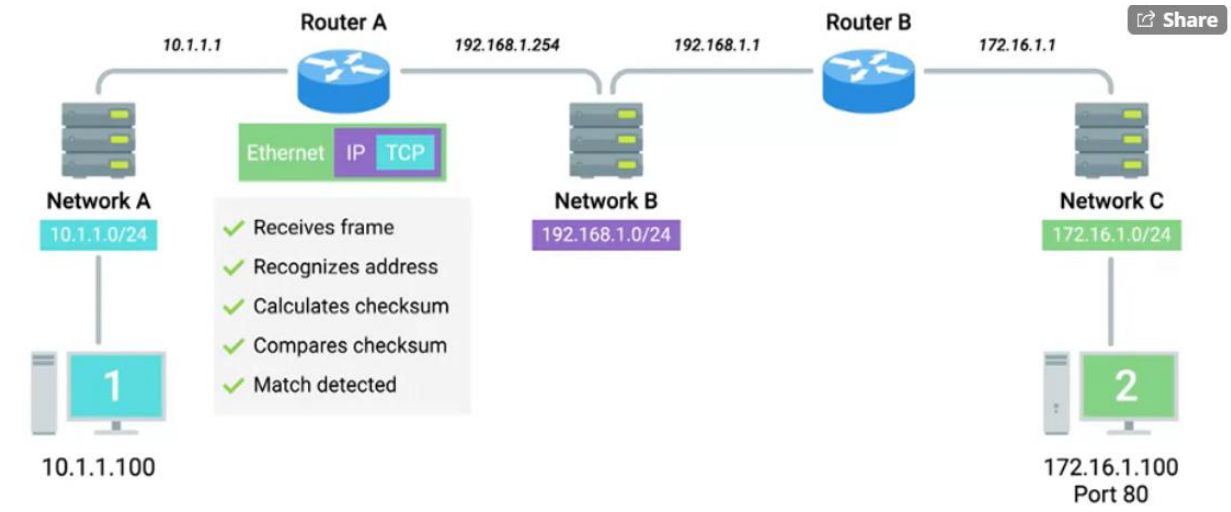


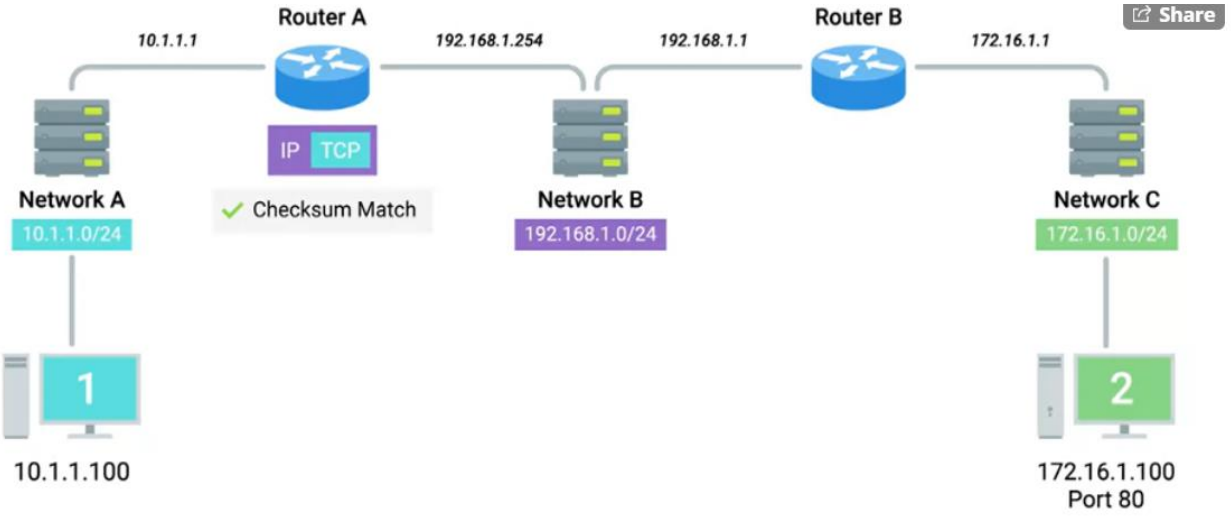

Share
Router A
Router B
10.1.1.1
192.168.1.254
192.168.1.1
172.16.1.1
Network A
10.1.1.0/24
Network B
192.168.1.0/24
Network C
172.16.1.0/24
FF:FF:FF:FF:FF:FF
00:11:22:33:44:55
1
2
10.1.1.100
172.16.1.100
Port 80
ARP Request
FF:FF:FF:FF:FF:FF
00:11:22:33:44:55

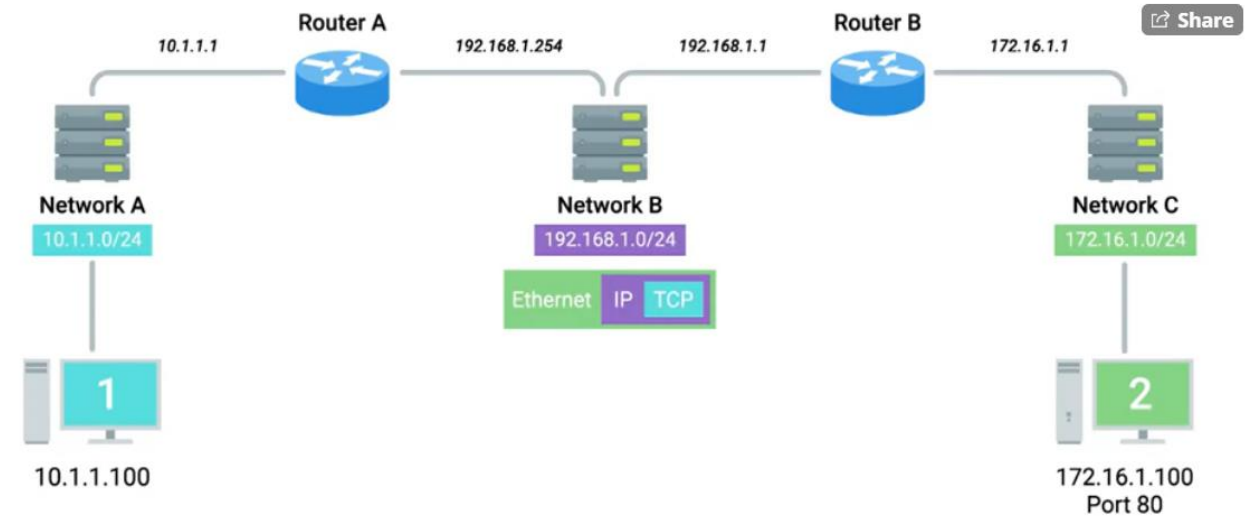

Share
Router A
Router B
10.1.1.1
192.168.1.254
192.168.1.1
172.16.1.1
Network A
10.1.1.0/24
Network B
192.168.1.0/24
Network C
172.16.1.0/24
1
2
10.1.1.100
172.16.1.100
Port 80
TCP Datagram
Source port: 50000
Destination port: 80
Sequence number (32)
Acknowledgment number (32)
Header Length (4)
Empty (6)
Control flags (6)
Window (16)
Checksum (16)
Urgent (16)
Options (0 or 16 if any)
Padding
Data payload (varies)

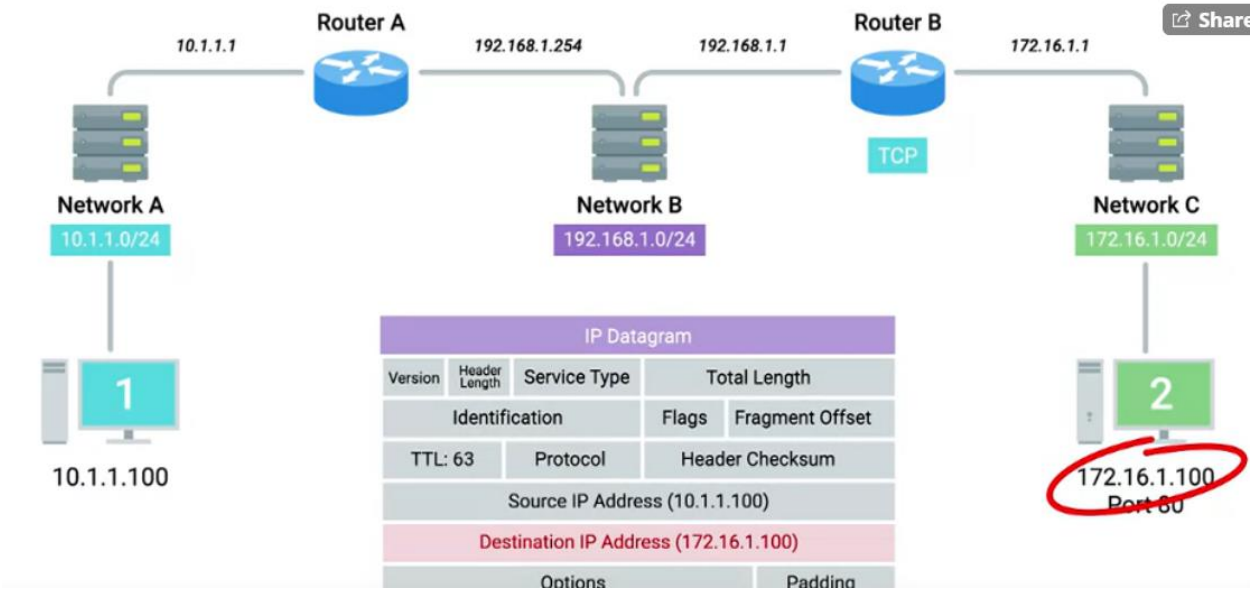

Share
Router A
Router B
10.1.1.1
192.168.1.254
192.168.1.1
172.16.1.1
Network A
10.1.1.0/24
Network B
192.168.1.0/24
Network C
172.16.1.0/24
1
10.1.1.100
2
172.16.1.100
Port 80
IP Datagram
Version
Header Length
Service Type
Total Length
Identification
Flags
Fragment Offset
TTL: 64
Protocol
Header Checksum
Source IP Address (10.1.1.100)
Destination IP Address (172.16.1.100)
Options
Padding

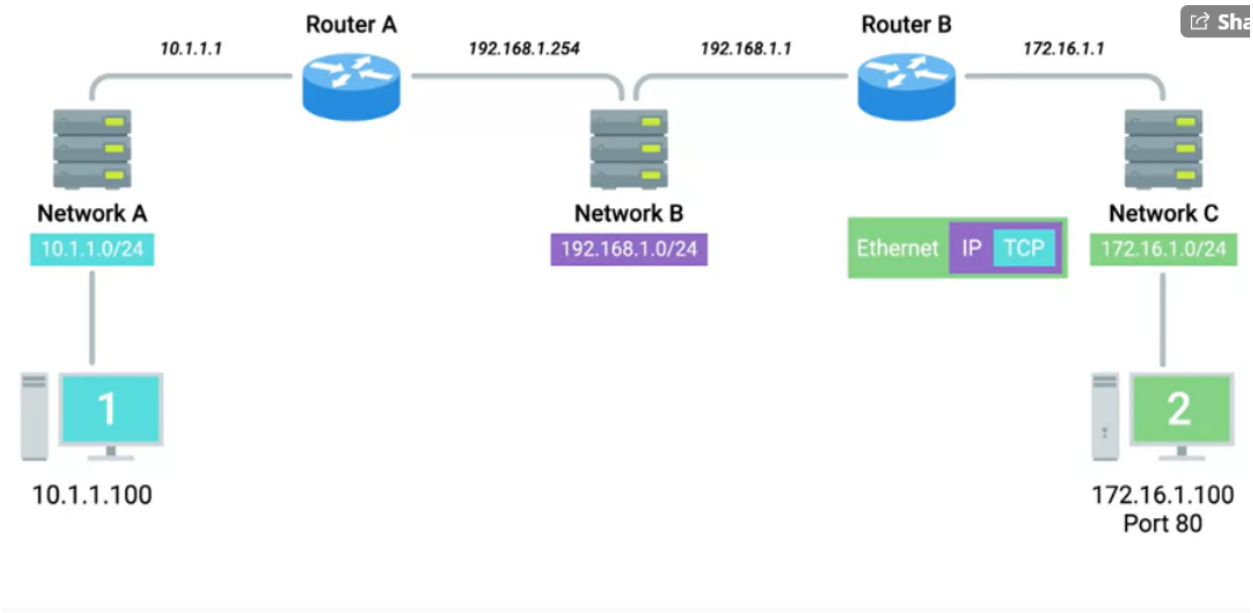

Share
Router A
Router B
10.1.1.1
192.168.1.254
192.168.1.1
172.16.1.1
Network A
10.1.1.0/24
Network B
192.168.1.0/24
Network C
172.16.1.0/24
1
10.1.1.100
2
172.16.1.100
Port 80
Ethernet Datagram
Preamble
Destination Address
Router A
Source Address
Computer 1
Type
Data
Frame Check Sequence


Share
Router A
10.1.1.1
192.168.1.254
192.168.1.1
Router B
172.16.1.1
Ethernet
IP
TCP
Network A
10.1.1.0/24
Network B
192.168.1.0/24
Network C
172.16.1.0/24
Receives frame
Recognizes address
Calculates checksum
Compares checksum
Match detected
1
2
10.1.1.100
172.16.1.100
Port 80


Share
Router A
10.1.1.1
192.168.1.254
192.168.1.1
Router B
172.16.1.1
IP
TCP
Network A
10.1.1.0/24
Network B
192.168.1.0/24
Network C
172.16.1.0/24
Checksum Match
1
2
10.1.1.100
172.16.1.100
Port 80


Share
Router A
Router B
10.1.1.1
192.168.1.254
192.168.1.1
172.16.1.1
Network A
10.1.1.0/24
Network B
192.168.1.0/24
Ethernet
IP
TCP
Network C
172.16.1.0/24
1
2
10.1.1.100
172.16.1.100
Port 80


Share
Router A
Router B
10.1.1.1
192.168.1.254
192.168.1.1
172.16.1.1
TCP
Network A
10.1.1.0/24
Network B
192.168.1.0/24
Network C
172.16.1.0/24
1
2
10.1.1.100
IP Datagram
Version
Header Length
Service Type
Total Length
Identification
Flags
Fragment Offset
TTL: 63
Protocol
Header Checksum
Source IP Address (10.1.1.100)
Destination IP Address (172.16.1.100)
Options
Padding
172.16.1.100
Port 80



3 Networks - computer 1 connected to network a wants data from computer 2 connected to network c

- Web browser on computer 1 -> (url - 172.16.1.100 entered in the address bar) - request the web page from 172.16.1.100;
- The web browser communicates with the local networking stack which is part of the OS - handles n/w functions - that it wants to establish a TCP connection to 172.16.1.100, port 80;
- The networking stack examines its own subnet. It sees that it lives on the network 10.1.1.0/24, which means that the destination 172.16.1.100 is on another network.
- At this point, computer 1 knows that it'll have to send any data to its gateway for routing to a remote network - 10.1.1.1;
- Computer 1 looks at its ARP table to determine what MAC address of 10.1.1.1 is, but it doesn't find any corresponding entry.
- Computer A crafts an ARP request for an IP address of 10.1.1.1, which it sends to the hardware broadcast address of all Fs. This ARP discovery request is sent to every node on the local network.
- When router A receives this ARP message, it sees that it's the computer currently assigned the IP address of 10.1.1.1. So it responds to computer 1 to let it know about its own MAC address of 00:11:22:33:44:55.
- Computer 1 receives this response and now knows the hardware address of its gateway. This means that it's ready to start constructing the outbound packet.
- Computer 1 knows that it's being asked by the web browser to form an outbound TCP connection - therefore needs an outbound TCP port;
- The operating system identifies the ephemeral port of 50000 as being available, and opens a socket connecting the web browser to this port.
- It'll need to establish a connection.

- The networking stack starts to build a TCP segment. It fills in all the appropriate fields in the header, including a source port of 50000 and a destination port of 80. A sequence number is chosen and is used to fill in the sequence number field. Finally, the SYN flag is set, and a checksum for the segment is calculated and written to the checksum field.

- The constructed TCP segment is now passed along to the IP layer of the networking stack. This layer constructs an IP header. This header is filled in with the source IP, the destination IP, and a TTL of 64, which is a pretty standard value for this field.

- The TCP segment is inserted as the data payload for the IP datagram. And a checksum is calculated for the whole thing. Now that the IP datagram has been constructed, computer 1 needs to get this to its gateway, which it now knows has a MAC address of 00:11:22:33:44:55, so an Ethernet Datagram is constructed. All the relevant fields are filled in with the appropriate data, most notably, the source and destination MAC addresses.

- Finally, the IP datagram is inserted as the data payload of the Ethernet frame, and another checksum is calculated.

The network interface connected to computer 1 sends this binary data as modulations of the voltage of an electrical current running across a CAT6 cable that's connected between it and a network switch.

- This switch receives the frame and inspects the destination MAC address. The switch knows which of its interfaces this MAC address is attached to, and forwards the frame across only the cable connected to this interface.
- At the other end of this link is router A, which receives the frame and recognizes its own hardware address as the destination.
- Router A knows that this frame is intended for itself. So it now takes the entirety of the frame and calculates a checksum against it. Router A compares this checksum with the one in the Ethernet frame header and sees that they match. Meaning that all of the data has made it in one piece. Next, Router A strips away the Ethernet frame, leaving it with just the IP datagram.
- Again, it performs a checksum calculation against the entire datagram. And again, it finds that it matches, meaning all the data is correct. It inspects the destination IP address and performs a lookup of this destination in its routing table. Router A sees that in order to get data to the 172.16.1.0/24 network, the quickest path is one hop away via Router B, which has an IP of 192.168.1.1. Router A looks at all the data in the IP datagram, decrements the TTL by 1, calculates a new checksum reflecting that new TTL value, and makes a new IP datagram with this data.
- Router A knows that it needs to get this datagram to router A, which has an IP address of 192.168.1.1. It looks at its ARP table, and sees that it has an entry for 192.168.1.1. Now router A can begin to construct an Ethernet frame with the MAC address of its interface on network B as the source. And the MAC address on router B's interface on network B as the destination. Once the values for all fields in this frame have been filled out, router A places the newly constructed IP datagram into the data payload field. Calculates a checksum, and places this checksum into place, and sends the frame out to network B.
- This frame makes it across network B, and is received by router B. Router B performs all the same checks, removes the the Ethernet frame encapsulation, and performs a checksum against the IP datagram.
- It then examines the destination IP address. Looking at its routing table, router B sees that the destination address of computer 2, or 172.16.1.100, is on a locally connected network. So it decrements the TTL by 1 again, calculates a new checksum, and creates a new IP datagram. This new IP datagram is again encapsulated by a new Ethernet frame. This one with the source and destination MAC address of router B and computer 2. And the whole process is repeated one last time.
- The frame is sent out onto network C, a switch ensures it gets sent out of the interface that computer 2 is connected to. Computer 2 receives the frame, identifies its own MAC address as the destination, and knows that it's intended for itself. Computer 2 then strips away the Ethernet frame, leaving it with the IP datagram. It performs a CRC and recognizes that the data has been delivered intact. It then examines the destination IP address and recognizes that as its own.
- Next, computer 2 strips away the IP datagram, leaving it with just the TCP segment. Again, the checksum for this layer is examined, and everything checks out. Next, computer 2 examines the destination port, which is 80. The networking stack on computer 2 checks to ensure that there's an open socket on port 80, which there is. It's in the listen state, and held open by a running Apache web server. Computer 2 then sees that this packet has the SYN flag set. So it examines the sequence number and stores that, since it'll need to put that sequence number in the acknowledgement field once it crafts the response.
- Finally get a single TCP segment containing a SYN flag from one computer to a second one. Everything would have to happen all over again for computer 2 to send a SYN-ACK response to computer 1. Then everything would have to happen all over again for computer 1 to send an ACK back to computer 2, and so on and so on.