CCNA

ICND-1 Notes

Interconnecting Cisco
Networking Devices,
Part 1

Somenath Sinha

May 2018

Contents

Part I

Fundamentals of Networking

Chapter 1

Network Reference Models and Protocols

1.1 Introduction

When talking about the components of a network such as Network devices and protocols, it's useful to have a common frame of reference. This is provided by the OSI and TCP/IP models. These reference models help us understand how a networking device works simply by comparing its functionality to an equivalent working device in the reference model.

1.2 The OSI Model

The **OSI(*Open Systems Interconnect*)** model is used to categorize or classify network components. It consists of 7 layers - each dealing with a specific type of functionality that a networking device must perform to meet its goal. Thus, certain devices operate primarily on one particular layer of the OSI model. For example, switches *live* (i.e., primarily operate on) Layer 2 of the OSI model while routers live on Layer 3.

The layers of the OSI model are stacked with the lower layers at the bottom, with Layer 1 at the base. Each layer interfaces with both the layers above and below it and passes them *data* in the format that they're expecting.

1.2.1 Terminology related to the OSI Model

To understand the different layers of the OSI stack, we first need to understand some basic terminology and devices:

Terms	Description
Ethernet Switch	A device which allows data to flow between multiple computers in a network and only sends the data to the correct device by the use of a physical address.
Physical Address	A **Layer-2** address *burned into* (programmed within) the Network Interface Card (**NIC**) via which the computer is connected to the Network.

Terms	Description
NIC	A **NIC(*Network Interface Card*)** is a physical device which is used to connect to the network and has physical interfaces (ports) for wired mediums.
MAC Address	In a PC, this physical address burned into the NIC is called the **MAC (*Media Access Control*)** address. It's a 48-bit address typically written in Hexadecimal.
Logical Address	A logical address is an address that's assigned to a device by an external agent (such as a DHCP server) for the primary purpose of routing data within the network and among devices.
IP Address	The IP Address (i.e., the **IPv4 (*Internet Protocol version 4*)** address is a 32-bit logical address used by Layer-3 devices such as routers.
TCP	The **TCP *(Transmission Control Protocol)*** is a framework or set of rules or method to send data from one device to another that's reliable because the receipt of data is acknowledged by the recipient.
UPD	The **UDP *(User Datagram Protocol)*** is a similar protocol to TCP, buUt doesn't acknowledge the receipt of data and thus data loss in transit is possible, making it less reliable, but faster.

1.2.2 The 7 layers of the OSI Model

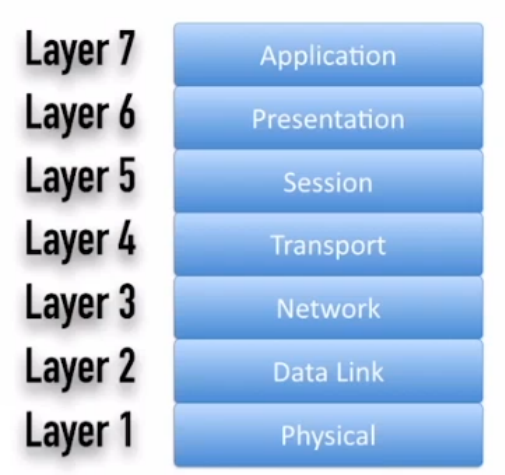


Figure 1.1: OSI Stack

Terms	Description
Physical Layer	This layer is concerned with actually sending the bits (0s & 1s) through the physical media (i.e, wires like Ethernet cable or via wireless mediums). It deals with a way to *electrically* represent them (in case of copper wire based mediums) or *optically* represent them (in case of fiber optics), etc.
Data-Link Layer	Layer-2 devices make forwarding decisions based on a physical address. An Ethernet switch in its basic form is a Layer-2 device. It makes forwarding decisions (i.e, where to send the incoming data) on the basis of a **physical** address, such as the 48-bit MAC Addresses in case of PCs.

Terms	Description
Network Layer	Layer-3 devices such as a router make forwarding decisions based on a **logical** Address such as the IP addresses of the machines in the network. A router analyzes the incoming data on it's ports and determines where to send the data next by finding out which of its other ports lead to the destination IP address.
Transport Layer	The TCP and UDP protocols operate on this layer, and determine how the data is transmitted over the network.
Session Layer	Sets up, maintains and tears down sessions. E.g., **SIP (*Session Initiation Protocol*)** used by IP phones for VoIP calls.
Presentation Layer	Deals with how data is represented on the network. For example, data may be encoded in ASCII (American Standard Code for Information Interchange) or UTF-8(Unicode Transformation Format 8-bit), etc. **Encryption** is also performed within this layer.
Application Layer	The name of this layer is a bit misleading since it is not any application which lives on this layer, but a Network service that allows other desktop applications to take advantage of that service. E.g., The Microsoft Active Directory (AD) service provides the end-user applications with the functionality of logging in to the AD via the network.

Note that it's perfectly possible for a devices to operate in more than one layers, and it's not required to neatly arrange a device or protocol in a single layer. The OSI model is like a book shelf. Similar devices and protocols are arranged in a layer but just like it's possible to have empty shelves in a book-shelf, there's no necessity for a networking device/protocol to have a component present in every single layer.

Since this stack forms the basis of all discussions in the domain of Networking, it's important to memorize the list of the components of the OSI stack and understand what each of the layers in the stack does. If we want to remember the names of the layers from Layer 7 downwards, the acrostic ***A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing* can be helpful. To remember from bottom-up, the acrostic is: ***P**lease **D**o **N**ot **T**hrow **S**ausage **P**izza **A**way*.

An important point to remember is that the OSI model was **designed** to be *generic* and comprehensive, to act as a reference model for more protocols than just IP. While IP can have certain features that live on certain layers, that's just not universally true for other protocols. Most of our networks run on the **TCP/IP(*Transmission Control Protocol/Internet Protocol*)** stack or the **DoD (*Department of Defence*)** stack.

1.3 TCP/IP Stack

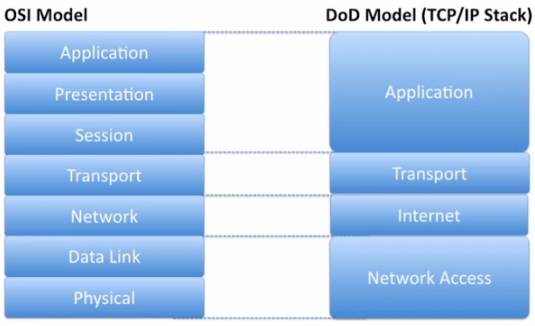


Figure 1.2: TCP/IP Stack

The **TCP/IP(*Transmission Control Protocol/Internet Protocol*)** stack or the **DoD (*Department of Defence*)** model is another reference model that deals directly with the TCP/IP protocols and have a direct mapping to the layers in the OSI stack. This stack was created by the United States Dept. of Defence (DoD) and is thus named after it.

Terms	Description
Network Access Layer	Corresponds to the Physical and Data-link layers of the OSI stack. It's concerned with addressing via physical addresses as well as the representation of the data on physical mediums such as cables. Another name for this layer is the **Network Interface** or the **Link** layer.
Internet Layer	Corresponds to the Network Layer of the OSI stack and deals with logical addressing via the IPv4 and IPv6 protocols.
Transport Layer	This is the equivalent of the Transport Layer of the OSI stack and also deals with the same function of determining the mode of transport, i.e., the protocol to be used for data transmission.
Application Layer	This is the equivalent layer to the combination of the Application, Presentation and Session layers of the OSI stack, and performs all their functions.

Note that some literature that discusses the DoD stack show it as a 5-layer model with the bottom Network Access Layer sub-divided into Data-link and Physical layers:

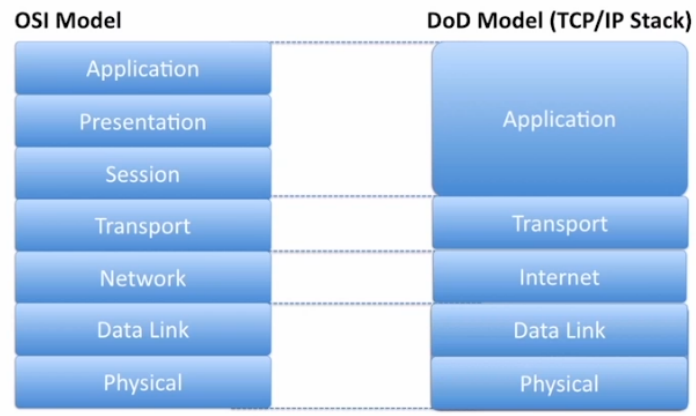


Figure 1.3: TCP/IP Stack Alternate Model

Some may even choose to call this Data-link layer in the TCP/IP stack as the Network Interface Layer. Irrespective of the naming, the functioning of these layers remain consistent.

1.4 TCP/IP Protocol Suite

The TCP/IP protocol suite consists of **IP, ICMP, TCP** and **UDP** protocols. In Layer 3, the Network layer of the OSI stack or the Internet layer of the TCP/IP stack, some of the most useful protocols are IP and ICMP.

IP (*Internet Protocol*) is used to forward *packets* of data to the right (intended) recipient. **ICMP (*Internet Control Message Protocol*)** is used to test the reachability of remote network devices by *pinging* them and can also report error conditions in a network.

1.4.1 Layer-3 (Network/Internet Layer) Protocols

Internet Protocol (IP)

IP is a protocol which contains the data of other higher layer protocols as it's payload. Thus, it's payload consists of segments from UDP or TCP that need to be encapsulated within IP packets to be correctly formatted and sent over the network. Thus, routers, which are layer-3 devices can make forwarding decisions based on the logical address (destination IP address) packed by following the Internet Protocol.

ICMP

A really common use of the **ICMP (*Internet Control Message Protocol*)** is to perform *pings*. The **ping** utility is used to test if one network device is reachable from another network device.

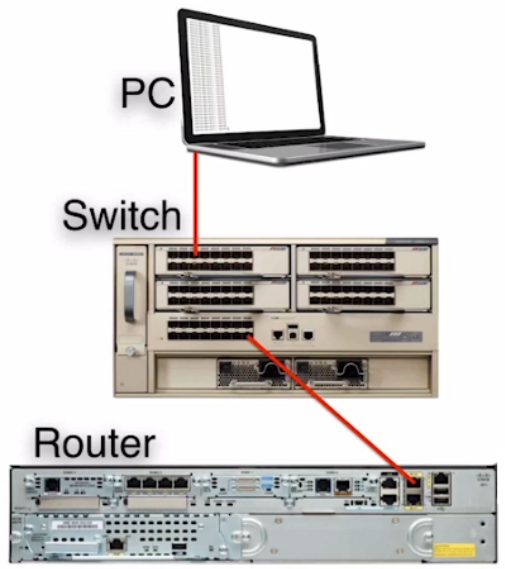


Figure 1.4: Pings using ICMP

Let us consider the scenario above. A PC is connected to a switch which in turn is connected to a router. The switch, a layer-2 device, has two connections - one to the Ethernet port on the PC, which has a MAC address and another to the router's port, which also has it's own MAC address. These MAC addresses (and which port on the switch they're connected to) are known to the switch, which is how it makes forwarding decisions and facilitates the transmission of data.

Now, if the PC can't connect to the internet, during the troubleshooting, one of the first things that we do is to check if we can *reach* the next hop router (the router that connects us to everybody else on the internet/network) from the PC. For this we can use a utility called **ping** that's built into most **OS (*Operating Systems*)**. When ping is used, the PC sends out an **ICMP Echo Request** to that router's IP address. If the router receives that echo request (and if the settings permit it), the router then sends an **ICMP Echo Reply** back to the PC.

The PC can then display that the router is reachable and tell us the *round-trip time*, i.e., the amount of time it took for the packet to reach the router from the PC and then for the ICMP echo reply to acknowledge connectivity to reach the PC.

If our **next-hop router** or our **default gateway**, i.e., the devices that connects us to the rest of the network, has an IP address of `10.10.30.1`, we could ping it from the command line using (The `-c` option specifies the number of *ICMP echo requests* to send.):

```
# ping -c 4 10.10.90.1
PING 10.10.90.1 (10.10.90.1) 56(84) bytes of data.
64 bytes from 10.10.90.1: icmp_seq=1 ttl=64 time=2.06 ms
64 bytes from 10.10.90.1: icmp_seq=2 ttl=64 time=1.31 ms
64 bytes from 10.10.90.1: icmp_seq=3 ttl=64 time=1.42 ms
64 bytes from 10.10.90.1: icmp_seq=4 ttl=64 time=3.37 ms

--- 10.10.90.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 1.313/2.044/3.377/0.821 ms
```

1.4.2 Layer-4 (Transport Layer) Protocols

User Datagram Protocol (UDP)

UDP is a connection-less and unreliable protocol, because it doesn't receive acknowledgments for the segments that it transmits. While sending a UDP segment, while we hope that it reaches its destination, there's no guarantee that the data will reach the recipient. Further, there's no retransmission for data that failed to reach the destination because we don't know which data was received and which wasn't.

Thus, UDP is used for real-time network applications such as in VoIP phones, where it's not important that every segment of data reach the destination, but the speed at which the segments flow is important. This is applicable for VoIP since we don't care if there's a slight stutter, but delay in communications (such as lag in voice and video, etc.) is unacceptable. Further, the use of TCP in such an application would be an extra overhead that would cause further delays, since a TCP header is much larger than the header on an UDP segment. Another reason TCP isn't used for VoIP is because even if a dropped segment is retransmitted, it may (and probably will) arrive out of order and is thus useless.

Transmission Control Protocol (TCP)

TCP is a connection-oriented protocol that's reliable since it can detect if segments are *dropped*, i.e., lost in transit. In TCP, a **connection** is set up between the two parties involved in the communication and then the data transfer occurs, with acknowledgement for each of the segments received from the recipient to the sender.

A **Three-way handshake** is performed by the two-parties involved to establish a TCP connection. The steps involved are:

- The sender sends a **SYN** (Synchronization message) to indicate that the sender wants to set up a *session*.
- The receiver now needs to send back a **SYN+ACK** message, i.e., acknowledge the `SYN` message from the sender and then send a `SYN` message of its own to the sender.
- The final step in the 3-way handshake is for the sender to reply to the `SYN` message from the recipient.

Now the TCP session is set up. One feature of TCP is that we can send varying amounts of data before expecting an acknowledgement, called TCP windowing.

Sliding Window Protocol & TCP Windowing

TCP's sliding window protocol deals with its ability to vary the amount of data that a sender can send before expecting an acknowledgement. If we're on a highly reliable network, it'd be prudent to send more data at one time before expecting an acknowledgement, because there's less time wasted waiting for acknowledgement. This is called having a *larger window size*. TCP can try to exponentially grow that window size. Let us consider the following case:

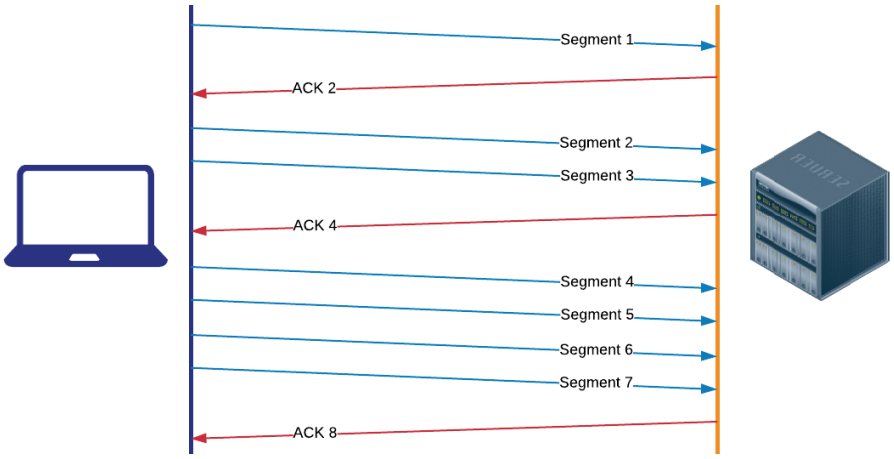


Figure 1.5: TCP's Sliding Window Protocol

Let us consider the laptop on the left is trying to communicate with the server on the right. After the three-way handshake is complete, the laptop sends the first segment of data. The server responds to this with and acknowledgement, and asks for the next segment, with **ACK 2**. The laptop now doubles the number of segments that it sends to 2. The server again responds with an acknowledgement, and demands segment 4 with `ACK 4`. Once more the laptop doubles the number of segments it sends, this time to 4. The server, again responds with `ACK 8`.

This doubling continues till no acknowledgement is received from the server. When that happens, the laptop realizes that it's sending data too aggressively. This might mean that either a packet was dropped, or maybe the sender needs to wait longer for the acknowledgement. Now the sender (i.e., laptop) is going to drop its window size back down and grow at a slower rate, much more cautiously.

1.5 Domain Name System (DNS)

Websites are content hosted on a server accessible by connecting to the IP address of a server (on the port on which the web-server is running). Thus, each public website on the internet has its own public IP. We could connect to the website using its IP, but it's impossible to remember the IPs for such a large number of websites. This is why we have domain names - an organized way to store information about websites. Domain names are hierarchical in nature for ease of indexing and categorization. To connect to a website, we can use its **FQDN (*Fully Qualified Domain Name*)**, a complete domain name specifying each level in that server's DNS hierarchy, instead of its IP. The job of a **DNS (*Domain Name***

System) server is to map each FQDN to an IP address to which our host machine can connect.

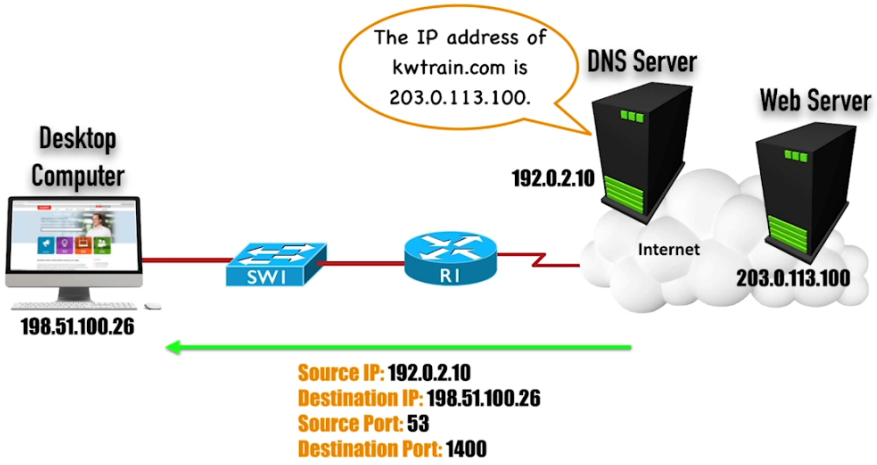


Figure 1.6: DNS lookup

Let us consider the PC above. It has the IP address of the DNS server set - either through manual settings or via the settings provided by DHCP. If we try to go to `kwtrain.com` on the PC, a DNS query is issued to lookup its IP. The query reaches port 53 on the DNS server, which either passes on the IP address if previously saved on it, or performs a recursive lookup on other DNS servers till the IP address is found. Now the client PC can send HTTP/HTTPS data to the web server and the web server can respond with the requested data.

1.5.1 Well-known Port

In a single session between two networking devices, there's a source port and a destination port number present. These ports act as the source of the incoming data/input and the port to which the output is sent. The port numbers in the range **0 - 1023** are called well-known ports, and common protocols such as DNS are assigned a port number in this range. DNS is assigned the port number 53 by default.

1.5.2 Hierarchical DNS Structure

There isn't just one centralized master DNS server that handles all DNS queries, but a distributed system responsible for handling the queries in their DNS zones. There are several different root or **TLD (*Top Level Domains*)** such as `.com` and `.edu`, the IP addresses of which are known to the **root name server**.

Whenever a new unknown domain is queried, a DNS server checks its cache for any name server that might lead it to the queried name server. For example, if we want to go to `science.purdue.edu`, then our DNS server checks it's caches. If it can't find the DNS record for `science.purdue.edu`, it tries to check if the DNS record for `.edu` zone's name server is in it's cache.

If not, since `.edu` is a TLD, it queries the root name server for `purdue.edu`'s IP. This is the first step in a recursive lookup process, where the DNS server queries each successive domain zone's server for the next domain till the required DNS record has been found. So, once the DNS server has the IP for the name server responsible for `purdue.edu`, it will query it for the `science.purdue.edu` and once it has it, the result will be returned to the host which originally asked for it.

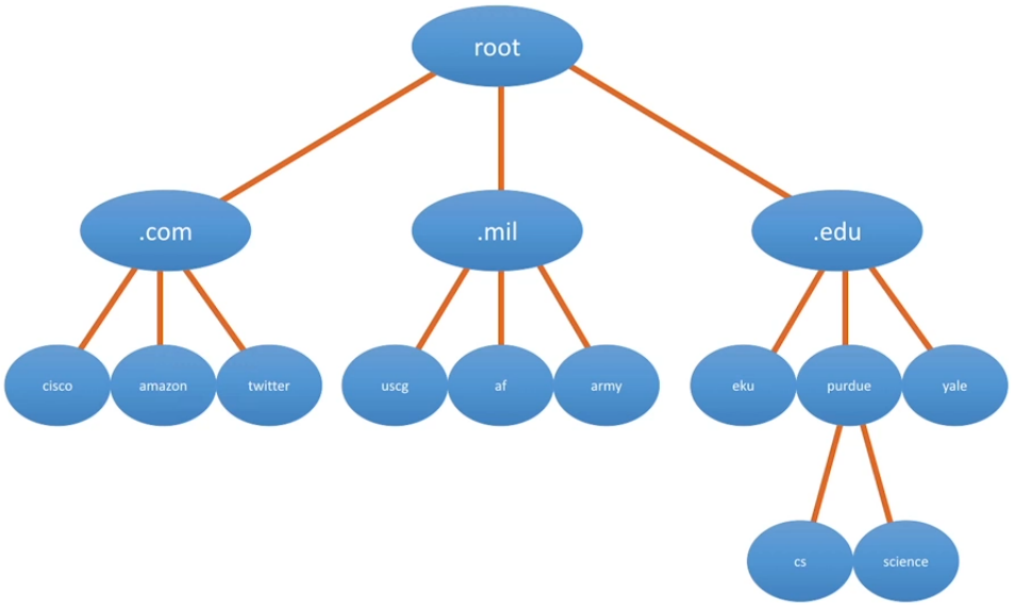


Figure 1.7: DNS Hierarchy

1.5.3 DNS Record Types

A DNS server or a name server doesn't simply map FQDNs to IP addresses, but has a database of several types of records:

Terms	Description
A	An **Address Record** is used to map a hostname to an IPv4 Address.
AAAA	An **IPv6 Address Record** is used to map a hostname to an IPv6 Address. There are 4 A's in the name because an IPv6 Address is 128-bits, i.e., 4 times as long as a 32-bit IPv4 Address.
CNAME	A **Canonical Name Record** is an alias of an existing record, which allows the DNS records for multiple hostnames to map to the same IP Address.
MX	A **Mail eXchange** record maps a domain name to an e-mail (or message transfer agent) server for that domain.
PTR	A **Pointer** record points to a canonical name. It's used to perform *reverse DNS lookups* where we find the domain name associated with a known IP address.
SOA	A **Start of Authority** record provides authoritative information about a DNS Zone, such as email contact information for the zone's administrator, the zone's primary name server and various refresh timers. Further, when an SoA record is found, we know that the name server responsible for that zone itself is providing the information we need.

1.5.4 Dynamic DNS (DDNS)

It might be the case that our server is hosted with a dynamic IP, acquired from a DHCP server from our ISP. This means that the IP is bound to change periodically. Normally this'd mean that our DNS records would need to be *manually* updated too, to reflect the changes and ensure that the lookup process isn't broken, but that's unnecessary with **DDNS** (***Dynamic DNS***).

In DDNS, a software (*DDNS service provider*) on the network monitors the changing IP address for an associated hostname, and once a change it detected, it automatically updates the DNS records for that hostname in the DNS tables in the name server responsible for the zone in which our host resides. From this point, further queries from other name servers propagate the new DNS record. Thus, the FQDN is always mapped to the right IP address.

1.6 Ports and Protocols

The TCP and UDP protocols, both residing on Layer-4 of the OSI or DoD stack, are designed to encapsulate and carry the data payload for our applications, i.e., the data sent from Layer-5 or above through the network.

Frequently, multiple applications are running on a single server with a given IP address. For example, a server may host applications that allow it to act as both the web server and the FTP server for a domain. If we try to access a web page in that domain, a query from our computer is sent to the server demanding the relevant info encapsulated within a TCP segment.

When this segment reaches the destination server, there needs to be a way for the server to tell if the data within the segment is meant for the web server or the FTP server. This is why these applications are associated with a **port number**. There are a set of standardized set of **well-known port numbers** that are the port numbers generally associated with applications, such as port 80 for HTTP, port 443 for HTTPS, port 53 for DNS, etc.

However, there are also un-assigned ports that can be used by any application, which range from **1024** to **65535**. These are used by applications to send data to another application on the network and receive replies from them.

1.6.1 Communication using Ports

Let us consider the situation below - a client with the IP `10.1.1.1` wants to access a web-server on the IP `172.16.1.2`. Since the application is a web-server, we know that the well-known port associated with is port `80`. So, the packet containing the query to access information on the web server is sent to `172.16.1.2:80` from the client by an application. Let's say the application uses the port `1248`. Since the application is requesting information from a web-server, the TCP protocol is used. Thus, we can say the request is made to *TCP port 80* on the web server.

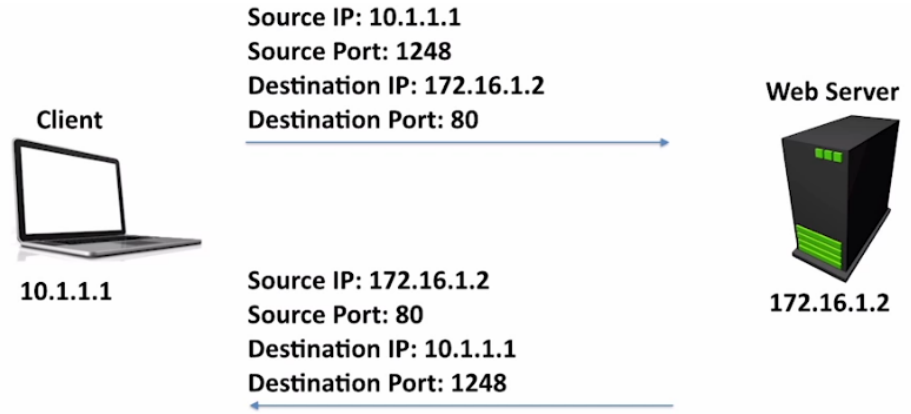


Figure 1.8: Port Numbers

Now, when the query is processed by the web server and a reply is sent back to the client, the packet will have a source IP of `172.16.1.2` from port `80`, and the destination IP will be the origin of the request, i.e., IP of the client, `10.1.1.1`. The reply will be sent to the same port on the client where the application that made the query is awaiting information, i.e., port `1248`.

1.6.2 List of some Well-Known ports

Protocol	Description	TCP Port	UDP Port
FTP	**File Transfer Protocol:** Transfers files with a remote host (typically requires authentication of user credentials)	20 and 21	
SSH	**Secure Shell:** Securely connect to a remote host (typically via a terminal emulator)	22	
SFTP	**Secure FTP:** Provides FTP file-transfer service over an SSH connection	22	
SCP	**Secure Copy:** Provides a secure file-transfer service over an SSH connection and offers a file's original date and time information, which is not available with SFTP	22	
Telnet	**Telnet:** Used to connect to a remote host (typically via a terminal emulator)	23	
SMTP	**Simple Mail Transfer Protocol:** Used for sending e-mail	25	
DNS	**Domain Name System:** Resolves domain names to corresponding IP addresses	53	53
TFTP	**Trivial File Transfer Protocol:** Transfers files with a remote host (does not require authentication of user credentials)		69
DHCP	**Dynamic Host Configuration Protocol:** Dynamically assigns IP address information (for example, IP address, subnet mask, DNS server's IP address, and default gateway's IP address) to a network device		67
HTTP	**Hypertext Transfer Protocol:** Retrieves content from a web server	80	
POP3	**Post Office Protocol version 3:** Retrieves e-mail from an e-mail server	110	
NNTP	**Network News Transport Protocol:** Supports the posting and reading of articles on Usenet news servers	119	
NTP	**Network Time Protocol:** Used by a network device to synchronize its clock with a time server (NTP server)		123
SNTP	**Simple Network Time Protocol:** Supports time synchronization among network devices, similar to Network Time Protocol (NTP), although SNTP uses a less complex algorithm in its calculation and is slightly less accurate than NTP		123
IMAP4	**Internet Message Access Protocol version 4:** Retrieves e-mail from an e-mail server	143	
LDAP	**Lightweight Directory Access Protocol:** Provides directory services (for example, a user directory—including username, password, e-mail, and phone number information) to network clients	389	
HTTPS	**Hypertext Transfer Protocol Secure:** Used to securely retrieve content from a web server	443	
rsh	**Remote Shell:** Allows commands to be executed on a computer from a remote user	514	
RTSP	**Real Time Streaming Protocol:** Communicates with a media server (for example, a video server) and controls the playback of the server's media files	554	554
RDP	**Remote Desktop Protocol:** A Microsoft protocol that allows a user to view and control the desktop of a remote computer	3389	

Figure 1.9: Well known ports

1.6.3 FTP (File Transfer Protocol)

The **FTP (*File Transfer Protocol*)** is used to transfer files over a network. For example, in an office if we want our supervisor to review a report that we wrote, but we're working remotely, then we could authenticate to our department's FTP server with our username and password, and upload the file. Then our supervisor can log in with his credentials, make any required changes or recommendations, and save it. Then we can pull the file back down from the FTP server, and view the changes.

Although there is an authentication in FTP, it's not secure because of the fact that the content as well as the username and password are sent over the network in an unencrypted (i.e., plain text) format and thus anyone snooping on our communications can get unauthorized access to our files. This problem is resolved using the **SFTP (*Secure File Transfer Protocol*)** instead, which uses FTP through a SSH tunnel. This is similar to SCP which copies file over the network within a SSH tunnel.

1.6.4 SSH (Secure SHell)

The **SSH (*Secure SHell*)** Protocol is used to log on to other computers/network devices on the network and access files/configurations on them. It uses *TCP port 22*, and it's a secure protocol, i.e., the information in the packets in the network layer are encrypted.

1.6.5 SMTP, IMAP and POP3

The **SMTP (*Simple Mail Transfer Protocol*)** is used to **send** mail from our devices (PC, laptop, smart-phones, etc.) to a mail server for a domain from *TCP port 25*, but to retrieve mail from our mail servers, we need a protocol like **IMAP4 (*Internet Message Access Protocol v4*)** [*TCP port 143*] or **POP3 (*Post Office Protocol v3*)** [*TCP port 110*].

A problem that arises frequently with POP3 is that when multiple devices are accessing the same mailbox, mails can go missing. This is because POP3 after downloading mails from the server and storing them on the local computer/device, POP3 deletes them from the mailbox, thus making them unavailable from other devices.

1.6.6 DNS

The **DNS (*Domain Name System*)** servers use port 53 (either *TCP port 53* or *UDP port 53*, but can use both) to resolve Fully Qualified Domain Names (FQDN) to IP addresses.

1.6.7 TFTP

The **TFTP (*Trivial File Transfer Protocol*)** is similar to FTP but doesn't require any authentication for file transfer, using *UDP port 69*. It's typically used internally within a controlled environment to provide files for operations like **PXE boot** (i.e., network booting).

1.6.8 DHCP (Dynamic Host Configuration Protocol)

The **DHCP (*Dynamic Host Configuration Protocol*)** is used by a computer/smart phone or any other network device boots up on the network and needs to be assigned an IP address. It then can request the DHCP server on the network (*UDP port 67*) for an IP address, which the DHCP server gives to it, along with some added information, such as the subnet mask, the default gateway IP, the DNS server IP, etc.

1.6.9 HTTP, HTTPS

The **HTTP (*HyperText Transfer Protocol*)** [*TCP port 80*] and **HTTPS (*HyperText Transfer Protocol - Secure*)** [*TCP port 443*] are used to access content on a web-server. Webpages on the internet are served from web-servers using of these two protocols.

1.6.10 NTP, SNTP

The **NTP (*Network Time Protocol*)** is used by network devices to synchronize their internal clocks using a time server (**NTP Server**). The **SNTP (*Simple Network Time Protocol*)** does the same job as NTP, but uses a less complex algorithm that generates a output that is less precise. Both NTP and SNTP use UDP port 123.

Both of the above protocols are especially useful for troubleshooting multiple devices, since if the clocks on multiple devices are synchronized, we can use the timestamps on them to determine if events might be correlated and/or if problems started on multiple devices at the exact same time (indicating the point of origin for the issues is external).

1.6.11 LDAP

The **LDAP (*Lightweight Directory Access Protocol*)** uses *TCP port 389*. It allows organizations to have a centralized repository of user information (username, password, email information, etc.) and multiple services can leverage this single database to provide a single sign on (SSO) system.

1.6.12 rsh

The **rsh (*remote shell*)** protocol is used in *nix (UNIX, Linux, etc.) shells to remotely execute commands on other *nix hosts on *TCP port 514*.

1.6.13 RealTime Streaming Protocol (RTSP)

The **RTSP (*RealTime Streaming Protocol*)** allows us to stream video content from a video server, using either *TCP port 554* or *UDP port 554*, allowing us to control the playback of the video coming from that server.

1.6.14 Microsoft's Remote Desktop Protocol (RDP)

This protocol isn't strictly well-known in the sense that it's port number is *TCP port 3389*, but it's still known industry-wide. It allows a user to view and control the desktop of a Microsoft Windows computer from another computer.

1.7 Protocol Data Units (PDU)

A **PDU (*Protocol Data Unit*)** is the information that is transmitted as a single unit among peer entities, residing on the same layer in a computer network. This means if the sender sends a *packet* of information in a certain format from it's layer-3, the recipient at the destination receives the same packet of information in the exact same format in its layer-3. Simply speaking, it's what we call data at a layer, specifically one among the bottom 4 layers of the OSI model. The PDUs for the layers of the OSI stack are:

Layer	PDU
Transport	**TCP:** Segments; **UDP:** Datagrams
Network	Packets
Data-link	Frames
Physical	Bits

In the physical layer, we're concerned with sending data *on the wire*, i.e., through a transmission medium in the form of bits. Hence, it's PDU is bits. In the data-link layer, we have switches, which make forwarding decisions based on the information in frames, which is it's PDU. Routers on the Network Layer transmit packets among themselves, which is the basis for its PDU. Further, the term **packet** is used to generically used to refer to traffic at any layer. At the Transport layer, TCP PDUs are called **Segments**, and UDP PDUs are called **Datagrams**. It's okay to refer to both as segments generically.

An acrostic to remember the PDUs in a top-down fashion, i.e., from the Transport layer to the Physical layer is : ***S**ome **P**eople **F**ear **B**irthdays*. To remember it in a bottom-up, i.e., from Physical to Transport layer, we use the acrostic ***B**acon **F**rying **P**roduces **S**alivation*.

Chapter 2

Infrastructure Components

With the help of the OSI and DoD reference models, we can describe the behaviour and operation of the common network infrastructure devices.

2.1 Common Network Infrastructure Devices

The figure below shows some of the common network devices that we might find in modern networks. The sample network topology consists of 3 locations: the **HQ** (*Head Quarters*), **Br1**(*Branch 1*) and **Br2**(*Branch 2*).

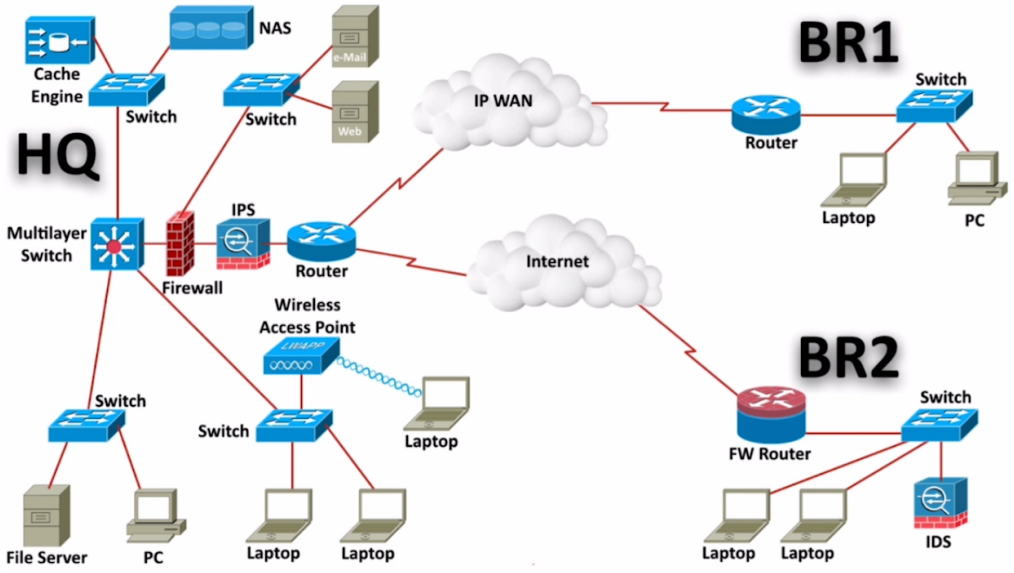


Figure 2.1: Sample Network Topology

2.1.1 Router

A router is a layer-3 device that makes forwarding decisions based on the destination IP (logical/layer-3 address) of the packet. We might be dealing with the IPv4 or IPv6 variance of the address, but the operating principles remain the same. A router will take a look at the destination address of an incoming packet,

and decide which port to forward the packet based on the networks connected to each of the other active ports, by looking up the network in it's **routing table**.

2.1.2 Wide Area Network (WAN)

The thunderbolt icon in the network topology diagrams represent a **WAN (*Wide Area Network*)**, a network connection that interconnects geographically separated networks. The two networks on either side of the WAN link aren't located within a building or a campus, but much further apart, such as different cities, countries, etc.

In the figure 2.1 *Sample Network Topology*, HQ is connected to Br1 via an IP WAN link (a dedicated connection), but to Br2 via the Internet. Both are examples of WANs. While the WAN link to Br1 is private and hence secure, the WAN link to Br2 is through the internet, which would mean that ordinarily our private and sensitive data is exposed to the internet. This requires the use of a VPN to keep the transmission secure.

2.1.3 Virtual Private Network (VPN)

A **VPN (*Virtual Private Network*)** allows us to set up a secure connection over an untrusted network, such as the internet. We could form a VPN tunnel that provides end-to-end encryption on our data while it travels through the internet. So, even if someone were to intercept our data, it'd remain undecipherable and hence, harmless. Further, due to the use of the VPN tunnel, we can be sure that the data hasn't been corrupted in transit.

2.1.4 End-user stations

End-user stations are devices on which end users can log in and gain access to our network. These can be PCs, Laptops, smart-phones, etc. though which they can connect to network resources such as *file servers* or *network printers*.

2.1.5 Wireless Access Points (WAP)

The end-user stations can connect wirelessly to our network using **WAP (*Wireless Access Points*)**, which is a device that communicates with wireless devices such as smartphones, printers and laptops and interconnects and integrates those devices into an existing wired network.

2.1.6 Switch

The Wireless Access Point (WAP), along with all the other end-user stations can connect to a switch, a layer-2 device capable of making forwarding decisions based on the layer-2 physical address, i.e., the MAC address. The MAC address is burnt into each NIC (Network Interface Card) contained within each end-user workstation and network resource, i.e., each network connected device. The switch starts building a table that maps each port to a particular MAC address. Then, when a frame has to be forwarded to a particular MAC address, it simply sends the frame through the port with the matching MAC address. Every port on a switch is in its own collision domain.

2.1.7 Intrusion Prevention System (IPS)

An **IPS (*Intrusion Prevention System*)** is a sensor that sits in-line with the network traffic and analyses it. The signature of well-known attacks is available to the IPS, and when the signature of a packet in the traffic matches that of an attack, it can drop it, thus preventing the attack from ever reaching the network. Thus, every incoming packet from the internet passes through the IPS (from the edge-router) before it can reach the firewall.

2.1.8 Intrusion Detection System (IDS)

An **IPS (*Intrusion Detection System*)** is a sensor that receives a copy of the network traffic and analyses it. Just like an IPS, the signature of well-known attacks is available to the IDS in a database, and when the signature of a packet in the traffic matches that of an attack, it can instruct the router to block all packets coming from that source on the internet. An example is in Br2, where the switch has been trained to send a copy of all incoming packets to the IDS. However, by the time the IDS detects an attack, some packets might have already reached the network, and might have done some damage. Thus, an IPS is more secure than an IDS.

2.1.9 Firewall

A firewall is a network device that analyses the incoming and outgoing traffic based on a set of rules to determine which traffic to allow to pass through it and which to deny, between different portions (i.e., **zones**) of a network. In the figure 2.1's HQ section, the firewall has a port that connects the outside of the network (the internet) to the inside of the network The port connecting to the internet is called the *outside interface* while the port connecting to the internal network is called the *inside interface*. It allows us to define rules that can do things like: *block all traffic that initiates on the internet*.

2.1.10 Demilitarized Zone (DMZ)

A **DMZ (*Demilitarized Zone*)** is a section of the network that should be accessible from the outside of the network, on external devices, such as devices on the internet. In the following figure, the HQ has a DMZ containing the web server and the email server such that those functionalities aren't hindered while also allowing for higher security in the network connected to the inside interface of the firewall.

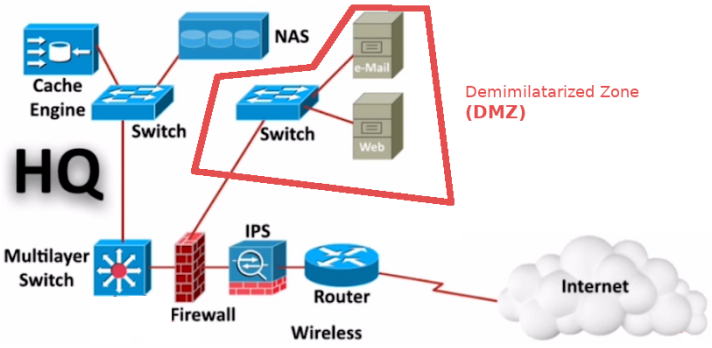


Figure 2.2: Demilitarized Zone (DMZ)

Firewall rules for the DMZ can be set up such that traffic can be allowed from devices on the internet that initiate the session, but only if the traffic is for certain ports, such as port 80 for web-servers, etc.

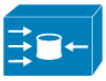
2.1.11 Firewall-Router

If the site is big like the HQ, we might need a firewall that's a separate device from the router, but if the site is small, a **firewall-router** like that in Br2 may suffice. A Firewall-Router is a router that's been configured to perform the tasks of a firewall in addition to it's job as a router. Some routers can even provide the functionality of an IPS device. These combined devices however, have to perform much more functionality of a router on the same processor as a router. Thus, the processing power becomes a bottleneck for larger networks.

2.1.12 Multilayer/Layer-3 Switch

A Multilayer switch is an Ethernet switch that can make forwarding decisions based on Layer-3 (and higher) information such as IP addresses just like a router, but can also make forwarding decisions based on Layer-2 MAC Addresses just like an ordinary switch. Thus, we can have some ports acting as switch ports while some other ports are configured to act as routing ports. These devices can give us a lot of flexibility. In the figure 2.1, the multilayer switch acts as a router with each network connected to it having a different address space (i.e., subnet).

2.1.13 Cache-Engine



The **Cache Engine** is a network appliance that locally stores content retrieved from a remote network and sends that content directly to the local devices requesting the same content, thus saving bandwidth and download (i.e., data usage) for FUPs. Typically, a user-base tends to download the same content over and over again, and the storage of these resources locally can dramatically improve the time required to access the content and bandwidth. Website assets such as graphics that are reused throughout a site are also prime candidates for caching, thus providing savings even when the requested webpage is different but in the same domain/sub-domain.

2.1.14 Network Attached Storage (NAS)

A **NAS (*Network Attached Storage*)** is a network appliance that makes storage resources, typically consisting of large, redundant hard-drives available to network clients. NAS generally has redundancy built in through the use of **RAID** arrays. They're generally more efficient than typical file servers and provide authentication mechanisms and enforce file system permissions (i.e, who can read/write those files).

2.2 Firewalls

The term *firewall* comes from the concept of an actual wall made from a non-flammable material such as a brick wall, that would be strategically constructed in a building to prevent the spread of fire from one portion of the building to another.

A **Firewall** is a network hardware (or software) that prevents the spread of malicious traffic from one part of the network (e.g., a part connected to the internet) to a secure part of the network via the implementation of sets of rules.

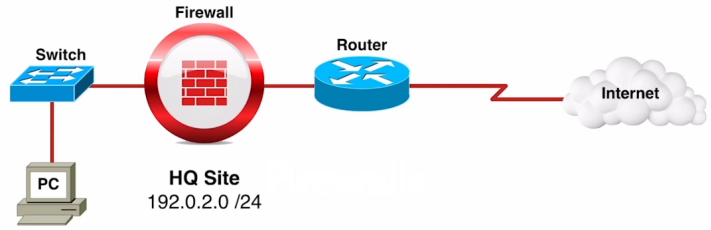


Figure 2.3: Firewall

Firewalls have evolved over the years into various types of firewalls.

2.2.1 Packet Filter

A basic firewall that can allow/deny traffic based on the source and destination IP addresses and port numbers. It is the equivalent of an *Access Control List (**ACL**)* set up on a router. The action of allowing or denying traffic is performed on the basis of a collection of rules called a rule-set. An example of a rule-set is:

Source	Destination	Action
`192.0.2.0/24`	Any	Allow
`203.0.113.0/24`	`192.0.2.0/24`	Allow
Any	Any	Deny

Firewall rule-sets are generally processed top-down, i.e., if a packet matches the a rule, then the rules below it on the rule-set are not evaluated. If the packet, however, doesn't match a rule, the next rule in the rule-set is then evaluated. Let us consider that our HQ has the network addresses `192.0.2.0/24` and a remote branch office has the addresses `203.0.113.0/24`. Then our rules in the above rule-set will allow traffic to go to any site from the HQ, allow incoming connections from the branch office, but deny everything else.

In the above rule-set, if the source IP of a packet is from the `192.0.2.0/24` network, we're going to pass it immediately, without considering any other rule. If not, we go to the next rule. If the packet is from the `203.0.113.0/24` network and has a destination IP within the `192.0.2.0/24` network, then we let it pass. Otherwise, we consider the next rule. In this case, since the packet matched no other rule, it'll match the default (**catch-all**) rule which is to deny all traffic that doesn't match any of the rules.

The problem with a packet filter is it won't let replies of our requests made to sites not allowed in the firewall to pass through. So, if we were visiting some website, we could send the request to it, but the reply from the web server would be dropped by the packet filter.

2.2.2 Stateful Firewall

A type of firewall that in addition to the duties performed by a packet filter (i.e., permit/deny packets based on IP address and port numbers) can also inspect sessions to recognize return traffic (as described in the last paragraph) and let it through if the session was initiated from a trusted network. Let us consider the rule-set below:

Source	Destination	Action
`192.0.2.0/24`	Any - TCP port 80 (`*:80`)	Allow
Any	Any	Deny

The above rule states that any traffic initiated from the HQ can go to any IP if the TCP port 80 is the destination, i.e., any website can be visited. The out-going packet can be inspected for the source and destination IP and port numbers, sequence numbers, etc. by a stateful firewall which can recognise return traffic.

So, if we went to a website hosted on `198.51.100.1:80`, the return traffic will have a source IP of 198.51.100.1 with a source port of 80, meant for our HQ. The stateful firewall can then recognize that the traffic is a part of the session initiated by someone within the HQ, and let it through. However, if traffic was initiated by someone outside the network, then the firewall's *catch-all* rule (deny: any to any) will drop the packets.

However, even the above can face issues with certain applications that may use a variety of port numbers and protocols as part of the application, which need an **Application Layer Firewall**.

2.2.3 Application Layer Firewall

An **Application Layer Firewall** is a special type of firewall that in addition to the powers of a stateful firewall (i.e., the ability to inspect sessions and allow/deny based on the source and destination IP and port numbers) also understands the nature of ab application and how the application uses the different protocols. An example rule-set for an application layer firewall:

Source	Destination	Action
`192.0.2.0/24`	Any - VoIP	Allow
Any	Any	Deny

The above rule states that the firewall should let any traffic related to VoIP initiated from inside the network should be allowed to pass, as well as any reply to the VoIP application. The related protocols may include **H.323** and **SIP (*Session Initiation Protocol*)**. So, when the Application Layer Firewall is inspecting the SIP protocol used by the VoIP application, it can understand that although the SIP protocol might be using port `5060` to initiate the session, the actual voice might be carried by the ports in the range of `16384 - 32767` (the typical range for Cisco VoIP phones). The firewall will understand that we might be going from *SIP* to **RTP(*Realtime Transport Protocol*)** to carry the voice traffic. It understands the needs of the application.

Large corporate networks need dedicated hardware firewalls, but for small sites, where a router isn't being utilized up to a high percentage, the router can be configured as a firewall in addition to a router. In case of end-user devices, the firewall contained in the OS can be used as a software firewall.

2.3 Wireless Access Points and Controllers

In the old days, before wireless networks, devices had to be physically connected to a hub/switch using a cable. This drawback has been eliminated with the use of *Wireless Access Points (**WAP**)*. Wireless devices don't even need an WAP to communicate!

2.3.1 Wireless Ad Hoc Networks

A **wireless ad hoc network** allows wireless devices to communicate with one another without the use of a network infrastructure. This solution however, doesn't scale very well, and is only good for occasional and, as the name suggests, ad hoc use. A much better permanent solution is to use Wireless Access Points (WAP).

2.3.2 Wireless Access Points (WAP)

A WAP has to be connected to a switch using a network cable, but it allows wireless devices such as smartphones and laptops to communicate wirelessly using some variance of the *Wireless LAN (**WLAN**)* standards as described in **IEEE 802.11**.

In larger enterprises, several Wireless Access Points have to be used to provide sufficient coverage for mobile devices. The challenge is then to manage all these separate APs. One method is to use Autonomous Access Points. With Autonomous APs, these devices can be managed individually and independent of each other. In this approach, we connect to one AP, configure all it's settings: the radio frequency, power level, etc. and then we move on to the next Autonomous AP and configure all the same settings on it one and so on.

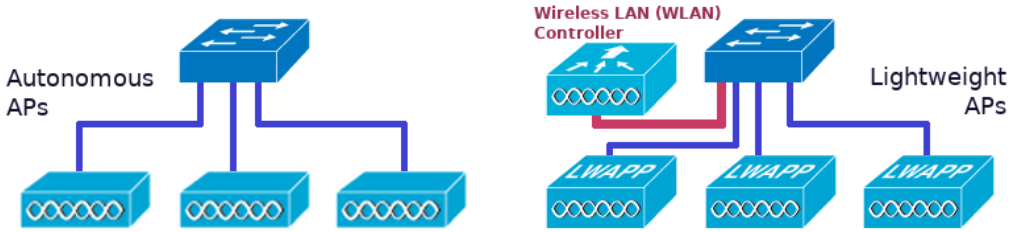


Figure 2.4: WAP Controllers

The above task is a huge administrative burden, since changing a single setting on the network would require connecting to each AP individually and changing the settings. A better approach is to use a **Wireless LAN (WLAN) Controller** with Lightweight Access Points. This method is scalable and hence used in enterprise networks. Several Lightweight APs are controlled using a WLAN controller using the **LWAPP**.

2.3.3 LightWeight Access Point Protocol (LWAPP)

The **LightWeight Access Point Protocol (LWAPP)** is the protocol used by a WLAN controller to communicate with the Lightweight APs it manages. Thus, the LWAN controller becomes a single point of administration from which all of the Lightweight APs can be administered all together. A newer protocol called the **Control And Provisioning of Lightweight Access Points (CAPLAP)** protocol, which performs a similar function, is now replacing many LWAPP deployments.

Chapter 3

Network Architecture

3.1 Star Topology

The **star topology** is probably the most common Network architecture used in Ethernet networks today. In this topology, a device is at the center, and other devices and connections radiate outwards from that centralized device. Generally we have an Ethernet Switch acting as that centralize device and several end-user devices and network resources/appliances are connected to that switch.

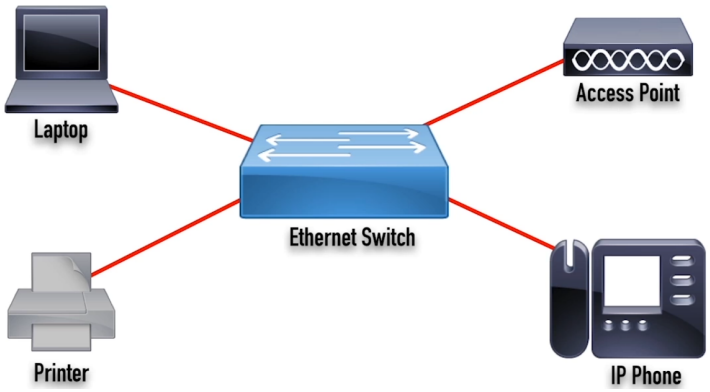


Figure 3.1: Star topology

To exchange data in this topology, the devices simply send the data on to the switch which then forwards the data to the appropriate device based on the destination MAC address in the frame. Some of the characteristics of star topology are:

- If one link fails, the others continue to function normally.
- The centralized device is still a potential single point of failure, i.e., it goes down and the entire network connected to it goes down.
- Used often in modern networks.

3.2 Mesh Topology

In mesh topology, we have multiple interconnections between the different devices and/or sites that make up our network. While multiple interconnections between Ethernet switches

isn't done typically, because Ethernet switches typically forward frames at *wire speed* and thus don't require a full-mesh. A few redundant links can still be beneficial though.

A full mesh of interconnections is much more common in Wide Area Networks (WANs) where multiple sites spread over a vast geographical distance are interlinked, with each site connected to every other site with a WAN link.

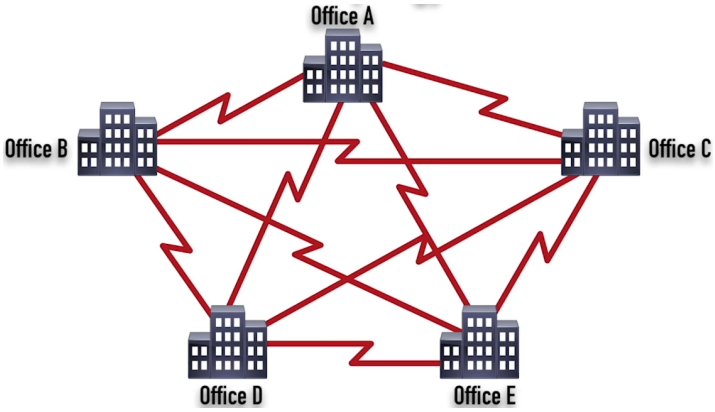


Figure 3.2: Full Mesh

However, in such a case, there's lots of redundant links - 10 links are required to connect 5 sites in a full mesh. If n be the number of sites that need to be interconnected to each other in a full mesh, the number of links required l is given by:

$$l = \frac{n \times (n-1)}{2}$$

This isn't very scalable. For 10 sites, the number of links in a full-mesh becomes $\frac{10 \times 9}{2} = \frac{90}{2} = 45$. Typically companies pay a fee per WAN link, and although such a high number of redundant links provide the ultimate reliability, it's ultimately wasteful (since it's extremely unlikely that most of the links will fail together).

Another option in this case would be to be more strategic about our connections - providing WAN links where a real need for direct connection is required, where traffic between sites is considerably large. Other than that, every office should be reachable from all other offices by relaying the traffic through one or more intermediary sites. What we end up is called a partial mesh topology.

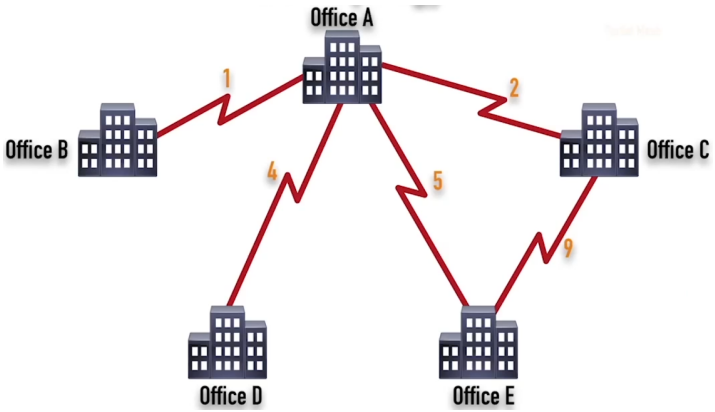


Figure 3.3: Partial Mesh

3.2.1 Comparison between Full and Partial Meshes

Full Mesh	Partial Mesh
Optimal Path	Might be suboptimal path if designed well the suboptimal path would be used less often.
Not Scalable - too many links to handle.	More Scalable.
More expensive	Less expensive - we pay only for what we need.

3.3 Collapsed Core vs Three-Tier Architectures

3.3.1 Three-Tier Architecture

There are three layers into which this architecture is divided: the **Access** layer, the **Distribution** Layer and the **Core** layer.

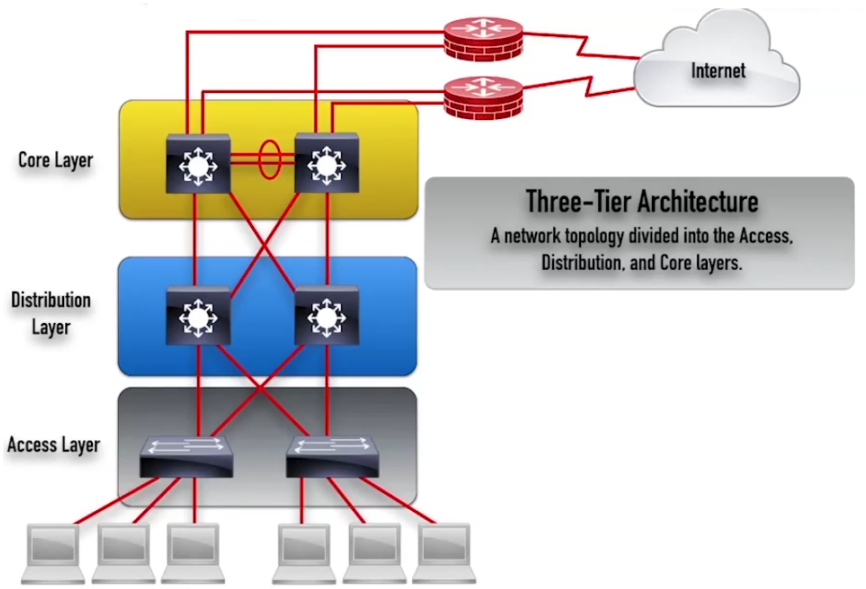


Figure 3.4: Collapsed core vs 3-Tier Architecture

Access Layer

In the example, we have Layer-2 Ethernet Switches in the Access Layer. The end-user devices like PCs, laptops, IP phones, etc. as well as network resources like printers are plugged in directly to the switches in the access layer. So, the end-user's devices are plugged into the wall Ethernet jack which have cables running till they reach the switches in the wiring closets at this layer.

At this point, each switch has their own connected networks, but they're not reachable from the other switch(es). We need to connect these switches such that each device on one switch is accessible from the others. If we have a file server connected to one switch, the devices on the other switch should be able to access it as well. One possible solution would be a full-mesh topology, but that would be impractical due to the huge number of switches in an office building. An alternative is this 3-tier architecture.

Distribution Layer

This layer has a bunch of Layer-3 switches, i.e., switches that are capable of routing (i.e., forwarding packets on the basis of IP addresses). The access layer switches are connected to these layer-3 switches. Some *literature* call this layer the **building distribution layer** because this layer provides interconnections between the switches (in the access layer) of a building or several buildings in close proximity. The layer-3 switches on the distribution layer are connected up at the core layer.

Core Layer

The job of the core layer is to get the traffic as quickly as possible from one distribution layer switch to another distribution layer switch. The core layer also connects our network to the rest of the internet (or some other remote site). In this case, the outgoing connections from the core layer conntect to a router-firewall each which then lead (perhaps via different ISPs) to the internet.

The core layer has a multiple links from the layer-3 switches on this layer surrounded by an oval. This is called an **Ether-channel**. An Ether-channel is a logical interface that is a created by aggregating multiple physical interfaces, allowing higher throughput between devices as compared to a single link. If there are two interconnected links, each of 10Gbps, in the core layer between the two layer-3 switches in the example - then logically, the aggregate of these two physical links will be a logical link of 20 Gbps, which would allow very quick data transfer and exchange between the two layer-3 switches at the core layer.

In this architecture, at the access layer, we've got star topologies connecting to the end-user devices. On the distribution layer and core layer, we've got some redundant connections that make (partial) mesh topologies. This makes this particular type of architecture a **hybrid topology**, where the topology contains elements of multiple topology types.

Some networks, however, are just not large enough to justify the high investment in so many multi-layer switches. In such networks, we might not need a distinct core and distribution layer, which leads us to a collapsed-core architecture.

3.3.2 Collapsed Core Architecture

In this architecture, the Distribution layer and the Core layer are merged to form the **Collapsed Core layer**. There are still redundant connections between the access layer switches, but an entire layer's been eliminated!

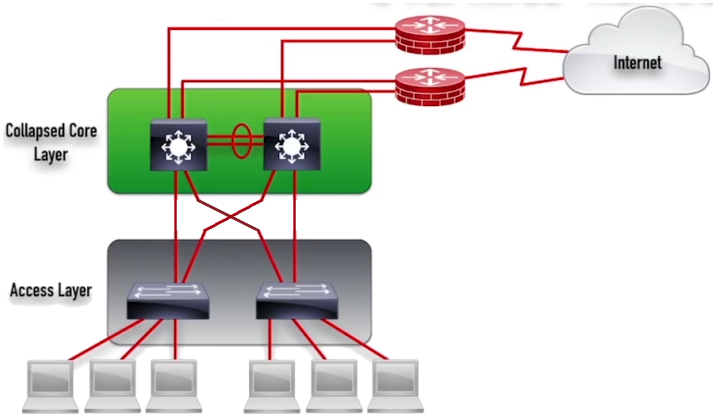


Figure 3.5: Collapsed Core Architecture

This architecture might not work well for a large campus with lots of buildings to interconnect, but works well for a smaller network within just a building or two. This is a two-tier architecture with a consolidated core and distribution layer.

Chapter 4

Network Cabling

We can connect our devices using copper or fibre cables, which would require copper and fibre connectors respectively, or connect them through wireless means. We're now going to take a look at the various kinds of network cables, i.e., copper and fibre wired mediums for connecting devices.

4.1 Copper Cables

When physical media are used to interconnect devices in a network, the cable used is most often either copper cable or fibre optic cables.

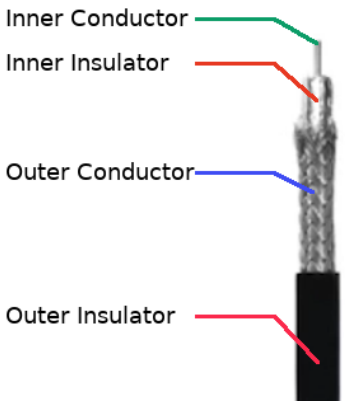


Figure 4.1: Coaxial Cable

The figure 4.3 to the left shows a coaxial cable that has an inner conductor, an insulator wrapped around it, another conductor that's generally either a wire mesh or a foil wrapped around the inner insulator and an outer conductor. The outer conductor provides great electrical characteristics and protection from **ElectroMagnetic Interference (*EMI*)** due to the fact that it's a mesh wrapped around the core containing the inner conductor that's primarily responsible for carrying the signal.

EMI is a phenomenon which occurs when radio waves are picked up by our cable or radiated by a cable carrying another signal, thus causing a degradation in the original signal carried by our cable. EMI can be caused by communication waves of high intensity (due to communication towers nearby), electrical spikes due to large machinery, etc. The insulation and shielding in a coaxial cable also helps prevent the cable from acting like an antenna by emitting frequencies that could interfere with the operation of other devices nearby.

4.1.1 Impedance of a Coaxial Cable

The impedance of a coaxial cable is an electrical property of that cable. **Impedance** is a circuit's opposition to current flow (measured in Ohms) which can have resistive, capacitative and/or inductive components. This impedance of the cable needs to match up to the impedance of the device to which it's connected.

4.1.2 Coaxial Cable Standards

RG-59

RG-59 is an older type of coaxial cable that is used to carry video over short distances, typically with an impedance of 75Ω. It can be used to carry video from an old VCR to the TV. This cable is extremely lossy over long distances, and is thus not used that much these days.

RG-6

The **RG-6** coaxial cable that's used by cable TV operators to transport video from their antenna/receiver on the client premises to the client's TV, or just connect the customer devices to their network directly. This cable too has an impedance of 75Ω. Overall, the RG-6 cable is replacing the RG-59 cable throughout the industry.

RG-8

To carry data over a coaxial cable, the **RG-58** cable is used. This cable has an impedance of 50Ω and is hence used to carry data in **10BASE-2** Ethernet networks. Most commercial network devices today, however, use twisted-pair cables over coaxial cables for carrying data.

4.1.3 Twisted Pair Cables

A **twisted pair** cable has 8 separate wires, each colour coded and presented in pairs. There's a blue wire paired and twisted with a blue & white wire, an orange wire twisted and paired with a orange & white wire, and similar pairs in green and brown colours.

Figure 4.2: Twisted-Pair Cable

In the twisted pair cable (figure on the left), the number of twists per unit length vary a little among the different colours to provide electromagnetic shielding from the other twisted pairs of other colours in the cable. Further, the twists are *tight* enough to ensure that these cables don't act like antennas and start receiving/transmitting signals and degrade the quality of the signal they're carrying. The cable on the left is called an **Unshielded Twisted Pair (*UTP*)** and is the most common twisted pair cable in the industry.

There is also a variance of the twisted pair cable called the **Shielded Twisted Pair (*STP*)** which has something like an aluminium wrapping around each pair of wires. This act of electrically insulating each pair of wires makes this cable much more resistant to EMI and making it possible to support higher data rates through the cable, but it also makes the cable costlier. For most applications, UTP suffices, thus not requiring the use of STP. In the case of UTP, the data throughput can be increased by increasing the number of twists.

4.1.4 Categories of Twisted-Pair cables

Category 3 (Cat-3)

The **Cat-3** cable was used in older Ethernet **10BASE-T** networks, where the maximum rate of data transfer was *10 Mbps*. This cable was common in buildings wired for a telephone

system, such as a **Private Branch Exchange (*PBX*)** within an office building. When we need speeds higher than *10 Mbps*, we should use *Cat-5* cable instead.

Category 5 (Cat-5)

Cat-5 cable is commonly used in Ethernet **100BASE-TX** networks providing data speeds of up to *100 Mbps* and typically uses 24 gauge wire.

Category 5e (Cat-5e)

The **Cat-5e** cable is an updated version of the *Cat-5* cable, sometimes used for Ethernet **1000BASE-T** networks, supporting data speeds up to *1 Gbps*, and offers reduced crosstalk (i.e., more protection against EMI) when compared to Cat-5 cables.

Category 6 (Cat-6)

Similar to Cat-5e cables, the **Cat-6** cables are also commonly used for Ethernet **1000BASE-T** Gigabit connections which provide data rates of up to *1000 Mbps* or *1 Gbps*. However, Cat-6 cable is generally made from thicker wires - instead of using 24 gauge wires, they use 23-gauge or 22-gauge wires.

Category 6a (Cat-6a)

A **Cat-6a** cable has tighter twists than Cat-6, allowing it to carry twice as many frequencies as Cat-6. Thus, the data throughput increases so much that it can be used in 10GBASE-T Networks, supporting speeds up to *10 Gbps*.

4.2 Fibre Cables

An alternative to copper cables in our networks is **fibre optic** cabling, in which light is used to transmit data instead of electricity. Since we're using light to transmit the binary 1s and 0s, the data in the wire is immune to EMI. Depending on the type of fibre, the light source, etc., data rates much higher than copper cables are possible.

A laser or an LED is used to inject light into the cable. Lasers tend to be more powerful and hence cover more distance. Broadly speaking, there are two categories of fibre optic cables.

4.2.1 Anatomy of a Fibre Optic cable

A fibre optic cable is constructed as shown in the following figure:

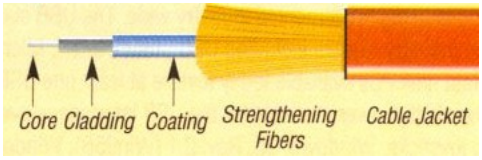


Figure 4.3: Fibre Optic Cable Construction

The **core** of the fibre optic cable is made of glass, with the surrounding **cladding** also made of glass, but of a different *refractive index* to that of the core, due to the injection of **dopants**. *Dopants* are impurities put into the glass specifically with the purpose of changing the refractive index. The core and the cladding are the only two components of the cable upon which the data transfer

speed and characteristics depend. The cladding is then covered by a buffer/coating, which is a hard plastic layer designed to prevent light leakage from the cladding due to scratches.

The buffer is then covered with strengthening fibres (generally *armid* fibres) which provide flexibility to the cable. All of these inner layers are then finally sheathed within a plastic jacket.

4.2.2 Total Internal Refraction in Fibre Optic cable

Refraction of light is the phenomenon of bending of light as it passes from a medium to another medium with a different *refractive index*. However, if the *incident* angle of light, i.e., the angle at which the light meets the boundary between the mediums, is beyond an angle called **critical angle**, the light is reflected by the boundary instead of being allowed to pass into the other medium.

This principle is used by fibre optic cables to *bend* light and have light travel through a cable. The cables are made so that the core has a higher refractive index than the cladding, and engineered to ensure that no matter what angle the cable is bent at, the cladding will always reflects the light. The *mode* of the fibre optic cable is dependent upon the relative thickness of the core and the cladding.

4.2.3 Single Mode Fibre (SMF)

In a **Single Mode Fibre**, the diameter of the core is so small that there's really just one path a ray of light can travel in - horizontal.

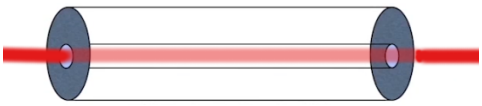


Figure 4.4: Single Mode Fibre Optic Cable

Practically, the angle of incidence remains near parallel to the core itself, which means all light travelling through the fibre take identical paths and take the same amount of time to arrive at the other end of the fibre. The **mode** is the path taken by light while travelling through a fibre optic cable. A single mode fibre optic cable's core is so thin that it only allows light to enter at a single angle (or a very small range of angles), thus allowing only a single mode of propagation of light in the fibre.

4.2.4 Multi Mode Fibre (MMF)

In the case of the **Multi Mode Fibre**, the diameter of the core is sufficiently large that there are multiple possible angle of incidences at which light can enter the fibre.

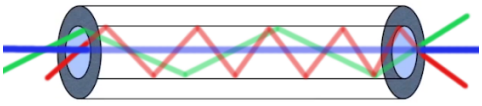


Figure 4.5: Multi Mode Fibre Optic Cable

So, there are multiple paths that the light can take, i.e., there's multiple modes of propagation of light in the fibre. This means that even if two different rays of light enter the fibre at the same time, one might be reflected (i.e., bounce around) more times than the other - which means it'll have to travel a greater distance to reach the other end. Since the speed of light in the medium is constant, this means that one ray will take a longer time to arrive at the other end than the other. Consequently, **Multi-mode Delay Distortion** may occur (if the distance is large enough). However, multi mode fibres are also cheaper to manufacture.

4.2.5 Multimode Delay Distortion

As explained in sub-section 4.2.4, in multi mode fibres, the difference in the angle of incidence may cause a substantial delay in travel time for the light waves over the same distance. This may cause the bits, represented by the light rays, to arrive out of order! This is why over longer distances, the use of single mode fibre becomes necessary. Often the distance limitation of multi mode fibre is going to be 2kms.

4.3 Copper Connectors

4.3.1 Common Connectors for Coaxial Cables

F-Connector

Figure 4.6: F-Connector

The **F-Connector** is a coaxial cable connector often used in cable TV and cable modem connections. This is used when a coaxial cable is used to transmit video. For connecting network devices that deal with data, however, we're more likely to use the **BNC** connector.

BNC (Bayonet Neill-Concelman or British Naval Connector)

Figure 4.7: BNC

The **BNC** is a coaxial connector which was often used for data connections involving coaxial cables such as those in 10BASE-2 Ethernet networks.

The connector needs to be pushed and then twisted, and an in-built spring mechanism secured the connector in place. Now-a-days twisted-pair and its associated connectors are much more common than coaxial cables and its connectors.

4.3.2 Common Connectors for Twisted-Pair Cables

DB-9 Connector

Figure 4.8: DB-9

The **DB-9** connector is a 9 pin connector used for asynchronous serial communications (or simply, serial connection) with devices.

DB-9 was (and in some cases, still is) a popular method of connecting to serial devices such as that of a console port from a router. Since most laptops today don't have a DB-9 port, we might need a *DB-9 to USB* adapter to be able to communicate with such devices.

RJ-11 (Type 11 Registered Jack)

Figure 4.9: RJ-11

The **RJ-11** connector is a 6 pin connector (which only uses 2 or 4 pins) that were commonly used with analogue telephones.

While it looks similar to an RJ-45 connector, the number of notches and the number of wires that can be *punched down*,

i.e., attached to these connectors is different. The RJ-45 connector is wider as compared to RJ-11.

RJ-45 (Type 45 Registered Jack)

Figure 4.10: RJ-45

The **RJ-45** connector is a 8 pin connector commonly found at the ends of Twisted-pair Ethernet cables such as Cat-5 or Cat-6 cables.

The RJ-45 connector is the most commonly used connector for Ethernet connections since it's compatible with Cat 5, Cat 6, Cat 6e cables - the most commonly used twisted pair cables for networking devices used today.

4.4 Fibre Connectors

ST (Straight Tip)

Figure 4.11: ST Connector

The **ST (*Straight Tip*)** connector, also sometimes referred to as the *bayonet* connector, is commonly used with *Multi Mode Fibre (MMF)* cables.

We use this connector by pressing down and twisting and the in-built spring mechanism will secure the connector in place with the port of the device that accepts ST connections, or the fibre patch panel.

LC (Lucent Connector)

Figure 4.12: LC Connector

A **LC (Lucent Connector)** connects by being pushed into the terminating device where the clip at the top of the connector holds it in place. It needs to be disconnected by pressing the tab on the top of the connector, followed by pulling out of the connecting device, similar to the RJ-45/RJ-11 connectors.

SC (Subscriber/Standard/Square Connector)

Figure 4.13: SC Connector

The SC connector is a simple connector that is connected by being pushed into the device or fibre patch panel and is disconnected simply by pulling out of said terminating device.

MTRJ (Media Termination Recommended Jack)

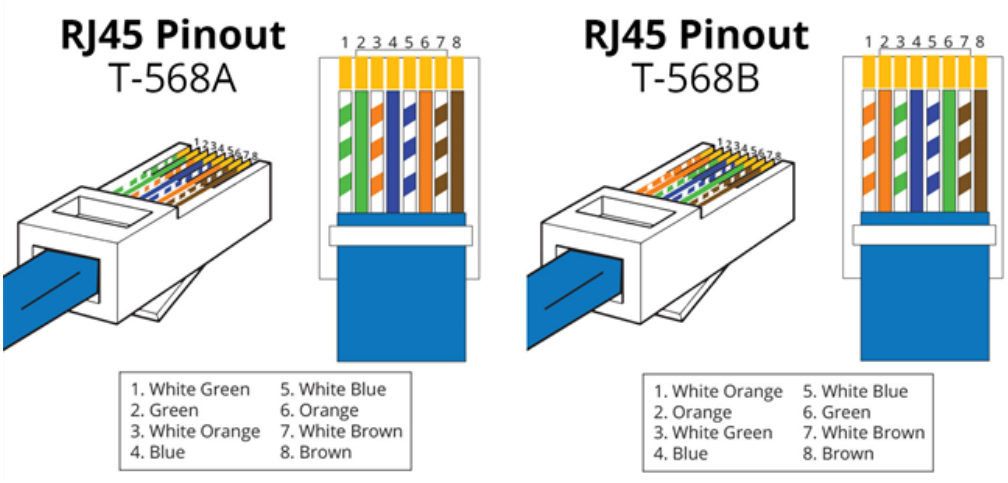
Figure 4.14: MTRJ

Unlike the other connectors discussed here, the MTRJ has two fibre strands in a single connector. These two strands are the transmit and the receive strands. This allows us to have more connections in a small space, and is extremely useful for fibre patch panels.

The connector needs to be pushed in to the device to connect and pulled out to disconnect, just like the *SC* connector.

4.5 ETA/TIA 568 Standards

In an **Unshielded Twisted Pair (*UTP*)** cable, we've got colour coded insulators around the 8 copper cables. The **Electronic Industries Alliance (*EIA*)** and the **Telecommunications Industry Association (*TIA*)** have a set of joint standards called the **EIA/TIA 568 Standards**, describing how wiring should be done in office buildings. In this, the **T568A** and **T568B** standards describe which colour of insulator should be attached to which pin in an RJ-45 jack.



Most often, we use the T568B colour scheme for wiring our RJ-45 connectors. Older installations however, might have been wired according to the T568A standard.

Chapter 5

Basic Troubleshooting

5.1 Troubleshooting Fundamentals

Troubleshooting is the identification and resolution of a problem. There can be several approaches to this, focused on several aspects. Some troubleshooting models are based on detecting the problem in the layers of the OSI stack.

5.1.1 Top-down Troubleshooting

This is an approach to troubleshooting based on the OSI model. With **top-down troubleshooting**, we start at the application layer of the OSI stack and work our way down to the physical layer.

For example, if someone is unable to access a web server, we first confirm the problem by trying it for ourselves. If the problem can be reproduced, then we try accessing other sites and find out how widespread the problem is. If we're unable to access any site on the internet, we next open up a command prompt on their machine, and telnet to port 80 on the IP address of a website. If we can connect, it'll give us *some* assurance that things are okay in layers 1-4.

Then we can figure out what the problem is in the upper 3 layers - perhaps a misconfiguration on the browser like an invalid proxy server settings. If we're unable to telnet to port 80 of the website, then we have to focus on the lower levels.

5.1.2 Bottom-up Troubleshooting

In this approach, we start from the bottom (physical) layer of the OSI stack, and work our way up. We start with checking the cabling - are the devices plugged in and the cables connected? If they are, then we can move up to the data-link layer, where we ensure that our switches have *learned* the appropriate MAC addresses. If they have, we move up to the Network layer, and ensure that the correct routes are configured on the routers, they know the correct routes to connect to the destination network.

5.1.3 Divide and Conquer Troubleshooting

In this approach, we start neither at the top, nor the bottom, but start from the middle by separating the three lower layers (physical, data-link and network) from the upper four layers (transport, session, presentation and application). We do this by first performing a **ping**. This tells us if the destination network is reachable using *ICMP echo request* packets. If we get replies for the ping, we're sure that the IP address is reachable, indicating a problem in the upper four layers. If not, the problem might be in the bottom 3 layers (or the server may have disabled replies to ICMP echo requests).

5.1.4 Other approaches

After we've pinged a server, if there's no reply, we might also try to **follow the path**, where we successively test reachability (for example, with the `traceroute` command), and find the device that's causing the problem, and then edit its configuration till the problem is resolved.

In case we have a copy of an earlier, working configuration, we could try to compare that to our present configuration, spot the differences and check which one of them could be causing the problem.

5.1.5 Swapping Components

This approach gets rid of problems in the physical layer, since we're replacing hardware components to get rid of any defective equipment in the network. This can be a port on a switch, a fibre jumper, a Cat 6 cable to see if an observed problem follows one of the components.

5.2 Cisco's Structured Troubleshooting Model

Since troubleshooting is an integral part of a Network professional's work, Cisco provides a **7 step troubleshooting methodology** since having a plan of attack while dealing with a problem will make us much more confident and attentive than trying things randomly.

5.2.1 Define the problem

The goal of this step is to *clearly* articulate/define the problem. So the problem isn't that the *internet is broken*, but *this particular PC can't get to this particular website*.

5.2.2 Collect Information about the problem

Now that we know what is wrong, we need to figure out all the parameters of the problem for the next steps. We need to understand the problem at this stage. Thus, we may need to interview the end-user who reported the problem. It might also involve running **show** or **debug** commands on the networking devices.

5.2.3 Analyse the Information

We now go through the data obtained in the last step, and we might need to research some of the errors/warnings/messages that the system generates, to better understand the problem.

5.2.4 Eliminate Potential Causes

This step involves removing things that don't make logical sense from the list of potential problems. For example, if a network suffers from poor performance *all the time*, then we can eliminate network load as a suspect because the problem occurs even during low network usage conditions.

5.2.5 Propose a Hypothesis

Once a few suspects have been eliminated, a clear picture about the issue can start to form in our minds, and eventually, we can present a hypothesis, i.e., *I think this is most likely what's going on*.

5.2.6 Test the Hypothesis

Now, we can test our hypothesis while **balancing** the urgency of fixing the issue right then, with the impact that our solution will have on the rest of the network. For example, if the solution is to reboot a router or a switch, we have to ask ourselves if the problem is critical enough at this point to reboot right now and disrupt the work of everyone else using the router/switch, or if it could wait for a scheduled maintenance window.

5.2.7 Solve and Document

If our hypothesis turns out to be true, we can finally implement the solution and then **document** the solution so that it can help us if we run into a similar situation ever again, or help our co-workers if they're facing the issue in the future.

If the hypothesis turns out to be wrong, we can then gather additional information if required and propose a new hypothesis, and repeat - till the issue is fixed.

Chapter 6

IPv4 Addressing

Even though the global pool of IPv4 addresses has been exhausted, and IPv6 is the future, we still have a lot of IPv4 addressing in networking today. Thus, we need the following basic skills:

- Calculating the number of subnets for a subnet mask.
- Calculating the number of hosts for a subnet mask.
- Calculating the usable address range for a specific subnet.
- Finding the Broadcast ID, Host IDs and Network ID in a network, etc.

6.1 Binary Numbering

An example of an IPv4 Address (or simply, an IP address) is **10.1.2.3**. Before we can start working with IP addresses and use them to design networks, and calculate the specifics of a network, we need to be able to convert an IP address into its binary equivalent. An IPv4 address is 32-bits, and we use the **Dotted Decimal notation** to represent it, dividing the 32 bits into four sets of 8 bits each, called an **octet** and then writing the decimal equivalent of each octet, separated by dots, to represent the IP address.

Dotted Decimal Notation	10	1	2	3
Binary Equivalent	1010	1	10	11
8-bit Binary (Octet)	00001010	00000001	00000010	00000011
Octet Number	1^{st} Octet	2^{nd} Octet	3^{rd} Octet	4^{th} Octet

6.1.1 Binary to Decimal Conversion

Since binary numbers are **base-2** like decimal numbers are **base-10**, we need a binary conversion table such as the one shown below to convert a binary number to it's decimal equivalent.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**

Table 6.1: Binary Conversion Table

The above table simplifies the formula:

$$\text{Decimal}D = \sum_{i=n}^{0} 2^n b_i \tag{6.1}$$

where b_i is the binary bit in the i^{th} position of the binary number B, n is the number of digits in B and D is the decimal representation of B.

Converting 10010110 to decimal

First we construct the binary conversion table and write the binary digits beneath the powers of 2.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	0	0	1	0	1	1	0

Table 6.2: Binary Conversion Table

Now, from the above table, we simply add the powers of 2 (column headings) that have a 1 below them, and ignore those that have a zero in the binary representation to get the decimal equivalent.

$$\begin{aligned} d &= 128 + 16 + 4 + 2 \\ &= 150 \end{aligned}$$

So, the binary number $(10010110)_2$ in decimal is $(150)_{10}$. This can be directly calculated with the equation 6.1:

$$\begin{aligned} d &= \sum_{i=7}^{0} 2^i b_i \\ &= (2^7 \times 1) + (2^6 \times 0) + (2^5 \times 0) + (2^4 \times 1) + (2^3 \times 0) + (2^2 \times 1) + (2^1 \times 1) + (2^0 \times 0) \\ &= 128 + 16 + 4 + 2 \\ &= 150 \end{aligned}$$

6.1.2 Decimal to Binary Conversion

Decimal numbers can also be converted to Binary by using the binary conversion table. In this method, for each column in the binary conversion table, we ask if the number is greater than or equal to the value in the column. If yes, then we put a **1** in the cell below it and subtract the value of the column from the number, and then repeat the same process with the new value.

Note that if at any point during the calculation, the remainder turns **0**, then all the columns to the right will also be **0**, since the number has been broken down completely into factors of 2.

Converting 167 to Binary

First we draw the binary conversion table. The first column is the value 128. Since $167 > 128$, we write **1** in the cell below 128, and then subtract $167 - 128 = 39$. This is the new value we'll compare against.

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Quotient	1	0	1	0	0	1	1	1
Remainder	39	39	7	7	7	3	1	0

Table 6.3: Binary Conversion Table

Since $32 < 64$, we just write a **0** in the cell below 64 and move on to the next column. This time, $39 > 32$, so we write a **1** below 32 and the new value to evaluate against is $39-32 = 7$. Again, $32 < 8 < 16$, so the columns of 16 and 8 both get a **0**. Since $7 > 4$, it gets a **1**, new value is $7-4 = 3$. This value, $3 > 2$, so the cell below 2 gets a value of **1** and the remainder is $3 - 2 = 1$. Finally, since 1 is the remainder from the last step, the cell for $1 == 1$ and hence, the last cell gets a 1 as well. Then, we have the equivalence $(167)_{10} = (10100111)_2$.

6.1.3 Exercises

i) Convert 01101011 to decimal

Solution - First we draw the binary conversion table. Then, we add all the cells that have a **1** below them and ignore the ones associated with a *0*.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
0	1	1	0	1	0	1	1

Table 6.4: Binary Conversion Table

Thus, we have the decimal value:

$$\begin{aligned} d &= 64 + 32 + 8 + 2 + 1 \\ &= 107 \end{aligned}$$

So, we have the equivalence: $(1101011)_2 = (107)_{10}$ (**Ans**).

ii) Convert 49 to binary

Solution - We have the table:

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Quotient	0	0	1	1	0	0	0	1
Remainder	49	49	17	1	1	1	1	0

Table 6.5: Decimal 49 to Binary

So, we find that $(49)_{10} = (110001)_2$. (**Ans**).

6.2 IPv4 Address Formatting

An IPv4 address can be divided into two parts: the Host ID and the Network ID. The Host ID is the address of the device (PC, laptop, smartphone, printer, etc.) on the network. These

are represented by the **host bits** of the IP address. The remaining bits for the Network ID and represent the network itself. Somewhere among those 32 bits, is a division that separates the Network ID from the Host ID, dictated by the **subnet mask**.

Since the number of bits in the IPv4 Address is constant, i.e., 32 - if we need to represent more devices in a network, we need more bits, which leaves less room for network address bits. If however, we need to represent a large number of network with few devices in each network (called **subnets**), then we need a larger number of bits for network ID and fewer in the host bits. The larger the size of the network (i.e., the number of devices per network) the fewer networks (i.e., subnets) there can be, and vice versa.

6.2.1 Subnet Mask

A **subnet mask** is used to figure out which bits in the IP address are parts of the network ID and which bits are the host ID. Thus, a subnet mask tells us about the size of each subnet (*based on the number of bits in the host id*) as well as the total number of subnets that there can be (*based on the number of bits in the network ID*).

An IP address is much like an address in the real world, where neighbours living on the same street share parts of the address (street name, city, country, etc.) but have different house numbers. So, the Network ID is like the street name, i.e., shared by all the hosts in the network, but the Host ID is like the house number, unique to every host. All of the devices in a network share a common network address space.

A subnet mask is a set of contiguous **1**s followed by a contiguous set of **0**s (i.e., a subnet mask always has a series of 1s followed by a series of 0s, and never mixes them).

Dotted Decimal Notation	10	1	2	3
Subnet Mask in DDN	255	0	0	0
IP in Binary	00001010	00000001	00000010	00000011
Subnet Mask	11111111	00000000	00000000	00000000
	Network Bits	|<——— *Host Bits* ———>|		
Network ID in Binary ($\wedge$)	00001010	00000000	00000000	00000000
Network ID in DDN	10	0	0	0
Host ID in DDN		1	2	3

6.2.2 Prefix/Slash Notation of Subnet Mask

Instead of writing the subnet mask in DDN, we can also just say the number of bits in the Network ID by writing the IP address as **(IP address)/(number of bits in Network ID)**. So, the above case, where the IP address is `10.1.2.3` and the number of bits in the network ID is `8`, the resultant **prefix notation** would be `10.1.2.3/8`.

6.2.3 Dotted Decimal Notation of Subnet Mask

We also have the option of writing the subnet mask in DDN, just like an IP address. Since the subnet mask consists of the first 8 bits, i.e., the first octet, the value of the subnet mask is **255** for the first octet (i.e., the maximum possible value of an octet) followed by 0s. Thus, the subnet mask in DDN is: `255.0.0.0`.

Terms	Description
IP address with no subnet information:	`10.1.2.3`
IP Address in Prefix Notation	`10.1.2.3/8`
IP Address in Dotted Decimal Notation	`10.1.2.3 255.0.0.0`

Table 6.6: Types of Subnet Mask representations

.

6.2.4 Calculating Network and Host ID

The network ID of a particular network is obtained by taking the original IP address and changing all the host bits to **0**s. Thus, the network address/ID for the given IP address is `10.0.0.0/8` in prefix notation, or `10.0.0.0 255.0.0.0` in DDN.

6.3 Address Classes

When we look at an IP address, we don't know which of the 32-bits represent the Network address and which represent the host ID unless we're provided with the Subnet Mask. While we can set the subnet mask ourselves, there are defaults, on the basis of which the entire IPv4 Address space (`1.0.0.0 - 255.255.255.254`) can be divided into 5 classes: Classes A, B, C, D & E.

Address Class	Value in First Octet	Classful Mask (DDN)	Classful Mask (Prefix)
A	1 - 126	255.0.0.0	/8
B	128 - 191	255.255.0.0	/16
C	192 - 223	255.255.255.0	/24
D	224 - 239	N/A	N/A
E	240 - 255	N/A	N/A

Table 6.7: Classes of IPv4 Addresses

.

Among these, only classes A, B & C are ever assigned to a normal host. Address **class D** is used for **Multicasts**, but even in that we send data to a multicast address, but no host is ever assigned an address in this range. It's a destination only address, and thus doesn't even have a default subnet mask. A **Class E** address is used for *experimental purposes and R&D*.

The **Classful Mask**, also called the *Natural Mask* is the default subnet mask for a specific address class. So, if an address such as `10.1.2.3` uses a *classful mask*, we know it's a Class A address (since first octet is in the range *1 - 126*) which has a default mask of `255.0.0.0` and so - `10.0.0.0` is the network ID and `.1.2.3` is the host ID.

The class of an IP address is entirely dependent upon the first octet. It doesn't matter what the subnet mask is, only the first octet determines the class of an IP address.

6.3.1 Loopback Addresses

The `127.0.0.0` network is entirely missing from the IP class table, because the entire network is used for a special purpose. It's called the **Loopback address**, and it's used to test the functionality of the TCP/IP stack on a host machine, by pinging **127.0.0.1**. Thus, it's also called the local address of a device.

6.3.2 Assigning IP Addresses

Publicly routable IP addresses are assigned by **ICANN (*Internet Corporation for Assigned Names and Numbers*)**. It's a non-profit organization that *gives* a block of numbers for registration to an *international registry*, such as **ARIN (*American Registry for Internet Numbers*)** in North America and **IANA (*Internet Assigned Numbers Authority*)** outside North America, operated by ICANN.

6.4 Private vs. Public IPv4 Addresses

There can be a total of $2^{32} = 4,294,967,296$ IPv4 Addresses, and the total number of internet connected devices has already exceeded that. Thus, there are not enough IPv4 addresses to identify each device uniquely, a problem which was solved by the use of IPv6. However, since there's a sizeable IPv4 deployment in the industry, a lot of companies use a **Private IP address range** for their hundreds/thousands of devices that are used internally.

These addresses are also called **RFC 1918** addresses. These addresses aren't publicly routable, i.e., no host on the internet should have one of these as their IP address. These exist solely to provide routing within a company/organization. These private IP addresses need to be translated to a public IP addresses when communicating with the internet (so that other computers can send data to them), and multiple private IP addresses can have the same public IP address.

For example, using a wireless router at home, one can have multiple devices connected to the internet at the same time, using the same external IP address. This is possible because of **NAT *(Network Address Translation)***, a way to convert private IP addresses to publicly routable IP addresses. The private IP address space as defined in RFC 1918 are:

Class	Addr Block	Address Range	Default Subnet Mask
A	10.0.0.0/8	10.0.0.0 - 10.255.255.255	255.0.0.0 (/8)
B	172.16.0.0/12	172.16.0.0 - 172.31.255.255	255.255.0.0 (/16)
C	192.168.0.0/16	192.168.0.0 - 192.168.255.255	255.255.255.0 (/24)

Table 6.8: Private IPv4 Address Ranges

.

6.4.1 Automatic Private IP Addressing (APIPA)

In addition to the RFC 1918 private IP ranges, there is another private IP address range, but one that's not routable - even by any router that we own, because this IP address range is self-assigned. Generally, we don't want a device in our network to have an IP within this range. When a network connected device isn't assigned an IP address, either manually or automatically from a **DHCP (*Dynamic Host Configuration Protocol*)** server, it self-assigns an IP address to itself from this range. Such an address is called an **APIPA (*Automatic Private IP Addressing*)** address.

Class	Addr Block	Address Range	Default Subnet Mask
B	169.254.0.0/16	169.254.0.0 - 169.254.255.255	255.255.0.0 (/16)

Table 6.9: APIPA Address Range

.

A host having an APIPA address is an indication that something on the network has failed and this host needs attention because it's not connected to the rest of the network. Another

case when APIPA is used is when two or more hosts have to communicate within themselves but there's no IP assignment, such as the case where two hosts are connected by a network cable but no further configuration.

6.5 Unicast

Unicast, Broadcast and Multicast are some of the different traffic flows in IPv4 networks. **Unicast** is a *one-to-one* communication flow, i.e., one device on the network is talking to another device on the network. For example, if a host sends some query to a server, and the server responds with the results only to that host, it's unicast.

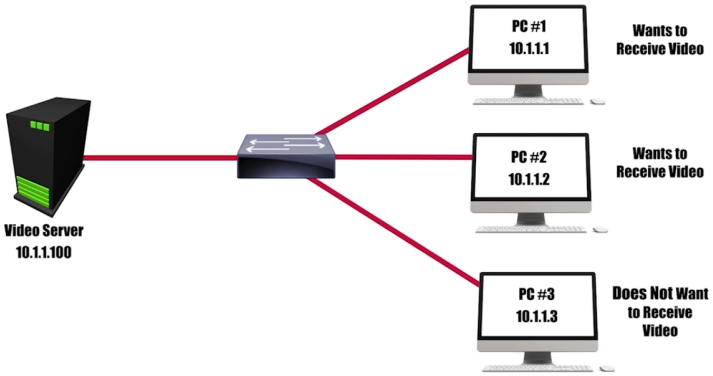


Figure 6.1: Unicast

Let us consider the above situation where the hosts with host IDs `.1` and `.2` want to receive a video, but the host `.3` doesn't. Then the video server has to send two packets - one two `10.1.1.1` and another to `10.1.1.2` when using an unicast model. While this is good for host `10.01.1.3`, since it didn't have to see unwanted traffic, the video server at `10.1.1.100` has an increased load. It has to send multiple copies of the same data to multiple hosts individually. If there had been 200 hosts that wanted the video, then it'd have to send 200 individual copies, causing more processor load for the video server and bandwidth demand on the link connecting to the video server.

6.6 Broadcast

A **broadcast** is a *one-to-all* communication flow. Thus, the communication occurs from one source to all the devices in a subnet.

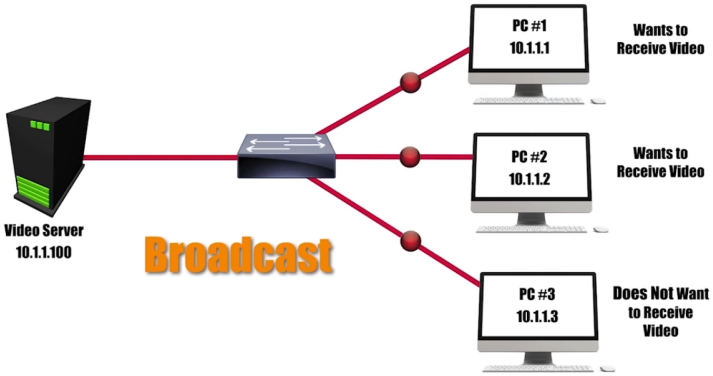


Figure 6.2: Broadcast

In the above case, the video server sends the data to all the connected hosts, including PC #3, i.e., `10.1.1.3`. So, `.3`'s NIC had to take some time to analyse the packet and then discard it, since it didn't request for that data. This will be done by PC #3 for every single packet in the video stream that's being broadcast.

This solved the problem of processor load of the video server, since it only has to broadcast one packet and bandwidth usage on the link connecting the video server, since only one packet flows through it. However, now PC #3 has to keep rejecting the packets of the video.

IPv4 utilizes broadcast a lot. The early versions of the **Routing Information Protocol (RIP)** used to advertise network information through broadcasts. Broadcasts can also be used to advertise services such as printer services on a local subnet. However, IPv6 doesn't use a broadcast at all!

6.7 IPv4 Multicast

Multicast is a *one-to-many* communication flow. In multicast, the devices that want to receive the data join a **multicast group**. The multicast group has an IP address in the **Class D** range, i.e., `224.0.0.1 - 239.255.255.254`. If the multicast group address is, say, `239.1.1.1`, then it represents every host within that group.

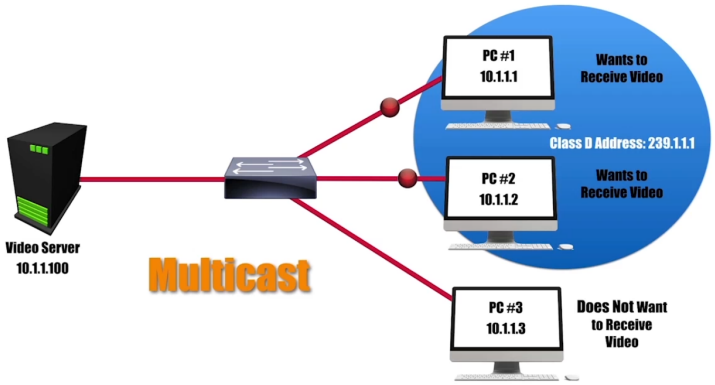


Figure 6.3: Multicast

So now, the video server can send out one packet with the destination address of `239.1.1.1` and it'll reach all the hosts within that group who want to receive the video, while those that don't want to, don't get any packets at all!

Multicasts are used quite a bit in both IPv4 as well as IPv6. In IPv4, many routing protocols such as RIPv2, OSPF, EIGRP, etc., use multicasting to efficiently send out network advertisements. A router can only send out network advertisements to the other routers in a multicast group.

6.8 The Need for Subnetting

6.9 Calculating Available Subnets

6.10 Calculating Available Hosts

6.11 Subnetting Practice Exercise #1

6.12 Subnetting Practice Exercise #2

6.13 Subnetting Practice Exercise #3

6.14 Calculating Usable Ranges of IPv4 Addresses

6.15 Subnetting Practice Exercise #4

6.16 Subnetting Practice Exercise #5

6.17 Classless Inter-Domain Routing (CIDR)

Chapter 7

IPv6 Addressing

Part II

LAN Switching

Chapter 8

Fundamentals of Ethernet

Chapter 9

Basic Cisco Catalyst Switch Configuration

Chapter 10

Virtual LANs (VLANs)

Chapter 11

Trunking

Chapter 12

Troubleshooting Switch Operation

Chapter 13

Basic Switch Security

Chapter 14

Voice VLANs

Part III

IP Routing

Chapter 15

Basic Router Operation

Chapter 16

Basic Router Configuration and Verification

Chapter 17

Routing Fundamentals

Chapter 18

Routing Information Protocol (RIP)

Part IV

Network Services

Chapter 19

Dynamic Host Configuration Protocol (DHCP)

Chapter 20

Network Address Translation (NAT)

Chapter 21

Network Time Protocol (NTP)

Part V

Network Management

Chapter 22

Network Management Protocols

Chapter 23

Device Management

Chapter 24

Troubleshooting with Cisco IOS Tools