Area 13: **<u>TCP/IP model (5-layer model)</u>** –

5 Layer Model: -

- **5Application Layer -**
  Interfaces for application programs to access network services, variety of protocols such as HTTP, SMTP, FTP etc
  Sessions and Presentation layer functionality is built into Application layer protocols that need it.
- **4Transport Layer: process-to-process delivery -**
  First true end-to-end layer (lower layers are implemented in intermediate-hop devices), packetization (segmentation/reassembly) of higher-level data, end-to- end reliability.
- **3Network Layer -**
  Global logical addressing, static/dynamic routing, congestion management, quality-of-service (e.g. delay, jitter), translating between heterogeneous link-layer technologies, managing differing packet-size limits, maximum transmission units (MTUs).
- **2Link Layer -**
  Framing, bit/byte stuffing, single-hop delivery, single-hop error detection, single-hop flow control, medium access control (for shared media), recovery strategies, automatic repeat request, sliding window, focus on point-to-point links.
- **1Physical Layer -**
  Transmission of raw bits, encoding bits in signals, data rates, synchronisation of sender & receiver clocks, transmission mode (simplex, half-duplex, full-duplex).

<u>Application Layer</u>

- The Application Layer builds upon the process-to- process delivery services provided by the Transport Layer below it.
- Application-Layer messages are encapsulated in Transport-Layer packets.
- Each Application-Layer message is formatted according to its own protocol rules and can encapsulate other types of data.

Domain Name System – DNS

- IP addresses and port numbers work well for computers but are not so workable for humans.
- DNS maps human-friendly names to IP addresses. This allows humans to work in terms of names while still allowing lower-level protocols to work in terms of space-efficient and processing-efficient fixed-size numbers.
- The DNS name space is hierarchical.

Traditional Applications

- Two of the most popular—
  - Email – request/reply
  - World Wide Web – request/reply
- It is important to distinguish between **application programs** and **application protocols**.
- Two very widely-used, standardised application protocols:
  - SMTP: Simple Mail Transfer Protocol is used to exchange electronic mail.
  - HTTP: HyperText Transport Protocol is used to communicate between Web browsers and Web servers.
- For example, the **HyperText Transport Protocol** (HTTP) is an application protocol that is used to retrieve web pages from remote servers.
- All browsers use the same HTTP protocol to communicate with web servers.

Email Architecture

- **User Agents -**
  The software programs that users run on their local machines to send/receive email. Examples include Thunderbird, Evolution, Mailspring, and Outlook.
- **Message Transfer Agents –**
  The software programs that manage mail delivery through the Internet. Examples include Send mail, Postfix, Exim & MS Exchange.
- **SMTP (Simple Mail Transfer Protocol) Push Protocol** –
  The protocol used for mail submission (user agent to message transfer agent) and transfer (between message transfer agents).
- **POP** (Post Office Protocol) **Pull Protocol**
- **IMAP** (Internet Message Access Protocol**) Pull Protocol-** p
- Both protocols are used for last-hop delivery from a message transfer agent to a user agent.

Electronic mail

Three major components

- User agents – e.g Mail reader
- Mail servers
- Simple mail transfer protocol: **SMTP**

World Wide Web

The original goal of the Web was to find a way to organise and retrieve information, drawing on ideas about hypertext - interlinked documents - that had been around since at least the 1940s.

The core idea of hypertext is that one document can link to another document, and the protocol (HTTP) and document language (HTML) were designed to meet that goal.

- HTTP (**Hyper-Text Transfer Protocol**) – the key WWW protocol for transferring data from servers to clients:
- HTTP Request – message sent by a client requesting a specific document from a server.
- HTTP Response – message sent back by the server. Ideally it contains the requested document, but other cases are possible. e.g. if the requested document does not exist, a "404 Not Found" response is sent.
- HTML (**Hyper-Text Markup Language**) – a document formatting language that allows, among other things, a document to include elements from elsewhere on the Web.

Transport Layer

- The Transport Layer builds upon the services provided by the Network Layer below it; and provides services to the Application Layer above it.
- Transport-Layer messages are encapsulated in Network-Layer packets (TPDU – Transport Protocol Data Unit).

Services:

- The Network Layer provides a host-to-host delivery service.
- Given an IP address, IP can deliver a message to the correct machine on the network.
- The primary contribution of the Transport Layer is to provide a process-to-process delivery service.
  - A process is a program that is executing (running).
  - Computers don't need to talk to each other.
  - Individual processes running on computers need to talk to each other
  - When a message arrives at the destination host, it needs to be delivered to a specific process running on that host – e.g. a web server, mail server, SSH login server, web browser, mail client, SSH login client etc
  - If an incoming message is handed to the wrong process, confusion will result – even if the message has been delivered to the correct host.
- The Transport Layer provides a network- independent API (Application Programming Interface).
- Application programs can access network functionality in a consistent manner regardless of the underlying Network-Layer or Link-Layer technologies in use.
- This makes network application code more portable and shields the application programmer from lower-layer differences.
- The Transport Layer may simultaneously offer a variety of service models to the Application Layer, allowing applications to choose whichever options best suit their needs.
- Typical offerings include:
  - reliable in-order connection-oriented delivery service
  - byte-stream or message-stream service
  - unreliable connectionless datagram service
  - congestion control
  - flow control

Transport-Layer Addressing

- The Transport Layer relies on host-addressing provided by the Network layer
- Within each host, however, the Transport Layer must add a mechanism for addressing an individual process.
- The TCP/IP / Internet way of doing this is for each socket opened by a program to have an identification number, called a port number.
- Port numbers are controlled by the operating system and are guaranteed unique within a host.

- In general terms, the combination of IP address and port number is unique within the Internet for a given socket type (protocol).
- Note that strictly speaking, a port number does not identify a process, so much as a socket. A process can have multiple o pen sockets, and therefore may be reachable on multiple different port numbers.
- However, a socket can only belong to one process. So, a port number does effectively identify a process.
- Port numbers are 16 bits, so can have any value up to 64K.
- Port numbers below 1024 are reserved for standard services that can usually only be started by privileged users.
- These are called well- known ports eg
- **Port:** *20,21* - **Protocol:** *FTP –* **Use:** *File Transfer*
- **Port**: *25-* **Protocol**: *SMTP –* **Use**: *Email*
- **Port**: *80-* **Protocol**: *HTTP –* **Use**: *World Wide Web*

Network Layer

- The Network Layer relies upon the services of the Link Layer below to send and receive packets.
- It provides a service interface to the Transport Layer above.
- The Network Layer receives data from the Transport Layer above and encapsulates it in Network-Layer packets.
- Communication between Network-Layer peers is virtual, with actual communication being via lower layers.

Network Layer Services

- **Host-to-host delivery**: delivering data across multiple networks with multiple underlying Link- Layer technologies
- **Fragmentation and reassembly**: chopping up Transport-Layer packets into pieces small enough to send in Link-Layer frames, and then putting the pieces back together at the destination.

Key problems for the Network Layer.

- **Routing -** making the choice of what path a packet will take through the network to get to its destination.
- **Addressing -**
  that will scale globally and still be configurable in local administrative domains, typically hierarchical in structure.
- **Address translation -** mapping between Network- Layer addresses and Link-Layer addresses.

Store-and-forward packet delivery model

- Data is divided into discrete units called packets. Each packet might contain a whole message from a higher layer or might contain only part of a message or a segment of a data stream.
- Each packet is labelled with addressing information.
- As a packet arrives at a node, it is fed into a memory buffer where it accumulates until it has fully arrived.
- The node performs error detection (e.g. check- summing).
- Then the node must choose how best to get the packet closer to its final destination – i.e. which outgoing link to forward it on.
- If the selected link is busy, the packet may have to wait in a queue until it can be allocated time on the link.
- Queuing times can vary with node workloads / network traffic volumes.
- The packet is then transmitted onto the chosen outgoing link.
- The Network-Layer packet is **re-encapsulated for every link it passes over**. The encapsulations can be quite different for different Link-Layer technologies, but the Network-Layer packet is recovered at each hop.

Area 4: **Fragmentation and Segmentation of packets** –

Internet Protocol – IP

- The primary method to build scalable heterogeneous internets is the Internet Protocol (IP).

The IP service model dictates two requirements:

  - An addressing scheme to uniquely identify every host on the internet.
  - A datagram (connectionless) model of data delivery.

IPv6 Datagram Header

- **Version**: ipv6.
- **Priority**:
- **Flow Label**: prioritised delivery of packets for services like voice or streaming
- **Payload Length**: the length of IPv6 payload; extension headers, upper layer protocol data
- **Next Header**: type of first extension or protocol in upper layer such as TCP/UDP etc,
- **Hop Limit**: packets hop travel limit
- **Source Address**: packet source.
- **Destination Address**: packet destination

Fragmentation and Reassembly

- We are trying to deliver over a collection of **heterogeneous** networks, each with its own limit on the allowable size of a packet.
- For example, Ethernet = 1500 bytes, FDDI (Fibre Distributed Data Interface) = 4500 bytes.
- We have two options within the IP service model:
  - Make all IP datagrams small enough to fit inside a packet on any network technology
  - Provide a means to fragment and reassemble datagrams when they are too big for a particular network link that t hey need to traverse.
- Two problems with first strategy:
  - We don't know what new network technology will be invented. How can we be sure we set our limit low enoug h?
  - Small packets waste bandwidth, because the ratio of header bytes to payload bytes is higher than for large packe ts. It also means there are more packets, adding to the workload on routers which have to make routing decisions for each individual packet.
- Every network type has a maximum transmission unit (MTU).
- This is the size of the largest IP datagram that it can carry in a single frame.
- Each IP datagram is re-encapsulated for each physical network it travels over
- Try to avoid fragmentation at source host; make size of datagram equal to MTU.
- Fragment only when necessary: if MTU < size of datagram
- Fragments can themselves be fragmented if necessary.
- Delay reassembly until fragments reach destination host
- Fragments themselves are self-contained datagrams.
  - Therefore, they are individually routable.
  - They may arrive out-of-order.
- If a fragment fails to arrive at the destination, all related fragments are discarded
- An IPv6 datagram exactly 370 bytes wide, consisting of a 40- byte IP header, four 30-byte extension headers, and 210 bytes of data.
- Two of the extension headers are unfragmentable, while two are fragmentable.
- We need to send this over a link with an MTU of only 230 bytes.
- • Require 3 fragments, because of the need to put the two 30-byte unfragmentable extension headers in each fragment, and the requirement that each fragment be a length that is a multiple of 8.