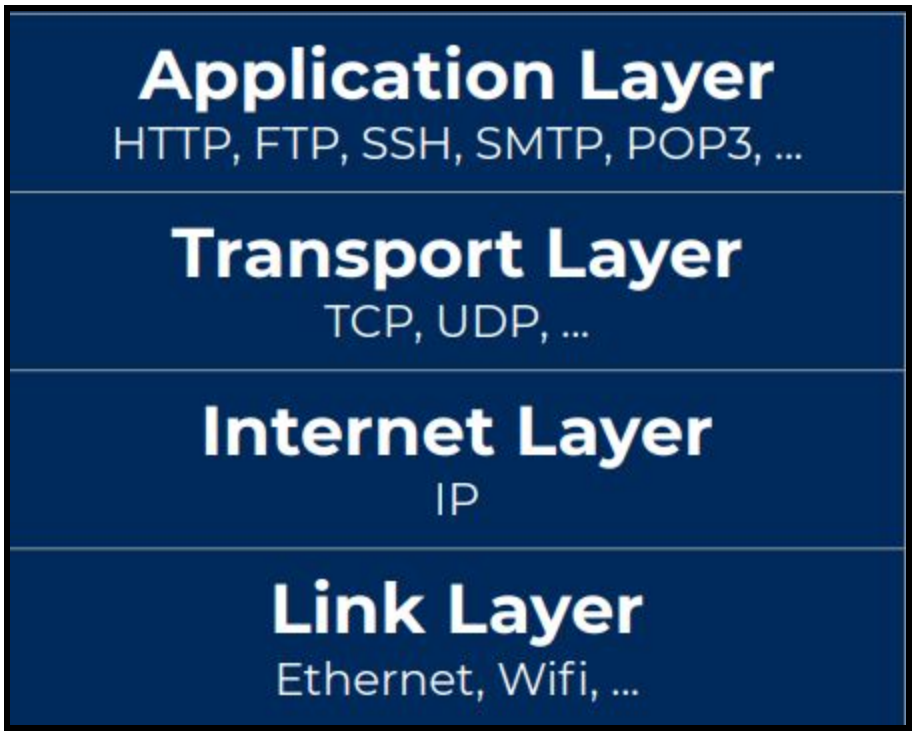
Computer Networks:

- Let's say we wanted to build a network of computers. At the minimum, we need the following:
    1. Computers.
    2. A method of communication between those computers.
- Furthermore, computers have to "talk" to each other. They need a language (set of rules). This language or set of rules is called a **network protocol**. **Network protocols** are sets of established rules that dictate how to format, transmit and receive data so computer network devices, from servers and routers to endpoints, can communicate regardless of the differences in their underlying infrastructures, designs or standards.

Network 4-Layer Model:



- **TCP/IP** stands for **Transmission Control Protocol/Internet Protocol**. It is specifically designed as a model to offer a highly reliable and end-to-end byte stream over an unreliable internetwork.
- TCP/IP Model helps you to determine how a specific computer should be connected to the internet and how data should be transmitted between them. It helps you to create a virtual network when multiple computer networks are connected together. The purpose of TCP/IP model is to allow communication over large distances.
- The functionality of the TCP/IP model is divided into four layers, and each includes specific protocols. All these four layers work collaboratively to transmit the data from one layer to another. These 4 layers are:
    - Application Layer
    - Transport Layer
    - Internet Layer
    - Network Interface
- In the **client-server model**:
    1. The client initiates a request to the server.

2. The server accepts or rejects the connection.
3. If a connection is established, data can flow until the connection terminates.

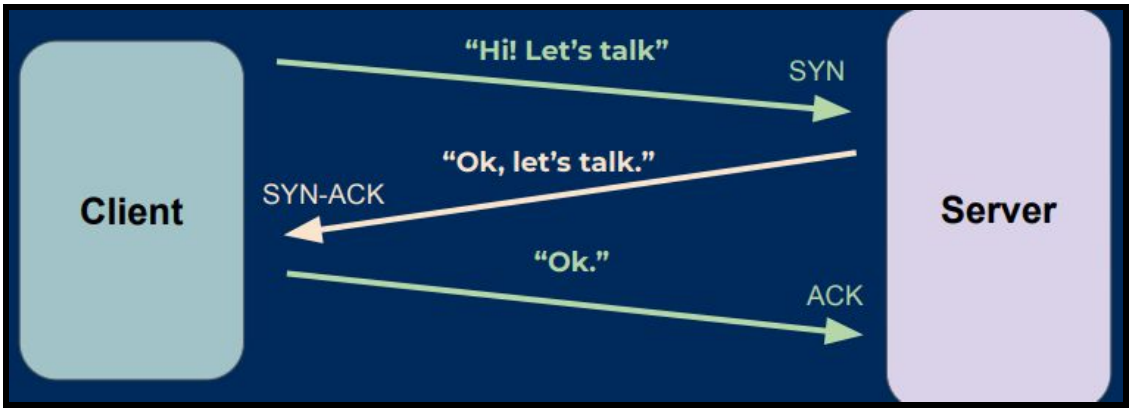
TCP/IP facilitates connections between clients and servers over many networks, including the internet.

- Here are the essential characteristics of TCP/IP protocol:
  - Support for a flexible architecture.
  - Adding more systems to a network is easy.
  - In TCP/IP, the network remains intact until the source and destination machines are functioning properly.
  - TCP is a connection-oriented protocol.
  - TCP offers reliability and ensures that data which arrives out of sequence should be put back into order.
  - TCP allows you to implement flow control, so the sender never overpowers a receiver with data.
- **Application Layer:**
  - Also known as **Process Layer**.
  - It is the topmost layer in the TCP/IP model.
  - Its functions are:
    - Helping you to identify communication partners, determining resource availability, and synchronizing communication.
    - Allowing users to log on to a remote host.
    - Providing various email services.
    - Offering distributed database sources and access for global information about various objects and services.
    - Responsible for handling high-level protocols, issues of representation.
    - Allowing the user to interact with the application.
    - When one application layer protocol wants to communicate with another application layer, it forwards its data to the transport layer.
    - Responsible for node-to-node communication and controls user-interface specifications.
    - Providing applications with standardized protocols to exchange data. E.g. Web browsers need a protocol to get and send data.
  - Protocols include HTTP, FTP, SSH, SMTP, POP3, etc.
  - **HTTP:** HTTP stands for Hypertext transfer protocol. This protocol allows us to access the data over the world wide web. It transfers the data in the form of plain text, audio, video. It is known as a Hypertext transfer protocol as it has the efficiency to use in a hypertext environment where there are rapid jumps from one document to another.
  - **SNMP:** SNMP stands for Simple Network Management Protocol. It is a framework used for managing the devices on the internet by using the TCP/IP protocol suite.
  - **SMTP:** SMTP stands for Simple mail transfer protocol. This protocol is used to send the data to another e-mail address.
  - **DNS:** DNS stands for Domain Name System. An IP address is used to uniquely identify the connection of a host to the internet, but people prefer to use the

names instead of addresses. The DNS maps the domain name to its respective IP address.

- **TELNET:** It is an abbreviation for Terminal Network. It establishes the connection between the local computer and remote computer in such a way that the local terminal appears to be a terminal at the remote system.
- **FTP:** FTP stands for File Transfer Protocol. FTP is a standard internet protocol used for transmitting the files from one computer to another computer.

- **Transport Layer:**
  - Also known as **Host-to-Host Layer**.
  - Its functions are:
    - Providing host-to-host communication services. It is **connection-oriented**, which means that the 2 hosts (client and server) must know each other before starting communication.
      I.e. In connection oriented service we have to establish a connection before starting the communication. When connection is established, we send the message or the information and then we release the connection.
    - Sending segments of data that came from the application layer. These segments of data are called **packets**.
    - Responsible for end-to-end communication and error-free delivery of data.
    - Determining how much data should be sent where and at what rate. This layer builds on the messages which are received from the application layer. It helps ensure that data units are delivered error-free and in sequence.
    - Responsible for the reliability, flow control, and correction of data which is being sent over the network.
  - The two main protocols in this layer are :
    1. **Transmission Control Protocol (TCP):**
       - It is connection-oriented. This means that it needs to have a pre-arranged connection before sending data. This pre-arranged connection should be bi-directional, meaning that both the client and the server should acknowledge when they get data.
       - To get a connection using TCP, we use the 3-way handshake:



       - The client and server can now send each other data, and must acknowledge each other when they recieve something.

        - Acknowledgements are an important part of TCP because it can check if a packet is from the correct host and can check if the other side received a packet(s). Losing packets is a real problem. If there is no acknowledgment that the packet was received, then the packet is sent again.
        - While TCP is reliable, it reacts to losing packets by slowing connection.
        - It is known to provide reliable and error-free communication between end systems.
        - It performs sequencing and segmentation of data. It also has an acknowledgment feature and controls the flow of the data through a flow control mechanism.
        - It is a very effective protocol but has a lot of overhead due to such features. Increased overhead leads to increased cost.
    2. **User Datagram Protocol (UDP):**
        - On the other hand, UDP does not provide any such features.
        - It is the go-to protocol if your application does not require reliable transport as it is very cost-effective.
        - Unlike TCP, which is a connection-oriented protocol, UDP is connectionless.
        - Furthemore, UDP is not reliable as it doesn't react to packet loss. UDP is good for live-streaming and playing games.
- **Internet Layer:**
    - Also known as **Network Layer**.
    - Its functions are:
        - Providing protocols for sending packets across a network or through multiple networks.
        - The main responsibility of the internet layer is to send the packets from any network, and they arrive at the destination irrespective of the route they take.
    - The **Internet Protocol (IP)** handles this in TCP/IP. It routes data across networks using IP addresses.
    - The IP is a connectionless protocol. There is no pre-arranged connection required to send data. IP just sends packets over networks.
        - No assurance that they will be delivered.
        - No way to find out if they were.
        - Nothing to let the destination know to expect a packet.
    - IP packets are easy to spoof. Connectionless protocol means you can send around packets that pretend they came from a specific IP address. These can cause **DDOS (Denial of Service) attacks**. Defense against this can come from higher network layers and network monitoring such as looking for suspicious activity and preventing attacks before they happen.
- **Link Layer:**
    - Also known as **Network Access Layer**.
    - It defines how the data should be sent physically through the network.
    - It is the lowest level. TCP/IP can sit on top of any link layer.
    - Its functions are:

        - Helping you to define details of how data should be sent using the network.
        - This layer is mainly responsible for the transmission of the data between two devices on the same network.
    - It uses the protocols of the physical link between the nodes of the network, such as Ethernet, WiFi, DSL, etc.
      I.e. The link layer is the group of methods and communications protocols confined to the link that a host is physically connected to. The **link** is the physical and logical network component used to interconnect hosts or nodes in the network and a link protocol is a suite of methods and standards that operate only between adjacent network nodes of a network segment.

The World Wide Web:

- The World Wide Web is not the Internet, but it happens to use the Internet to do its work.
  I.e. The World Wide Web works over the Internet and especially through the HTTP protocol.
- The **web** is a global collection of resources which are identifiable and linked together.
- "A global collection of resources" means:
    - A web resource can be any data we can send through the internet.
      E.g. Text, images, video, audio, etc.
    - Global means that we want to access these resources no matter where they are in the world.
    - **Note:** These resources are stored on **web servers**, which are computers with resources that are accessible.
- "Which are identifiable" means:
    - We need a way to **get these resources** from their web servers. Note that it is necessary that we can **locate** where they are in the entire web and that we need a **consistent** way to identify and access each resource.
      This is where **Uniform Resource Locator** (URL) comes in.
    - URLs provide us with a way of specifying the location of a web resource.
      I.e. A web address
- "And linked together" means:
    - Resources link to other resources. This allows us to easily discover the web.
      Note: Those that are similar tend to link to each other.
      E.g. Suppose you're on the Wikipedia page for Canada. There should be a link to the Prime Minister of Canada.

HTTP Protocol:

- The protocol of the web.
- Gives the client and server a mutual language at the application layer.
- Since all machines can use HTTP through applications, it is a global collection of resources.
- Identified from URLs.

- **Note:** URLs are not unique to HTTP. They are used in other protocols as well.
- **Note:** Technically speaking, we should be putting a dot "." at the end of the URL. All top-level domains have a dot "." at the end.
- Computers can only understand numbers (IP addresses) while humans like remembering strings (Hostnames). To solve this issue, hostnames are translated to IP addresses by the **Domain Name System (DNS)**. Furthermore, IP addresses can change, while the hostname stays the name.
  E.g. Google has many webservers, all with different IP addresses, that will take you to their homepage.
  The DNS is basically a complex lookup system. It resolves hostnames to IP through a DNS resolver. Your (Internet Service Provider) ISP will give you an address to the resolver, which will communicate with a bunch of name servers to find the IP.

- URLs give us one resource, a web page. Most websites however, have more than one resource. We can access them by extending the URL as needed.
  E.g. http://google.com → http://google.com/location/of/resource

Using HTTP:

- We use the web to:
  1. Download things
  2. Upload things
  3. Change things
     I.e. Update our credit card info
  4. Delete things
     I.e. Embarrassing pictures
- HTTP works by **request-response**.
  I.e. Request from client ○ Response from server.
  Request and response originate from the application layer on both sides.
- **HTTP Request includes:**
  - **URL:** To get to the resource on the server we want.
  - **HTTP Method:** To tell the server what we want to do with that resource.
  - **Request Headers and Body:** Gives the server additional information about our request.

- **HTTP Methods:**
  - **HTTP Methods** are verbs that are used to label the actions we expect a server to take.
  - Technically speaking, the server doesn't have 100% obligation to do these expected actions, but they are pretty well followed standards.
  - List of HTTP Methods:

Verb	Expected Server Action
GET	Retrieve a resource
POST	Create a resource
PATCH	Update a resource
DELETE	Delete a resource

- **Example of a GET Request:**
  - Let's say we wanted to access a course website homepage: www.teach.cs.toronto.edu/~csc148h/winter/index.html.
  - Since we are retrieving a resource (a web page), we use the GET method. The request looks like:

```
GET /~csc148h/winter/index.html HTTP/1.1
Host: www.teach.cs.toronto.edu
```

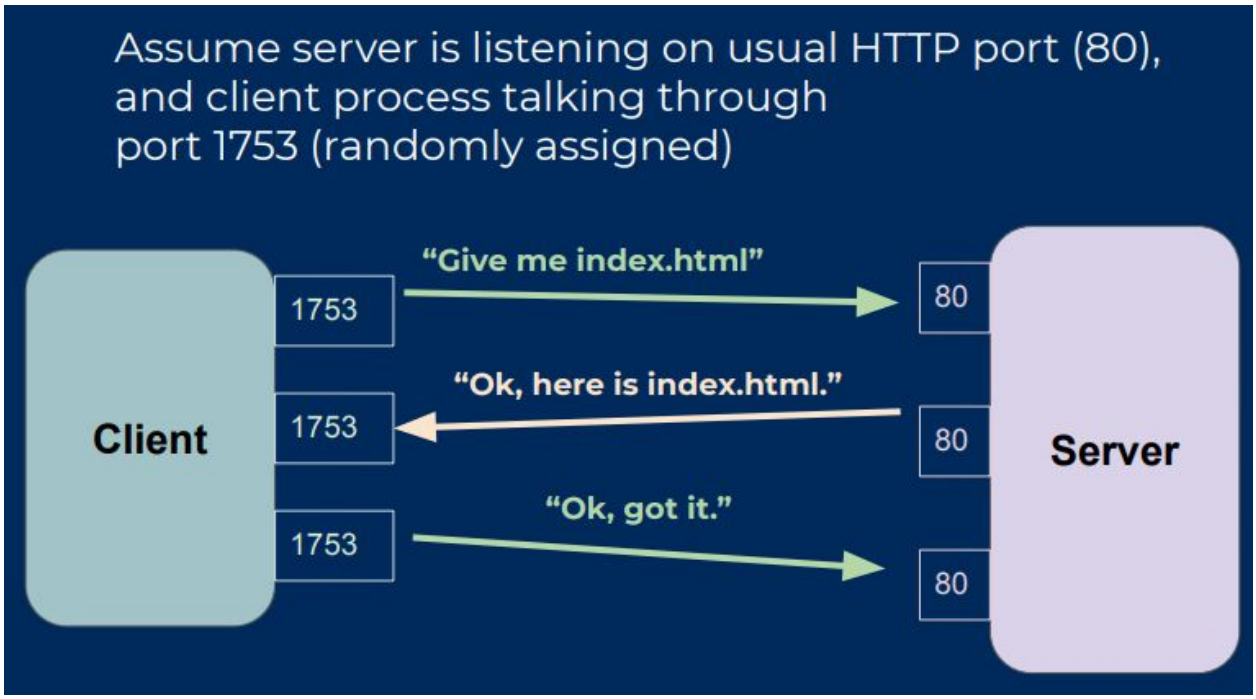
  - **HTTP method:** GET
  - **Resource:** /~csc148h/winter/index.html
  - **Host:** www.teach.cs.toronto.edu
  - **HTTP version:** HTTP/1.1
    Note: HTTP didn't go through many revisions until introduction of HTTP/2.0 in 2015. This allowed for pushing of resources that the client may want at a later time but hasn't requested yet. Some newer apps take advantage of these additional features, and there is backwards compatibility to older standards.
- **Note:** The web server decides what the URL does. Just because it looks like a path to some file in a filesystem, doesn't mean it actually looks like that on the server.

HTTP Linking Together:

- HTTP stands for **HyperText Transfer Protocol**.
- **Hypertext** is text/resources with **hyperlinks**, which are links to other resources.
- Similar resources are often linked together.
- This is what gives us the feeling of a connected web.

How does this look one layer down:

- Recall that HTTP is part of the application layer. We will look at what happens in the TCP layer.
- Remember that a web server listens for a request. That means there needs to be a process on the server that is listening. Issue: what if there are multiple processes that want to listen for connections? This is where ports come in.
- Every process on a computer that uses the internet is assigned a port. Specifically, a TCP or UDP port.
- Server process that listens for HTTP requests usually uses port 80.
  E.g.



- HTTP is a stateless protocol. This means that each request is independent and that the server doesn't need to keep track of previous requests and doesn't care how many are sent at once. While this simplifies the protocol, it's not always ideal. For example, if you log in to a website, you shouldn't be continuously logging in. There should be a way to track when you've logged in and when you've logged out. Similarly, consider an Amazon account. When you log in, you should be able to see your previous purchases.
- Illusion of state (knowing which pages the user browsed) can still occur, but is not part of the protocol.

Extra Information:

Note: We did not talk about the OSI model or about the DNS in this much detail in the lecture. This part is extra.

DNS:

- The **Domain Name System (DNS)** is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com. Web browsers interact through **Internet Protocol (IP)** addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.

- Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses.
- **How does DNS work:**
- The process of DNS resolution involves converting a hostname (such as www.example.com) into a computer-friendly IP address (such as 192.168.1.1). An IP address is given to each device on the Internet, and that address is necessary to find the appropriate Internet device, like a street address is used to find a particular home. When a user wants to load a webpage, a translation must occur between what a user types into their web browser (example.com) and the machine-friendly address necessary to locate the example.com webpage.
- In order to understand the process behind the DNS resolution, it's important to learn about the different hardware components a DNS query must pass between. For the web browser, the DNS lookup occurs " behind the scenes" and requires no interaction from the user's computer apart from the initial request.
- **There are 4 DNS servers involved in loading a webpage:**
    1. **DNS recursor**: The recursor can be thought of as a librarian who is asked to go find a particular book somewhere in a library. The DNS recursor is a server designed to receive queries from client machines through applications such as web browsers. Typically the recursor is then responsible for making additional requests in order to satisfy the client's DNS query.
    2. **Root nameserver:** The **root server** is the first step in translating (resolving) human readable host names into IP addresses. It can be thought of like an index in a library that points to different racks of books - typically it serves as a reference to other more specific locations.
    3. **TLD nameserver:** The **top level domain server (TLD)** can be thought of as a specific rack of books in a library. This nameserver is the next step in the search for a specific IP address, and it hosts the last portion of a hostname (In example.com, the TLD server is "com").
    4. **Authoritative nameserver:** This final nameserver can be thought of as a dictionary on a rack of books, in which a specific name can be translated into its definition. The authoritative nameserver is the last stop in the nameserver query. If the authoritative name server has access to the requested record, it will return the IP address for the requested hostname back to the DNS Recursor (the librarian) that made the initial request.
- **What are the steps in a DNS lookup:**
- For most situations, DNS is concerned with a domain name being translated into the appropriate IP address. To learn how this process works, it helps to follow the path of a DNS lookup as it travels from a web browser, through the DNS lookup process, and back again. Let's take a look at the steps.
- **Note:** Often DNS lookup information will be cached either locally inside the querying computer or remotely in the DNS infrastructure. There are typically 8 steps in a DNS lookup. When DNS information is cached, steps are skipped from the DNS lookup process which makes it quicker. The example below outlines all 8 steps when nothing is cached.
- The 8 steps in a DNS lookup:
    1. A user types 'example.com' into a web browser and the query travels into the Internet and is received by a DNS recursive resolver.

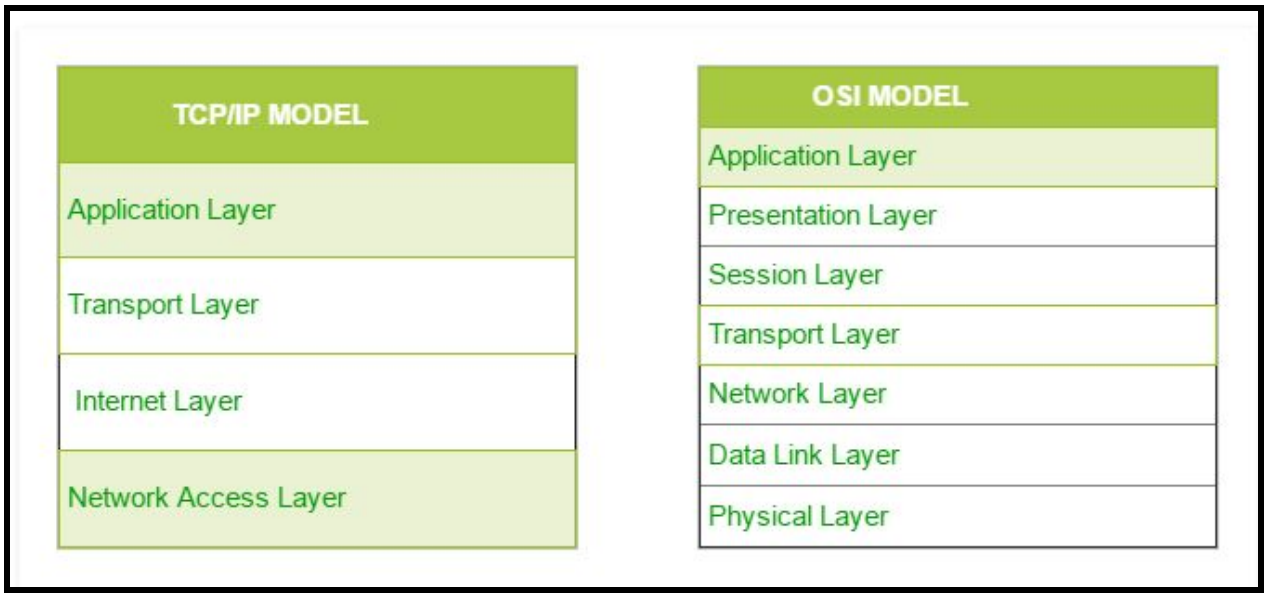
2. The resolver then queries a DNS root name server (.).
3. The root server then responds to the resolver with the address of a Top Level Domain (TLD) DNS server (such as .com or .net), which stores the information for its domains. When searching for example.com, our request is pointed toward the .com TLD.
4. The resolver then makes a request to the .com TLD.
5. The TLD server then responds with the IP address of the domain's nameserver, example.com.
6. Lastly, the recursive resolver sends a query to the domain's nameserver.
7. The IP address for example.com is then returned to the resolver from the nameserver.
8. The DNS resolver then responds to the web browser with the IP address of the domain requested initially.

   Once the 8 steps of the DNS lookup have returned the IP address for example.com, the browser is able to make the request for the web page:

9. The browser makes a HTTP request to the IP address.
10. The server at that IP returns the webpage to be rendered in the browser.

OSI Model:

- OSI stands for Open Systems Interconnection.
- It was developed in 1984 while the TCP/IP model was developed in 1960.
- It has 7 layers while the TCP/IP has 4 layers. The TCP/IP model is a concise version of the OSI model.
- A comparison of the TCP/IP and OSI models are shown below:



- The application layer in the TCP/IP model corresponds to the application, presentation and session layers in the OSI model.
- The transport layer in the TCP/IP model corresponds to the transport layer in the OSI model.
- The internet layer in the TCP/IP model corresponds to the network layer in the OSI model.

- The network access layer in the TCP/IP model corresponds to the data link and physical layers in the OSI model.
- Table of differences between the TCP/IP and OSI models:

TCP/IP	**OSI**
TCP refers to Transmission Control Protocol.	OSI refers to Open Systems Interconnection.
TCP/IP has 4 layers.	OSI has 7 layers.
TCP/IP is more reliable	OSI is less reliable
TCP/IP does not have very strict boundaries.	OSI has strict boundaries
TCP/IP follow a horizontal approach.	OSI follows a vertical approach.
TCP/IP uses both the session and presentation layers in the application layer itself.	OSI uses different session and presentation layers.
TCP/IP developed protocols then a model.	OSI developed a model then protocol.
The transport layer in TCP/IP does not provide assurance delivery of packets.	In the OSI model, the transport layer provides assurance delivery of packets.
The TCP/IP model network layer only provides connectionless services.	Connectionless and connection-oriented services are both provided by the network layer in the OSI model.
Protocols cannot be replaced easily in the TCP/IP model.	In the OSI model, protocols are better covered and are easier to replace with the change in technology.