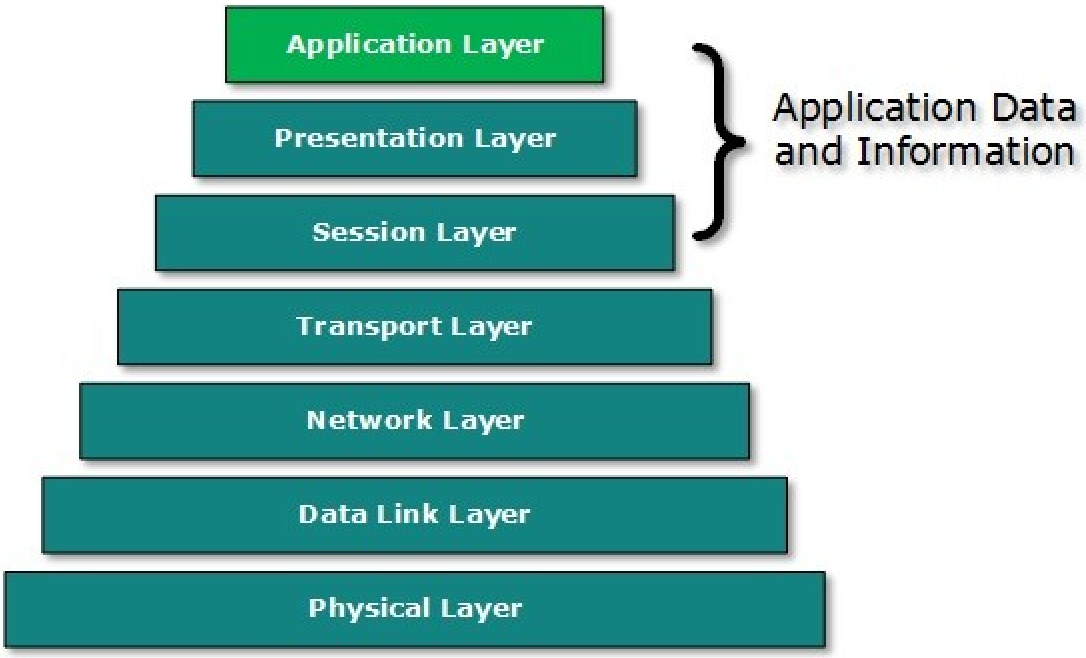
APPLICATION LAYERS PROTOCOL

Application layer is the highest most layer in OSI and TCP/IP layered model and. This layer exists in both layered Models because of its significance which is interacting with user and user applications. This layer is for applications which are involved in communication system.

A user may or may not directly interacts with these applications. Application layer is where the actual communication is initiated and reflects. Because this layer is on the top of the layer stack it does not serve any other layers. Application layer takes the help of Transport and all layers below it to communicate or transfer its data to the remote host.

When an application layer protocol wants to communicate with its peer application layer protocol on remote hosts it hands over the data or information to the Transport layer. The transport layer does the rest of the things with help of all layers below it.



There’s an ambiguity in understanding Application Layer and its protocol. Not every user application can be put into Application Layer. Only application which interacts with the communication system. For example, a designing software or text-editor cannot be considered as application layer programs.

On the other hand when we use a Web Browser, which is actually using HTTP (Hyper Text Transfer Protocol) to interact with the network. So in this case, HTTP is Application Layer protocol which we take into consideration when we study layered models.

Another example is File Transfer Protocol, which helps a user to transfer a text based or binary file across the network. A user can use this protocol in either GUI based software like FileZilla or CuteFTP and the same user can use FTP in Command Line mode.

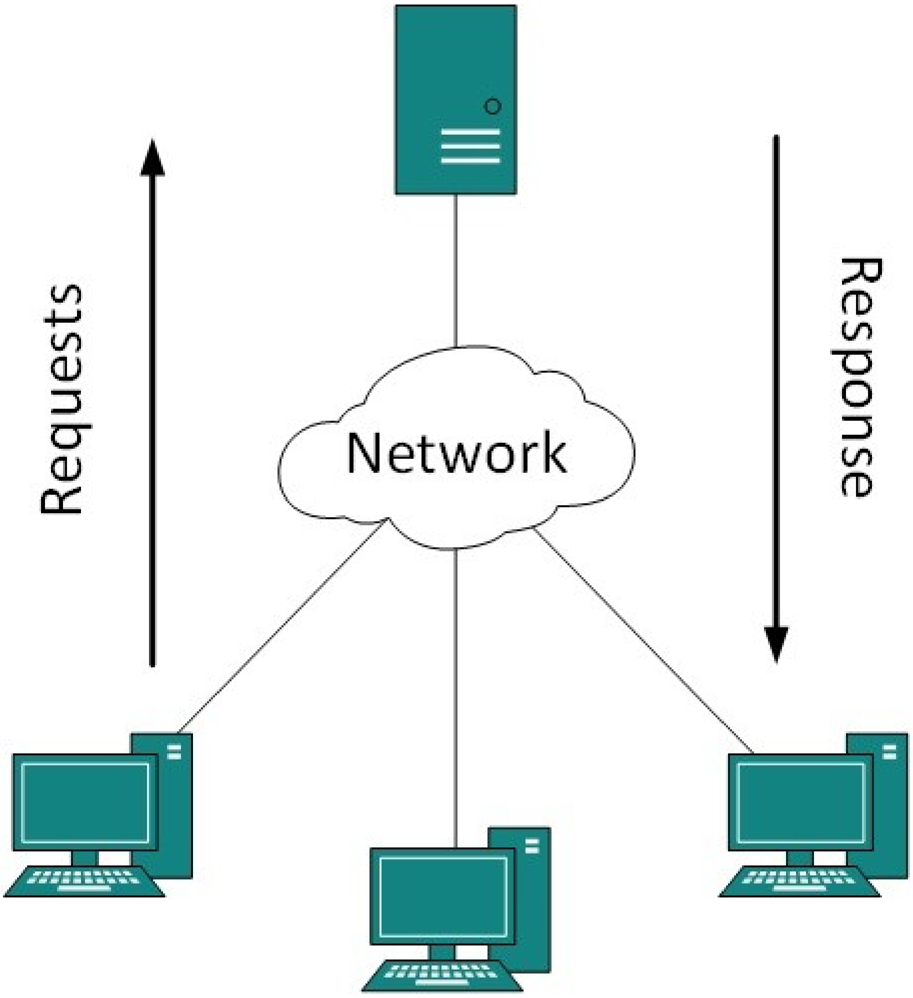
So it is not important what software you use, it the protocol which is considered at Application Layer used by that software. DNS is a protocol which helps user application protocols like HTTP to accomplish its work.

Client Server Model

Two remote application process can communicate in mainly two different fashions:

- **Peer-to-peer:** Both remote processes are at same level and exchange data using some shared resource.
- **Client-Server:** One the remote process acts as Client and requests some resource from another application process acting as Server.

In client-server model, any process can act as Server or Client. This not the machine or size of the machine or its computing power which makes it server but it is the feature of serving request that makes it server.



A system can act as Server and Client simultaneously. That is, one process is acting as Server and another is acting as a client. This may also happen that both client and server processes reside on the same machine.

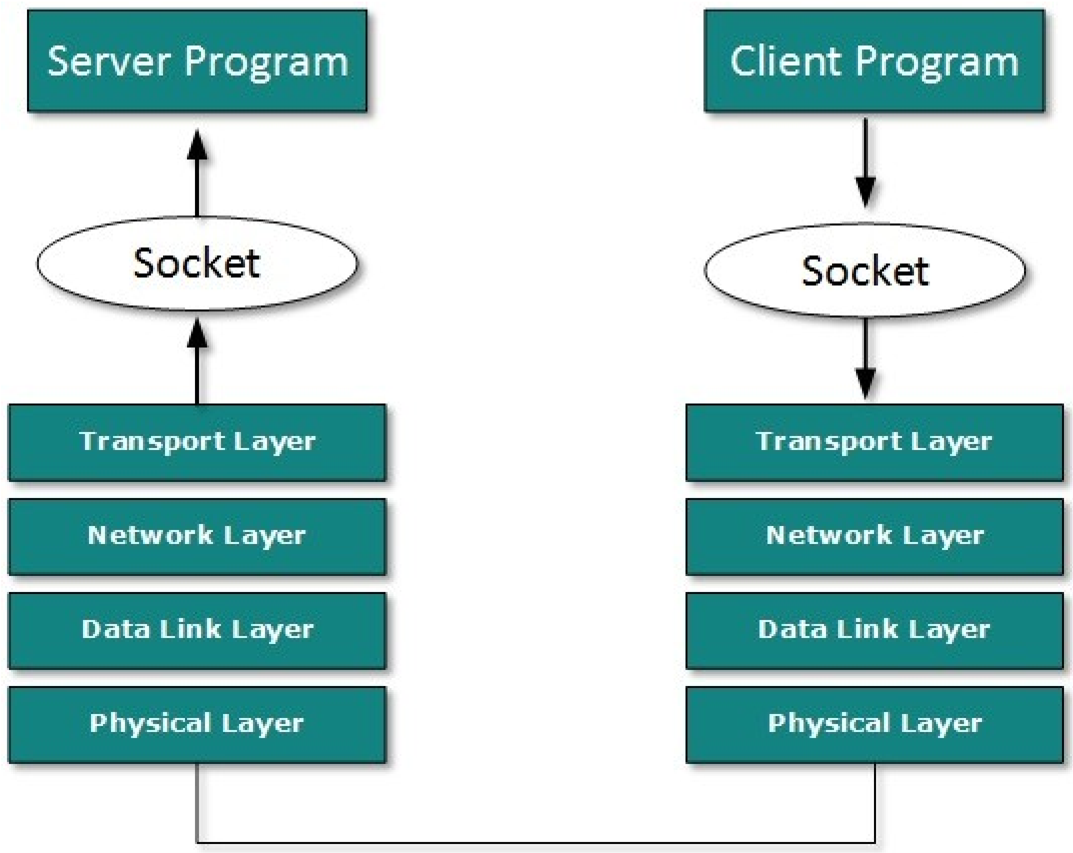
Communication:

Two processes in client-server model can interact in various ways:

- Sockets
- Remote Procedure Calls (RPC)

Sockets:

In this paradigm, the process acting as Server opens a socket using a well-known (or known by client) port and waits until some client request comes. The second process acting as Client also opens a socket but instead of waiting for an incoming request, client process 'requests first'.



When the request is reached to server, it is served. It can either be an information sharing or resource request.

Remote Procedure Call

This is a mechanism where one process interacts with another by means of procedure calls. One process (client) calls the procedure lying on remote host. The process on remote host is said to be Server. Both processes are allocated stubs. This communication happens in the following way:

- The client process calls the client stub. It passes all the parameters pertaining to program local to it.
- All parameters are then packed (marshalling) and a system call is made to send them to other side of the network is made.
- Kernel sends the data over the network and the other end receives it.
- The remote host passes data to the server stub where it is unmarshalled.
- The parameters are passed to the procedure and the procedure is then executed.
- The result is sent back to the client in the same manner.