# Open Systems Interconnection

* OSI is a model on how applications can communicate over a network.
* Any communication between two products must follow the OSI rules for communicating with each other.

OSI explains 7 layers in a device and set of protocols for communication between them.

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | Data Format | Protocols | Function |
| Application | Data | HTTP, Telnet, SSH, FTP, Wifi, DNS, SMTP | Deals with User interface, ways to communicate b/w networks to transfer data. |
| Presentation | MIME | Data Encoding, Compression, data Encryption |
| Session | SSL, TLS, NetBIOS | Establish, control, terminate a session |
| Transport | Segment(TCP)/Datagram(UDP) | TCP, UDP | Reliable data delivery, error recovery, data flow control, data delivery. |
| Network | Packet | IP(v4/v6), ICMP, IGMP,  IPsec | Logical addressing (IP) for path determination, routing, traffic control. |
| Data Link | Frame  (Header,Trailer) | PPP, HDLC, Ethernet | * Deals with data transmission b/w 2 nodes connected by physical layer. * Header has src and dest MAC address. * Trailer has frame check sequence for error detection. |
| Physical | Bits | Ethernet | Physical movement of data(bits) b/w nodes connected physically by connectors, pins, cables, etc |

# Layer 1 (Physical Layer)

* Deals with relationship of device and the physical transmission medium (Electric cable, Optical fibre cable, RF link).

|  |  |
| --- | --- |
| Connected Transmission | Connectionless/  Wireless Transmission |
| Done through Electric cable, Optical fibre cable etc. | Microwave, Infrared, Radio Frequency with help of Antenna. |
| Transmission Mode   1. Simple    * Single Channel, One way communication.    * Ex: Loud Speaker, Monitor (non-touch screen), Remote, Mouse, Keyboard. 2. Half-Duplex    * Single Channel, Two way communication (one way data transfer at a time).    * Ex: Walkie-Talkie 3. Full-Duplex    * Dual Channel, Two way communication (with dedicated channel for each, hence two way data transfer at any time).    * Ex: Telephone |  |
| Network Topology: point-to-point, Bus, Mesh, Star, Ring, hybrid, tree etc. |  |
| Thus deals with Signal loss, voltages, impedance, and cable specification. | Thus deals with Frequency, amplitude, antenna losses. |
| Ethernet, Repeaters, Hubs, Router, Switch, Modem | Bluetooth, Wifi. |

# Layer 2 (Data Link Layer)

* It detects and possibly corrects errors that may occur in the physical layer.
* It defines the protocol to establish and terminate a connection between two physically connected devices.
* It also defines the protocol for flow control (process of managing the rate of data transmission between two nodes to prevent a fast sender from overwhelming a slow receiver) between them.
* The data link layer is divided into two sub layers:
* [Medium access control](https://en.wikipedia.org/wiki/Medium_access_control) (MAC) layer – responsible for controlling how devices in a network gain access to a medium and permission to transmit data.
* [Logical link control](https://en.wikipedia.org/wiki/Logical_link_control) (LLC) layer – responsible for identifying and encapsulating network layer protocols, and controls error checking and frame synchronization.

# Layer 3 (Network Layer)

* Deals with functional and procedural ways of transferring data from one **node** to another connected in different networks with help of the addresses.

NOTE: In any network (wired/wireless) there will be number of nodes/devices listening to different ports, frequency etc hence each device while communicating should detect the correct destination device if not communication fails. Hence this job is done in this layer.

* Deals with routing protocols, multicast group management, network-layer information and error check, and network-layer address assignment.

# Layer 4 (Transport Layer)

* Deals with functional and procedural ways of transferring data from source to destination host.
* Deals with Segmentation/De-segmentation, error control.
* Keeps track of the segments and re-transmit those that fail delivery.
* Provides the acknowledgement of the successful data transmission.

# Layer 5 (Session Layer)

* Controls Connection (session) between devices.
* It establishes, manages and terminates the connections between the local and remote application/devices.
* It provides for [full-duplex](https://en.wikipedia.org/wiki/Duplex_(telecommunications)), [half-duplex](https://en.wikipedia.org/wiki/Half-duplex), or [simplex](https://en.wikipedia.org/wiki/Simplex_communication) operation, and establishes check pointing, adjournment, termination, and restart procedures

# Layer 6 (Presentation Layer)

* Establishes context between application-layer entities, in which the application-layer entities may use different syntax and semantics if the presentation service provides a mapping between them.
* Transforms data into the form that the application layer accepts (Ex: Encoded bits into ASCII file).

# Layer 7 (Application Layer)

* This layer is the closest to the end user, i.e., both the OSI application layer and the user interact directly with the software application.
* The functions include dealing with communication protocol, interface methods between source and destination devices.

# Ping

* Query/command to check if valid connection can be established between 2 machines and data can be transferred between them.
* Ping uses ICMP protocol for message exchange.

# ICMP (Internet Control Message Protocol)

* This is an Application Layer protocol in IP suite which is used by [network devices](https://en.wikipedia.org/wiki/Network_device), including [routers](https://en.wikipedia.org/wiki/Router_(computing)), to send error messages and operational information indication. (example: if a requested service is not available or that a host or router could not be reached)
* It isn’t used to exchange data between systems but just diagnostic error messages.
* If error, then the ICMP errors are directed/forwarded to the source IP address of the originating packet.

## WORKING

Every device forwarding an IP [datagram](https://en.wikipedia.org/wiki/Datagram) first decrements the [time to live](https://en.wikipedia.org/wiki/Time_to_live) (TTL) field in the IP header by one. If the resulting TTL is 0, the packet is discarded and an “ICMP [time exceeded in transit](https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol#Time_exceeded)” message is sent to the datagram's source address.

# DHCP (Dynamic Host Configuration Protocol)

* IP address is required for the user/device to access Internet. Allocation o fIP address can be done in 2 ways:

1. Manually by network Admin of Device
2. By ISP or DHCP Automatically on the go.

Thus DHCP is a network management Protocol used on TCP/IP which dynamically assigns IP address and other network configuration parameters (by enabling computers to request IP addresses and networking parameters automatically from the ISP) to make device communicate with other IP network.

## WORKING

The DHCP operates based on the [client–server model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model). When a computer or other device connects to a network, the DHCP client software sends a DHCP [broadcast](https://en.wikipedia.org/wiki/Broadcasting_(computing)) query requesting the necessary information. Any DHCP server on the network may service the request. The DHCP server manages a pool of IP addresses and information about client configuration parameters such as [default gateway](https://en.wikipedia.org/wiki/Default_gateway), [domain name](https://en.wikipedia.org/wiki/Domain_name), the [name servers](https://en.wikipedia.org/wiki/Name_server), and [time servers](https://en.wikipedia.org/wiki/Time_server). On receiving a DHCP request, the DHCP server may respond with specific information for each client, as previously configured by an administrator, or with a specific address and any other information valid for the entire network and for the time period for which the allocation (*lease*) is valid. A DHCP client typically queries for this information immediately after [booting](https://en.wikipedia.org/wiki/Booting), and periodically thereafter before the expiration of the information. When a DHCP client refreshes an assignment, it initially requests the same parameter values, but the DHCP server may assign a new address based on the assignment policies set by administrators.

## DHCP server has three methods of allocating IP addresses

1. **Dynamic allocation:**

A [network administrator](https://en.wikipedia.org/wiki/Network_administrator) reserves a range of IP addresses for DHCP, and each DHCP client on the [LAN](https://en.wikipedia.org/wiki/LAN) is configured to request an IP address from the DHCP [server](https://en.wikipedia.org/wiki/Server_(computing)) during network initialization. The request-and-grant process uses a lease concept with a controllable time period, allowing the DHCP server to reclaim and then reallocate IP addresses that are not renewed.

1. **Automatic allocation:**

The DHCP server permanently assigns an IP address to a requesting client from the range defined by the administrator. This is like dynamic allocation, but the DHCP server keeps a table of past IP address assignments, so that it can preferentially assign to a client the same IP address that the client previously had

1. **Manual allocation (static allocation):**

The DHCP server issues a private IP address dependent upon each client's MAC address, based on a predefined mapping by the administrator. This feature is variously called static DHCP assignment by DD-WRT, fixed-address by the dhcpd documentation, address reservation by Netgear, DHCP reservation or static DHCP by Cisco and Linksys, and IP address reservation or MAC/IP address binding by various other router manufacturers. If no match for the client's MAC address is found, the server may or may not optionally fall back to either Dynamic or Automatic allocation IP Address

IP address is the identity/path to trace a Device.

# (Domain Name System) – uses UDP

* DNS are set of servers maintained by ISP and other network users to map hostnames with IP addresses.
* DNS converts Hostname/Web Address ([www.google.co.in](http://www.google.co.in)) into IP (v4/v6) addresses (8.8.8.8).
* Domain Name Registrar registers Hostname with IP address so that users can reach the website.
* DNS name server is a server that stores the DNS records (in Database) for a domain, hence responds with answers to queries (when a Hostname is searched) against its database.
* Hostnames and IP addresses are not required to match in a one-to-one relationship. Multiple hostnames may correspond to a single IP address, which is useful in [virtual hosting](https://en.wikipedia.org/wiki/Virtual_hosting), in which many web sites are served from a single host. Alternatively, a single hostname may resolve to many IP addresses to facilitate [fault tolerance](https://en.wikipedia.org/wiki/Fault-tolerance) and [load distribution](https://en.wikipedia.org/wiki/Load_balancing_(computing)) to multiple server instances
* DNS Record types:
  + Start of Authority ([SOA](https://en.wikipedia.org/wiki/List_of_DNS_record_types#SOA))
  + [IP addresses](https://en.wikipedia.org/wiki/IP_address) ([A](https://en.wikipedia.org/wiki/List_of_DNS_record_types#A) and [AAAA](https://en.wikipedia.org/wiki/AAAA_record))
  + [SMTP](https://en.wikipedia.org/wiki/SMTP) [mail exchangers](https://en.wikipedia.org/wiki/Mail_exchanger) (MX)
  + [name servers](https://en.wikipedia.org/wiki/Name_server) (NS)
  + pointers for [reverse DNS lookups](https://en.wikipedia.org/wiki/Reverse_DNS_lookup) (PTR)
  + [domain name aliases](https://en.wikipedia.org/wiki/Domain_name_alias) (CNAME)
* The client side of the DNS is called a DNS resolver. A resolver is responsible for initiating and sequencing the queries that ultimately lead to a full resolution (translation) of the resource sought, e.g., translation of a domain name into an IP address. DNS resolvers are classified by a variety of query methods, such as *recursive*, *non-recursive*, and *iterative*.

## **DNS message format**

The DNS protocol uses two types of DNS messages, queries and replies, and they both have the same format.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Header | Question | Answer | Authority | Additional space |
| Identification, No of questions, *Number of answers*, *Number of authority resource records* (RRs), and *Number of additional RRs*. |  |  |  |  |

# SUBNET

* A **subnetwork** or **subnet** is a logical subdivision of an IP network
* The practice of dividing a network into two or more networks is called **subnetting**.
* Computers participating in a network such as the Internet have at least one network address. Usually this address is unique to each device and can either be configured:
  + automatically with the [Dynamic Host Configuration Protocol](#_DHCP_(Dynamic_Host) (DHCP) by a network server,
  + or manually by an administrator,
  + Or automatically by stateless address auto configuration.
* IPv4 – 32bits (8bits each – 4 byte – 4 octets)

255.255.255.255

|  |  |  |  |
| --- | --- | --- | --- |
| 0 - 255 | 0 - 255 | 0 - 255 | 0 - 255 |
| 20 - 28-1 | 20 - 28-1 | 20 - 28-1 | 20 - 28-1 |
| 11111111 | 11111111 | 11111111 | 11111111 |

* IPv4 = 127.0.0.1 🡪 same computer itself **(access your pc locally)**
* IPv6 =0.0.0.0:0.0.0.1 = ::1 🡪 same computer
* IPv6 – 128 bits (eight groups of four hexadecimal digits, each group representing 16 bits)

FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF = F:F:F:F:F:F:F:F

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 - FFFF | 0 - FFFF | 0 - FFFF | 0 - FFFF | 0 - FFFF | 0 - FFFF | 0 - FFFF | 0 - FFFF |

* This IPv4 address will be divided into further division 🡪

|  |  |
| --- | --- |
|  |  |

# Port Number

Port Number is the port/connection that is used for interacting.

Default ports for connections: HTTP 🡪 80, HTTPs 🡪 443

All of the network traffic that computers send and receive — everything from web requests, to login sessions, to file sharing — is split up into messages called *packets*.

Each packet has :

* IP addresses of the computer that sent it,
* IP address of the computer that receives it,
* port number for the sender and recipient (with the exception for some low-level packets, such as ping)

IP addresses 🡪 distinguishes computers;

Port numbers 🡪 distinguish programs on those computers.

# SSH (Secured Shell)

**SSH**: Secure Shell (shell = CLI) is a cryptographic network protocol for operating network services securely over an unsecured network.

An **SSH client** is a software program which uses the [secure shell](https://en.wikipedia.org/wiki/Secure_shell) protocol to connect to a remote computer.

Ex: WinSCP, Mobaxterm, PuTTy, XShell, FileZilla, etc.

An **SSH server** is a software program which uses the [secure shell](https://en.wikipedia.org/wiki/Secure_shell) protocol to accept connections from remote computers.

Ex: OpenSSH, Apache MINA, Dropbear, CrushFTP Server, Ish etc.

**Find below SSH remote Access Tools:**

1. **MobaXterm**:

* Mobaxterm is a remote computer access tool, to transfer files b/w 2 systems.
* So, any client/server can be accessed remotely with any required protocol using their IP address.
* Terminal uses Linux/Unix based commands for interaction.
* Network Tools: Secured shell (ssh), FTP, SFTP, HTTP, rsync, telnet, Rsh.

1. **WinSCP**: supports FTP, SFTP, WebDAV, SCP protocols only on windows OS.
2. **PuTTy** : supports SSH protocol on Windows, Solaris, Linux, BSD OS, MAC.
3. **FileZilla**: FTP protocol
4. **XShell**: supports SCP, FTP, SFTP, rlogin, SSH, Telnet on Windows OS.

**Brainstorm**

1. There are 2 major specifications of SSH (SSH-1, SSH-2).
2. SSH was developed as replacement for Telnet (secured), rlogin, rsh, rexec(unsecured) protocols which send sensitive information such as passwords in plain text while leads to their interception using packet analysis

# NAT (Network Address Translation)

* IPv4 has only 232 address = 4,294,967,296 (4 billion) addresses. But, world population is 7.6 Billion and also one single person can use many devices with different IP.
* Hence to resolve this inadequacy (shortage) IPV6 is introduced with 2128 addresses.
* IP address isn’t mapped one to one basis it can be single host to many IP address or many host to single IP address.
* So NAT is a workaround for the shortage of IPv4 Addresses.
* **Network address translation** (**NAT**) is a method of remapping one IP address space into another by modifying **network address information** in IP header of packets while they are in transit across a traffic routing device.
* NAT isn’t required in IPv6 address because of abundance of IP addresses.

## WORKING

The majority of NATs map multiple private hosts to one publicly exposed IP address. In a typical configuration, a local network uses one of the designated private IP address. A router on that network has a private address in that address space. The router is also connected to the Internet with a *public* address assigned by an [Internet service provider](https://en.wikipedia.org/wiki/Internet_service_provider). As traffic passes from the local network to the Internet, the source address in each packet is translated on the fly from a private address to the public address. The router tracks basic data about each active connection (particularly the destination address and port). When a reply returns to the router, it uses the connection tracking data it stored during the outbound phase to determine the private address on the internal network to which to forward the reply.

REFERENCE: <https://en.wikipedia.org/wiki/Network_address_translation>

# Firewalls/Filters

* Firewalls are devices that network operators can use to filter traffic that's coming into or leaving their network.
* A firewall is one example of a class of network devices called **middleboxes** — devices that inspect, modify, or filter network traffic.

Other examples of middleboxes include intrusion detection systems and load balancers. Technically, it's only a middle-box if it's a separate device from the client or server — server-side "firewalls" like Linux iptables aren't middleboxes.

* A firewall can be a real boon to an organization's network security. The most common configuration for a firewall is to drop any incoming traffic except traffic to (host, port) pairs that are supposed to be receiving connections from the Internet. This lets the network administrator be sure that other machines on the network — like backend databases or administrative systems — aren’t going to get direct attacks from outside.
* But firewalls can cause trouble for application developers. If you're trying to test or deploy a network app and there's a firewall between your server and the user, that firewall can potentially interfere with your app or block it completely. In order to deploy an application on a particular server and port, it helps to know what kind of firewall might be between you and your user. One of the reasons that many non-Web applications use HTTP as a transport is that HTTP is often unblocked at firewalls even when other ports are blocked.
* Aside from blocking traffic outright, middleboxes can also alter traffic, for instance replacing web pages with error messages. This is often done for social or political purposes. For instance, in the U.S., many schools use traffic filters of various sorts to prevent students from accessing web sites deemed inappropriate for children. But what sites get counted as "inappropriate" can reflect the biases or opinions of the people who wrote or configured the filter.
* And people who program these things can always make mistakes, too. For instance, there's a whole class of bugs that arise from filters that try to block rude words, but end up blocking or replacing innocuous words that contain a rude word as a substring.

# Proxies & NAT

With NAT, several devices can access Internet resources through a single public IP address, with the NAT device using port numbers to match up connections on the inside and outside.

For end-users, NAT devices overlap with firewalls. Typical home routers can act as both a NAT and a simple firewall, often having the ability to block or filter at a very basic level. At a larger scale, ISPs and other organizations have deployed NAT devices for their whole customer networks, called [**carrier-grade NAT**](https://en.wikipedia.org/wiki/Carrier-grade_NAT). This is very common for mobile networks, and also for ISPs in the developing world, where there never were anywhere near enough addresses allocated for the number of users.

Usually we imagine an end-user computer as having only one person using it at a time. After all, there's generally only one mouse and keyboard. Two people typing on the same keyboard at the same time doesn't generally happen outside of [**poorly thought-out TV shows**](https://www.youtube.com/watch?v=u8qgehH3kEQ). But in the case of NAT, your web site can see requests from the same IP address that actually come from different users on different computers.

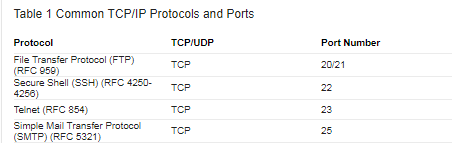
Another way that can happen is through the use of **web proxies**. Whereas a NAT works at the IP level, rewriting packets, a web proxy works at the HTTP level, taking queries from browsers and sending them out to web servers. Many organizations use web proxies for caching, including some ISPs. From the standpoint of a web developer or site operator, traffic from a busy proxy looks much the same as traffic from a busy NAT: queries for many users, on many actual computers, are funnelled through a single public IP address.

# Bandwidth

<http://blog.chriszacharias.com/page-weight-matters>

# **TCP: Transport Layer Protocol**

* TCP enables two hosts to establish a connection and exchange streams of data i.e., TCP is ***used to interconnect network devices on the internet.***
* This is a connection oriented protocol i.e, it first establishes connection between two hosts and then data can be sent bidirectional.
* UDP 🡪 similar to TCP but it is connectionless
* Below are the common protocols using TCP (connection oriented) model for communication.



# HTTP (Hyper Text transfer Protocol)

GUIDE: <https://developer.mozilla.org/en-US/docs/Web/HTTP>

**It is** an **application layer** **stateless protocol** used to transfer **Hypertext** documents over the internet.

* **Stateless protocol**: no info retained by server/client, i.e sender sends packet and doesn’t expect acknowledgement from client
* But it is **Session Protocol** (Session is information exchange between client & server)
* Hypertext is **structured text** (could be a file/document/img/video etc) that uses logical links (hyperlinks) for transfer of data.

Example: Go to Google.com 🡪 search “Hello” you would find links i.e., hyperlinks to related documents, images, videos, text, design etc that has anything relative to the text “Hello”.

* Default port 80 to connect from client to browser (https uses port 443).

## **WORKING**

HTTP functions as a request–response protocol in the client–server computing model. A web browser, for example, may be the client and an application running on a computer hosting a website may be the server.

An HTTP client initiates a request by establishing a Transmission Control Protocol (TCP) connection to a particular port on a server (typically port 80, occasionally port 8080. An HTTP server listening on that port waits for a client's request message. Upon receiving the request, the server sends back a response i.e., status line, such as "HTTP/1.1 200 OK", and a message of its own. The body of this message is typically the requested resource or an error message or other information.

## **Authentication**

HTTP uses Basic Access Authentication (BAC) and Digest Access Authentication (DAC) for security.

### BAC

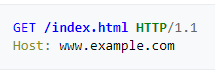
This is a simple user control access with help of login ID and password stored in easily reversible [Base64](https://en.wikipedia.org/wiki/Base64) encoding format (which isn’t encrypted or hashed hence HTTPs is used).

### DAC

This applies a hash function (MD5 cryptographic hash algorithm) to the username and password before sending them over the network.

## Client Request Format

|  |  |  |
| --- | --- | --- |
| Method | Path | HTTP version |
| GET | /house/tree.png | HTTP/1.0 |

Example: 

Safe 🡪 doesn’t affect status of server, method used only to access data.

Unsafe 🡪 affects status of server, method used to send data to server.

Idempotent 🡪 multiple requests are considered as single request.

Header fields: <https://en.wikipedia.org/wiki/List_of_HTTP_header_fields>

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Version** | **SAFE / IDEMPOTENT** | **Usage** |
| **GET** | Http /1.0 | Safe | To retrieve data from Web server with message body |
| **HEAD** | Http /1.0 | safe | Same as GET but response is without message body |
| **POST** | Http /1.0 | Unsafe. Non-Idempotent | Requests the server to accept data enclosed in request, thus causing change in state on the server.  Used when we do not know specific URL where the data enclosed will reside.  Ex: Submitting form, new login to a webpage. |
| **PUT** | Http /1.1 | Unsafe, Idempotent | Same as POST, replaces current data of target source with the request payload.  Used to create/overwrite resource of URL known by the client.  It contains ID of resource where new data to be modified  Ex: New user creation then navigation to that user page. |
| **PATCH** | Http/1.1 |  | Used to apply partial modifications to the resource.  Ex: Modify only email ID/name of the account, thus a patch of modification is done. |
| **OPTIONS** | Http /1.1 | Safe | Displays the request methods the server supports.  Ex: GET, PUT, POST etc displayed as output |
| **DELETE** | Same | Unsafe, Idempotent | Deletes specified resource |
| **TRACE** | Same | Safe | Echoes the received request so that a client can see what changes or additions have been made by intermediate servers. |
| **CONNECT** | Same | unsafe | Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy. |

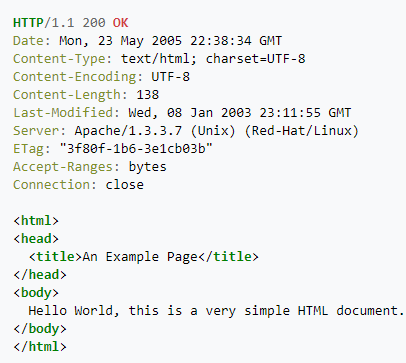
## Server Response Format

|  |  |  |
| --- | --- | --- |
| HTTP version | Status code | Status description |
| HTTP/1.0 | 200 | OK |

Above format then followed by the HTML code of the webpage.

Example:

Note: default file name of web documents will be index.html/default.htm/index.php



|  |  |  |
| --- | --- | --- |
| **Status Code** | **Status** | **Description** |
| 1xx | Informational | Request in progress/another step to take. |
| 2xx | Success | Request succeeded, server is sending data. |
| 3xx | Redirection | Server is telling client a different URI to access the data (due to URI change or something) |
| 4xx | Client Error | Server didn’t understand client request hence doesn’t know how to process it. |
| 5xx | Server Error | Issue on Server end (server off/hack/downtime etc) |

Status codes: <https://en.wikipedia.org/wiki/List_of_HTTP_status_codes>

Status Codes are responses sent by the server to the client’s request.

|  |  |  |
| --- | --- | --- |
| **Status Code** | **Status** | **Comment** |
| 100 | Continue |  |
| 101 | Switching Protocols |  |
| 102 | Processing |  |
| 103 | Early Hints |  |
|  |  |  |
| 200 | OK |  |
| 201 | Created |  |
| 202 | Accepted |  |
| 203 | Non Authorative Information |  |
| 204 | No Content |  |
| 205 | Reset Content |  |
| 206 | Partial Content |  |
| 207 | Multi Status |  |
| 208 | Already Reported |  |
| 226 | IM Used |  |
|  |  |  |
| 300 | Multiple Choices |  |
| 301 | Moved Permanently |  |
| 302 | Found |  |
| 303 | See Other |  |
| 304 | Not Modified |  |
| 305 | Use Proxy |  |
| 306 | Switch Proxy |  |
| 307 | Temporary Redirect |  |
| 308 | Permanent Redirect |  |
|  |  |  |
| 400 | Bad Request |  |
| 401 | Unauthorized |  |
| 402 | Payment required |  |
| 403 | Forbidden |  |
| 404 | Not Found |  |
| ------451 |  |  |
|  |  |  |
| 500 | Internal Server Error |  |
| 501 | Not Implemented |  |
| 502 | Bad Gateway |  |
| 503 | Service Unavailable |  |
| 504 | Gateway Timeout |  |
| 505 | HTTP Version not supported |  |
| 506 | Variant also negotiates |  |
| 507 | Insufficient Storage |  |
| 508 | Loop Detected |  |
| 510 | Not extended |  |
| 511 | Network authentication required |  |

## HTTP Headers

Headers contain data of client/server (Accept, connection, content-Type etc)

*Format: header\_name: value; header\_name:value; ….*

Headers are a sort of metadata for the response. They aren't displayed by browsers or other clients; instead, they tell the client various information about the response.

HTTP Headers list: <https://en.wikipedia.org/wiki/List_of_HTTP_header_fields>

## Caching

HTTP is a **Session protocol** i.e, ***Cache heade*r** stores transaction data on client/server for ease in rendering webpage quickly with the last stored **Session** thus **speeding up performance** (modify cache header in request to not save cache)

## Cookie

## It is the data that is passed between client and server during a session, thus sharing state of the request for the HTTP stateless protocol.

## Example to keep a user logged in save username, passwords, session ID etc will be stored in browser temporarily.

## Proxy

A proxy is a middle person between client and server.

Requests will be routed through proxies thus hiding client data/server data.

## Web Hosting Points

* A Server is a simple version of Client which doesn’t need any UI, Animation, Graphics etc. It just has to handle incoming requests and render/process them (http.server module of Python does this).
* Minification of JS, CSS, HTML file helps in enhancing page rendering time.

# AJAX

**Asynchronous Javascript and XMLHttpRequest -** is a set of many techniques working on client side to create asynchronous web Apps.

i.e With Ajax, web apps can send & retrieve data from Server asynchronously without affecting content (HTML), behaviour (JS) and display (CSS, BS) of existing page.

# JSON

**JavaScript Object Notation** – A common data format used for Asynchronous Client (browser) - Server communication.

# URL / URI (Uniform Resource Locator/Identifier)

A URI/URL is a Web Address, helps to identify webserver that is hosting a website/webpage.

URI Format 🡪 Scheme://hostname.port/path?query#fragment

Example

🡪 <https://www.example.com/8008/puppy.png?q=a&i=768#fjlak>

🡪 <mailto:example@xyc.com>

🡪 ftp:localhost/8080/pathfile.jpg

### Scheme/Protocol declaration

* Tells us how to access the resource, must be lowercase.
* Popular schemes: http, https, ftp, mailto, file, data, irc, magnet.

### Hostname

* Host is a computer on network i.e., the one that hosts services registered on DNS.
* Can be a name (if registered) or else an IPv4/v6 address.

### Path

* This will be absolute path to access data (displayed after hostname)
* Path = ‘/’ if not present. (Ex: <http://www.ex.com/>) 🡪 / indicates zero path.

### Query

* Extra information loaded onto web address by browser while accessing webpage.
* Attribute value pair started with “**?”** and separated by delimiter “&”

(Ex ?q=a&jk=909182&12=slo01/)

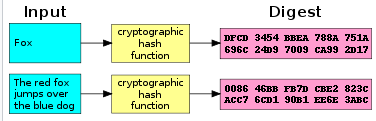
### Fragment

* Provides direction to secondary resource.
* Ex: if URL is a HTML page, fragment contains info to article name/id element etc.

# ENCRYPTION

## Hashing

* Hashing is a type of Data Encryption Method where in a Cryptographic Hash Function maps a data of random size into a data of Fixed Size.



* This function is a one way function i.e., it cannot be reversed.
* It is deterministic, so the same message always results in the same hash.
* It is quick to compute the hash value for any given message.
* It is infeasible to generate a message from its hash value except by trying all possible messages.
* A small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value.
* It is infeasible to find two different messages with the same hash value.

## Secure Hash Algorithm

This is a Cryptographic Hash function which accepts input and produces XX-Byte long hexadecimal number called a message digest.

### **SHA1: Secure Hash Algorithm 1**

* Produces a 20Byte(160-bit) hexadecimal output
* Not much safe since 2 different inputs gave similar output, hence no long supported.

### **SHA2: Secure Hash Algorithm 2**

* Successor to SHA1 which provides better Security and safeguard from Attacks.
* Produces 224, 256, 384, 512 bits hexadecimal output data.
* Includes six hash functions: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256

# **SECURITY**

## **SSL (Secure Sockets Layer)**

* Provides communication security over computer networks.
* Predecessor of TLS which is now prohibited from use by IETF.

## **TLS (Transport Layer Security) protocol**

* Provides security of Data over Computer networks with the help of Crypto graphical functions.
* Has 2 layers 🡪 TLS record, TLS Handshake
* Uses Algorithm for Hashing data which ensures it cannot be reversed.
* Each Message transmitted includes message integrity check to prevent packet loss/altercation.

## **Certification Authorities**:

* It is An Entity that issues Digital Certificates/Signatures.
* This digital certificate certifies ownership of public key (client end).
* Thus CA acts as third party to assert signatures digitally and is thus trusted by both client and server (referring to the communication interface).
* CA will be present on both client and server side, so user has to accept to CA guidelines.
* There are many CAs in market who provide Certifications commercially some of the prominent and high market shares ones are Comodo, Symantec, Godaddy, Verizon.

### **WORKING**

A CA issues [digital certificates](https://en.wikipedia.org/wiki/Digital_certificate) that contain a [public key](https://en.wikipedia.org/wiki/Public_key) and the identity of the owner.

The matching private key is not made available publicly, but kept secret by the end user who generated the key pair.

The certificate is also a confirmation or validation by the CA that the public key contained in the certificate belongs to the person, organization, server or other entity noted in the certificate.

Thus the Servers/relying parties can trust the client is the person he is claiming to be.

### **POINTS**

* The locked secure logo on a web page shows the security of the web page being accessed.

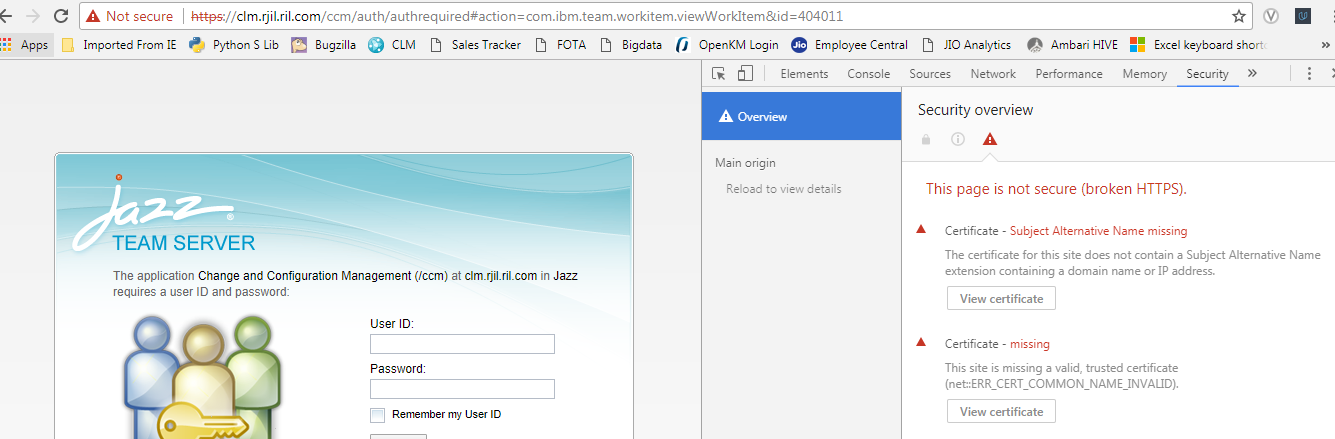


* On clicking this secure button we can check the certificate details, example: Certificate issuing authority, Hash algorithm used, Date of expiry, Public key, etc.
* Sometimes some of the web pages hosts data which is secured(HTTPS) but has source data from unreliable sources (HTTP instead of HTTPS) such as images, links etc and hence this to “Mixed Content”.

This makes webpages mixed secured thus making the data Unsecured.

This can be analysed with help of Dev tools (F12 🡪 Security tab)

To avoid this all the data must be transferred by HTTPS.



# HTTP 1.1

* HTTP /0.9 and HTTP /1.0 closes connection after a single request-response pair, hence in HTTP/1.1 continuous connection alive facility is made.

* HTTP /1.0 was revised to HTTP/1.1 to make the Data being transferred encrypt and secured.
* Disadvantage of HTTP/1.1:

1. No default data encryption.
2. One request at a time:

To display each page, a lot of data has to be received and requests to be handled, thus on an average it takes 100 request to render a complete webpage.

At any point of time, there can be a max 6 connections between a client&server.

Each HTTP Request will be completed once there is HTTP response from server thus making one round trip (request + response) which takes 20-50 milliseconds to complete since the data is in plain text form.

This is called Head of Line Problem where in it takes a lot of time for Webpage consisting of huge data to be rendered completely.

This issue can be solved by HTTP 2

1. Uncompressed headers in the request.

time can be saved by compressing and sending this over the connection, also called as gzip which is done only by HTTP2 as this isn’t supported in HTTP 1.

# HTTP 2

* It solves Head of Line Blocking by having a Single connection between client and server with help of Multiplexing i.e., requests can be sent by client without waiting for response from server.
* Enhances webpage rendering speed by removal of request and response headers (plain text format data) by having a Compressor (which replaces common Headers with references to them) shared by client and server for request and response.
* HTTP 2 is backward compatible i.e. pages designed with HTTP 2 will work with HTTP 1 as well.

# Router

A networking device that forwards data packets between computer networks.

## Working

A router is connected to two or more data lines from different networks.[b] When a data packet comes in on one of the lines, the router reads the network address information in the packet to determine the ultimate destination. Then, using information in its routing table or routing policy, it directs the packet to the next network on its journey.

# Switch

A computer networking device that connects devices together on a computer network by using packet switching to receive, process, and forward data to the destination device.

# Gateway

* A network node equipped for interfacing with another network that uses different protocols, also called as protocol converters.
* A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.
* A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.