ASSIGNMENT

**What is a server**

A server is a [computer program](https://en.wikipedia.org/wiki/Computer_program) or a [device](https://en.wikipedia.org/wiki/Computer) that provides functionality for other programs or devices, called "[clients](https://en.wikipedia.org/wiki/Client_(computing))".

**Types of server**

1.Physical server

A [physical server](https://www.rosehosting.com/dedicated-servers.html) is just as the name says, a server (physical computer) on which an Operating System, like Windows or Linux runs just as on any other computer. The physical servers are in almost all aspects like desktop computers, with many improvements that desktop PCs lack featuring things like redundant power supplies, raid controllers, multiple network cards etc. The physical servers are larger in size with much more powerful components in general. They all require a separate space in the server rack. Most of the servers also have two or more physical CPUs with multiple cores each.

2.Virtual server

Virtual Servers have hardware that is abstracted from the local hardware and run on top of physical servers. Once you virtualize a server, that server really is a file or object you run and you back it up, move it, and clone it easily.

Traditionally,the other types of server are

**1.FTP Server**

File Transfer Protocol (FTP ) is one of the oldest server types. It is responsible for transferring files from server to a computer and vice versa.

**2.Proxy Servers**

The Proxy server is responsible for a connection between a client(web browser or an app) with and an external server to entertain the request for connection, performance enhancement, and accessibility

**3.Online Gaming Server**

Gaming server has gained its popularity in a recent decay. This type of server is responsible for connecting hundreds of gamers around the world to an external server(s) for accessing gaming data.

Xbox live is one of the examples for gaming servers.

**4.Web servers**

The web server is responsible for hosting website files and serve it up through a web browser. It loads an individual file of a web page and loads it to display in the browser as one complete page.

**5-Application Servers**

Application servers have lion’s share in computer territory between database servers and the end user, where servers are often connected to the two.

**6-List Servers**

List servers are used to enhance the functionality & management of mailing lists. Whether they are an interactive database that is open to the public or one-way lists that deliver newsletters, announcements or advertising.

**7-Chat Servers**

This server enables a number of people to share information in the environment of an internet newsgroup that offer real-time discussion capabilities.

**8-IRC Servers**

Internet Relay Chat is comprised of various independent networks of servers that allow users to connect to each other via an IRC network. It is an option for those who are seeking real-time competence.

**9-Fax Servers**

Those organizations that want to reduce the incoming and outgoing telephone resources; a fax server is an ideal solution. However, there is a need to fax the actual document.

**10-Groupware Servers**

It is software that is designed to make the users able to work together, regardless of their location, through the Internet or a corporate Intranet and to work together in a virtual environment.

**11-Mail Servers**

The mail server just is as important as a web server is. A mail server is to send/receive and store emails on the corporate networks through LANs and WANs and across the internet.

**12-Telnet Servers**

By the help of it, users log on to a host computer and perform work as if they are working on an isolated computer.

**13-News Servers**

They work as a source of distribution and delivery for hundreds of available public newsgroups accessible over the USENET news network.

**Who is a client**

A client is a piece of [computer hardware](https://en.wikipedia.org/wiki/Computer_hardware) or [software](https://en.wikipedia.org/wiki/Software) that accesses a service made available by a [server](https://en.wikipedia.org/wiki/Server_(computing))

Characteristics of a server:

· It is initially passive (or slave, waiting for a query);

· It is listening, ready to respond to requests sent by clients;

· When a request comes, he treats it and sends a response

Characteristics of a client:

· It is the first active (or master);

· Sends requests to the server;

· It expects and receives responses from the server

**Client server architecture**

Client-server architecture, architecture of a computer network in which many clients (remote processors) request and receive service from a centralized server (host computer). Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns. Servers wait for requests to arrive from clients and then respond to them. Ideally, a server provides a standardized transparent interface to clients so that clients need not be aware of the specifics of the system (i.e., the hardware and software) that is providing the service. Clients are often situated at workstations or on personal computer, while servers are located elsewhere on the network, usually on more powerful machines.



**HTTP**

The request/response message consists of the following:

* Request line, such as GET /logo.gif HTTP/1.1 or status line, such as HTTP/1.1 200 OK,
* Headers
* An empty line
* Optional HTTP message body data

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred as *HTTP verbs*. Each of them implements a different semantic, but some common features are shared by a group of them: e.g. a request method can be [safe](https://developer.mozilla.org/en-US/docs/Glossary/safe), [idempotent](https://developer.mozilla.org/en-US/docs/Glossary/idempotent), or [cacheable](https://developer.mozilla.org/en-US/docs/Glossary/cacheable).

GET

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

HEAD

The HEAD method asks for a response identical to that of a GET request, but without the response body.

[PO](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)ST

The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.

PUT

The PUT method replaces all current representations of the target resource with the request payload.

DELETE

The DELETE method deletes the specified resource.

CONNECT

The CONNECT method establishes a tunnel to the server identified by the target resource.

OPTIONS

The OPTIONS method is used to describe the communication options for the target resource.

TRACE

The TRACE method performs a message loop-back test along the path to the target resource.

PATCH

The PATCH method is used to apply partial modifications to a resource.

The response-header fields allow the server to pass additional information about the response which cannot be placed in the Status- Line. These header fields give information about the server and about further access to the resource identified by the Request-URI.

· Accept-Ranges

· Age

· ETag

· Location

· Proxy-Authenticate

· Retry-After

· Server

· Vary

· WWW-Authenticate

HTTP header fields provide required information about the request or response, or about the object sent in the message body. There are four types of HTTP message headers:

· General-header: These header fields have general applicability for both request and response messages.

· Client Request-header: These header fields have applicability only for request messages.

· Server Response-header: These header fields have applicability only for response messages.

· Entity-header: These header fields define meta information about the entity-body or, if no body is present, about the resource identified by the request.

**TCP**

Transmission Control Protocol is one of the most used protocols in digital network communications and is part of the Internet protocol suite, commonly known as the TCP/IP suite. Primarily, TCP ensures end-to-end delivery of data between distinct nodes. TCP works in collaboration with Internet Protocol, which defines the logical location of the remote node, whereas TCP transports and ensures that the data is delivered to the correct destination.

Before transmitting data, TCP creates a connection between the source and destination node and keeps it live until the communication is active. TCP breaks large data into smaller packets and also ensures that the data integrity is intact once it is reassembled at the destination node.

A three-way handshake is a method used in a TCP/IP network to create a connection between a local host/client and server. It is a three-step method that requires both the client and server to exchange SYN and ACK (acknowledgment) packets before actual data communication begins.

A three-way handshake is also known as a TCP handshake.

Steps:

* A client node sends a SYN data packet over an IP network to a server on the same or an external network. The objective of this packet is to ask/infer if the server is open for new connections.
* The target server must have open ports that can accept and initiate new connections. When the server receives the SYN packet from the client node, it responds and returns a confirmation receipt – the ACK packet or SYN/ACK packet.
* The client node receives the SYN/ACK from the server and responds with an ACK packet



**UDP**

User datagram protocol is an open systems interconnection (OSI) transport layer protocol for client- server network applications. UDP uses a simple transmission model but does not employ handshaking dialogs for reliability, ordering and data integrity. The protocol assumes that error-checking and correction is not required, thus avoiding processing at the network interface level.

UDP is widely used in video conferencing and real-time computer games. The protocol permits individual packets to be dropped and UDP packets to be received in a different order than that in which they were sent, allowing for better performance.

UDP network traffic is organized in the form of datagrams, which comprise one message units. The first eight bytes of a datagram contain header information, while the remaining bytes contain message data. A UDP datagram header contains four fields of two bytes each:

* Source port number
* Destination port number
* Datagram size
* Checksum

**Difference between tcp and udp**

1.In case of tcp ,connection must be established before communication.Udp is connectionless

2.Tcp is suited for large amount of data and udp is suited for small amount of data

3.Tcp is therefore reliable mode of transfer and udp is unreliable

4.Transmission speed is high in udp compared to tcp

**Dictionary implementations using hash tables**

In general several different solutions to the map / dictionary problem: hash tables, Red-Black Trees, AVL Trees, and Skip Lists.

Hash tables use a hash function h : K → V to compute the location of a given value v in a table. The function is called a ’hash function’ because it ’mixes’ the data of its input, so that the output for similar inputs appears totally unrelated. When two members of K, say k1 and k2 have the same hash value, i.e. h(k1) = h(k2), then we say there is a hash collision, and this collision must be resolved.

**Hash Node Data Type**

The functions in hash map are

* get(K key) : returns the value corresponding to the key if the key is present in HT (Hast Table)
* getSize() : return the size of the HT
* add() : adds new valid key, value pair to the HT, if already present updates the value
* remove() : removes the key, value pair
* isEmpty() : returns true if size is zero

Every Hash Map must have an array list/linked list with an initial size and a bucket size which gets increased by unity every time a key, value pair is added and decreased by unity every time a node is deleted

ArrayList<HashNode<K, V>> bucket = new ArrayList<>();

A Helper Function is implemented to get the index of the key, to avoid redundancy in other functions like get, add and remove. This function uses the in built java function to generate a hash code and we compress the hash code by the size of the HT so that the index is within the range of the size of the HT

get()

The get function just takes a key as an input and returns the corresponding value if the key is present in the table otherwise returns null. Steps are:

* Retrieve the input key to find the index in the HT
* Traverse the linked list corresponding to the HT, if you find the value then return it else if you fully traverse the list without returning it means the value is not present in the table and can’t be fetched so return null

remove()

* Fetch the index corresponding to the input key using the helper function
* The traversal of linked list similar like in get() but what is special here is that one needs to remove the key along with finding it and two cases arise
* If the key to be removed is present at the head of the linked list
* If the key to be removed is not present at head but somewhere else

add()

Now to the most interesting and challenging function of this entire implementation.It is interesting because we need to dynamically increase the size of our list when load factor is above the value we specified.

* Just like remove steps till traversal and adding and two cases (addition at head spot or non-head spot) remain the same.
* Towards the end if load factor is greater than 0.7
* We double the size of the array list and then recursively call add function on existing keys because in our case hash value generated uses the size of the array to compress the inbuilt JVM hashcode we use ,so we need to fetch new indices for the existing keys. This is very important to understand please re read this paragraph till you get a hang of what is happening in the add function.