How Internet works | What happens when you type google.com in the browser?

- Notes By : Milind Mishra

> Prequisite : Networking basics like IP address, DNS, TCP/IP, HTTP, etc.

What happens when you type google.com in the browser?

- When we hit enter in the browser, the browser will send a request to the DNS server to resolve the IP address of the google.com.
- The DNS server will send a request to the root DNS server to resolve the IP address of the google.com.
- The root DNS server will send a request to the top level domain DNS server to resolve the IP address of the google.com.
- Internally DNS works on the concept of hierarchy. The root DNS server will send a request to the top level domain DNS server to resolve the IP address of the google.com.

How the Internet works?

- Lot of times the details are generally abstracted from us, but actually behind the scenes there are lot of things happening.
- The Internet is a network of networks. It is a network of computers that are connected to each other.
- A lot of components work together to make the Internet work, and its nessasary as an Engineer to understand how the Internet works.

> The URL consists of three parts: the protocol, the domain name, and the path. ex : `https://www.google.com/search?q=how+internet+works`

- The URL gets resolved to an IP address. And this resolution is done by the DNS server. [URL <-> IP Address]
- Earlier DNS used to be managed by Hosts.txt file in the local machine. But now it is managed by the DNS server.
- This Hosts.txt consisted of the IP address and the domain name. Drawback of this approach is that if the IP address changes, we need to update the Hosts.txt file.
- This all used to be managed by Network Information Manager (NIM) which used to hold a source of truth for the IP address and the domain name.
- However this was not scalable and hence the DNS server was introduced.

> DNS (Domain Name System) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols. By providing a worldwide, distributed directory service, the Domain Name System is an essential component of the functionality of the Internet, that has been in use since 1985.

- The DNS server is a server that holds the IP address and the domain name. It is a distributed system. Clearly, a better architecture than the NIM.
- The DNS being hirarchical, it has a root DNS server, which is the top level DNS server. It has a list of all the top level domain DNS servers.
- The DNS uses Prefix tree, trie used for efficient prefix matching.
- The tries ofcourse is a vast topic in itself, but the basic idea is that the tries are used to store the domain names in a tree like structure and are able to query faster as compared to the hash tables.
- The top level domain DNS server has a list of all the second level domain DNS servers.
- The second level domain DNS server has a list of all the third level domain DNS servers.
- The third level domain DNS server has a list of all the fourth level domain DNS servers.

```
Root DNS Server
|
|
- Top Level Domain DNS Server (.com, .org, .in, .net, etc.)
|
|
- Second Level Domain DNS Server (google.com, facebook.com, etc.)
|
|
- Third Level Domain DNS Server (www.google.com, www.facebook.com,
cs.harvard.edu, etc.)
|
|
- Fourth Level Domain DNS Server
```

- Machine as well maintains a local DNS copy if not available in the local then it will go to the DNS server and query for the IP address.

> `dig` is a network administration command-line tool for querying Domain Name System (DNS) name servers. It performs DNS lookups and displays the answers that are returned from the name server that was queried. It is a flexible tool that can be used to query any DNS server for any DNS record type. It is also a useful debugging tool for verifying DNS records.

```
11:26:07 milind-lab@milind-labpc ~ → dig google.com

; <<>> DiG 9.18.12-0ubuntu0.22.04.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 684
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
```

```
;; QUESTION SECTION:
;google.com.            IN  A

;; ANSWER SECTION:
google.com.     123 IN  A   142.250.205.238

;; Query time: 36 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Thu Apr 13 23:26:12 CST 2023
;; MSG SIZE  rcvd: 55
```

- `-short` is a flag to just get the IP address.

```
11:28:49 milind-lab@milind-labpc ~ → dig google.com +short
142.250.205.238
```

> `nslookup` is a computer network administration command-line tool available for many operating systems for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or for any other specific DNS record.

```
11:30:32 milind-lab@milind-labpc ~ → nslookup google.com
Server:     127.0.0.53
Address:    127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.205.238
Name:   google.com
Address: 2404:6800:4007:82d::200e
```

- This communication behind the scenes works in a Client Server Architecture.
- Lets say, client is the browser and the server is the DNS server.
- Anything that makes a request is a client and anything that responds to the request is a server.
- This communication between the client and the server is done using the TCP/IP protocol.
- The TCP is a connection oriented protocol. It establishes a connection between the client and the server.
- The TCP incurrs with a 3 way handshake. The client sends a SYN packet to the server, the server responds with a SYN-ACK packet and the client responds with an ACK packet.
- Comming to servers, there are two types of servers, the web servers and the application servers.
- The web servers are the servers that serve the static content. The web servers are the servers that serve the HTML, CSS, JS, images, etc.

- The application servers are the servers that serve the dynamic content. The application servers are the servers that serve the data from the database.
- These databases are generally relational databases like MySQL, PostgreSQL, etc.
- The servers can be hosted locally or on the public cloud like AWS, GCP, Azure, etc.
- Thrashing happens when the CPU is not able to keep up with the memory access. This is because the CPU is not able to keep up with the memory access.
- Inorder to cope up with this, the CPU caches the memory. The CPU caches the memory in the L1, L2, L3 cache.
- Then there are strategies like Horizontal Scaling, Vertical Scaling, etc.
- *Vertical Scaling* is when we increase the size of the server. For example, if we have a 2GB RAM server, we can increase it to 4GB RAM server.
- Drawback is that it is expensive and it is not scalable, also single point of failure.
- *Horizontal Scaling* is when we increase the number of servers. For example, if we have a 2GB RAM server, we can add another 2GB RAM server.
- Its distributed and hence balanced out the load.
- `Load Balancer` is a server that distributes the load across the servers.
- It comprises of a `Reverse Proxy` and a `Load Balancer`, uses algorithms like Round Robin, Least Connection to route the requests to the servers.
- It accomodates the health of the servers, if a server is down, it will not route the requests to the server.
- *Auto Scaling* is when the load balancer automatically scales up and down the servers based on the load.
- *Auto Scaling* is a feature of AWS, GCP, Azure, etc.
- Drawback is that when we spin up a new server, it takes time to configure the server.

A very interesting video on how hotstar scaled for 25 million concurrent users

- Based on use case, the server architecture can be designed and implemented.
- The server architecture can be designed and implemented using the `Microservices Architecture`, `Monolithic Architecture`, etc.
- Message Queues are used to park a request and process it later kinda like pub-sub model.
- Message Queues are used to decouple the services from each other and also to scale the services.
- Use case like Codechef where the submissions are queued and processed later is a good example of Message Queues.

- All codebase in a single repository is an example of Monolithic Architecture.
- Each service has its own repository is an example of Microservices Architecture.
- Dev issues with Monolithic Architecture:
    - Codebase is huge.
    - Difficult to scale.
    - Difficult to maintain.
    - Difficult to test.
    - Difficult to deploy.
    - Difficult to debug.
    - Difficult to understand.
- Dev Benifits of Microservices Architecture:
    - Easy to scale.
    - Easy to maintain.
    - Easy to test.
    - Easy to deploy.
    - Easy to debug.
    - Easy to understand.
- Customer Pain points of Monolithic Architecture:
    - Slow.
- Customer Benifits of Microservices Architecture:
    - Fast as like lets say Payment service can be scaled up and down individually.

> There is no one size fits all architecture. It depends on the use case. For example, if the use case is to build a simple blog, then Monolithic Architecture is the best choice. But if the use case is to build a complex application like Facebook, then Microservices Architecture is the best choice.

- Setting up of application servers like message queues, databases are some things that you must know as a Developer.
- WebSockets is a protocol that allows for bidirectional communication between the client and the server.
- Its not a Client Server Architecture, its a Peer to Peer Architecture.
- Its a persistent connection between the client and the server.
- The piped connection between the client and the server is called a `Socket`, half duplex connection or full duplex connection.
- Based on TCP/IP protocol it simmilarly recieves ACK packets from the server and sends ACK packets to the server.

> Usecase for WebSockets is like a chat application where the messages are sent in real time, example WhatsApp, Slack, etc.

- `PM2` Process Manager 2 can spawn multiple instances of the application and can also restart the application if it crashes.
- PM2 can also be used to deploy the application to the production server.
- `Nginx` is a web server that can be used to serve the static content.
- Ngnix is also a Proxy Server that can be used to route the requests to the application servers.
- Popular frameworks for development is MVC which stands for Model View Controller.
- Model is the data layer, View is the presentation layer and Controller is the business logic layer.

> Lets talk with an example, Chef is the Model and the View is the UI and the Controller is the business logic. The business logic is to add a recipe, delete a recipe, etc. The UI is the View and the data is the Model. Model is the Chef that cooks the food and the View is the UI that shows the food and the Controller is the business logic that decides what to cook and what not to cook.

- `MVC` is a design pattern that is used to separate the business logic from the presentation layer.
- Separation of Concerns is the main benifit of MVC as the business logic is separated from the presentation layer and the data layer.
- The business logic is separated from the presentation layer and the data layer. That it must be dumb towards SQL Queries, etc.
- Likewise Presentation Layer is separated from the business logic and the data layer. That it must be dumb towards the business logic.
- Likewise Data Layer is separated from the business logic and the presentation layer. That it must be dumb towards the business logic and the presentation layer.
- Unit Testing becomes easy because of keeping this good code hygiene.
- The code becomes more maintainable and scalable.
- The code becomes more readable and understandable.
- And likely the code becomes more testable.
- For an instance src being the root folder, the folder structure for MVC is like this:
    - src
        - controllers
        - models
        - views
    - tests
        - controllers
        - models
        - views

- This type of structure makes good code hygiene and makes the code more maintainable and scalable.