1. **Port Number**

* **HTTP: 80**
* **HTTPS: 443 TCP/UDP**
* TCP/IP connection is always made to an IP address (you can think of an IP-address as the address of a certain computer, even if that is not always the case) and a specific (logical, not physical) port on that address.
* Usually one port is coupled to a specific process or "service" on the target computer. Some port numbers are [standardized](https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers), like 80 for http, 25 for smtp and so on. Because of that standardization you usually don't need to put port numbers into your web adresses.
* So if you say something like [http://www.stackoverflow.com](http://www.stackoverflow.com/), the part "stackoverflow.com" resolves to an IP address (in my case 64.34.119.12) and because my browser knows the standard it tries to connect to port 80 on that address. Thus this is the same as [http://www.stackoverflow.com:80](http://www.stackoverflow.com/).

**localhost**/web is equal to **localhost:80**/web OR to **127.0.0.1:80**/web

**localhost:8080**/web is equal to **localhost:8080**/web OR to **127.0.0.1:8080**/web

## HTTP - 80

Port 80 is associated with HTTP, Hypertext Transfer Protocol. It comes under the category of a **TCP protocol**. It is one of the most famous and widely used ports in the world. The main purpose of port 80 is to allow the browser to connect to the web pages on the internet. Port 80 basically expects or waits for the web client to ask for a connection. Once this connection has been made, you will get the privilege to connect to the World Wide Web and get access to various web pages out there. In fact, HTTP - 80 is one of the most important ports associated with the TCP protocol. Moreover, this port is generally used during the application layer of the TCP/IP Model.

## HTTPS - 443

HTTPS - 443 is also associated with the **TCP protocol**. HTTPS port 443 also lets you connect to the internet by establishing a connection between the webpages and the browser. This lets you connect to the World Wide Web. However, **this port has an added feature of security to it**, which HTTP port 80 does not have. This port is intended for establishing secure connections to make sure that the data is transmitted over a secure network. The use receives a warning if the browser is trying to access a webpage which is not secure. This port comes into being during the application layer. It basically encrypts and authenticates the network packets before transferring them over the network to increase the security. This feature of security is introduced by the use of SSL, which can also be referred to as Secure Socket Layer.

## FTP - 20, 21

FTP is the abbreviation of "File Transfer Protocol". The purpose of FTP is to transfer files over the internet. It basically lays down all the rules which are to be followed during the transfer of data. Due to the concern of security, it also asks for authentication by the user before the transfer of data. It is associated with the **TCP protocol** and corresponds to two ports, port 20 and 21. Both of these ports function during the application layer.

Port 20 performs the task of forwarding and transferring of data. It takes over the task of transferring FTP data when it is in active mode.

Port 21 performs the task of signaling for FTP. It listens to all of the commands and provides a flow control for data. It is quite essential for maintaining the flow of data.

## TELNET - 23

TELNET port 23 comes under the category of **TCP Protocols**. Its main function is to establish a connection between a server and a remote computer. It establishes a connection once the authentication method has been approved. However, this port is not suitable to establish secure connections and does not cater to the concern of security. It enables the remote connection of a computer to be established with routers and switches as well. It makes use of a virtual terminal protocol to make a connection with the server. It comes into existence during the application layer of the TCP/IP protocol.

## IMAP - 143

IMAP is the abbreviation of 'Internet Message Access Protocol'. The IMAP -143 Port lies under the category of **TCP protocol**. The primary purpose of this port is to retrieve emails from a remote server without having the need to download the email. You have the liberty to access the emails from anywhere by connecting to the server and viewing your email after providing authentication. This opportunity has been provided to you because of the existence of this port. It reserves a virtual memory for the email which enables you to read it by connecting to the server. However, you may also download the mail if you wish to. It also provides you the ability to search for your messages from a bunch of them to get to your desired one. IMAP 143 Port generally operates at the Application Layer of a TCP/IP Model. In addition to this, it also makes sure that the data remain secure during this connection.

## SSH - 22

SSH is also referred to as 'Secure Shell'. It operates on the port number 22 of the **TCP protocol**. It carries out the task of remotely connecting to a remote server or host. It allows you to execute a number of commands and move your files remotely as well. However, it is one of the most secure ways of accessing your files remotely. Using this port, you can remotely connect to a computer and move your files with ease. This port sends the data over the network in an encrypted form which adds an extra layer of security on it. In addition to this, only authorized people will be able to remotely log on to their systems using the Port 22 which makes sure that the information does not get into unauthorized hands. It provides the chance to move files within networks as well as gives the privilege to move files between different networks securely. It operates at the Application Layer of the TCP/IP Model and is considered as one of the most secure and reliable ports for accessing files remotely.

## DNS – 53

## DNS query is initially sent with UDP by client, if no response from server in 5 sec, client will send TCP query.

## DNS response from DNS server is always in TCP.

DNS is referred to as 'Domain Name System'. It operates on the port 53 of **TCP and UDP protocols**. DNS makes use of relational databases to link the host names of the computers or networks to their respective IP Addresses. The port 53 waits for requests from **DHCP** to transfer the data over the network. It operates on the Application Layer of the TCP/IP Model.

*TCP protocol* is used by the **Zone Transfer** function of the **DNS server**. Once the connection is established, the zone data will be sent by the server using the **TCP 53 port**. However, when the query has to be transferred from **the client computer**, it will be sent using the port 53 on **UDP protocol**. However, if no response is received from the server within 5 seconds, the DNS query will be sent using the port 53 of TCP Protocol.

## DHCP - 67, 68

DHCP is also known as **'Dynamic Host Configuration Protocol'**. It basically runs on the **UDP protocol**. The basic purpose of DHCP is to **assign IP Address** related information to **the clients** on a network automatically. This information may comprise of subnet mask, IP Address etc. Many of the devices are automatically configured to look for IP Addresses using DHCP when they connect on a network. It makes it quite reliable to assign all the devices on a network with automatically produced IP Addresses. It generally operates on the Application layer of the TCP/IP Model. DHCP basically makes use of 2 ports; Port 67 and Port 68.

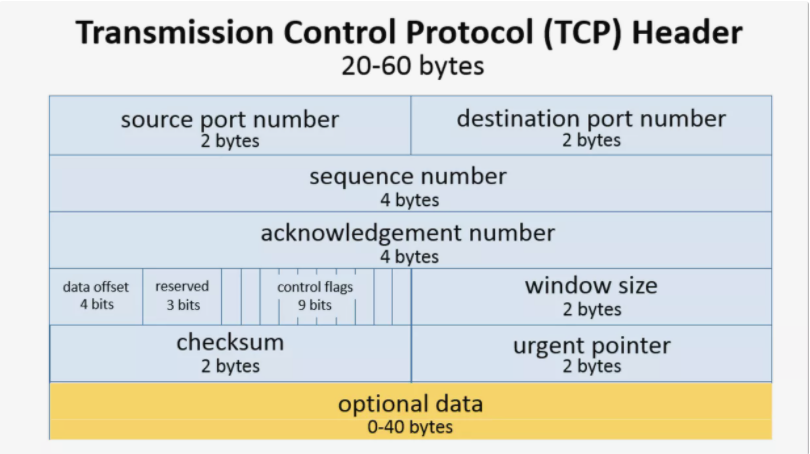
**UDP Port 67** performs the task of accepting address requests from DHCP and sending the data to the server. On the other hand, **UDP Port 68** performs the task of responding to all the requests of DHCP and forwarding data to the client.

**POP3-110**

POP3 is also referred to as Post Office Protocol Version 3. It operates on the port 110 of **TCP Protocol**. It allows the email messages to be retrieved from the SMTP servers. Using this port, you can download the messages from the server and then read them. However, this means that you will not be able to access the messages and read them without downloading them. Furthermore, the messages are also deleted from the server once they are downloaded. However, this port does not cater to the issue of security. The authentication details transferred over the network are not encrypted and sent in plain text. This means that any hacker can easily intercept this information and misuse it. Port 110 generally operates on the Application layer of the TCP/IP Model.

We have discussed some of the most common and widely used Ports above. We have seen how each of these ports are either related to the UDP protocol or TCP protocol and are used at the Transport or Application layer. All of these ports perform different tasks and different processes. While we have some ports where our data can be sent securely, there are some others where the transfer of data is of more significance than its security. We can also combine different protocols to add the feature of security. For example, SSL can be added to HTTPS port to add a feature of security to it. Considering the uses and applications of these ports, it is important to realize their significance in the process of transmission of data over a network. Not only do they help you to transfer data, they also let you enjoy some other facilities as well. In fact, it is not wrong to say that networking will not be complete without the existence of these TCP and UDP Ports.

The [Transmission Control Protocol (TCP)](https://www.lifewire.com/transmission-control-protocol-and-internet-protocol-816255) uses a set of communication channels called [ports](https://www.lifewire.com/port-numbers-on-computer-networks-817939) to manage system messaging among several different applications running on the same physical device. Unlike the physical ports on computers like [USB ports](https://www.lifewire.com/what-is-a-usb-port-818166) or [Ethernet ports](https://www.lifewire.com/what-is-an-ethernet-port-817546), TCP ports are virtual — programmable entries numbered between 0 and 65535.



1. **Domain Names**

[the anatomy of a URL](https://themeisle.com/blog/what-is-a-website-url/)

https://themeisle.com/

This URL contains two parts:

* **A protocol** (https**:**) is a set of guidelines that a browser follows to send a request to the server.
* The **domain**, themeisle.com, **or URL** to the main website. A domain consists of two parts:
  + the TLD (**top level domain**) which is the **.com** part (or [another domain extension](https://themeisle.com/blog/domain-extensions-guide/)), and
  + the SLD **(second level domain**), themeisle, the name that you buy from a [domain registrar](https://themeisle.com/blog/best-domain-registrars/).

A subdomain contains a second name before the SLD. For instance, if the ThemeIsle blog was hosted on https://**blog**.themeisle.com, the blog would be the **subdomain**.

A subdomain is commonly used to logically separate a website into sections. You can use a subdomain to launch a career site (careers.yoursite.com), a forum (forum.yoursite.com) or for customer support (support.yoursite.com). You may use subdomains to create blogs of different themes too. For instance, [sbnation.com](https://www.sbnation.com/) is a sports news blog. However, it uses blogs like [weaintgotnohistory.sbnation.com](https://weaintgotnohistory.sbnation.com/) and [theshortfuse.sbnation.com](https://theshortfuse.sbnation.com/) for specific teams on different subdomains

 Before you can set up any subdomain, you have to have a main domain. If you don’t have that taken care of yet, here are our guides on [choosing a domain name](https://themeisle.com/blog/how-to-choose-a-domain-name/) and [how to register a domain name](https://themeisle.com/blog/how-to-register-a-domain-name/).

Once you buy a domain, you also buy the rights for subdomains within it. Broadly, these are the steps that you must follow to create a subdomain:

* Come up with the name of the subdomain, enter it as a record in your DNS settings.
* Redirect to the server that hosts your subdomain.

When you are entering a record in your DNS settings, you would notice that www.yoursite.com points to yoursite.com. This essentially makes www a subdomain too!

<https://themeisle.com/blog/fix-broken-links-in-wordpress/>

1. The protocol – *HTTP* or *HTTPS*.
2. The domain name (including the TLD) that identifies a site.
3. The path leading to a specific web page.



Your **main domain** – also known as a primary domain or a root domain – is essentially the name of your website

In Brafton’s case, our **main domain** name is [**brafton.com**](http://brafton.com/). Notice I didn’t say [**www.brafton.com**](http://www.brafton.com/) or [**https://www.brafton.com**](https://www.brafton.com/), which are technically our site URLs.

A subdomain is a division of your website that you want to distinguish with its own unique identity and content. For instance, if Brafton wanted to create a subdomain for our blog page (we don’t), our subdomain name would be [**blog.brafton.com**](http://blog.brafton.com/).

**The structure and components of a URL**

A URL is usually made up of several parts. To understand the structure and the components, we will dismantle the following example URL:



General language use

If we talk about a URL, we usually mean the concrete path to a directory (http://www.domain.com/a-directory/) or a file (http://www.domain.com/documents/study.pdf) on a website.

Subdomain vs. Subfolder

Another common distinction you’ll need to make when it comes to subdomains is the difference between a subdomain and a subfolder.

By now, you already know what a subdomain is, but a subfolder is a bit different. With a subfolder, you’re adding a folder to your existing domain.

So, instead of creating a new subdomain for your blog like “blog.mysite.com,” you’ll use a subfolder instead “mysite.com/blog.”

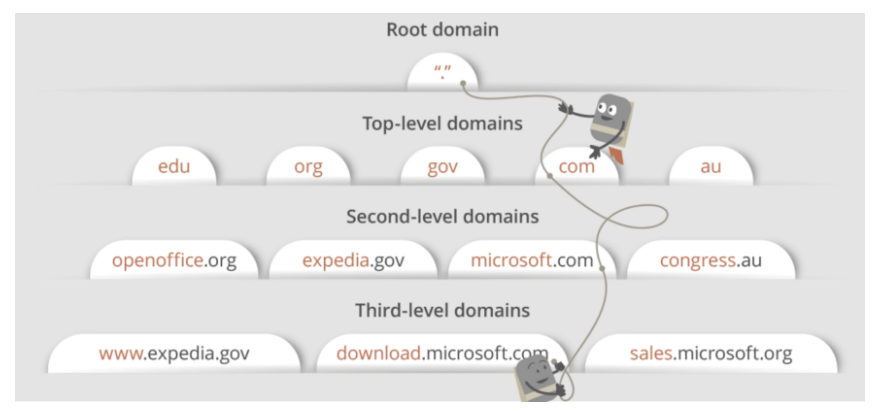
For example, here at HostGator you can access the blog by navigating to “https://www.hostgator.com/blog” , not “blog.hostgator.com”.

When you’re thinking about using a subdomain, you should really view it as creating a separate website. Although subdomains branch off of an existing domain name, they do take more work to build, grow, and maintain.

A subfolder is a way to organize your site more easily. Think of it as creating categories for your blog and blog posts. If you have a sports website, you could create subfolders for each sport you cover. So, you’d end up with a URL structure something like the following: “sports.com/basketball,” “sports.com/football,” “sports.com/hockey,” and on and on. Each page could operate as its own separate sports-specific blog with each page filled with unique content about that sport.

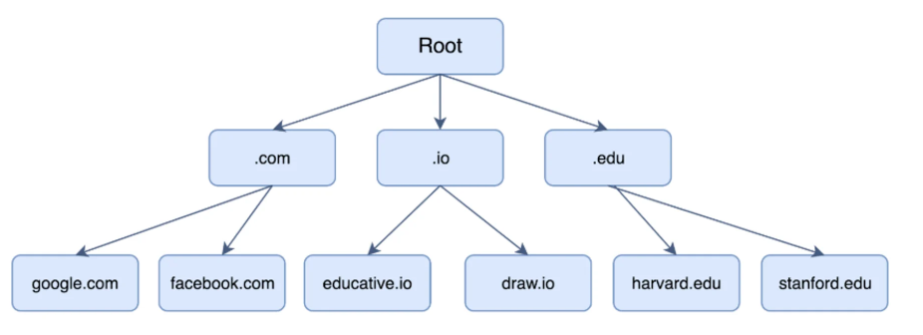
1. **What Happens When You Type in a URL**
2. You enter a URL into a web browser
3. The browser looks up the IP address for the domain name via **DNS**
4. The browser sends a HTTP *request* to the server
5. The server sends back a HTTP *response*
6. The browser begins rendering the HTML
7. The browser sends requests for additional objects embedded in HTML (images, css, JavaScript) and repeats steps 3-5.
8. Once the page is loaded, the browser sends further async requests as needed.
9. **DNS (Domain name system)**

The browser extracts the domain name from the URL.



## DNS lookup to find IP address

After hitting the URL, the first thing that needs to happen is to resolve IP address associated with the **domain name**.



There are 4 local cache to check :

1. The browser’s local cache is checked
2. The operating system’s cache is checked
3. The router is checked for the record.
4. Lastly, the query is sent to the Internet Service Provider (ISP) for it to check its cache.

Hence, if the record cannot be found locally, a full DNS resolution is conducted as follows:

1. The first point of contact for a full resolution is a root server. As of the writing of this post, 1017 instances of root servers exist.
2. The root server returns the IP address of the relevant top level domain server.
3. The top level domain returns the IP address of the second level domain server.
4. The second-level domain server contains the DNS record of the server we are looking for. The second-level domain server returns the IP address to the browser.

This is the overview, but there are **four layers** through which this domain name query goes through. Let’s understand the steps:

1. After hitting the URL, the **browser cache** is checked. As browser maintains its DNS records for some amount of time for the websites you have visited earlier. Hence, firstly, DNS query runs here to find the IP address associated with the domain name.

2. The second place where DNS query runs in **OS cache** followed by **router cache**.

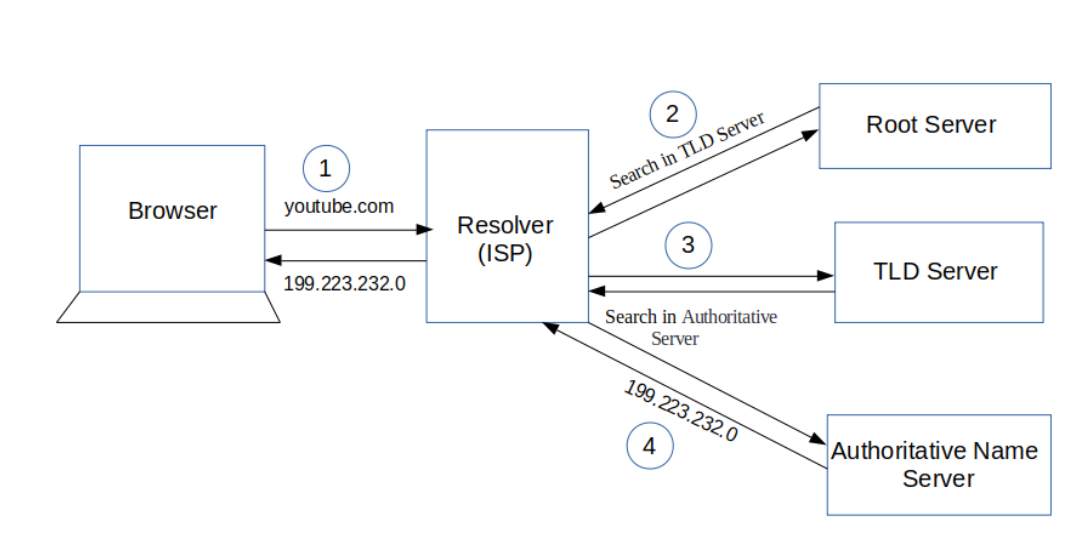
3. If in the above steps, a DNS query does not get resolved, then it takes the help of resolver server. **Resolver server** is nothing but your ISP (Internet service provider). The query is sent to ISP where DNS query runs in**ISP cache.**

4. If in 3rd steps as well, no results found, then request sends to **top or root server** of the DNS hierarchy. There it never happens that it says no results found, but actually it tells, from where this information you can get. If you are searching IP address of the top level domain (.com,.net,.Gov,. org). It tells the resolver server to search **TLD server** (Top level domain).

5. Now, resolver asks TLD server to give IP address of our domain name. TLD stores address information of domain name. It tells the resolver to ask it to **Authoritative Name server.**

6. The authoritative name server is responsible for knowing everything about the domain name. Finally, resolver (ISP) gets the IP address associated with the domain name and sends it back to the browser.

After getting an IP address, **resolver stores it in its cache** so that next time, if the same query comes then it does not have to go to all these steps again. It can now provide IP address from their cache.



## TCP connection initiates with the server by Browser

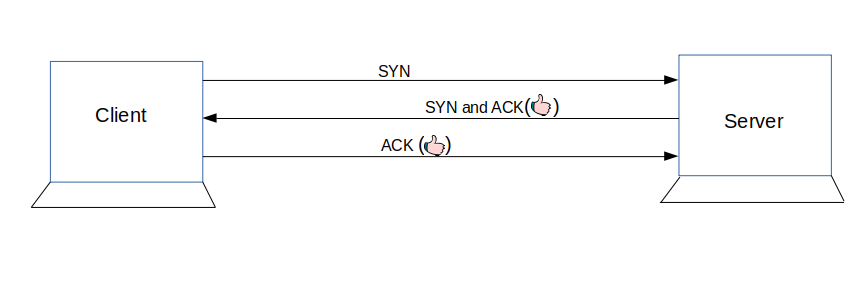
Once the **IP address** of the computer (where your website information is there) is**found**, it **initiates connection** with it. To communicate over the network,**internet protocol** is followed. **TCP/IP** is most common protocol. A connection is built between two using a process called **‘TCP 3-way handshake’**. Let’s understand the process in brief:

1. A client computer sends a **SYN message** means, whether second computer is open for new connection or not.

2. Then**another computer**, if open for new connection, it sends **acknowledge message** with SYN message as well.

3. After this, **first computer** receives its message and acknowledge by **sending** an**ACK message.**

To better  understand, look below diagram.



## What is HTTP?

These requests follow a ‘protocol’ or ‘rules of communication’ called **HyperText Transfer Protocol (HTTP)**. This protocol dictates the format of the messages, when what message is sent, appropriate responses, and how messages are interpreted. **HTTP messages** are of two types: **request** and **response**.

An HTTP **request message** consists of a **request line** and **headers**. The message starts with a request line and is followed by headers. Here’s a sample HTTP request:

GET /path/to/file/index.html HTTP/1.1  
  
Host: www.educative.io  
  
Connection: close  
  
User-agent: Mozilla/5.0  
  
Accept-language: fr  
  
Accept: text/html

The **request line** consists of a **request method**, a **path**, and the **HTTP version**.

The request method, GET, in the example above tells the server what to do. GET, for example, tells the server that the client wants to get the resource found at the given file path.

Other examples of request methods include DELETE, which tells the server to delete a resource at the given path, and PUT, which tells the server to put a supplied resource at the given path. The HTTP version is also specified to cater for the differences between each.

Next come the HTTP **headers**. Headers allow the client to communicate extra information such as the server type and the date. Each header is on a seperate line and contains a name and value, separated by a colon.

There are many headers which provide different functions. For example, the connection header indicates whether user is on a HTTP connection type.

The server then sends an HTTP **response message**. Here’s a sample response message:

HTTP/1.1 200 OK  
  
Connection: close  
  
Date: Tue, 18 Aug 2015 15: 44 : 04 GMT  
  
Server: Apache/2.2.3 (CentOS)  
  
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT   
  
Content-Length: 6821  
  
Content-Type: text/html  
   
[The object/file that was requested]

Response messages consist of a **status line** to start with, followed by a number of **headers**, followed by a blank line and ends with a resource if any was requested.

The **status line** consists of the **HTTP version** and a **status code**. There are a few types of status codes. A common example is the infamous 404 Not Found status code.