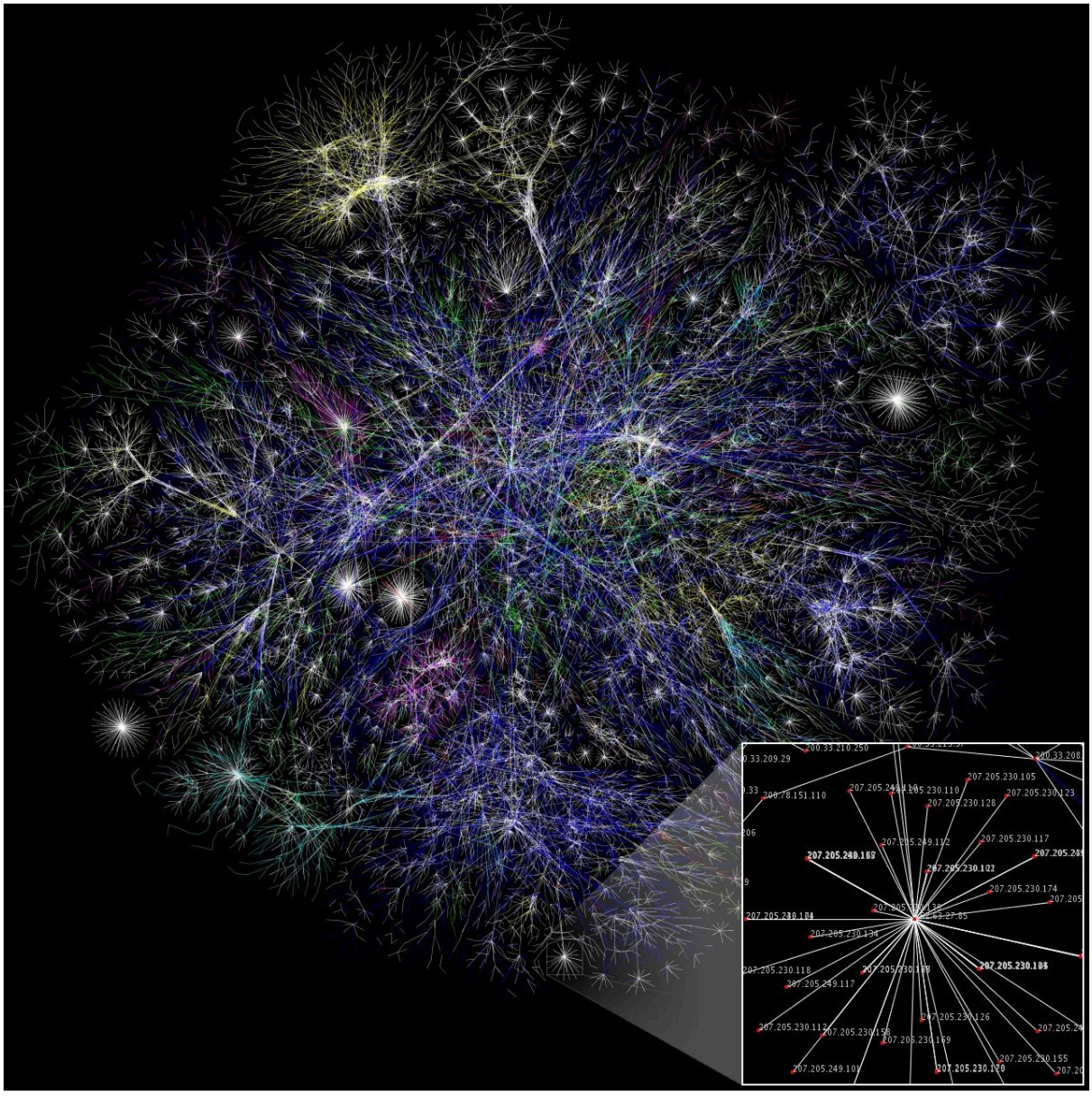
0x08. What happens when you type https://holbertonschool.com in your browser and press Enter

Internet map 1024 - Each line is drawn between two nodes, representing two IP addresses



The internet is a vast and wonderful place full of wonder and information that we use almost constantly in our daily lives. It is easy to take a resource like this for granted, especially for younger generations that have always enjoyed the benefits of this resource. In this blog post, we will take a look at some of what goes on behind the scenes when a Uniform Resource Locator (colloquially: web address) is typed into a client's browser and the user presses enter.

One question we need to answer before we get too deep is "What is a web browser?" A web browser is, simply put, a piece of software that is used for accessing websites. A browser fetches content, either locally or from the internet, and displays the web page on the client's screen. For now, this definition will suffice.

Let's start by breaking down the format of the URL <https://www.holbertonschools.com>

The URL starts with the Uniform Resource Identifier scheme <https>. https denotes HTTP connections secured using SSL/TLS. SSL/TLS stands for Secure Sockets Layer/Transport Layer Security. SSL/TLS is an encryption protocol used to secure communications over the internet via a digital signature on requests and responses.

The second section of the URL is the domain <holbertonschools.com>

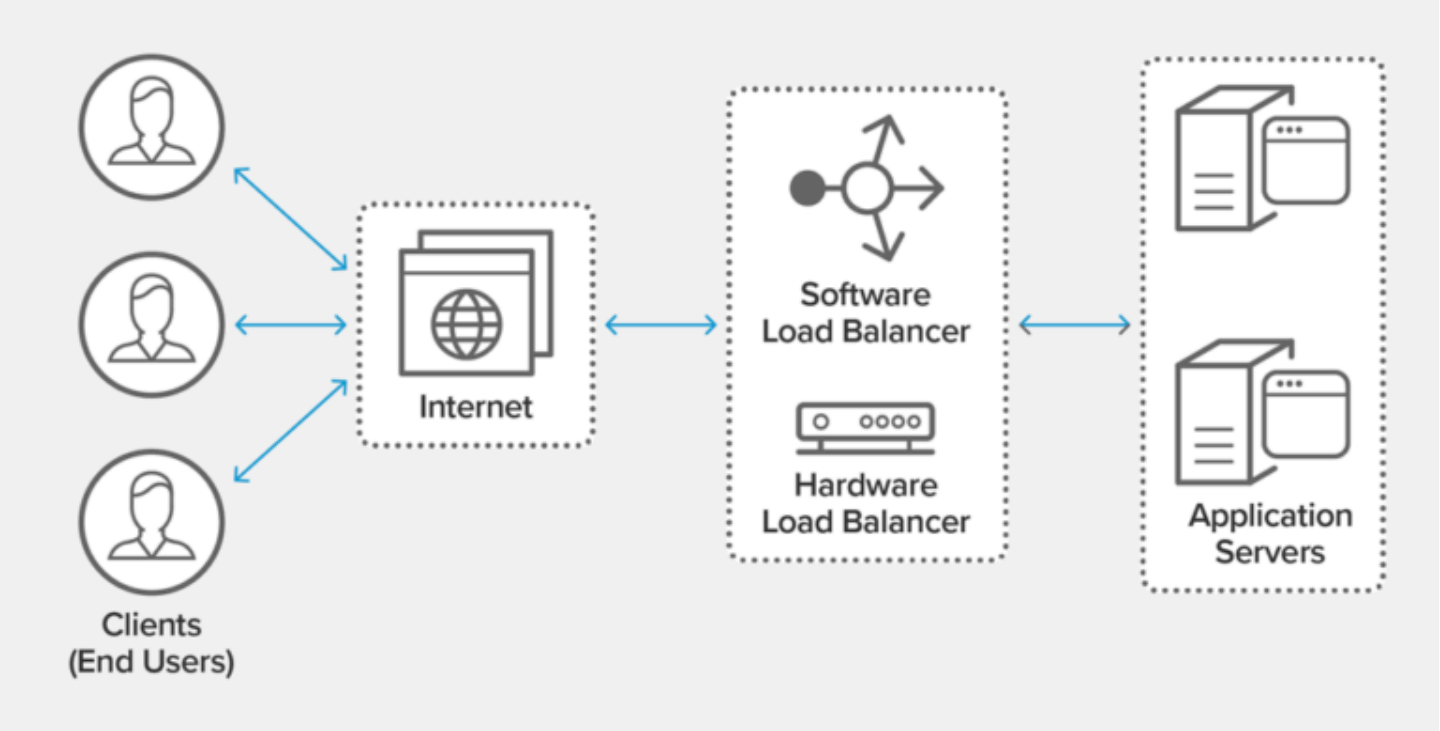
A domain name is an "easy for people to remember" word, or set of words, that is paired with an IP address. The first thing that needs to be done is to convert the domain name into an IP address. The browser accomplishes this by creating a DNS request. A DNS request searches the Domain Name System for the text representation of a domain, and returns the IP address. This is similar to how we used to use a phone book to find a phone number or home address when we knew someone's name. Along this line, we can think of an IP address similar to how we think of a home or business address. An IP address is where a website "lives" on the world wide web. Most people are familiar with IPv4, which is a 32-bit number． IPv4 limits the number of available addresses to 2^{32} (4,294,967,296). A new version, IPv6 was standardized in 1998. IPv6 increased the size of an IP address from 32 bits to 128 bits (3.403×10^{38} unique addresses). In our example, the IP address of holbertonschools.com is `99.83.190.102`.

The other part of internet communication protocols we need to discuss is TCP (Transmission Control Protocol). TCP provides a way to deliver and receive information packets over a network connection. The information packets that travel via TCP arrive ordered and error-checked. If we think of the IP addresses of two devices on the internet, we can think of TCP as the certified mail system. All packages are numbered so we can ensure they were delivered correctly, and free from corruption.

While SSL/TLS is a tool used to encrypt communications over the internet, we also need to discuss the use of firewalls as a network security system. Firewalls monitor and control incoming and outgoing network traffic. Firewalls provide isolation for our trusted network (home/work) and an untrusted network (internet). The earliest type of network firewall was a packet filter. Packet filters inspected packets as they were transferred between computers. The firewall maintained an access

control list that dictated what packets will be looked at, and what action should be applied upon inspection.

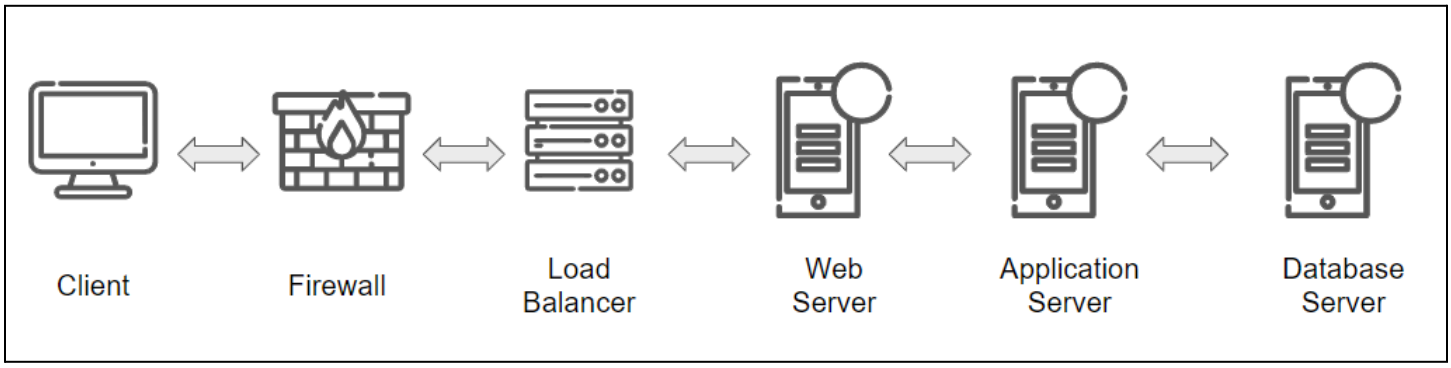
Now that we've reached a point where our browser has successfully identified the TCP/IP protocols of the website we want to access thanks to the DNS request and are allowed to communicate with this website via HTTPS through any firewalls, let's move on to how files are retrieved to be sent back to our browser.



load balancing diagram from nginx.com/resources/glossary/load-balancing/

The first device we are likely to encounter is a specialized application server called a load balancer. This application server serves a couple of purposes. First, a load balancer ensures that if there are multiple web servers hosted by an organization, no individual server becomes overloaded. There are different algorithms to facilitate this. One algorithm is Round Robin balancing. In this format, requests are distributed across a group of servers in sequential order. Another algorithm is Least Connection. In this scenario, each new request is sent to the server with the fewest number of current connections to clients. The relative capacity of each server is also factored in this decision. For example, if one server has 10 times the capacity when compared to a second server, the first 10 requests could go to the first server, while the 11th request goes to the second. Another benefit of a load balancer is redundancy. A load balancer can keep an eye on its application and web servers and if one goes down, redirect traffic to a properly working server to isolate the client from seeing any downtime.

The load balancer will send traffic along to a web server, which in turn may communicate with an application server and/or database server. The image below shows a simplification of this communication scheme.



A web server serves static web pages to clients via HTTPS requests. For example, a web server is meant to serve HTML and CSS files back to the browser. Some common tasks of a web server are to listen to client connections and requests and manage client connections. A web server receives client requests by reading HTTPS messages, executes or refuses the requested HTTPS method, and replies to the client request.

An application server, on the other hand, provides more dynamic information. This may include an API, object life cycle management, and/or resource management. Most application servers have a web server as an integral part of them. In other words, an application server can do anything a web server can do, but a web server can't necessarily do everything an application server can do.

The last type of server to discuss is a database server. A database server is a server which uses a database application to provide database services to other computers or computer programs. A database server is designed to handle database queries.

Now that we have a basic understanding of each connection in the process, let's take a look at how information is actually sent from client to server and back.

This process begins with the client's browser sending a request to the server. A client request consists of three parts: a request line, headers, and a body.

There are different types of HTTP requests methods that are included in the request line. Here are the definitions of some, but not all, methods as given by MDN Web Docs:

- GET - requests a representation of the specified resource. Requests using GET should only retrieve data (like going to a library to get a book)
- POST - submits an entity o the specified resource, often causing a change in stat or side effects on the server (like adding an item to a grocery list)
- PUT - replaces all current representations of the target resource with the request payload (like replacing last week's grocery list with a brand new one for this week)
- DELETE - deletes the specified resource (like throwing away the grocery list)

Along with the request method, the request line will also include a path to the resource it is looking for, and the HTTP version for communication. Here is an example request line "GET /hello HTTP/1.1".

In this case, the client's browser is telling the server it wants to get a resource at /hello and communicate with HTTP/1.1

The next part of the request is the header. Headers send extra information to the server to help process the request. Headers consist of key-value pairs.

The last part of the request is the body. For a GET request, the body will be empty, but for a POST request it would contain the information we want to add to the resource, or a DELETE request would contain the information we want to remove from the resource.

When the client initiates a request, the server replies with a response code. We've all seen 404 - Not Found in some way or another before. This is the server replying by saying "I couldn't find what you asked me to look for". Another response, the one we hope to get is 200 - OK. This means everything went according to plan.

If all has gone according to plan, the server has responded with a 200 - OK response, and sent back all of the information and data the client's browser needs to render the webpage in HTML format.

Let's review the steps that happened to get the client's webpage 'https://holbertonschool.com' to render in their browser.

1. The user types the URL https://holbertonschool.com into the address bar of the browser of their choice and presses enter.
    a. The URL consists of two parts:
        i. Scheme: https - hypertext transfer protocol secured with SSL/TLS
        ii. Domain: holbertonschool.com - text that is tied to an IP address
2. The browser searches the DNS for the IP address that matches the text, in this case `99.83.190.102`
3. The browser initiates a TCP connection with the server
    a. Most of the time this is a special application server that is a load balancer
        i. The load balancer will connect to the appropriate web server.
        ii. The web server will use it's associated application and database servers to find all of the data it needs
4. The browser sends an HTTP request to the server
    a. The HTTP request consists of a request line, header, and body
5. The server sees the request and responds appropriately
    a. The server responds with a 200 - OK and has built the HTML to send back to the client
6. Finally, the client's browser receives the HTML file from the server and renders the web page.