How the Web Functions

Dmitri Brengauz

August 4, 2018

Abstract

This essay gives a brief introduction to what goes on “behind the scenes” when a user requests a web page in a browser. Specifically, it answer the following questions:

1. When we hit `https://www.techtonicgroup.com/` what happens? Dont focus too much on architecture (Monolithic, SOA, Microservices, etc.). Try to focus more on how the web functions.
2. From start to finish, how does that data reach you to be rendered in the browser? - DNS/TCP/IP
3. What code is rendered in the browser
4. What is the server-side code’s main function?
5. How many instances of the client-side assets (HTML, CSS, JS, Images, etc.) are created?
6. How many instances of the server side code are available at any given time?
7. What is runtime?
8. How many instances of the databases connected to the server application are created?

1 Introduction

With all the magic surrounding the World Wide Web, it's easy to lose sight of what it is and was designed to be: a document retrieval system. Behind the scenes, the web browser, the Internet, and the server should work seamlessly together to retrieve and compose that document for viewing without the user having to worry about how any of that is done.

2 Down the Stack, Accross the Internet, Up the Stack

To send and recieve data, the web browser itself forms the top layer of the Internet Protocol Suite. This is a stack of appplications, organized together into four abstraction layers, that provides a uniform interface for all of the programs on a computer to communicate over the Internet. Each layer depends on the layer below to fulfil it's function, but has no need to know how that function is achieved.

In this example, the web browser forms the Application Layer. It requires a data connection to send and recieve messages from the computer where information is located, the server. The browser uses another application level program, the Domain Name Service, DNS, to translate the human-readable address (like $SITE) into a numerical address (IP address) that is used by the rest of the stack to find a route to the server send data back and forth across it.

The web browser turns to the layer below it on the stack, the Transport Layer. Here, the application has a choice: does it need a continuous session and verification that each bit of data arrived in the correct order, or can the remote computer just broadcast a more efficient stream of data, with the occasional bits of data being lost? A streaming video program would chose the latter option, and use the UDP protocol in the Transport Layer. However, the protocol that the browser uses to talk to the server, HTTP/1.1, requires a stable connection with error checking, and thus the web browser makes the first choice, and sends its message using the Transmisson Control Protocol, or TCP.

The job of TCP is to establish that the machine at the other end is ready to recieve and transmit data (establish a *session*) divide the data sent to it into chunks called segments. TCP adds a sequence number to each segment in a *header*, which is sort of like a data envelope. These sequence numbers make sure the data stays in the correct order, no matter in which order the segments were received by the remote machine. TCP is also in charge of requesting that missing segments be re-transmitted, and transmitting segments that have failed to arrive. TCP itself doesn't know how to send and receive segments from the network, however. For that, it relies on the layer below it, the Internet Layer.

The Internet layer, using, IP or Internet Protocol, bundles TCP segments into network packets. These have their own header, containing the destination Internet address, the source (return) address, as well as a checksum to check the payload—the TCP created segment–for corruption.

IP knows the address of the machine on its own network known as the gateway, which routes packets to the outside world, the Internet. It uses these IP addresses to instruct the final layer, the Link layer, where to physically transmit the packets.

The Link layer is what communicates with the hardware. It allows the transmission of IP network packets over copper Ethernet wires, or electromagnetic waves though the air—WiFi signals. Once the packets make it to the gateway, the Link layer there sends the packets back up to the IP layer, which examines them, and using routing protocols determines where they should be sent next.

Packets hop across networks like this until they reach their destination. There, the IP layer sends them back up to the Transport layer, which checks the data for completeness and puts it in order, and finally delivers it to the application in charge of handling web browser requests, the web server.

3 Server Side

The Application Level program that handles requests from web browsers is called a web server. The handling of requests Modern web servers can handle tens of thousands of simultaneous connections, depending on the memory and network bandwidth available, although sites that expect high traffic use a device that distributes traffic across multiple web servers called a load balancer. The content that a web server will send back to a browser can be separated into two types: static and dynamic.

Static content can be delivered to the user's browser exactly as stored on the server. If it has to be generated or processed, then it is called dynamic. A web server can read static content directly from the memory cache, or permanent storage if no cache exists.

Requests that require processing need to use a program that interprets a set of instructions as to how this processing should be handled. The Apache web server includes such programs internally as modules. By contrast the Nginx server calls on an external application server like the JavaScript runtime environment, Node.js, or Phusion Passenger for Ruby on Rails. Regardless of implementation, the code run on a server to send back a web response to the client from the server is called server-side scripting, or just "server side."

A major part of server-side scripting is interacting with a database to retrieve or edit structured data. The term "database" refers to a set of physical files, while an "instance" in an Oracle environment refers to the combination of software and memory used to access that data. In SQL an instance is a specific installation of SQL Server. Multiple SQL Server instances are used to manage server resources as each instance is allocated a specific amount of memory and CPU resources.

Each database instance can handle multiple web server connections, with one connection typically being necessary per request, although high-traffic databases use a technique to accommodate multiple requests per connection known as pooling.