A DEFINITIVE AND COMPREHENSIVE GUIDE

FRONT-END ENGINEER'S GUIDE TO COMPUTER NETWORKS

WHAT EVERY FRONT-END ENGINEER NEED TO KNOW ABOUT COMPUTER NETWORKS

Front-End Engineer's Guide to Computer Networks

Version 1.0.0

Orkhan Huseynli

November 2024

Introduction

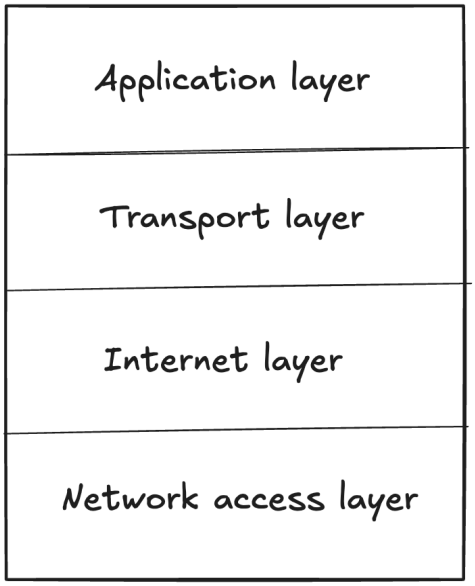
Understanding the basics of computer networks is important for front-end engineers who want to create web applications that are fast, secure, and reliable. Every time a user visits a website, there is a complex process happening behind the scenes. Data travels across the internet, passing through different layers and following rules set by protocols like TCP and UDP. Knowing how this works can help engineers improve the performance of their applications and solve problems when things go wrong, such as slow loading times or broken connections.

It's also important for front-end engineers to understand protocols like HTTP and WebSocket, which they use every day. These protocols manage how data is sent and received in web applications. Learning about security features like TLS and understanding how DNS works to resolve domain names can make front-end engineers utilize browser's features to make their website faster. This guide explains these concepts in a simple way, connecting theory to practical examples that will help engineers make better choices in their work. Whether it's optimizing how data is loaded or ensuring real-time communication features work smoothly, this book provides the essential knowledge front-end engineers need.

TCP/IP Model

Let's say we have two computers and in a hypothetical world, we want them to send and receive data between them. The first thing that we need is a **transmission medium**; a channel where data can flow. It can be a copper wire, fiber-optic cable or even air. Each of the computers need to have a **network interface** to send and receive signals. In order to properly communicate both of the interfaces need to agree on a set of rules. We call those sets of rules a **protocol**. A very basic protocol would be how each computer encodes and decodes electrical signals. If they don't agree on a rule, the same electric signal might mean different things for each device.

We're going to talk about a handful of protocols, so it might be a good idea to separate different protocols into abstraction layers and look at each layer independently. Luckily such abstraction layers already exist and to learn computer networks, all we need to do is study those layers independently. Introducing the **TCP/IP model:**



TCP/IP model

You might have heard about the **OSI model** which has more layers (7 to be specific), but for the sake of learning TCP/IP model is more than enough for me. As a front-end engineer you might have heard people saying “layer 7 proxy” or “layer 3 proxy” so this has something to do with which layer the proxy operates on. If you don’t know what a proxy is, don’t worry you’ll understand soon enough.

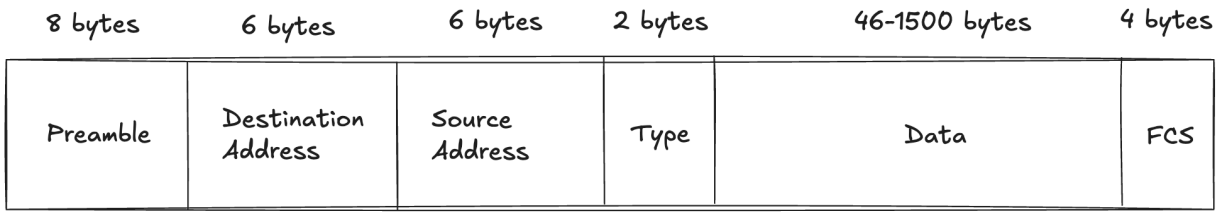
TCP/IP model is simply an abstraction to easily understand what happens on each layer, without being forced to understand everything at once. Each layer builds on top of the bottom layer. For example, the network access layer deals with the transmission medium whether it is an air hosting radio waves or a copper wire, so when we’re on the application layer we don’t have to worry about it. Talking about the network access layer, let’s dive into it in the next section.

Network access layer

As we mentioned before, each device has an interface to send and receive data over the network. This is a small piece of hardware called **Network Interface Card** (NIC), and each of them has a unique identifier called a **MAC address** (Medium Access Control). MAC addresses are primarily assigned by device manufacturers, and are therefore often referred to as the burned-in address, or as hardware address, or physical address. MAC address is simply a 48 bit number represented as six pairs of hexadecimals.

The first half of the MAC address is unique to each manufacturer and assigned by the Institute of Electrical and Electronics Engineers, or IEEE. There are some MAC addresses that do not belong to anyone but are used for special purposes. For example FF:FF:FF:FF:FF address is called broadcast address. When your phone isn't connected to WiFi it broadcasts to FF:FF:FF:FF:FF address that it is looking for a WiFi access point. WiFi access points that are discoverable will respond to this message so that you can connect it.

When a device wants to send a message to another device it composes a special packet called **ethernet frame** and sends it over the transmission medium. Ethernet frame is a simple packet that contains sender's MAC address, receiver's MAC address, the actual data and additional information such as checksum calculated to verify integrity of the package.



Ethernet frame

Let's go over the frame's fields to understand what they are. Preamble is just alternating 0s and 1s to allow the receiver's clock to synchronize with the sender. After that we have 6 bytes destination and 6 bytes source MAC addresses which denote who sent the message and who is the receiver. Type field is not important for our purposes, but depending on the value it might show the length of the payload. The data field contains the actual data and at the end we have 4 bytes of **Frame Check Sequence** which is an error detection mechanism to see if data is lost or tampered with during transmission.

In a nutshell, if a device wants to send another device a message, it would simply put it into an ethernet frame and send it. The receiving side would understand how the frame is structured and interpret it as such.

What happens in the network access layer is basically a Point-to-Point Protocol. Network switches are devices that usually operate on the network access layer. If you

send a message to your friend who lives in another country, there are a whole lot of things that happen. One of which is **routing** a package to the correct location. Let's look at routing in the next section.

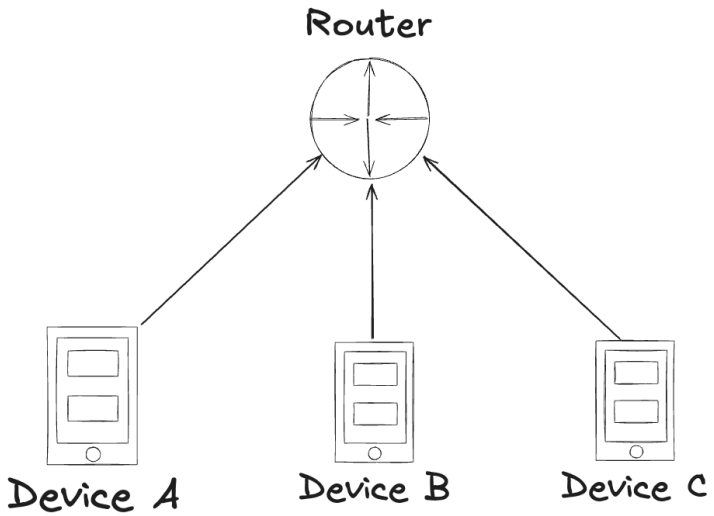
Internet layer

Let's suppose that you're sending a message from your phone to your friend who is living in another country. Clearly, you two are not connected directly, so a Point-to-Point Protocol is not very much useful here. We want our message to be routed across the internet to where it needs to go. If you're chatting on WhatsApp, your message needs to find WhatsApp servers and from there it would go to your friend's device. Keep in mind that we're now one layer above, so we don't need to worry about the things that happen at the network access layer. We assume that they just happen to happen. Let's look at how a message is routed on the internet.

IP address

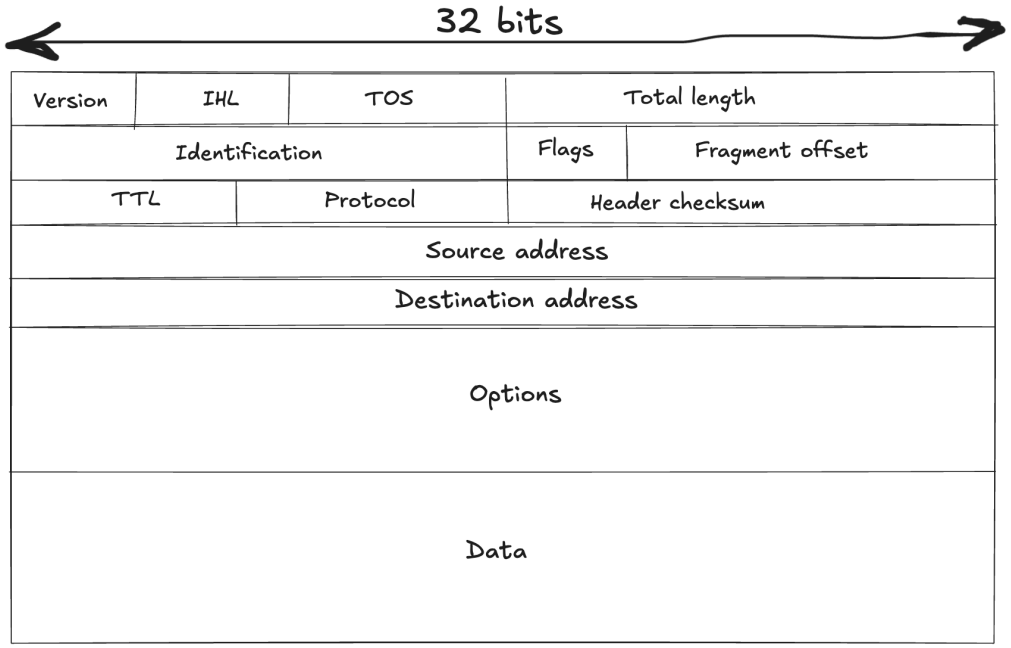
When a device is connected to a network, it is assigned a **network address**, called IP address. IP has several versions (e.g. v4 and v6), but for simplicity, we'll talk about IPv4. In this case the IP address is a **32 bit number** and using this number it is easy to route a message to where it needs to go.

IPv4 addresses are usually written as 4 decimals separated by a dot. Since each number between dots is 8 bits (8 x 4 = 32), each decimal will be between 0 and 255. For example, **192.168.1.1** is a private IP address. You've probably seen it when you open your WiFi Modem's home page on your browser.



Devices connected via a router

Keep in mind that if you are not connected to a network, you don't get an IP address, that's why it's called a network address. If we are connected to a **Local Area Network (LAN)** and we get assigned an IP address. If device A wants to send a message to device B, it needs to compose an **IP packet** and send it to the router. After that, it's the router's job to send it to the recipient. Here is what an IP packet looks like:

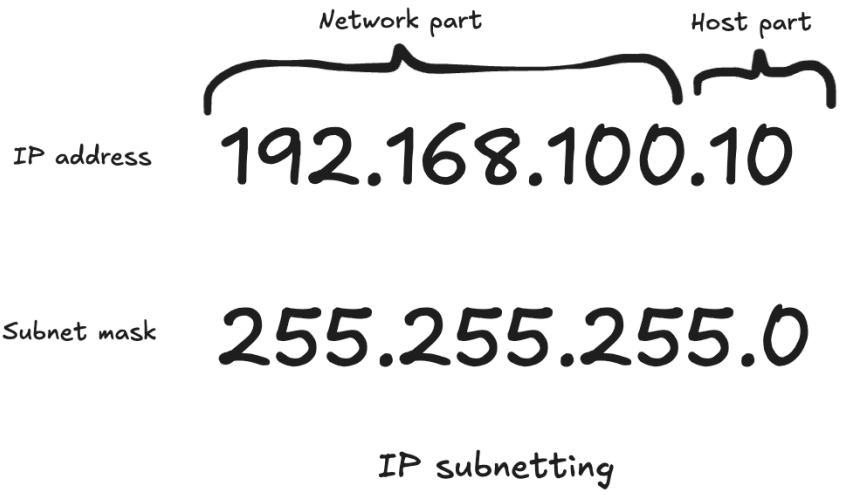


IP packet

There're plenty of fields inside an IP packet, but let's talk about some of the important ones. You don't need to know all of them and you can read detailly about them on the internet if you're curious. As you might have already guessed, destination and source addresses are IP addresses of sender and receiver. And the data is the actual data that is going to be sent. The other important field here is the **TTL (Time To Live)** field which is often used as **hop count**. When a router receives an IP packet, it decrements the hop count by one and this way avoids the routing loop. Routing loop happens when a packet's destination cannot be found, so it keeps looping inside a bunch of routers.

Subnetting

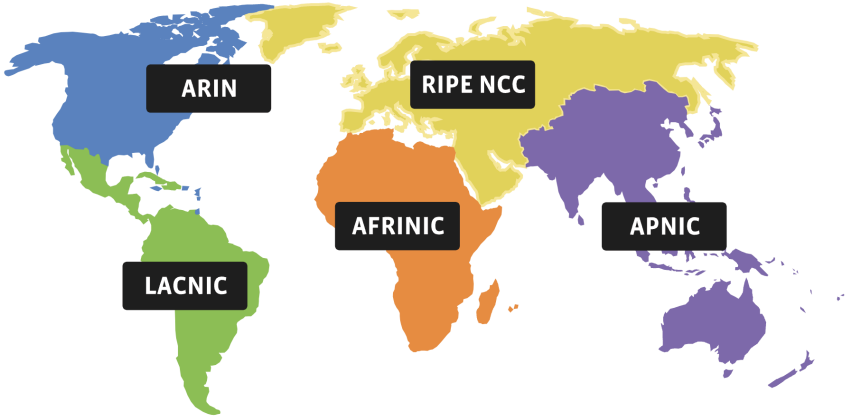
Routers use subnetting to determine where the packet needs to go. Subnetting is simply dividing the IP address into two parts: **network part** and **host part**.



IP subnetting

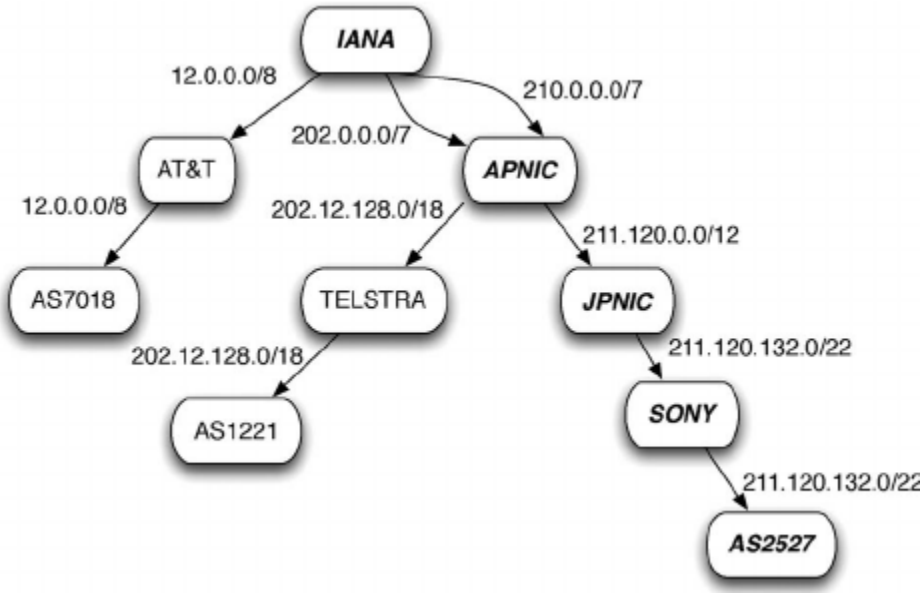
A router uses a subnet mask to determine which is the network part of and which is the host part. If the router sees that destination IP address is inside the local network, it will send the packet to the proper device. If the router sees that the destination IP address is not inside this network it will send it to a **default gateway**. The default gateway is another router who will do the same to determine where the packet needs to go.

Routing



Internet Assigned Numbers Authority (IANA) is an organization who controls which region takes control of which IP prefixes. You can see these regions in the picture above. Those regions in itself are also divided into few regions leaving a small number of

bits for host addresses. For example, **192.168.100.10/24** means, the first 24 bits determine the network address, then the last 8 bits determine the host address. It means there can be only 2^8 host machines on this network. You can see an example hierarchy in the following picture:



To simplify subnetting and routing let's look at an example. Let's say you're a postman and you received a packet. You know who sent this message and you see the destination address. Looking at the prefix of destination address (network portion) you decide if the recipient is in your neighborhood. If not, you give this packet to some other post office of your current state. looking at the prefix of the address, they determine if this address is inside this state. If not, they redirect it to the post office of the country. If the address is inside this country the packet will be sent to the proper state's post office. If not... Well, you got the idea.

Non-routable addresses

Since, IPv4 is only 32 bits long and host addresses are limited, there are some special IP addresses called **non-routable addresses** used for only inside private networks. If you wondered why at every house there is 192.168.1.1 address, it's because this is in range of non-routable addresses.

There are three main ranges of non-routable IP addresses specified by the **Internet Engineering Task Force (IETF)**:

10.0.0.0 – 10.255.255.255 (10.0.0.0/8)
172.16.0.0 – 172.31.255.255 (172.16.0.0/12)
192.168.0.0 – 192.168.255.255 (192.168.0.0/16)

Non-routable IP addresses

Routers usually use **Network Address Translation (NAT)** to convert between private and public IP addresses. This is out of scope of this book but if you're curious, I recommend checking out Hussein Nasser's video on NAT. Your private IP address is only visible to your router, and anyone in the public knows you by your **public IP address**.

Now let's say the IP packet found its way to the correct device. Inside a device there're lots of applications that send and receive messages over the network. For example, on your laptop you might chat on WhatsApp desktop, visit Twitter using Google Chrome and be inside a Zoom meeting. How do we know to which application the data belongs? The answer is using a **port number**. Let's look at the transport layer in the next section to learn the details.