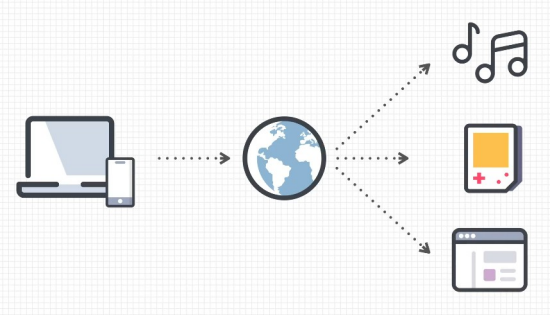
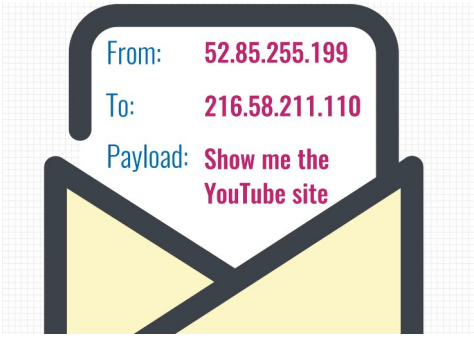
DNS: The Internet's Phonebook [1]

1. When you're connected to the Internet and you are listening to music, playing online games or browsing the web, there are many standard protocols running behind the scenes to make sure that your computer can indeed communicate with these services. There is for instance the ***Internet Protocol***, or IP, that is responsible for delivering messages from one computer to another. These messages are also called packets and, just like postal packages, they always have a sender, a receiver and a payload.



2. But instead of using street names and postal codes, the Internet uses ***IP addresses***, and they look something like this. Without them, computers can't communicate with each other, which means no computer networks and no Internet. So if you open up *YouTube*, your computer actually sends a message to *YouTube*'s IP address, asking for the data that it needs to show the webpage. "But, wait a minute", you say. "When I want to open *YouTube*, I just enter `youtube.com` into my browser, I don't need the IP address. So, how does my computer know the address then?"



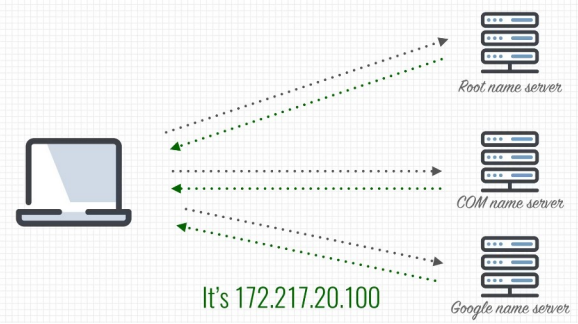
3. Well, I'm glad you asked! This issue was solved in the early days of the Internet when the *Stanford Research Institute* created a text file called **`hosts.txt`**. This file translated names like `youtube.com` into IP addresses that computers need to communicate. The file was then installed on all the computers connected to the ARPANET, so they could *translate* a domain name into an IP address. However, as the Internet kept growing, so did the demand for these domain names.

4. The small team that was managing the hosts file was quickly overwhelmed, and so in 1983, a specification was published to automate this task, and DNS or the ***Domain Name System*** was born. DNS is basically a big phonebook for the Internet, matching domain names like `google.com` to an IP address. This phonebook is hosted on DNS servers that are distributed across the world.

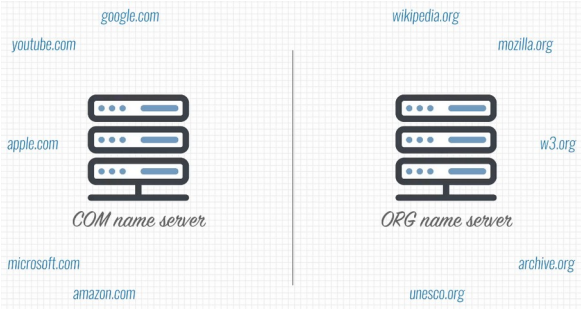
5. So let's take a look at what happens when you want to go to `www.google.com`. For starters, your operating system splits the domain name into multiple parts, or also called ***labels***, that are separated by dots. So in this case we have three labels: `www`, `google` and `com`. They create a hierarchy that has to be read from right to left. The right-most label is called the top-level domain, in this case `com`. We can then say that `google` is a subdomain of `com` and `www` is a subdomain of `google`. To resolve the IP address of `google.com`, your computer reaches out to a root name server and asks the question: "What's the IP address for `www.google.com`?"

1 https://www.youtube.com/watch?v=FJYa6C-MXno

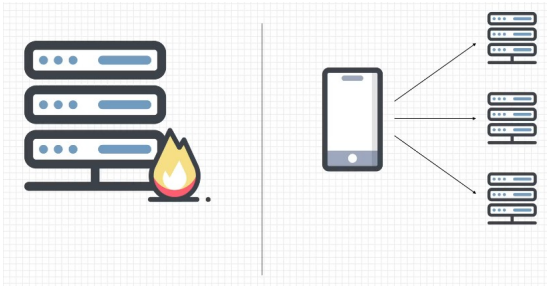
6. ***Root name servers*** never give a direct answer, instead they refer you to a server that is more likely to be able to help you. In this case, that'll be the name server in charge of the `.com` top-level domain. Your computer now asks the same question to the `com` name server, and this one is likely to refer you to yet another name server. In this case, it will redirect you to a name server that *Google* itself hosts. This one is very likely to be able to tell what IP address is connected to `www.google.com`.



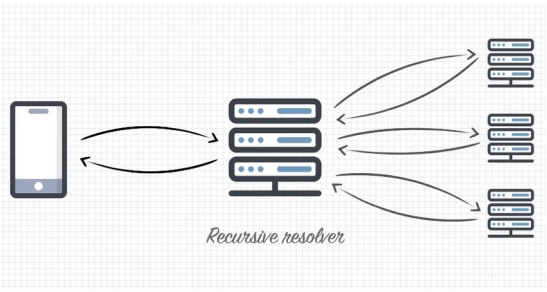
7. This mechanism, along with the hierarchy of domain names, makes DNS ***very scalable***, because after all, each name server only stores a small set of IP addresses. The `com` name servers don't know anything about websites that are hosted on the `org` domain for instance. The phonebook analogy holds up here as well: my phone number is listed in the Belgian phone book but not in the US one.



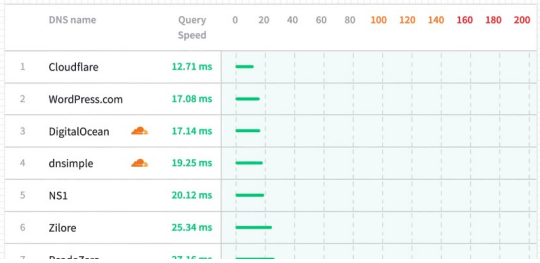
8. But there are however ***two drawbacks*** to this:
   - first of all, it puts a lot of pressure on the root name servers, as they will be contacted every time someone wants to connect to a website;
   - and secondly, devices need to be able to follow a referral, which it will get from root servers and perhaps other name servers as well.



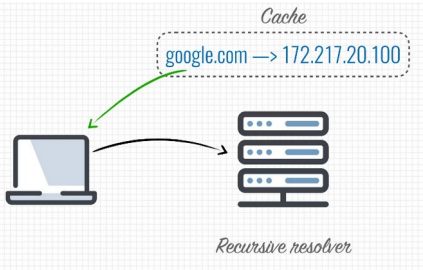
9. Both of these problems are solved by ***recursive resolvers***. These are special DNS servers that will take care of the entire resolving process. Instead of having your devices contact multiple name servers, they just contact a recursive resolver which does it all for them. They are often hosted by Internet service providers and more recently are also hosted by companies like *Google* and *Cloudflare*. Most home routers pull double duty and serve as a recursive resolver as well.



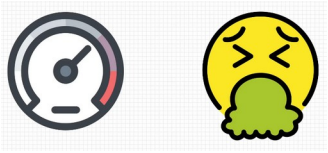
10. So how do our devices know what resolver to use? Well, by default, they will use the one that is configured by the network administrator. In a home network, that is your ISP and they'll likely configure their own resolvers, but you can always choose another one. Some recursive resolvers are ***faster than others***, so switching to a resolver hosted by *Google* or *Cloudflare* could give you a slight speed bump.



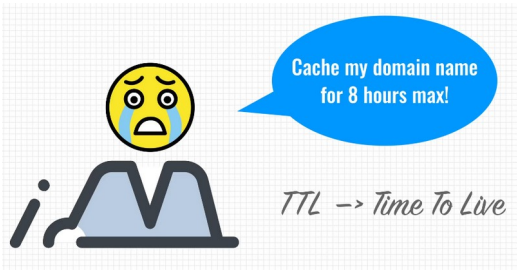
11. To speed up DNS even further, recursive resolvers also have a ***cache***, which stores the IP address of domain names that are most frequently being requested. When you go to `google.com` on your phone, the recursive resolver on your router will look up *Google*'s IP address, and once it figures that out, stores it in its cache for future reference. If another device on your network wants to resolve `google.com` as well, your router can instantly give an answer, without having to go through all the hoops of contacting multiple name servers.



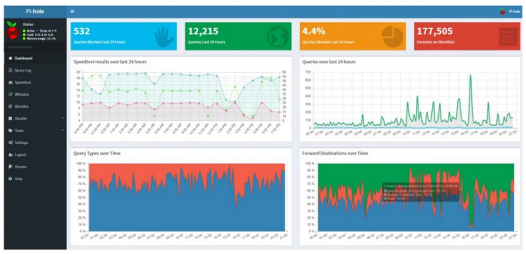
12. Caching can greatly increase the speed of DNS queries, but it can also be ***poisonous***. Changes to the IP addresses of domain names aren't immediately reflected across the world, because the old address is still stored in the cache of many recursive resolvers.



13. To combat this issue, domain owners can define how long an IP address may be cached. This is called the TTL or ***time-to-live*** and is expressed in seconds. If a cache record is older than the given TTL, the resolver must delete it, and after which it has to use the traditional resolving process again. But some recursive resolvers don't adhere to this TTL and keep records in their cache for a longer period of time to reduce the load. This practice is problematic for website owners that want to change the IP address linked to their domain names. But that's a minor hiccup.



14. It's clear that DNS is a major cornerstone of the way we use the Internet today. It also has some cool alternative use cases. You can for instance use a custom DNS server to block advertisements or to protect yourself from domain names that spread malware. Sounds complicated but you can easily do it by installing ***Pi-Hole*** on a *Raspberry Pi* and plugging it into your home network. *Pi-Hole* acts as a recursive DNS resolver for all your devices, and when a device wants to resolve `ads.google.com` for instance, the *Pi-Hole* will return a local IP address and essentially stop the advertisement from loading. Genius!



15. So that's a quick overview of the Domain Name System. It's a very ***open system*** and undeniably a protocol that makes the web accessible and easy to use for everyone: replacing hard to remember IP addresses with easy to remember domain names. That was it for this video! Let me know what you thought about it in the comments below. If you liked it, give it a thumbs-up and consider getting subscribed to my channel. Thank you so much for watching and till next time!