6.033 Lecture 13
Distibuted Naming & DNS

Americo De Filippo

June 19, 2023

References

[1] Saltzer, Jerome H. and M. Frans Kaashoek. Principles of Computer System Design: An Introduction (2009): **Section: 4.4**

The DNS (domain name system) is a bridge that covers the topic of networking (that we talked about) and tolerant reliable computing that we are gonna talk about.

1 The DNS

At the network layer the IP address (that are needed for naming the endpoint of a connection) are name that tells you where in the net topology the computer is located. But cause human are not good at remembering long number that are at the first hand meaningless we instead of that have crated some kind of parallel translation that is human like.
The DNS solve this problem, what he basically does is: a map from host names to an IP address.

1.1 The Designing goals

The main goals of DNS are Scalability (implemeted via the Distibuited database model) and Reliability.

1.1.1 The Distribuited Database Model

Which ais a federated model, where organization are managing everything about that namespace, and all the stuff about naming. It's a pretty nice model cause everybody does some about of resouces, and this is one of the reason cause the model scaled so good.

1.1.2 Reliability

Fondamentaly it is implemented via an abstraction called Lookups, that work via giving a name a returing a Record (Ip address) dns_resolve(dnsname). Application specify a context.

1.2 Namespace

The DNS is a structured namespace, that is organized as a tree. The root is at the top of domains that then are going to split into subdomains. The first layer is constitued by the generic top lavel domains (com, edu, netm org ...). Every layer of the tree is translated into a dot, and is read from bottom to top.

1.3 Delagation

This is the reason how the system scales. The best way to understeading this is recursevely. Is divided into Root (the one who ones the network), the Top-Level-Domain (which are the owner of .edu, .com ...), this structure is usefull cause we have only to go to the owner of the up label in the three and ask for creating a new down label.

1.4 Records

This is what outputs the dns when requesting for some name. What he contains is: A : Address, MX : (mail exanger for email), CNAME (canincal name) : "Synonsym", NS : Name server record.

1.4.1 Name Server Record

let's that there is a NS record for mit.edu, that NS record gives the name of the machine that is resposable for mapping the name on that zone. In this way things up in the tree do not know anything about who is associated with some name but they know someone which now more about it which is the NS record down them. And this is done recursevely.

Stab Resolver You have an application that what to call dns_resolve(), there the stab resolver send the dns request to some Name Server that help for resolving some name. Usually the request is send out to the root name server, him reading the name for the last character is able to handle the request by for example forwarding them to .edu that them is going to forward it to .mit and so on so fourth. The delegation could be done also without being called with the .edu or .com but is could also be done using the CNAME(s) which rappresents essetialy the one that are willing to handle the delegation.

1.5 Bootstrap

The bootstrap is needed after having the root ns. In fact this is just being published. The second issue is how dost the stub connect to any name server? The most common approch is just with DHCP that we'll tell which name server to use.

1.6 Performance

The approch to solve the too many round trip problem is caching. The kind of name resolution that is going on in DNS, the iterative resolution is done by pointers that refers you to the right place. Instead recursevely resolution, which is done by giving you directly the answer, but having directly the answer you could actually cache it. On the other hand you can cache referral that are actually more useful cause you could cache the address of the general zone, instead of caching the address of just some kind of specific machine in that zone.

1.6.1 Expiration Times (ttl)

It says that here is the answer and is valid for such and such time, so you have to uncache this entry after the ttl ends.