Development of the Domain Name System

Authors: Paul V. Mockapetris and Kevin J. Dunlap

In this paper, authors analyze the design, implementation, surprises, successes and failures of the DNS system. The genesis of the DNS was the observation in 1982 that the HOSTS.TXT system for publishing the mapping between host names and addresses was headed for problems. HOSTS.TXT was a single file and obliviously it couldn’t scale and a replacement was needed.

The replacement had to contain the same info as HOSTS.TXT, allow database to be maintained in a distributed manner, with no obvious limits and extensible services. Also, it required to have tolerable performance and interoperability across different networks.

The DNS architecture is simple. There are two major active components. Firstly, the name servers that are repositories of the information and answer queries using whatever information they possess. Secondly, the resolvers that are interfaces to client programs that include the algorithm to find the name server that has the info the client queries for.

The name space is a variable-depth tree and each node has a label. Domain name of a node is concatenation of all labels on path from node to root of tree. Data associated with each name is a set of resource records also known as RRs. Each record contains a tuple of (type and data) where the set of type is well known and extensible.

Database distribution is organized in two major mechanisms: zones and caching. Zones are sections of the system wide database which are controlled by a specific organization. More specifically, zones are contiguos pieces of a tree and are created by convincing a parent organization to delegate a sub-zone consisting of a node and then the owner of the node can create names in this zone. Caching mechanism consists of resolvers and name servers that cache responses for use by later queries. Resolvers are pointing to series for the DNS’s root node and the top node of the local domain. But if the required name cannot be found locally, one the root nodes of the DNS has to be hit. Finally, both mechanisms are invisible to the user.
One of the surprises of this work was the bad performance of the underlying network that was worse than they expected, although the DNS hierarchy still performed well. Authors expected to eliminate the amount of negative caching responses but due to misspellings or because programmers used the DNS lookup to check if address was valid in DARPA internet this was not achievable.

On the other hand they had many successes. More specifically variable-depth hierarchy makes possible to encapsulate other names spaces. Also organizational structuring of names makes names independent from network etc. Selection of datagrams was successful and probably essential, given the unexpectedly bad performance of the DARPA Internet and additional section processing that allows responding server to anticipate the next logical request and answer it before it was asked (reduces query traffic in half). At last, caching was very successful except of some security problems.

About the shortcomings, first of all there are difficulties relating to type and scale of growth. It is difficult to create new definitions and if so their semantics have to be clearly designed and published. Additionally, in order to be useful, new applications have to be created to use them. About the difficulty in application upgrading, it is difficult to convert network applications to use DNS and DNS resolver has to be part of the OS. Finally, regarding to the distribution of control and the distribution of expertise or responsibility, organizations are required to have redundant servers with real data before acquiring a domain and also simple but well written documentation is indispensable.

As a conclusion, it has to be mentioned that caching is effective for heterogeneous systems but caching for negative responses can’t be omitted. Moreover, it is more difficult to remove a function than creating a new one. Additionally, once the system reaches the required performance, designers lose the interest for extra optimization and last but not least variations in implementation is effective but variations in service provided is not allowed.