A Distributed Multicast DNS System: Research Paper

Daniel Johnson

October 30, 2009

1 Abstract

There exists a need for less dependance on the single point of failure that is DNS servers. It is quite possible for a computer to be completely connected to the internet, but run slowly or not at all in the case of a malfunctioning DNS server. My goal is to allow a group of workstations as present on a medium-size subnet to survive complete loss of a DNS server through collaboration. The syslab workstations provide an example of an appropriate size.

2 Background

Several efforts have been made to solve similar problems:

- The Avahi project provides name lookups for machines on the local subnet vial a .local pseudo-TLD
- DistributedDNS attempts to surpass the traditional ICANN-based name service, which is too ambitious to succeed.

My solution with work with the local link only, which will keep speed as fast as or faster than traditional nameservers, provide lookups for all hosts, not just nearby ones, and honor the authority of the root nameservers.

Another important algorithm may be DHT: Distributed Hash Tables. Since DNS records are key-value pairs, this may be perfect.

3 Development

3.1 Computer Language/Software

Python is being used for prototyping and network simulation. Eventual client/server will be written in C++.

3.2 Procedure

I will be using python for the initial proof-of-concept simulation and to make sure the protocol will allow for all necessary features. After the simulated nodes can function properly, I will translate the protocol and implementation into C++, a better language for lower-level operating system functions.

Eventually, I will implement a NSS (Name Service Switch) module in order to allow a native linux system to take advantage of these features.

3.3 Testing/Analysis

The simulation will test the network chatter of the protocol and the redundancy. By systematically removing nodes, it will be able to test how resilent the network is to changing topography.

4 Results

The benefits of offloading routine and emergency duties from the nameserver has several practical benefits. First, in the event of a nameserver outage, not all systems need to fail. While non-cached entries may not be available, those that have seen high use (google.com, for example) will still be available. This helps to eliminate one instance of a single point of failure. With a sufficient number of hosts, processing queries on the main nameserver can lead to performance issues. By dividing responsability for name lookups among hosts, the speed and scalability of lookups can be improved.

Hopefully after this project is completed it will be good enough to put into production in UNIX computer labs around the world. With enough effort and review, it should be possible to gain acceptance into the community, assuming the security requirements and social requirements are met.