# DNS

Alex S.*

There are a bunch of various protocols that are key to the functioning of the whole Internet. While we know that all messages and routing use IP addresses, most of the human population on this planet doesn't deal with that... They deal with Names—thus, there is a system that maps names to IP addresses.

## 1 DNS, Domain Name System

### 1.1 Hosts File

The Internet as a whole deals with IP addresses. To send a packet from computer to computer, that's all you need. Humans on the other hand, find it very difficult to remember and type these IP addresses. Just think, that instead of `google.com`, you'd have to type something like `216.239.37.99`. So names for IP addresses were invented. That's pretty much how simple it really is. You have an address, and you map it to a name.

Initially, that's how it worked. Every user on the "Internet" would have a `hosts` file, which would have names mapped to IP addresses. Follows is an excerpt from my `/etc/hosts` file from one of my home machines:

```
127.0.0.1       localhost
192.168.1.222   particle
192.168.1.223   fritz
192.168.1.224   ganz
192.168.1.225   frail
192.168.1.226   lucky
192.168.1.227   particle0

127.0.0.1 www.doubleclick.net
127.0.0.1 ad.doubleclick.net
...
```

What all that essentially says is that whenever I refer to `localhost` anywhere, it will resolve to `127.168.1.222`, or whenever I refer `frail` (which is my laptop), it will resolve to

*alex@theparticle.com

192.168.1.226. Once it is resolved to an IP address, that's what the networking system uses. The names are mostly only for human benefit.

Notice that in my sample hosts file, I have several names resolving to the same address of 127.0.0.1. That is mostly so my machine never talks to those other evil computers. You can find a list of these evil computers by typing "hosts file" into Google. Mike's Ad Blocking Hosts File is what you're looking for.

By now, you should be thinking "how can this work for the whole Internet with millions of computers..." Well, it sort of doesn't. There is a bigger, more complicated (and distributed) name system for the whole Internet called DNS, or Domain Name System. It does name-to-IP and also IP-to-name translations for everyone.

## 1.2 DNS

The DNS system is just a beefed up hosts file. It is a hierarchical system, with a few 'top-level' domain names at the top, such as .com, .net, .org, etc., (and by now, once people figured out they can make a fortune selling names, there are hundreds of top level domains).

Anyway, it looks like a tree, with the root node supposedly knowing everything—but it doesn't. Whenever the 'root' receives a query that it doesn't know how to resolve, it either forwards the request or responds with a computer that might be able to resolve the request. Whatever it does depends on the configuration.

Whenever your system is resolving a name, after the initial resolution, your system keeps the name/address in a cache, that generally doesn't expire all that quickly (names don't change often). A similar thing happens with the DNS servers who forward your request on your behalf—when they get an answer, they cache it, and respond with the answer to you. Next time they get a query for the same domain, it will resolve much quicker since it's already in cache. (more on this in class).

## 1.3 DNS Examples

These examples are presented in Perl[1]. You will also need to install the Net::DNS[2] package. All these examples are taken from Net::DNS man pages. They're all pretty much the same—you figure out what you need, setup a query, and then loop for results.

There is a way to do all of these in every language/system. In Java, you can use JNDI (or simpler InetAddress object—which allows you to do IP lookup and Reverse loookup). In C, there are gethostbyname and gethostbyaddr functions. In C# (well, .NET Framework), there's the System.Net.Dns class.

### 1.3.1 Looking up name servers for a host

```
use Net::DNS;
```

[1]Windows version freely available at: http://www.activeperl.com/

[2]Available at http://www.cpan.org/

```
my $res   = Net::DNS::Resolver->new;
my $query = $res->query("example.com", "NS");

if ($query) {
    foreach $rr (grep { $_->type eq 'NS' } $query->answer) {
        print $rr->nsdname, "\n";
    }
} else {
    warn "query failed: ", $res->errorstring, "\n";
}
```

### 1.3.2 Looking up SOA (Start Of Authority) record for host

```
use Net::DNS;
my $res   = Net::DNS::Resolver->new;
my $query = $res->query("example.com", "SOA");

if ($query) {
    ($query->answer)[0]->print;
} else {
    print "query failed: ", $res->errorstring, "\n";
}
```

### 1.3.3 Looking up IP address for host

```
use Net::DNS;
my $res   = Net::DNS::Resolver->new;
my $query = $res->search("host.example.com");

if ($query) {
    foreach my $rr ($query->answer) {
        next unless $rr->type eq "A";
        print $rr->address, "\n";
    }
} else {
    warn "query failed: ", $res->errorstring, "\n";
}
```

### 1.3.4 Looking up domain for IP address (reverse lookup)

```
use Net::DNS;
my $res   = Net::DNS::Resolver->new;
$res->nameservers( "NS.EXAMPLE.COM" );
```

```
my $query = $res->search("192.168.1.1");

if ($query) {
    foreach my $rr ($query->answer) {
        next unless $rr->type eq "PTR";
        print $rr->rdatastr, "\n";
    }
} else {
    warn "query failed: ", $res->errorstring, "\n";
}
```

### 1.3.5 Looking up MX (Male eXchange) servers for domain

```
use Net::DNS;
my $name = "example.com";
my $res  = Net::DNS::Resolver->new;
my @mx   = mx($res, $name);

if (@mx) {
    foreach $rr (@mx) {
        print $rr->preference, " ", $rr->exchange, "\n";
    }
} else {
    warn "Can't find MX records for $name: ", $res->errorstring, "\n";
}
```