

# CSIS0234B Computer and Communication Networks (Class B)

## Reading for Tutorial 6

### Using and Managing DNS Servers

We have seen that there are at least three different ways to address computers in the Internet: by **domain names** like `study.csis.hku.hk`, by **IP addresses** like `147.8.178.15` (which contains routing information), and by **Ethernet addresses** like `00:A0:C9:32:7F:83` (which allows Ethernet cards to filter traffic). During the last tutorial we see how the datalink layer Ethernet address are related with the network layer IP address. This time we will see how IP addresses are related with the domain names that we usually use.

#### 1. Domain name system: objectives

The **Domain Name System** (DNS) is designed to map domain names to **resources** like IP addresses. Apart from providing a name which is **easy to remember**, the system is designed with administration and delegation in mind.

- **A host can have multiple domain names:** the administrator might want different services of the same server to be accessed by different domain names: later if one of the services is moved, the users don't need to reconfigure their programs (instead the administrator modify the DNS entries). Typically, one of the domain names can be mapped to an IP address, while the others (**aliases**) are mapped to that as **canonical names**. E.g., our `www.csis.hku.hk` has no IP address; instead `webinfo.csis.hku.hk` is its canonical name.
- There are a few **types** of resources, the domain name mapping of **different types is independent**. For example, `virtue.csis.hku.hk` normally resolves to `147.8.178.13`, but if it is coming from a mail application, it resolves to an alias to `ns.csis.hku.hk`, with address `147.8.178.10` (all mails in the department are processed in `ns`).
- **A domain name can be resolved to multiple IP addresses**, they may be addresses of network interfaces in the same computer or in different computers. This allows hosts with multiple network interfaces to use the same name for all interfaces. This also allows one service to be provided by multiple computers, sharing the load. E.g., `microsoft.com` is mapped to both `207.46.245.214` and `207.46.245.222`.
- The domain name is **independent of the IP addresses**. IP addresses provide routing information: if you move from one ISP to another, your IP address must change. But you can keep using the domain names you've registered. Also, two computers with similar domain names might be located very far away. E.g., while `www.debian.org` has an IP address `192.25.206.10`, `security.debian.org` has the address `194.109.137.218`. Indeed they are in different countries!
- The domain name system is **managed in a distributed way**. Using a single server to hold information about the whole world is too unscalable. Instead, the system is run by many **DNS servers**, each holding part of the information. Each DNS server can be run by a different organization, each authoritative for only domains that it is responsible for. E.g., our departmental name server is authoritative for the `csis.hku.hk` domain, while the name server of the university is authoritative for the `hku.hk` domain.
- The domains are delegated to organizations in a **hierarchical way**. E.g., "Web Domain Hong Kong" controls the `hk` domain, which delegates the `hku.hk` domain to HKU, and the latter delegates the `csis.hku.hk` domain to our department. Each party can modify their own part of the database without first asking for permission from somebody else.

## 2. DNS from the viewpoint of a client

As one can probably guess correctly, a domain name is composed of **labels** (e.g., `www`, `csis`, etc.) separated by the “dot” character. Labels consist of one or more characters, with the exception that the **root** label has no character at all. Here, “character” means International (Unicode) character. A domain name is interpreted backwards: `www.csis.hku.hk`. `intuitively` refers to `www` of `csis` of `hku` of `hk` of the root domain. A name like the above, with a terminating dot (and thus the root domain), are said to be **absolute**. In contrast, a **relative** domain name has no terminating dot, and is searched with context. E.g., most computers of our department are configured to also try `label.csis.hku.hk` when you look for `label` as a relative domain name.

With a domain name and a specific type, we can **query** a DNS server to ask for the resources, using the DNS protocol (RFC 1034 and 1035) over UDP port 53. Each host has a library called the **resolver** which performs such the queries. The DNS server finds all the resources matching the query, and make a **DNS response**. In a **non-recursive** query, the DNS response may contain a better DNS server to ask instead of the actual resources. In contrast, a **recursive** query asks the DNS server to query other DNS server until the resources are found (or known to be absent). Most queries are recursive, to improve cache performance (see the next section).

In our computers, the resolver consists of a few functions like `res_query()` which performs DNS queries. The `getaddrinfo()` function uses it to turn hostnames to IP addresses. With the `/etc/resolv.conf` file, the resolver is configured to use specific DNS servers. It might also use other mechanisms (like a hard-coded file) to find the IP address, but we will not go into the details. If interested, see the manpages `resolv.conf(5)` and `nsswitch.conf(5)`.

We can also use `dig` command to perform a DNS query. E.g., `dig virtue.csis.hku.hk` will make a query to the DNS server asking for the default A resource of the domain name `virtue.csis.hku.hk` and show the response. One can add a type to the end of the command to ask for a different type. The available types include:

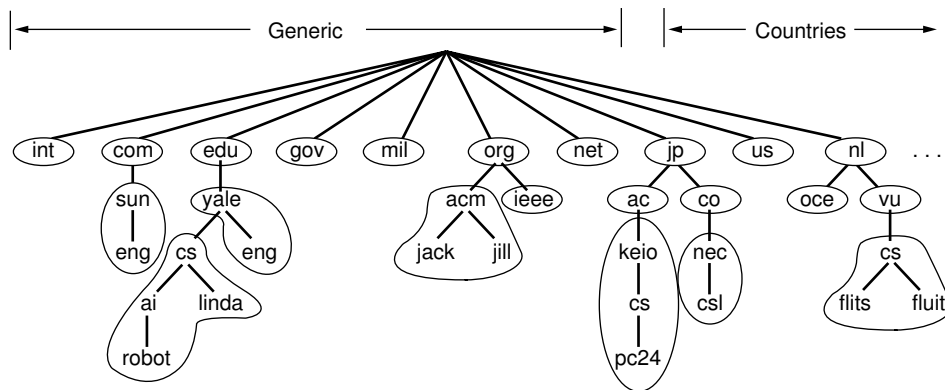
A	A 32-bit integer for its IP address.
AAAA	A 128-bit integer for its IPv6 address.
CNAME	The canonical name for a host, to establish aliases.
HINFO	The CPU and OS used by a host.
MX	The host willing to process mails for a domain name.
NS	The authoritative DNS server for a domain name.
PTR	Point to another domain name, usually for reverse lookups.
SOA	“Start Of Authority”: the top-node in a “zone”.

Apart from the A and AAAA resources which are expressed as an integer, all of the above resources are expressed in a string. DNS protocol also allows the use of different **classes**, for different types of networks. We will only use the `IN` class (i.e., Internet).

## 3. DNS from the viewpoint of a DNS server

The domain name system is run by **DNS servers**. The DNS server program in Unix is `bind` (Berkeley Internet Name Domain). DNS servers of most other platforms are based on it. As mentioned above, each DNS server only hold part of the domain information. The domain name space can be seen as a tree, where the labels in the domain names are the nodes of the tree. Each DNS server is **authoritative** for (intuitively, being the “owner” of) a number of **zones** in the tree. The zones define the administrative boundaries: E.g., in the following figure, `acm.org` and

jack.acm.org are administered by the same party, different from org.



There are a number of **DNS root servers** which serve the **root zone**. Most DNS servers know the root servers: when presented with a query outside its own domain, the DNS server can forward the query to the root servers.

Each DNS server knows the DNS servers for all its **child zones**. For example, all root servers know which DNS servers serve the top-level zones like `int`, `com`, `edu`, etc. (although they don't need to know about `sun.com`, etc). In this way, when presented with a query that is within a child zone, the request can be forwarded to a DNS server closer to the authoritative source. Any DNS server administrator may create a child zone and delegate it to another administration. Normally, when an organization wants a domain name, it decides under which domain should its new domain appear, and ask the corresponding DNS administrator to create a child zone for it (typically at a charge).

As mentioned above, DNS servers answer DNS queries for domain names outside a zone or its descendents by forwarding the query to root servers. But this can easily overload the root servers, so it is slow. DNS servers will **cache** recently accessed resources for some time, specified by the TTL (Time To Live) parameter of the resource, so that the root server is not queried again if the server is given the same query again. Furthermore, it is possible for a name server which is not busy enough and keep missing its own cache (causing queries to the root servers) to use a **forwarder**, so that all queries are forwarded to another, hopefully more busy, DNS server (which cache contains more entries and thus is more likely to be able to avoid the lengthy query from the root servers).

The owner of each resource decides what TTL values to use, which determines for how long other DNS servers cache the resource. Typical values can be as short as 5 minutes (300 seconds) for resources that keep changing (like IP addresses of hosts with a dynamic IP address allocated by their ISP), to as long as a day (86400 seconds) for resources that seldom change if at all. Cached answers are non-authoritative, and is marked as such. Sometimes clients might request for an authoritative answer, if it is critical to get up-to-date information of the resource.

#### 4. Redundancy and Zone files

Each zone usually has multiple DNS servers to provide redundancy: if at least one DNS servers is not failed, domain names can be resolved. Ideally, these DNS servers are located in completely differently geographic locations, so that even a power failure cannot cause all the name servers to fail at the same time. For example, one of the name servers serving `.hku.hk` is actually a server of the Hong Kong Chinese University.

In such cases, one of the servers is said to be a **master**, while the others are called **slaves**. Whenever the resources in the master is modified, they are copied to the slaves. To make the operation automatic, all DNS servers use the same format to represent the information in its zones: all information in a zone is stored in a **zone file** (or **master file**). The zone file stores all the resource information within the zone: the value of each of the domain names and types, the TTL, etc. The format of a zone file is described in Appendix B: we need to hand-edit such a file during the tutorial. The `bind` DNS server uses a configuration file, as described in Appendix A, to find zone files and to configure other parameters.

## 5. Reverse lookup

DNS is a process taking domain names to information, usually IP addresses. Sometimes we want the reverse process: given an IP addresses, find its domain name. The DNS system is also used for such **reverse lookup**. The primary difficulty comes from that the IP address is written in the direction where the **last** number is the “least significant”, while a domain name is written in the direction where the **first** label is the least significant. For example, `147.8.178.15` is “close” to `147.8.178.16`; while `www.csis.hku.hk` is “close” to `virtue.csis.hku.hk`, where “close” means under similar administration.

The answer is... to write the IP address in **reverse**! In particular, an IP address like `147.8.178.15` is written as `15.178.8.147.in-addr.arpa`, and given to the DNS server, asking for its PTR resource. The `dig` program has a flag (`-x`) to do this automatically. The DNS server then finds the resource just like normal DNS lookups: forward the request to a root server (which serves the `in-addr.arpa` domain), which redirects the query to the DNS server for `147.in-addr.arpa` domain. Continuing this way, the DNS server holding `8.147.in-addr.arpa` (i.e., HKU name server) and `178.8.147.in-addr.arpa` (i.e., CSIS department name server) are queried, and eventually the resource is found and returned.

For this to work, the provider of the IP address must delegate one or more **reverse zones** to the organization, which must provide **reverse zone files** to provide the resources. For example, our department DNS server serves the zones `175.8.147.in-addr.arpa` to `179.8.147.in-addr.arpa`. Recall that the parent DNS server (i.e., that for `8.147.in-addr.arpa`) must know the NS entry for each of them. For delegating a small number of domains (here there are 5), it is practical to have a NS entry for each of them. For cases where more domains are involved and the nameserver does change from time to time, usually an alias and a delegation is made instead. For example, it is possible to make `175.8.147.in-addr.arpa` to have a CNAME of `175.csis.8.147.in-addr.arpa`, and have an NS entry for `csis.8.147.in-addr.arpa` to delegate the domain to our department.

## Appendix A. Configuration file of bind

The configuration file of bind consists of some global options (e.g., what directory to store master files), followed by specific options about how to serve each zone. For example, a simple configuration file might look like this:

```
options {
    directory "/var/named";
    pid-file "/var/run/named";
};
zone "." {
    type hint;
    file "named.ca";
};
zone "localhost" {
    type master;
    file "localhost.zone";
    notify no;
};
zone "0.0.127.in-addr.arpa" {
    type master;
    file "named.local";
    notify no;
}
zone "mynet.com" {
    type slave;
    file "mynet.zone";
    masters {
        192.168.1.1;
    };
};
```

There are quite a few types of zones, here we see a `hint` zone, a `master` zone and a `slave` zone. A `hint` zone defines a startup zone file for the server to find the root servers. A `master` zone defines the zone files for each domain that the server is authoritative. A `slave` zone defines a zone which is a replication of another DNS, i.e., its `masters`. In all cases, the `file` option is the location where the zone file is stored in the filesystem. The `notify` option of a `master` zone allows a master to choose whether to notify its slaves when the zone file is modified. Some other options that relate with our previous discussions include:

- `forwarders` (global/zone): A list of DNS servers to act as forwarders, so that all queries that cannot be locally satisfied are forwarded to it. Its value is a list of IP addresses, like the `masters` option above.
- `forward` (global/zone): What to do if the forwarders fails. If the option has the value `only`, it will return `fail`. If it has the value `first`, it will try to resolve the name on its own.
- `notify` (global/zone): Whether to tell slaves when a zone is updated. The slaves are found by looking at the actual zone files and checking the `NS` resource of the zone.

## Appendix B. Zone file format

A zone file contains the actual data stored in the DNS server. Here is a simple example:

```
$TTL      86400
$ORIGIN localhost.
@          1D IN SOA      @ root (
                                42          ; serial (d. adams)
                                3H          ; refresh
                                15M         ; retry
                                1W          ; expiry
                                1D )        ; minimum TTL

                                1D IN NS     @
@          1D IN A       127.0.0.1
```

The first line specifies the default TTL for the resource records in this zone file. Then it specifies an ORIGIN for appending to relative domain names.

After that it is a list of resource records. The SOA record is always present in all zone files. It begins with the @ symbol, which is a shorthand for the domain name of the zone as specified in the configuration file, immediately after the keyword “zone”. It also has its own TTL (1D, i.e., 1 day), and a symbol IN specifying the class. Both can be omitted. The next symbol is the @ symbol again, this time specifying the “primary source of information” provided by the zone. This is conventionally set to the forward zone even in the reverse zone. Then comes a relative name root, which expands to root.localhost. This is interpreted as an E-mail address root@localhost, specifying the administrator of the domain. Finally, there is a list of operating parameters (the comments above tells their meaning). One particularly important argument is the serial number: whenever it is updated, the slaves may be notified to update their entries.

After the SOA resource record, there are two lines. The first line, with the resource record NS, specifies a nameserver for the zone. Note that, unlike the SOA record, it begins with no domain name (but instead spaces), which means “use the value of the last resource record”.

The last line is an A record, which specifies an IP address. The same address should not be given to two or more records as its A record (otherwise reverse mapping won’t work). If an alias is needed, one should use the CNAME record, in the following form:

```
localhost-alias. CNAME localhost.
```

This specifies that localhost-alias. is an alias of localhost..

The nameserver line above is not very revealing, since both sides are the same. A more interesting nameserver line looks like this:

```
mysubdomain.domain. NS host.mysubdomain.domain.
```

It says that host.mysubdomain.domain. is an authoritative DNS server for the sub-domain mysubdomain.domain. If the line appears in a zone file outside mysubdomain.domain. (and the name server is within mysubdomain.domain, as shown here), it is necessary to add an A record for host.mysubdomain.domain.: otherwise finding the IP address of that nameserver becomes a chicken-and-egg problem. Such an A record is usually called a glue record: logically it does not belong to the zone, but it is necessary to add it anyway to glue up with other DNS servers.