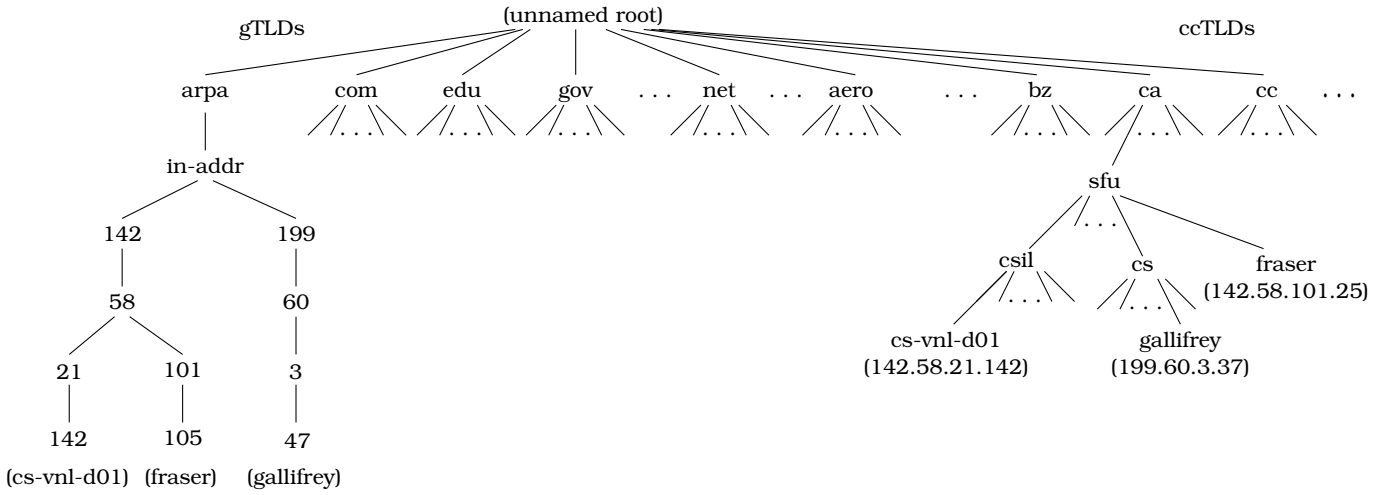
Domain Name Service

- It's nice to be able to refer to machines by names, instead of numbers. Humans do better with `fraser.sfu.ca` than with 142.58.101.25.
- When the Internet was small & cute and still the ARPAnet, the business of relating names to numbers was handled by a computer and some humans at the Network Information Center. As the Internet grew and became large and fractious, maintaining a single host database file became impossible. There were too many changes to be processed, and too many translation requests to be answered. No single office or computer could handle the load.
- What was needed was a new system that was scaleable and practical. The solution was the Domain Name Service (DNS). Defined in RFCs 1034 and 1035, it specifies how to partition the name space and delegate authority for name to address translations based on that partition.
- The organisation chosen for the DNS name space is a tree.



  - At the top is an unnamed root, sometimes visible as a terminating '.' at the end of a domain name.
  - Immediately below the root are the 'top-level domains', or TLD's.

    There are geographic TLDs (commonly known as 'country code' domains or ccTLDs), with two letter codes specified by the ISO. For Canada, the code is '`ca`'.

    There are the well-known generic top-level domains (gTLDs):

`com`	commercial organisations
`edu`	educational institutions
`gov`	U.S. government
`mil`	U.S. military
`int`	international (treaty) organisations
`net`	networks
`org`	other organisations (non-profits, *etc.*)

More recently, new gTLDs have been added. It's now possible for an organisation to sponsor a domain by providing the infrastructure necessary for name registration. For example,

Name	Constituency	Type
`aero`	air transport industry	sponsored
`biz`	business	generic – restricted
`coop`	cooperatives	sponsored
`info`		generic
`museum`	museums	sponsored
`name`	individuals	generic – restricted
`pro`	professionals	generic – restricted

Most recently, the ICANN[1] has introduced *internationalised domain names* expressed using unicode. Technical planning for IDNs dates to 2003; at present, IDN ccTLDs and gTLDs are in use. See `http://www.icann.org/en/topics/idn` for additional information.

At present (2014) there are over 100 TLDs (not counting ccTLDs), some sponsored by quite small user communities, others open to anyone. A list of TLDs can be found at `http://www.iana.org/domains/root/db`.

The whole business of domain names is, well, a business, and litigation for desirable domain names is common.

There are competing organisations at the level of the DNS root, and this has generated controversy and name conflicts. A Wikipedia article, `http://en.wikipedia.org/wiki/Alternative_DNS_root`, gives a nice introduction.

❄ For some interesting history and discussion of the pressures being brought to bear on the DNS system, have a look at RFCs 3071 and 3467.

❄ The `sfu` domain is a second-level domain, registered under the `ca` ccTLD. To refer to a system at SFU, `fraser`, for example, you might use the name `fraser.sfu.ca`.

[1] The Internet Corporation for Assigned Names and Numbers.

- The `sfu` domain is itself divided into lower-level domains, such as `cs`, `fas`, `csil`, `ensc`, `cecm`, and others.

- One other TLD that needs special mention is the `in-addr.arpa` domain. This is used for translation of IP addresses back to host names.
  - The host `fraser` has IP address 142.58.101.25.
  - A query for the name `25.101.58.142.in-addr.arpa` would return the name `fraser`.
  - Notice that the octets of the IP address are written according to the standard DNS ordering, where the portions of the name are written from most local to most global.

- Let's be precise about what we mean by a DNS *domain* and a DNS *zone*.
  - A DNS *domain* is a complete subtree in the DNS name tree.
  - A DNS *zone* is a subtree, minus any delegated subtrees. Generally, we would phrase this as "a zone is a DNS domain, minus any delegated subdomains". Delegation is an administrative matter, in which responsibility for a portion of the DNS name space is handed off by one organisation to another.
  - The point is that a domain is defined as a structure (a subtree) in the DNS name tree. A zone is partially defined by the DNS tree structure (a subtree) and partially by administrative boundaries (minus delegated subtrees).
  - When an administrative authority takes control of a zone, it must agree to provide two completely independent DNS servers (*i.e.*, on different machines, on different networks, independent power supplies, *etc.*) so that any reasonable single-point failure will not leave the domain without a DNS server.
  - Most often, an administrative authority will have control of several zones. In the simplest case (no delegated subdomains), there would be a zone for the domain and a zone for the corresponding `in-addr.arpa` domain.
  - While there's often a correspondence between physical network structure, organisational administrative structure, internet addresses, and domain name structure, recognise that there's no requirement that this be so. It's just that most often, an organisation will apply for a block of IP addresses and a domain, and then create physical subnets (IP address assignments) and DNS subdomains that correspond to its internal organisational structure.

- Let's look again at the `sfu.ca` domain, a subdomain of the `ca` ccTLD.

- ❄ The Faculty of Applied Sciences Network Support Group (NSG) is an independent administrative authority, delegated from IT Services (ITS), the administrators of the `sfu.ca` domain. The NSG administers the `cs` (Computing Science) domain, as well as several others: `fas`, `ensc` (Engineering Science), and `math` (Mathematics and Statistics). Physically, this group of workstations and network segments is the FASNet.

  If you look up the name servers for the `fas.sfu.ca` domain (using `nslookup` or `dig`) you'll find that the NSG runs a name server, `cs.sfu.ca`, and has an agreement with ITS to provide independent DNS service using the servers `whistler.sfu.ca`, `seymour.sfu.ca`, and `ns3.sfu.ca`.
- ❄ Moving up the tree, ITS provides DNS service for `sfu.ca` using the servers `whistler.sfu.ca`, `seymour.sfu.ca`, and `ns3.sfu.ca`. Since `ns3.sfu.ca` is located at Harbour Centre, this satisfies the requirement for physically independent DNS servers.
- ❄ The Canadian Internet Registration Authority (CIRA) operates or contracts for the operation of seven DNS servers for the `ca` domain[2]. It's common for one organisation to make an agreement with another to provide an independent name server.
- ❄ The Internet Corporation for Assigned Numbers and Names (ICANN) contracts with various organisations to operate the root servers for the '.' domain, `a.root-servers.net` through `m.root-servers.net`. It administers some TLDs directly and contracts with companies or organisations for administration of others.

  Nominally, there are 13 root servers, each with an individual name and network address. Through clever use of the Internet packet forwarding infrastructure, some 250 physical root servers are distributed around the globe. A query to one of the 13 root server network addresses is automatically distributed to one of the 250 physical servers.

❄ If you want to apply for a subdomain within the `ca` domain today, you'd apply to the CIRA. CIRA took over from the previous organisation, a volunteer group called the .CA committee, in Fall, 2000. You, however, cannot deal directly with CIRA — you'll have to go through a CIRA certified registrar, thus providing additional Internet business opportunities.

In general, administration of a ccTLD is handled by an organisation in that country.

❄ If you want to register a subdomain in one of the generic domains (gTLDs), you'd go to one of the registrars accredited by ICANN.

[2]Six servers, `c`, `e`, `j`, `k`, `l`, and `any.ca-servers.ca` are operated by CIRA. One, `tld.isc-sns.net`, is operated by the Internet Systems Consortium (ISC).

- It's worth repeating that registering a domain name is entirely independent of acquiring an IP address. You need both in order to construct DNS entries for your domain.
- Now that we know the basic structure, let's delve into the workings of the protocol.
- The DNS system architecture is implemented as two components:
  - The *resolver* is most often implemented as a subroutine library used by client programs to compose DNS queries and receive answers.
  - The *name server* is a dæmon (`named` in the Unix world). It listens for queries at well-known port 53 (UDP and TCP). (Look for `domain` in `/etc/services`.)
  - UDP is used for most queries and responses. In the special case where the entire contents of a DNS database is being transferred (an activity called a *zone transfer*) TCP may be used to avoid datagram size limitations and ensure reliable data transfer.
- The most visible use of DNS is to answer queries for the IP address associated with a host name. In fact, it handles several other types of information and could potentially be used as a general key-value service.
- To explain how a query is answered we'll stick with name to address translation. The resolver composes the query and sends it to one of the DNS servers that it knows about (call this server the local server).
  - For the typical case of a lightweight resolver library dealing with a set of local DNS servers, the local DNS servers will be configured to accept recursive queries (defined immediately below) from processes running on the local workstations.
- If the local server doesn't know the answer, the question is passed to a server at a higher level (closer to the root of the DNS name tree). In the absence of cached information (as explained below), this will be one of the Internet root servers at the root of the tree. The query is then passed down the tree until it reaches a server which knows the answer.
  - There are two ways in which this can be accomplished, *iterative* and *recursive*.
  - In an iterative query, the name server which receives the query will respond with the answer or with the name(s) of the name server(s) that it would consult to find the answer.

- In a recursive query, the name server which receives the query will take it on itself to send off a query to another name server, if necessary, and will respond with the final answer.
- As mentioned above, a common configuration is for a local DNS server to be configured to accept recursive queries from hosts and resolve the query using iterative queries to other DNS servers. This is shown in Figure 2.21 in the text.

  Figure 2.22 in the text illustrates a fully recursive query. You will not see this in practice, as the higher-level servers do not have time to deal with recursive queries.
- You can see the process of delegation from the Internet root servers by using the trace option to the `dig` command ('`dig +trace` *name*').

- In a standard configuration, a name server starts off knowing the names and IP addresses of the 13 root servers for the Internet.
  - There are only 13 root servers because the designers of DNS decided that the DNS records required to answer a query for the name servers for the '.' domain should fit in a 512 byte UDP datagram[3].

    At one time, this was an actual physical limit. More recently, use of anycast[4] IP addresses has removed the physical limit, and there are now many root servers at over 250 geographically distributed locations. See `http://www.icann.com/announcements/announcement-08mar07.htm` for an interesting article about a distributed denial-of-service attack on the Internet root servers on February 6, 2007. The article includes a description of the Internet root server system.

    See `http://www.icann.org/maps/root-servers.htm` for one of several maps to be found on the Internet.
  - DNS servers do not start with more extensive knowledge of servers higher in the DNS name tree because this would imply an obligation by those servers to distribute notifications if they changed their IP address. Given the expansion at each level of the DNS tree, this is an unreasonable burden. (See, for example, RFC 2826 §1.3.)

    In particular, and despite a common belief to the contrary, a DNS server does *not* start life knowing the address of its immediate parent server in

[3] The figure of 512 bytes was chosen during the design of the Internet protocols. Adding a 64 byte header allowance gives the minimum IP packet size of 576 bytes. Because DNS defines a compression scheme to limit the repetition of data in DNS messages, it's not straightforward to demonstrate that this limits the number of root servers to 13.

[4] Anycast is a relatively new use of IP addressing and the routing infrastructure which allows a single IP address to be assigned to many different physical interfaces.

the DNS name tree. (See, for example, RFC 2181 §6.) It will, however, quickly learn the name and IP address of its parent in the normal course of answering queries.

- A DNS server must be told the names of the servers for domains below it in the hierarchy. This is what allows a server at a higher level to pass a request down the tree to the server for a particular domain. This implies an obligation by the lower server to notify the (single) parent server if it changes its IP address.
- A subset of the DNS servers for a domain, called primary servers, must also be initialised (from files) with knowledge of hosts within their domain. Typically there is exactly one primary server, to avoid replication of the data files.

- Efficient operation of the system depends on local caches of DNS records.
  - Suppose the resolver on a host contacts a local DNS server with a query for the IP address associated with a specific host name.
  - When an answer to the query comes back to the local server, it will typically include not only the address for the specific name given in the request, but also records for the authoritative name servers for the domain, plus additional records that are likely to be useful in the near future.
  - All the records are cached, along with the answer for the specific host.
  - The next time there's a query for that specific host, the local DNS server will answer it from its own cache. The next time there's a query for an unknown host in the same domain, it'll be passed directly to the domain's name server, using the cached records that specify the DNS servers for the domain.

- That's our second cut at describing the system. Now, let's fill in a few more details.
  - The reference implementation for the DNS server and resolver library is an open source package called BIND (for Berkeley Internet Name Dæmon), maintained by the Internet Software Consortium (ISC).

- The resolver doesn't need to be told very much — just the IP addresses of the DNS servers it should use.
  - On unix systems, the place to look is `/etc/resolv.conf`. It can contain the IP addresses of up to three DNS name servers.

- The operation of the name server dæmon is a bit more complicated. There are three types of servers, primary, secondary, and cache-only.
  - A *primary* or *master* server for a zone initialises its cache of DNS records from a set of files that are maintained by a human. These files contain information about the hosts in the zone. We haven't entirely gotten rid of the need for a human to maintain the name to address information, simply parcelled it out into manageable bits.
  - A *secondary* or *slave* server for a zone initialises its cache of DNS records from the primary server for a domain.
  - Primary and secondary servers are *authoritative* for a domain. When one of them provides an answer, that's the end of it. The reason for having primary and secondary servers is so that the files which are the source of host information for the domain exist in exactly one place. The primary server initialises itself from the files, and then the secondary servers initialise themselves from the primary server.
  - A *cache-only* server is defined by the absence of primary or secondary server status. A cache-only server may be initialised with the names of the Internet root servers, or it may be given a short list of name servers to which it will forward all queries.
  - All servers, as they receive answers to queries, can gradually build the capability to answer more and more queries from their local cache. The reason for having cache-only servers is efficiency. A workstation might run its own cache-only server, or an organisation might provide several, scattered about its intranet.
- When a DNS server starts up, it looks for a configuration file. The default file for unix is typically `/etc/named.conf`.
  - The configuration file can specify lots of options for `named`, but the important thing we need to know here is that it will contain a set of records which associate domain names with database files called *zone files*.
  - Each data file contains DNS records for the domain, minus delegated subdomains, plus 'glue' records specifying the name servers for delegated subdomains. The data files are called zone files because they contain information for a single zone.
- Just what is defined in DNS records? There are about 50 DNS resource record (RR) types. These notes describe six record types that are the most important in practice in IPv4.

- The ones we'll talk about are Start of Authority (SOA), Name Server (NS), Address (A), Canonical Name (CNAME), Domain Name Pointer (PTR), and Mail Exchange (MX).

- The overall organisation of a DNS zone file is a SOA record, which defines the primary DNS server for a domain, followed by a series of other records which define hosts and services in the domain and DNS servers for subdomains (called points of delegation in DNS terminology).

- A SOA record specifies the host that is the primary server for a zone. It also provides contact information and time limits for DNS records for the zone. By definition, there is only one SOA record in a zone file.

- A NS (name server) record defines an authoritative nameserver for a domain (primary or secondary).

- An A (address) record provides the host to IP address translation information.

- MX (mail exchange) records allow mail to a domain, or to a specific host, to be directed to a designated host.

  - One can specify multiple mail exchange hosts, in which case a mailer will try them in order until it finds one that works. The order is specified by a preference value specified in the MX records.

- The CNAME (canonical name) record is used to specify alternate names for a host.

  - The serious purpose of this type of record is to allow various standard nicknames to be easily mapped to hosts, *e.g.*, loghost, mailhost.
  - You can assign as many nicknames to a host as you like (keeping in mind that you're really assigning nicknames for a particular IP address).

- Recall that the `in-addr.arpa` domain exists to translate IP addresses back to names. The PTR (pointer) record is used to specify this inverse translation.

- Now that we've seen how the primary server's DNS database is established from files, let's have a quick look at DNS message types and how they're used to communicate between resolver and server, and between servers. We won't look at the details of DNS messages, which use a fair bit of compression to reduce the number of bytes actually transmitted.

- DNS is, in some ways, a simple protocol. There are queries, and responses to queries. Originally, there were three types of query: standard (QUERY),

inverse (IQUERY), and server status (STATUS). Inverse queries are now obsolete. Figure 2.23 in the text shows a high-level representation of the content of a DNS message.

- A header is present in both queries and responses. It specifies the type of query or response, various control flags, and the size of the remaining sections of the message.
- A query has only one additional section: the 'question' section, which contains a question record (a domain name, and the type and class of query).
- A response has up to three additional sections: answer, authority, and additional.
- The answer section contains resource records which satisfy the query. (*I.e.*, they matched the specified name, type, and class keys.)

  The authority section contains resource records which specify an authoritative name server for the domain specified in the query.

  The additional section contains resource records which are related to the query but not strictly part of the answer (address records, for example, to go with SOA, NS, or MX records).

  There's a fair bit of flexibilty in terms of what records are included in each section.

Dynamic DNS

- As originally conceived, DNS had a static database of files, and updates were made by manually editing those files. This seemed a reasonable choice given that DNS information was expected to be relatively static.
- That changed as networks grew. What finally broke the 'relatively static' mold was the introduction of environments like ISPs, 'drop-in' networks, and wireless networks, where it was routine to have computers dropping in and out of the network in a time frame measured in minutes.
- Dynamic DNS, defined in RFC 2136, is an extension of the DNS protocol which provides for on-line updates.
  - The basic DNS protocol provides QUERY and STATUS messages.
  - Dynamic DNS adds an UPDATE message, to allow for online incremental updating of resource records. Prior to the introduction of UPDATE, the only way to update the DNS database for the primary server was to edit a zone file and reload it.

- An UPDATE message specifies what is to be updated and prerequisite conditions for applying the update.
  - The prerequisite conditions take the form of requirements for the presence or absence of particular resource records or sets of records.
  - The update can result in the addition or deletion of one or more records. (Modification is deletion followed by addition.)
  - Updates are atomic. All prerequisites are checked for satisfaction, then all updates are applied. If, for some reason, any individual change fails, the entire update is retracted.
- A program making an update request (a dhcp dæmon, for example) can contact any primary or secondary DNS server for the zone it wants to update[5]. If the server isn't the master server for the zone, it is obligated to forward the request to the master server and then forward the response to the original requestor.
- Allowing dynamic update introduces all the complications of maintaining a database in the presence of concurrent transactions.
  - A particularly nasty quirk is that either UDP or TCP can be used to request an update; the answer is returned by the same protocol.

    Suppose an UPDATE message delivered by UDP is successfully processed by the server but the UDP reply is lost. The requestor repeats the request. The repeat request fails, because the successful execution of the original request changed the database records in the server and the prerequisites no longer hold. This time the reply — failure — is successfully delivered to the requestor. The server and the requestor are now inconsistent, because the server's database has been updated but the requestor thinks the update has failed.

    The writers of the RFC recommend the use of TCP if this sort of situation is possible.
- Security is another big concern if on-line updates are possible.
  - One solution is to configure the DNS server so that it will only accept updates from a known set of trusted IP addresses.

[5]The details as stated in RFC 2136 are more complicated. If a server receiving an update request is not authoritative for the zone being updated, it returns a 'not authorised' error to the requestor. In practice, this restricts requests to the primary and secondary servers for a zone.

- A better solution is to use a secured version of the DNS protocol (RFC 2137), or run DNS using a secure version of IP. (For 'secure', read 'encrypted, with authentication'.)