Addressing Schemes and Domain Names

At every level of networking, each host is assigned one or more numeric identifiers that uniquely represent it within a particular network (though they are not always globally unique). The exact nature of those identifiers varies depending on whether you are working with networking at a high level (IP, for example) or a low level (raw Ethernet packets, for example).

Link-Layer Addressing

At the link layer (physical layer), each network interface is usually identified by a globally unique hardware ID:

- Ethernet—MAC address (short for Media Access Control)
- Wi-Fi—MAC address (short for Media Access Control)
- Bluetooth—Bluetooth hardware ID (similar to a MAC address, but it does not share the same namespace)
- GSM Cellular—IMEI (International Mobile Equipment Identity)
- CDMA Cellular—ESN (Electronic Serial Number) or MEID (Mobile Equipment Identifier)

Because the identifier uniquely identifies a single network interface, you should never encounter two physical devices that share the same hardware address on the same physical network. More importantly, if a single host has more than one network interface of the same type, each interface has a *different* hardware ID.

The hardware ID is used to determine whether a particular device should listen to a packet or ignore it. Some physical networks also limit which packets get sent down a particular wire, based on whether an interface matching that hardware ID was seen on that wire previously. These networks are referred to as *switched networks*. Networks in which every host sees every packet, by contrast, are called *shared networks*.

> **Note:** Most programs do not need to use link-layer addresses directly.

IP-Layer Addressing

At the IP layer and above, hosts are identified by an *IP address* (which is also sometimes called an IP number). An IP address can be in one of two forms: IPv4 or IPv6.

An *IPv4 address* consists of four bytes, and is usually represented to the user as a series of four numbers separated by decimal points. For example, the loopback address that always sends data back to the sending machine is `127.0.0.1`.

Because the IPv4 address space is limited in size (and rapidly running out), the IPv6 standard was created. This standard greatly expands the number of available addresses.

An *IPv6 address* is a 128-bit value, and is usually written as eight groups of 16-bit hexadecimal numbers separated by colons. Leading zeros in each group can be omitted as long as there is at least one digit in each group. For example, the IPv6 loopback address, `0000:0000:0000:0000:0000:0000:0000:0001`, can be simplified to `0:0:0:0:0:0:0:1`.

Also, if an IPv6 address contains several groups of zeros in a row, you can omit those groups and replace them with a double colon. You can do this only once per address (because otherwise the

address would be ambiguous). Thus, the IPv6 loopback address, `0:0:0:0:0:0:0:1`, can be further simplified to `::1` (by replacing all the leading zero groups with a double colon).

Domain Name System (DNS)

IPv4 addresses are hard to remember, and IPv6 addresses are quadruply hard (because they are four times as long). To make it easier to describe a particular host, the domain name system (DNS) was invented. Besides being easier to remember, domain names have several other advantages over IP addresses. Domain names:

- Minimize service disruption when an IP address changes. The IP addresses of a given host can change over time—when a mobile phone goes out of Wi-Fi range, when a user reboots his or her cable modem at home, when the owner of a server changes ISPs, and so on. By updating the domain name record to point to the server's new address, users can continue to access the server by the same name.
- Allow a host to be accessed by more than one address. To improve robustness and performance, a server might have multiple IPv4 and IPv6 addresses on multiple physical network connections. If they all share the same domain name, clients can connect to any of its addresses. And when new addresses are added, clients can automatically learn about them.
- Allow multiple physical hosts to pretend to be a single host. For example, a single domain name might be serviced by a dozen servers that are geographically distributed around the world to improve performance.
- Can adapt to changes in the underlying technology. For example, an app that connects to a host by name does not need to care whether that host is reachable by IPv4 or IPv6. It asks for the address, and the domain name resolver returns an appropriate data structure for connecting to that host.

This section describes the parts of a domain name and the DNS lookup process at a high level.

What's in a (Domain) Name?

A *domain name* is a human-readable name that describes a particular host. Each domain name is made up of a series of parts separated by periods. The easiest way to explain these parts is by example. Consider the hostname `mail.example.com`:

- `com`—This part of the domain name is outside your administrative control. This may have multiple parts, but at a minimum, it contains a top-level domain (TLD). These top-level domains fall into one of the following categories:
  - A generic TLD such as `.com`, `.org`, or `.net`
  - A restricted TLD such as `.biz`, `.edu`, or `.gov`
  - A sponsored TLD such as `.mobi`, `.museum`, or `.travel`
  - A two-letter country code such as `.us`

  When the top-level domain contains a country code, the portion of the domain name outside your control may have multiple parts. For example, a business in the UK generally has an address in the `.co.uk` namespace.

  Each of the top-level domains, in turn, is part of the dot (`.`) root domain. In common usage, the root domain does not contribute any characters to the domain name itself. Under the hood, however, every fully qualified domain name ends in a period. For example, the name `apple.com.` can resolve to exactly one domain. The name `apple.com` typically resolves to `apple.com.`, but if that lookup fails, it could also resolve to `apple.com.example.com` if your computer is configured with `example.com` as its default search domain.
- `example`—The domain part. This part is under the administrative control of your company or organization, and applies to the company or organization as a whole.
- `mail`—The host part. This part allows you to uniquely identify multiple servers in a single domain. In addition, the domain name itself (with no host part) is also a valid hostname—apple.com, for example.

Domains can also contain any number of subdomains. For example, `www.david.example.com` would be the `www` host in the `david` subdomain of the `example.com` domain. For that matter, even the `example.com` domain is a subdomain of the `com` domain, which is itself a subdomain of the root domain. Every domain (except the root domain) is a subdomain of another domain.

Be aware that in the context of end-user discussions, the term domain is often used to refer to the largest part of the domain hierarchy that someone who wants to set up a service can actually buy or otherwise request. Thus, in the `com` hierarchy, a domain would contain two parts—`apple.com`, for example. Similarly, a domain in the `au` hierarchy would include at least three parts—`apple.com.au`, for example. It is not strictly correct to limit the use of these terms in this way, but this usage is not uncommon.

In general, domain names must be in ASCII (although there is a standard for internationalized domain names) and can contain only letters, numbers, and hyphens. Domain names are case-insensitive. For more information about internationalized domain names, see http://www.icann.org/en/resources/idn.

Looking Up Names

Before a computer or other device can contact the host associated with a domain name, it must first look up the name to obtain the corresponding IP addresses. To do this, it contacts a DNS server. After the server returns one or more IP addresses, the computer or device can then connect to the remote host using any of those addresses.

Because a number of different agencies in different countries work together to manage the domain name system, the process for looking up addresses is somewhat complicated. A typical domain lookup includes the following steps:

1. Your computer or other device sends a query to a DNS server.
2. Your local DNS server sends a query to one of several central servers (called the root servers), asking what server is authoritative for the domain.
3. The root servers generally delegate responsibility for the lookup. Instead of providing a response with the IP address for `www.david.example.com`, the root servers instead tell you to ask another server at a specific IP address.
4. Your local DNS server sends a query to that TLD server, asking what server is authoritative for the domain.
5. The TLD server consults a database to see which servers answer requests for the specified domain.
6. The TLD server delegates responsibility for the lookup to the authoritative servers for the domain (and provides their IP addresses).
7. Your local DNS server sends a query to one or more of the servers provided by the TLD. That query asks for the host's IP addresses.
8. That server either returns one or more IP addresses or further delegates responsibility to another server.

For example, if you are looking up a host in the `com` top-level domain, your computer makes two requests: one asking for a list of IPv4 addresses and one asking for a list of IPv6 addresses. The root server then delegates responsibility to a gTLD (generic top-level domain) server, which in turn delegates responsibility to a server that is authoritative for the domain. That authoritative server may either return the answer or further delegate responsibility to another server for a specific subdomain.

This process of asking other servers to provide information is called *recursion*. In general, DNS servers that are intended for use by end users (caching servers) support recursion, whereas DNS servers that are authoritative for a specific domain do not. For this reason, your local DNS server may have to talk to several DNS servers before it finally reaches one that answers for a specific domain or subdomain.

You can use the `whois` tool to learn about the domains within most registries, including which domain name servers are authoritative for the domain. For more information, see the `whois` man page. You can also use the `nslookup` and `dig` tools to perform traditional unicast DNS lookups, and `dns-sd` to browse, resolve, and advertise Bonjour services.

Other Uses of DNS

DNS lookups can provide more than just IP addresses. The DNS record for a hostname can contain various record types that each provide different kinds of information. A few of the more interesting record types include:

- `A`—An IPv4 address.
- `AAAA`—An IPv6 address.
- `CNAME`—A canonical name (mapping one hostname onto another hostname).
- `DNSKEY`—An encryption key used by DNSSEC (a cryptographically secure enhancement to the domain name system that is in the process of being phased in) when verifying the authenticity of a DNS reply.
- `MX` (mail exchanger)—The mail server (or servers) that should accept mail on behalf of the specified domain.
- `NS`—The name server delegation for a particular record (indicating that a request for that record should be answered by another server).
- `PTR`—A pointer to a canonical name. Similar to a `CNAME` record except that resolving typically stops at this point, and the client must then resolve the resulting `CNAME`, if desired. This is primarily used for reverse DNS lookups (because the goal is to get a name from an IP address, not to get the IP address back again). It is also used by DNS Service Discovery to store the human-readable name for a service.
- `SOA` (start of authority)—Used primarily to indicate how long clients should cache the results and which other servers are authoritative for the domain.
- `SRV`—Contains the hostname and port for a provided service. This record type is used by DNS Service Discovery.
- `TXT`—Contains a series of informational attributes used by DNS Service Discovery.