A quick introduction to the Domain Name System (DNS). Shows the basic purpose of DNS, hierarchy of domain names, and an example of how the DNS protocol is used. There are many details of DNS hidden from this presentation. It is more complex than it appears here (and you may see some of those complexities in the lab and your own network), but this should still give you a good idea of the main principles.

Computers use IP Addresses

- To send an IP datagram to another computer on Internet (e.g. a web server), must know its IP address
- What is the IP address of the computer running ICT website? Registration? Facebook?
- Humans prefer words (compared to numbers)
- Domain names identify computers in Internet
- What is domain name of ICT website? Registration? Facebook?
- All computers in Internet have IP address; some (usually servers) also have domain name

2

The core protocol for Internet communications is IP. To get data from one computer to another, IP is used. And importantly, the IP address of the destination computer is needed by the source (and intermediate routers) to correctly deliver the datagram. IP addresses identify computers (or more precisely, network interfaces on computers) in the Internet.

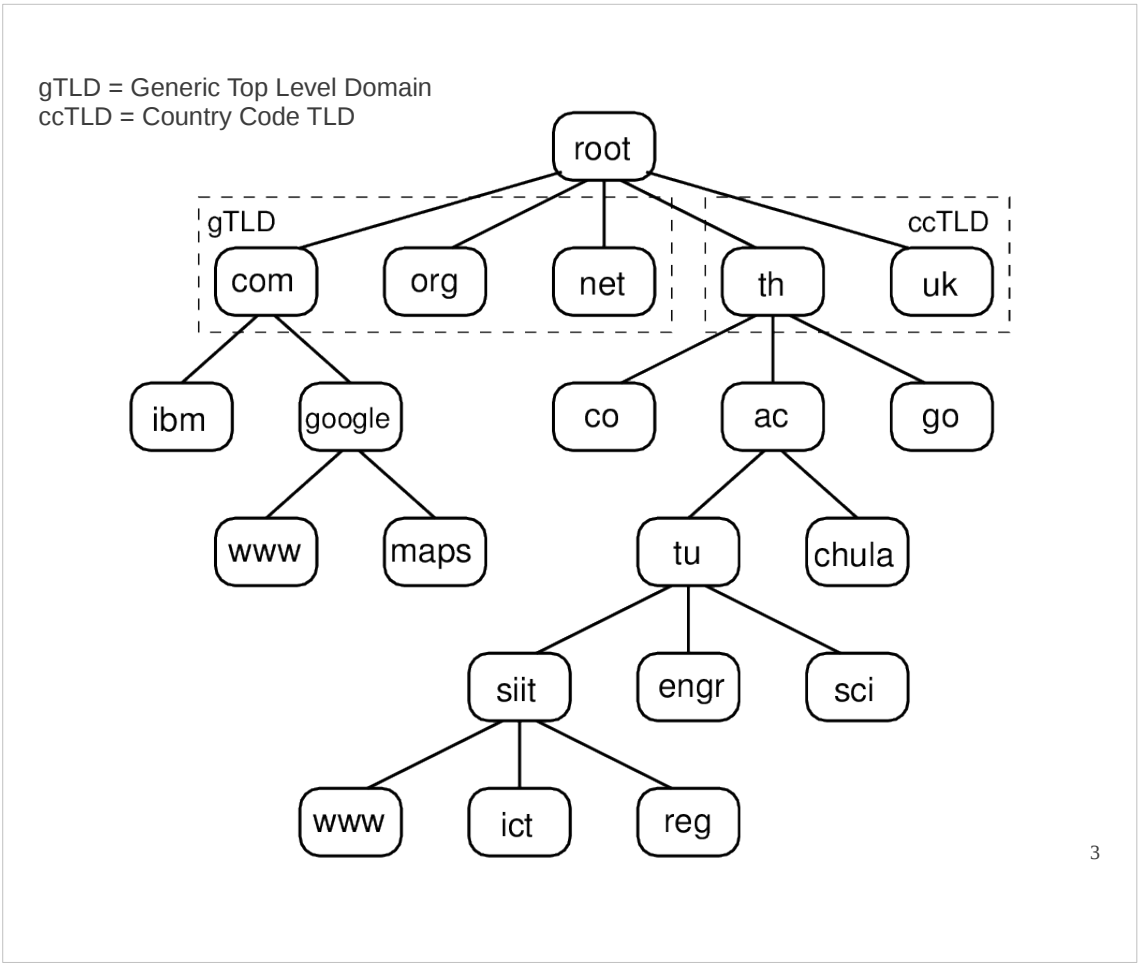
So computers use IP addresses to identify who they want to communicate with. However the human users of computers usually select the destination. For example, if you want to tell your computer, via the web browser, to download a web page of the Facebook web server, you, the human user need to tell the computer the address of the computer running the Facebook web server.

The problem: IP addresses (decimal numbers) are not convenient for humans to remember.

The solution: introduce another addressing system, using words, that humans can remember, and have a computer automatically map those human-friendly addresses to computer-friendly IP addresses. The human-friendly addresses are domain names, e.g. www.facebook.com.

In general, all computers in the Internet have IP addresses. Some computers also have a domain name. Computers running servers, especially web servers, often have domain names because they are the most common computers that humans want to communicate with. A computer doesn't need a domain name, and in fact most don't have one.

The following will show the basic structure of domain names, and then focus on how computers automatically map domain names to IP addresses. This is called the Domain Name System (DNS).



Domain names are organised into a hierarchy. The picture shows a very small subset of the hierarchy of domain names.

The top of the hierarchy is the root. This is not a domain.

The top-level of domains (TLDs) include generic domains, like com, org, net, info, biz, as well as country-code domains, such as th, uk, de, us, au.

TLDs are then subdivided into subdomains (e.g. google.com, ac.th), which may be further divided (e.g. maps.google.com, tu.ac.th, siit.tu.ac.th, www.siit.tu.ac.th). The main restriction on the number of sub-divisions is the lengths supported by related protocols.

Sub-domains are separated by dots (.). We can say com is a domain, google.com is a domain, and maps.google.com is a domain. However not all domains have a corresponding IP address (e.g. com does not, google.com does). Those that do are also called hostnames (e.g. com is a domain, but not a hostname; google.com is a domain and hostname).

Allocation of domains is delegated by ICANN to different domain registrars (companies, government organisations). They may further delegate to other registrars.

A registrar is responsible for allocated domains to organisations and individuals. There is usually a fee to be payed to register a domain. However sub-domains within that registered domain may be managed internally by the organisation or individual that owns the domain. For example, Google owns google.com – it was obtained via a regsitrar. Sub-domains such as www.google.com and maps.google.com can be created and managed by Google.

Importantly, domains (specifically hostnames) should map to IP addresses. The following will explain how.

Domain Name System (DNS)

- Map domain names to IP addresses
- Clients have DNS Resolver, configured with IP address of 1 or more DNS Servers
  - /etc/resolv.conf
- DNS servers store mappings from domain to IP
  - Server operators register their domain/IP with DNS server
- DNS protocol
  - Request/Response
  - UDP, Server port 53
- Hierarchy of DNS servers
  - Authoritative server is where domain/IP originally registered
  - Caching used in resolvers and DNS Servers

4

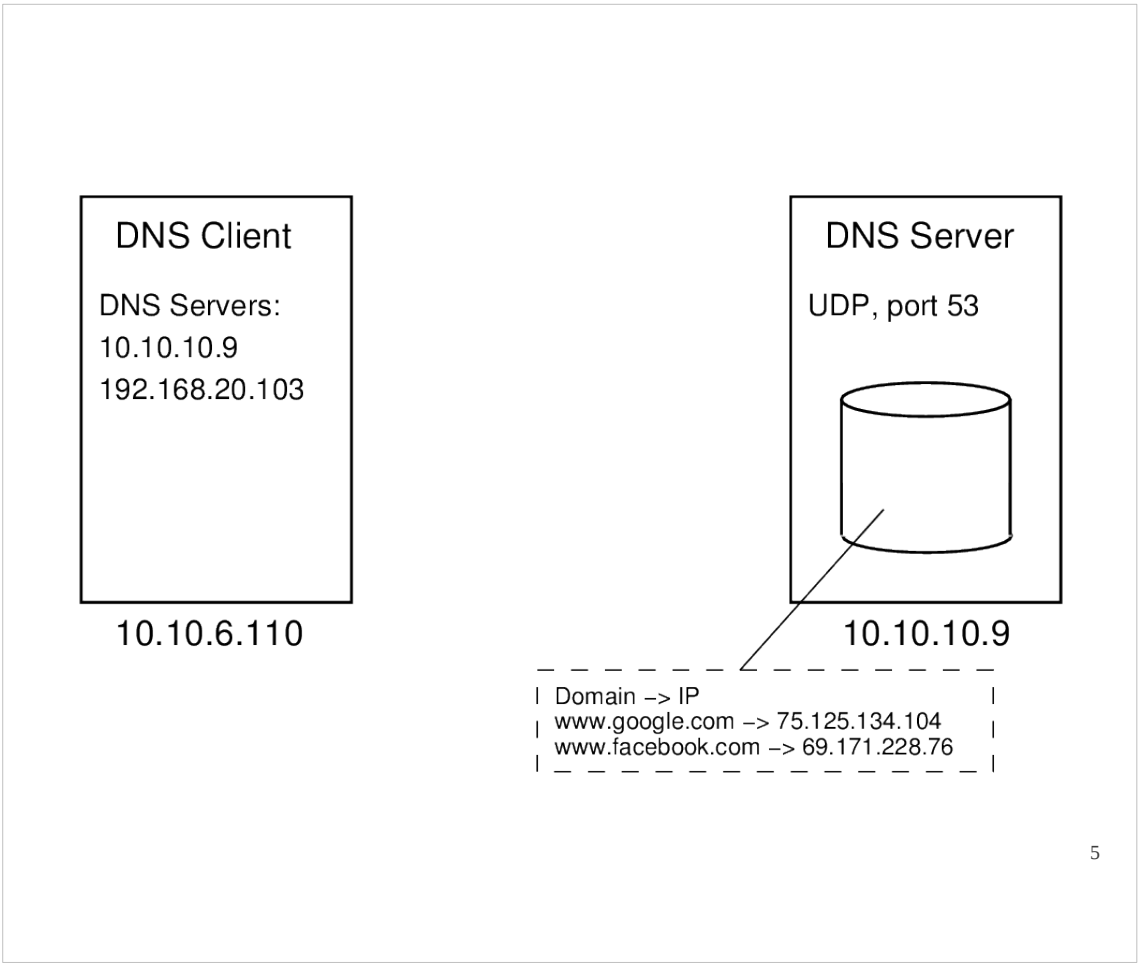
Assume that a domain owned by an organisation, and all of its sub-domains, are to be used to identify computers in the Internet. That organisation must know the IP addresses of those computers. The organisation then stores the list of domains and corresponding IP addresses in a database. Now, when a human user types a domain into the web browser to identify a server computer they want to communicate with, the database is queried to find the corresponding IP address of that server computer. Once the IP address is known, an IP datagram can be sent to the server computer.

The Domain Name System (DNS) implements the above mapping of domain names to IP addresses. There are several key components:

• The database storing mappings of domain to IP is in fact a globally distributed database. The mappings are stored in DNS servers (computers running a DNS server application). An individual DNS server doesn't store all mappings, only a small subset. DNS servers are arranged in a hierachical manner (following the arrangement of domain names) to make the storage and lookup of domain names efficient. In practice, many organisations and ISPs maintain their own DNS server. Organisations that own a domain, register the mapping of domain to IP in one or more DNS servers.

• Computers which take domain names as input (e.g. your computer with a web browser) usually have DNS Resolvers. These are configured with the IP address of 1 or more DNS servers. When an application like a web browser has a domain name, it contacts the DNS resolver, which then contacts the DNS server to find the corresponding IP address.

• The method for resolvers to communicate to DNS servers, and also DNS servers to other DNS servers, is the DNS protocol. An application layer request/response protocol.

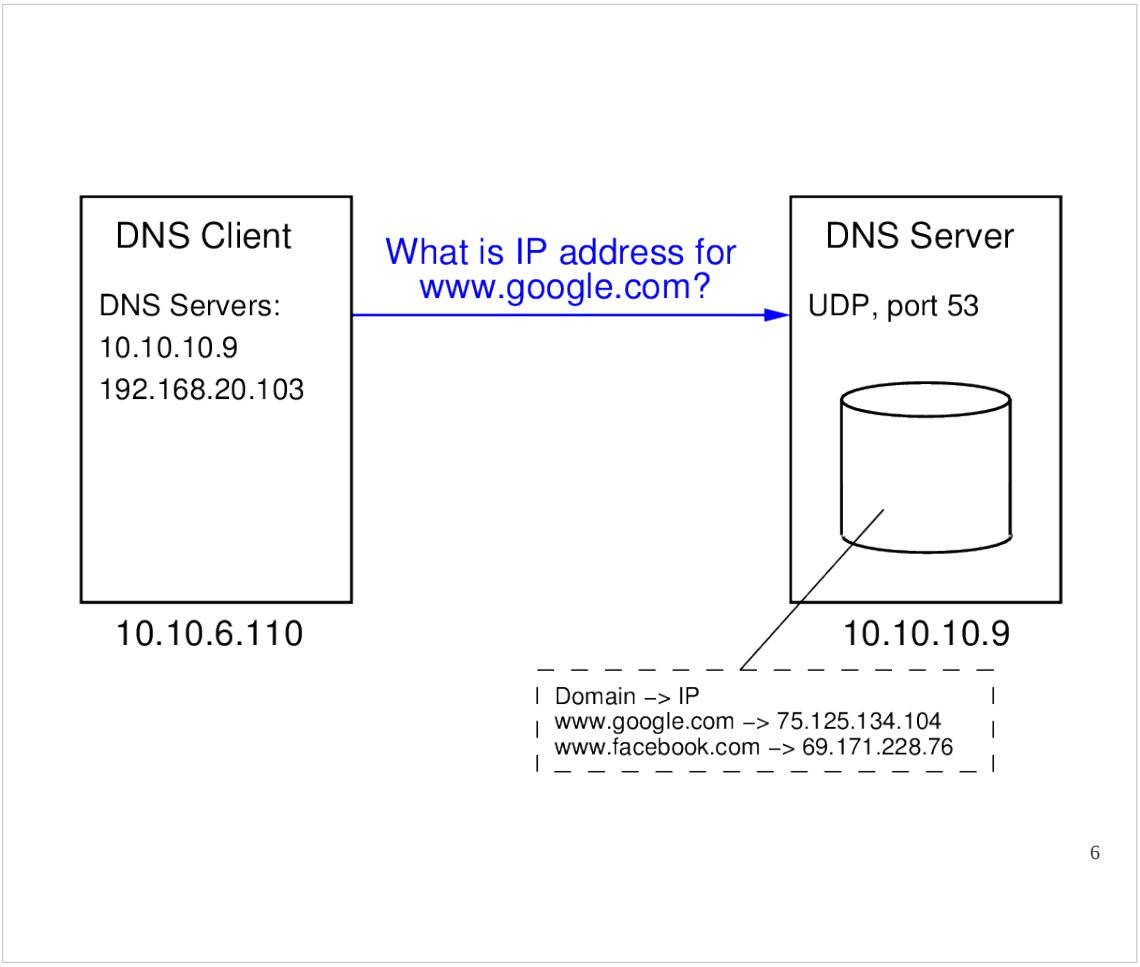


Lets look at a simple example of DNS working. Computer 10.10.6.110 above is an end-user computer running a web browser, as well as DNS Resolver. Assume that the human user has typed http://www.google.com/ into the address bar on the browser. Before a HTTP Request can be sent to the correct computer (the one running the web server for www.google.com), computer 10.10.6.110 needs to map the domain www.google.com to the corresponding IP. The DNS Resolver does this, using the DNS protocol. We will see on the next slides that it will send a query to a DNS server. Which DNS server? The resolver must be configured with a list of 1 or more DNS server IP addresses. In Linux, this is normally stored in the file /etc/resolv.conf.

Assume that there is a computer with IP address 10.10.10.9 running a DNS server. DNS servers normally use port number 53, and UDP as the transport protocol. The DNS server has a database storing mappings from domain to IP. How does it obtain these mappings? Two ways:

• Owners of domains register the mapping from domain to IP in several DNS servers. The DNS servers that store these register mappings are called authoritative servers. For example, Google maintains its own DNS Server, ns.google.com, which is the authoritative server for the domain www.google.com.

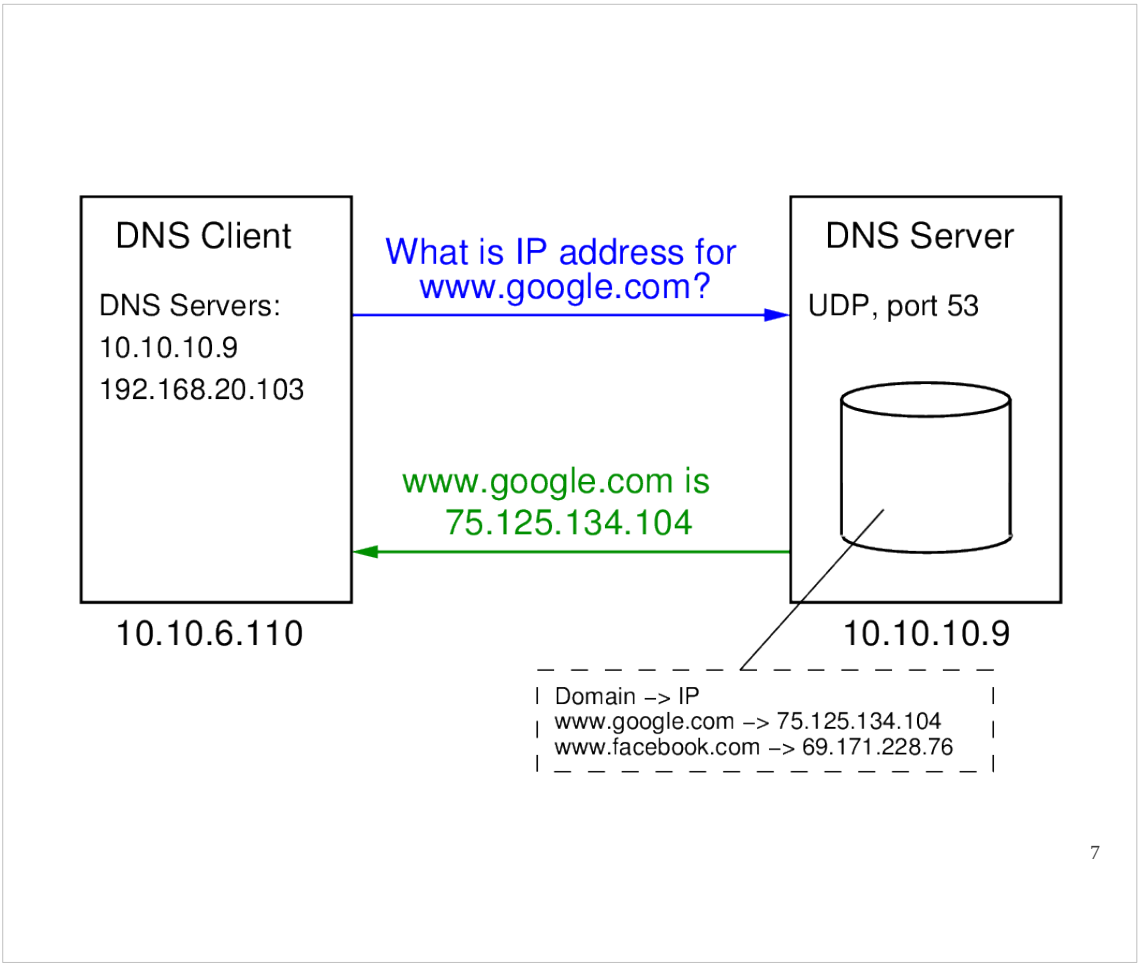
• When DNS servers receive queries for domains they don't yet know about, they send the queries to other DNS servers. When a response comes, the DNS server may cache the mapping of domain to IP. In this case the DNS server is a non-authoratitive server for the domain. In the example above, 10.10.10.9 has learnt the IP addresses for www.google.com and www.facebook.com from previous queries (not shown), and caches the values.



The DNS Resolver on 10.10.6.110 has the domain www.google.com. It wants to know the IP address of the computer that this domain identifies.

The DNS Resolver uses the DNS protocol to send a query to a DNS server. It selects a DNS server from its pre-configured list (in the file /etc/resolv.conf). (Who configured this list? Either the human user of the computer or it was automatically configured with DHCP. Why multiple DNS servers? In case one doesn't respond).

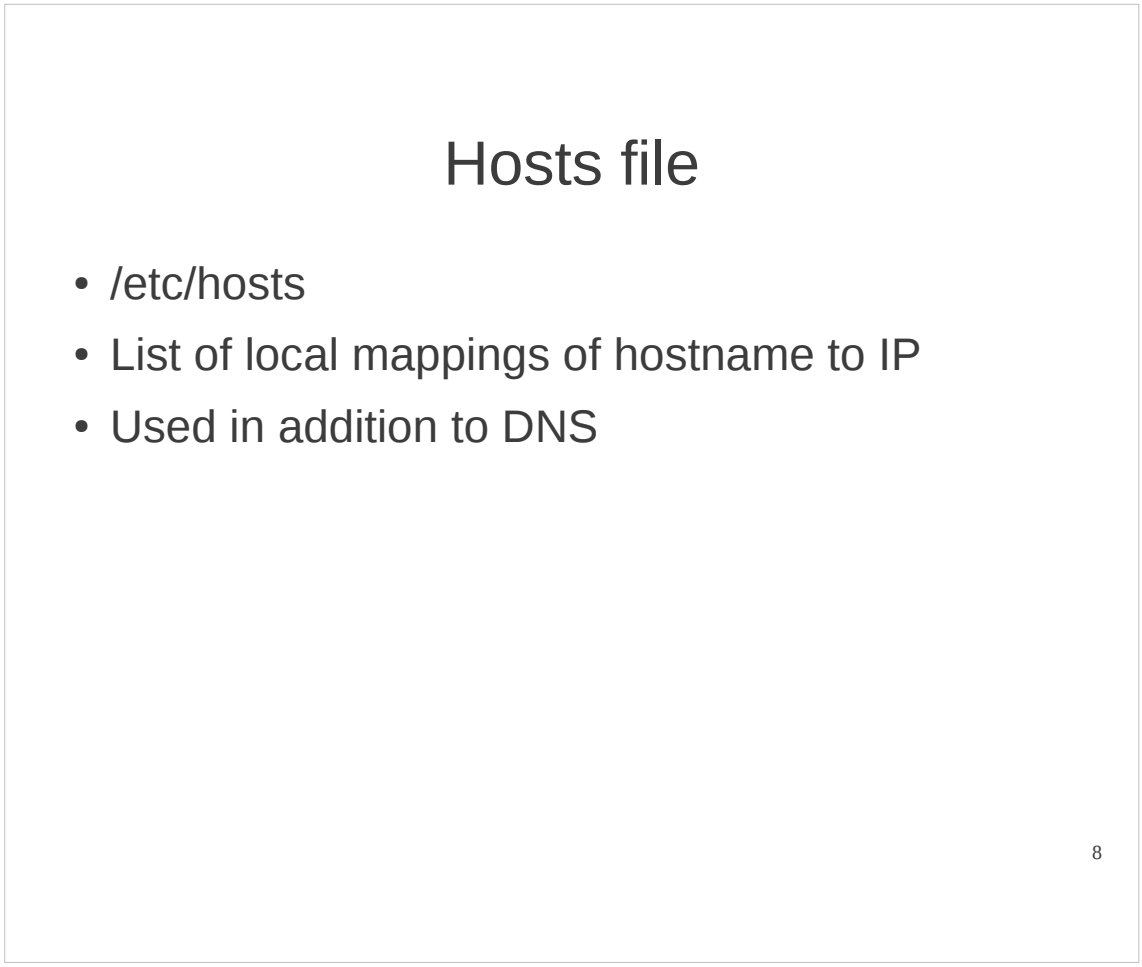
The meaning of the query is effectively: “What is the IP address for the domain www.google.com?”.



When the DNS server receives the query, it checks its database of mappings. In this example, the domain www.google.com is in the database. Therefore the DNS Server responds, indicating the IP address for the computer identified by the domain.

If the domain was not in the database of the DNS Server, the server may contact other DNS servers; there is a hierarchical arrangement of servers to help manage the storage of domains and efficiency in finding domains. If another DNS server sends a response to 10.10.10.9, then 10.10.10.9 may cache the response, and then forward it to the DNS resolver. Subsequent requests for the domain to 10.10.10.9 will then obtain an immediate response.

Once the DNS Resolver receives the response, it knows the IP address of the computer it needs to contact. In our example using HTTP, the IP address of the destination as well as the HTTP Request message are sent to TCP, which then delivers a TCP segment and IP address of the destination to IP. Then the datagram can be sent to the computer hosting the Google web server.



Normally, once the DNS servers are set for a computer (e.g. in /etc/resolv.conf), the human user has no involvement with DNS. It is handled automatically by the operating system.

However most operating systems allow the human user to sent their own, local mappings of domains (specifically hostnames) to IP addresses. In Linux this is down in the /etc/hosts file.

The hosts file is a text file that lists hostnames and IP addresses. It is used in addition to DNS, often consulted before DNS is used. For example, if you add the entry

127.0.0.1 www.google.com

To your hosts file, and then try to visit the Google web server, your computer will send the HTTP request to 127.0.0.1 (instead of the IP address that DNS would normally use).