A DNS Tutorial

http://ntrg.cs.tcd.ie/undergrad/4ba2/multicast/

Table of Contents

What is a DNS? **3**
Why do we need a DNS? 3
Why do computers prefer addresses based on numbers? 3
What is a Domain Name, and how does a computer name look like? 3
How many segments does a Domain Name have? 3

TCP/IP Internet Domain Names **4**
Official And Unofficial Internet Domain Names 5
Items Names and Syntax 6
Mapping Domain Names To Addresses 6

Domain Name Resolution **7**
Efficient Translation 8
Caching: The Key To Efficiency 8

DNS Message Format **9**
Compressed Name Format 10

Abbreviation Of Domain Names **10**

Object Types **11**

Summary **11**

What is a DNS?

The Domain Name System is a system which helps internet users to use the net more easily by allowing them to specify a meaningful names to web sites and/or other users they want to communicate with.

Why do we need a DNS?

When computers talk to each other via the internet, they use the IP protocol. The Internet Protocol distinguish hosts from each other by an IP address which is a string of numbers appended to each other and separated by periods. An example of such a string might be 197.15.3.2.

The IP addresses are unique, which means that each host has its own IP address which is different from all other IP addresses in the world.

However, these IP addresses are hard for us, humans, to remember and we prefer to use meaningful names, which we are used to refer to in the everyday life.

The DNS is needed by applications to convert humans meaningful names into computer meaningful names (IP addresses) and provide the final user an easier way to communicate via the Internet.

Why do computers prefer addresses based on numbers?

Computers prefer number addresses because they perform better with numbers, which require less computation. For example, lets say that an address named 'com' is represented by the number 231. In binary representation, the computer needs at least 3 Bytes to represent the name 'com', because each character is represented with one byte at least. However, representing the number 231 requires only 8 binary bits (a single byte). As a result comparing the name 'com' with other names will require comparing at least 3 bytes while comparing the number 231 will require only a single byte comparison.

What is a Domain Name, and how does a computer name look like?

Each computer name consists of a sequence of alpha-numeric segments separated by periods. For example, a computer name might be:

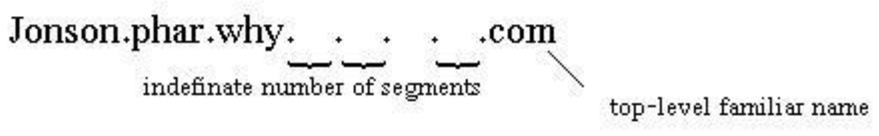
```
www.lingo.com
```

A computer name is also called a 'Domain Name'.

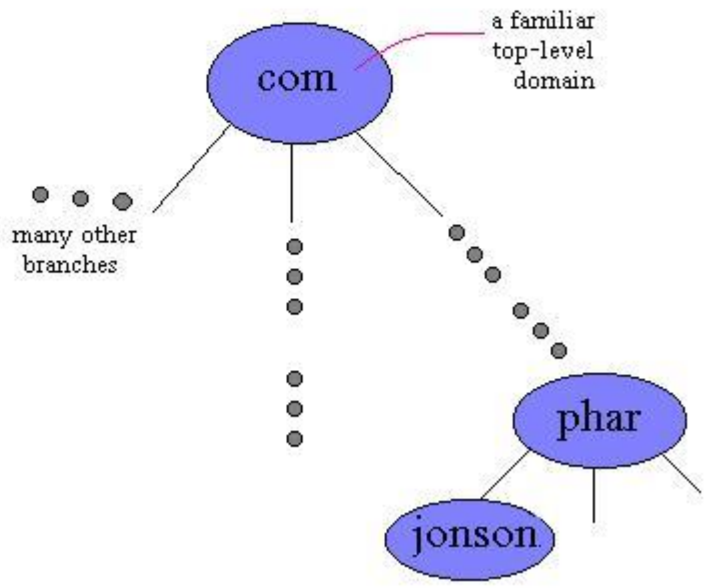
Domain names are hierarchical, with the most significant part of the name on the right. The left-most segment of a name (*www* in the example) is the name of an individual computer. Other segments of the full name identify the group that owns the individual name. In the example, the individual name *www* is owned by the *lingo* group of names, which is itself owned by the *com* group (which holds a lot of company names).

How many segments does a Domain Name have?

The Domain Name System does not specify the exact number of segments in a name. Each organization can choose its own names hierarchy and specify as many segments as it needs. However, all names should be under a familiar top level domains.



A hierarchical illustration of the example above might be:



TCP/IP Internet Domain Names

The mechanism that implements a machine name hierarchy for TCP/IP Internet is called the *Domain Name System* (DNS).

DNS has two independent aspects:

- It specifies the name syntax and rules for delegating authority over names
- It specifies the implementation of a distributed computing system that efficiently maps names to addresses.

The domain name system uses a hierarchical naming scheme known as domain names. A domain name consists of a sequence of sub-names separated by a delimiter character, the period. The domain calls each section a *label*.

The domain name

```
cs.purdue.edu
```

contains three *labels: cs, purdue, and edu.* Any suffix of a label in a domain name is also called a domain. In the above example the lowest level domain is *cs.purdue.edu*, the second level domain is *purdue.edu* and the top level domain is *edu.*

Writing domain names with the local label first and the top domain last makes it possible to compress messages that contain multiple domain names.

Official And Unofficial Internet Domain Names

The domain name standard specifies an abstract hierarchical name-space with arbitrary values for labels. Therefor it is possible for any group that builds an instance of the domain system to choose labels for all parts of its hierarchy.

For example, a private company can establish a domain hierarchy in which the top level labels specify corporate subsidiaries, the next level labels specify corporate divisions, and the lowest level specify departments.

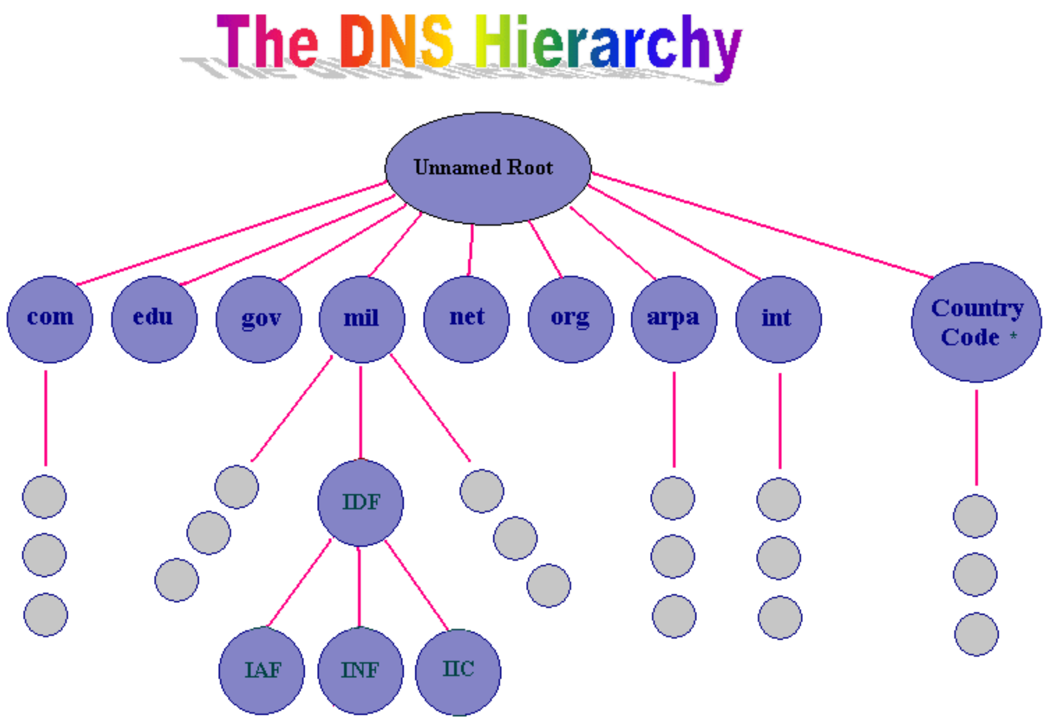
However most users of the domain technology follow the hierarchical labels used by the official Internet domain system. There are two reasons:

- The Internet scheme can accommodate a wide variety of organizations, and allows each group to choose between geographical or organizational naming hierarchies.
- Most sites follow the Internet scheme so they can attach their TCP/IP installations to the global Internet without changing names.

The Internet authority has chosen to partition its top level into the following domains:

Domain Name	**Meaning**
COM	Commercial organizations
EDU	Educational institutions
GOV	Government institutions
MIL	Military groups
NET	Major network support centres
ORG	Organizations other than those above
ARPA	Temporary ARPANET domain
INT	International organizations
Country code	Each country (geographical scheme)

The following is a figure of the DNS top level hierarchy:



A small part of the internet domain name hierarchy tree. In practice, the tree is broad and flat; most host entries appear by the fifth level.
* The hierarchy allows to use a geographic registration. An organization can register its domain under its country code.

The top level names permit two completely different naming hierarchies: geographic and organizational. The geographic scheme divides the universe of machines by country. Each country is assigned its own top level domain with the country's international 2 letter identifier as its label.

As an alternative to the geographic hierarchy, the top level domains also allows organizations to be grouped by organizational type. An organization chooses how it wishes to be registered and request approval. The central authority reviews the application and assigns the organization a sub domain under one of the existing top level domains.

Items Names and Syntax

The domain name system is quite general because it allows multiple naming hierarchies to be embedded in one system. In order to allow clients to distinguish among multiple kinds of entries, each named item stored in the system is assigned a *type* the specifies whether it is the address of a machine, a mailbox, a user, and so on.

When a client asks the domain system to resolve a name, it must specify the type of answer desired. For example, when an electronic mail application uses the domain system to resolve a name, it specifies that the answer should be the address of a *mail exchanger*.

In addition to specifying the type of answer sought, the domain system allows the client to specify the protocol family to use.

The domain system partitions the entire set of names by *class* , allowing a single database to store mappings for multiple protocol suites.

The syntax of a name does not determine what type of object it names or the class of protocol suite. In particular, the number of labels in a name does not determine whether the name refers to an individual object or a domain.

Mapping Domain Names To Addresses

The domain name scheme includes an efficient, reliable, general purpose, distributed system for mapping names to addresses. The system is distributed in the technical sense, meaning that a set of servers operating at multiple sites co-operatively solve the mapping problem. It is efficient in the sense that most names can be mapped locally; only a few require Internet traffic. It is general purpose because it is not restricted to machine names.

Finally, it is reliable in that no single machine failure will prevent the system from operating correctly.

The domain mechanism for mapping names to addresses consists of independent, co-operative systems called *name servers*. A name server is a server program that supplies name-to-address translation, mapping from domain names to IP addresses. The client software, called a *name resolver*, uses one or more name servers when translating a name.

The easiest way to understand how domain servers work is to imagine them arranged in a tree structure that corresponds to the naming hierarchy.

The root of the tree is a server that recognizes the top level domains and knows which server resolves each domain. Given name to resolve, the root can choose the correct server for that name. At the next level, a set of name servers each provide answers for one top level domain (e.g., *edu).* A server at this level knows which servers can resolve each of the sub domains under its domain.

The conceptual tree continues with one server at each level for which a sub domain has been defined. The servers may be located at arbitrary locations on an Internet. The servers uses the Internet for communication.

Domain Name Resolution

Domain name resolution proceeds top-down, starting with the root name server and proceeding to servers located at the leaves of the tree. There are two ways to use the domain server system:

- Contacting name servers one at a time
- Asking the name server system to perform the complete translation

In either case, the client software forms a domain name query that contains the name to be resolved, a declaration of the class name, the type of answer desired, and a code that specifies whether the name server should translate the name completely. It sends the query to a name server for resolution.

When a domain name server receives a query, it check to see if the name lies in the sub domain for which it is an authority. If so, it translates the name to an address according to its database, and appends an answer to the query before sending it back to the client.

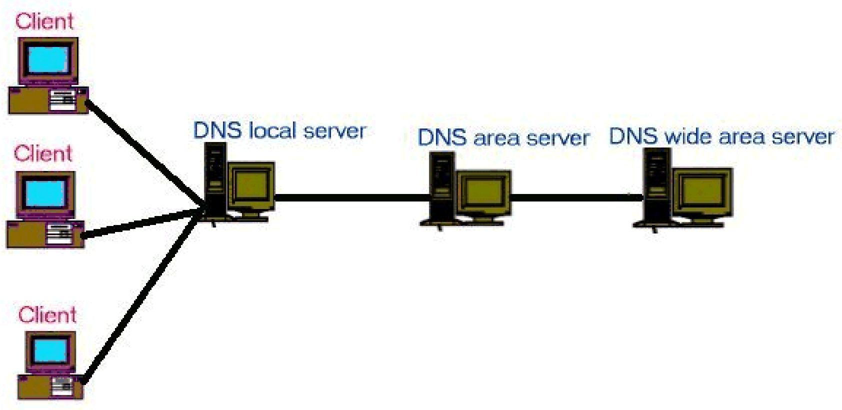
If the name server cannot resolve the name completely, it checks to see what type of interaction the client specified. If the client requested complete translation, the server contacts a domain name server that can resolve the name and returns the answer to the client.

If the client requested non-recursive resolution, the name server cannot supply an answer. It generates a reply that specifies the name server the client should contact next to resolve the name.

A client must know how to contact at least one name server. To ensure that a domain name server can reach others, the domain system requires that each server know the address of at least one root server. In addition, a server may know the address of a server for the domain immediately above it.

Domain name servers use a well-known protocol port for all communication, so clients know how to communicate with a server once they know the IP address of the machine in which the server executes. There is no standard way for hosts to locate a machine in the local environment on which a name server runs; that is left to whoever designs the client software.

In some systems, the address of the machine that supplies domain name service is bound into application programs at compile time, while in others, the address is configured into the operating system at start-up. In others, the administrator places the address of a server in a file on secondary storage.



Efficient Translation

It may seem natural to resolve queries by working down the tree of the name servers, but it can lead to inefficiencies for three reasons.

- Most name resolution refers to local names, those found within the same subdivision of the name space as the machine from which the request originates. Tracing a path to contact the local authority would be inefficient.
- If each name resolution always started by contacting the topmost level of the hierarchy, the machine at that point would become overloaded.
- Failure of machines at the topmost levels would prevent name resolution, even if the local authority could resolve the name.

These ideas lead to a two-step name resolution mechanism that preserves the administrative hierarchy but permits efficient translation.

In the two-step name resolution process, resolution begins with the local name server. If the local server cannot resolve a name, the query must then be sent to another server in the domain system.

Caching: The Key To Efficiency

The cost for lookup for non local names can be extremely high if resolvers send each query to the root server. Even if queries could go directly to the server that has authority for the name, name lookup can present a heavy load to an Internet.

Therefor to improve the overall performance of a name server system, it is necessary to lower the cost of lookup for non local names.

Internet domain name servers use *name caching* to optimize search costs. Each server maintains a cache of recently used names as well as a record of where the mapping information for that name was obtained. When a client asks to resolve a name, the server first checks to see if it has authority for the name according to the standard procedure. If not, the server checks its cache to see if the name has been resolved recently. Servers report cached information to clients, but mark it as a *non authoritative* binding, and give the domain name of the server, S, from which they obtained the binding. The local server also sends along additional information that tells the client the binding between S and an IP address. Therefor clients receive answers quickly, but the information may be out-of-date.

If efficiency is important, the client will choose to accept the non authoritative answer and proceed. If accuracy is important, the client will choose to contact the authority and verify that the binding between name and address is still valid.

Name to address binding changes, therefore to keep the cache correct, servers time each entry and dispose of entries that exceed a reasonable time. Servers do not apply a single fixed time out to all entries, but allow the authority for an entry to configure its time-out. Whenever an authority responds to a request, it includes a *time to live* (TTL) value in the response that specifies how long it guarantees the binding to remain. Authorities can reduce network overhead by specifying long time-outs for entries that they expect to remain unchanged, while improving correctness by specifying short time-outs for entries that they expect to change frequently.

Caching is important in host as well as in local domain name servers. The host downloads the complete database of names and addresses from a local domain server at start-up, maintains its own cache of recently used names, and uses the server only when names are not found. A host that maintains a copy of the local server database must check with the server periodically to obtain new mappings, and the host must remove entries form its cache after they become invalid.

Keeping a copy of the local server's database has several advantages:

- It makes resolution on local hosts extremely fast because no network activity is involved.
- The local site has protection in case the local name server fails.
- It reduces the computational load on the name server, and makes it possible for a given server to supply names to more machines.

DNS Message Format

When the user wants to send a message, it invokes an application program and supplies the name of a machine with which the application must communicate. The application program must find the machine's IP address. It passes the domain name to a local resolver (L.R.) and requests an IP address. The local resolver checks its cache and:

- If the L.R. has an answer, it returns the answer.
- If the L.R. hasn't one, it sends the message to the server. The server then returns a similar message that contains the answer to the questions for which the server has bindings. If the server can't answer, it sends responsive information about other servers that the client can contact.

0 15	16 31	
Identification	Flags	↑
Number of Questions	Number of Answer RRs	12 Bytes
Number of Authority RRs	Number of Additional RRs	↓
Questions		
Answers		
Authority		
Additional Information		

The format of DNS query and Response

The IDENTIFICATION is set by the client and returned by the server.

The 16-bit PARAMETER consists of:

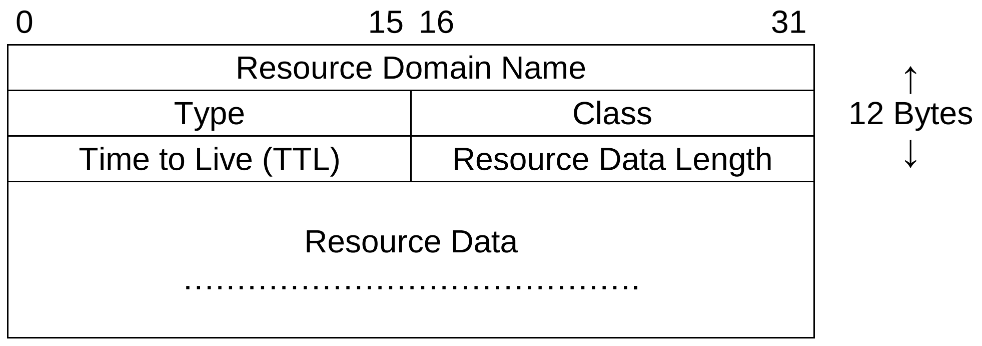
- th bit field -qr: 0 means the message is a query,1 means it's a response.
- 1-4 bit fields - OPCODE:
    - 0 -is a normal value (Standard query).
    - 1 - an inverse query.
    - 2 - the server status request.
- 5-th bit field - Authoritative answer. The name server is authoritative for the domain in the question section.
- 6-th bit field is set if message truncated. With UDP this means that the total size of the reply exceeded 512 bytes, and only the first 512 bytes of the reply were returned.
- 7-th bit field - Recursion Desired. This bit can be set in a query and is then returned in the response.

- 8-th bit field - Recursion Available.
- 9-11 -th bits field has to be 0.
- 12-15 -th bits field - Return Code. 0- no error, 3- name error.

The fields labelled NUMBER OF ... give each a count of entries in the corresponding sections in the message.

The QUESTION SECTION contains queries for which answers are desired. The client fills in only the question section; the server returns the question and answers with its response. Each question has Query Domain Name followed by Query Type and Query Class fields .

ANSWER, AUTHORITY, ADDITIONAL INFORMATION sections consist of a set of resource records that describe domain names and mappings. Each resource record describes one name.



The format of resource records used in later sections of messages returned by Domain Name Server

The RESOURCE DOMAIN NAME contains the destination name, and can be in an arbitrary length. The TYPE field specifies the type of the data record. The CLASS field specifies its class. The TIME TO LIVE field contains an integer that specifies the number of seconds information in this resource record can be cached. It is used by clients who have requested a name binding and may want to cache the results. The RESULTS DATA LENGTH field specifies the count of octets in the RESOURCE DATA field.

Compressed Name Format

In a message , domain names are stored as a sequence of labels.

Each label begins with an octet that specifies its length. The receiver is reading the length and then the label.

DNS might return multiple answers to a query. In order to conserve space in the reply packet, the name servers compress names by storing only one copy of each domain name.

when the client extract the name it must check whether it consist literal string or a pointer to literal string.

Abbreviation Of Domain Names

Abbreviation provides a method shortening names when the resolver can supply part of the name automatically. As in the telephone number hierarchy, when we make a call within the area , we do not dial the area code.

For example , within the school of mathematics in Tel-Aviv University, the abbreviate name **libra** is equivalent to **libra.math.tau.ac.il**

Most client software implements abbreviation with a domain suffix list. The net master configure a list of suffixes that can be appended.

Abbreviations are not part of the DNS but exists in many clients software's to make local names easy for the users.

Object Types

The domain name system can be used for translating a host name to an IP address or a domain name to mail exchanger. The domain name system can be used for arbitrary hierarchical names. One might store names of applications along with mapping to the names and addresses of vendors that offer such applications.

The system accommodates a variety of mappings by including a *type* in each resource code.

A client must specify the type in its query. Servers specify the *type* in all resource records they return.

Domain name system resource record types:

Type	**Meaning**	**Contents**
A	Host Address	32 bit IP address
CNAME	Canonical Name	Canonical domain name for an alias
HINFO	CPU & OS	Name of cpu and os
MINFO	Mailbox info	information about mail box or mail list
MX	Mail Exchanger	16-bit preference and name of host that acts as mail exchanger for the domain
NS	Name Server	Name of authoritative server for domain
PTR	Pointer	Domain name (symbolic link)
SOA	Start of Authority	Multiple fields that specify which parts of the naming hierarchy a server implements
TXT	Arbitrary Text	string of ASCII text

Most of the data is of type A . The second most useful domain type ,MX, is assigned to names used for E-mail. For the mail address user@domain-part, the mail system uses the DNS to resolve *domain-part* with query type MX. For each MX resource record the mailer extracts the domain name and uses type A query.

Summary

The hierarchical naming systems let us use a lot of names without one site of administration.

The DNS offers an hierarchical naming scheme. The DNS uses distributed search in which domain name servers map each name to an IP address. Client begin with the local resolver. If the name is not found the name resolving is done through a tree of servers.