DNS Introduction

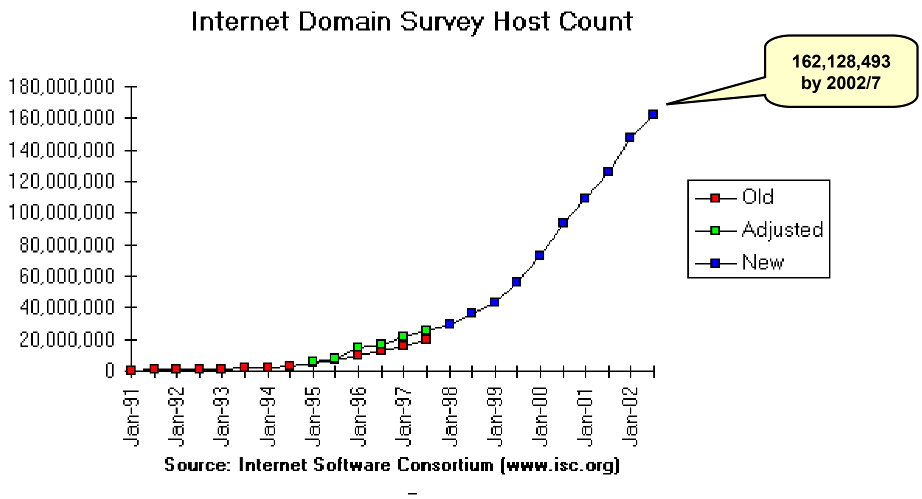
www.what-is-my-ip-address.com

“Except for Great Britain. According to ISO 3166 and Internet tradition, Great Britain's top-level domain name should be gb. Instead, most organizations in Great Britain and Northern Ireland (i.e., the United Kingdom) use the top-level domain name uk. They drive on the wrong side of the road, too.”

DNS and BIND book

Footnote to the ISO 3166 two-letter country code TLDs

DNS Tree Growth



The ISC about the new DNS survey method:

The new survey works by querying the domain system for the name assigned to every possible IP address. However, this would take too long if we had to send a query for each of the potential 4.3 billion (2^32) IP addresses that can exist. Instead, we start with a list of all network numbers that have been delegated within the IN-ADDR.ARPA domain. The IN-ADDR.ARPA domain is a special part of the domain name space used to convert IP addresses into names. For each IN-ADDR.ARPA network number delegation, we query for further subdelegations at each network octet boundary below that point. This process takes about two days and when it ends we have a list of all 3-octet network number delegations that exist and the names of the authoritative domain servers that handle those queries. This process reduces the number of queries we need to do from 4.3 billion to the number of possible hosts per delegation (254) times the number of delegations found. In the January 1998 survey, there were 879,212 delegations, or just 223,319,848 possible hosts.

With the list of 3-octet delegations in hand, the next phase of the survey sends out a common UDP-based PTR query for each possible host address between 1 and 254 for each delegation. In order to prevent flooding any particular server, network or router with packets, the query order is pseudo-randomized to spread the queries evenly across the Internet. For example, a domain server that handles a single 3-octet IN-ADDR.ARPA delegation would only see one or two queries per hour. Depending on the time of day, we transmit between 600 and 1200 queries per second. The queries are streamed out asynchronously and we handle replies as they return. This phase takes about 8 days to run.

See RFC 1296 about details of how traditional DNS surveys were made.

Top Host Names – Worldwide

Top Host Names July 2002

956841	www	3883	venus
336393	mail	3867	dev
56958	cpe	3795	zeus
36107	router	3765	jupiter
35004	ftp	3720	mars
33720	ns2	3656	l0
33128	gw	3647	t3
27548	ns1	3567	www3
23019	pc1	3511	loopback0
21775	pc2	3470	pop
16432	smtp	3452	mercury
15265	pc3	3438	intranet
15177	pc4	3404	demo
14979	broadcast	3397	alpha
14891	pc5	3388	pc13
14877	gateway	3330	pluto
14138	server	3308	exchange
...big gap...		3253	linux
3884	cisco		

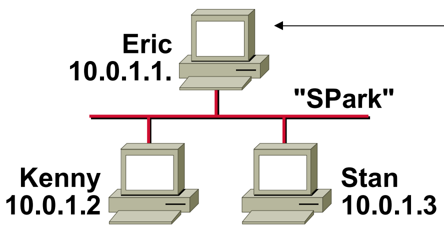
Top Host Names Jan 1992

384	venus	204	mac4	172	mac9
356	pluto	201	hobbes	172	mac11
323	mars	201	hermes	170	mac8
288	jupiter	198	thor	169	phoenix
286	saturn	198	sirius	169	mac12
285	pc1	196	gw	169	hal
282	zeus	195	calvin	168	snoopy
262	iris	194	mac5	168	mac13
260	mercury	191	mac10	167	mac15
259	mac1	190	fred	167	mac14
258	orion	189	titan	167	grumpy
254	mac2	189	pc3	163	gandalf
240	newton	186	opus	162	pc4
234	neptune	186	mac6	160	uranus
233	pc2	185	charon	159	mac16
224	gauss	185	apollo	158	sleepy
222	eagle	179	mac7	158	io
213	mac3	179	athena	157	earth
209	merlin	177	alpha	156	europa
207	cisco	172	mozart	155	rigel

Notice that the people used more fancy names 10 years ago. What can we conclude from this?

History

- **Even in the early Arpanet hosts have been identified by names**
  - **For People, not machines!**
- **Name/Address bindings in HOSTS.TXT files**



`/etc/hosts`

127.0.0.1	eric	localhost
10.0.1.1	eric.spark	eric
10.0.1.2	kenny.spark	kenny
10.0.1.3	stan.spark	stan

(Kenny and Stan have similar hostfiles)

Through the 1970s, the ARPAnet was a small community of a few hundred hosts. A single file called *HOSTS.TXT*, contained a name-to-address mapping for every host connected to the ARPAnet. The familiar UNIX host table, */etc/hosts*, was compiled from *HOSTS.TXT*.

HOSTS.TXT was maintained by SRI's *Network Information Center* ("the NIC") and distributed from a single host, *SRI-NIC*. SRI is the Stanford Research Institute in Menlo Park, California. SRI conducts research into many different areas, including computer networking.

ARPAnet administrators typically emailed any changes to the NIC, and periodically fetched the current *HOSTS.TXT* by FTP. Any changes were compiled into a new *HOSTS.TXT,* typically once or twice a week. The /etc/hosts file which is used by any UNIX host has been generated by using HOSTS.TXT.

Hostfile Problems

- Centrally maintained by Network Information Center (NIC)
- Copied by all hosts
- Scalability problem
- Consistency problem
- Maintenance problem

Unfortunately this approach did not scale as the Arpanet were growing faster and faster. Every additional host not only caused another line in *HOSTS.TXT*, but also produced additional update traffic from and to *SRI-NIC*. Thus the total network bandwidth necessary to distribute
a new version of the hosts file is proportional to the square of the total number of hosts! In these days memory was very expensive and additionally modifying hostnames on a local network became visible to the Internet only after a long (distribution-) delay. Furthermore the name space was not yet hierarchical organized and this "directory" became chaotic.

For example name collisions occurred, that is two hosts in *HOSTS.TXT* could have the same name. While the NIC could assign unique addresses, it had no authority over host names. There was nothing to prevent someone from adding a host with a conflicting name and violating the rules of the name organization. For example if somebody adds a host with the same name as a major mail hub he could disrupt mail service for many users.

The decentralization of administration would eliminate the single-host bottleneck and relieve the traffic problem. And local management would make the task of keeping data up-to-date much easier. It should use a hierarchical name space to name hosts. This would ensure the uniqueness of names.

1984: DNS

- **Paul Mockapetris (IAB) created DNS**
- **Distributed database**
  - **World-wide and redundant**
  - **Maintained by Name Servers**
  - **Simulates hierarchical tree of mnemonic names**
  - **Each domain name is a node in a database**
  - **Goal: Simple "Hostname resolution"**
  - **But also stores other information**

(C) Herbert Haas 2005/03/11 7

Paul Mockapetris, a member of USC's Information Sciences Institute, was responsible for designing the architecture of the new system. In 1984, he released RFCs 882 and 883, which describe the Domain Name System. Later. these RFCs were superseded by RFCs 1034 and 1035, the current specifications of the Domain Name System. RFCs 1034 and 1035 have now been augmented by many other RFCs, which describe potential DNS security problems, implementation problems, administrative gotchas, mechanisms for dynamically updating name servers and for securing domain data, and much more.

A few RFC example about basic DNS concepts:

- RFC 1034: Domain Names - Concepts and Facilities
- RFC 1035: Domain Names - Implementation and Specification
- RFC 1713: Tools for DNS debugging
- RFC 1032: Domain Administrators Guide
- RFC 1033: Domain Administrators Operations Guide

The basic idea was simply to "split the HOSTS.TXT file is into thousand of fragments". DNS "replaces" the hostaddress to a human readable format and enables a mapping between names and addresses (and many other types of information).

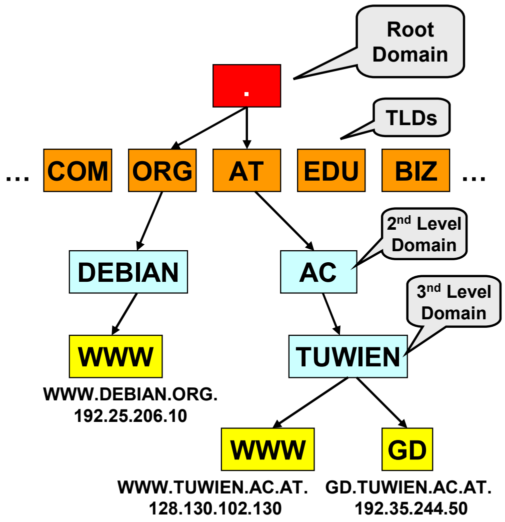
Note: Domain Names are just indexes of the database, which may store whatever information—not only IP addresses!

The domain system is also important to forward emails. There are entry types to define what computer handles mail for a given name, to specify where an individual is to receive mail, and to define mailing lists.

But most often it is only used for "hostname resolution", that is, finding an IP address for a given domain name.

Logical Tree of Names

- IP net-IDs are "flat"
  - Arbitrary assignment without semantical or logical considerations
  - Hard to remember
- DNS maps addresses to names
- DNS allows hierarchical tree of names
  - No name collisions anymore!
  - Max 127 levels
  - Concatenation results in Fully Qualified Domain Name (FQDN)



(C) Herbert Haas 2005/03/11 8

Note that IP **network** addresses are **flat**. Although we often call IP addresses structured, the net-IDs are indeed flat, that is, they have no further structure.

Moreover, IP address assignment had been done rather arbitrary without taking semantic or logical considerations into account. But what's most important: people cannot easily remember a 32 byte decimal number by heart.

The DNS maps the whole "flat" IP address space into a logical and **hierarchical tree** of **names**. The tree origins at the **root** domain, which is represented by a single dot "**.**", while all other domains—first level domains, second level domains, and so on—are attached below the root. The first level domains are also known as "Top Level Domains" (TLDs).

The leaves of this tree and each node in between can be specified by concatenating all names from here to the root. This is called a "**Fully Qualified Domain Name**" (**FQDN**).

Note: This tree does not reflect any physical or geographical location of hosts! For example ten different hosts might be physically located in different networks and each in a different country, but all can belong to the same domain!

Name Servers

- The DNS tree is realized by Name Servers
- The Domain Name Tree does NOT reflect the physical network structure!
- Each NS cares for a subset of the DNS tree: zones
- Flexible mappings
  - 1:n (Routers or servers with several network interfaces)
  - n:1 (Multiple services behind a single IP address)

(C) Herbert Haas 2005/03/11 9

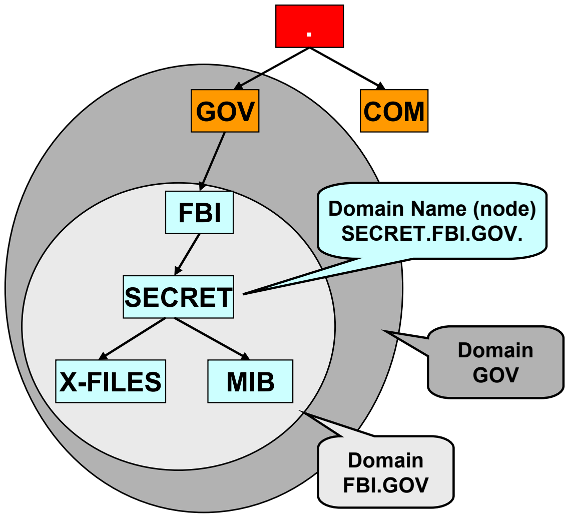
How is this hierarchical tree implemented? All information is stored in world-wide **distributed name servers**, each of which knows only a fragment of course. This fragment is called a "**zone**" information. A "zone" is simply a part of the tree or a subdomain. Zones are explained later in more detail.

DNS allows flexible mappings between addresses and names—they do not need to be one-to-one! For example a router might be known by a unique name but is reachable by **multiple addresses** because it employs a number of interfaces.

Furthermore, a workstation might offer different services such as FTP, HTTP, MAIL, and so on, and each service is identified by a separate name, for example ftp.x.y.z, www.x.y.z or mail.x.y.z. These mappings are implemented by so-called **aliases**.

Terminology

- A "Domain" is a subtree of the domain name space
- A "Domain Name" is the name of a node in the tree
  - Concatenated labels from the root to the current domain
  - Listed from right to left
  - Separated by dots
  - Max 255 characters
- A "Label" is a component of the domain name
  - Max 63 characters



A "Domain" is everything under a particular point in the tree and relates to the naming structure itself, not the way things are distributed.

A "**Domain Name**" is the name of a node in this tree—the index of the database. It consists of all concatenated labels from the root to this node and must not exceed 255 characters.

Thus a domain name is made up of several "**Labels**", which need only be unique at a particular point in the tree. That is, both "name.y.z" and "name.x.y.z" are allowed. Labels must not exceed 63 characters.

Note that DNS is not case sensitive—although DNS originates from UNIX systems. That is, "www.nic.org" is the same as "WWW.NIC.ORG"

Due to SMTP restrictions, domain names may contain only characters of the following sets: {a-z}, {A-Z}, {0-9}, and the dash character "-". Additional language specific characters might be supported in future implementations.

The Root Domain

- The root of the DNS tree is represented as a dot "."
  - A true FQDN includes the dot
  - Otherwise "relative" domain name
  - Most people/applications don't care
  - However, DNS does care!
- The root is implemented by several root-servers (currently 13)
- Below the root, a domain may be called top-level, second-level, third-level etc...

The root domain "." is always the rightmost "label" of a FQDN, although most applications such as web browsers do not care about it. However, any DNS configuration is absolutely sensitive of the proper use of this dot. Any domain name without the root-dot is regarded as relative domain name.

The root is realized by **13 root servers** (as of 2002) which are world wide dispersed for performance and redundancy reasons.

However…9 root name servers are indeed located in the USA…

Top Level Domains

- **Seven "generic domains" (gTLDs)**
  - COM, EDU, GOV, INT, ORG, MIL, NET
  - Initially inside USA, now globally used
- **244 Two-letter country codes**
  - E.g. AT, DE, UK, ES, RU, CH, IT, AQ, …
  - Initially outside USA only, now also "US"
  - Country code does not necessarily reflect real location!
- **Seven new TLDs**
  - BIZ, INFO, NAME, MUSEUM, COOP, AERO, PRO

 12

The Arpanet defined seven generic top level domains, short gTLDs, which were originally only assigned inside the USA.

com	Commercial
edu	Educational
org	Non Profit Organizations (NPOs)
net	Networking providers
mil	US military (e. g. navy.mil, army.mil)
gov	US government organisations (e. g. nasa.gov, nsf.gov)
int	International organizations

In 1996, the restrictions for gTLDs have been relaxed (except mil and gov), for example even commercial organisation can use the net and org TLDs. Additionally the two letter coutry code, which is defined in defined in ISO-3166 is also used. Currently, there are 244 country specific registries.

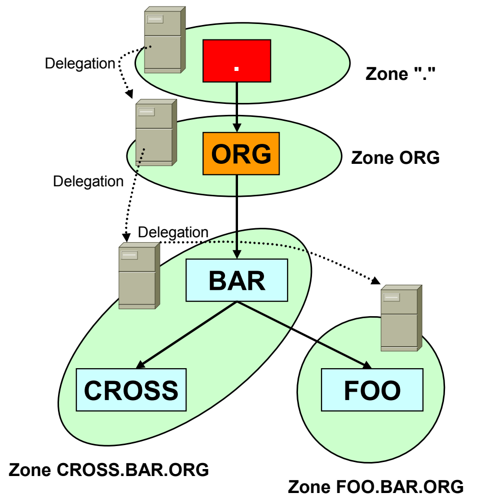
Also new TLDs have been introduced recently: **AERO** for the airport industry, **BIZ** for businesses, **COOP** for "Cooperatives", **INFO** for unrestricted use, **MUSEUM** for (*surprise*) museums, **NAME** for individuals, and **PRO** for "Professionals" such as accountants, lawyers, physicians, and so on.

The **us** domain has fifty subdomains that correspond to the fifty U.S. states. Each is named according to the standard two-letter abbreviation for the state which has been defined by the U.S. Postal Service.

Country and state domains typically reflect geographical locations—but not necessarily!

Delegation and Zones

- **To ease administration, the authority over subdomains is delegated to other nameservers**
- **A zone is a point of delegation or "Start of Authority" (SOA)**
- **Zones relate to the way the database is partitioned and distributed**



(C) Herbert Haas 2005/03/11 13

Obviously root and TLD name servers cannot hold all information about a domain, and even many organizations are as big that it is not reasonable to maintain a whole domain database at a single server. Because of this, administration is simplified by delegating the authority of a subdomain—also called a zone—to another nameserver.

That is: name servers generally deal with zones—not domains!

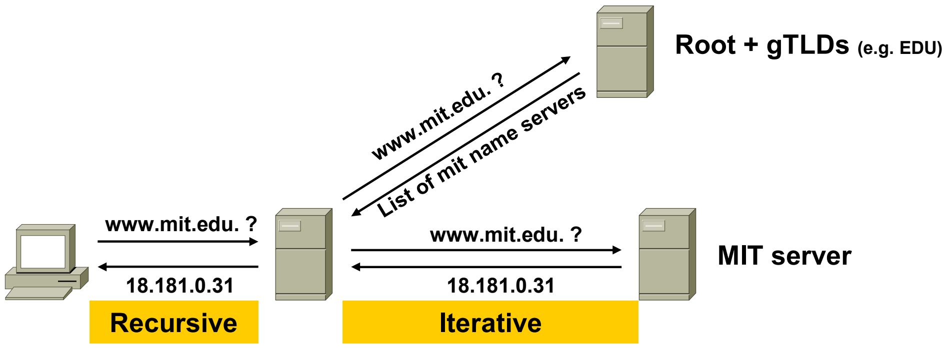
The so-called "**Start of Authority**" (**SOA**) record of a name server specifies the realm of the particular zone. Or in simpler words: Each name server stores information about a zone and each zone is therefore a "Start of Authority".

A zone can span over a whole domain or just be part of it. In this case a zone is like a **pruned domain**. It contains all names from this point downwards the domain-tree except those which are delegated to other zones (i.e. to other name servers).

Also the org name servers control different zones. Imagine if a root name server loaded the root domain instead of the root zone: it would be loading the entire name space! Now if a name server is asked for data in some subdomain, it can reply with a list of the right name servers to talk to.

Hostname Resolution

- **Recursive queries = the job is forwarded**
  - The response must be exact (or error message)
  - Most burden on next name server
- **Iterative queries = All NS are queried top-down**
  - The response contains best answer already known
  - Requested name server makes no further queries



There are two ways for hostname resolution: **recursive** and **iterative** queries.

The recursive query is more **burdening** for the server which is being queried, because this server is asked to do the whole job of name resolution by its own. Of course it may also forward this query and the next server must perform the whole work.

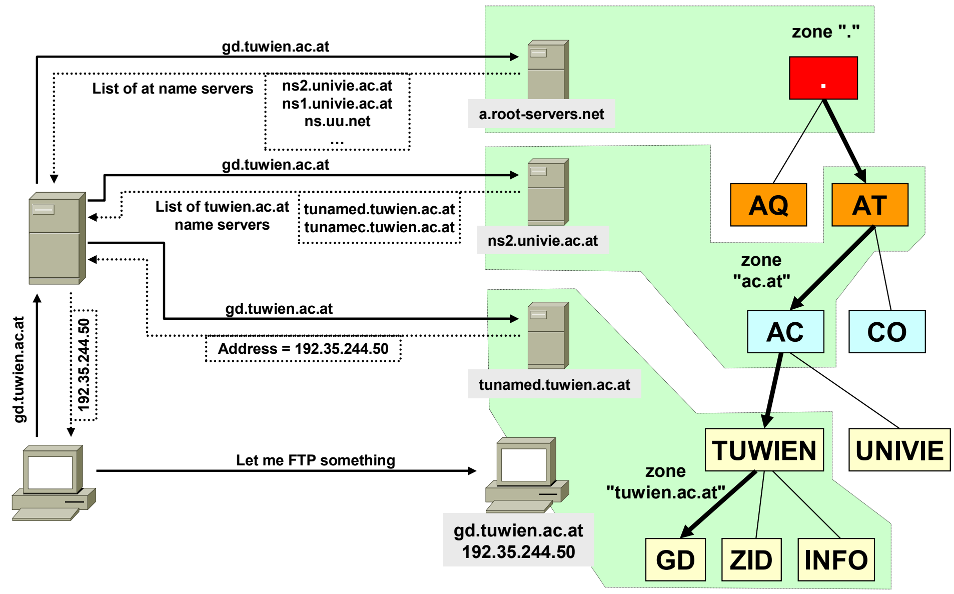
A name server configured to forward all unresolved queries to a designated name server is called a "**forwarder**".

Being requested by an iterative query is much easier. The queried server only has to reply using the best information **currently known**. If the queried name server isn't authoritative for the data requested, the initiator will have to query other name servers to find the answer.

Most name servers will recurse, since this permits them to *cache* the various resource records used to access the foreign domain, in anticipation of further similar requests.

The BIND 8 name server can be configured to refuse recursive queries.

A Detailed Real-World Example



(C) Herbert Haas 2005/03/11 15

The diagram above shows a real world example of name resolution, starting at the root name servers. Of course not any request needs to start at the root since most ISP name servers cache a lot of information or know at least the addresses of authoritative name servers.

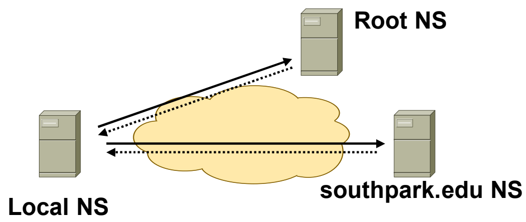
But in our example we start at the top. The whole process can be verified by using standard DNS tools such as **dig**.

N e

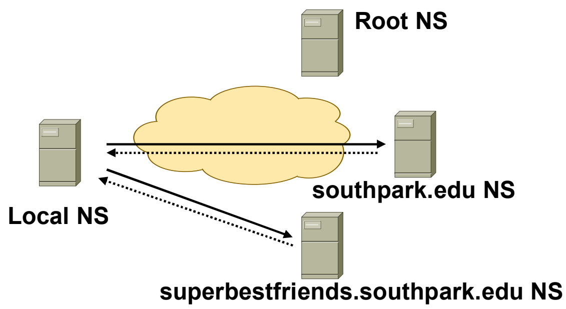
- **Each questioned name server replies with more detailed information…or the desired information itself!**
- **A reference to another NS gives precious information about new zone authority – cached!**

After a reference to another NS, this (recursing) NS learns the IP address of a new NS which is authoritative about a new zone. This is precious information and therefore it is cached.

Caching



- First, the local NS resolves the name kenny.southpark.edu
- Hereby it learns also the addresses of the southpark.edu NS
- All this information is cached!



- When resolving the name seamen.superbestfriends.southpark.edu the local NS notices that this name is member of southpark.edu
- Address of southpark.edu NS is cached
- No need to start at root NS!

Caching greatly reduces the resolving duration and unburdens the root name servers.

Reverse Lookups

- Very often reverse lookups are necessary
  - "Have address but want name"
  - For logging purposes or service restriction
- Therefore the in-addr.arpa domain was created
  - Given an IP-address the associated hostname can be found
  - Otherwise an exhaustive search in the domain space would be necessary to find any desired hostname

Reverse lookups are commonly used by WWW servers to log its users in a file or IRC servers that want to restrict their service to a certain domain, for example a closed discussion group exclusive for IEEE.ORG members.

In order to support reverse lookups the old **arpa** domain is reused today, which is connected to the **in-addr subdomain**. All IP addresses are attached as labels to in-addr.arpa.

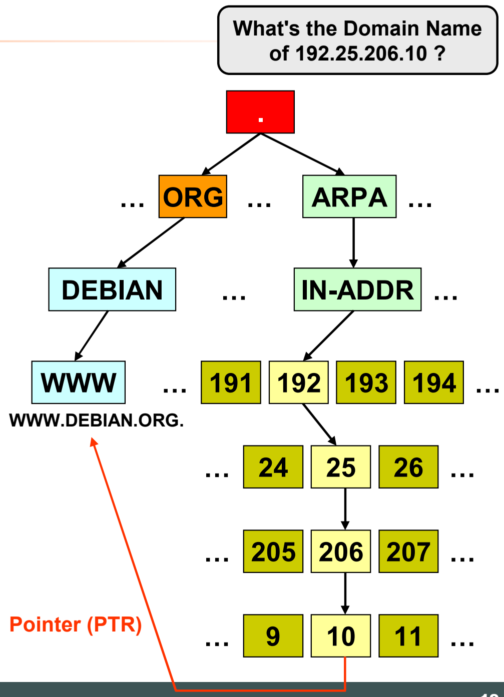
The **ARPA TLD** was originally only used while changing from HOSTS.TXT to DNS. All hosts were originally members of the arpa domain, then all hosts moved to the specific TLDs. Today ARPA is reused for inverse lookups.

Reverse delegation is becoming increasingly important as organizations attempt to verify the origin of requests to their servers by looking up the domain name associated with the IP address making the request. Among the services this applies to are FTP and mail.

Customers may not be able to access services if reverse lookup on their host IP numbers is not setup.

In-Addr.Arpa

- Each byte of an IP address is treated as label and attached under the in-addr.arpa TLD
  - Expressed as character string for its decimal value ("0" - "255")
- Labels are concatenated in reverse order
  - "10.206.25.192.in-addr.arpa"



The "in-addr.arpa." domain is the reverse tree for IPv4 addresses. The name derives from "Inverse (IP) address", and "ARPA" was once of the organizations behind the creation of the Internet.

The whole IP address space is represented as a four-level tree which is attached to the in-addr.arpa domain name. Each byte of the IP address is interpreted as ordinary label, allowing normal lookups. But at the leaves of this tree a **pointer** (**PTR**) is found, which points to the official domain name of this host.

For simplicity the domain names should be organized on byte boundaries, however, today tricks are used to assign even names for subnets that are not aligned on byte boundaries.

This is called the "**classless in-addr**" trick and is not discussed here. Hint: Just introduce an artificial 5th level in the tree…

BIND

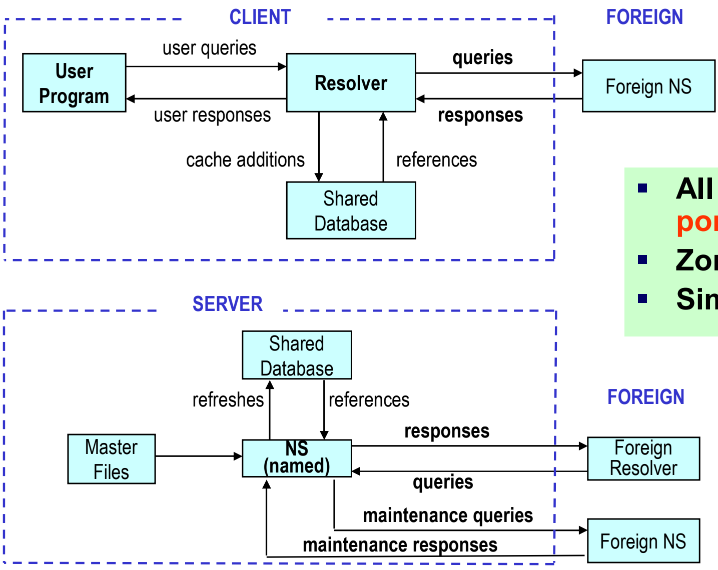
- Berkeley Internet Name Domain (BIND)
  - Implemented by Paul Vixie as an Internet name server for BSD-derived systems
  - Most widely used name server on the Internet
  - Version numbers: 4 (old but still used), 8, 9
- BIND consists of
  - A name server program "named"
  - A resolver library for client applications
- BIND deals with zones!

The most important implementation for DNS is the Berkeley Internet Name Domain (BIND), which has been created by Paul Vixie. BIND consists of a server (**named**, "d" stands for "daemon") and a client, the **resolver library**.

The "resolver" is a collection of functions like **gethostbyname(2)** and **gethostbyaddr(2)** and is used by all Internet applications, such as Telnet, FTP, webbrowsers, and others.

By the way: Windows systems use their own DNS implementation based on BIND.

Resolver and Name Server



- All DNS messages use port 53
- Zone transfers use TCP
- Simple queries use UDP

The diagram above shows the **principle** design of a DNS server and resolver according to the **IETF**.

All DNS messages use port **53**. Zone transfers use TCP for reliability and simple queries which are originated from clients use UDP for speed.

Note that replies that are longer than **512 bytes** (check the implementation!) might also be send via TCP.

Early (up to version 4) BIND implementations did not cache query responses. All modern DNS implementations do cache—unless disabled.

Types of Name Servers

- **Primary Masters** (or "Master")
  - Has data about a zone in a local file
  - Therefore is authoritative about a zone
  - Each zone has exactly one Primary
- **Secondary Masters** (or "Slave")
  - Copies zonefiles from a Master Server (P or S)
  - This is called "zone transfer" (TCP)
  - Therefore also authoritative
  - Each zone must have at least one Secondary

There are exactly two types of name servers which are authoritative for a zone: the **Primary** Master and the **Secondary** Master. With BIND 8/9 the terms **master** and **slaves** are used instead.

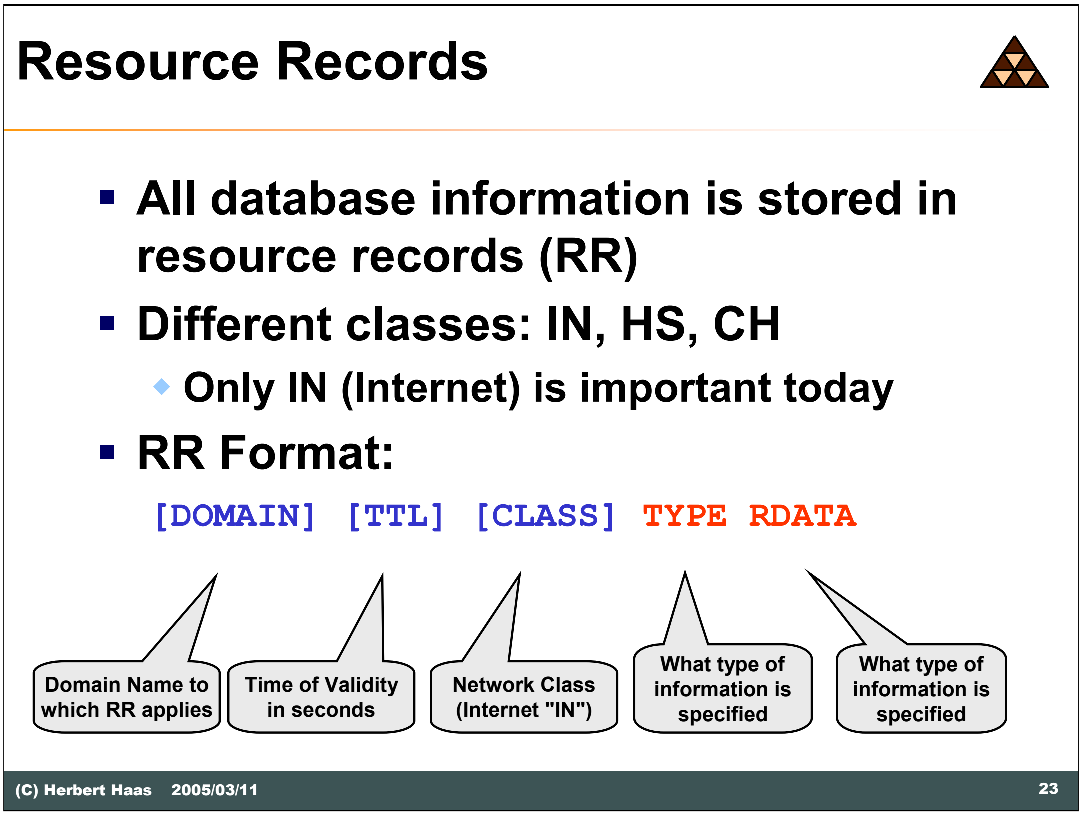
Each zone must have exactly one primary name server. All configuration is done in the master files or "zone files" of the primary.

A secondary name server for a zone gets the zone data from another name server that is authoritative for the zone, called its master server. The master server is either a primary or a secondary name server. When a secondary starts up, it contacts its master name server and, if necessary, pulls the zone data over. This is referred to as a **zone transfer**.

Note that secondary name servers are not second-class name servers. DNS provides these two types of name servers to make administration easier. Just configure set up a primary master name server and specify some secondaries. Once they are set up, the secondaries will transfer new zone data when necessary.

A name server can be a primary master for one zone and a secondary for another, hereby providing enough redundancy to tolerate failures.

Secondary NS are initially suggested in RFC 1035.



Records are divided into **classes**, each of which defines various information types. Today only the Internet class (**IN**) is important, while the others--Chaosnet (CH) and Hesiod (HS)—have only historic significance, and has been used at the MIT.

Within a class, records also come in several types, which correspond to the different varieties of data that may be stored in the domain name space.

All DNS operations are formulated in terms of **Resource Records** (**RRs**, RFC 1035), furthermore, each query is answered with a copy of matching RRs. RRs are the smallest unit of information available through DNS

Some Important RR Types

Type	Value	Meaning
A	1	Host address
NS	2	Authoritative name server
CNAME	5	Canonical name for an alias
SOA	6	Marks the start of a zone of authority
WKS	11	Well known service description
PTR	12	Domain name pointer
HINFO	13	Host information
MINFO	14	Mailbox or mail list information
MX	15	Mail exchange
TX	16	Text strings

The table above shows some important RR types. Most important is the **address** (**A**), which specifies an IP address for address resolution, and the **name server** (**NS**), which specifies other name servers, authoritative for another zone. NS records are used for delegations and constitute the "glue" of the hierarchical tree.

CNAME entries are used to assign a certain host an alias, for example "WWW".

SOA marks the "Start of Authority" and is used as a preamble in each zone file.

Pointer (PTR) records are used for inverse queries, and Mail Exchange (MX) entries are used to specify mail transfer agents, which are responsible to forward mails.

Root Servers

- 13 root servers implement the "."
  - Maintained by ICANN
  - Each of them knows all TLD name servers
  - Most are even authoritative for the generic top-level domains
- Name Servers must maintain a list of root servers
  - Stored in "root.hints" file (BIND)
  - Queried one after the other until positive reply
  - This list is also updated by requesting single root servers

In the absence of other information, resolution has to start at the root name servers. The Internet has **thirteen** root name servers (as of this writing) spread across different parts of the network. Two are on the MILNET, the U.S. military's portion of the Internet; one is on SPAN, NASA's internet; two are in Europe; and one is in Japan. Clearly when root servers go offline there is no name resolution anymore, therefore redundancy is cruxial.

Each root server might be implemented by several physical servers. The utilization of these root servers varies from some kbit/s (rarely) to some Mbits/s (average) up to 100 Mbit/s and more (peaks). Root servers are typically connected to several ISPs, some of them provide free transit service. Furthermore, a backup power supply is needed e.g. battery and generator in the case of outage of commercial power supplies.

Server	Operator	Cities
A	VeriSign Global Registry Services	Herndon VA, US
B	Information Sciences Institute	Merina Del Rey CA, US
C	Cogent Communications	Herndon VA, US
D	University of Maryland	College Park MD, US
E	NASA Ames Research Center	Mountain View CA, US
F	Internet Software Consortium	Palo Alto CA, US - San Francisco CA
G	U.S. DOD Network Information Center	Vienna VA, US
H	U.S. Army Research Lab	Aberdeen MD, US
I	Autonomica	Stockholm, SE
J	VeriSign Global Registry Services	Herndon VA, US
K	Reseaux IP Europeens	London, UK
L	IANA	Los Angeles CA, US
M	WIDE Project	Tokyo, JP

Root Hints Example

.	604800	IN	NS	G.ROOT-SERVERS.NET.
.	604800	IN	NS	K.ROOT-SERVERS.NET.
.	604800	IN	NS	H.ROOT-SERVERS.NET.
.	604800	IN	NS	A.ROOT-SERVERS.NET.
.	604800	IN	NS	B.ROOT-SERVERS.NET.

root — Internet — Name servers

G.ROOT.SERVERS.NET.	604800	IN	A	192.112.36.4
K.ROOT.SERVERS.NET.	604800	IN	A	193.0.14.129
H.ROOT.SERVERS.NET.	604800	IN	A	128.63.2.53
A.ROOT.SERVERS.NET.	604800	IN	A	198.41.0.4
B.ROOT.SERVERS.NET.	604800	IN	A	128.9.0.107

TTL [s] — Address

The slide above shows an actual example (fragment) of a **root.hints** file. Note the five-column specifications of each entry, which is typical for all DNS entries.

From left to right each line specifies:

1. The domain for which this information applies
2. The TTL in seconds, how long this entry is valid if it is cached
3. The network class (almost always IN for Internet)
4. The type of entry—here NS for Name Server and A for Address
5. The data itself, whose meaning has been specified by column four (type)

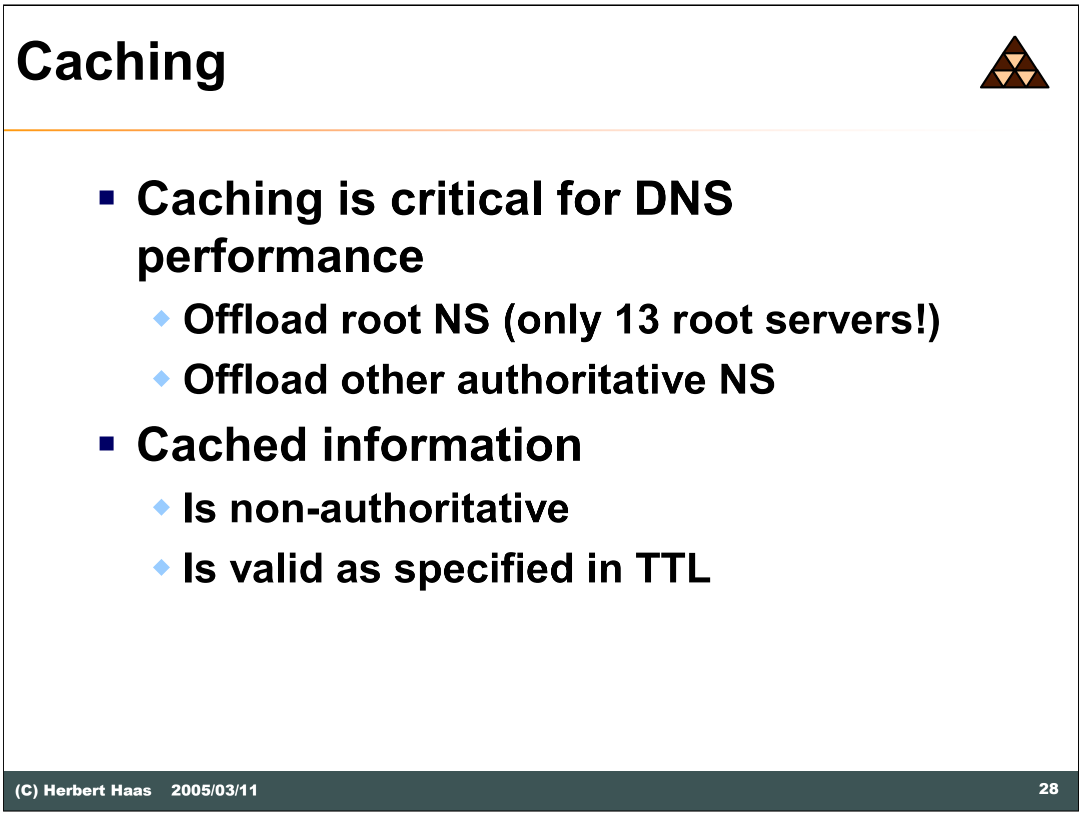
Behind the Scenes

- Frequently private root servers are used within organizations
  - Isolated from official DNS
- Recently several unofficial "roots" were available in the Internet
  - Overlaps official DNS and introduces new unofficial TLDs
- Now ICANN is responsible for managing and coordinating the DNS to ensure universal resolvability
  - ICANN: Global, NPO, public interest
  - Cares for distribution of unique IP addresses and domain names

Current efforts of the ICANN is to assure one **single root** for the Internet. Recently unofficial root name servers "polluted" the Internet with non officially registered TLDs and caused wrong hostname resolutions.

Some companies persuade their users to have their resolvers point to their alternate root instead of the authoritative root.

Others (New.net for example) create special browser plug-ins and other software workarounds to accomplish the same effect.



A name server processing a recursive query discovers a lot of information about the domain name space as. Each time it is referred to another list of name servers, it learns that those name servers are authoritative for some zone, and it learns the addresses of those servers.

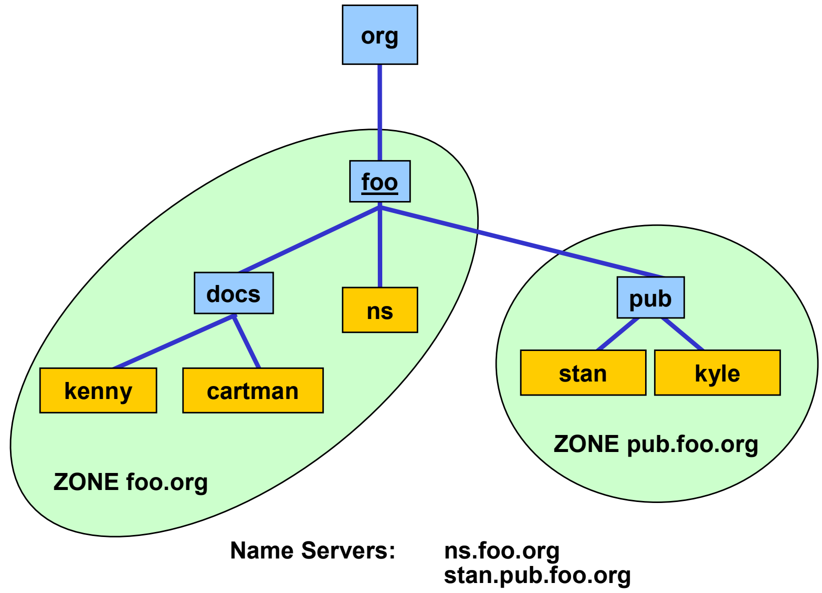
In order to accelerate future client request and to reduce DNS traffic, all these information is cached.

With version 4.9 and all version 8 BINDs, name servers even implement **negative caching,** that is, if an authoritative name server responds to a query with an answer that says the domain name or data type in the query doesn't exist, the local name server will temporarily cache that information, too.

Every piece of DNS information has a **Time To Live** (**TTL**) assigned which specifies the number of seconds this information may be cached before it must be discarded. With BIND 8.2 the TTL is only used for negative caching.

Deciding on a TTL is essentially deciding on a trade-off between performance and consistency. A small TTL ensures that data is consistent across the network, because remote name servers will time it out more quickly and be forced to query authoritative name servers more often for new data. On the other hand, this will increase the DNS traffic and processing load and lengthen the resolution time on the average.

Example Config (1)



The picture above shows an example domain "foo" having two name servers "ns" and "stan", each responsible for another zone. Ns is authoritative for the foo.org zone and stan is authroitative for the pub.foo.org zone.

The following slides show example BIND configurations.

Example Config (2)

```
; zone file for the foo.org. zone
@            IN     SOA  ns.foo.org.        admin.kenny.docs.foo.org (
                       199912245            ;serial number
                         360000             ;refresh time
                         3600               ;retry time
                         3600000            ;expire time
                         3600               ;default TTL )
             IN     NS   ns.foo.org.
             IN     NS   ns.xyz.com.        ;secondary nameserver for @
             IN     MX   mail.foo.org.      ;mailserver for @
Pub          IN     NS   stan.pub.foo.org.
; glue records
ns           IN     A    216.32.78.1
stan.pub     IN     A    216.32.78.99
; hosts in the zone foo.org
Mail         IN     A    216.32.78.10
Linus        IN     A    216.32.78.20
kenny.docs IN       A    216.32.78.100
cartman.docs  IN    A    216.32.78.150
```

Records describing zone .foo.org. = @

Delegation for the zone pub.foo.org.

The slide above shows an example configuration in the ns.foo.org name server. Note the "**glue records**" that assign IP addresses to the NS records so that delegations make sense.

Timers in the SOA RR

- **Refresh time**
  - Tells slave at which time intervals it should check for zone changes
  - Some hours (3-12 typically)
- **Retry time**
  - If master could not be reached
  - Typically shorter than refresh time
- **Expire time**
  - Time after which unrefreshed zone data is definitely outdated (removed)
  - Typically one week (also months)
- **TTL**
  - BIND pre 8.2: Specifies how long any cached entry is valid
  - BIND 8.2 and later: Only valid for negative caching!
  - Performance versus consistency!

Before BIND 8.2 all these values were configured in **seconds**. Post BIND 8.2 releases also allow time values in **hours**, **minutes**, **days**, and **weeks** (h, m, d, w).

When the TTL expires the DNS must remove the respective entries from the cache. This is also true for negative data ("negative caching").

This is different with BIND 8.2 and later: The TTL is actually a "Negative Caching TTL" and is only valid for the negative caches.

Example Config (3)

```
; zone file for the 78.32.216.in-addr.arpa domain
@      IN SOA   ns.foo.org    admin.kenny.docs.foo.org.
                             (
                             1034
                             3600
                             600
                             3600000
                             86400
                             )
       IN NS       ns.foo.org.
1      IN PTR      ns.foo.org.
10     IN PTR      mail.foo.org.
20     IN PTR      linus.foo.org.
99     IN  PTR     stan.pub.foo.org.
100    IN PTR      kenny.docs.foo.org.
150    IN PTR      cartman.docs.foo.org.
```


The slide above shows an example configuration in the ns.foo.org name server, used for **inverse** resolution.

Note the PTR entries.

Example Config (4)

```
; zone file for pub.foo.org
@           IN  SOA  stan.pub.foo.org   hostmaster.stan.pub.foo.org.
                            ( 1034
                            3600
                            600
                            3600000
                            86400 )
; Name Servers
            IN  NS          stan
            IN  NS          ns.foo.org.  ; secondary NS
; glue records
   stan     IN  A           216.32.78.99
```

```
nameserver  IN    CNAME             stan
; other hosts:
kyle        IN    A                 216.32.22.50
            IN    MX                1  mail.foo.com
            IN    MX                2  picasso.art.net
            IN    MX                5  mail.ct.oberon.tuwien.ac.at
butters     IN    A                 216.32.22.51
garison     IN    A                 216.32.22.52
            IN    HINFO             VAX-11/780  UNIX
            IN    WKS               216.32.22.52  TCP
                                (telnet ftp netstat finger pop)
wendy        IN   A                 216.32.34.2
            IN    HINFO             SUN  UNIX
; etc.....
```


The slide above shows other example entries found in the stan.pub.foo.org name server.

Note the additional information, such as **MX** records, host information (**HINFO**), and well-known services (**WKS**).

Consider the security relevance of HINFO and WKS.

Delegations

- Delegations are made when a zone has a parent domain
- A parent name server acting as delegation point keeps a Name Server record (NS) that specifies responsible name servers for that subzone
- A-records that correspond with associated NS records are called glue records
- Glue records are only necessary if the specified nameserver (NS record) is inside the subzone it serves!
  - AND the parent is no secondary server for that zone

Every zone needs **at least two** nameservers. One is called the primary or master, the other is the secondary or slave. Today we should use the terms master and slaves only.

Delegations are implemented using NS records, which specify authoritative (master or slave) name servers for some specific zone of this domain. Additionally A records are necessary to specify the associated IP addresses. These A records are the so-called "**glue records**".

Registration Terms

- Registry
  - Responsible of TLD zone maintenance
  - One unique registry per TLD
- Registrar
  - Intermediate agent between customer and registry (ISP)
- Registration
  - Customer tells registrar which NS should be used for delegation to reach a subdomain
  - Plus contact information

Network Solutions Inc is responsible (and hereby the only registry) for the TLDs com, net, org, and edu. Network Solutions Inc. also acts as registrar for these TLDs. Since June 1999 the ICANN allowed other registrars for the TLDs com, net, and org (see http://www.internic.net/regist.html for a list).

Domain Registrations

- Many providers act as "registrars"
- ICANN controls continental registrars
  - USA: InterNIC (www.internic.net)
  - Europe: RIPE (www.ripe.net)
  - Asia: APNIC (www.apnic.net)

ICANN

(C) Herbert Haas 2005/03/11 36

Domain name registration is independent from IP address assignment and usually any provider can act as a registrar, who applies for a registration at the regional network information center (RIPE, APNIC, InterNIC) in behalf of the customer.

The overall control over the DNS has recently been directed to the **ICANN**, the Internet Corporation for Assigned Names and Numbers. Check out http://www.icann.org.

Diagnostic Tools

- DIG - Domain Information Groper
  - Send domain name query packets to name servers
  - Results are printed in a human-readable format
- NSLOOKUP
  - Query Internet name servers interactively

There are two standard DNS tools available on most workstations. The Domain Information Groper (**DIG**) is the most important one and typically found on any UNIX and LINUX distribution.

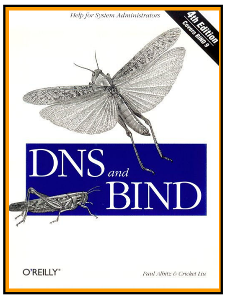
The syntax is:

```
dig [@server] domain [<query-type>] [<query-class>][+<query-option>]
[-<dig-option>] [%comment]
```

The other important tool is **NSLOOKUP** and also found on Windows operating systems. NSLOOKUP is operated interactively, just enter "help" for a command and option list.

Recommended Resources

- **DNS and BIND (4th Edition)**
  - **by Paul Albitz, Cricket Liu**
  - **The "Bible"**
- **The Internet Software Consortium**
  - **http://www.isc.org/**
  - **Where BIND comes from**
- **The Linux Documentation Project**
  - **http://www.tldp.org/**
  - **HOWTOs, FAQs, BOOKS, ...free!**



INTERNET SOFTWARE CONSORTIUM

The Linux Documentation Project

Selected RFCs (1)

- **RFC 1034**
  - **Domain Name Concept And Facilities**
- **RFC 1035**
  - **Domain Name Implementation and Specification**
- **RFC 1101**
  - **DNS Encoding Network Names And Other Types**
- **RFC 1183**
  - **New DNS RR Definitions**

RFC - 881 - The Domain Names Plan And Schedule

RFC - 882 - Domain Names Concepts And Facilities

RFC - 883 - Domain Name Implementation and Specification

RFC - 897 - Domain Name System Implementation Schedule

RFC - 921 - Domain Name System Implementation Schedule (Rev)

RFC - 973 - Domain System Changes And Observations

RFC - 974 - Mail Routing And the Domain System

RFC - 1032 - Domain Administrators Guide

RFC - 1033 - Domain Administrators Operations Guide

RFC - 1034 - Domain Name Concept And Facilities

RFC - 1035 - Domain Name Implementation and Specification

RFC - 1101 - DNS Encoding Network Names And Other Types

RFC - 1183 - New DNS RR Definitions

RFC - 1348 - DNS NSAP RRs

RFC - 1383 - An Experiment In DNS Based IP Routing

RFC - 1386 - The US Domain

RFC - 1394 - Relationship Of Telex Answerback Codes To Internet Domains

Selected RFCs (2)

- **RFC 1591**
  - **Domain Name System Structure And Delegation**
- **RFC 1664**
  - **Using The Internet DNS To Distribute RFC1327 Mail Address Mapping Tables**
- **RFC 1712**
  - **DNS Encoding Of Geographical Location**
- **RFC 1788**
  - **ICMP Domain Name Messages**
- **RFC 1794**
  - **DNS Support For Load Balancing**

RFC - 1401 - Correspondence Between The IAB And DISA On The Use Of DNS Throughout The Internet

RFC - 1464 - Using The Domain Name System To Store Arbitrary String Attributes

RFC - 1480 - The US Domain

RFC - 1535 - A Security Problem And Proposed Correction With Widely Deployed DNS Software

RFC - 1536 - Common DNS Implementation Errors And Suggested Fixes

RFC - 1537 - Common DNS Data File Configuration Errors

RFC - 1591 - Domain Name System Structure And Delegation

RFC - 1664 - Using The Internet DNS To Distribute RFC1327 Mail Address Mapping Tables

RFC - 1637 - DNS NSAP Resource Records

RFC - 1612 - DNS Resolver MIB Extensions

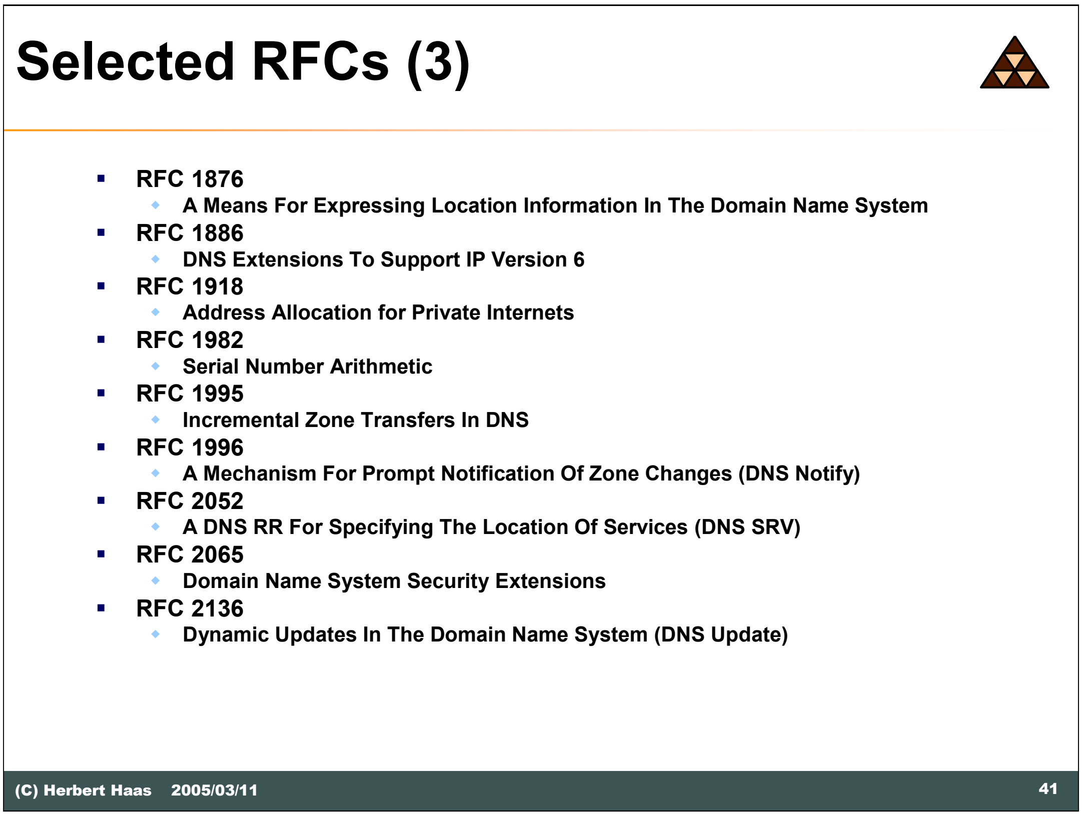
RFC - 1611 - DNS Server MIB Extensions

RFC - 1706 - DNS NSAP Resource Records

RFC - 1712 - DNS Encoding Of Geographical Location

RFC - 1788 - ICMP Domain Name Messages

RFC - 1794 - DNS Support For Load Balancing



RFC - 1876 - A Means For Expressing Location Information In The Domain Name System

RFC - 1886 - DNS Extensions To Support IP Version 6

RFC - 1912 - Common DNS Operational and Configuration Errors

RFC - 1918 - Address Allocation for Private Internets

RFC - 1982 - Serial Number Arithmetic

RFC - 1995 - Incremental Zone Transfers In DNS

RFC - 1996 - A Mechanism For Prompt Notification Of Zone Changes (DNS Notify)

RFC - 2052 - A DNS RR For Specifying The Location Of Services (DNS SRV)

RFC - 2065 - Domain Name System Security Extensions

RFC - 2136 - Dynamic Updates In The Domain Name System (DNS Update)

RFC - 2137 - Secure Domain Name System Dynamic Update

RFC - 2163 - Using the Internet DNS To Distribute MIXER Conformant Global Address Mapping (MCGAM)

RFC - 2168 - Resolution of Uniform Resource Identifiers Using The Domain Name System

RFC - 2181 - Clarifications To The DNS Specification

Selected RFCs (4)

- **RFC 2308**
  - **Negative Caching Of DNS Queries (DNS Ncache)**
- **RFC 2535**
  - **Domain Name System Security Extensions**
- **RFC 2541**
  - **DNS Security Operational Considerations**
- **RFC 2606**
  - **Reserved Top Level DNS Names**

RFC - 2182 - Selection And Operation Of Secondary DNS Servers

RFC - 2219 - Use Of DNS Aliases For Network Services

RFC - 2230 - Key Exchange Delegation Record For The DNS

RFC - 2240 - A Legal Basis For Domain Name Allocation

RFC - 2247 - Using Domains In LDAPX.500 Distinguished Names

RFC - 2308 - Negative Caching Of DNS Queries (DNS Ncache)

RFC - 2352 - A Convention For Using Legal Names As Domain Names

RFC - 2517 - Building Directories From DNS Experiences From WWW Seeker

RFC - 2535 - Domain Name System Security Extensions

RFC - 2536 - DSA KEYs And SIGs In The Domain Name System

RFC - 2537 - RSAMD5 KEYs And SIGs In The Domain Name System

RFC - 2538 - Storing Certificates In The Domain Name System

RFC - 2539 - Storage Of Diffie-Hellman Keys In The Domain Name System

RFC - 2540 - Detached Domain Name System Information

RFC - 2541 - DNS Security Operational Considerations

RFC - 2606 - Reserved Top Level DNS Names

Selected RFCs (5)

- **RFC 2672**
  - **Non-Terminal DNS Name Redirection**
- **RFC 2673**
  - **Binary Labels In The Domain Name System**
- **RFC 2845**
  - **Secret Key Transaction Authentication For DNS (TSIG)**
- **RFC 2870**
  - **Root Name Server Operational Requirements**
- **RFC 2874**
  - **DNS Extensions To Support IPv6 Address Aggregation And Renumbering**
- **RFC 3007**
  - **Secure Domain Name System Dynamic Update**

RFC - 2671 - Extension Mechanisms For DNS (EDNS0)

RFC - 2672 - Non-Terminal DNS Name Redirection

RFC - 2673 - Binary Labels In The Domain Name System

RFC - 2694 - DNS Extensions To Network Address Translators (DNS_ALG)

RFC - 2782 - A DNS RR For Specifying The Location Of Services (DNS SRV)

RFC - 2826 - IAB Technical Comment On The Unique DNS Root

RFC - 2845 - Secret Key Transaction Authentication For DNS (TSIG)

RFC - 2870 - Root Name Server Operational Requirements

RFC - 2874 - DNS Extensions To Support IPv6 Address Aggregation And Renumbering

RFC - 2915 - The Naming Authority Pointer (NAPTR) DNS Resource Record

RFC - 2916 - E.164 number and DNS

RFC - 2929 - Domain Name System IANA Considerations

RFC - 2931 - DNS Request And Transaction Signatures (SIG(0)s)

RFC - 3007 - Secure Domain Name System Dynamic Update

RFC - 3008 - Domain Name System Security (DNSSEC) Signing Authority

RFC - 3071 - Reflections On The DNS, RFC 1591, And Categories Of Domains

Selected RFCs (6)

- **RFC 3090**
  - DNS Security Extension Clarification On Zone Status
- **RFC 3152**
  - Delegation Of IP6.ARPA
- **RFC 3172**
  - Management Guidelines & Operational Requirements For the Address And Routing Parameter Area Domain (ARPA)
- **RFC 3363**
  - Representing Internet Protocol Version 6 Addresses In The Domain Name System
- **RFC 3364**
  - Tradeoffs In Domain Name System Support For Internet Protocol Version 6

RFC - 3088 - OpenLDAP Root Service An experimental LDAP referral service

RFC - 3090 - DNS Security Extension Clarification On Zone Status

RFC - 3110 - RSASHA-1 SIGs And RSA KEYs In The Domain Name System

RFC - 3123 - A DNS RR Type For Lists Of Address Prefixes (APL RR)

RFC - 3130 - Notes From The State Of The Technology DNSSEC

RFC - 3152 - Delegation Of IP6.ARPA

RFC - 3172 - Management Guidelines & Operational Requirements For the Address And Routing Parameter Area Domain (ARPA)

RFC - 3197 - Applicability Statement For DNS MIB Extensions

RFC - 3225 - Indicating Resolver Support Of DNSSEC

RFC - 3226 - DNSSEC And IPv6 A6 Aware Serverresolver Message Size Requirements

RFC - 3258 - Distributing Authoritative Name Servers Via Shared Unicast Addresses

RFC - 3363 - Representing Internet Protocol Version 6 Addresses In The Domain Name System

RFC - 3364 - Tradeoffs In Domain Name System Support For Internet Protocol Version 6

RFC - 3397 - Dynamic Host Configuration Protocol (DHCP) Domain Search Option

RFC - 3403 - Dynamic Delegation Discovery System Part Three The Domain Name System Database

RFC - 3425 - Obsoleting IQUERY

Summary

- DNS initially only created for humans
- Hierarchical tree of names
- Addresses and other database information
- Inverse resolution using in-addr.arpa TLD
- Primary vs Secondary nameservers
- Port 53, TCP and UDP

Any Questions?