Domain Name System

From Wikipedia, the free encyclopedia

The **Domain Name System** (**DNS**) is a hierarchical naming system for computers, services, or any resource participating in the Internet. It associates various information with domain names assigned to such participants. Most importantly, it translates domain names meaningful to humans into the numerical (binary) identifiers associated with networking equipment for the purpose of locating and addressing these devices world-wide. An often used analogy to explain the Domain Name System is that it serves as the "phone book" for the Internet by translating human-friendly computer hostnames into IP addresses. For example, *www.example.com* translates to *208.77.188.166*.

The Domain Name System makes it possible to assign domain names to groups of Internet users in a meaningful way, independent of each user's physical location. Because of this, World-Wide Web (WWW) hyperlinks and Internet contact information can remain consistent and constant even if the current Internet routing arrangements change or the participant uses a mobile device. Internet domain names are easier to remember than IP addresses such as `208.77.188.166` (IPv4) or `2001:db8:1f70::999:de8:7648:6e8` (IPv6). People take advantage of this when they recite meaningful URLs and e-mail addresses without having to know how the machine will actually locate them.

The Domain Name System distributes the responsibility of assigning domain names and mapping those names to IP addresses by designating authoritative name servers for each domain. Authoritative name servers are assigned to be responsible for their particular domains, and in turn can assign other authoritative name servers for their sub-domains. This mechanism has made the DNS distributed, fault tolerant, and helped avoid the need for a single central register to be continually consulted and updated.

The Internet Protocol Suite

Application Layer

BGP · DHCP · **DNS** · FTP · GTP · HTTP · IMAP · IRC · Megaco · MGCP · NNTP · NTP · POP · RIP · RPC · RTP · RTSP · SDP · SIP · SMTP · SNMP · SOAP · SSH · STUN · Telnet · TLS/SSL · XMPP · (more)

Transport Layer

TCP · UDP · DCCP · SCTP · RSVP · ECN · (more)

Internet Layer

IP (IPv4, IPv6) · ICMP · ICMPv6 · IGMP · IPsec · (more)

Link Layer

ARP · RARP · NDP · OSPF · Tunnels (L2TP) · Media Access Control (Ethernet, MPLS, DSL, ISDN, FDDI) · Device Drivers · (more)

In general, the Domain Name System also stores other types of information, such as the list of mail servers that accept email for a given Internet domain. By providing a world-wide, distributed keyword-based redirection service, the Domain Name System is an essential component of the functionality of the Internet.

Other identifiers such as RFID tags, UPC codes, International characters in email addresses and host names, and a variety of other identifiers could all potentially utilize DNS [1].

The Domain Name System also defines the technical underpinnings of the functionality of this database service. For this purpose it defines the DNS protocol, a detailed specification of the data structures and communication exchanges used in DNS, as part of the Internet Protocol Suite (TCP/IP). The context of the DNS within the Internet protocols may be seen in the following diagram. The DNS protocol was developed and defined in the early 1980s and published by the Internet Engineering Task Force (cf. History).

Contents

History

The practice of using a name as a more human-legible abstraction of a machine's numerical address on the network predates even TCP/IP. This practice dates back to the ARPAnet era. Back then, a different system was used. The DNS was invented in 1983, shortly after TCP/IP was deployed. With the older system, each computer on the network retrieved a file called *HOSTS.TXT* from a computer at SRI (now SRI International)[2][3]. The HOSTS.TXT file mapped numerical addresses to names. A hosts file still exists on most modern operating systems, either by default or through configuration, and allows users to specify an IP address (eg. 208.77.188.166) to use for a hostname (eg. www.example.net) without checking DNS. Systems based on a hosts file have inherent limitations, because of the obvious requirement that every time a given computer's address changed, every computer that seeks to communicate with it would need an update to its hosts file.

The growth of networking required a more scalable system that recorded a change in a host's address in one place only. Other hosts would learn about the change dynamically through a notification system, thus completing a globally accessible network of all hosts' names and their associated IP Addresses.

At the request of Jon Postel, Paul Mockapetris invented the Domain Name System in 1983 and wrote the first implementation. The original specifications appear in RFC 882 and RFC 883. In November 1987, the publication of RFC 1034 and RFC 1035 updated the DNS specification and made RFC 882 and RFC 883 obsolete. Several more-recent RFCs have proposed various extensions to the core DNS protocols.

In 1984, four Berkeley students—Douglas Terry, Mark Painter, David Riggle and Songnian Zhou—wrote the first UNIX implementation, which was maintained by Ralph Campbell thereafter. In 1985, Kevin Dunlap of DEC significantly re-wrote the DNS implementation and renamed it BIND—Berkeley Internet Name Domain. Mike Karels, Phil Almquist and Paul Vixie have maintained BIND since then. BIND was ported to the Windows NT platform in the early 1990s.

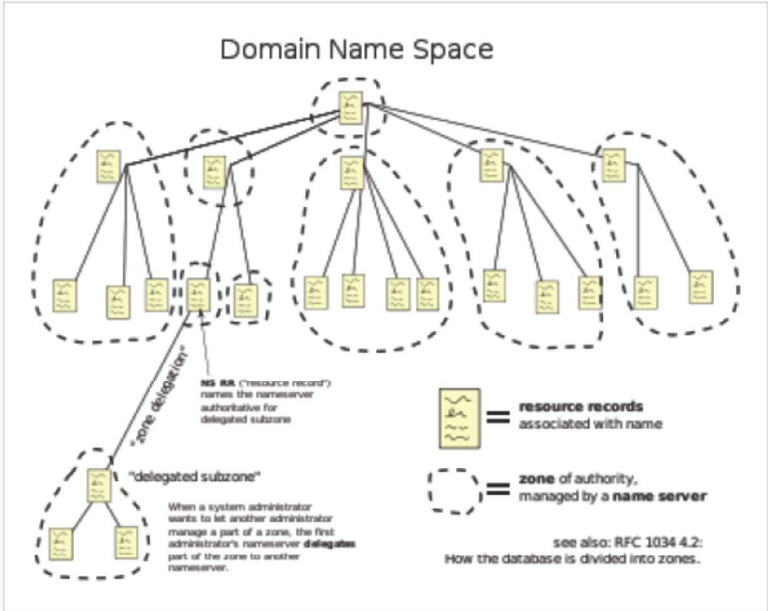
BIND was widely distributed, especially on Unix systems, and is the dominant DNS software in use on the Internet.[4] With the heavy use and resulting scrutiny of its open-source code, as well as increasingly more sophisticated attack methods, many security flaws were discovered in BIND. This contributed to the development of a number of alternative nameserver and resolver programs. BIND itself was re-written from scratch in version 9, which has a security record comparable to other modern Internet software.

Structure

The domain name space

The domain name space consists of a tree of domain names. Each node or leaf in the tree has zero or more *resource records*, which hold information associated with the domain name. The tree sub-divides into *zones* beginning at the root zone. A DNS zone consists of a collection of connected nodes authoritatively served by an *authoritative nameserver*. (Note that a single nameserver can host several zones.)

Administrative responsibility over any zone may be divided, thereby creating additional zones. Authority is said to be *delegated* for a portion of the old space, usually in form of sub-domains, to another nameserver and administrative entity. The old zone ceases to be authoritative for the new zone.



Domain names, arranged in a tree, cut into zones, each served by a nameserver.

Parts of a domain name

A domain name usually consists of two or more parts (technically **labels**), which are conventionally written separated by dots, such as `example.com`.

- The rightmost label conveys the top-level domain (for example, the address `www.example.com` has the top-level domain `com`).
- Each label to the left specifies a subdivision, or subdomain of the domain above it. Note: "subdomain" expresses relative dependence, not absolute dependence. For example: `example.com` is a subdomain of the `com` domain, and `www.example.com` is a subdomain of the domain `example.com`. In theory, this subdivision can go down 127 levels. Each label can contain up to 63 octets. The whole domain name may not exceed a total length of 253 octets. [5] In practice, some domain registries may have shorter limits.
- A hostname refers to a domain name that has one or more associated IP addresses; ie: the '`www.example.com`' and '`example.com`' domains are both hostnames, however, the '`com`' domain is not.

DNS servers

The Domain Name System is maintained by a distributed database system, which uses the client-server model. The nodes of this database are the name servers. Each domain or subdomain has one or more authoritative DNS servers that publish information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root nameservers: the servers to query when looking up (*resolving*) a top-level domain name (TLD).

DNS resolvers

See also: resolv.conf

The client-side of the DNS is called a DNS resolver. It is responsible for initiating and sequencing the queries that ultimately lead to a full resolution (translation) of the resource sought, e.g., translation of a domain name into an IP address.

A DNS query may be either a recursive query or a non-recursive query:

- A *non-recursive query* is one in which the DNS server may provide a partial answer to the query (or give an error).
- A *recursive query* is one where the DNS server will fully answer the query (or give an error). DNS servers are not required to support recursive queries.

The resolver (or another DNS server acting recursively on behalf of the resolver) negotiates use of recursive service using bits in the query headers.

Resolving usually entails iterating through several name servers to find the needed information. However, some resolvers function simplistically and can communicate only with a single name server. These simple resolvers rely on a recursive query to a recursive name server to perform the work of finding information for them.

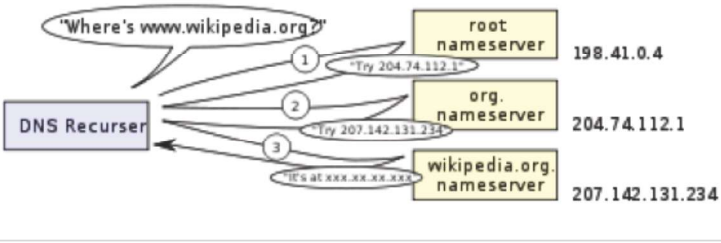
Address resolution mechanism

(This description deliberately uses the fictional .example TLD in accordance with the DNS guidelines.)

In *theory* a full host name may have several name segments, (e.g *ahost.ofasubnet.ofabiggernet.inadomain.example*). In practice, full host names will frequently consist of just three segments (*ahost.inadomain.example*, and most often www.*inadomain.example*). For querying purposes, software interprets the name segment by segment, from right to left. At each step along the way, the program queries a corresponding DNS server to provide a pointer to the next server which it should consult.

As originally envisaged, the process was as simple as:

1. the local system is pre-configured with the known addresses of the root servers in a file of *root hints*, which need to be updated periodically by the local administrator from a reliable source to be kept up to date with the changes which occur over time.
2. query one of the root servers to find the server authoritative for the next level down (so in the case of our simple hostname, a root server would be asked for the address of a server with detailed knowledge of the *example* top level domain).
3. querying this second server for the address of a DNS server with detailed knowledge of the second-level domain (*inadomain.example* in our example).
4. repeating the previous step to progress down the name, until the final step which would, rather than generating the address of the next DNS server, return the final address sought.



A DNS recursor consults three nameservers to resolve the address www.wikipedia.org.

The diagram illustrates this process for the real host www.wikipedia.org.

The mechanism in this simple form has a difficulty: it places a huge operating burden on the root servers, with every search for an address starting by querying one of them. Being as critical as they are to the overall function of the system, such heavy use would create an insurmountable bottleneck for trillions of queries placed every day. In practice caching is used to overcome this problem, and in actual fact root nameservers deal with very little of the total traffic.

Circular dependencies and glue records

Name servers in delegations appear listed by name, rather than by IP address. This means that a resolving name server must issue another DNS request to find out the IP address of the server to which it has been referred. Since this can

introduce a circular dependency if the nameserver referred to is under the domain that it is authoritative of, it is occasionally necessary for the nameserver providing the delegation to also provide the IP address of the next nameserver. This record is called a *glue record*.

For example, assume that the sub-domain en.wikipedia.org contains further sub-domains (such as `something.en.wikipedia.org`) and that the authoritative name server for these lives at `ns1.something.en.wikipedia.org`. A computer trying to resolve `something.en.wikipedia.org` will thus first have to resolve `ns1.something.en.wikipedia.org`. Since `ns1` is also under the `something.en.wikipedia.org` subdomain, resolving `ns1.something.en.wikipedia.org` requires resolving `something.en.wikipedia.org` which is exactly the circular dependency mentioned above. The dependency is broken by the glue record in the nameserver of `en.wikipedia.org` that provides the IP address of `ns1.something.en.wikipedia.org` directly to the requestor, enabling it to bootstrap the process by figuring out where `ns1.something.en.wikipedia.org` is located.

Wildcard DNS records

DNS also supports **wildcard DNS records** that will match requests for non-existent domain names. A wildcard DNS record is specified by using a "*" as the left most label (part) of a domain name, e.g. `*.example.com`. The exact rules for when a wild card will match are specified in RFC 1034, but the rules are neither intuitive nor clearly specified. This has resulted in incompatible implementations and unexpected results when they are used.

In practice

When an application (such as a web browser) tries to find the IP address of a domain name, it doesn't necessarily follow all of the steps outlined in the *Theory* section above. We will first look at the concept of caching, and then outline the operation of DNS in "the real world."

Caching and time to live

Because of the huge volume of requests generated by a system like DNS, the designers wished to provide a mechanism to reduce the load on individual DNS servers. To this end, the DNS resolution process allows for *caching* (i.e. the local recording and subsequent consultation of the results of a DNS query) for a given period of time after a successful answer. How long a resolver caches a DNS response (i.e. how long a DNS response remains *valid*) is determined by a value called the time to live (TTL). The TTL is set by the administrator of the DNS server handing out the response. The period of validity may vary from just seconds to days or even weeks.

Caching time

As a noteworthy consequence of this distributed and caching architecture, changes to DNS do not always take effect immediately and globally. This is best explained with an example: If an administrator has set a TTL of 6 hours for the host `www.wikipedia.org`, and then changes the IP address to which `www.wikipedia.org` resolves at 12:01pm, the administrator must consider that a person who cached a response with the old IP address at 12:00noon will not consult the DNS server again until 6:00pm. The period between 12:01pm and 6:00pm in this example is called *caching time*, which is best defined as a period of time that begins when you make a change to a DNS record and ends after the maximum amount of time specified by the TTL expires. This essentially leads to an important logistical consideration when making changes to DNS: *not everyone is necessarily seeing the same thing you're seeing*. RFC 1912 helps to convey basic rules for how to set the TTL.

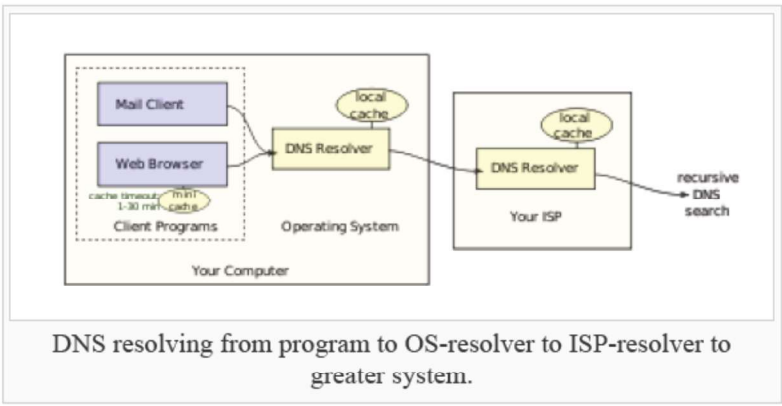
Note that the term "propagation", although very widely used in this context, does not describe the effects of caching well. Specifically, it implies that [1] when you make a DNS change, it somehow spreads to all other DNS servers (instead, other DNS servers check in with yours as needed), and [2] that you do not have control over the amount of time the record is cached (you control the TTL values for all DNS records in your domain, except your NS records and any authoritative DNS servers that use your domain name).

Some resolvers may override TTL values, as the protocol supports caching for up to 68 years or no caching at all. Negative caching (the non-existence of records) is determined by name servers authoritative for a zone which MUST include the Start of Authority (SOA) record when reporting no data of the requested type exists. The MINIMUM field of the SOA record and the TTL of the SOA itself is used to establish the TTL for the negative answer. RFC 2308

Many people incorrectly refer to a mysterious 48 hour or 72 hour propagation time when you make a DNS change. When one changes the NS records for one's domain or the IP addresses for hostnames of authoritative DNS servers using one's domain (if any), there can be a lengthy period of time before all DNS servers use the new information. This is because those records are handled by the zone parent DNS servers (for example, the .com DNS servers if your domain is example.com), which typically cache those records for 48 hours. However, those DNS changes will be immediately available for any DNS servers that do not have them cached. And any DNS changes on your domain other than the NS records and authoritative DNS server names can be nearly instantaneous, if you choose for them to be (by lowering the TTL once or twice ahead of time, and waiting until the old TTL expires before making the change).

In the real world

Users generally do not communicate directly with a DNS resolver. Instead DNS-resolution takes place transparently in client-applications such as web-browsers, mail-clients, and other Internet applications. When an application makes a request which requires a DNS lookup, such programs send a resolution request to the local DNS resolver in the local operating system, which in turn handles the communications required.



DNS resolving from program to OS-resolver to ISP-resolver to greater system.

The DNS resolver will almost invariably have a cache (see above) containing recent lookups. If the cache can provide the answer to the request, the resolver will return the value in the cache to the program that made the request. If the cache does not contain the answer, the resolver will send the request to one or more designated DNS servers. In the case of most home users, the Internet service provider to which the machine connects will usually supply this DNS server: such a user will either have configured that server's address manually or allowed DHCP to set it; however, where systems administrators have configured systems to use their own DNS servers, their DNS resolvers point to separately maintained nameservers of the organization. In any event, the name server thus queried will follow the process outlined above, until it either successfully finds a result or does not. It then returns its results to the DNS resolver; assuming it has found a result, the resolver duly caches that result for future use, and hands the result back to the software which initiated the request.

Broken resolvers

An additional level of complexity emerges when resolvers violate the rules of the DNS protocol. A number of large ISPs have configured their DNS servers to violate rules (presumably to allow them to run on less-expensive hardware than a fully-compliant resolver), such as by disobeying TTLs, or by indicating that a domain name does not exist just because one of its name servers does not respond.[6]

As a final level of complexity, some applications (such as web-browsers) also have their own DNS cache, in order to reduce the use of the DNS resolver library itself. This practice can add extra difficulty when debugging DNS issues, as it obscures the freshness of data, and/or what data comes from which cache. These caches typically use very short caching times — on the order of one minute. Internet Explorer offers a notable exception: recent versions cache DNS records for half an hour.[7]

Other applications

The system outlined above provides a somewhat simplified scenario. The Domain Name System includes several other functions:

- Hostnames and IP addresses do not necessarily match on a one-to-one basis. Many hostnames may correspond to a single IP address: combined with virtual hosting, this allows a single machine to serve many web sites. Alternatively a single hostname may correspond to many IP addresses: this can facilitate fault tolerance and load distribution, and also allows a site to move physical location seamlessly.
- There are many uses of DNS besides translating names to IP addresses. For instance, Mail transfer agents use DNS to find out where to deliver e-mail for a particular address. The domain to mail exchanger mapping provided by MX records accommodates another layer of fault tolerance and load distribution on top of the name to IP address mapping.

- E-mail Blacklists: The DNS system is used for efficient storage and distribution of IP addresses of blacklisted e-mail hosts. The usual method is putting the IP address of the subject host into the sub-domain of a higher level domain name, and resolve that name to different records to indicate a positive or a negative. A hypothetical example using blacklist.com,
  - 102.3.4.5 is blacklisted => Creates 5.4.3.102.blacklist.com and resolves to 127.0.0.1
  - 102.3.4.6 is not => 6.4.3.102.blacklist.com is not found, or default to 127.0.0.2
  - E-mail servers can then query blacklist.com through the DNS mechanism to find out if a specific host connecting to them are in the blacklist. Today many of such blacklists, either free or subscription-based, are available mainly for use by email administrators and anti-spam software.

- Software Updates: many anti-virus and commercial software now use the DNS system to store version numbers of the latest software updates so client computers do not need to connect to the update servers every time. For these type of applications, the cache time of the DNS records are usually shorter.

- Sender Policy Framework and DomainKeys, instead of creating their own record types, were designed to take advantage of another DNS record type, the TXT record.
- To provide resilience in the event of computer failure, multiple DNS servers are usually provided for coverage of each domain, and at the top level, thirteen very powerful root servers exist, with additional "copies" of several of them distributed worldwide via Anycast.

Protocol details

DNS primarily uses UDP on port 53 [8] to serve requests. Almost all DNS queries consist of a single UDP request from the client followed by a single UDP reply from the server. TCP comes into play only when the response data size exceeds 512 bytes, or for such tasks as zone transfer. Some operating systems such as HP-UX are known to have resolver implementations that use TCP for all queries, even when UDP would suffice.

Extensions to DNS

EDNS is an extension of the DNS protocol which allows the transport over UDP of DNS replies exceeding 512 bytes, and adds support for expanding the space of request and response codes. It is described in RFC 2671.

DNS resource records

Further information: List of DNS record types

A **Resource Record** (RR) is the basic data element in the domain name system. Each record has a type (A, MX, etc.), a TTL, a class and some type-specific information. All resource records of the same type define a **Resource Record Set** (RR set). The order that resource records in a RR set are returned by the resolver to an application is undefined (the server typically uses round-robin DNS). DNSSEC, however, works on complete RR sets in a canonical order.

When sent over the Internet, all records use the common format specified in RFC 1035 shown below.

RR (Resource record) fields

Field	Description	Length (octets)
NAME	Name of the node to which this record pertains.	(variable)
TYPE	Type of RR. For example, MX is type 15.	2
CLASS	Class code.	2
TTL	Signed time in seconds that RR stays valid.	4
RDLENGTH	Length of RDATA field.	2
RDATA	Additional RR-specific data.	(variable)

The NAME is the fully qualified domain name of the node in the tree. On the wire, the name may be shortened using label compression where ends of domain names mentioned earlier in the packet can be substituted for the end of the current domain name.

The TYPE of the record indicates what the format of the data is, and gives a hint of its intended use; for instance, the **A** record is used to translate from a domain name to an IPv4 address, the **NS** record lists which name servers can answer lookups on a DNS zone, and the **MX** record is used to translate from a name in the right-hand side of an e-mail address to the name of a machine able to handle mail for that address.

The RDATA is type-specific information, such as the actual IP address for A records, or the mail host for MX records. Well known record types may use label compression in the RDATA field, but "unknown" record types can not (see RFC 3597).

The CLASS of a record is almost always set to "IN" or "Internet". There are also the very rarely used "CH" (Chaos) and "HS" (Hesiod) classes. In theory, each class can be completely independent trees with different delegation DNS zones and different names, but in practice they all mirrored the Internet class.

In addition to resource records defined in a zone file, there are also some pseudo record types that are used only on the wire, such as to perform zone transfers (AXFR/IXFR) or for EDNS (OPT).

Internationalized domain names

While domain names technically have no restrictions on the characters they use and can include non-ASCII characters, the same is not true for host names.[9] Host names are the names most people see and use for things like e-mail and web browsing. Host names are restricted to a small subset of the ASCII character set known as **LDH**, the **L**etters A–Z in upper and lower case, **D**igits 0–9, **H**yphen, and the dot to separate LDH-labels; see RFC 3696 section 2 for details. This prevented the representation of names and words of many languages natively. ICANN has approved the Punycode-based IDNA system, which maps Unicode strings into the valid DNS character set, as a workaround to this issue. Some registries have adopted IDNA.

Security issues

DNS was not originally designed with security in mind, and thus has a number of security issues.

One class of vulnerabilities is DNS cache poisoning, which tricks a DNS server into believing it has received authentic information when, in reality, it has not.

DNS responses are traditionally not cryptographically signed, leading to many attack possibilities; The Domain Name System Security Extensions (DNSSEC) modifies DNS to add support for cryptographically signed responses. There are various extensions to support securing zone transfer information as well.

Even with encryption, a DNS server could become compromised by a virus (or for that matter a disgruntled employee) that would cause IP addresses of that server to be redirected to a malicious address with a long TTL. This could have far-reaching impact to potentially millions of Internet users if busy DNS servers cache the bad IP data. This would require manual purging of all affected DNS caches as required by the long TTL (up to 68 years).

Some domain names can spoof other, similar-looking domain names. For example, "paypal.com" and "paypa1.com" are different names, yet users may be unable to tell the difference when the user's typeface (font) does not clearly differentiate the letter l and the numeral 1. This problem is much more serious in systems that support internationalized domain names, since many characters that are different, from the point of view of ISO 10646, appear identical on typical computer screens. This vulnerability is often exploited in phishing.

Techniques such as Forward Confirmed reverse DNS can also be used to help validate DNS results.

Domain registration

The right to use a domain name is delegated by domain name registrars which are accredited by the Internet Corporation for Assigned Names and Numbers (ICANN), the organization charged with overseeing the name and number systems of the Internet. In addition to ICANN, each top-level domain (TLD) is maintained and serviced technically by a sponsoring organization, the TLD Registry. The registry is responsible for maintaining the database of names registered within the TLDs they administer. The registry receives registration information from each domain name registrar authorized to assign names in the corresponding TLD and publishes the information using a special service, the whois protocol.

Registrars usually charge an annual fee for the service of delegating a domain name to a user and providing a default set of name servers. Often this transaction is termed a sale or lease of the domain name, and the registrant is called an "owner", but no such legal relationship is actually associated with the transaction, only the exclusive right to use the domain name. More correctly authorized users are known as "registrants" or as "domain holders".

ICANN publishes a complete list of TLD registries and domain name registrars in the world. One can obtain information about the registrant of a domain name by looking in the WHOIS database held by many domain registries.

For most of the more than 240 country code top-level domains (ccTLDs), the domain registries hold the authoritative WHOIS (Registrant, name servers, expiration dates, etc.). For instance, DENIC, Germany NIC, holds the authoritative WHOIS to a .DE domain name. Since about 2001, most gTLD registries (.ORG, .BIZ, .INFO) have adopted this so-called "thick" registry approach, i.e. keeping the authoritative WHOIS in the central registries instead of the registrars.

For .COM and .NET domain names, a "thin" registry is used: the domain registry (e.g. VeriSign) holds a basic WHOIS (registrar and name servers, etc.). One can find the detailed WHOIS (registrant, name servers, expiry dates, etc.) at the registrars.

Some domain name registries, also called Network Information Centres (NIC), also function as registrars, and deal directly with end users. But most of the main ones, such as for .COM, .NET, .ORG, .INFO, etc., use a registry-registrar model. There are hundreds of Domain Name Registrars that actually perform the domain name registration with the end user (see lists at ICANN (https://web.archive.org/web/20090320152858/http://www.icann.org/registrars/accredited-list.html) or VeriSign (https://web.archive.org/web/20090320152858/http://www.verisign.com/information-services/naming-services/com-net-registry/page_002166.html)). By using this method of distribution, the registry only has to manage the relationship with the registrar, and the registrar maintains the relationship with the end users, or 'registrants' -- in some cases through additional layers of resellers.

In the process of registering a domain name and maintaining authority over the new name space created, registrars store and use several key pieces of information connected with a domain:

- **Administrative contact**. A registrant usually designates an administrative contact to manage the domain name. The administrative contact usually has the highest level of control over a domain. Management functions delegated to the administrative contacts may include management of all business information, such as name of

record, postal address, and contact information of the official registrant of the domain and the obligation to conform to the requirements of the domain registry in order to retain the right to use a domain name. Furthermore the administrative contact installs additional contact information for technical and billing functions.
- **Technical contact**. The technical contact manages the name servers of a domain name. The functions of a technical contact include assuring conformance of the configurations of the domain name with the requirements of the domain registry, maintaining the domain zone records, and providing continuous functionality of the name servers (that leads to the accessibility of the domain name).
- **Billing contact**. The party responsible for receiving billing invoices from the domain name registrar and paying applicable fees.
- **Name servers**. Most registrars provide two or more name servers as part of the registration service. However, a registrant may specify its own authoritative name servers to host a domain's resource records. The registrar's policies govern the number of servers and the type of server information required. Some providers require a hostname and the corresponding IP address or just the hostname, which must be resolvable either in the new domain, or exist elsewhere. Based on traditional requirements (RFC 1034), typically a minimum of two servers is required.

Abuse and regulation

Critics often claim abuse of administrative power over domain names. Particularly noteworthy was the VeriSign Site Finder system which redirected all unregistered .com and .net domains to a VeriSign webpage. For example, at a public meeting with VeriSign to air technical concerns about SiteFinder [10], numerous people, active in the IETF and other technical bodies, explained how they were surprised by VeriSign's changing the fundamental behavior of a major component of Internet infrastructure, not having obtained the customary consensus. SiteFinder, at first, assumed every Internet query was for a website, and it monetized queries for incorrect domain names, taking the user to VeriSign's search site. Unfortunately, other applications, such as many implementations of email, treat a lack of response to a domain name query as an indication that the domain does not exist, and that the message can be treated as undeliverable. The original VeriSign implementation broke this assumption for mail, because it would always resolve an erroneous domain name to that of SiteFinder. While VeriSign later changed SiteFinder's behaviour with regard to email, there was still widespread protest about VeriSign's action being more in its financial interest than in the interest of the Internet infrastructure component for which VeriSign was the steward.

Despite widespread criticism, VeriSign only reluctantly removed it after the Internet Corporation for Assigned Names and Numbers (ICANN) threatened to revoke its contract to administer the root name servers. ICANN published the extensive set of letters exchanged, committee reports, and ICANN decisions [11].

There is also significant disquiet regarding the United States' political influence over ICANN. This was a significant issue in the attempt to create a .xxx top-level domain and sparked greater interest in alternative DNS roots that would be beyond the control of any single country.[12]

Additionally, there are numerous accusations of domain name "front running", whereby registrars, when given whois queries, automatically register the domain name for themselves. Recently, Network Solutions has been accused of this.[13]

Truth in Domain Names Act

In the United States, the "Truth in Domain Names Act" (actually the "Anticybersquatting Consumer Protection Act"), in combination with the PROTECT Act, forbids the use of a misleading domain name with the intention of attracting people into viewing a visual depiction of sexually explicit conduct on the Internet.

Internet standards

The Domain name system is defined by Request for Comments published by the Internet Engineering Task Force (Internet standards). The following is a list of some of the RFCs that pertain to the core DNS protocol.

- RFC 920 Specified original TLDs: .arpa, .com, .edu, .org, .gov, .mil and two-character country codes
- RFC 1032 Domain administrators guide
- RFC 1033 Domain administrators operations guide
- RFC 1034 Domain Names - Concepts and Facilities.
- RFC 1035 Domain Names - Implementation and Specification
- RFC 1101 DNS Encodings of Network Names and Other Types
- RFC 1123 Requirements for Internet Hosts -- Application and Support
- RFC 1912 Common DNS Operational and Configuration Errors
- RFC 1995 Incremental Zone Transfer in DNS
- RFC 1996 A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)
- RFC 2136 Dynamic Updates in the domain name system (DNS UPDATE)
- RFC 2181 Clarifications to the DNS Specification
- RFC 2182 Selection and Operation of Secondary DNS Servers
- RFC 2308 Negative Caching of DNS Queries (DNS NCACHE)
- RFC 2317 Classless IN-ADDR.ARPA delegation
- RFC 2671 Extension Mechanisms for DNS (EDNS0)
- RFC 3597 Handling of Unknown DNS Resource Record (RR) Types
- RFC 3696 Application Techniques for Checking and Transformation of Names
- RFC 4343 Domain Name System (DNS) Case Insensitivity Clarification
- RFC 4592 The Role of Wildcards in the Domain Name System
- RFC 4892 Requirements for a Mechanism Identifying a Name Server Instance
- RFC 5001 DNS Name Server Identifier Option (NSID)

See also

- Dynamic DNS
- Alternative DNS root
- Comparison of DNS server software
- Round robin DNS
- Split-horizon DNS
- DNS management software

References

1. ^ Mockapetris, Paul (2004-01-02). "Letting DNS Loose". CircleID. http://www.circleid.com/posts/letting_dns_loose/.
2. ^ "History of the DNS". http://www.lagunainternet.com/techsupport/history_of_dns.htm. Retrieved on 2008-04-29.
3. ^ Cricket Liu, Paul Albitz. "DNS & BIND". O'Reilly (shown via Google Books). http://books.google.co.uk/books?id=zkZN52WhG8sC&pg=PA3&lpg=PA3&dq=sri+HOSTS.TXT&source=web&ots=wuZ79E-zJ2&sig=btF0Z2nclOnX_UgNj7a1f5S7Uqg&hl=en. Retrieved on 2008-04-29.
4. ^ http://mydns.bboy net/survey/ DNS Server Survey
5. ^ What is the maximum length of a domain name? (https://web.archive.org/web/20090320152858/http://www.ops.ietf.org/lists/namedroppers/namedroppers.2003/msg00964.html) on the IETF DNSOP working group mailing list. On the wire, in the DNS binary format, it can be at most 255 octets as per RFC 1034 section 3.1. For an all-ASCII hostname, this can be represented in traditional dot notation as 253 characters.
6. ^ "Providers ignoring DNS TTL ?". Slashdot. 2005. http://ask.slashdot.org/article.pl?sid=05/04/18/198259. Retrieved on 2009-01-03.
7. ^ "How Internet Explorer uses the cache for DNS host entries". Microsoft. 2004. http://support.microsoft.com/default.aspx?scid=KB;en-us;263558. Retrieved on 2006-03-07.
8. ^ Mockapetris, P (November, 1987). "RFC1035: Domain Names - Implementation and Specification". http://www.ietf.org/rfc/rfc1035.txt. Retrieved on 2007-07-31.
9. ^ The term *host name* is here being used to mean an FQDN for a host, such as eg. `en.wikipedia.org.`, and not just (to use the same example) `en` .
   While most domain names do indeed designate hosts, some domain name DNS entries may not. In this sense, a (FQDN) hostname is a type of domain name, but not all domain names are actual host names. Cf. this host name vs domain name explanation (https://web.archive.org/web/20090320152858/http://www.ops.ietf.org/lists/namedroppers/namedroppers.2005/msg00889.html) from the DNS OP IETF Working Group.

10. ^ McCullagh, Declan (2003-10-03). "VeriSign fends off critics at ICANN confab". CNET News.com. http://www news.com/2100-1038-5088128 html. Retrieved on 2007-09-22.
11. ^ Internet Corporation for Assigned Names and Numbers (ICANN). "Verisign's Wildcard Service Deployment". http://www.icann.org/topics/wildcard-history.html. Retrieved on 2007-09-22.
12. ^ Mueller, M (March 2004). *Ruling the Root*. MIT Press. ISBN 0262632985.
13. ^ Slashdot | NSI Registers Every Domain Checked (https://web.archive.org/web/20090320152858/http://slashdot.org/article.pl?sid=08/01/08/1920215)

External links

- DNS Complexity (https://web.archive.org/web/20090320152858/http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=481) , Paul Vixie, ACM Queue
- Open Source Guide - DNS for Rocket Scientists (https://web.archive.org/web/20090320152858/http://www.zytrax.com/books/dns/) , an on-line technical book for further reading
- A listing of some DNS tools (https://web.archive.org/web/20090320152858/http://www.bind9.net/)
- CircleID - Open news and opinion hub for all DNS related topics (https://web.archive.org/web/20090320152858/http://www.circleid.com/topics/dns/)

Retrieved from "http://en.wikipedia.org/wiki/Domain_Name_System"
Categories: Internet protocols | Domain name system | Application layer protocols
Hidden categories: Articles containing potentially dated statements from 2007 | All articles containing potentially dated statements