Domain Name System

The **Domain Name System** (**DNS**) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates easily memorized domain names to the numerical IP addresses needed for the purpose of locating computer services and devices worldwide. The Domain Name System is an essential component of the functionality of the Internet.

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, the domain name www.example.com translates to the addresses 93.184.216.119 (IPv4) and 2606:2800:220:6d:26bf:1447:1097:aa7 (IPv6). Unlike a phone book, the DNS can be quickly updated, allowing a service's location on the network to change without affecting the end users, who continue to use the same host name. Users take advantage of this when they use meaningful Uniform Resource Locators (URLs), and e-mail addresses without having to know how the computer actually locates the services.

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 supported domains, and may delegate authority over subdomains to other name servers. This mechanism provides distributed and fault tolerant service and was designed to avoid the need for a single central database.

The Domain Name System also specifies the technical functionality of this database service. It defines the DNS protocol, a detailed specification of the data structures and data communication exchanges used in DNS, as part of the Internet Protocol Suite.

The Internet maintains two principal namespaces, the domain name hierarchy[1] and the Internet Protocol (IP) address spaces.[2] The Domain Name System maintains the domain name hierarchy and provides translation services between it and the address spaces. Internet name servers and a communication protocol implement the Domain Name System.[3] A DNS name server is a server that stores the DNS records for a domain name, such as address (A or AAAA) records, name server (NS) records, and mail exchanger (MX) records (see also list of DNS record types); a DNS name server responds with answers to queries against its database.

History

Using a simpler, more memorable name in place of a host's numerical address dates back to the ARPANET era. The staff at Stanford Research Institute (now SRI International) created and updated a file named HOSTS.TXT that mapped intelligible names to the numerical addresses of computers on ARPANET. SRI transmitted this file, and updates of this file, to each computer connected to ARPANET, which later became the Internet.[4][5] The Internet's rapid growth required an automated system for maintaining and distributing domain names and their corresponding numerical addresses, to replace SRI's centrally maintained, manually entered HOSTS.TXT file.

Paul Mockapetris designed the Domain Name System at UC Irvine in 1983, and wrote the first implementation at the request of Jon Postel from UCLA. The Internet Engineering Task Force published the original specifications in RFC 882 and RFC 883 in November 1983.

Domain Name System

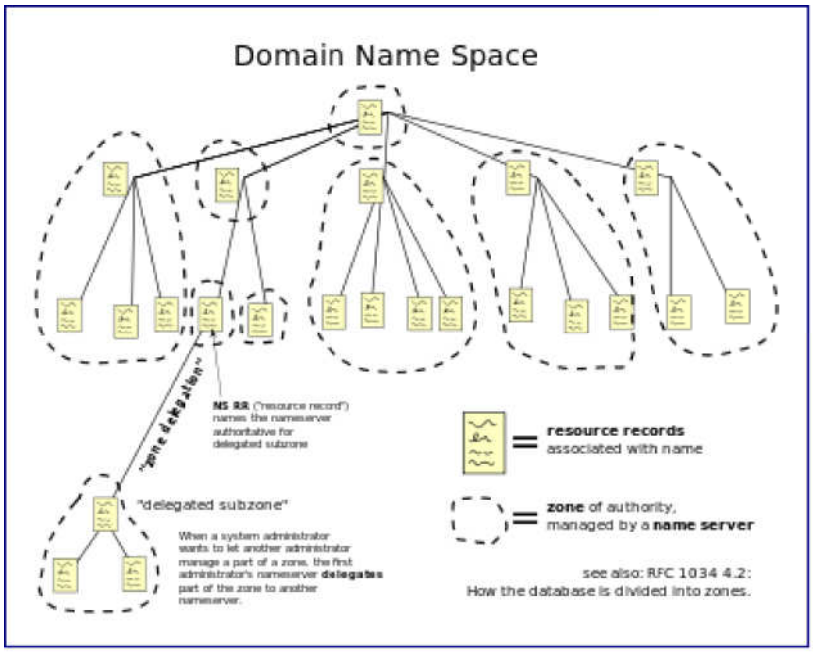
In 1984, four UC Berkeley students—Douglas Terry, Mark Painter, David Riggle, and Songnian Zhou—wrote the first Unix name server implementation, called the Berkeley Internet Name Domain (BIND) Server.[6] In 1985, Kevin Dunlap of DEC substantially revised the DNS implementation. Mike Karels, Phil Almquist, and Paul Vixie have maintained BIND since then.[7] BIND was ported to the Windows NT platform in the early 1990s. BIND was widely distributed, especially on Unix systems, and is still the most widely used DNS software on the Internet.[7]

In November 1987, RFC 1034[1] and RFC 1035[3] superseded the 1983 DNS specifications. Several additional Request for Comments have proposed extensions to the core DNS protocols.

Structure

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 may consist of only one domain, or may consist of many domains and sub-domains, depending on the administrative authority delegated to the manager.



The hierarchical Domain Name System, organized into zones, each served by a name server

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

Domain Name System

Domain name syntax

The definitive descriptions of the rules for forming domain names appear in RFC 1035, RFC 1123, and RFC 2181. A domain name consists of one or more parts, technically called *labels*, that are conventionally concatenated, and delimited by dots, such as example.com.

- The right-most label conveys the top-level domain; for example, the domain name www.example.com belongs to the top-level domain *com*.
- The hierarchy of domains descends from right to left; each label to the left specifies a subdivision, or subdomain of the domain to the right. For example: the label *example* specifies a subdomain of the *com* domain, and *www* is a sub domain of example.com. This tree of subdivisions may have up to 127 levels.
- Each label may contain up to 63 characters. The full domain name may not exceed the length of 253 characters in its textual representation.[1] In the internal binary representation of the DNS the maximum length requires 255 octets of storage, since it also stores the length of the name.[3] In practice, some domain registries may have shorter limits.
- DNS names may technically consist of any character representable in an octet. However, the allowed formulation of domain names in the DNS root zone, and most other sub domains, uses a preferred format and character set. The characters allowed in a label are a subset of the ASCII character set, and includes the characters *a* through *z*, *A* through *Z*, digits *0* through *9*, and the hyphen. This rule is known as the *LDH rule* (letters, digits, hyphen). Domain names are interpreted in case-independent manner.[8] Labels may not start or end with a hyphen.[9] There is an additional rule that essentially requires that top-level domain names not be all-numeric.[9]
- A hostname is a domain name that has at least one IP address associated. For example, the domain names www.example.com and example.com are also hostnames, whereas com is not.

Internationalized domain names

The limited set of ASCII characters permitted in the DNS prevented the representation of names and words of many languages in their native alphabets or scripts. To make this possible, ICANN approved the Internationalizing Domain Names in Applications (IDNA) system, by which user applications, such as web browsers, map Unicode strings into the valid DNS character set using Punycode. In 2009 ICANN approved the installation of internationalized domain name country code top-level domains. In addition, many registries of the existing top level domain names (TLD)s have adopted the IDNA system.

Name 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 has at least one authoritative DNS server that publishes information about that domain and the name servers of any domains subordinate to it. The top of the hierarchy is served by the root name servers, the servers to query when looking up (*resolving*) a TLD.

Authoritative name server

An *authoritative* name server is a name server that gives answers that have been configured by an original source, for example, the domain administrator or by dynamic DNS methods, in contrast to answers that were obtained via a regular DNS query to another name server. An authoritative-only name server only returns answers to queries about domain names that have been specifically configured by the administrator.

In other words, an authoritative name server lets recursive name servers know what DNS data (the IPv4 IP, the IPv6 IP, a list of incoming mail servers, etc.) a given host name (such as "www.example.com") has. As just one example, the authoritative name server for "example.com" tells recursive name servers that "www.example.com" has the IPv4 IP address 192.0.43.10.

An authoritative name server can either be a *master* server or a *slave* server. A master server is a server that stores the original (*master*) copies of all zone records. A slave server uses an automatic updating mechanism of the DNS protocol in communication with its master to maintain an identical copy of the master records.

A set of authoritative name servers has to be assigned for every DNS zone. An NS record about addresses of that set must be stored in the parent zone and servers themselves (as self-reference).

When domain names are registered with a domain name registrar, their installation at the domain registry of a top level domain requires the assignment of a *primary* name server and at least one *secondary* name server. The requirement of multiple name servers aims to make the domain still functional even if one name server becomes inaccessible or inoperable.[10] The designation of a primary name server is solely determined by the priority given to the domain name registrar. For this purpose, generally only the fully qualified domain name of the name server is required, unless the servers are contained in the registered domain, in which case the corresponding IP address is needed as well.

Primary name servers are often master name servers, while secondary name servers may be implemented as slave servers.

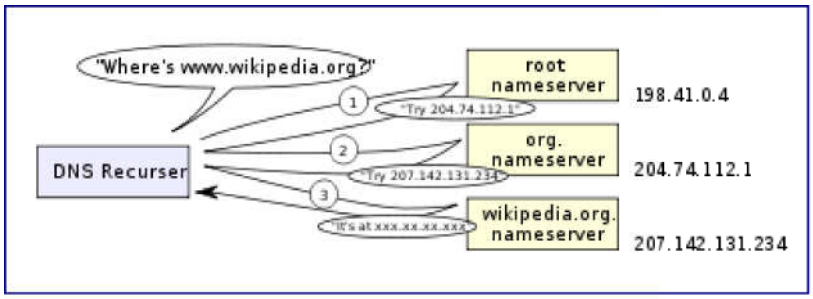
An authoritative server indicates its status of supplying definitive answers, deemed *authoritative*, by setting a software flag (a protocol structure bit), called the *Authoritative Answer* (*AA*) bit in its responses.[3] This flag is usually reproduced prominently in the output of DNS administration query tools (such as dig) to indicate *that the responding name server is an authority for the domain name in question.*[3]

Operation

Address resolution mechanism

Domain name resolvers determine the appropriate domain name servers responsible for the domain name in question by a sequence of queries starting with the right-most (top-level) domain label.

Domain Name System



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

The process entails:

1. A network host is configured with an initial cache (so called *hints*) of the known addresses of the root name servers. Such a *hint file* is updated periodically by an administrator from a reliable source.
2. A query to one of the root servers to find the server authoritative for the top-level domain.
3. A query to the obtained TLD server for the address of a DNS server authoritative for the second-level domain.
4. Repetition of the previous step to process each domain name label in sequence, until the final step which returns the IP address of the host sought.

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

The mechanism in this simple form would place a large 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 in DNS servers to overcome this problem, and as a result, root name servers actually are involved with very little of the total traffic.

Recursive and caching name server

In theory, authoritative name servers are sufficient for the operation of the Internet. However, with only authoritative name servers operating, every DNS query must start with recursive queries at the root zone of the Domain Name System and each user system would have to implement resolver software capable of recursive operation.

To improve efficiency, reduce DNS traffic across the Internet, and increase performance in end-user applications, the Domain Name System supports DNS cache servers which store DNS query results for a period of time determined in the configuration (time-to-live) of the domain name record in question. Typically, such *caching* DNS servers, also called *DNS caches*, also implement the recursive algorithm necessary to resolve a given name starting with the DNS root through to the authoritative name servers of the queried domain. With this function implemented in the name server, user applications gain efficiency in design and operation.

As one example, if a client wants to know the address for "www.example.com", it will send, to a recursive caching name server, a DNS request stating "I would like the IPv4 address for 'www.example.com'." The recursive name server will then query authoritative name servers until it gets an answer to that query (or return an error if it's not possible to get an answer)--in this case 192.0.43.10.

The combination of DNS caching and recursive functions in a name server is not mandatory; the functions can be implemented independently in servers for special purposes.

Internet service providers (ISPs) typically provide recursive and caching name servers for their customers. In addition, many home networking routers implement DNS caches and recursors to improve efficiency in the local network.

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 non-recursive query or a recursive query:

- A *non-recursive query* is one in which the DNS server provides a record for a domain for which it is authoritative itself, or it provides a partial result without querying other servers.
- A *recursive query* is one for which the DNS server will fully answer the query (or give an error) by querying other name servers as needed. 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 more simply by communicating only with a single name server. These simple resolvers (called "stub resolvers") rely on a recursive name server to perform the work of finding them.

Circular dependencies and glue records

Name servers in delegations are identified 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. If the name given in the delegation is a subdomain of the domain for which the delegation is being provided, there is a circular dependency. In this case the name server providing the delegation must also provide one or more IP addresses for the authoritative name server mentioned in the delegation. This information is called *glue*. The delegating name server provides this glue in the form of records in the *additional section* of the DNS response, and provides the delegation in the *answer section* of the response.

For example, if the authoritative name server for example.org is ns1.example.org, a computer trying to resolve www.example.org first resolves ns1.example.org. Since ns1 is contained in example.org, this requires resolving example.org first, which presents a circular dependency. To break the dependency, the name server for the top level domain org includes glue along with the delegation for example.org. The glue records are address records that provide IP addresses for ns1.example.org. The resolver uses one or more of these IP addresses to query one of the domain's authoritative servers, which allows it to complete the DNS query.

Record caching

The DNS Resolution Process reduces the load on individual servers by *caching* DNS request records for a period of time after a response. This entails the local recording and subsequent consultation of the copy instead of initiating a new request upstream. The time for which a resolver caches a DNS response is determined by a value called the time to live (TTL) associated with every record. The TTL is set by the administrator of the DNS server handing out the authoritative response. The period of validity may vary from just seconds to days or even weeks.

As a noteworthy consequence of this distributed and caching architecture, changes to DNS records do not propagate throughout the network immediately, but require all caches to expire and refresh after the TTL. RFC 1912 conveys basic rules for determining appropriate TTL values.

Some resolvers may override TTL values, as the protocol supports caching for up to 68 years or no caching at all. Negative caching, i.e. the caching of the fact of non-existence of a record, 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 value of the *minimum* field of the SOA record and the TTL of the SOA itself is used to establish the TTL for the negative answer.

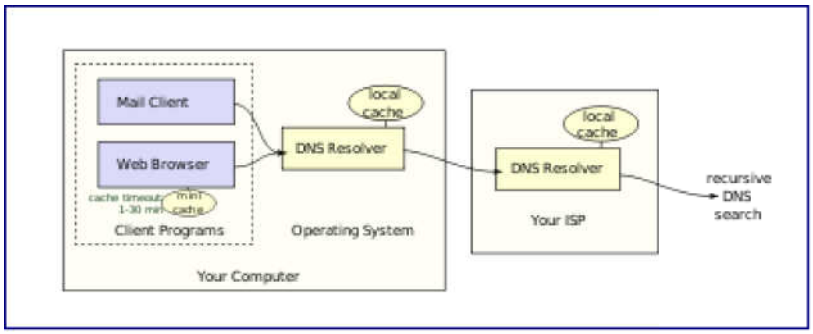
Reverse lookup

A reverse lookup is a query of the DNS for domain names when the IP address is known. Multiple domain names may be associated with an IP address. The DNS stores IP addresses in the form of domain names as specially formatted names in pointer (PTR) records within the infrastructure top-level domain arpa. For IPv4, the domain is in-addr.arpa. For IPv6, the reverse lookup domain is ip6.arpa. The IP address is represented as a name in reverse-ordered octet representation for IPv4, and reverse-ordered nibble representation for IPv6.

When performing a reverse lookup, the DNS client converts the address into these formats before querying the name for a PTR record following the delegation chain as for any DNS query. For example, assuming the IPv4 address 208.80.152.2 is assigned to Wikimedia, it is represented as a DNS name in reverse order: 2.152.80.208.in-addr.arpa. When the DNS resolver gets a pointer (PTR) request, it begins by querying the root servers, which point to the servers of American Registry for Internet Numbers (ARIN) for the 208.in-addr.arpa zone. ARIN's servers delegate 152.80.208.in-addr.arpa to Wikimedia to which the resolver sends another query for 2.152.80.208.in-addr.arpa, which results in an authoritative response.

Domain Name System

Client lookup



DNS resolution sequence

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

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 name servers 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

Some large ISPs have configured their DNS servers to violate rules, 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.[11]

Some applications, such as web browsers, maintain an internal DNS cache to avoid repeated lookups via the network. This practice can add extra difficulty when debugging DNS issues, as it obscures the history of such data. These caches typically use very short caching times on the order of one minute.[12]

Internet Explorer represents a notable exception: versions up to IE 3.x cache DNS records for 24 hours by default. Internet Explorer 4.x and later versions (up to IE 8) decrease the default time out value to half an hour, which may be changed in corresponding registry keys.[13]

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. Multiple 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 locations 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 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. Here is a hypothetical example blacklist:
  - 102.3.4.5 is blacklisted → Creates 5.4.3.102.blacklist.example and resolves to 127.0.0.1
  - 102.3.4.6 is not → 6.4.3.102.blacklist.example is not found, or default to 127.0.0.2
  - E-mail servers can then query blacklist.example through the DNS mechanism to find out if a specific host connecting to them is 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.
- 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 name servers exist, with additional "copies" of several of them distributed worldwide via anycast.
- Dynamic DNS (sometimes called DDNS) allows clients to update their DNS entry as their IP address changes, as it does, for example, when moving between ISPs or mobile hot spots.

Protocol details

DNS primarily uses User Datagram Protocol (UDP) on port number 53 to serve requests.[3] DNS queries consist of a single UDP request from the client followed by a single UDP reply from the server. The Transmission Control Protocol (TCP) is used when the response data size exceeds 512 bytes, or for tasks such as zone transfers. Some resolver implementations use TCP for all queries.

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.), an expiration time limit, a class, and some type-specific data. Resource records of the same type define a resource record set (RRset). The order of resource records in a set, returned by a resolver to an application, is undefined, but often servers implement round-robin ordering to achieve Global Server Load Balancing. DNSSEC, however, works on complete resource record sets in a canonical order.

When sent over an IP network, all records use the common format specified in RFC 1035:[14]

RR (Resource record) fields

Field	Description	Length (octets)
NAME	Name of the node to which this record pertains	(variable)
TYPE	Type of RR in numeric form (e.g. 15 for MX RRs)	2
CLASS	Class code	2
TTL	Count of seconds that the RR stays valid (The maximum is 2^{31}-1, which is about 68 years)	4
RDLENGTH	Length of RDATA field	2
RDATA	Additional RR-specific data	(variable)

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. A free standing @ is used to denote the current origin.

TYPE is the record type. It indicates the format of the data and it gives a hint of its intended use. For example, 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 specifies the mail server used to handle mail for a domain specified in an e-mail address.

RDATA is data of type-specific relevance, such as the IP address for address records, or the priority and hostname for MX records. Well known record types may use label compression in the RDATA field, but "unknown" record types must not (RFC 3597).

The *CLASS* of a record is set to IN (for *Internet*) for common DNS records involving Internet hostnames, servers, or IP addresses. In addition, the classes Chaos (CH) and Hesiod (HS) exist.[15] Each class is an independent name space with potentially different delegations of DNS zones.

In addition to resource records defined in a zone file, the domain name system also defines several request types that are used only in communication with other DNS nodes (*on the wire*), such as when performing zone transfers (AXFR/IXFR) or for EDNS (OPT).

Wildcard DNS records

The domain name system supports wildcard DNS records which specify names that start with the *asterisk label*, '*', e.g., *.example.[1][16] DNS records belonging to wildcard domain names specify rules for generating resource records within a single DNS zone by substituting whole labels with matching components of the query name, including any specified descendants. For example, in the DNS zone *x.example*, the following configuration specifies that all subdomains, including subdomains of subdomains, of *x.example* use the mail exchanger *a.x.example*. The records for *a.x.example* are needed to specify the mail exchanger. As this has the result of excluding this domain name and its subdomains from the wildcard matches, all subdomains of *a.x.example* must be defined in a separate wildcard statement.

The role of wildcard records was refined in RFC 4592, because the original definition in RFC 1034 was incomplete and resulted in misinterpretations by implementers.[16]

Protocol extensions

The original DNS protocol had limited provisions for extension with new features. In 1999, Paul Vixie published in RFC 2671 an extension mechanism, called Extension mechanisms for DNS (EDNS) that introduced optional protocol elements without increasing overhead when not in use. This was accomplished through the OPT pseudo-resource record that only exists in wire transmissions of the protocol, but not in any zone files. Initial extensions were also suggested (EDNS0), such as increasing the DNS message size in UDP datagrams.

Dynamic zone updates

Dynamic DNS updates use the UPDATE DNS opcode to add or remove resource records dynamically from a zone data base maintained on an authoritative DNS server. The feature is described in RFC 2136. This facility is useful to register network clients into the DNS when they boot or become otherwise available on the network. Since a booting client may be assigned a different IP address each time from a DHCP server, it is not possible to provide static DNS assignments for such clients.

Security issues

Originally, security concerns were not major design considerations for DNS software or any software for deployment on the early Internet, as the network was not open for participation by the general public. However, the expansion of the Internet into the commercial sector in the 1990s changed the requirements for security measures to protect data integrity and user authentication.

Several vulnerability issues were discovered and exploited by malicious users. One such issue is DNS cache poisoning, in which data is distributed to caching resolvers under the pretense of being an authoritative origin server, thereby polluting the data store with potentially false information and long expiration times (time-to-live). Subsequently, legitimate application requests may be redirected to network hosts operated with malicious intent.

DNS responses are traditionally not cryptographically signed, leading to many attack

possibilities; the Domain Name System Security Extensions (DNSSEC) modify DNS to add support for cryptographically signed responses. Several extensions have been devised to secure zone transfers as well. Another approach to DNS security is DNSCurve.

Some domain names may be used to achieve spoofing effects. For example, paypal.com and paypa1.com are different names, yet users may be unable to distinguish them in a graphical user interface depending on the user's chosen typeface. In many fonts the letter *l* and the numeral *1* look very similar or even identical. This problem is acute in systems that support internationalized domain names, since many character codes in ISO 10646, may appear identical on typical computer screens. This vulnerability is occasionally exploited in phishing.[17]

Techniques such as forward-confirmed reverse DNS can also be used to help validate DNS results.

Domain name 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 an administrative organization, operating a registry. A registry is responsible for maintaining the database of names registered within the TLD it administers. 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.

ICANN publishes the complete list of TLD registries and domain name registrars. Registrant information associated with domain names is maintained in an online database accessible with the WHOIS service. For most of the more than 290 country code top-level domains (ccTLDs), the domain registries maintain the WHOIS (Registrant, name servers, expiration dates, etc.) information. For instance, DENIC, Germany NIC, holds the DE domain data. Since about 2001, most gTLD (Generic top-level domain) registries have adopted this so-called *thick* registry approach, i.e. keeping the WHOIS data in central registries instead of registrar databases.

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

Some domain name registries, often called *network information centers* (NIC), also function as registrars to end-users. The major generic top-level domain registries, such as for the domains COM, NET, ORG, INFO, use a registry-registrar model consisting of many domain name registrars.[18][19] In this method of management, the registry only manages the domain name database and the relationship with the registrars. The *registrants* (users of a domain name) are customers of the registrar, in some cases through additional layers of resellers.

Internet standards

The Domain Name System is defined by Request for Comments (RFC) documents published by the Internet Engineering Task Force (Internet standards). The following is a list of RFCs that define the DNS protocol.

- RFC 920, *Domain Requirements* – Specified original top-level domains
- 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 1178, *Choosing a Name for Your Computer* (FYI 5)
- RFC 1183, *New DNS RR Definitions*
- RFC 1591, *Domain Name System Structure and Delegation* (Informational)
- 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 2100, *The Naming of Hosts* (Informational)
- 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* (BCP 20)
- RFC 2671, *Extension Mechanisms for DNS (EDNS0)*
- RFC 2672, *Non-Terminal DNS Name Redirection*
- RFC 2845, *Secret Key Transaction Authentication for DNS (TSIG)*
- RFC 3225, *Indicating Resolver Support of DNSSEC*
- RFC 3226, *DNSSEC and IPv6 A6 aware server/resolver message size requirements*
- RFC 3597, *Handling of Unknown DNS Resource Record (RR) Types*
- RFC 3696, *Application Techniques for Checking and Transformation of Names* (Informational)
- RFC 4343, *Domain Name System (DNS) Case Insensitivity Clarification*
- RFC 4592, *The Role of Wildcards in the Domain Name System*
- RFC 4635, *HMAC SHA TSIG Algorithm Identifiers*
- RFC 4892, *Requirements for a Mechanism Identifying a Name Server Instance* (Informational)
- RFC 5001, *DNS Name Server Identifier (NSID) Option*
- RFC 5452, *Measures for Making DNS More Resilient against Forged Answers*
- RFC 5625, *DNS Proxy Implementation Guidelines* (BCP 152)
- RFC 5890, *Internationalized Domain Names for Applications (IDNA):Definitions and Document Framework*
- RFC 5891, *Internationalized Domain Names in Applications (IDNA): Protocol*

- RFC 5892, *The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)*
- RFC 5893, *Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)*
- RFC 5894, *Internationalized Domain Names for Applications (IDNA):Background, Explanation, and Rationale* (Informational)
- RFC 5895, *Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008* (Informational)
- RFC 5966, *DNS Transport over TCP - Implementation Requirements*
- RFC 6195, *Domain Name System (DNS) IANA Considerations* (BCP 42)

Security

- RFC 4033, *DNS Security Introduction and Requirements*
- RFC 4034, *Resource Records for the DNS Security Extensions*
- RFC 4035, *Protocol Modifications for the DNS Security Extensions*
- RFC 4509, *Use of SHA-256 in DNSSEC Delegation Signer (DS) Resource Records*
- RFC 4470, *Minimally Covering NSEC Records and DNSSEC On-line Signing*
- RFC 5011, *Automated Updates of DNS Security (DNSSEC) Trust Anchors*
- RFC 5155, *DNS Security (DNSSEC) Hashed Authenticated Denial of Existence*
- RFC 5702, *Use of SHA-2 Algorithms with RSA in DNSKEY and RRSIG Resource Records for DNSSEC*
- RFC 5910, *Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP)*
- RFC 5933, *Use of GOST Signature Algorithms in DNSKEY and RRSIG Resource Records for DNSSEC*

References

1. RFC 1034, *Domain Names - Concepts and Facilities*, P. Mockapetris, The Internet Society (November 1987)
2. RFC 781, *Internet Protocol - DARPA Internet Program Protocol Specification*, Information Sciences Institute, J. Postel (Ed.), The Internet Society (September 1981)
3. RFC 1035, *Domain Names - Implementation and Specification*, P. Mockapetris, The Internet Society (November 1987)
4. RFC 3467, "Role of the Domain Name System (DNS)", J.C. Klensin, J. Klensin (February 2003).
5. Liu, Cricket; Albitz, Paul (2006). *DNS and BIND* (5th ed.). O'Reilly Media. p. 3. ISBN 978-0-596-10057-5.
6. Terry, Douglas B., et al. (June 12–15, 1984). "The Berkeley Internet Name Domain Server". *Summer Conference, Salt Lake City 1984: Proceedings*. USENIX Association Software Tools Users Group. pp. 23–31.
7. Internet Systems Consortium. "The Most Widely Used Name Server Software: BIND". History of BIND. Retrieved 28 July 2013.
8. Network Working Group of the IETF, January 2006, RFC 4343: Domain Name System (DNS) Case Insensitivity Clarification

9. RFC 3696, *Application Techniques for Checking and Transformation of Names*, J.C. Klensin, J. Klensin
10. "Name Server definition at techterms.com".
11. "Providers ignoring DNS TTL?". Slashdot. 2005. Retrieved 2012-04-07.
12. Ben Anderson: Why Web Browser DNS Caching Can Be A Bad Thing
13. "How Internet Explorer uses the cache for DNS host entries". Microsoft Corporation. 2004. Retrieved 2010-07-25.
14. RFC 5395, *Domain Name System (DNS) IANA Considerations*, D. Eastlake 3rd (November 2008), Section 3
15. RFC 5395, *Domain Name System (DNS) IANA Considerations*, D. Eastlake 3rd (November 2008), p. 11
16. RFC 4592, *The Role of Wildcards in the Domain Name System*, E. Lewis (July 2006)
17. APWG. "Global Phishing Survey: Domain Name Use and Trends in 1H2010." 10/15/2010 apwg.org
18. ICANN accredited registrars
19. VeriSign COM and NET registry