A survey of DNS

Praveen Yalagandula

October 4, 2000

Abstract

The Domain Name System (DNS) is the one of the Internet's fundamental building block. This is a distributed host information database that is responsible for translating names into IP addresses, routing mail to its proper destination, and many other services. We discuss basic working of DNS, different services exported by it and then describe the security and dynamic update extensions done to original simple DNS.

1 Introduction

During 1970s the ARPAnet was very small with just a few hundred hosts. All the information about these machines, like their addresses, etc., was maintained in a single file HOSTS.TXT. This file is maintained by Network Information Center (NIC) and was distributed from a single host. Members of ARPAnet periodically *ftp*ed this file to /etc/hosts (in unix) and thus were able to connect to different machines using just names. As the size of the ARPAnet grew, the traffic for ftp process grew a lot.

In 1984, Paul Mockapetris (who was assigned the responsibility for designing a new architecture) released two RFCs 882 and 883 describing the Domain Name System. These RFCs are then superseded by RFCs 1034 [4] and RFC 1035 [5]. These documents were augumented by many other RFCs which describe potential DNS security problems, methods for authorising domain data [3], mechanisms for dynamically updating name servers [1], secure dynamic update mechanisms [2], etc.

Domain Name System is basically a hierarchial and distributed host information database. The whole name space is divided into several domains and the maintenance of these domains are distributed to different authorities. The domains may have further subdomains and managed by different groups. For example, consider the machine name *begonia.cs.utexas.edu*. This name belongs to domain *cs.utexas.edu* which is under *utexas.edu* domain. The *utexas.edu* is a subdomain of *edu* domain, which is in turn a subdomain of root denoted by ".". Each domain in our example is maintained by different people. This way of dividing the name space and delegation allowed the scalibility much needed for naming service in the present Internet.

Name server (NS) is a machine which maintains a database of host information of the zone for which that device is the name server. For example, *deepthought.cs.utexas.edu* is a name server for the *cs.utexas.edu* domain. Anyone needing the IP address of a machine in *cs.utexas.edu* domain need to contact *deepthought.cs.utexas.edu* for the information.

Security is the main concern in the present Internet. How will we know that the address information returned by a query for a machine is not corrupted in the middle?. RFC 2535 [3] describes security extensions for DNS which enables the name servers to send a signature along the query result and the clients can verify the signature to check the integrity of the information received.

Initial RFCs (1034 and 1035) describe the name server in which the NS is provided with a configuration file with all the host information. With a change in host information, the name server has to be restarted to bring the changes into effect. In RFC 2136 [1], several interfaces are defined for dynamically updating the name server. Updates are allowed from only particular machines defined in the configuration file. But this is not reliable as this has many security vulnerabilities. RFC 2137 [2] describes how to use the techniques described in previous two RFCs (2136 and 2535) to restrict the updates by only those authorised devices to perform them. The authorised hosts are identified by the possession of cryptographic keys required for that update. We describe the mechanisms later in detail.

2 DNS Operation

The DNS has three major components:

1. Domain Name Space and Resource Records.
2. Name Servers
3. Resolvers

2.1 Domain Name Space and Resource Records

The domain name space is a tree structure. Each node and leaf of the domain name space tree has a set of information associated with it and query operations try to retrieve specific types of information from a particular set. Each node has a label and the domain name of a node is the list of the labels on the path from the node to the root of the tree (separated by dots). An example of a domain name space is shown in Figure 1.

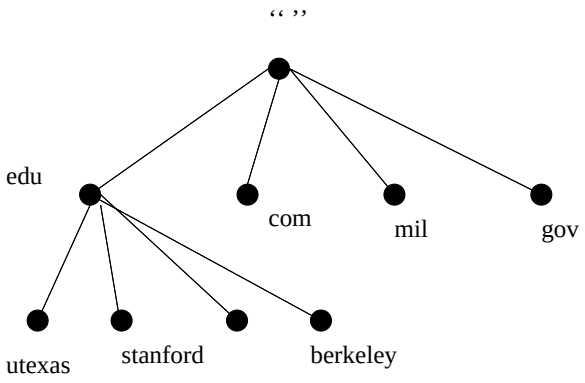

Figure 1: The structure of Domain Name Space

Each node in the domain name space has a set of resource information associated with it. This set is composed of separate resource records (RRs). Each RR will atleast have the following fields:

1. **owner :** The domain name where the RR is found.
2. **type :** The type of the resource found in this specific record. Some of the types are as follows:
   (a) *A :* a host address.
   (b) *CNAME :* canonical name of an alias.
   (c) *NS :* authoritative name server for the domain.
   (d) *SOA :* start of a zone of authority.
3. **class :** This identifies a protocol family or a instance of a protocol. *IN* is the presently used only value of this field which represents the Internet system.
4. **TTL :** The time to live field, which decides the length of the validity of the record. This is used to determine the time for which this can be cached in resolvers or other name servers before it can be discarded.
5. **RDATA :** Data part of the record. For type A records, this field will have IP address. For other types relevant data appears in this field.

Queries are the messages generated and sent to name servers for extracting a resource record from the DNS. A standard query specifies a target domain name (QNAME), query type (QTYPE), and query class (QCLASS) and asks for RRs which match. These queries can have wildcard character (“*”) in QTYPE and QCLASS to mean all records that match all types and classes, respectively.

2.2 Name Servers

Name servers are the repositories of information that make up the domain database. The database is divided up into sections called zones, which are distributed among the name servers. While name servers can have several optional functions and sources of data, the essential task of a name server is to answer queries using data in its zones. A given zone will be available from several name servers to insure its availability in spite of host or communication link failure. A given name server will typically support one or more zones, but this gives it authoritative information about only a small section of the domain tree. It may also have some cached non-authoritative data about other parts of the tree. The name server marks its responses to queries so that the requester can tell whether the response comes from authoritative data or not.

The difference between domain and zone is illustrated in Figure 2. The domain database is divided into zones by "cuts" made in the name space between nodes. The connected part after the cuts belong to a single zone. The data that described the zone has following parts: (i) Authoritative data for all nodes within the zone, (ii) Data that defines the top node of the zone, (iii) Data that describes delegated subzones, i.e., cuts around the bottom of the zone, and (iv) Data that allows access to name servers for subzones.

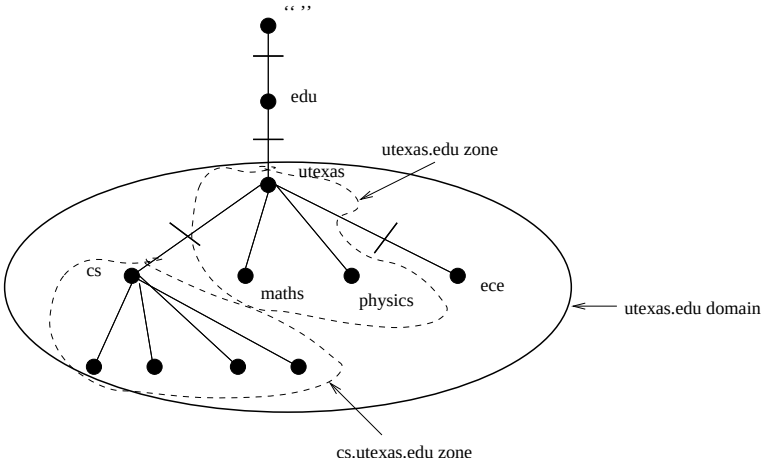

Figure 2: Domain vs Zone

2.2.1 Queries and Responses

The way that the name server answers the query depends upon what mode it is working in. Following are the two modes a name server generally supports:

- The simplest mode for the server is non-recursive, since it can answer queries using only local information: the response contains an error, the answer, or a referral to some other server "closer" to the answer. All name servers must implement non-recursive queries.
- The simplest mode for the client is recursive, since in this mode the name server acts in the role of a resolver and returns either an error or the answer, but never referrals. This service is optional in a name server, and the name server may also choose to restrict the clients which can use recursive mode.

The zone maintenance is done at one of the name servers assigned for that zone. This specific name server is called the primary name server and all other name servers for that zone are secondary name servers. All changes to zone data are made at the primary server and the secondary servers periodically update their data by contacting the primary server.

2.3 Resolvers

Resolvers are programs that interface user programs to domain name servers. In the simplest case, a resolver receives a request from a user program (e.g., mail programs, TELNET, FTP) in the form of a subroutine call, system call etc., and returns the desired information in a form compatible with the local host's data formats.

The resolver is located on the same machine as the program that requests the resolver's services, but it may need to consult name servers on other hosts. Because a resolver may need to consult several name servers, or may have the requested information in a local cache, the amount of time that a resolver will take to complete can vary quite a bit, from milliseconds to several seconds. A very important goal of the resolver is to eliminate network delay and name server load from most requests by answering them from its cache of prior results.

2.3.1 Stub Resolvers

One option for implementing a resolver is to move the resolution function out of the local machine and into a name server which supports recursive queries. This can provide an easy method of providing domain service in a PC which lacks the resources to perform the resolver function, or can centralize the cache for a whole local network or organization.

All that the remaining stub needs is a list of name server addresses that will perform the recursive requests. This type of resolver presumably needs the information in a configuration file, since it probably lacks the sophistication to locate it in the domain database. The user also needs to verify that the listed servers will perform the recursive service; a name server is free to refuse to perform recursive services for any or all clients. The user should consult the local system administrator to find name servers willing to perform the service.

This type of service suffers from some drawbacks. Since the recursive requests may take an arbitrary amount of time to perform, the stub may have difficulty optimizing retransmission intervals to deal with both lost UDP packets and dead servers; the name server can be easily overloaded by too zealous a stub if it interprets retransmissions as new requests. Use of TCP may be an answer, but TCP may well place burdens on the host's capabilities which are similar to those of a real resolver.

2.4 An example of DNS query

We describe an example of how a query by a host to find the ip address of *begonia.cs.utexas.edu* is resolved by a stub resolver sitting on the qurying machine. The process is illustrated in the Figure 3.

3 Extensions

In this section we will discuss various extensions proposed to the original DNS specification. We will mainly talk about the security extensions and mechanisms for allowing dynamic updates.

3.1 DNS Security Extensions

Security extensions to the DNS are first proposed in RFC 2065 and then this is recently obsoleted by RFC 2535 [3]. This document describes methods that provide data integrity and authentication to security aware resolvers and application through use of cryptographic digital signatures. These digital signatures are included in secured zones as resource records.

The extensions provide for the storage of authenticated public keys in the DNS. This storage of keys can support general public key distribution services as well as DNS security. The stored keys enable security aware resolvers to learn the authenticating key of zones in addition to those for which they are initially configured. Keys associated with DNS names can be retrieved to support other protocols. Provision is made for a variety of key types and algorithms. In addition, the security extensions provide for the optional authentication of DNS protocol transactions and requests.

We describe the three distinct services offered by the security extensions in detail in following sections.

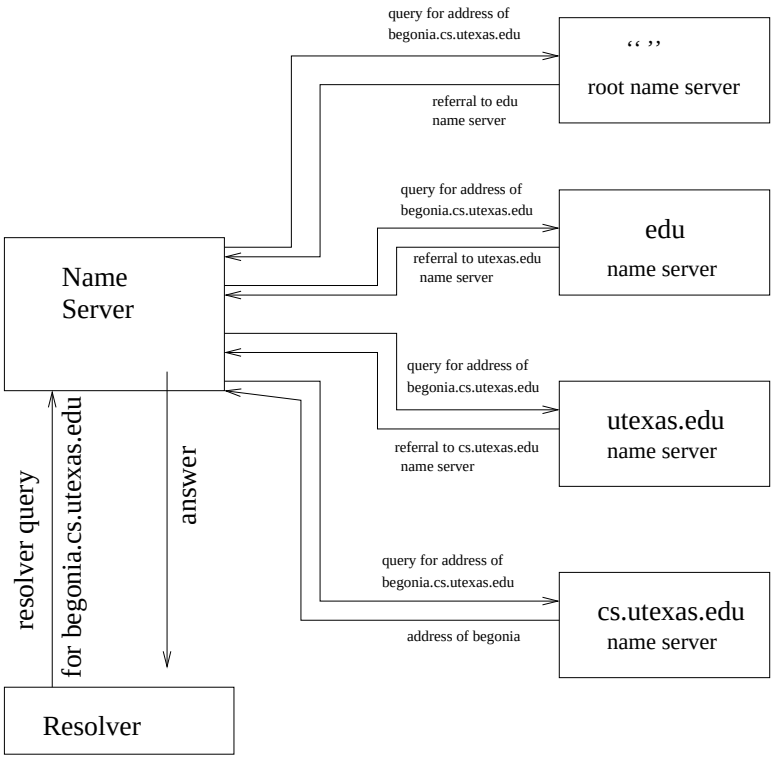

Figure 3: Resolution of *begonia.cs.utexas.edu*

3.1.1 Key Distribution

A new resource record format is defined to associate keys with DNS names. These resource records are called KEY RRs. This permits the DNS to be used as a public key distribution mechanism in support of DNS security itself and other protocols.

The data part (RDATA field) of KEY RRs consists of following things:

- **Flags :** These are used for identifying whether the key can be used in authentication, confidentiality, etc., and also to encode the name type (like whether the key is a zone key, a user key, etc.).
- **Protocol Octet :** These denote the protocol for which the keys should be used. Currently specified values include email, dnssec, IPSEC and Transport-Layer Security (TLS).
- **Algorithm :** This specifies the algorithm used.
- **Public Key**

The KEY RRs provide a way for distributing the public keys. But the distribution procedure itself is not secured. Hence the name servers provide a signature on the KEY RRs to authenticate the data. In next section, we describe how data origin authentication is done.

3.1.2 Data Origin Authentication

Authentication is provided by associating with resource record sets in the DNS, cryptographically generated digital signatures. Commonly, there will be a single private key that authenticates an entire zone but there might be multiple keys for different algorithms, signers, etc. If a security aware resolver reliably learns a public key of the zone, it can

authenticate, for signed data read from that zone, that it is properly authorized. The most secure implementation is for the zone private key(s) to be kept off-line and used to re-sign all of the records in the zone periodically.

The data origin authentication key(s) are associated with the zone and not with the servers that store copies of the data. That means compromise of a secondary server or, if the key(s) are kept off line, even the primary server for a zone, will not necessarily affect the degree of assurance that a resolver has that it can determine whether data is genuine.

A resolver could learn a public key of a zone either by reading it from the DNS or by having it staticly configured. To reliably learn a public key by reading it from the DNS, the key itself must be signed with a key the resolver trusts. The resolver must be configured with at least a public key which authenticates one zone as a starting point. From there, it can securely read public keys of other zones, if the intervening zones in the DNS tree are secure and their signed keys accessible.

The SIG or ”signature” resource record (RR) is the fundamental way that data is authenticated in the secure Domain Name System (DNS). The SIG RR unforgably authenticates an RRset of a particular type, class, and name and binds it to a time interval and the signer’s domain name. This is done using cryptographic techniques and the signer’s private key. The signer is frequently the owner of the zone from which the RR originated.

Every name in a secured zone will have associated with it at least one SIG resource record for each resource type under that name except for some RRs. A security aware server will attempt to return, with RRs retrieved, the corresponding SIGs. If a server is not security aware, the resolver must retrieve all the SIG records for a name and select the one or ones that sign the resource record set(s) that resolver is interested in.

3.1.3 Transaction and Request Authentication

The data origin authentication service described above protects retrieved resource records and the non-existence of resource records but provides no protection for DNS requests or for message headers.

If header bits are falsely set by a bad server, there is little that can be done. However, it is possible to add transaction authentication. Such authentication means that a resolver can be sure it is at least getting messages from the server it thinks it queried and that the response is from the query it sent (i.e., that these messages have not been diddled in transit). This is accomplished by optionally adding a special SIG resource record at the end of the reply which digitally signs the concatenation of the server’s response and the resolver’s query.

Requests can also be authenticated by including a special SIG RR at the end of the request. Authenticating requests serves no function in older DNS servers and requests with a non-empty additional information section produce error returns or may even be ignored by many of them. However, this syntax for signing requests is defined as a way of authenticating secure dynamic update requests, explained later, or future requests requiring authentication.

The private keys used in transaction security belong to the entity composing the reply, not to the zone involved. Request authentication may also involve the private key of the host or other entity composing the request or other private keys depending on the request authority it is sought to establish. The corresponding public key(s) are normally stored in and retrieved from the DNS for verification.

Because requests and replies are highly variable, message authentication SIGs can not be pre-calculated. Thus it will be necessary to keep the private key on-line, for example in software or in a directly connected piece of hardware.

3.2 Dynamic Updates

The Domain Name System was originally designed to support queries of a statically configured database. While the data was expected to change, the frequency of those changes was expected to be fairly low, and all updates were made as external edits to a zone’s Master File.

Using the UPDATE opcode in the request message, it is possible to add or delete RRs or RRsets from a specified zone. Prerequisites are specified separately from update operations, and can specify a dependency upon either the previous existence or nonexistence of an RRset, or the existence of a single RR. UPDATE is atomic, i.e., all prerequisites must be satisfied or else no update operations will take place.

3.3 Secure Dynamic Update

Previous section informs how DNS can be dynamically updated. But allowing updates to records from any server will cause lot of security problems. In this section we describe the methods proposed in RFC 2137 [2] that describe how to use digital signatures covering requests and data to secure updates and restrict updates to only those authorised to perform.

A dynamic secure zone is any secure DNS zone containing one or more KEY RRs that can authorize dynamic updates, i.e., entity or user KEY RRs with the flags denoting that the updates are allowed from such entity, and whose zone KEY RR flags indicates that updates are implemented.

Request signatures are always expected if the request is an update request. A client requesting a change to any resource record of a domain name should prove itself that it is authorised to do so. To authenticate itself, the client will sign the update request with his private key. The name server can then verify the validity of request by first looking up for KEY resource records in its database corresponding to that domain name. Once public key corresponding to that domain name is found, the server will verify the signature.

The important point to note is that this mechanism assumes that there exists a KEY RR for domain names which are expected to be updated dynamically.

4 BIND

BIND is an implementation of DNS provided for free by the Internet Software Consortium (`www.isc.org`). Current version of BIND implements the security extensions described previously. But the secure dynamic update is not implemented yet. A new version expected for beta release in the first quarter of 2000 will cover all security extensions and will provide secure dynamic update capability.

References

[1] D Eastlake 3rd. RFC 2136: Dynamic Updates in the Domain Name System (DNS UPDATE). `ftp://ftp.rfc-editor.org/in-notes/rfc2136.txt`, April 1997.

[2] D Eastlake 3rd. RFC 2137: Secure Domain Name System Dynamic Update. `ftp://ftp.rfc-editor.org/in-notes/rfc2137.txt`, April 1997.

[3] D Eastlake 3rd. RFC 2535: Domain Name System Security Extensions. `ftp://ftp.rfc-editor.org/in-notes/rfc2131.txt`, March 1999.

[4] P Mockapetris. RFC 1034: Domain Names - Concepts and Facilities. `ftp://ftp.rfc-editor.org/in-notes/rfc1034.txt`, Novermber 1987.

[5] P Mockapetris. RFC 1035: Domain Names - Implementation and Specification. `ftp://ftp.rfc-editor.org/in-notes/rfc1035.txt`, Novermber 1987.