

DNS

The Domain Name System on Windows NT 3.5

Kimble Consultancy Services Ltd
<http://www.kimble.co.uk/>
24th July 1995

David Harvey-George

Last printed 05/03/99 4:00 PM

Introduction

The Windows NT 3.5 resource kit includes a free implementation of the Internet **Domain Name System** (DNS). DNS is similar in function to the **Windows Internet Naming Service** (WINS), the main difference between them is that DNS was not designed to work with dynamic host configuration protocols such as **DHCP**. DNS databases are static, entries are added manually by a system administrator, necessitating a restart of the DNS server.

This is a problem for two classes of hosts:

1. Dial-up hosts where, due to local addressing limitations, a dynamic IP addressing scheme is used.
2. Nomadic hosts, which may connect to the 'network' from different nodes.

Obviously this is only a problem for clients performing name or address lookups but it does rather preclude such hosts running globally accessible Internet services, such as a Web server. In other words if you run Internet services within this environment there is no way for clients to find the IP address of your machine.

What is DNS then?

Chapter 2 of Go Web! includes a brief introduction to DNS. Basically it's a distributed database of mappings between Internet host names and addresses. Applications, using **resolver libraries** can convert human readable host names such as: `www.thomson.com` to the IP address they need in order to send packets over the Internet.

The database is distributed because each **domain** only maintains information about its own hosts, and it may even split itself into sub-domains and delegate authority to them.

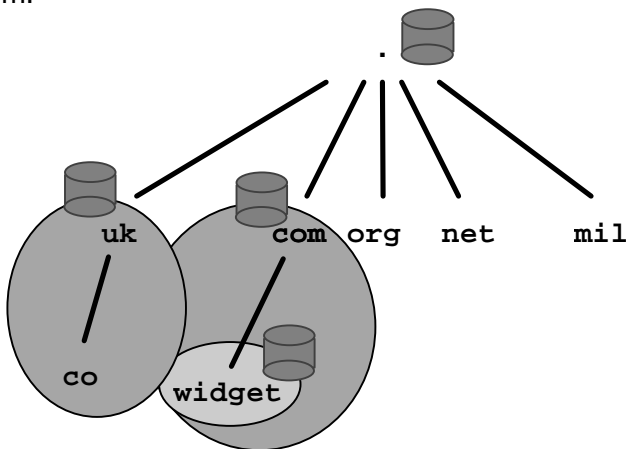


Figure 1: DNS Database

This has three big advantages:

1. There is no central database to become overloaded or crash. This makes the system fault tolerant in the event of local node or communications failures.
2. Each domain has control over its own hosts and can allocate, and reallocate addresses at will without recourse to some central authority.

3. The namespace is no longer flat obviating name clashes. The names `tardis.widget.co.uk` and `tardis.widget.com` are different even though the local administrator has named each host `tardis`.

Figure 1 illustrates some of the top level DNS domains. The `com` and `uk` domains contain sub-domains `co` and `widget` respectively. The `uk` **zone** includes the `co.uk` domain. That is the `uk` name server also maintains information for the `co.uk` domain. However the `.com` server has delegated authority for the `widget` domain to a server running within the `widget` domain. Thus a zone and a domain differ, a zone refers to the hosts over which the name server maintains **authoritative** information. So changes that occur on Widget's network can be made locally on the `widget` DNS database without bothering the administrators of the `com` domain.

If a machine in the `.co.uk` domain needs the IP address of a machine in the `widget.com` domain it will ask its local name server to perform the resolution. The `uk` name server doesn't hold information about the `widget.com` domain (unless it has cached it from an earlier request), instead it goes directly to a **root name server** in this case that of the `com` domain. It does this on behalf of the client and this is termed a **recursive query**. The `com` name server doesn't hold the information itself, instead it supplies the name and address of the server which is authoritative for the `widget.com` domain.

It can be seen that DNS fits in with the Internet philosophy of **delegation** by acting as a loose confederation. Each domain pretty much has total control over its affairs and may choose to delegate its name space.

Who needs DNS?

In the above example we saw that the `co.uk` domain relied on the `uk` name server to look after its affairs. Hosts in the domain were still consumers of DNS services, they just didn't want the bother of having their own name server. This is the usual state of affairs for Internet users with dial-up accounts, and may even be used by customers with ISDN or leased lines. However it is less flexible; system administrators must notify the name server administrator of new host names and addresses on their network. Many Internet Service Providers make a requirement for leased-line customers to run DNS. Which is really the whole point of the system.

What DNS isn't

DNS is simply a way of converting Internet host names and address from one to another. It relies on someone, somewhere, entering the information about their domain into DNS database files. These are plain text and (almost) human readable files.

DNS has no affect on packet routing on the Internet, indeed a domain may contain a fairly random set of IP addresses from various networks. A host on Widget's local network may live in the `co.uk` domain simply by inserting a line in the relevant DNS database.

However DNS has an integral role to play in the routing of electronic mail. For example `widget.com` may also be known as `widget.co.uk` requiring a mail routing record telling hosts which machines accept mail for that address. There is

no requirement for the Widget Corporation to be located in the United Kingdom for this to work.

Getting a Connection

Let's assume we're setting up a Web server to run on an NT system, connecting to `acme.net` our Internet Service Provider (ISP) using a leased line. In the US leased lines are measured with T numbers, T1 is a 1.544 Mbps leased line, when running Internet Protocols (which have an overhead) a T1 line has a capacity of about 180,000 bytes per second. That's great for a large server but is generally too big, and expensive for a small company. Telecommunications companies will split a T1 line 24 ways, called partial T1 the lowest capacity is 56 Kbps. Anyone who is good at arithmetic will notice some capacity has gone missing, this is because the Telcos use one bit in eight for sending control information. In Europe they use E numbers, E1 is 2 Mbps and the smallest line speed is 64 Kbps. This is sometimes dubbed kilostream. An ISDN line is really just a lowest capacity dial-up kilostream connection.

Where the ISP and customer live on the same exchange or cable network its possible to get very cheap connectivity. In the case of a Telco the link is effectively a direct connection using their copper wires through the exchange. This is referred to as a baseband connection.

Configuring DNS

So we're using `acme.net`. They've applied for and been allocated a Class C network address on our behalf, this is: 192.42.173.0. Although we're going to receive our IP packets from `acme` we actually want to under `com` domain. `Acme` have also registered our domain name: `widget.com`. This requires an addition to the `com` zones DNS database files:

```
widget                86400    IN NS      kimble.widget.com.
kimble.widget.com.   86400    IN A      192.42.173.1
```

This gives the host name of the server responsible for the `widget.com` domain and its IP address. We have to give the IP address of the name server because without that their is no way of making DNS lookups to retrieve other records. Typically we would be required to run a **secondary** DNS server which would also be entered here, more on that later.

MX records will be required for mail forwarding:

```
widget.com.          IN MX 0    relay.widget.com
                    IN MX 10   post.acme.net
*.widget.com.       IN MX 0    widget.com.
```

In this example mail to `name@widget.com` will be sent to the machine `relay.widget.com`, and if this is unavailable, to `post.acme.net`. This is `Widget's` Internet service provider and therefore the next closest site to `Widget`. We've also added a catch all for anyone mailing a machine directly. As would occur when replying to mail where **mail header re-writing** is not performed by the **message transfer agent** (MTA).

A method is also required for resolving IP addresses into host names. This is used by certain Internet utilities for security control and by Web masters analyzing log files. A special domain called: `in-addr.arpa` enables us to perform this reverse mapping. This domain is simply a tree of IP addresses, split on octet (byte) boundaries.

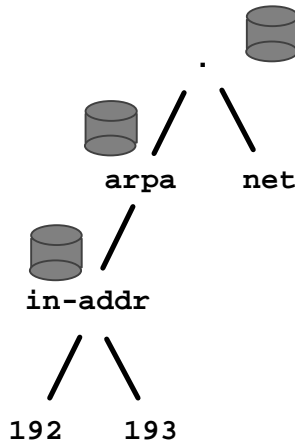


Figure 2: The `in-addr.arpa` domain

Again the reverse address space is delegated and an entry is required in the `in-addr.arpa` DNS database to specify the name server for the `192.42.172.0` network.

```
173.42.192.in-addr.arpa.    IN NS          kimble.widget.com
```

After that it's up to the administrators of the `widget.com` domain.

The BOOT file

The first step is to create a boot file in the `WINNT35\SYSTEM32\drivers\etc` directory (the `WINNT35` portion of this path varies and depends on where NT was originally installed).

The BOOT file controls the operation of the name server. The `directory` line specifies where the rest of the DNS database files are stored. This document follows the naming conventions used in the O'Reilly DNS & BIND book, in this case `\usr\local\named` was chosen. This has the advantage of keeping them separate from other files in the `.\etc` directory.

```

;
; Master configuration for DNS service
;
directory      c:\usr\local\named

cache          .                db.cache

primary       widget.com        db.widget
primary       173.42.192.inaddr.arpa  db.192.42.173
primary       0.0.127.in-addr.arpa  db.127.0.0
    
```

The `cache` file contains hints to the location of root names servers. Lines are of the form:

```
.          99999999   in ns          ns.internic.net.
```

Which gives `ns.internic.net.` as the location of a root name server. The `99999999` specifies a very long Time To Live. This is a fixed 'tablet of stone' type record so should stay current while DNS is running.

The Microsoft version of DNS comes with an example `cache` file, but the latest information can be obtained from:

<URL `ftp://nic.ddn.mil/netinfo/root-servers.txt`>.

Primary fields are then used to specify domains and information files. Any filename can be used, the convention `db` (database) followed by the domain is used here. This name server is only concerned with the `widget.com` domain, a reverse name file for the allocated Class C address is required as is a file for the local loopback address: `127.0.0.1`.

A single machine can be a primary for more than one domain and may also act as secondary for other domains. The location of all the database files is specified in the single `BOOT` file.

The `db.DOMAIN` file

```
widget.com.          IN SOA   blue.widget.com.
                    postmaster.blue.widget.com. (
                        1          ; serial
                        10800       ; refresh after 3 hours
                        3600        ; retry after 1 hour
                        604800      ; expire after one week
                        86500 )    ; Minimum TTL of 1 day
;
; nameservers
;
widget.com.          IN      blue.widget.com.
                    NS
;
; host addresses
;
localhost.widget.com. IN A    127.0.0.1
red.widget.com.      IN A    192.42.173.3
green.widget.com.    IN A    192.42.173.4
blue.widget.com.     IN A    192.42.173.1
;
; multihomed host
;
blue.widget.com.     IN A    192.42.173.1
blue.widget.com.     IN A    158.152.116.8
;
; Aliases so we can explicitly test both i/f of blue
;
blue1.widget.com.    IN A    192.42.173.1
blue2.widget.com.    IN A    158.152.116.8
```

The `db.DOMAIN` file contains the name to address mappings for the zone. It contains a number of different record types specified by the third field. Each file begins with a single Start of Authority (SOA) record. This record designates the start of a zone and gives information relating to the propagation of zone information to primary and secondary name servers. In our example the zone's domain name is `widget.com`. Note the trailing dot, this is a fully qualified domain name, DNS won't try adding further domain information to this field. A shortcut for this field is to use the `@` symbol, DNS will replace this with the 2nd (domain) field from the corresponding BOOT file record. The `IN` states that the record corresponds to the Internet network. We then give the record type, `SOA`, the fully qualified address of the name server and the Email address of the person responsible for administering DNS locally (note that dot is used to separate the user name: `postmaster`, from the host name). This information is useful as it can be retrieved by users if they suspect a problem with the DNS configuration. Finally a set of numbers give information about how often **zone transfers** should be performed and how long information should be cached by **resolvers**. The parenthesis permits this information to be split over several lines for readability. This record is pretty much standard and can be used as a template for other systems by simply replacing the domain and host information at the start.

Information about the machines which maintain the DNS files follows the SOA record, these machines are specified using name server (NS) records. For a small network there will normally be one primary and one secondary name server. The secondary is required in case the primary machine fails. The name server records don't distinguish between primary and secondary name servers. DNS will simply use the first working machine it finds.

Name servers should be located on reliable hosts, for a start the gateway machine should run a name server, probably the primary. File servers are also good locations for name servers, but if a **firewall** is in place they must be visible to the outside world.

Finally address records give the name to address mappings for the hosts located within the zone. The local naming scheme uses colors for machine names, we also insert a generic record for the name: `localhost.widget.com`. Any host within the domain looking up the `localhost` record will receive the address `127.0.0.1`, the **loopback** address.

Another interesting host is the name server itself. As it is the Internet gateway it has two IP addresses, one for each interface. Resolving its hostname returns two address records, one for each interface. Aliases specified with unique hostnames so a specific interface can be tested (*pinged*).

The `db.ADDR` file

Records for the reverse mapping of hosts within the local Class C address space are specified in the `db.ADDR` file. The file begins with a Start of Authority record because it is a domain in its own right, in this case: `173.42.192.in-addr.arpa`. Again the `in-addr.arpa` database will have name server records pointing to our reverse name server.

Pointer (PTR) records are used to specify the address to name mappings for the local machines.

```

173.42.192.in-addr.arpa.    IN SOA   blue.widget.com.
                             postmaster.blue.widget.com. (
                             1          ; serial
                             10800     ; refresh after 3 hours
                             3600      ; retry after 1 hour
                             604800    ; expire after one
                             week
                             86400 ) ; Minimum TTL of 1 day
;
; name servers
;
173.42.192.in-addr.arpa.    IN NS    blue.widget.com.
;
; Addresses point to
canonical names
;
3.173.42.192.in-addr.arpa.  IN PTR   red.widget.com.
4.173.42.192.in-addr.arpa.  IN PTR   green.widget.com.

```

The *db.127.0.0* file

Finally the *db.127.0.0* file performs the address to name mapping for the local loopback address. It requires its own DNS database file because it lives in the separate domain: *0.0.127.in-addr.arpa.* for the special 127.0.0.0 network. However the file is pretty much the same on all hosts and this template can easily be reused. The file is not that essential and really gets used by Internet utilities such as the Unix 'r' commands for security.

```

0.0.127.in-addr.arpa.    IN SOA   blue.widget.com.
                             postmaster.blue.widget.com. (
                             1          ; serial
                             10800     ; refresh after 3 hours
                             3600      ; retry after 1 hour
                             604800    ; expire after one
                             week
                             86400 ) ; Minimum TTL of 1 day
;
; name servers
;
0.0.127.in-addr.arpa.    IN NS    blue.widget.com.
;
; Addresses point to
canonical names
;
1.0.0.127.in-addr.arpa.   IN PTR   localhost.widget.com.

```

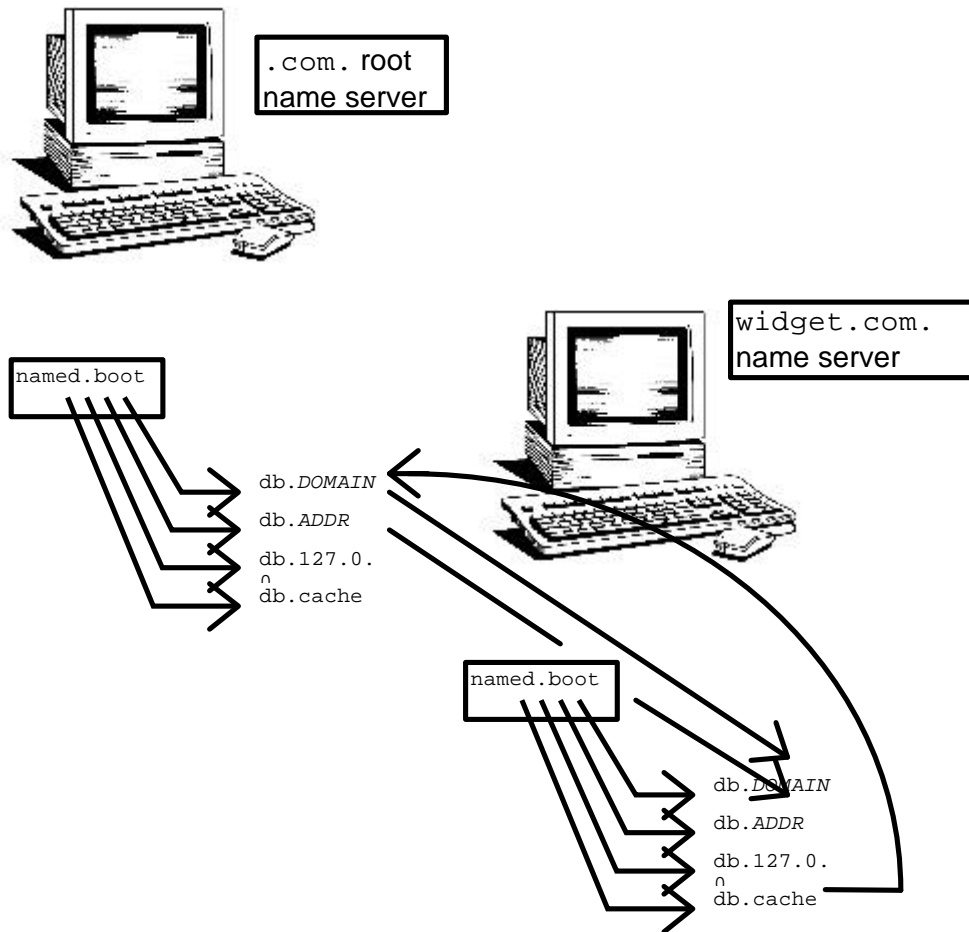


Figure 3: DNS File Structure.

Dynamic Addressing

Dynamic addressing is often used by Internet Service Providers (ISP) to allow more users to be serviced than their addressing scheme would ordinarily permit. For instance a Class C address (254 effective addresses) could service 64 modems (with a simple subnetting scheme). The ISP could have many more, maybe a thousand customers, sharing those modems. Each time a customer connects they will receive a different IP address. This kind of access is geared towards client use, e.g. Email or Web surfing, not running a full blown Web site.

The problem with dynamic addressing is how should address information be communicated to hosts. The Internet Protocols can't be used because the dynamic hosts don't yet have a valid IP address, the thing we are trying to communicate! A number of schemes exist to support dynamic addressing. Where **SLIP** (Serial Line Internet Protocol) is used on dial-up links may run some initial script, **PPP** (Point to Point Protocol) supports address negotiation. In the LAN environment variations on **RARP** (Reverse Address Resolution Protocol) and **BOOTP** have been used.

Another solution is the **Dynamic Host Configuration Protocol** (DHCP) which is derived from BOOTP. However this scheme does not specify how it should interoperate with naming services like DNS.

Microsoft has its own naming service for a homogenous Windows LAN environment, the previously mentioned WINS. WINS provides a naming service for hosts running Windows 3.1, Windows 95 or NT and supports dynamic host configuration. The WINS server itself runs on an NT server machine.

WINS

The recent huge growth in the Internet has probably entrenched DNS for the near future, in any event there is little chance of WINS taking over as it is currently restricted to the Windows platform and as it can't interoperate directly with DNS. Faced with the reality of the situation Microsoft has built connectivity between its implementation of DNS and WINS. This enables non-Windows hosts to obtain addressing information from the WINS database. It would appear that Microsoft has accepted that DNS will be used on the Internet and for non-Microsoft hosts.

WINS resolution is enabled for each WINS domain by using the special **\$WINS** directive following the Start of Authority record in the DNS domain database file. This is similar to the standard DNS **\$INCLUDE** directive, used to include other DNS database files. Hosts using DNS still talk to the local NT DNS server, and this retrieves information on their behalf from the WINS server, not unlike a recursive query. The dynamic nature of WINS networks is not ideally suited to the slower paced DNS world. In particular DNS servers use caching to reduce DNS lookups. Records are supplied with a time to live (TTL) field and are cached for this time. Because of the potential dynamic nature of WINS hosts this TTL field would have to be set to zero, which increases the load on DNS. For this reason WINS-DNS connectivity is most use within a local environment. Full time Internet Servers should really have fixed IP addresses and a DNS entry.

[**Note:** it's not clear from the documentation whether the DNS-WINS interaction is a one shot, at DNS boot time (which I suspect) or really is dynamic.]

WINS uses the DHCP to provide dynamic addressing capabilities and is geared to the LAN environment. Homogenous Windows networks may prefer to use WINS in preference to DNS for these reasons.

Limitations with Microsoft's 3.5 DNS

Name servers can be **primary** masters or **secondary** masters. A primary master uses local files to retrieve information for the zones its authoritative for. A secondary master doesn't have these local files, instead it pulls them over from the primary master. This is called a **zone transfer**. The Microsoft implementation of DNS in the 3.5 reskit cannot perform these zone transfers.

Secondary name servers are useful for spreading the load, for reliability and in large zones, to make sure a name server is *electronically* close.