In this chapter:
- *Domains and zones*
- *Setting up a name server*
- *Passive DNS usage*
- *Name server on a standalone system*
- *Name server on an end-user network*
- *Reverse lookup*
- *Slave name servers*
- *The next level down: delegating zones*
- *Messages from named*
- *Upgrading a Version 4 configuration*
- *Looking up DNS information*
- *Checking DNS for correctness*
- *DNS security*

21

The Domain Name Service

Ever since the beginning of the ARPAnet, systems have had both names and IP addresses. UNIX systems, as well as many others who have copied the BSD IP implementation, used the file */etc/hosts* to convert between names and addresses. This file contains a list of IP addresses and the corresponding host names, one per line.

It's clearly impossible to have an */etc/hosts* file that describes the complete Internet. Even if you had disk space, the number of updates would overload your network. The solution is a distributed database, the *Domain Name System*, or *DNS*. The most common implementation of DNS is *BIND*, the *Berkeley Internet Name Domain*.[1] You'll notice the word *Berkeley* in there. BIND is part of BSD, and it's about the only game in town. Despite these names, the daemon that performs the resolution is called *named* (the *name daemon*, pronounced ''name-dee'').

DNS provides the information needed to connect to remote systems in the form of *Resource Records*, or *RR*s. Unfortunately, the names of the records aren't overly intuitive.

- *A (Address) records* translate host names to IP addresses. For example, one A record tells you that *www.FreeBSD.org* (currently) has the IP address `216.136.204.117`. These are what most people think of when they hear the name DNS. The name specified in the A record is called the *canonical* name of the interface, and it should be the one to which the PTR record (see below) refers.

1. Does this sound like an acronym in search of a name?

- *PTR (Pointer) records* provide a translation from IP address to name. This process is also called *reverse lookup*.
- *MX (Mail Exchange) records* specify the IP addresses of mail servers for a domain.
- *SOA (Start Of Authority) records* define *zones*, which roughly correspond to domains. We'll look at the distinction between zones and domains below.
- *NS (Name Server) records* describe name servers for a zone.
- *HINFO (Hardware Information) records* describe the hardware and software that runs on a particular system.
- *CNAME (Canonical Name) records* describe alternative names for a system.

FreeBSD allows you to use both */etc/hosts* and DNS. One reason for this might be to have name resolution of local hosts at startup time: there's a chicken-and-egg problem with mounting NFS file systems before *named* is running.

The common objections to using DNS include:

- It's supposedly difficult to set up DNS configuration files.
- DNS supposedly generates a lot of network traffic.
- DNS supposedly causes a dial-on-demand system to dial all the time.

These statements are all untrue. We'll look at them in the rest of this chapter as we set up DNS for our reference network.

Domains and zones

In Internet parlance, a *domain* is a group of names ending with a specific *domain name*. We looked at domain names in Chapter 18, *Connecting to the Internet*, page 318. Note that, like file names, there are two kinds of domain names:

- A *fully qualified domain name* (*FQDN*) ends in a period (`.`). This domain name relates to the root domain `.` (a single period).
- A *relative domain name* relates to the current domain. You'll see them occasionally in the configuration files.

Most times, when you write a domain name, you intend it to be fully qualified. But if you write it without the terminating period, DNS will frequently append your own domain name. For example, if you specify a name like *freebie.example.org*, DNS won't find a fully qualified name: it's a misspelling of *freebie.example.org.*. As a result, it will look for the name *freebie.example.org.example.org*. It won't find it, of course, but it may spend a long time trying. The moral is simple: when writing DNS configuration files, always put a period (full stop) at the end of names that are fully qualified.

Zones

In many ways, a *zone* is the same thing as a domain: it's the subset of the DNS name space that is maintained by a specific set of name servers—in DNS-speak, name servers are *authoritative* for the zone. The difference is mainly in the way it's used. There is one exception, however: usually, a *subdomain* will have a different name server. This subdomain is part of the domain, but not of the zone.

For example, in our reference network, the name servers on *freebie* and *presto* are authoritative for *example.org*. The owner of the domain might give permission for somebody, maybe in a different country, to run a subdomain *china.example.org*, with name servers *beijing.china.example.org* and *xianggang.china.example.org*. Because there are different name servers, there are two zones: *freebie.example.org* would be authoritative for the zone *example.org*, but not for *china.example.org*. *beijing.china.example.org* and *xianggang.china.example.org* would be authoritative for the zone *china.example.org*, but not for *example.org*.

Setting up a name server

DNS service is supplied by the *name daemon*, called *named*. *named* can be run in a number of different modes. In this chapter, we'll concentrate on setting the appropriate configurations for our reference network. If you want to go further, check the following documents:

- The *BIND Online Documentation*, in the source distribution in the directory */usr/src/contrib/bind/doc/html/index.html*.
- *TCP/IP Network Administration*, by Craig Hunt (O'Reilly).
- *DNS and BIND*, by Paul Albitz and Cricket Liu (O'Reilly).

In the last few years, BIND has undergone some significant changes, mainly as a result of abuse on the net. The current release is Version 9, but FreeBSD still ships with Version 8. The differences are relatively minor: Version 9 introduces a number of new features, but the contents of this chapter should also apply to Version 9. The previous version was Version 4, and you'll still find a lot of old documentation referring to it. There were no Versions 5, 6 or 7, and the main configuration file changed its format completely in Version 8; even the name changed. We'll look at how to convert the formats on page 380. Before using the documentation above, make sure that it refers to the correct version of BIND.

Passive DNS usage

Not every system needs to run its own name daemon. If you have another machine on the same network, you can send requests to it. For example, in the reference network, *freebie* and *presto* may be running name servers. There's no particular reason for *bumble* and *wait*, both presumably slower machines, to do so as well. Instead, you can tell them to use the name servers on the other two machines.

To do this, make sure that you don't enable *named* in your */etc/rc.conf*, and create a file */etc/resolv.conf* with the following contents:

```
domain example.org
nameserver 223.147.37.1         # freebie
nameserver 223.147.37.2         # presto
```

Specify the IP addresses, not the names, of the name servers here. This is a classic chicken-and-egg problem: you can't access the name server to get its address until you know its address.

With this file in place, this machine will send all name server requests to *freebie* or *presto*. We'll look at how to configure them later.

Name server on a standalone system

If you only have a single machine connected to the network, and your own machine is part of the ISP's zone, you can use the *resolv.conf* method as well. This is a fairly typical situation if you're using a PPP or DSL link. It's still not a good idea, however. Every lookup goes over the link, which is relatively slow. The results of the lookup aren't stored anywhere locally, so you can end up performing the same lookup again and again. DNS has an answer to the problem: save the information locally. You can do this with a *caching-only name server*. As the name suggests, the caching-only name server doesn't have any information of its own, but it stores the results of any queries it makes to other systems, so if a program makes the same request again—which happens frequently—it presents the results much more quickly on subsequent requests. Set up a caching-only name server like this:

- Either rename or remove */etc/resolv.conf*, and create a new one with the following contents:

  ```
  nameserver 127.0.0.1                    local name server
  ```

- Put this line in */etc/rc.conf*:

  ```
  named_enable="YES"                    # Run named, the DNS server (or NO).
  ```

  If */etc/rc.conf* doesn't exist, just create one with this content.

- Create a file */etc/namedb/localhost.rev* containing:

```
$TTL 1d
@       IN SOA     @host@. root.@host@.  (
                                @date@  ; Serial
                                1h      ; Refresh
                                5m      ; Retry
                                100d    ; Expire
                                1h )    ; Negative cache
        IN NS      @host@.
1       IN PTR     localhost.@domain@.
```

We'll look at the meaning of this file in the next section. To create it, you can start with the file */etc/namedb/PROTO.localhost.rev*, which contains a template for this file. Replace `@host@` with the FQDN of your host (*freebie.example.org* in this example), `@date@` (the serial number) with the date in the form *yyyymmddxx*, where *xx* are a small integer such as 01,[1] and `@domain@` with *example.org.*. Make sure that the FQDNs end with a trailing period. Alternatively, you can run the script */etc/namedb/make-localhost*.

- Edit the file */etc/namedb/named.conf* to contain:

```
options {
        directory "/etc/namedb";

        forwarders {
                139.130.237.3;   139.130.237.17;
        };

zone "0.0.127.in-addr.arpa" {
        type master;
        file "localhost.rev";
};
```

/etc/namedb/named.conf should already be present on your system as well. It contains a lot of comments, but at the end there's a similar zone definition, which you can edit if you want. The addresses `139.130.237.3` and `139.130.237.17`. are the ISP's name server addresses. The `forwarders` line contains up to ten name server addresses.

- Start *named*:

```
# ndc start
```

1. We'll look at the serial number on page 368.

Name server on an end-user network

Of course, a simple caching-only name server won't work when you have your own domain. In fact, most of the authorities who allocate domain names won't even let you register an Internet domain unless you specify two functional name servers, and they'll check them before the registration can proceed. In this section, we'll look at what you need to do to run a ''real'' name server.

The first thing we need to do is to create a zone file for our zone *example.org*. We'll put it and all other zone files in a directory */etc/namedb* and call it */etc/namedb/db.example.org* after the name of the zone it describes.

The SOA record

The first thing we need is a record describing the *Start of Authority*. This defines a new zone. Write:

```
$TTL 1d
example.org.    IN SOA     freebie.example.org. grog.example.org.  (
                             2003031801 ; Serial (date, 2 digits version of day)
                             1d   ; refresh
                             2h   ; retry
                             100d ; expire
                             1h ) ; negative cache expiry
```

The first line, `$TTL 1d`, is relatively new. It's not strictly part of the SOA record, but it's now required to fully define the SOA. It specifies the length of time that remote name servers should cache records from this zone. During this time they will not attempt another lookup. In older versions of BIND, this value was stored in the last field of the SOA record below.

The remaining lines define a single SOA record. the name on the left is the name of the zone. The keyword `IN` means *Internet*, in other words the Internet Protocols. The BIND software includes support for multiple network types, most of which have now been forgotten. The keyword `SOA` defines the type of record. *freebie.example.org* is the master name server.

The next field, *grog.example.org*, is the mail address of the DNS administrator. ''Wait a minute,'' you may say, ''that's not a mail address. There should be an `@` there, not a `.`.'' That's right, but unfortunately DNS uses the `@` sign for other purposes, and it would be a syntax error in this position. So the implementors resorted to this kludge. To generate the mail ID, replace the first `.` with an `@`, to give you *grog@example.org*.

The *serial number* identifies this version of the zone configuration. Remote name servers first retreive the SOA record and check if the serial number has incremented before deciding whether to access the rest of the zone, which could be large. Make sure you increment this field every time you edit the file. If you don't, your updates will not propagate to other name servers. It's a good idea to use a format that reflects the date, as here: the format gives four digits for the year, two digits for the month, two for the day, and two for the number of the modification on a particular day. The serial number in this

example shows it to be the second modification to the zone configuration on 18 March 2003.

The remaining parameters describe the timeout characteristics of the zone. Use the values in the example unless you have a good reason to change them. The data formats for the records require all times to be specified in seconds, and in previous versions of BIND, this was the only choice you had. In current versions of BIND, you can use scale factors like d for day and h for hours in the configuration file. *named* converts them to seconds before transmission.

- The *refresh* time is the time after which a remote name server will check whether the zone configuration has changed. 1 day is reasonable here unless you change your configuration several times per day.
- The *retry* time is the time to wait if an attempt to load the zone fails.
- The *expire* time is the time after which a slave name server will drop the information about a zone if it has not been able to reload it from the master name server. You probably want to make this large.
- In previous versions of BIND, the last field was the *minimum time to live*. Now the $TTL parameter sets that value, and the last parameter specifies the *negative caching* time. If an authoritative name server (one that maintains the zone) reports that a record doesn't exist, it returns an SOA record as well to indicate that it's authoritative. The local name server maintains this information for the period of time specified by this field of the returned SOA record and it doesn't retry the query until the time has expired. The only way things can change here is if the remote hostmaster changes the DNS configuration, so it's reasonable to keep the negative cache time to about an hour.

The A records

The most obvious requirement are the IP addresses of the systems on the network. In the zone *example.org*, define the A records like this:

```
localhost       IN A    127.0.0.1           local machine, via loopback interface
freebie         IN A    223.147.37.1
presto          IN A    223.147.37.2
bumble          IN A    223.147.37.3
wait            IN A    223.147.37.4
gw              IN A    223.147.37.5
```

In practice, as we will see in the completed configuration file, we tend to put the A records further towards the end of the list, because they are usually the most numerous. It makes the file easier to read if we put them after the shorter groups of entries.

The NS records

DNS uses a special kind of record to tell where your name servers are. In our case, we're running name servers on *freebie* and *presto*. We could write:

```
                IN NS      freebie.example.org.
                IN NS      presto.example.org.
```

This would work just fine, but in fact we'll do it a little differently, as we'll see in the next section.

Nicknames

We're running a whole lot of services on the reference network, in particular a web server and an ftp server. By convention, a web server machine is called *www*, an ftp server is called *ftp*, and a name server is called *ns*. But they're both running on machines with different names. What do we do? We give our machines nicknames:

```
www             IN CNAME           freebie
ftp             IN CNAME           presto
```

We'd like to do the same with the name servers, but unfortunately DNS doesn't like that, and will complain about your DNS configuration all over the world if you make *ns* a CNAME. There's a good reason for this: if you use CNAME records to define your name servers, remote systems have to perform two lookups to find the address of the name server, one to retreive the CNAME and one to get the corresponding A record for the CNAME. Define new A records for them:

```
                IN NS      ns
                IN NS      ns1

ns              IN A       223.147.37.1
ns1             IN A       223.147.37.2
```

You'll note that we're using relative domain names in these examples. They are taken to be relative to the name that starts the SOA record.

The MX records

As we will see on page 493, you could send mail to hosts listed in an A record, but it's not a good idea. Instead, you should have at least two MX records to tell SMTP what to do with mail for your domain. This method has an added advantage: it allows you to rename individual machines without having to change the users' mail IDs. We'll take this advice and assume that all mail is sent to *user*`@example.org`. In addition, we'll use the ISP's mail server *mail.example.net* as a backup in case our mail server is down. That way, when it comes back up, the delivery will be expedited. The resulting MX records look like:

```
                IN MX      50  bumble.example.org.
                IN MX      100 mail.example.net.
```

The numbers 50 and 100 are called *preferences*. Theoretically you could make them 1 and 2, except that you might want to put others in between. A mail transfer agent sends mail to the system with the lowest preference unless it does not respond—then it tries the MX record with the next-lowest preference, and so on.

The HINFO records

Finally, you may want to tell the world about your hardware and this great operating system you're running. You can do that with the HINFO record:

```
freebie         IN HINFO   "Pentium/133"     "FreeBSD 4.0-CURRENT (4.4BSD)"
presto          IN HINFO   "Pentium II /233" "FreeBSD 3.2 (4.4BSD)"
bumble          IN HINFO   "Pentium/133"     "SCO OpenServer"
wait            IN HINFO   "Pentium Pro 266" "Microsoft Windows 95%"
gw              IN HINFO   "486/33"          "FreeBSD 3.2 (4.4BSD)"
```

Of course, telling the world the truth about your hardware also helps crackers choose the tools to use if they want to break into your system. If this worries you, don't use HINFO. It's still the exception to see HINFO records.

Putting it all together

In summary, our configuration file */etc/namedb/db.example.org* looks like:

```
; Definition of zone example.org
$TTL 1d
example.org.    IN SOA     freebie.example.org. grog.example.org.  (
                             2003031801 ; Serial (date, 2 digits version of day)
                             1d   ; refresh
                             2h   ; retry
                             100d ; expire
                             1h ) ; negative cache expiry

; name servers
                IN NS      ns
                IN NS      ns1

; MX records
                IN MX      50  bumble.example.org.
                IN MX      100 mail.example.net.

ns              IN A       223.147.37.1
ns1             IN A       223.147.37.2

; Hosts
localhost       IN A       127.0.0.1
freebie         IN A       223.147.37.1
presto          IN A       223.147.37.2
bumble          IN A       223.147.37.3
wait            IN A       223.147.37.4
gw              IN A       223.147.37.5

; nicknames
www             IN CNAME   freebie
ftp             IN CNAME   presto

; System information
freebie         IN HINFO   "Pentium/133"     "FreeBSD 4.0-CURRENT (4.4BSD)"
presto          IN HINFO   "Pentium II/233" "FreeBSD 3.2 (4.4BSD)"
bumble          IN HINFO   "Pentium/133"     "SCO OpenServer"
```

```
wait            IN HINFO   "Pentium Pro 266"  "Microsoft Windows 95%"
gw              IN HINFO   "486/33"          "FreeBSD 3.2 (4.4BSD)"
```

You'll notice that comment lines start with `;`, and not with the more usual #. Also, we have rearranged the MX records and the A records for the name servers. If we placed the MX records below the A records for the name servers, they would refer to *ns1.example.org*.

That's all the information we need for our zone *example.org*. But we're not done yet—we need another zone. Read on.

Reverse lookup

It's not immediately apparent that you might want to perform *reverse lookup*, to find the name associated with a specific IP address. In fact, it's used quite a bit, mainly to confirm that a system is really who it says it is. Many mail servers, including *FreeBSD.org*, insist on valid reverse lookup before accepting mail. We'll look at that in more detail in Chapter 27, on page 501. It's not difficult, but many systems, particularly those using Microsoft, don't have their reverse lookup set up correctly.

/etc/hosts is a file, so you can perform lookup in either direction. Not so with DNS: how can you know which name server is authoritative for the domain if you don't know its name? You can't, of course, so DNS uses a trick: it fabricates a name from the address. For the address `223.147.37.4`, it creates a domain name *37.147.223.in-addr.arpa*. The digits of the address are reversed, and the last digit is missing: it's the host part of the address. It asks the name server for this domain to resolve the name ***4***.*37.147.223.in-addr.arpa*.

To resolve the names, we need another zone. That means another file, which we'll call */etc/namedb/example-reverse*. It's not quite as complicated as the forward file:

```
$TTL 1d
@               IN SOA     freebie.example.org. grog.example.org.  (
                             2003022601 ; Serial (date, 2 digits version of day)
                             1d    ; refresh
                             2h    ; retry
                             100d  ; expire
                             2h )  ; negative cache
                IN NS      ns.example.org.
                IN NS      ns1.example.org.

1               IN PTR     freebie.example.org.
2               IN PTR     presto.example.org.
3               IN PTR     bumble.example.org.
4               IN PTR     wait.example.org.
5               IN PTR     gw.example.org.
```

In this case, the SOA record is identical to that in */etc/namedb/db.example.org*, with two exceptions: instead of the zone name at the beginning of the line, we have the `@` symbol, and the serial number is different—you don't normally need to update reverse lookup domains so often. This `@` symbol represents the name of the zone, in this case *37.147.223.in-addr.arpa.*. We'll see how that works when we make the

/etc/named/named.root file below. We also use the same name server entries. This time they need to be fully qualified, because they are in a different zone.

Finally, we have the PTR (reverse lookup) records. They specify only the last digit (the host part) of the IP address, so this will be prepended to the zone name. The host name at the end of the line is in fully qualified form, because it's in another zone. For example, in fully qualified form, the entry for *wait* could be written:

```
4.37.147.223.in-addr.arpa.              IN PTR     wait.example.org.
```

The distant view: the outside world

So far, we have gone to a lot of trouble to describe our own tiny part of the Internet. What about the rest? How can the name server find the address of, say, *freefall.FreeBSD.org*? So far, it can't.

What we need now is some information about other name servers who can help us, specifically the 13 *root name servers*. These are named *A.ROOT-SERVERS.NET.* through *M.ROOT-SERVERS.NET.*. They are described in a file that you can get from *ftp://ftp.rs.internic.net/domain/named.root* if necessary, but you shouldn't need to: after installing FreeBSD, it should be present in */etc/namedb/named.root*. This file has hardly changed in years—the names have changed once, but most of the addresses have stayed the same. Of course, it's always a good idea to check from time to time.

The named.conf file

So far, we have two files, one for each zone for which our name server is authoritative. In a large system, there could be many more. What we need now is to tell the name server which files to use. That's the main purpose of *named.conf*. There's already a skeleton in */etc/namedb/named.conf*. With the comments removed, it looks like:

```
options {
        directory "/etc/namedb";
        forwarders {
                127.0.0.1;
        };

zone "." {
        type hint;
        file "named.root";
};

zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "localhost.rev";
};

zone "domain.com" {
        type slave;
        file "s/domain.com.bak";
        masters {
                192.168.1.1;
        };
};
```

```
zone "0.168.192.in-addr.arpa" {
        type slave;
        file "s/0.168.192.in-addr.arpa.bak";
        masters {
                192.168.1.1;
        };
};
```

Each entry consists of a keyword followed by text in braces (`{}`). These entries have the following significance:

- The `directory` entry tells *named* where to look for the configuration files.
- The first zone is the top-level domain, `.`. It's a hint: it tells *named* to look in the file *named.root* in its configuration directory. *named.root* contains the IP addresses of the 13 top-level name servers.
- We've seen the entry for `0.0.127.IN-ADDR.ARPA` already on page 367: it's the reverse lookup for the localhost address.
- The `hint` entry specifies the name of the file describing the root servers (domain `.`).
- The zone entries for *domain.com* and *0.168.192.in-addr.arpa* define *slave name servers*. A slave name server addresses all queries to one of the specified *master name servers*. In earlier versions of DNS, a slave name server was called a *secondary name server*, and the master name server was called a *primary name server*. This is still current usage outside BIND, but you should expect this to change.

This file already contains most of the information we need. The only things we need to add are the information about the names of our zones and the location of the description file:

```
zone "example.org" {
        type master;
        file "db.example.org";
};

zone "37.147.223.in-addr.arpa" {
        type master;
        file "example-reverse";
};
```

When we've done that, we can start the name server with *ndc*, the *named* control program:[1]

```
# ndc start
new pid is 86183
```

If it's already running, we can restart it:

```
# ndc reload
Reload initiated.
```

1. In Release 9 of *named* it will change its name to *rndc*.

Starting or restarting the name server doesn't mean it will work, of course. If you make a mistake in your configuration files, it may not work at all. Otherwise it might start, but refuse to load specific zones. *named* logs messages with *syslog*, and if you are using the standard *syslog* configuration, the messages will be written to the console and to the file */var/log/messages*. After starting *named*, you should check what it said. *named* produces a number of messages, including:

```
Mar 18 15:01:57 freebie named[69751]: starting (/etc/namedb/named.conf).  named 8.3.
4-REL Wed Dec 18 13:38:28 CST 2002 grog@freebie.example.org:/usr/obj/src/FreeBSD/5-S
TABLE-FREEBIE/src/usr.sbin/named
Mar 18 15:01:57 freebie named[69751]: hint zone "" (IN) loaded (serial 0)
Mar 18 15:01:57 freebie named[69751]: master zone "example.org" (IN) loaded (serial
2003031801)
Mar 18 15:01:57 freebie named[69751]: Zone "0.0.127.in-addr.arpa" (file localhost.re
verse): No default TTL ($TTL <value>) set, using SOA minimum instead
Mar 18 15:01:57 freebie named[69751]: master zone "0.0.127.in-addr.arpa" (IN) loaded
 (serial 97091501)
Mar 18 15:01:57 freebie named[69751]: listening on [223.147.37.1].53 (rl0)
Mar 18 15:01:57 freebie named[69751]: listening on [127.0.0.1].53 (lo0)
Mar 18 15:01:57 freebie named[69752]: Ready to answer queries.
```

Note the warning output for *0.0.127.in-addr.arpa*: this is obviously an old-style zone file, as the serial number also suggests. It doesn't have a $TTL entry, so *named* defaults to the old-style behaviour and uses the last field (which used to be called ''minimum'') of the SOA record instead. This warning is not very serious, but you probably want a longer default TTL than you do for caching failed lookups, which is what the field is used for now.
What you don't want to see are error messages like:

```
May 10 14:26:37 freebie named[1361]: db.example.org: Line 28: Unknown type: System.
May 10 14:26:37 freebie named[1361]: db.example.org:28: Database error (System)
May 10 14:26:37 freebie named[1361]: master zone "example.org" (IN) rejected due to
errors (serial 1997010902)
```

As the last message states, this error has caused the zone to be rejected. Funny: if you look at line 28 of */etc/namedb/db.example.org*, it looks straightforward enough:

```
# System information
freebie         IN HINFO          "Pentium/133"     "FreeBSD 3.0-CURRENT (4.4BSD)"
presto          IN HINFO          "Pentium II/233" "FreeBSD 2.2.5 (4.4BSD)"
```

The problem here is that *named* doesn't use the standard UNIX convention for comments: the comment character is a semicolon (;), not a hash mark (#).

Most other configuration errors should be self-explanatory. On page 379 we'll look at messages that *named* produces during normal operation.

Slave name servers

A lot of software relies on name resolution. If for any reason a name server is not accessible, it can cause serious problems. This is one of the reasons why most registrars insist on at least two name servers before they will register a domain.

If you run multiple name servers, it doesn't really matter which one answers. So why a distinction between *master* and *slave* name servers? It's purely organizational: a master name server loads its data from the configuration files you create, as we saw above. A slave name server loads its data from a master name server if it is running. It saves the information in a private file so that if it is restarted while the master name server isn't running, it can reload information about the zones it is serving from this file. This makes it a lot easier to configure a slave name server, of course; everything we need is in */etc/namedb/named.conf*:

```
zone "." {
        type hint;
        file "named.root";
};

zone "example.org" {
        type slave;
        file "backup.example.org";
        masters {
                223.147.37.1;
        };
};

zone "37.147.223.in-addr.arpa" {
        type slave;
        file "backup.example-reverse";
        masters {
                223.147.37.1;
        };
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "localhost.rev";
};
```

Although this is a slave name server, there's no point in being a slave for *localhost*'s reverse mapping, so the last entry is still a `master`.

The numerical address is for *freebie.example.org*, the name server from which the zone is to be loaded. We use the numerical address because the name server needs the address before it can perform resolution. You can specify multiple name servers if you want. The *backup file* is the name of the file where the zone information should be saved in case the name server is restarted when the master name server is not accessible.

The next level down: delegating zones

In the previous example, we configured a name server for a single zone with no subzones. We did briefly consider what would happen if we created a subdomain *china.example.org*. In this section, we'll create the configuration files for this subzone and see how to link it to the parent zone, a process called *delegation*.

china.example.org

For the subdomain *china.example.org*, the same considerations apply as in our previous example: we have a domain without subdomains. Only the names and the addresses change.

In the following examples, let's assume that *china.example.org* has two name servers, *beijing.china.example.org* and *xianggang.china.example.org*. Let's look at the files we might have on these systems, starting with */etc/namedb/db.china.example.org*:

```
; Definition of zone china.example.org
$TTL 1d
@               IN SOA    beijing.china.example.org. zhang.china.example.org.  (
                             2001061701 ; Serial (date, 2 digits version of day)
                             1d    ; refresh
                             2h    ; retry
                             100d  ; expire
                             2h )  ; negative cache

; name servers
                IN NS      ns
                IN NS      ns1
ns              IN A       223.169.23.1
ns1             IN A       223.169.23.2
; MX records
                IN MX      50  xianggang.china.example.org.
                IN MX      70  bumble.example.org.
                IN MX      100 mail.example.net.

; Hosts
beijing         IN A       223.169.23.1
xianggang       IN A       223.169.23.2
shanghai        IN A       223.169.23.3
guangzhou       IN A       223.169.23.4
gw              IN A       223.169.23.5

; nicknames
www             IN CNAME   shanghai
ftp             IN CNAME   shanghai
```

Then, */etc/namedb/china-reverse*:

```
; Definition of zone china.example.org
@               IN SOA     beijing.china.example.org. zhang.china.example.org.  (
                             1997090501 ; Serial (date, 2 digits version of day)
                             86400   ; refresh (1 day)
                             7200    ; retry (2 hours)
                             8640000 ; expire (100 days)
                             86400 ) ; minimum (1 day)
```

```
; name servers
                IN NS      ns.china.example.org.
                IN NS      ns1.china.example.org.

; Hosts
1               IN PTR     beijing
2               IN PTR     xianggang
3               IN PTR     shanghai
4               IN PTR     guangzhou
5               IN PTR     gw
```

and finally */etc/namedb/named.conf*:

```
zone "." {
        type hint;
        file "named.root";
};

zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "localhost.rev";
};

zone "china.example.org" {
        type master;
        file "db.china.example.org";
};

zone "23.169.233.IN-ADDR.ARPA" {
        type master;
        file "china-reverse";
};
```

These files look very much like the corresponding files for *example.org*. The real difference happens in the configuration for *example.org*, not for *china.example.org*. We'll look at that next.

example.org with delegation

What does *example.org*'s name server need to know about *china.example.org*? You might think, ''nothing, they're separate zones,'' but that's not completely true. For a remote name server to find *china.example.org*, it first goes to *example.org*, so the parent domain must maintain enough information to find the child domain. This process is called *delegation*. The parent name server maintains NS records (''delegation records'') and corresponding A records (''glue records'') for the child zone. It might also be a good idea for the name servers for *example.org* to maintain a secondary name server for *china*: that way we can save a lookup to the master name servers for *china.example.org* most of the time. To do so, we add the following line to */etc/namedb/named.conf*:

```
zone "china.example.org" {
        type slave;
        file "backup.china";
        masters {
                223.169.23.1;
                223.169.23.2;
        };
};
```

```
zone "23.169.223.in-addr.arpa" {
        type slave;
        file "backup.china-reverse";
        masters {
                223.169.23.1;
                223.169.23.2;
        };
};
```

We add the following information to */etc/namedb/db.example.org*:

```
@               IN SOA     freebie.example.org. grog.example.org.  (
                             1997090501 ; Serial (date, 2 digits version of day)
                             86400   ; refresh (1 day)
                             7200    ; retry (2 hours)
                             8640000 ; expire (100 days)
                             86400 ) ; minimum (1 day)

china           IN NS      ns.china.example.org.
china           IN NS      ns1.china.example.org.

ns.china        IN A       223.169.23.1
ns1.china       IN A       223.169.23.2
```

We changed the information, so we also change the serial number of the SOA record so that the secondary name servers for *example.org* will reload the updated information.

We need to specify the addresses of the name servers as well. Strictly speaking they belong to the zone *china*, but we need to keep them in the parent zone *example.org*: these are the addresses to which we need to send any kind of query.

After changing the configuration like this, we restart the name server:

```
# ndc reload
```

We check the output, either by looking on the system console or by using the command `tail /var/log/messages`. We'll see something like:

```
Mar 18 15:23:40 freebie named[69752]: reloading nameserver
Mar 18 15:23:40 freebie named[69752]: master zone "china.example.org" (IN) loaded (s
erial 2001061701)
Mar 18 15:23:40 freebie named[69752]: Forwarding source address is [0.0.0.0].4673
Mar 18 15:23:40 freebie named[69752]: Ready to answer queries.
```

Messages from named

Once your *named* is up and running, it may still produce a number of messages. Here are some examples:

```
May 10 15:09:06 freebie named[124]: approved AXFR from [223.147.37.5].2872 for "exam
ple.org"
May 10 15:09:06 freebie named[124]: zone transfer of "example.org" (IN) to [192.109.
197.137].2872
```

These messages indicate that another name server has loaded the zone specified. This will typically be one of your secondary name servers. This should happen about as often as you have specified in your *refresh* parameter for the zone.

```
Mar 18 19:21:53 freebie named[69752]: ns_forw: query(tsolyani.com) contains our add
ress (freebie.example.org:223.147.37.1) learnt (A=example.org:NS=66.47.255.122)
```

This message indicates that the server indicated by the A record has asked us to forward a query whose name server list includes our own names or address(es). This used to be called a *lame delegation*. It's interesting that the address in this (real) message was *a.root-servers.net*, one of the 13 base servers for the whole Internet, which was probably forwarding a query from some other system. The server doesn't check the validity of the queries it forwards, so it's quite possible for them to be in error.

```
Mar 19 14:53:32 freebie named[13822]: Lame server on '182.201.184.212.relays.osirus
oft.com' (in 'relays.osirusoft.com'?): [195.154.210.134].53 'ns1-relays.osirusoft.c
om': learnt (A=216.102.236.44,NS=216.102.236.44)
```

This message indicates that a name server, listed as authoritative for a particular zone, is in fact not authoritative for that zone.

```
Sep 14 03:33:18 freebie named[55]: ns_forw: query(goldsword.com) NS points to CNAME
(ns-user.goldsword.com:) learnt (CNAME=199.170.202.100:NS=199.170.202.100)
```

As we saw above, a name server address should be an A record. The administrator of this system didn't know this, and pointed it to a CNAME record.

```
Sep 14 15:55:52 freebie named[55]: ns_forw: query(219.158.96.202.in-addr.arpa) A RR
negative cache entry (ns.gz.gdpta.net.cn:) learnt (NODATA=202.96.128.68:NS=202.12.28
.129)
```

This message indicates that the name server has already determined that the name server specified cannot be found, and has noted that fact in a *negative cache entry*.

Upgrading a Version 4 configuration

What we've seen so far applies to Versions 8 and 9 of *named*. The previous version was Version 4 (don't ask what happened to 5, 6 and 7; until Version 9 came along, there were rumours that the next version would be 16). Version 8 of *named* introduced a completely new configuration file format. If you have an existing DNS configuration from Version 4, the main configuration file will be called */etc/named.boot* or */etc/named/named.boot*. You can convert it to the *named.conf* format with the script */usr/sbin/named-bootconf*:

```
# named-bootconf < /etc/namedb/named.boot > /etc/namedb/named.conf
```

Looking up DNS information

You can use *dig*, *host* or *nslookup* to look up name information. It's largely a matter of preference which you use, but you should note that *nslookup* uses the resolver interface, which can result in you getting different results from what your name server would get. The output format of *dig* gets on my nerves, so I use *host*. Others prefer *dig* because it formulates the queries exactly the same way the name server does, and its output is more suited as input to *named*. For example, the command `dig @a.root-servers.net . axfr` produces a *named.root* file that *named* understands. We'll look briefly at *host*. Here are some examples:

```
$ host hub.freebsd.org                                  look up an A record
hub.freebsd.org has address 216.136.204.18
hub.freebsd.org mail is handled (pri=10) by mx1.freebsd.org
$  host 216.136.204.18                                  perform a reverse lookup
18.204.136.216.IN-ADDR.ARPA domain name pointer hub.freebsd.org
$ host ftp.freebsd.org                                  another one
ftp.freebsd.org is a nickname for ftp.beastie.tdk.net   this is a CNAME
ftp.beastie.tdk.net has address 62.243.72.50            and the corresponding A record
ftp.beastie.tdk.net mail is handled (pri=20) by mail-in1.inet.tele.dk
ftp.beastie.tdk.net mail is handled (pri=30) by mail-in2.inet.tele.dk
$ host -v -t soa freebsd.org                            Get an SOA record
Trying null domain
rcode = 0 (Success), ancount=1
The following answer is not authoritative:
freebsd.org              3066 IN SOA     ns0.freebsd.org hostmaster.freebsd.org(
                                         103031602       ;serial (version)
                                         1800    ;refresh period
                                         900     ;retry refresh this often
                                         604800  ;expiration period
                                         1800    ;minimum TTL
                        )
For authoritative answers, see:
freebsd.org              3066 IN NS      ns0.freebsd.org
freebsd.org              3066 IN NS      ns1.iafrica.com
freebsd.org              3066 IN NS      ns1.downloadtech.com
freebsd.org              3066 IN NS      ns2.downloadtech.com
Additional information:
ns0.freebsd.org         92727 IN A       216.136.204.126
ns1.iafrica.com         92727 IN A       196.7.0.139
ns1.downloadtech.com    92727 IN A       170.208.14.3
ns2.downloadtech.com    92727 IN A       66.250.75.2
ns2.iafrica.com         22126 IN A       196.7.142.133
```

There are a number of things to look at in the last example:

- We used the `-v` (verbose) option to get more information.
- Note the message `Trying null domain`. This comes because the name supplied was not a fully qualified domain name: the period at the end was missing. *host* decides that it looks like a fully qualified name, so it doesn't append a domain name to the name.
- The local name server at *example.org* already had the SOA record for *FreeBSD.org* in its cache; as a result, it didn't need to ask the name server that was authoritative for the zone. Instead, it tells you that the answer was not authoritative and tells you where you can get a valid answer.

- The output is in pretty much the same format as we discussed earlier in the chapter, but there are some numbers in front of `IN` in all the resource records. These are the *time-to-live* values for each individual record, in seconds. You can put these in the zone files, too, if you want, and they'll override the `TTL` value for the zone. In this printout, they specify how long it will be before the cached entry expires. Try it again and you'll see that the value is lower.

To get an answer from one of the authoritative name servers, we simply specify its name at the end of the request:

```
$ host -v -t soa freebsd.org. ns0.freebsd.org.
host -v -t soa freebsd.org. ns0.sd.org.
Using domain server:
Name: ns0.freebsd.org
Addresses: 216.136.204.126

rcode = 0 (Success), ancount=1
freebsd.org              3600 IN SOA     ns0.freebsd.org hostmaster.freebsd.org(
                                         103031602       ;serial (version)
                                         1800    ;refresh period
                                         900     ;retry refresh this often
                                         604800  ;expiration period
                                         1800    ;minimum TTL
                         )
```

This time we specified the names as FQDNs, so the message about the null domain no longer appears. Also, the TTL value is now the correct value for the record, and it won't change. Apart from that, the only difference is the missing message that the answer is not authoritative. The rest of the printout is the same.

You can also use the `-t` option to look for a specific record:

```
$ host -t mx freebsd.org.                          get the MX records
freebsd.org mail is handled (pri=10) by mx1.freebsd.org
$ host -t hinfo hub.freebsd.org.                   get HINFO records
$ host -t hinfo freefall.freebsd.org.
freefall.freebsd.org host information Intel FreeBSD
```

These invocations don't use the `-v` (*verbose*) option, so they're much shorter. In particular, *hub.freebsd.org* doesn't have any HINFO records, so we got no output at all.

Checking DNS for correctness

Several programs are available for diagnosing DNS configuration problems. They're outside the scope of this book, but if you're managing large DNS configurations, take a look at the collection at *http://www.isc.org/*.

DNS security

named was written at a time when the Internet was run by gentlemen. In the last few years, a relatively large number of security issues have been found in it. The FreeBSD project fixes these problems quickly, and you can expect that the version you get will have no known security issues. That can change, though: keep an eye on the security advisories from the FreeBSD project and update your name server if necessary.