Configuring the Raspberry Pi as a DNS Server under Raspbian Wheezy
(© Kenneth A Spencer)

1. Introduction

What is DNS?
DNS stands for "Domain Name Service". An Internet Domain Name is a textual method of identifying a network, or just a part of a network, connected to the Internet.
Domain Names usually comprise two parts: a *top level name* (TLD) and a *domain name*. It is usually of the form *kaspencer.com* (a fully qualified domain name) where *kaspencer* is the domain name and *com* is the TLD name. Originally, TLDs were exclusively American, but that is no longer so except for certain TLDs such as *.edu*, *.gov* or *.mil*.
The parts of a complete, or fully qualified domain name (FQDN), are separated by a full stop (.).
As the Internet has grown larger, control of parts of it have been handed over to country-level authorities. Such authorities have country-level domains (CLDs) under their control. Where the Internet is large in a country, it may be split into smaller levels such as *.gov.uk*, *.org.uk* or *.co.uk*. Thus *kaspencer.co.uk* would be expected to belong to a company based in the United Kingdom.

The essential purpose of the Internet Domain Name system is to make a more memorable means of identifying a specific network on the Internet than is provided by its IP (Internet Protocol) Address. An IP version 4 Address is a string of four "octets" separated by full stops. Thus *212.58.241.131* may well be the IP Address of the *bbc.co.uk* domain but that string of numbers is not as memorable as the name. Once your request is received on the target network at 212.58.241.131 a server will decide which website you have requested to visit and using the domain which you typed, it will direct you to the correct server and web pages.

Where larger, busier sites are concerned, it is not uncommon for a domain to be served by several servers: thus *www* may serve web pages, whilst *downloads* may provide file downloads. These server computers in the domain may be referred to as *hosts*. Thus the FQDN structure is *host.domain.tld*.

If the page that you are requesting is at a lower level on the web server, then the Unix convention of indicating a sequence of directory names separated by a forestroke ("/") ("slash" to our American friends) is followed. An example might be *www.bbc.co.uk/radio4/programmes*.

And so, in short: a Domain Name Server is a computer running a program (called a DNS Service) which will translate a Fully Qualified Domain Name (or even just a host name if it is on the same network) into an IP Address.

2. Why install your own DNS Server Software?

It will have become clear to those who have connected their computers to the Internet, that their PCs will connect to websites without issues, even without a DNS Server operating on their local area network (LAN). This is because there are DNS Servers all over the Internet which will answer DNS queries and permit your request for a website to be directed to the correct network and domain.

However, if you have several computers on your LAN, you will soon discover the advantage of having your own *local* DNS Service. With no local DNS Service, in order to refer to a computer by its name, rather than its IP Address, each computer on the LAN must hold a list of machine names allied to their IP Addresses, commonly in a file such as that found in `/etc/hosts`. That file must be kept up-to-date and consistent across all computers. Running a local DNS Service eliminates the need for that file, allowing machine names and IP Addresses to be maintained on just one computer, which can also store locally your commonly-visited Internet Domain details in a cache, for rapid retrieval.

3. Obtaining and installing a Domain Name Service program

There are several DNS programs which will run on the Raspberry Pi under Raspbian Wheezy. Many of them are quite similar. However, for this exercise the program `BIND` version 9 has been selected.

This author has set up Microsoft's DNS Service software successfully on many servers over many years. However, whilst the MS setup can be very dense and involves a steep learning curve, none was quite as fussy and involved as the Linux/Unix equivalents! For this reason I have tried to simplify the setup as far as possible - this inevitably leads to the omission of some of the more involved or difficult functions which the DNS Service may provide, but at the advantage of being useful for a typical LAN user.

First, obtain and install the BIND9 DNS Server software and some supporting utilities:

```
      apt-get install bind9 bind9utils
```

This installation should go ahead without error. When complete, your DNS Server is ready to setup.

4. Configuring your DNS Service

We will configure BIND9 as a LAN-only DNS Server - therefore all requests for domain services outwith your own LAN will be passed on to another DNS Server. This simplifies your tasks somewhat!

The `/etc/resolv.conf` file

One file that must be configured early on is the `/etc/resolv.conf` file. This file tells your computer about the available DNS Services. Edit the file in `nano` putting in the text which follows:

```
      nano /etc/resolv.conf
```

The text you need to enter is:

```
      search          domain.local
      nameserver      10.0.0.2       ;     this machine
      nameserver      10.0.0.253     ;     our router
      domain          domain.local   ;     choose any name
```

Enter the `search` directive at the top of the file, otherwise, you will not be able to use a hostname alone when communicating with a machine on your LAN. Specify your own local domain for your LAN. Choose freely as it does not require any registration. An example is `domain.local`.
The `nameserver` directive(s) specify the IP Addresses of the DNS Services on your LAN, and also in this case, a router, which will forward queries not resolved on your LAN to remote DNS Servers. You must replace these values with those appropriate to your own LAN setup.
The `domain` directive is usually the same as the `search` directive on a small LAN.

The `named.conf` File

This file is the first file in the BIND9 configuration. It generally performs nothing itself, but rather calls up other files in sequence. In our setup, you will use the `nano` editor to allow you to create the correct content:

```
      nano /etc/bind/named.conf
```

Our setting up of BIND9 requires the following content (only):

```
      // named.conf BIND9 configuration file
      //
      include "/etc/bind/named.conf.options" ;
      include "/etc/bind/named.conf.local" ;
```

Close `nano` and save your content with the usual `[Ctrl]+O` and `[Ctrl]+X`.
Note that all items and lines end with a semi-colon (`;`) and a carriage-return [`Enter`].
The two forestrokes (`//`) notify the system that the line is a comment-only and not for action.
The `include` directive is a standard instruction to read the file whose location and name is quoted. We will create the two files next.

The `named.conf.options` File

Create this file in the `nano` editor:

```
nano /etc/bind/named.conf.options
```

Then add the following content:

```
// named.conf.options BIND9 configuration file
//
options
     {
     directory "/var/cache/bind" ;
     forwarders
       { 8.8.8.8 ; 8.8.4.4 ; } ;
     auth-nxdomain no ;
     listen-on port 53 { 127.0.0.1; 10.0.0.2; } ;
     } ;
// end of file.
```

The most important contents of this file are:

```
forwarders { …; …; } ;
```

which specifies the DNS Server(s) to be used where the domain to be called is outwith the LAN. We have specified Google servers, but all ISPs also provide two name servers which you may also use.
The directive `auth-nxdomain no ;` indicates to its clients that your DNS Server may not be authoritative when a negative response is given after a local search.
The directive

```
listen-on port 53 { …; …; } ;
```

specifies the `port number` and the local loopback and machine network IP Addresses for the DNS Service on this machine. Do not forget the semi-colons after every item.
Close `nano` and save your content with the usual `[Ctrl]+O` and `[Ctrl]+X` when complete.

The `named.conf.local` File

Create this file in the `nano` editor:

```
nano /etc/bind/named.conf.local
```

Then add the following content:

```
// named.conf.local file for BIND9 configuration
//
zone "domain.local"
     {
     type master ;
     file "/etc/bind/zone.domain.local" ;
     } ;
//
zone "0.0.10.in-addr.arpa"
     {
     type master ;
     notify no ;
     file "/etc/bind/zone.0.0.10.in-addr.arpa" ;
     } ;
```

This file contains information about your local domain. Your domain is divided into `zone`s.

The most important is the Forward Lookup Zone, in this case named `domain.local` (but call it what you will, as long as you use its details correctly in all references to your network), whose details are held in the file given within the `zone`'s curly brackets.
The Forward Lookup Zone file contains the information necessary to enable the lookup of an IP Address from a machine name or an FQDN - the primary function of the DNS Service.
It is advisable to enter the full path for the file, which we will create shortly.

The second `zone` in the list is known as the Reverse Lookup Zone, and enables the lookup of a machine name from its IP Address.
The naming convention for the Reverse Lookup Zone is to reverse the first three octets of your network address (in my case `10.0.0.0` which reverses to `0.0.10`) and to append "`.in-addr.arpa`".
I have named the file which contains the details for the `zone` in a similar manner.
We will create that file shortly.

The `zone` file `zone.domain.local`
Create the Forward Lookup Zone file in the `nano` editor:

```
     nano /etc/bind/zone.domain.local
```

Use your own domain in the name of this file. The domain can be whatever you wish - it is on your private LAN and so it need not be a registered Internet domain - but it must be consistently referred to across your network configuration. The file which is given below shows a minimum content of servers and host entries. Simply repeat the pattern shown for the hosts in your own domain.

```
; zone.domain.local BIND9 configuration file.
;
$TTL 604800
@    IN   SOA  node2.domain.local. root.localhost. (
     201212037 ; serial no. (increment by +1 after every edit!)
        604800 ; refresh
         86400 ; retry after failure
       2419200 ; expired
        604800); TTL negative cache
;
@    IN   NS   node2.domain.local.
@    IN   A    127.0.0.1
;
; A records - Local machines and addresses:
; Servers:
node1          IN   A    10.0.0.1  ; file server
node2          IN   A    10.0.0.2  ; DNS and DHCP server
node3          IN   A    10.0.0.3  ; RPi web server
node4          IN   A    10.0.0.4  ; RPi media centre
;
; Windows Workstations:
node11         IN   A    10.0.0.11 ;
node12         IN   A    10.0.0.12 ;
```

This file is the longest that we have dealt with so far, and it seems to have more content. However, much of the content is in a repetitive pattern. The various parts of the file have the following meaning:
The semi-colons ("`;`") signify that everything the follows on the line are comments.

`$TTL n`	indicates, by default in second, the Time To Live. It states how long the data should stay in the DNS cache;
`@`	represents the current domain;
`IN`	signifies that an Internet record is being defined;
`SOA`	signifies that the host is the Start Of Authority for the domain;

These items are followed by the FQDN of the DNS Server being defined, and the support email address for the domain. (Note that the usual email @ symbol is replaced by a full stop in this definition).

`serial no`	this must be incremented by +1 after each edit so that the client can be up-to-date;

There are then a number parameters (in second by default) defining the behaviour of the DNS Server:

`refresh`	the duration after which a slave DNS Server should refresh its data;
`retry`	the period, after a refresh failure, which a slave DNS Server should retry;
`expired`	the period, after retry failures, indicating that a DNS Sever is no longer available
`TTL negative`	the period for which a negative search result should be stored.

What then follows are two lines defining the FDQN of the `nameserver` and its local loopback address (nearly always `127.0.0.1`). On these lines, the symbols have the following meanings:

`@`	represents the current domain;
`IN`	signifies that an Internet record is being defined;
`NS`	signifies that a `nameserver` record for the domain is being defined;
`A`	signifies that an IP Address record for a host is being defined.

There then follows a list of hosts whose IP Addresses are defined for serving to DNS Clients by the DNS Service. The standard form for these records is:

```
     hostname  IN   A    IP Address     ; comment
```

Provided that the file `/etc/resolv.conf` has been setup as advised above, the DNS Service will resolve both a hostname alone and in a FDQN.

<u>The `zone` file `zone.0.0.10.in-addr.arpa`</u>

Create the <u>Reverse Lookup Zone</u> file in the `nano` editor:

```
     nano /etc/bind/zone.0.0.10.in-addr.arpa
```

The file which is given below shows a minimum content of servers and host entries. Simply repeat the pattern shown for the hosts in your own domain.

```
; file zone.0.0.10.in-addr.arpa BIND9 configuration file.
;
$TTL 604800
@    IN   SOA  node2.domain.local. root.localhost. (
     201212038 ; serial no. (increment by +1 after each edit!)
      604800   ; refresh
       86400   ; retry after failure
     2419200   ; expired
      604800); TTL negative cache
;
@    IN   NS   node2.domain.local.
;
; PTR records - Local machines and addresses:
; Servers:
1         IN   PTR  node1.domain.local. ; file server
2         IN   PTR  node2.domain.local. ; DNS and DHCP server
3         IN   PTR  node3.domain.local. ; RPi web server
4         IN   PTR  node4.domain.local  ; RPi media centre
;
; Workstations:
11        IN   PTR  node11.domain.local.     ;
12        IN   PTR  node12.domain.local.
;
```

Note the presence of the full stop immediately after each pointer record. These are essential.
Most of the symbols in this file are defined above. However, the `PTR` symbol indicates that the record is a reverse pointer record, enabling the return of a hostname from a submitted IP Address.

Missing Symbols in these files

As mentioned at the start of this document, we decided to serve DNS clients on your LAN only. For this reason, there are some functions of the DNS Service which are not present in the files which have been defined.
NB: you do not need to add any of the items below for our LAN-only configuration to work.
Furthermore, we suggest that the meaning and significance of the extra functions and symbols are fully investigated before use.
The missing functions and symbols include:

`MX` this symbol indicates that the record being defined is that of a mail server (mail exchange). The MX symbol is usually followed by a number, such as 10, which indicates the priority of the server. Rather than using 1 or 2, the use of 10 permits mail servers to be added into the domain without having to edit the priority indicator for all existing mail server records. An MX record could take the form:

```
mail        IN  A       10.0.0.5 ; mail server Address record
IN          MX  10      mail     ; mail server MX record
```

`CNAME` this symbol indicates that the record being defined represents a Canonical Name of a host. A Canonical Name can be considered to be an alternative or alias hostname. If, for example, the web server www also functions as an `ftp` server, the following records could be created:

```
www         IN  A       10.0.0.6 ; web server Address record
ftp         CNAME       www      ; ftp server canonical name
```

Root hint servers

The final item which we have not covered in these files is that of incorporating the so-called *root hint* servers into the DNS Service. There are a series of servers connected to the Internet across all the continents of the world. These servers carry key DNS records for major DNS Servers. They are updated each 24 hours using a series of tapes flown around the world, and they provide the means of ensuring that each new Internet domain that is registered can have its IP Address made available to each client worldwide. Our configuration does not require the incorporation of the root servers into its definition as we are expecting the remote DNS Server(s) which we have specified as forwarders to be up-to-date.
However, it is worth noting that when the BIND9 server software package is downloaded and installed, a file, called either `named.root` or `db.root` is placed in the `/etc/bind` directory. That file contains a definition of the hint servers. The root zone can be incorporated into our DNS configuration by including the following additional zone into the file `/etc/bind/named.conf` immediately before the first zone:

```
zone "." ;
{
type hint ;
file "/etc/bind/db.root" ;
} ;
```

In this definition, the address "`.`" represents the "root" of the Internet. The `db.root` file (which may be present as `named.root` instead) contains the records for the servers in the root.

Securing Transfers with a Key

When BIND9 installs, you will note that a so-far unused file, commonly named `rndc.key` has been placed in the `/etc/bind` directory. As our configuration does not support the use of our DNS Service by clients outwith our LAN, we do not need to verify queries. However, the key can be used for that purpose by adding the following line to the end of the `/etc/named.conf` file:

```
include        /etc/bind/rndc.key
```

Secondary (Slave) DNS Servers
On a larger network it is useful to have more than one DNS Server so as to allow the primary name server to be taken offline in maintenance or hardware failure, without jeopardising the ability of clients to access remote Internet Domains. However, you may have, only one Authoratitive server - the slave DNS Servers will get their updates from the master server, using the timing parameters detailed in the Forward and Reverse Lookup Zones sections. As most Raspberry Pi users will have no requirement for secondary DNS Servers, we will not complicate our document with this aspect of BIND9 configuration.

5. Starting and Stopping your DNS Service

Starting and stopping the BIND9 DNS Server is easy:

To Stop the server: `service bind9 stop`

To Start the server: `service bind9 start`

If there are no errors reported, proceed to check the operation of the DNS Service using the procedures listed in the section below. Section 7. also contains hints which will help the investigation of any errors.

6. Testing the Configuration

First of all, it must be said that BIND9 is very fussy indeed about the way in which the configuration files are laid out and presented. If there is any problem at all, it will fail to run properly, but it may issue no visible warning. A very quick and easy test is to enter the following commands:

```
host node1
```

and `host node1.domain.local` where `node1` is one of your hosts and `domain.local` is the name your domain on your LAN.

Repeat the exercise quoting all hosts on your LAN. They should all evoke a correct response in the form: `node1.domain.local has address 10.0.0.1`

Next, you can use the nslookup command as follows:

```
nslookup node1
```

and `nslookup node1.domain.local`

Both commands should evoke a response of the form:

```
server:   10.0.0.2
address:  10.0.0.253
name:     node1.domain.local
address:  10.0.0.1
```

There are a few other utilities which may be useful in testing your DNS Service. On such is the command `dig`. You can install `dig` using the following command:

```
apt-get install dnsutils
```

Once installed, try typing:

```
dig  node1.domain.local
```

The command should evoke a response of the form:

```
; <<>> DiG 9.8.4-P1 <<>> node1.domain.local
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25400
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1
;; QUESTION SECTION:   ;node1.domain.local.        IN  A
;; ANSWER SECTION:      node1.domain.local. 604800  IN  A   10.0.0.1
;; AUTHORITY SECTION:   domain.local.       604800  IN  NS  node2.domain.local.
;; ADDITIONAL SECTION:  node2.domain.local. 604800  IN  A   10.0.0.2
;; Query time: 3 msec
;; SERVER: 10.0.0.2#53(10.0.0.2)
;; WHEN: Thu Jan 10 15:31:16 2013
;; MSG SIZE  rcvd: 86
```

The response looks complicated at first sight, but a short study should reveal the meaning of each line.

7. Troubleshooting your Configuration

If your DNS Server fails to function, there are several avenues of investigation:

1. Read carefully through each configuration file. Check for the presence of forestrokes ("`//`") and semicolons ("`;`") where indicated in this document;
2. Ensure that you have tapped the Carriage Return key [Enter] at the end of each line;
3. Ensure that all filenames are correctly stated - give an absolute path in each case.

Remember that BIND9 is very fussy indeed! But if you still cannot see any errors, try this command:

```
named-checkconf
```

It will scan through all of your configuration files and report any errors which it may find. However, it doesn't find all errors!

If you still have errors, try the following two commands in sequence:

`service bind9 restart` followed immediately by

```
tail /var/log/syslog
```

The final command is your last line of defence. If there are errors, they will be reported with reference to the filename(s) of the file(s) containing the error(s), the line number in the file, and a stated line fragment indicating the section of the line which was causing the problem.
Load each file into the `nano` editor in turn, and visit the line(s) in question. In `nano`, you can execute a `goto-line` command by tapping `[Ctrl] +"_"` followed by the entry of the line number.

If the error(s) persist, you must re-execute the pair of commands above, and re-examine any file or line that seems to cause the issue. Remember also that sometimes an error might just be caused by the entry on the preceding or succeeding line, so check all around the reported error location.

If this configuration of a BIND9 DNS Service is your first contact with textual configuration or text-line programming, you may be in for a shock! Examining these files for errors must be done most diligently and with considerable thought. It will however, work in the end!

Finally I hope that you enjoy using your Raspberry Pi in its new rôle!