PicoDNS User Manual

Adam Risi

February 17, 2009

Contents

1 Introduction

PicoDNS is a tiny DNS server programmed by Adam Risi. It is currently being distributed as part of the torus project, a distributed peer to peer program for resource sharing. PicoDNS was developed in order to allow peers on the torus network to resolve usernames into IP addresses transparently with programs other than the origonal torus peer to peer client.

PicoDNS was designed with several features in mind. PicoDNS had to be small (target compile size under 1M static), fast, and programmed using portable standards. Portability was achieved by using the GNU GLibc library for C support routines. As for program size, PicoDNS currently compiles to just over 800K statically.

2 Installation

PicoDNS can be installed either by building it from source using the supplied makefile, or by downloading and executing a statically compiled version of the program. Versions of PicoDNS are labeled by their main SVN revision. At the time this document was authored, the current version is 34.

2.1 Building from source

Getting the source

The source for PicoDNS can be gathered from the dns directory of its parent project, torus. Torus is located at www.torusp2p.com, and has a SVN server located at svn co http://torusp2p.com/svn/. To check out a copy of the most current version of the software, execute the command:

```
svn co http://torusp2p.com/svn/trunk/src/dns ./picodns
```

Once you have checked out the source code, navigate into the picodns folder and execute:

```
make
sudo make install
```

This will install a copy of the program 'picodns' under the /usr/bin folder.

Requirements

Picodns was written using GLib, the GNU c library, version 2.18. libConfuse was used for reading the configuration files. If on a debian based system (a linux system that uses aptitude), you can install the necessary libraries by executing:

```
sudo apt-get install glibc libconfuse-dev
```

2.2 Binary installation

A copy of PicoDNS is available statically linked and stripped at http://torusp2p.com/index.php?title=Picodns. That website also contains the most recent information about versions, requirements, and software updates. Once a copy of PicoDNS has been downloaded from that website, it can be installed by copying it to your /usr/bin directory.

3 Configuration

PicoDNS was designed with easy configuration in mind. Although the majority of the configuration is done via file entries, some command line configurations are also possible.

3.1 Command line

If PicoDNS is executed with the –help option, then the available command line configurations are displayed to the screen:

```
Usage:
  picodns [OPTION...] - PicoDNS server program

Help Options:
  -?, --help            Show help options

Application Options:
  -d, --daemon          Enter daemon mode
  --log-file            Set log file for this session
```

```
-c, --config-file     Set the configuration file to use for this session
--force-port          Force set the UDP port for the DNS server
```

Although the options are fairly self explanatory, they are explained in more depth below

Daemon Mode

Instructs the program to enter "daemon mode" where it disconnects from the terminal it was spawned from and runs without any messages being displayed to the user. Here is an example of starting the program in daemon mode:

```
$ picodns --daemon &
  [1] 1234
```

Using a specific log file

If the prorgam should be started using a specific file for logging errors/information, that file can be specified with the –log-file option. Note that if the log file can not be created or opened, PicoDNS will alert you and shut down.

3.2 Configuration files

The majority of the configuration for a PicoDNS server is done via the "pnds" configuration files. A demonstration copy of these files are supplied with all downloads of PicoDNS. Picodns officially supports 4 different record types - A, AAAA, PTR, and MX (as of version 34).

3.2.1 Main Configuration File

The main confguration file is where all of the PicoDNS server options are set. Below is a list of all of the main configuration options:

udp_port This is the UDP port that the DNS server will run on. If it is not set, the server will run on port 53, the default port for DNS.

loalhost_only Provided so that the server can be restricted to replying to requests from the localhost only. This is good to set to true if your server will only be used on your own computer.

main_records_file Set to a file path, this is the path to the file containing all of the records for PicoDNS. Often, this file makes use of the **include** function for each of the different record types.

3.2.2 Main Records File

This file serves as the "gateway" to all of the other record files. Although a record can be included in this file, the best practice is to set this file up like so:

```
include("a_records.pdns")
include("aaaa_records.pdns")
include("mx_records.pdns")
```

This way, each of the record types can be seperated into different, and easy to modify files.

3.2.3 A Records

A records, often thought of as IPv4 records, are records where a host name resolves to an IPv4 (format xxx.xxx.xxx.xxx IP address). A records are often stored in a file called a_records.pdns; however, A records can be stored anywhere, as long as they are included (using the **include** directive in the main configuration file). Here is an example of an A record:

Listing 1: Example A Record

```
1 a_record example_a_record {
2   class = "IN"
3   host = "myrecord2.com"
4   answer {
5     TTL = 3600
6     addr = "127.0.0.1"
7   }
8 }
```

Lets analyze this record line by line:

1: Here we see the beginning of the a_record. a_record is the type of the record, and example_a_record is the name of the record. Note that names primarily are set for debugging purposes, and is not replated to any hosts or answers.

2: Set the class of the record to IN (most records will be IN (abbreviation for Internet) typed

3: Set the host for this record. This is the domain name that will resolve into the answers provided later in this record

4: Start an answer section. There can be multiple answer sections in one record.

5: The Time To Live for this record - this value informs the client how long the IP supplied is good for

6: The address for this answer. Note that because this is an A record, this MUST be an IPv4 standard xxx.xxx.xxx.xxx formatted IP address

From this example, the basic form of an A record becomes fairly clear. There are a couple of other options you can set within an A record:

authenticated can be set to true or false. This sets if the record should be declared authenticated when replying to the server.

authoratative can be set to true or false. This sets if the record should be considered authoratative.

ignore_authentication can be set to true or false. When a DNS request is made, the client can ask for only authenticated answers to be sent. PicoDNS normally enforces this, but if this option is enabled, it will ignore the client's request for authenticated data only and reply with this record even if it is not authenticated.

auto_ptr can be set to true or false. This option is set within an answer, not the record itself. If true, a PTR record is automatically generated for that answer and is added to the LUT (Lookup Table) for the DNS server. This functionality allows for "reverse" DNS queries - where an IP is supplied and hostnames are returned.

Most of these options can be set within all other records (if they can not be, it will be mentioned in that record type's documentation);

3.2.4 AAAA Records

Similar to A records, AAAA records are the IPv6 variety. All of the options presented in the A records section are also available here; the only change is the value of **addr** in the answers (it must be in the accepted IPv6 format). Here is an example of an IPv6 AAAA record:

Listing 2: Example A Record

```
1 aaaa_record basic_record {
2   class = "IN"
3   authoratative = true
4   host = "myrecord.com"
5   answer {
6     TTL = 3600
7     addr = "2001:0db8:85a3:0000:0000:8a2e:0370:7334"
8   }
9 }
```

As you can see, the name of this record is basic_record, its type is 'IN', and it is used for the host "myrecord.com". TTL can be set as before, and this time, the address is in IPv6 standard format[1]. Also in this example we can see **authoratative** being set to true, meaning that this PicoDNS server will be considered the authoratative server for the host "myrecord.com".

3.2.5 MX Records

MX records, or Mail eXchange records are those pretaining to mail servers. Here is an example of a mail exchange record:

Listing 3: Example A Record

```
1 mx_record mymx {
2   class = "IN"
3   host = "myrecord.com"
4   answer {
5     TTL = 200
6     preference = 10
7     exchange = "mx1.myrecord.com"
8   }
9   answer {
```

```
10        TTL = 200
11        preference = 10
12        exchange = "mx2.myrecord.com"
13     }
14  }
```

The largest difference here can be seen in the answer section, where the field **addr** is no longer present, and has been replaced by the **exchange** field. The exchange field is set to the host of a mail server. Also included in mx answers is the field **preference**. Preference is used by clients to help determine what is a primary, and what is a secondary server.

3.2.6 PTR Records

PTR records are a special type of record, used for doing a "reverse" lookup. A reverse lookup is where an IP address is supplied and all known domains that resolve to that IP address are returned. As mentioned previously, automatic PTR records can be generated for A, AAAA, and MX records. This feature helps to keep the DNS records organized, allowing for the reverse lookup capability of a given record to be enabled automatically.

As of version 34, there is no way to manually create a PTR record. This feature will be coming in future revisions.

4 Usage

Once configured, PicoDNS is very easy to start and use. Simply call

```
picodns --daemon
```

to begin running PicoDNS in daemon mode. Once the program is running, it will continue serving queries until it recieves the kill signal. Sending a kill signal can be done easily by calling

```
killall picodns
```

PicoDNS will shutdown and exit.

5 Testing

In order to test your installation of PicoDNS, nslookup, DiG, or any other DNS querying tool can be used. DiG is recommended because it gives more information (and serves as a better debugging tool). The following sections will instruct you on how to test your running picodns installation.

5.1 Testing with nslookup

nslookup is not recommended for testing PicoDNS, not because it doesn't work as expected, but because nslookup does not display all of the answers recieved. To test a PicoDNS server with ns lookup, issue the command **nslookup** on any supported system (nslookup is often found on Windows and Linux). Here is an example of using nslookup for testing a PicoDNS server:

```
$ nslookup picodns_test2.com localhost

Server:         localhost
Address:        127.0.0.1#53

Name:   picodns_test2.com
Address: 127.0.0.1
```

5.2 Testing with DiG

DiG will give the most accurate test results of your PicoDNS server. To use DiG for server testing, issue the **dig** command on any supported system. Here is an example of the DiG command:

```
$ dig @localhost picodns_test2.com A

; <<>> DiG 9.5.1-P1 <<>> @localhost picodns_test2.com
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62983
```

```
;; flags: qr aa rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;picodns_test2.com.             IN      A

;; ANSWER SECTION:
picodns_test2.com.      3600    IN      A       127.0.0.1

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Feb 17 04:17:26 2009
;; MSG SIZE  rcvd: 68
```

In this test, we are specifing localhost as the DNS server and requesting the record for picodns_test2.com, type A. The reply is of type A, IP address 127.0.0.1.

Note: the domain picodns_test2.com is active in the default configuration files.

On the command line where picodns was executed, you should see the following debugging lines

```
** (picodns:13737): DEBUG: dns request from ::ffff:127.0.0.1
** Message: got a query for picodns_test2.com:A
** Message: DNS LUT has 1 answer(s) matching picodns_test2.com:A
** (picodns:13737): DEBUG: packing question 0
```

Note: the number 13737 is the process ID of the PicoDNS application - it will most likely be different on your system.

6 Credits

One of the largest code contributing groups for PicoDNS was GLib[2]. GLib was used almost exclusively for all of the internal data structures. Beej's Guide to Network Programming[3] was used as a general guide for the network components.

7 Licence

PicoDNS was authored under the GNU General Public Licence, allowing any user to download, modify or use the software. The full body of the licence can be found at:
http://www.gnu.org/licenses/gpl.html

8 Refrences

References

[1] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. RFC 2460, Internet Engineering Task Force, December 1998.

[2] Gnome Group. Glib. http://library.gnome.org/devel/glib/2.18/, 2009.

[3] Brian Hall. Beej's guide to network programing using internet sockets. http://beej.us/guide/bgnet/.

Updated 17 Feb 2009 .