DHCPKit

An IPv6 DHCP Server Framework

steffann

It started with:

- I was doing a project at Solcon (Dutch ISP)
  - Deploying IPv6 for FttH customers
  - Static /56 prefix per connection
  - Provisioning to CPE with IPv6 DHCP-PD
  - Connections identified by Remote-ID (RFC 4649)

Looking for a DHCP server

- Existing DHCPv6-PD servers good at "Dynamic"
  - But we just want static prefixes per remote-id
  - So first configure dynamic pools, and then add configuration to prevent dynamic assignments and make everything static again
- Maturity and stability not as good as we hoped
  - You don't want your DHCP server to commit suicide when encountering something it doesn't expect

When you can't find existing solutions…

- You write your own…
  - … make it available under an open source license (GPLv3)
  - … talk about it at RIPE meetings
  - because this might be useful to other people

Goals

- I wanted something flexible
- Easy to adapt to different requirements
- Easy to configure and use
- Performance wasn't a big issue (yet)

The result

- DHCPv6 server and library for Python 3.4+
- The basic server process handles initialisation, sockets, parsing and generating packets etc.
  - And nothing else…
  - It's a framework :-)
  - Incoming messages are dispatched to handlers

What it does:

- It receives DHCPv6 requests
- And lets you construct any response you want
  - Just requires a bit of code to put DHCPv6 Options in the response
  - Code for many common options already included

What handlers?!

- Currently two message handlers:
  - The standard handler implements the basics of RFC 3315
  - Another one just dumps incoming messages on stdout
  - And if you want something different it's very easy to implement

THE STANDARD MESSAGE HANDLER

- Wraps the incoming message in a transaction bundle which contains the incoming message and the response-under-construction
- Calls a bunch of option handlers which do the real work
- Returns response to server so it can send it

Those option handlers…

- This is where all the real work happens
  - ServerIdOptionHandler: check if client asks for specific server, and stop when that server isn't us
  - ClientIdOptionHandler: copy client's DUID to response
  - UnansweredIAOptionHandler: answer IA_NA and IA_TA if no other handler provided addresses
  - RecursiveNameServersOptionHandler: provide DNS info
  - Etc…

And the most interesting option handlers…

- Implementing IA_NA, IA_TA and IA_PD
- Currently one simple implementation
  - Hand out static IA_NA and IA_PD based on DUID, interface-id or remote-id
  - Data in a CSV file, DBM file or SQLite database
- But other implementation are easy to write
  - Already got asked about integrating with Apache CloudStack

And that is enough for a small ISP deployment

- This is running for a pilot project at Solcon
- Provisioning for pilot:
  - Export a new CSV file from the customer database
  - kill -HUP <pid>
- Production provisioning:
  - Export a new CSV file from the customer database
  - ipv6-dhcp-build-sqlite <csv-file> <sqlite-file>

Some configuration details

Main configuration

(simplified a bit, but you get the idea)

```
[interface eth1]
global-unicast-addresses = all
multicast = no

[option preference]
preference = 255

[option CSV-Based-fixed-assignment 2001:9e0:xyz::/64]
assignments-file = assignments.csv

[option RecursiveNameServers]
server-address-1 = 2001:9e0:4:32::3
server-address-2 = 2001:9e0:1:101::3

[option NTPServers]
server-address-1 = 2001:9e0:4:32::250
```

ASSIGNMENTS.CSV

```
id,address,prefix
remote-id-str:3561:CUST1,,2001:9e0:xyz:0100::/56
remote-id-str:3561:CUST2,,2001:9e0:xyz:0200::/56
remote-id-str:3561:CUST3,,2001:9e0:xyz:0300::/56
remote-id-str:3561:CUST4,,2001:9e0:xyz:0400::/56
remote-id-str:3561:CUST5,,2001:9e0:xyz:0500::/56
remote-id-str:3561:CUST6,,2001:9e0:xyz:0600::/56
```

- No IA_NA, only IA_PD
  - The WAN link is unnumbered from the user's point of view in this example

After we started using it…

KEEPING AN EYE ON THINGS

- Reading log files is boring…
- Write a new looking glass handler in a few hours!
  - At pre-handling capture the incoming request
  - At post-handling capture the outgoing response
  - Spawn a separate process that stores the last request and response per customer in a database
  - Add a simple (e.g. Django) view on that database for the NOC

Roadmap

steffann

Intention

- Keep developing this, adding more handlers and tools to the base set
- Hope that this is useful to others as well
  - ISPs, CPE testing, etc…
- Asked for some funding from SIDN Fund

Roadmap

- Performance improvements
- More documentation (https://dhcpkit.readthedocs.org)
- More unit tests (https://travis-ci.org/sjm-steffann/dhcpkit)
- Better test coverage (https://coveralls.io/github/sjm-steffann/dhcpkit)
- Implement more DHCP options and handlers
  - e.g. get assignments from RADIUS

Links

- Code: https://github.com/sjm-steffann/dhcpkit
- Documentation: http://dhcpkit.readthedocs.org
- Releases: https://pypi.python.org/pypi/dhcpkit
  - "pip install dhcpkit"

Basic option and HANDLER STRUCTURE

Basic option implementation

```
class Option(ProtocolElement):
    def __init__(self, ...):
        ...

    def load_from(self, buffer, offset, length):
        self.xyz = ...

    def save(self):
        return bytes(...)

    def validate(self):
        ...
```

An option class is just a representation of bits on the wire, in an easy to use format

How does an option handler work?

```
class OptionHandler:
    @classmethod
    def from_config(cls, se                    ne):
        ...

    def pre(self, bundle:                      dler
        ...                                    the

    def handle(self, bund                      st,
        ...

    def post(self, bundle: Tr
        ...
```

This is where the work is done. Add options to response, determine which

Clean up before sending response. Commit offered addresses to storage etc.

decide if we need to handle it

What does that transaction bundle do?

- Contain request and response
- Handle relay message wrappers
  - Split on incoming message, re-wrap on outgoing message
- Keep track of handling status
  - Which IA_NA/IA_TA/IA_PD options have been handled?
  - Are all handlers willing to participate in rapid commit