DHCPKit

Reaching version 1.0

steffann

What is DHCPKit?

- An IPv6 DHCP Library and Server Framework
  - Customisable to fit any situation
  - Written in Python 3.4+
  - GPLv3 license

Why did I start this project?

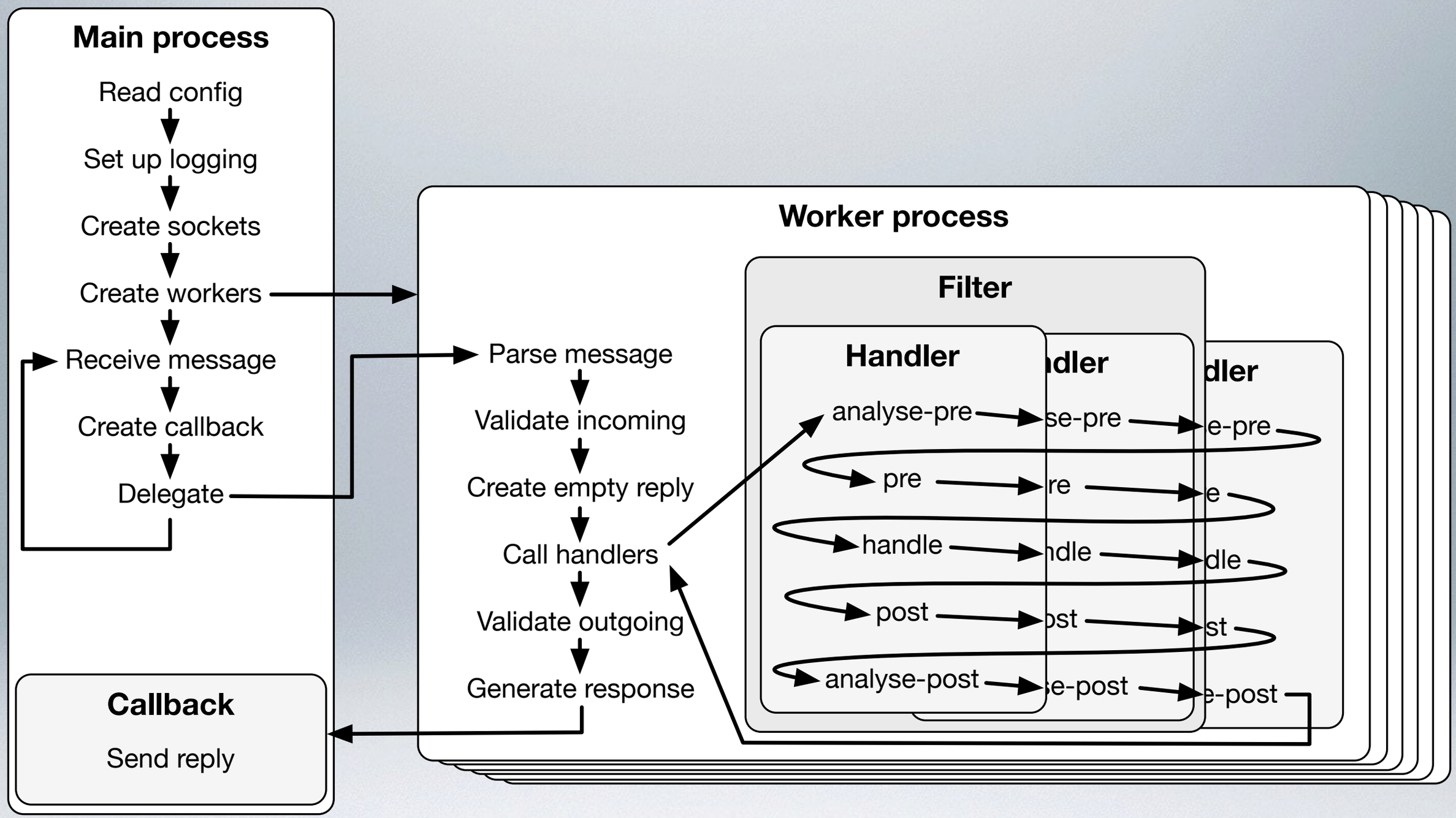
- Doing IPv6 FttH project for Solcon (Dutch ISP)
- Existing DHCPv6 servers:
  - Good at "dynamic", but wanted static prefixes per remote-id
  - Good "standard" implementations, but not much flexibility for less common scenarios
- So I wrote a new DHCPv6 server:
  - Not much pre-defined behaviour, all about flexibility

SIDN fonds

- Received funding from SIDN Fund in 2016
  - Provided both financial and strategic support
  - This made a lot of improvements possible!
  - Open source work is extra nice if you can also eat :)

Basic structure

steffann


Main process
Read config
Set up logging
Create sockets
Create workers
Receive message
Create callback
Delegate
Callback
Send reply
Worker process
Parse message
Validate incoming
Create empty reply
Call handlers
Validate outgoing
Generate response
Filter
Handler
analyse-pre
pre
handle
post
analyse-post

Structure

Configuration

- Apache-style (using ZConfig)

```
<subnet 2a00:8640:1::/64>
  <static-csv static.csv />

  <recursive-name-servers>
    address 2001:4860:4860::8844
    address 2001:4860:4860::8888
  </recursive-name-servers>
</subnet>

<sol-max-rt>
  limit 43200
</sol-max-rt>
```

Filter

Handler

Handler

Handler

Included filters and handlers

(lot of slides, just giving a quick overview)

steffann

What is already available?

- Basic / mandatory functionality
  - Copy Client-ID option from request to response
  - Check Server-ID option if provided
  - Copy Interface-ID option in relay messages to response
  - Add preference option to response
  - Reject unicast requests to server unless advertised
  - Rapid-commit handler (promotes advertise response to reply)
  - Set correct status-code option in response

What is already available?
Filters

- Subnet
  - Based on link-address of interface that request came in from
- Elapsed-time
  - Based on the elapsed-time option provided by the client
  - Adapt behaviour based on how long the client has been trying
  - For example in a failover cluster: let one server ignore clients that haven't been trying for very long, reducing noise

What is already available?
Informational options

- DNS
  - Recursive name servers option
  - Domain search list option
- SIP
  - SIP Server addresses option
  - SIP Server domain names option

What is already available?
Informational options

- NTP
  - NTP servers option
    - FQDN, unicast address and multicast address sub-options
- SNTP
  - SNTP server address option

What is already available?
Informational options

- DS-Lite
  - AFTR name option
- MAP
  - MAP-E and MAP-T options
  - Map-rule sub-options

What is already available?
Relay provided options

- Remote-ID / Subscriber-ID / Link-layer ID
  - Usable in address/prefix assignment handlers
  - Optionally copy from incoming to outgoing relay messages

What is already available?
Client Behaviour related

- Client timing
  - SOL-MAX-RT and INF-MAX-RT options
- Rate-limiting
  - Some clients keep sending requests if they don't like the answer
  - So stop answering them at all and they'll slow down
  - Identify clients by DUID / Remote-ID / etc.

What is already available?
Server Behaviour related

- Ignore
  - Configure the server to ignore certain requests
  - Very useful in combination with the elapsed-time filter to build a simple DHCPv6 server cluster

```
<elapsed-time>
    less-than 30s

    <ignore-request>
        message-type solicit
    </ignore-request>
</elapsed-time>
```

What is already available? Address and Prefix assignment

- Static assignments
  - CSV and SQLite based mapping from DUID, Interface-ID, Remote-ID, Subscriber-ID or LinkLayer-ID to address and prefix
  - SQLite doesn't need to reload server when data changes
  - Converter to convert CSV to SQLite
- Timing options
  - Automatically adjust T1 and T2 timers based on assignments

Writing your own extensions

steffann

Extension layout
Example

```
dhcpkit_xyz/
    __init__.py
    options.py
    setup.py

server_extension/
    __init__.py
    component.xml
    config.py
```

Option Implementation

Handler Implementation

Configuration definition

Python objects representing config

Extension projects

- Using setup.py entry-points for:
  - DUID types
  - Message types
  - Option types
  - Server extensions (configuration sections, filters, handlers etc)

Tell DHCPKit where to find it

```
setup(
    name="dhcpkit_xyz",
    ...
    entry_points={
        "dhcpkit.ipv6.options": [
            "999 = dhcpkit_xyz.options:XYZOption",
        ],
        "dhcpkit.ipv6.server.extensions": [
            "xyz = dhcpkit_xyz.server_extension",
        ],
    }
)
```

Link option type number to implementation

Which directory contains component.xml?

Configuration section definition in ZConfig XML

```
<component xmlns="http://.../schema.dtd"
           prefix="dhcpkit_xyz.server_extension.config">
    <sectiontype name="xyz"
                 implements="handler_factory"
                 datatype=".XYZOptionHandlerFactory">

        <description><![CDATA[
            This sections adds xyz to the response.
        ]]></description>

        <example><![CDATA[
            <xyz>
                address 2001:db8::1
            </xyz>
        ]]></example>
```

Configuration section definition in ZConfig XML

```
<multikey name="address"
          attribute="addresses"
          required="yes"
          datatype="ipaddress.IPv6Address">
    <description><![CDATA[
        The IPv6 address of a XYZ.
    ]]></description>

    <example><![CDATA[
        2001:db8:1::2
    ]]></example>
</multikey>
</sectiontype>
</component>
```

Extension code execution

- Configuration is read
  - Server process is still running as root
  - Instantiate factory object representing configuration
- Factory object creates handler in main process
  - By now server process has dropped privileges
- Handler object is copied to worker processes
  - Run second-stage initialisation in worker for things that can't be shared between workers, like database connections

Existing extension projects

Extension project
Technicolor options

- Technicolor CPEs implement SOL-MAX-RT
- But with a non-standard option number...

- Distributed as an extension project
  - Keep non-standard stuff out of the main code
  - Nice small and simple example of how to write extensions :)

Extension project
Kafka integration

- Analyse-pre
  - Store incoming request
- Analyse-post
  - Store outgoing response (if any)
  - Post both to Kafka for further analysis

Extension project
Looking glass

- Read messages from Kafka
- Store last x interactions per client in a database
- Django admin based looking glass interface for customer support desk and debugging

Current status

Current development

- Working on Lease Query and Bulk Lease Query
  - New message types and options
  - Different way of interaction: relay to server
  - Bulk Lease Query uses TCP, needs some changes to framework

After that

- When Lease Query implementation is done…
  - It will be time to release version 1.0!

PARTICIPATE!

- Everybody who is interested in doing interesting things with DHCPv6 is welcome!
  - ISPs
  - Enterprises
  - Students studying IPv6 and DHCPv6
  - Etc
- Contact me at sander@steffann.nl