LEDE Firmware optimization for wired deployments using BGP (Bird Daemon) and BMX for routing by enhancing and extending Bird Daemon's configuration and UI integration

Theale, June 2017

Major: Computer Science
Minor: Open Source Software

Author:
Eloi Carbó Solé

Director:
Joan Manuel Marquès Puig
Computer Science, Multimedia and Telecomunications Department (DPCS-ICSO)

External consultant:
Víctor Oncins Biosca
Routek SL

Universitat Oberta de Catalunya
2017

Index

List of Figures

List of Tables

Chapter 1

Introduction

[1]

1.1 Structure of the document

1.2 Description of the problem and motivation

1.2.1 Bird Daemon administration issues

1.3 Scope of the project

1.3.1 Deviations from the original plan

1.3.2 Methodology and communication

1.4 Background concepts

1.4.1 Community Networks

1.4.2 OpenWRT/LEDE Project

1.4.3 Dynamic Routing Protocols

1.4.4 Static Routing Protocols

BMX6

BGP

OSPF

1.5 State of the art

1.6 Planning

1.6.1 Tasks

Project's background

Project's validation

Document and support

1.6.2 Deviations and modifications in the planning

1.7 Budget

Chapter 2

Network Architecture

2.1 Guifi.net target environment

2.2 Bird Daemon environment translation

Chapter 3

Package improvements implementation

3.1 Administration requirements

3.2 Changes in the code

3.2.1 Apply code standards

3.2.2 init.d script and service management

3.2.3 UCI Configuration improvements

3.2.4 LUCI UI improvements

Status Page

Log Page

Filters & Functions Page

BGP and Classic protocols Pages

3.2.5 Align documentation and upgrade to Markdown

3.3 Package Testing

Testing an integration/translation Package, and this one specifically, is a rather complex task to evaluate as Bird configuration files are modular and desired settings can be achieved in different ways. Even more, although a *it works/it does not work* policy could be accepted, it does not mean that there are not other possible implementations that could work in a better way. For example, filters and functions can be either written in the *.conf* file or included using **%include** mechanism, being the second one a better approach as it enhances code readability as well as it avoids bloating the configuration file unnecessarily.

With this introduction in mind, the following sections will explain how this package has been tested following Bird's configuration base requirements and service behaviour and some *future work* ideas to achieve automatic and unit tests.

3.3.1 Configuration Composition Tests (future work)

To perform configuration integrity tests in current package, it is required to repeat the execution of */etc/bird{4|6}/init.d/bird4 restart* in order to trigger the UCI-bird.conf translation from a target UCI file. The code to do this translation has been refactored in an functional manner to allow future unit tests or, at least, make it easier to integrate in an automated test framework or process. For example, an automated CI/CD build process could build an update of the package, push it into a test node, execute the translation process and compare it against the previous (or a stable) version as well as check its correctness by querying Bird's status.

Reviewing v0.2 against v0.3

Testing the outputs from the old and new packages, and taking into account that there are some manual changes in the old one, the following example is configured as follows:

- Router IDs follow node's IP Address
- Kernel, Device and Static Protocols have been set by default
- A Static Route has been added (identical)
- BGP Template and Instance have been configured following v0.2 scheme with matching settings to avoid Bird failures
- BGP Instance AS and Neighbours are dummy values
- A BGP Filter called "all_ok" (accept all routes) has been added using each version's process.

In the new package, we have instantaneous configuration correctness feedback as we can check Bird's status in the Status Page. In the old package, after executing */etc/bird{4|6}/init.d/bird4 start*, Bird will fail and it is required to move the Filter "all_ok" to the top of the document. Bird will start correctly after this modification.

After checking that both daemons are running, we can then perform a *diff* between the configuration files and look for any noticeable difference

Listing 3.1: diff of bird4.conf in v0.2 (<) and v0.3 (>)

```
3,9d2
<  #Filter filter1:
<  filter all_ok
<  {
<      accept "all ok";
<      }
<
13c6
< router id 192.168.1.200;
---
> router id 192.168.1.100;
17a11,17
> #Functions Section:
> #End of Functions --
>
> #Filters Section:
> include "/etc/bird4/filters/filter1";
> #End of Filters --
>
19c19
< protocol kernel {
---
> protocol kernel kernel1 {
46c45
<     source address 192.168.1.200;
---
>     source address 192.168.1.100;
57c57
<     neighbor 192.168.1.201 as 1002;
---
>     neighbor 192.168.1.101 as 1002;
```

As shown in this *diff* snippet, almost all the translated configuration is identical apart from:

- Different Router IDs and BGP neighbours (expected)

- Kernel Protocol definition (minor change in the API)
- BGP Filter definition (major change in the API)

3.3.2 Bird Daemon Errors

Bird Daemon provides an error exit code together with different text outputs in order to highlight errors in the configuration. Although most of the times it can be easily spotted using Bird's feedback, there are also instances where the Daemon's documentation may be required to fix them.

Bird Daemon Error output examples

Most common errors that an administrator could face are:

- A configured field has incorrect syntax. Bird will give you hints about what is wrong most of the times: wrong IP address format *bird: /tmp/bird4.conf, line 7: Invalid IPv4 address 1921.68.1.1.* But some *rare* times the message is less helpful and you may need to check the contents of the file and understand the error.

  As an example of this: *bird4: Failed - bird: /tmp/bird4.conf, line 65: syntax error.* We need to check the bird4.conf file and see that in line 65:

```
64:    protocol bgp BGPExample {
65:        import Filter NonExistingFilter;
66:    }
```

Listing 3.2: Bird4.conf contents

  We will need to find out that the shown filter used in the **import** field of BGP Protocol, does not exist.

- Non-compatible configuration. The other set of common errors is non-compatible fields in a Protocol.

  As an example of this: *bird: /tmp/bird4.conf, line 76: Only internal neighbor can be RR client.* We need to remove the Route Reflector Client setting from the BGP Instance to fix this behaviour.

- Missing filter or function If you include a filter name in any of the Protocols or if any of your filters use a non-existing function, Bird will fail to start showing an error as follows: *bird: /tmp/bird4.conf, line 71: No such filter.*

- Syntax errors in a filter or function. This error follows the same approach as the first bullet: *bird: /etc/bird4/filters/filter-20170507-0951, line 4: syntax error.* You are required to go to command line and fix the problem checking the configuration and filter or function files.

- Filter calling to non-existing functions. If your filter executes a command that is not defined by Bird's syntax, it will handle it as a function. If that function does not exist in any of the handled files, it will show this error: *bird: /tmp/bird4.conf, You can't call something which is not a function. Really.*

- Filters not accepting/rejecting routes. Bird Daemon filters must return an *accept* or *reject* policy per route received. If any of your filters does not return any policy per route, it will be silently ignored and substituted with an "accept".

  As an example of this issue:

```
filter doNothing
{
    print "HelloWorld";
}
```

Listing 3.3: Filter printing message

Bird Daemon will succeed starting up but, if we check the log information in the Log Page, this error message will be shown:

```
<ERR> Filter doNothing did not return accept nor reject.
   Make up your mind
<INFO> HelloWorld
}
```

Listing 3.4: Filter printing message

3.3.3 Simple BGP Instance Tests

3.3.4 Full Network Virtual Environment

Bibliography

[1] "NITOS Wireless Testbed - Network Implementation Testbed Laboratory." http://nitlab.inf.uth.gr/NITlab/index.php/testbed.