
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

1	Introduction	1
1.1	Structure of the document	2
1.2	Description of the problem and motivation	2
1.2.1	Bird Daemon administration issues	2
1.3	Scope of the project	2
1.3.1	Deviations from the original plan	2
1.3.2	Methodology and communication	2
1.4	Background concepts	2
1.4.1	Community Networks	2
1.4.2	OpenWRT/LEDE Project	2
1.4.3	Dynamic Routing Protocols	2
1.4.4	Static Routing Protocols	2
1.5	State of the art	2
1.6	Planning	2
1.6.1	Tasks	2
1.6.2	Deviations and modifications in the planning	2
1.7	Budget	2
2	Network Architecture	3
2.1	Guifi.net target environment	3
2.2	Bird Daemon environment translation	3
3	Package improvements implementation	4
3.1	Administration requirements	4
3.2	Changes in the code	4
3.2.1	Apply code standards	4
3.2.2	init.d script and service management	4
3.2.3	UCI Configuration improvements	4
3.2.4	LUCI UI improvements	4
3.2.5	Align documentation and upgrade to Markdown	4
3.3	Package Testing	5
3.3.1	Configuration Translation Tests (future work)	5
3.3.2	Bird Daemon Errors	7

3.3.3	Real Scenario: VM with simple BGP configuration connected to Guifi.net	8
3.3.4	Full Network Virtual Environment	10

Appendices

Appendix	12
-----------------	-----------

A Bird Daemon's Configuration using v0.3 Package - UOC's VM in Guifi.net	12
A.1 UCI Configuration	12
A.2 Bird Configuration	13

Bibliography	17
---------------------	-----------

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 Translation 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

```

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;

```

Listing 3.1: Battlemesh experiment code

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 examples

Most common errors that an administrator may need to resolve 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 Real Scenario: VM with simple BGP configuration connected to Guifi.net

As part of the acceptance tests, a VM was set up by a sysadmin in the *Universitat Oberta de Catalunya* to act as a pre-production machine. This VM is connected to a *Mikrotik* Router acting as Gateway to *Guifi.net* but this scenario does **not** connect or communicate through any Mesh Network using BMX6, so it is an end point.

The configuration of this system is almost identical, component-wise, to the ones available in Guifi.net. However, this system will only route itself (1 route) and import any.

Bird UCI configuration set through the WEB UI and its translation into Bird4 configuration can be reviewed in appendix A.

This VM is communicating to Guifi.net through a Mikrotik which is already doing some filtering but, in any case, it is still able to import 3000+ Routes and export itself:

```
root@LEDE-eloi:~# birdc14 show protocols all
[...]
BGPIImportALL BGP      master    up      2017-05-10  Established
  Preference:      100
  Input filter:     ebgp_in
```

```

Output filter:  ebgp_out
Import limit:   3000 [HIT]
  Action:      warn
Routes:        2999 imported, 1 exported, 2999 preferred
Route change stats:  received  rejected  filtered
   ignored  accepted
Import updates:      1208383          0          0
      88    1208295
Import withdraws:    337268          0          ---
      300    336968
Export updates:      1208298    1208295          2
      ---          1
Export withdraws:    336968          ---          ---
      ---          0
BGP state:          Established
Neighbor address:   172.25.35.25
Neighbor AS:        59361
Neighbor ID:        10.90.224.65
Neighbor caps:      refresh AS4
Session:            external AS4
Source address:     172.25.35.26
Route limit:        2999/3000
Hold timer:         160/180
Keepalive timer:    29/60

```

Listing 3.5: UCI Configuration

Using Bird Lightweight Remote Control (**birdcl4**) we can verify Bird's BGP instance. As key information:

- BGP Instance: BGPImportALL
- Filters applied: *ebgp_in* and *bgp_out*
- We are connected to our neighbour 10.90.224.65 with Autonomous System ID 59361
- The number of routes received fluctuates but the data shown presents 2999 routes imported.
- We do not know when, but the import Limit reached (HIT) and that generated warnings. From our Package's Log Page: 2017-05-21 22:09:13
<WARN> Protocol BGPImportALL hits route import limit (3000),
action: warn
- We are exporting 1 Route.

As a health check, we can query Bird of its last reconfiguration, reboot time or status using **birdcl4 status**:

```

root@LEDE-eloi:~# birdcl4 show status
BIRD 1.6.3 ready.

```

```
BIRD 1.6.3
Router ID is 10.139.173.161
Current server time is 2017-05-22 00:20:23
Last reboot on 2017-05-10 19:31:09
Last reconfiguration on 2017-05-10 19:31:09
Daemon is up and running
```

Listing 3.6: UCI Configuration

3.3.4 Full Network Virtual Environment

Appendices

Appendix A

Bird Daemon's Configuration using v0.3 Package - UOC's VM in Guifi.net

A.1 UCI Configuration

```
config bird 'bird'
    option use_UCI_config '1'
    option UCI_config_file '/tmp/bird4.conf'
    option UCI_config_File '/tmp/bird4.conf'

config global 'global'
    option log_file '/tmp/bird4.log'
    option router_id '10.139.173.161'
    option log 'all'

config table
    option name 'aux'

config kernel 'kernel1'
    option import 'all'
    option export 'all'
    option scan_time '10'
    option learn '1'
    option disabled '0'

config device 'device1'
    option scan_time '10'
    option disabled '0'

config bgp_template 'BGP_COMMON'
    option receive_limit_action 'warn'
    option local_as '92099'
    option igp_table 'bgpTable'
    option export_limit_action 'warn'
```

```

        option import_limit_action 'warn'
        option next_hop_self '0'
        option next_hop_keep '0'
        option rr_client '0'

config table
    option name 'bgpTable'

config bgp 'BGPIImportALL'
    option receive_limit_action 'warn'
    option template 'BGP_COMMON'
    option neighbor_as '59361'
    option neighbor_address '172.25.35.25'
    option export_limit_action 'warn'
    option import_limit_action 'warn'
    option import_limit '3000'
    option import 'filter ebgp_in'
    option export 'filter ebgp_out'
    option next_hop_self '0'

config kernel 'Kernel_BGP'
    option disabled '0'
    option table 'bgpTable'
    option kernel_table '251'
    option scan_time '10'
    option learn '1'
    option import 'all'
    option export 'all'

config pipe 'pipe1'
    option disabled '0'
    option peer_table 'bgpTable'
    option table 'aux'
    option import 'all'
    option export 'all'
    option mode 'transparent'

config direct 'direct1'
    option disabled '0'
    option interface '"br-lan","br-wan", "br-mgmt"'

config static 'static1'
    option disabled '0'
    option table 'aux'

```

Listing A.1: UCI Configuration

A.2 Bird Configuration

```

#Bird4 configuration using UCI:

log "/tmp/bird4.log" all;

```

APPENDIX A. BIRD DAEMON'S CONFIGURATION USING V0.3 PACKAGE - UOC'S VM IN GU

```
#Router ID
router id 10.139.173.161;

#Secondary tables
table aux;
table bgpTable;

#Functions Section:
include "/etc/bird4/functions/function-20170507-1038";
#End of Functions --

#Filters Section:
include "/etc/bird4/filters/filter-20170507-0951";
#End of Filters --

#kernel1 configuration:
protocol kernel kernel1 {
#   disabled;
    learn;
    persist;
    scan time 10;
    import all;
    export all;
}

#Kernel_BGP configuration:
protocol kernel Kernel_BGP {
#   disabled;
    table bgpTable;
    kernel table 251;
    learn;
    persist;
    scan time 10;
    import all;
    export all;
}

#static1 configuration:
protocol static {
    table aux;
}

#device1 configuration:
protocol device {
#   disabled;
    scan time 10;
}

#direct1 configuration:
protocol direct {
#   disabled;
    interface "br-lan","br-wan", "br-mgmt";
}
```

APPENDIX A. BIRD DAEMON'S CONFIGURATION USING V0.3 PACKAGE - UOC'S VM IN GU

```
#pipe1 configuration:
protocol pipe pipe1 {
#   disabled;
    table aux;
    peer table bgpTable;
    mode transparent;
    import all;
    export all;
}

#BGP_COMMON template:
template bgp BGP_COMMON {
    local as 92099;
#   next hop self;
#   next hop keep;
    igp table bgpTable;
#   rr client;
}

#BGPImportALL configuration:
protocol bgp BGPImportALL from BGP_COMMON {
    import filter ebgp_in;
    export filter ebgp_out;
#   rr client;
    import limit 3000 action warn;
    neighbor 172.25.35.25 as 59361;
}

=====
BGP Filters and Functions:

root@LEDE-eloi:~# cat /etc/bird4/filters/filter-20170507-0951
filter ebgp_in {

    krt_prefsrc = 10.139.173.161;

    if match_guifi_prefix() then accept;
    reject;
}

filter ebgp_out {

    if match_guifi_prefix() then accept;
    reject;
}

root@LEDE-eloi:~# cat
/etc/bird4/functions/function-20170507-1038
function match_guifi_prefix()
{
    return net ~ [ 10.0.0.0/8{9,32} ];
}
```

Listing A.2: Bird4.conf Configuration

Bibliography

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