Documentation - Fibbing

Vanden Bulcke Cedric & Lejoly Florent

June 7, 2017

Contents

1 Installation **2**
1.1 Setup . . . 2
1.2 Main command . . . 2
1.3 Vagrant basic command . . . 3

2 ConfD Agent **4**

3 Network Manager CLI **4**
3.1 Commands . . . 4
3.2 ConfD Configuration CLI . . . 7
3.2.1 Creating the network . . . 7
3.2.2 Enabling OSPF on network link . . . 8
3.2.3 Destinations . . . 8
3.2.4 The controllers . . . 8
3.2.5 Fibbing requirements . . . 9

4 Getting Started **10**
4.1 The network with fibbing . . . 11
4.1.1 Create the routers . . . 11
4.1.2 Create the links . . . 11
4.1.3 Adding the hosts . . . 11
4.1.4 The controllers . . . 12

5 Autotopo framework **12**
5.1 Getting Started with Autotopofib . . . 12

6 Simulation **12**

7 Troubleshooting **13**
7.1 Clearing all stored configurations . . . 13
7.2 `Aborted: illegal reference` when configuring a requirement 14

1 Installation

1.1 Setup

First and foremost, you will need to have *Vagrant*[1] and *Virtualbox*[2] installed in your local machine. You can find link toward the respective website in footnote. Second will we need to download the Fibbing VM (https://github.com/Fibbing/virtual-machine). Finally, we need to replace the Vargantfile and provision script of the Fibbing VM by the provide files in *fibbing-vm*.

It is important that the Fibbing VM directory is located in the **Network-ManagerCLI**. Note that we will refer to directory where the Fibbing VM is installed as *fibbing-vm* for now on.
Then you can enter the `fibbing-vm` directory and run:

```
./install.sh
```

This will boot and provision the Virtual Machine. You can then enter the VM with:

```
vagrant ssh
```

At this point you should have a terminal open inside the VM. You can go into the shared directory `/Thesis/NetworkManager` with:

```
cd /Thesis/NetworkManager
```

Then you can run these two commands to setup the ConfdAgent and SNMP:

```
make install
make snmp-config
```

You might need to exit, halt and reboot the VM at this point.

1.2 Main command

A *Makefile* is provided in `/Thesis/NetworkManager` that gathers all the main commands.

When the VM has started make sure the `confdagent` service is running:

```
service confdagent start
```

Start the Network Manager CLI with:

```
make manager-CLI
```

or just run:

```
python main/main.py -D
```

to start the Network Manager CLI in debug mode.

[1] https://www.vagrantup.com/
[2] https://www.virtualbox.org

1.3 Vagrant basic command

Here are listed Vagrant basic commands. All those command must be run inside the `fibbing-vm` directory.

- Start the VM:

```
vagrant up
```

- Enter the VM:

```
vagrant ssh
```

- Halt the VM:

```
vagrant halt
```

  Once the VM is halted you can run `vagrant up` to restart the VM some times later.

- Destroy the VM:

```
vagrant destroy
```

  Destroying the VM will remove all stored states corresponding to this VM.

2 ConfD Agent

From the working directory (*/Thesis/NetworkManager/*), the ConfD Agent can be recompiled with:

```
root@jessie:/Thesis/NetworkManager# make compile
```

This could be required whenever either the YANG model or the source code of the ConfD Agent is changed.

For debugging purposes, you might need to see the output of the ConfD Agent. To do so you can run:

```
root@jessie:/Thesis/NetworkManager# make start-daemon
```

This requires to have first compiled the ConfD Agent, and need to run on a separate terminal window. Another way to start the ConfD Agent without running on a terminal window, is run:

```
root@jessie:/Thesis/NetworkManager# service confdagent start
```

This requires to have compiled the ConfD Agent as well. You can see the state of the ConfD Agent running via the service with:

```
root@jessie:/Thesis/NetworkManager# service confdagent status
```

And you can stop the ConfD Agent running via the service with:

```
root@jessie:/Thesis/NetworkManager# service confdagent stop
```

You can enter the ConfD CLI from the working directory with:

```
root@jessie:/Thesis/NetworkManager# make cli
```

Finally, you stop the ConfD Agent started with the command `make start-daemon` with:

```
root@jessie:/Thesis/NetworkManager# make stop-daemon
```

3 Network Manager CLI

3.1 Commands

apply_new_requirement — When new requirements are configured and committed, run this command for the application to process the new requirements. Obviously requires for the application to have received the configured requirements from the ConfD agent, and to have a running controller.

config — Start the configuration CLI supported by ConfD. Requires that the ConfD agent is running (see section 1.1). For more information about the configuration CLI, see section 3.2.

enable_auto_schedule	When the application has successfully received and processed *scheduled* requirements, run this command for the application to start to periodically (see `set_time_granularity`) checking the conditions of the scheduled requirements.
halt_auto_schedule	Run this command to stop the application from periodically checking the conditions of the scheduled requirements.
halt_requirement	Run this command to remove a running requirements already pushed by the Fibbing controller.
help	Run this command to display all existing commands. Run `help [COMMAND]` to display some information about the usage of `[COMMAND]`.
info	Displays some information about the application.
mininet_cli	Run this command to enter the CLI Mininet.
quit	Run this command to quit the application. Requires that both the Fibbing controller and the network be stopped.
run_auto	When running this command the application will automatically start (or restart) the network when it received new network configuration from the ConfD agent. It will also start the Fibbing controller when new requirements are received from the ConfD agent.
set_time_granularity	Run `set_time_granularity [TIME]` to set the time period at `[TIME]` (in seconds). This is the period at which the application checks the scheduled requirement conditions.
show_requirements	Displays the requirements stored by the application. The application only stores the requirements configure with state *running* or *scheduled*.
show_status	Displays some state information, such as the state of the network and controller (*running*/*not-running*), the state of the configurations (if the application has received new configurations from the ConfD agent).
show_time_granularity	Display the current value of the time period. Change this value with `set_time_granularity [TIME]`.
start_controller	Run this command to start the Fibbing controller. Requires that the network is running, and that the controller have some initial requirements.

start_network	Run this command to start the network. Requires for the application to have received the configuration of the network.
stop_network	Run this command to stop the network. Requires the Fibbing controller to be stopped before.
stop_controller	Run this command to stop the Fibbing controller.
xterm_config	Run this command to pop a Xterm terminal window running the ConfD configuration CLI.
terminal	Run this command to start a bash terminal. This is mostly useful for debugging.

3.2 ConfD Configuration CLI

This section presents the ConfD Configuration CLI.

3.2.1 Creating the network

In order to configure the network, we will need to start the ConfD CLI. This can be done using the command *config* in the Network Manager CLI or *make cli* in the `NetworkManager/` directory. Once the ConfD configuration CLI is started, use *config* to enter the network configuration mode.

In order to create a network, one must first declare the routers present.

3.2.1.1 create a new router

To create a new router, use the following command in config mode:

```
jessie(config)# fibbing network router <router-name> addr>
```

where <router-name> is the name you want to give to a router.

3.2.1.2 create a new link

To create a new link, we first create a link by giving it a unique name. Then the user has to specify the source and destination of his link using previously declared routers and by assigning them an ip. The user can then change the bandwidth and cost of the link (otherwise the default values will be used).

```
jessie(config)# fibbing network link <link-name>
jessie(config-link-linkA-B)# src name <RouterName> ip <IPv4>
jessie(config-link-linkA-B)# dest name <RouterName> ip <IPv4>
jessie(config-link-linkA-B)# cost <Integer cost>
jessie(config-link-linkA-B)# bw <bandwidth in Mbps>
```

where $< link - name >$ is the name you want to give to the link. It should be noted that the IPv4 address field ($< IPv4 >$) needs to have the network mask specified as well and that the bandwidth has a default value of 100Mbps. The `bidirectional` attribute has a Boolean value indicating whether the link is bidirectional (i.e. `bidirectional=true`) or not. In the case where the link is bidirectional only the `cost` attribute of the link is used to indicate that the link in both direction from `src` to `dest` and from `dest` to `src` have the same `cost`. If the `bidirectional` attribute is set to `false`, the link is considered asymmetric. The cost of the link in the direction from `src` to `dest` can be specified with the `src cost` attribute. Similarly the cost in the direction from `dest` to `src` can be specified with the `dest cost` attribute. For example,

```
jessie(config-link-linkA-B)# bidirectional false
jessie(config-link-linkA-B)# src cost 10
jessie(config-link-linkA-B)# dest cost 100
```

would mean that for the given link (e.g. src **A** and dest **B**) the cost from **A** to **B** is 10 while the cost from **B** to **A** is 100.

3.2.2 Enabling OSPF on network link

In order to enable OSPF on the network links, you should use:

```
jessie(config)# fibbing network link ospf-link-config <link-name>
```

where $< link-name >$ referenced the name of link configured in section 3.2.1.2. From there, you can modified the configuration specific for OSPF (it should be noted that there are default values for those field, so just referencing the link will enable OSPF with those default values).

```
jessie(config)# fibbing network ospf-link-config  <link-name> src
Possible completions:
  area  dead-interval  hello-interval  lsa  throttle
jessie(config)# fibbing network ospf-link-config  <link-name> dest
Possible completions:
  area  dead-interval  hello-interval  lsa  throttle
```

3.2.3 Destinations

Destinations for the Fibbing solutions are modeled as hosts. Adding a host is almost similar to creating the link between two routers, the user just needs to create the link between the host and the previously declared router.

```
jessie(config)# fibbing network hosts-link <link-name>
jessie(config-hosts-link-Host1-A)# router name <RouterName> ip <IPv4>
jessie(config-hosts-link-Host1-A)# host name <RouterName> ip <IPv4>
jessie(config-hosts-link-Host1-A)# bw 100
```

Once the hosts are added, the last thing to do for the Fibbing solution is to add the controllers to the network.

3.2.4 The controllers

There are 2 types of controllers: One unique Main Controller and Fibbing Controllers. The main controller is added by specifying a host as the main controller. Therefore to add the Main controller, a new host must first be created as done previously:

```
jessie(config)# fibbing network hosts-link <link-name>
jessie(config-...)# router name <RouterName> ip <IPv4>
jessie(config-...)# host name <ControllerName> ip <IPv4>
jessie(config-...)# bw 100
jessie(config)# fibbing network main-controller <ControllerName>
```

The $< ControllerName >$ being the name of the Main Controller.

Then we can add the Fibbing Controller, by creating the link between the Fibbing controller and a router:

```
jessie(config)# fibbing network fibbing-controller links <link-name>
jessie(config-links-...)# router name <RouterName> ip <IPv4>
jessie(config-links-...)# controller name <FibCtrlName> ip <IPv4>
```

Then there are some configurations specific for the Fibbing controller. We do not suggest modifying them, unless you are sure of what you're doing.

First, there is the *ospf-config*:

```
jessie(config)# fibbing network fibbing-controller controller-config ospf-con
Possible completions:
  area  dead-interval  hello-interval  lsa  throttle
```

[style=DOS] these *ospf-config* should be the same as the OSPF configs of the routers present in the network, as the Fibbing controller needs to be able to inject OSPF messages destined to those routers[3]. Then, there is the *private-ip-prefix* and *base-net-prefix*:

```
jessie(config)# fibbing network fibbing-controller controller-config private-
jessie(config)# fibbing network fibbing-controller controller-config base-net-
```

The *private-ip-prefix* should be a `/8` prefix, and the *base-net-prefix* should be a `/16` prefix, that no IP addresses of the network belong to.

3.2.5 Fibbing requirements

Requirements are used to redirect and modify the traffic flows in the network. We decided to implement four types of requirements: Simple requirements, Time requirements, Bandwidth requirements, Backup requirements.

3.2.5.1 Simple requirements

To configure a simple requirement use:

```
jessie(config)# fibbing fibbing-requirement  requirement <name1>
jessie(config-... )# dest <name2>
jessie(config-... )# Path <path>
```

where $< name1 >$ is the name you want to give to your requirement, $< name2 >$ is the name of the host you want to reach, $< path >$ is the path you want this specific flow to follow. The format to use is [A B C]. where A,B and C are existing routers in the network. By default the requirement is not-running. To change that the state of the requirement must be changed:

```
jessie(config-requirement-<name>)# state running
```

Where $< name >$ is the name that was chosen for the requirement.

3.2.5.2 Time requirements

Time requirements are created in the same way as simple requirements. The only difference is that the state has to be set to scheduled. One must also configure the schedule during the creation of the requirement. This is done using:

```
jessie(config-requirement-<name>)# schedule Type time
jessie(config-requirement-<name>)# schedule days <days>
jessie(config-requirement-<name>)#  schedule start-hour <start> end-hour <end
```

Where $< name >$ is the name that was given to the requirement, $< days >$ are the days you want the requirement to be active. It has to be written in the format: [Monday Wednesday]. $< start >$ is the time at which you want the

[3]So if you changes the default OSPF config for the routers, you will need to change the OSPF config of the controller

requirement to start being active and $< end >$ the time at which it should stop. The times need to be written in the format 12:00

3.2.5.3 Bandwidth requirements

To specify a Bandwidth requirement the schedule field needs to be changed as well:

```
jessie(config-requirement-<name>)# schedule Type bandwidth
jessie(config-requirement-<name>)# schedule link bw-percentage <num>
jessie(config-requirement-<name>)# schedule link from <router1> to <router2>
```

where $< name >$ is the name that was given to the requirement. $< num >$ is the bandwidth percentage that should be exceeded before the requirements becomes active and $< router1 >$ and $< router2 >$ are the routers on each side of the concerned link.

3.2.5.4 Backup requirements

The last type of requirements are the backup requirements. Those requirements will be active if the link for which they are the backup is down. To specify that a requirement is a backup requirement, the schedule field needs to be set as follow:

```
jessie(config-requirement-<name>)# schedule Type backup
jessie(config-requirement-<name>)# schedule link from <router1> to <router2>
```

where $< name >$ is the name that was given to the requirement and $< router1 >$ and $< router2 >$ are the routers on each side of the concerned link.

4 Getting Started

In this section we will illustrate with an example how to configure a network (shown on image 1) using the Fibbing solution and how to create a requirement.

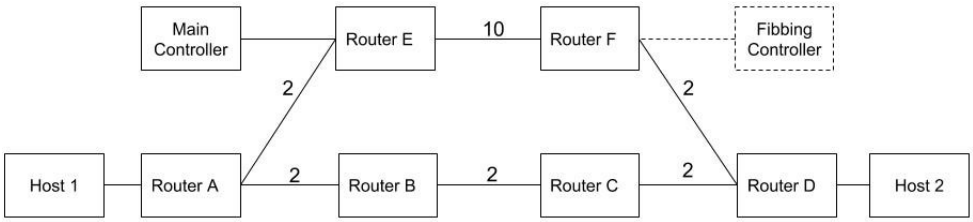

Figure 1: Exemple of a simple network

4.1 The network with fibbing

In order to configure the network, we will need to start the ConfD CLI. Once the CLI is started, use *config* to enter the network configuration mode. To create a network, one must first declare all the routers present in the network.

4.1.1 Create the routers

To create a router, use the following command in config mode:

```
jessie(config)# fibbing network router <router-name>
```

where <router-name> is the name you want to give to a router. In our example this command needs to be entered six different time, with each time a different router-name ranging from RouterA to RouterF. Now that the routers have been declared, one can declare and configure the network links.

4.1.2 Create the links

To create a new link, we first create a link by giving it a unique name. Then the user has to specify the source and destination of his link using previously declared routers and by assigning them an ip. The user can then change the bandwidth and cost of the link (otherwise the default values will be used). As example to create the link between router A and router B that we will call linkA-B:

```
jessie(config)# fibbing network link linkA-B
jessie(config-link-linkA-B)# src name RouterA ip 192.168.1.2/24
jessie(config-link-linkA-B)# dest name RouterB ip 192.168.1.1/24
jessie(config-link-linkA-B)# cost 2
jessie(config-link-linkA-B)# bw 100
```

Once all the links are created, we will need to enable OSPF on the links, before that the user also needs to enable OSPF on each router and to give each router a router-id. For linkA-B This is done by using:

```
jessie(config)# fibbing network router A ospf enable true
jessie(config)# fibbing network router A router-id 1.1.1.1
jessie(config)# fibbing network router B ospf enable true
jessie(config)# fibbing network router B router-id 2.2.2.2
jessie(config)# fibbing network ospf-link-config linkA-B
```

The user can also modify the OSPF configs of the interfaces if wanted.

4.1.3 Adding the hosts

Adding a host is almost similar to creating the link between two routers, the user just needs to create the link between the host and the previously declared router. As example for Host1 that is connected to RouterA:

```
jessie(config)# fibbing network hosts-link Host1-A
jessie(config-hosts-link-Host1-A)# router name RouterA ip 111.111.1.2/24
jessie(config-hosts-link-Host1-A)# host name Host1 ip 111.111.1.1/24
jessie(config-hosts-link-Host1-A)# bw 100
```

Once the hosts are added, the last thing to do for the Fibbing solution is to add the controllers to the network.

4.1.4 The controllers

There are 2 types of controllers: One unique Main Controller and Fibbing Controllers. The main controller is added by specifying a host as the main controller. Therefore to add the Main controller, a new host must first be created as done previously:

```
jessie(config)# fibbing network hosts-link MainCtrl-A
jessie(config-...)# router name RouterA ip 222.222.2.1/24
jessie(config-...)# host name MainController ip 222.222.2.2/24
jessie(config-...)# bw 100
jessie(config)# fibbing network main-controller MainController
```

Then we can add the Fibbing Controller, by creating the link between the Fibbing controller and a router:

```
jessie(config)# fibbing network fibbing-controller links FibCtrl-F
jessie(config-links-...)# router name RouterF ip 67.45.78.1/24
jessie(config-links-...)# controller name FibCtrl ip 67.45.78.2/24
```

5 Autotopo framework

The Autotopo framework enables to generate NETCONF configuration files, based on an input topology and some requirements. For the topology only the basic information are required, such as link connectivity and the role of the different nodes (e.g. controller, router, destination). The framework will generate the IP address and router-ID automatically.

5.1 Getting Started with Autotopofib

To start writing a topology with Autotopofib, we suggest to copy the `build.py` file located in with the framework code: **autotopo/autotopofib**. As an example, the **netconf/AutoAbilene** contains a practical example of how to use the framework to generate the Abilene topology[4], and to specify some requirements.

6 Simulation

Simulation can be used for easy experimentation. It allows to automatically simulate user interaction with the CLI, and events on the network (typically, fast forward time condition, or bandwidth detection). We suggest to only use the simulation framework once you are comfortable enough with the CLI, and the Autotopo framework. We provide an example of how to use simulation in **simulation/example/**. ATTENTION, the simulation must still always be launched from the working directory **NetworkManager**. We suggest to rely on the `run_automated_simulation.py` script to launch the simulation. Example:

```
.../NetworkManager# python run_automated_simulation.py <simulation-scenario>
```

[4] http://noc.net.internet2.edu/i2network/maps-documentation/maps/internet2-ip-igp-metrics2.html

where <*simulation-scenario*> is the sub-directory name of **simulation/** that you want to launch. In order to use the **run_automated_simulation.py** script your simulation file **must** be named `simulation.py`. We strongly advised to take as inspiration the examples already present in the **simulation/** directory.

7 Troubleshooting

7.1 Clearing all stored configurations

In order to clear all stored configurations on the ConfD Agent, there are different ways.

The first one is **not recommended** but will have the intended effect : In the main working directory (`NetworkManager/`), first make sure that no `confdagent` is still running with :

```
service confdagent stop
```

then run

```
make clear
```

This will remove all compiled files concerning the Confd Agent, and clear all previous history. You will have then to re-compile the ConfD Agent, and restart it:

```
make compile
service confdagent start
```

The second one consist on manually clear the previous configurations via the ConfD CLI. To this end you will need to first enter the ConfD CLI by one of the two ways: either you are already running the main application (`make manager-CLI` or `python main/main.py`) or you are on the working directory but not running the application.
If you are running the application, you can run the command `config` that will enter the ConfD CLI. If you are in the working directory, you can run:

```
make cli
```

From here the following command are the same.

First enter configuration mode then, delete all stored configurations related to the network requirement:

```
jessie(config)# no fibbing fibbing-requirement
```

delete all stored configurations related to the network topology:

```
jessie(config)# no fibbing network
```

Commit your modifications to make them permanent.

7.2 `Aborted: illegal reference` when configuring a requirement

If you encounter a `Aborted: illegal reference` error message when configuring a requirement. Either you have reference a router in the requirement that does not exist. In which case verify the configuration of the router

```
jessie(config)# show full-configuration fibbing network
```

If you are configuring a requirement with the ***** symbol, please verify that the ***** symbol is well configure as being a router.

```
jessie(config)# mininet network ospf ospf-routers *
```

This router will be ignored by the application but nonetheless need to be stored on ConfD, so that it can be referenced by the requirement.