Documentation - Segment Routing

Vanden Bulcke Cedric & Lejoly Florent

June 7, 2017

Contents

1 Installation

The current solution works with the SR-IPv6 - Linux Kernel implementation (see segment-routing.org).

1.1 Dependencies

You would need to have installed:

- Quagga: apt-get install quagga (for debian/ubuntu)
- SNMP: install snmp snmp-mibs-downloader (for debian/ubuntu) or follow tutorial at https://www.digitalocean.com/community/tutorial_series/monitoring-and-managing-your-network-with-snmp
- ConfD: install from http://www.tail-f.com/management-agent/
- other dependecies: python-matplotlib, python-paramiko libxml2-utils, python-termcolor

1.2 Using the VM

We may provide a Virtualbox VM containing all the relevant packages, and the source code. We advice to connect to the VM via SSH, and with X server enabled (i.e. with the `-X` or `-Y` option). You can request the VM at network-managercore@gmail.com.

1.3 Setup

Once you have ssh the VM, you can enter **root** privilege with:

```
sr6@sr6:~$ sudo su
```

We have disable the root password for the sake of simplicity. It should be noted that most of the command that you will need to run needs root privilege.

You can then enter the main working directory *NetworkManagerCLI/SegmentRouting/* with:

```
root@sr6:/home/sr6# cd NetworkManagerCLI/SegmentRouting/
```

From the main working directory (*NetworkManagerCLI/SegmentRouting/*), you can start the application with:

```
root@sr6:/home/sr6/NetworkManagerCLI/SegmentRouting# python main.py
```

or with `python main.py -D` for **debug mode**. In Debug mode, more feedback information will be displayed.

2 ConfD Agent

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

```
root@sr6:/home/sr6/NetworkManagerCLI/SegmentRouting# 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@sr6:/home/sr6/NetworkManagerCLI/SegmentRouting# 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@sr6:/home/sr6/NetworkManagerCLI/SegmentRouting# 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@sr6:/home/sr6/NetworkManagerCLI/SegmentRouting# service confdagent statu
```

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

```
root@sr6:/home/sr6/NetworkManagerCLI/SegmentRouting# service confdagent stop
```

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

```
root@sr6:/home/sr6/NetworkManagerCLI/SegmentRouting# make cli
```

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

```
root@sr6:/home/sr6/NetworkManagerCLI/SegmentRouting# make stop-daemon
```

3 Segment Routing CLI

Here will be presented the commands that can be used to interact with the application.

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 network.

config	Start the configuration CLI supported by ConfD. Requires that the ConfD agent is running (see section 1.3). 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 SR requirements already pushed by the controller. This command takes the *key* of the stored requirement as argument: run *show_requirements* to have information about stored requirements. Note that the key is the name of the requirement configured via the ConfD CLI.
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.
quit	Run this command to quit the application. Requires that the network be stopped.
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, 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_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.
terminal	Run this command to start a bash terminal. This is mostly useful for debugging.
connect_ospf6d	Run `connect_ospf6d [ROUTERNAME]` to connect to the Quagga ospf6d daemon running on the router [ROUTERNAME].
connect_zebra	Run `connect_zebra [ROUTERNAME]` to connect to the Quagga zebra daemon running on the router [ROUTERNAME].
ping6	Run `ping6 [ROUER1] [ROUTER2]` to test ping connectivity between the [ROUTER1] and [ROUTER2]. Note that the ping6 will be conducted based on the loopback address of the router.
show_ip_route	Run `show_ip_route [ROUTER]` to display the routing table of [ROUTER].
ssh_router	Run `ssh_router [ROUTER]` to ssh [ROUTER] via the controller.
traceroute6	Run `traceroute6 [ROUTER1] [ROUTER2]` to display the output of traceroute6[1] between [ROUTER1] and [ROUTER2].
xterm	Run `xterm [ROUTER]` to open a xterm terminal on [ROUTER]. This is useful for running command on specific router (e.g. Iperf to send traffic from a router to another).

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 CLI. This can be done using the command *config* in the Segment Routing Manager CLI or *make cli* in the `SegmentRouting/` 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.

[1] https://linux.die.net/man/8/traceroute6

3.2.1.1 create a new router

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

```
sr6(config)# segment-routing network routers <name> loopback-addr <ipv6-addr>
```

where $< name >$ needs to be replaced by the name that you want to give to the router and $< ipv6 - addr >$ needs to be replaced by a IPv6 address (including network mask). The loopback address should be unique for each router, as it will be used to identify the router when configuring a SR[2] requirement.
You can then configure OSPF on this router. It should be noted that all the routers that will be present in SR requirement **must** have OSPF enable.

```
sr6(config-routers-<router-name>)# ospf6 enable <boolean>
```

$< boolean >$ being `true` when enabling OSPF (or `false` otherwise). We strongly recommend to enable OSPF on all router except if you know what you are doing. If OSPF is enable on this router, you **must** configure a unique Router-ID.

```
sr6(config-routers-A)# ospf6 router-id <routerid>
```

where $< routerid >$ has as input format x.x.x.x (or IPv4 address).
As you can see when pressing TAB:

```
sr6(config-routers-A)# ospf6
Possible completions:
  area  dead-interval  enable  hello-interval  router-id
```

there are some OSPF specific variable that can be configured as well:

area — Area is used to specify the area of the network. By default this is set to 0.0.0.0. We do not support multiple area networks, so changing this value has no real meaning.

dead-interval — This defines how long (in ms) we should wait for hello packets before we declare the neighbor dead. By default this is set to 40.

hello-interval — This defines how often (in ms) we send the hello packet. By default it is set to 5.

We provide default value for those variables, so we do not recommend modifying them unless you know what you are doing.

Now that the routers have been declared, one can declare and configure the network links.

3.2.1.2 create a new link

To create a new link, we will need to reference two existing different routers. First we will choose the source router:

[2]Segment Routing

```
sr6(config)# segment-routing network link <link-name>
```

where $< link - name >$ is the name[3] you want to give to the link. Then you can configure the source interface of the link:

```
sr6(config-link-AtoB)# src name <router-name> ip <ipv6-addr>
```

where $< router - name >$ is the name of a router previously declared (see section 3.2.1.1). $< ipv6 - addr >$ is the IPv6 address (including network mask) that is configured to the interface of the router ($< router - name >$) connected to this link. Similarly, you can configure the other interface of the link:

```
sr6(config-link-AtoB)# dest name <router-name> ip <ipv6-addr>
```

note that a good way to keep track of the IP address of the link is to use a common prefix for the link (e.g. IPv6 of interface of A is fc00:42:1::1, and interface of B is fc00:42:1::2).
You can configure some parameters specific to the link itself, such as bandwidth, cost, and delay.

bw Bandwidth of the link in kbps. By default it is set to 100 000.

cost cost associated with this link. By default set to 1.

delay delay in milliseconds associated with this link. By default set to 3.

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,

```
sr6(config-link-AtoB)# bidirectional false
sr6(config-link-AtoB)# src cost 10
sr6(config-link-AtoB)# dest cost 100
```

would mean that for the given link (e.g. **A-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:

```
sr6(config)# segment-routing network ospf-link-config <link-name>
```

where $< link - name >$ referenced the name of link cnfigured 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).

[3] for example "AtoB" for a link from router A to router B

```
sr6(config)# segment-routing network ospf-link-config  <link-name> src
Possible completions:
  area  dead-interval  hello-interval  lsa  throttle
sr6(config)# segment-routing network ospf-link-config  <link-name> dest
Possible completions:
  area  dead-interval  hello-interval  lsa  throttle
```

3.2.3 Destinations

Destination here are routers for which we want to create requirement. In order to configure them, you just need to specify a name ($< dest >$) for the destination and the name of the associated router ($< name >$):

```
sr6(config)# segment-routing network destinations <dest> router-name <name>
```

3.2.4 Main Controller

In order to be able to interact with the network, you **must** configure a controller. We consider out of simplicity that the controller is configure as simple OSPF router. You can thus declare a router with OSPF enabled (see section 3.2.1.1), and then configure a link between the controller and a router of the network, with OSPF enabled as well (see section 3.2.1.2). You can finally declare which router ($< name >$) will act as the controller with:

```
sr6(config)# segment-routing network main-controller <name>
```

3.2.5 (SR) 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:

```
sr6(config)# segment-routing sr-requirement  requirement <name1>
sr6(config-... )# dest <name2>
sr6(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:

```
sr6(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:

```
sr6(config-requirement-<name>)# schedule Type time
sr6(config-requirement-<name>)# schedule days <days>
sr6(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 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:

```
sr6(config-requirement-<name>)# schedule Type bandwidth
sr6(config-requirement-<name>)# schedule link bw-percentage <num>
sr6(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:

```
sr6(config-requirement-<name>)# schedule Type backup
sr6(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[4]) using the Segment Routing solution and how to create a requirement.

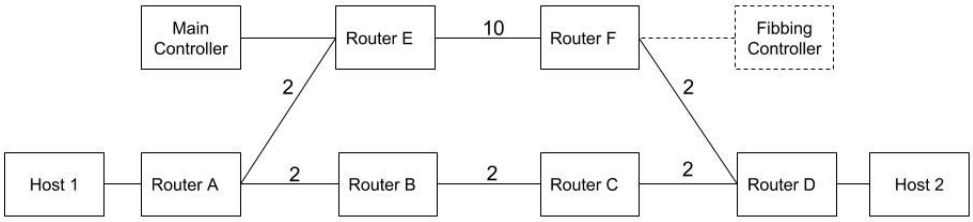

Figure 1: Exemple of a simple network

4.1 The network with Segment Routing

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.

4.1.1 Create the routers

Like for Fibbing, we will first declare all the routers. The difference being that the Segment Routing routers need a loopback adress and that the hosts need to be created as routers too. As exemple here how to create router A:

```
sr6(config)# segment-routing network router RouterA
sr6(config-router-RouterA)# loopback-addr fe12::01/64
sr6(config-router-RouterA)# ospf enable true
sr6(config-router-RouterA)# router-id 1.1.1.1
```

4.1.2 Create the links

The links are created in the same way as for Fibbing (except for the fact that ipv6 is used):

```
sr6(config)# segment-routing network link linkA-B
sr6(config-link-linkA-B)# dest name RouterB ip fe13::01/64
sr6(config-link-linkA-B)# src name RouterA ip fe13::02/64
sr6(config-link-linkA-B)# bw 100
sr6(config-link-linkA-B)# cost 2
```

Once the links are created OSPF needs to be enabled on the links. This is done like for Fibbing. For linkA-B we have:

```
sr6(config)# segment-routing network ospf-link-config linkA-B
```

[4]For the Segment Routing Solution we can ignore the *Fibbing Controller*

Once again the user can modify the OSPF configuration of the interfaces if wanted. For more information about the OSPF configs see the chapter about the YANG (chapter??).

```
sr6(config)# segment-routing network ospf-link-config linkA-B src
Possible completions:
  area  dead-interval  hello-interval  lsa  throttle
sr6(config)# segment-routing network ospf-link-config linkA-B dest
Possible completions:
  area  dead-interval  hello-interval  lsa  throttle
```

4.1.3 Adding the destinations

In Segment Routing, destination are declared as routers and are then given the role of destinations. Therefore to create Host1, Host1 must have first be created as a router. Next following command is used:

```
sr6(config)# segment-routing network destinations Host1
```

4.1.4 The controller

Using Segment Routing, there is only one controller: the main controller. The main controller is added by specifying a router into the main controller. Therefore to add the Main controller, the main-controller must have been created as a router previously:

```
sr6(config)# segment-routing network main-controller Main-controller
```

4.2 Requirement

4.2.1 Simple requirements

Let's say we want to configure a requirement that forces the traffic with as destination Host2 to go via the path < RouterA RouterE RoouterF RouterD >. We will give this Requirement the name 'Requirement1'. The command to use when working with fibbing will be:

```
sr6(config)# segment-routing sr-requirement requirement Requirement1
sr6(config-...)# dest Host2
sr6(config-...)# Path [ RouterA RouterE RouterF RouterD ]
```

Once the requirement is created, the user has to change the state of the requirement to 'running' to push the requirement into the network:

```
sr6(config-...)# state running
```

4.2.2 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. Lets create the same requirement as previously, but this time the requirement will only be

running between 12:00 and 15:00, during the weekend. We will also give our new requirement the name 'requirement2'.

```
sr6(config)# segment-routing sr-requirement requirement Requirement2
sr6(config-...)# dest Host2
sr6(config-...)# Path [ RouterA RouterE RouterF RouterD ]
sr6(config-...)# state scheduled
sr6(config-...)# schedule Type time
sr6(config-...)# schedule days [ Saturday Sunday ]
sr6(config-...)# schedule start-hour 12:00 end-hour 15:00
```

4.2.3 Bandwidth requirements

To specify a Bandwidth requirement the schedule field needs to be changed as well. The bandwidth requirement will become running when the bandwidth percentage of the specified link exceeds the bandwidth percentage that was specified. As example we will create the same requirement as before but this time it will only be running if the traffic between RouterB and RouterC exceeds 60% of the bandwidth, every time:

```
sr6(config)# segment-routing sr-requirementrequirement Requirement3
sr6(config-...)# dest Host2
sr6(config-...)# Path [ RouterA RouterE RouterF RouterD ]
sr6(config-...)# state scheduled
sr6(config-...)# schedule Type bandwidth
sr6(config-...)# schedule days [ Monday Tuesday ... Saturday Sunday ]
sr6(config-...)# schedule start-hour 00:00 end-hour 23:59
sr6(config-...)# scheduled link from RouterB to RouterC
sr6(config-...)# scheduled link bw-percentage 60
```

4.2.4 Backup requirements

The last type of requirements is the Backup requirements. As for the two previous ones, the schedule field must be changed. The Backup requirement will become running if the link for which it is a backup goes down. As example we will once again create the same requirement as before, but this time the requirement will become running if the link between RouterB and RouterC goes down, every time:

```
sr6(config)# segment-routing sr-requirement requirement Requirement4
sr6(config-...)# dest Host2
sr6(config-...)# Path [ RouterA RouterE RouterF RouterD ]
sr6(config-...)# state scheduled
sr6(config-...)# schedule Type backup
sr6(config-...)# schedule days [ Monday Tuesday ... Saturday Sunday ]
sr6(config-...)# schedule start-hour 00:00 end-hour 23:59
sr6(config-...)# schedule link from RouterB to RouterC
```

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 AutotopoSR

To start writing a topology with AutotopoSR, we suggest to copy the `build.py` file located in with the framework code: **autotopo/autotopoSR**. As an example, the **netconf/AutoAbilene** contains a practical example of how to use the framework to generate the Abilene topology[5], 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 examples of how to use simulation in **simulation/**. There are different scenarios in this directory, with for each a README explaining the test. ATTENTION, the simulation must still always be launched from the working directory **SegmentRouting/**. We suggest to rely on the `run_automated_simulation.py` script to launch the simulation. Example:

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

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.

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

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 (`SegmentRouting/`), 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 consists on manually clearing 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.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:

```
sr6(config)# no segment-routing sr-requirement
```

then, delete all stored configurations related to the network topology:

```
sr6(config)# no segment-routing network
```

Commit your modifications to make them permanent.
ATTENTION the order of the operations is important! You need for FIRST clear the requirements and SECOND clear the network configuration. If you inverse the two operations, it may be that there are some consistency issues. Since the requirements explicitly references router names in the network configurations if you delete all network configurations and then all requirements it may be that ConfD prevent it.