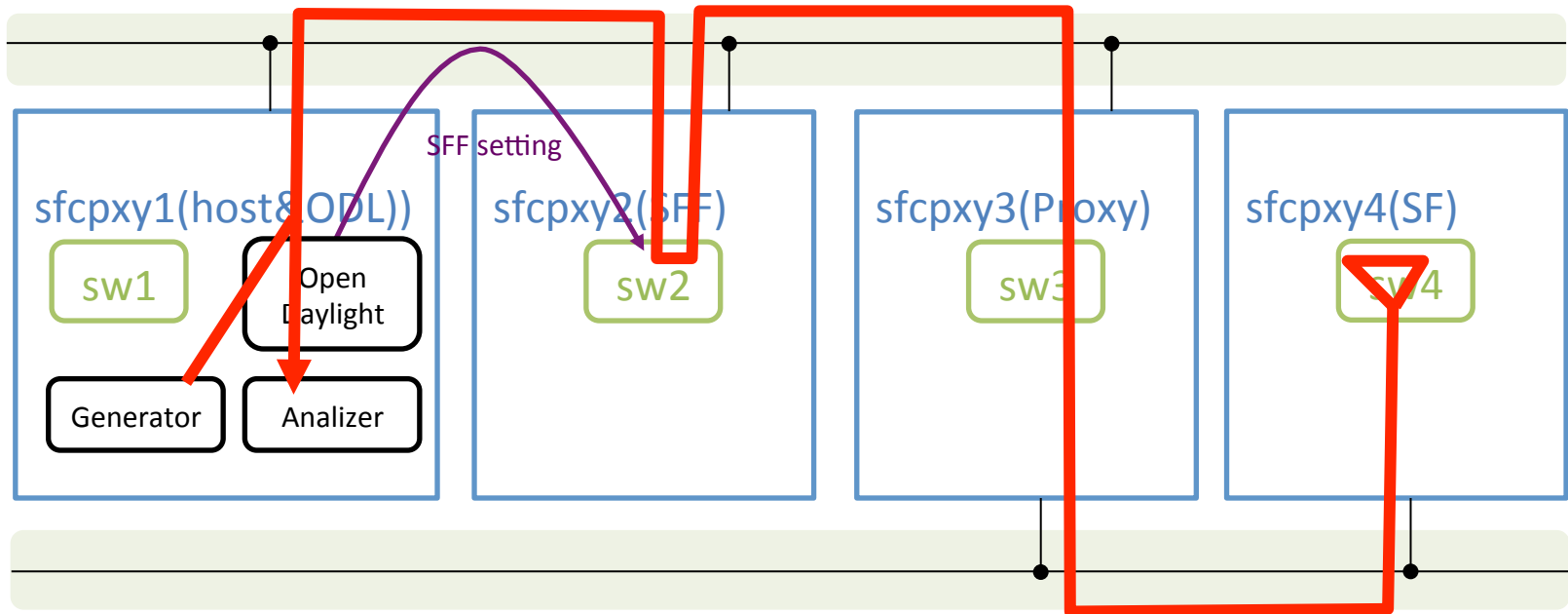


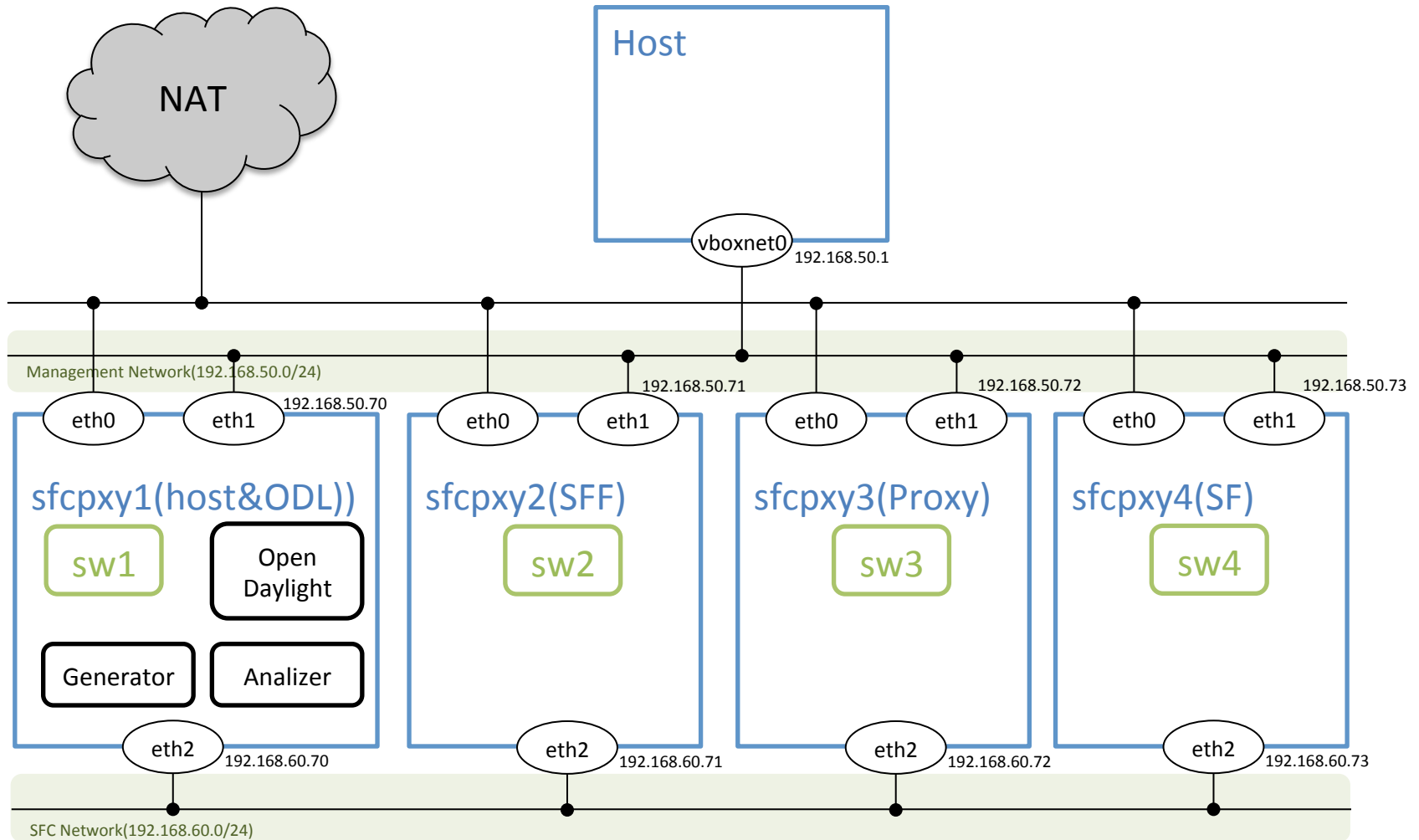
Revision

- 31 Oct 2015 23:38 JST
 - Topologies are changed.
- 31 Oct 2015 15:05 JST
 - Add “How to analyze nsh header in Wireshark”.

Network Topology(logical)

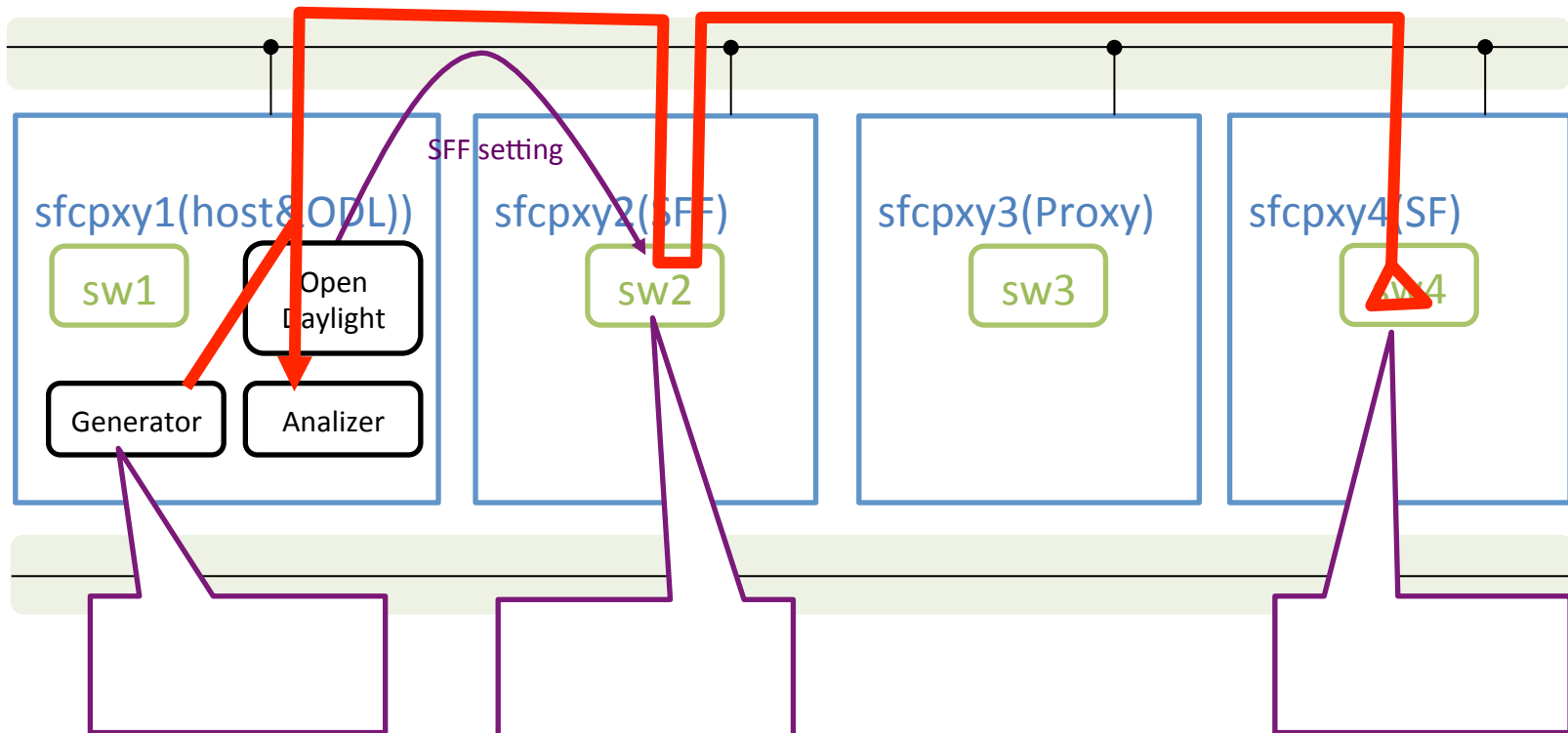


Network Topology(detailed)



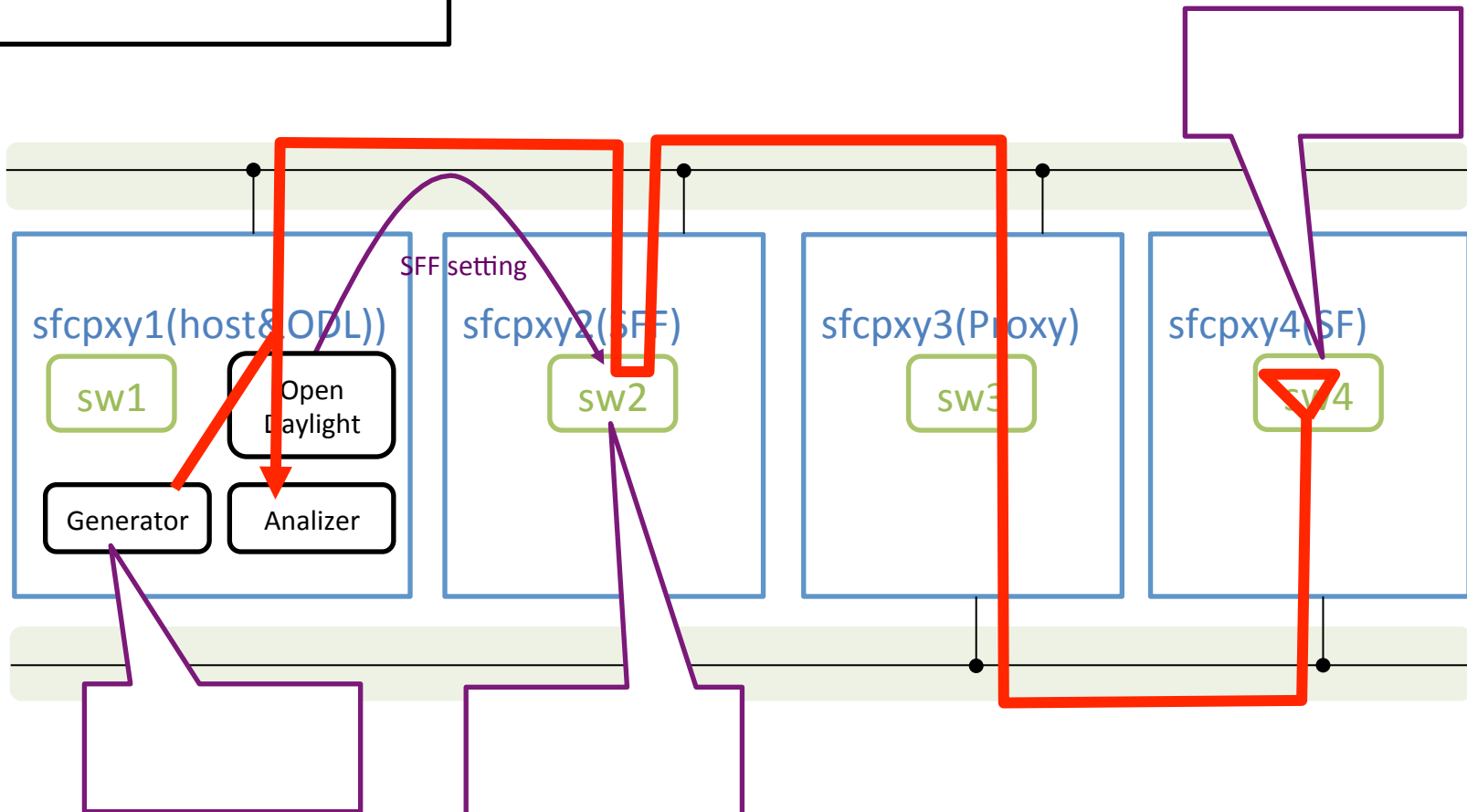
Scenario1: NSH-aware SF

This scenario does not use SFC-Proxy.
It's for checking test environment.



Scenario2: non NSH-aware SF

This scenario is our expectation.



Preparation (Before testing)

- Install the applications below.
 - VirtualBox
 - <https://www.virtualbox.org/wiki/Downloads>
 - Vagrant
 - <https://www.vagrantup.com/downloads.html>
- ~~Install the OpenDaylight.~~
 - ~~<https://www.opendaylight.org/downloads>~~
 - It includes in the VM images.
- Download VM image file.
 - https://atlas.hashicorp.com/toshirin/boxes/sfcpxy_base/versions/1.1/providers/virtualbox.box
 - Large file about 1.17GB.

Start Network & VMs(1)

- 1) Clone the “sfcpxy” git repository.
 - `git clone https://github.com/toshirin/sfcpxy.git`
- 2) In the repository, modify the “Vagrantfile”.
 - Find “###Your Directory Here###”, and replace to your VM image file path which you have downloaded before.
- 3) Read the configuration, then start VMs.
 - **`source ./env.sh`**
 - `vagrant up`
 - This may take several minutes.

Start Network & VMs(2)

4) Run the OpenDaylight.

- 1) Login to sfcpxy1: `vagrant ssh sfcpxy1`
- 2) Extract the OpenDaylight: `tar -xvzf distribution-karaf-0.3.2-Lithium-SR2.tar.gz`
- 3) Run the OpenDaylight: `cd distribution-karaf-0.3.2-Lithium-SR2; bin/karaf`
- 4) Install the features: `feature:install odl-sfc-core odl-sfc-ui odl-sfc-ui odl-sfcofl2`

5) Run the setting scripts.

- `./startdemo.sh nsh-aware`
 - The argument phrase is for selection of test scenarios.
 - As a first step, select “nsh-aware”.

6) If you want to login to the VMs, use “vagrant ssh sfcpxyN”.

- N is a host number.

Test procedure

- (All settings are in sfcpxy1.)
- Generate NSH packet
 - `cd sfc/sfc-py/sfc`
 - `python3.4 sff_client.py --remote-sff-ip=10.0.35.1 --sfp-id=1 --sfp-index=255 --encapsulate=vxlan-nsh-ethernet-legacy --inner-src-ip=10.1.0.10 --inner-dest-ip=10.2.0.10`
 - adjusting...
- Analyze NSH packet
 - `sudo tcpdump -i eth0`

Stop Networks & VMs

- Stop networks:
 - `./cleandemo.sh`
- Stop VMs temporarily:
 - `vagrant halt`
 - If you want to re-run, you also execute “vagrant up”.
- Remove VMs:
 - `vagrant destroy`

Some tips

- Directory share
 - Host:<vagrant execution dir> = sfcpxyN:/vagrant
 - It's useful for sharing the packet capture file.

Trouble shooting

- If you don't go well by the environment, check below.
 1. Don't you forget "source ./env.sh"?
 2. What is your vagrant version?
 - The older version doesn't work. We recommend **over 1.7**.
 3. Are your VMs running longer?
 - Longer operation may cause dirty status.
 - Clean the current VMs by "vagrant destroy; vagrant up".
 4. Did you update the vagrant image file?
 - Vagrant may cache the old image file.
 - Delete cache file in "~/.vagrant.d/boxes"(as MacOS).

How to re-build the OVS

- 1) Login to the “sfcpxy3”.
 - `vagrant ssh sfcpxy3`
- 2) Checkout the ovs-nsh repository and build script.
 - `mkdir ovs_work`
 - `cd ovs_work` `# cd=<start dir>/ovs_work`
 - `git clone https://github.com/pritesh/ovs.git`
 - `curl https://raw.githubusercontent.com/pritesh/ovs/nsh-v8/third-party/start-ovs-deb.sh > start-ovs-deb.sh`
 - `chmod +x start-ovs-deb.sh`
 - `cd ovs` `# cd=<start dir>/ovs_work/ovs`
 - `git checkout nsh-v8`
 - `git branch -v`
- 3) Work in the ovs directory. `# cd=<start dir>/ovs_work/ovs`
- 4) Re-build the ovs.
 - `cd ..` `# cd=<start dir>/ovs_work`
 - `./start-ovs-deb.sh -n`

How to analyze nsh header in Wireshark

1) Checkout the ovs-nsh repository.

- `mkdir ovs_work`
- `cd ovs_work` `# cd=<start dir>/ovs_work`
- `git clone https://github.com/pritesh/ovs.git`
- `cd ovs` `# cd=<start dir>/ovs_work/ovs`
- `git checkout nsh-v8`
- `git branch -v`

2) Copy the “vxlan-nsh-draft-7.lua” to the wireshark plugin directory.

- `mkdir ~/.wireshark/plugins`
- `cp third-party/vxlan-nsh-draft-7.lua ~/.wireshark/plugins/`
 (as MacOS)

3) The default outer UDP port is 6633. If you use another port, add the command below to the lua file.

- `udp_encap_table:add(<port number>, protocol_nsh)`