Digital Modelling and Simulation

Switch controller project

Team 3:

- Jakub Błaszczyk
- Przemysław Kozieł
- Adam Kornacki
- Tomasz Depta
- Michał Szuścik
- Olha Kalniei

Team 1:

- Michał Witas
- Adam Andrzejczak
- Krzysztof Zieliński
- Marcin Bojarski
- Michał Danisz
- Marcin Wojaczek

Project goals

- Create a simulated network topology according to the schema:

INTERNET (HOST) <-> GATEWAY/FIREWALL <-> SWITCH <-> ... <-> SWITCH <-> HOST

- Create a semi-intelligent controller able to act as the gateway/firewall but working in parallel to the switch chain.

Project assumptions

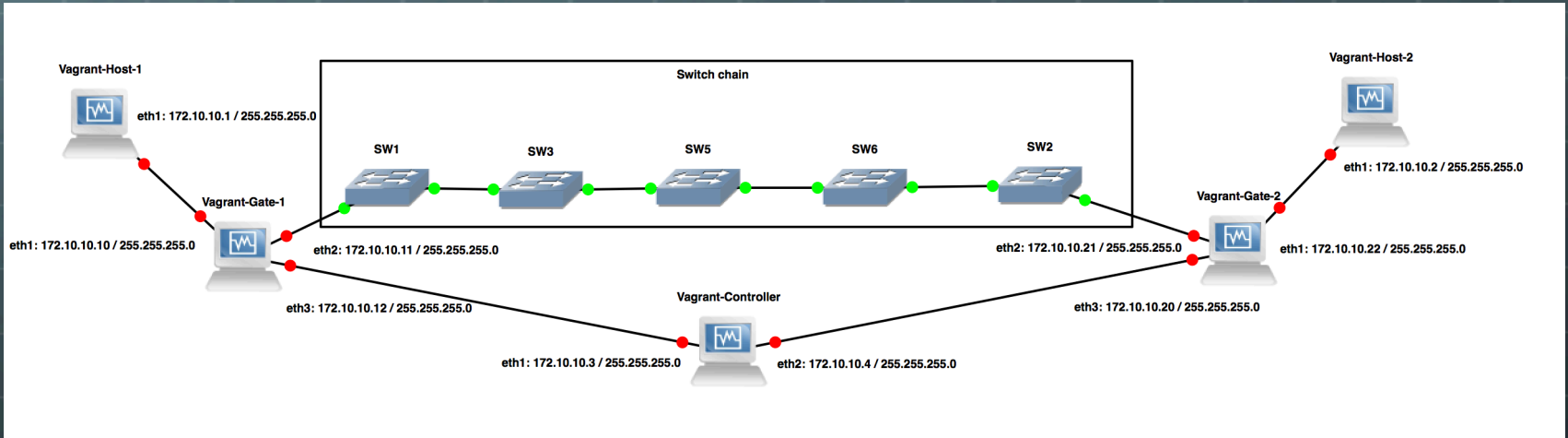
- Use the GNS (Graphical Network Simulator) to create the network with real (emulated*) machines (switches, routers if needed) to see the use of such a network in real life applications.

* Emulated means that the machines run the real OS of machines (CISCO IOS)

Project architecture – GNS network

- Use the GNS to build the network:
  - Hosts are simple VM hosts (created using Vagrant)
  - Switches are simple L2 machines
  - Controller is a more advanced VM (created using Vagrant)

GNS network


Vagrant-Host-1
eth1: 172.10.10.1 / 255.255.255.0
Switch chain
SW1
SW3
SW5
SW6
SW2
Vagrant-Host-2
eth1: 172.10.10.2 / 255.255.255.0
Vagrant-Gate-1
eth1: 172.10.10.10 / 255.255.255.0
eth2: 172.10.10.11 / 255.255.255.0
Vagrant-Gate-2
eth2: 172.10.10.21 / 255.255.255.0
eth1: 172.10.10.22 / 255.255.255.0
eth3: 172.10.10.12 / 255.255.255.0
Vagrant-Controller
eth3: 172.10.10.20 / 255.255.255.0
eth1: 172.10.10.3 / 255.255.255.0
eth2: 172.10.10.4 / 255.255.255.0

GNS Solution outcome

- Base network functions correctly (hosts are able to ping each other) and packets are copied to the controller.
- This method has a great drawback: the packets are copied on the OS level creating a delay and using system resources
- CISCO machines run by GNS do not support Openflow, requiring the controller to manage packets also on the OS level, which is not at all efficient.

GNS solution fix

- To improve the performance we would need to incorporate the openflow mechanism into the network.
- GNS offers a solution to this problem: use the Mininet VM as the controller + switch chain.
- However this solution is not attractive as we can do the same using only Mininet.

Mininet solution

Project architecture

- To properly develop the solution we have created a git repository: https://github.com/bolek117/mininet_sdn
- We have used Vagrant to create a VM box to avoid downloading the reloading the whole VM each time.

Mininet architecture

```
vagrant@mininet:~/dmas$ sudo ./runner.py -s 4
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s4-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth1
s2 lo:  s2-eth1:s1-eth2 s2-eth2:s3-eth1
s3 lo:  s3-eth1:s2-eth2 s3-eth2:s4-eth2
s4 lo:  s4-eth1:h2-eth0 s4-eth2:s3-eth2
c0
```

Connectivity

```
mininet> links
h1-eth0<->s1-eth1 (OK OK)
s1-eth2<->s2-eth1 (OK OK)
s2-eth2<->s3-eth1 (OK OK)
s3-eth2<->s4-eth2 (OK OK)
s4-eth1<->h2-eth0 (OK OK)
```

```
mininet> pingallfull
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results:
 h1->h2: 1/1, rtt min/avg/max/mdev 67.353/67.353/67.353/0.000 ms
 h2->h1: 1/1, rtt min/avg/max/mdev 14.083/14.083/14.083/0.000 ms
```

Mininet Controller

```
vagrant@mininet:~/pox$ ./pox.py spam_class
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:spam_class:Mode is 1
INFO:spam_class:Setting switches to forward full packet payloads
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[00-00-00-00-00-03 1] connected
INFO:spam_class:Changing gateway id to 3
INFO:openflow.of_01:[00-00-00-00-00-04 2] connected
INFO:spam_class:Changing gateway id to 4
INFO:openflow.of_01:[00-00-00-00-00-02 3] connected
INFO:openflow.of_01:[00-00-00-00-00-01 4] connected
INFO:openflow.of_01:[00-00-00-00-00-03 1] closed
INFO:openflow.of_01:[00-00-00-00-00-02 3] closed
INFO:openflow.of_01:[00-00-00-00-00-04 2] closed
INFO:openflow.of_01:[00-00-00-00-00-01 4] closed
INFO:openflow.of_01:[None 5] closed
INFO:openflow.of_01:[00-00-00-00-00-04 6] connected
INFO:openflow.of_01:[00-00-00-00-00-03 7] connected
INFO:openflow.of_01:[00-00-00-00-00-02 8] connected
INFO:openflow.of_01:[00-00-00-00-00-01 9] connected
INFO:openflow.of_01:[00-00-00-00-00-04 6] closed
INFO:openflow.of_01:[00-00-00-00-00-03 7] closed
INFO:openflow.of_01:[00-00-00-00-00-02 8] closed
INFO:openflow.of_01:[00-00-00-00-00-01 9] closed
INFO:openflow.of_01:[None 10] closed
INFO:openflow.of_01:[00-00-00-00-00-02 13] connected
INFO:openflow.of_01:[00-00-00-00-00-01 14] connected
INFO:openflow.of_01:[00-00-00-00-00-03 11] connected
INFO:openflow.of_01:[00-00-00-00-00-04 12] connected
```

Main task: Spam filter

- As spam filtering is relatively simple (we can search key-word in the email and judge the packets accordingly), so this task has been done in two ways:
- 1 – simple scanning in the controller basing on the email content and a banned word list.
- 2 – integration of a complex open source solution: SpamAssasin and Postfix (+ SMTP server)

Main task: Mail Scanner

- This task however required the scanning of not the mail content but of the attachment. Because of this, it was to complex to implement it as a controller action (opening packets, searching of file parts, opening it and scanning).
- Instead we implemented the ClamAV which is an open source standard for mail gateway scanning.

Machine preparation

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure(2) do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  config.vm.synced_folder "../", "/home/vagrant/dmas"
  config.ssh.forward_x11 = true

  config.vm.box = "ktr/mininet"
  config.vm.box_version = "1.1.0"

  # Currently use a private IP for the box
  config.vm.network :private_network, ip: "192.168.0.110"

   # Access through a GUI
   config.vm.provider "virtualbox" do |v|
     v.name = "KTR-Mininet"
     v.gui = false
     v.customize ["modifyvm", :id, "--cpuexecutioncap", "50"]
     v.customize ["modifyvm", :id, "--cpus", "2"]
     v.customize ["modifyvm", :id, "--memory", "2048"]
     # Internet access
     v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
     v.customize ["modifyvm", :id, "--natdnsproxy1", "on"]
  end

  config.vm.provision "shell", inline: <<-SHELL
    rm -r /home/vagrant/pox/ext
    ln -s /home/vagrant/dmas/pox/ext /home/vagrant/pox/ext
    echo "----> Update sources"
    sudo apt-get -qq update
    sudo sh /home/vagrant/dmas/vagrant-box/scripts/smtp_server.sh
    sudo sh /home/vagrant/dmas/vagrant-box/scripts/spam_filter.sh
    sudo sh /home/vagrant/dmas/vagrant-box/scripts/mail_scanner.sh
  SHELL
end
```

Presentation

Thanks for your attention.