Anycast communication in SDN

TP3

2º Semestre 2019/2020

Gestão e Virtualização de Redes

4º ano MIEI

Fábio Gonçalves

Computer Communications and Networks

Departamento de Informática, Universidade do Minho

Using anycast addresses is possible to send a packet to any of many destinations. It is unlike unicast addresses – only one destination - and multicast/broadcast – multiple destinations.

Unlike multicast, it is syntactically indistinguishable from unicast addresses. It allows several machines to have the same IP address. Then the routing protocol will choose the destination according to some pre-defined rule.

SDNs, are a good strategy to define these rules on the go, without needing to reconfigure the network. These allow to define rules to route packets according to the type of traffic, port, etc. SDNs separate the data and control plane assuring more flexibility but add cost to the performance in the data plane.

The goal of this work is to implement a network with several services. These should have both anycast and unicast addresses being the management of the traffic being made using SDN. The architecture of the network is presented in the next section.

The implementation of this work will be made using Graphical Network Simulator-3. This is a network simulator that allows the combination of virtual and real devices. It also allows to create devices that are docked based, enabling to import previous docker works.

More specifically the goals of this work are:

- Build a network with several services using GNS3;
- Implement several docker based services (Using previously made containers in TP2);
- Implement a DNS server;

1. Topology

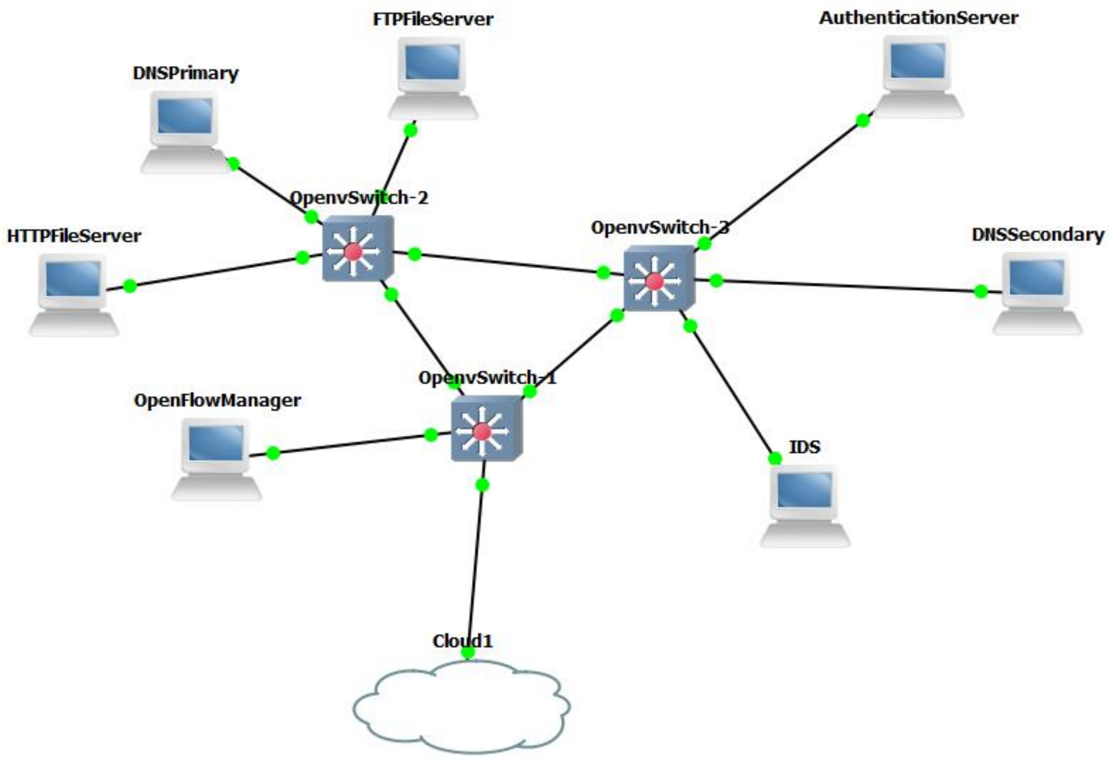


Figure 1 – Topology

Some of the components can be directly imported from the second practical work. GNS3 supports docker and so, it can be imported directly.

All the machines in this topology should have two IP addresses, a unicast and an anycast address (Except de IDS). Then, using SDN rules, the traffic will be redirected to the correct machine. For example, if the anycast address is the 10.1.1.250, and the request to http://10.1.1.254:80 is made, this should be redirected to the HTTP File Server. If the port 81 is used, it should be redirected to the Authentication Server.

All the messages should be redirected to the IDS, this should log all received messages.

Two DNS servers should be implemented (Bind9 can be used, for example). Both of them should have an anycast address that will be used by the clients. The zone transfer should happen using the unicast addresses.

2. Objectives

- Implement the presented architecture in GNS3;
- Use docker containers based on TP2;
- Install Open Flow Manager;
- Install OpenDayLight Lithium;
- All inside traffic should be made using the unicast IPs, except for the DNS queries;
- Outside traffic should only be accepted if using the anycast IPs;
- Traffic redirection:
  - Port 80 traffic should go to the authorization server;
  - Port 81 traffic should go the http file server;
  - FTP traffic should go to the ftp server;
  - Outside DNS queries should go to the primary server;
  - Inside DNS queries should go to the secondary server;
- All messages should be duplicated and directed to the IDS;

3. Implementation Steps

1. Install GNS3
2. Create the topology and test it:
   a. Configure the IP addresses of the machines;
   b. Assure that all the machines are communicating;
3. Create a tap interface for the cloud connection;
4. Install the OpenFlow Manager;
5. Configure the OpenVSwitches;
6. Use the images created in TP2 to create the (If needed other machines can be created, for the databases, for example):
   a. Authentication Server;
   b. Http File Server;
   c. FTP File Server;
7. Test The installed containers:
   a. Verify if all are functioning and communicating between them;
   b. Test outside connections using unicast addresses;
8. Create the openflow rules:
   a. Test and verify if the traffic is correctly redirected;
9. Create the IDS, this machine only needs to receive, and log all received messages;
   a. Create rules for the traffic to be redirected to this machine;
10. Create the DNS machines:

a. Create the rules for the traffic to be correctly redirected for the DNS servers;

The evaluation will be made using an external application connected through the cloud.

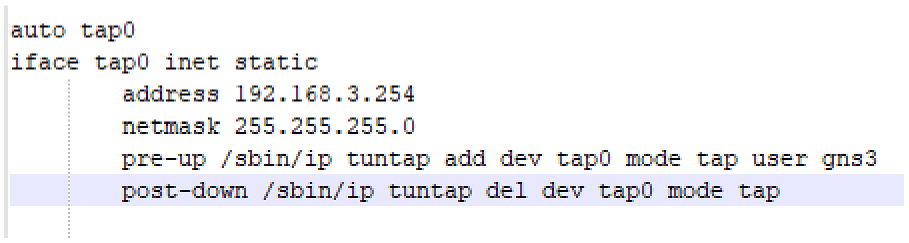
4. GNS3 and OpenFlow Manager installation and configuration

To install GNS3, follow the instructions in the website https://www.gns3.com/. The usage of the available VM with the preconfigured GNS3 VM (Not mandatory is advised).

So, after the installation the first step is to create a connection from inside the topology to the external world. So, in the machine were the GNS3 server is running create a tap interface (this are instructions for a linux machine).

3.1. Tap Interface

Edit the file /etc/network/interface. Add the following (change the IP address of the tap interface according to your needs):



```
auto tap0
iface tap0 inet static
        address 192.168.3.254
        netmask 255.255.255.0
        pre-up /sbin/ip tuntap add dev tap0 mode tap user gns3
        post-down /sbin/ip tuntap del dev tap0 mode tap
```

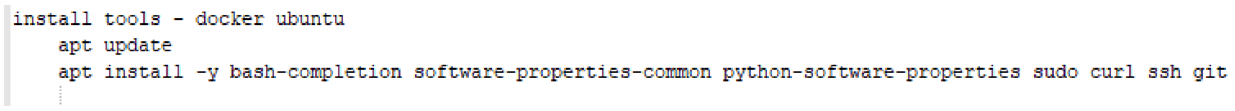
Restart the machine so the new configurations are applied.

The in the topology add a cloud and connect it to the **tap0** interface.

3.2. OpenFlow Manager

To create the OpenFlow, first add an OpenvSwitch and a docker ubuntu to the topology. Then, configure their IP addresses so they can communicate with each other and with the outside world.

In the ubuntu machine install some tools:



```
install tools - docker ubuntu
    apt update
    apt install -y bash-completion software-properties-common python-software-properties sudo curl ssh git
```

Also, install Java8 and set the environment variable JAVA_HOME.

Download and extract the OpenDaylight Lithium from: https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-karaf/0.3.0-Lithium/distribution-karaf-0.3.0-Lithium.tar.gz

Use the specified version. Other versions can create conflicts.

Then download the openflow manager app:

```
git clone https://github.com/CiscoDevNet/OpenDaylight-Openflow-App.git
```

Then use the following command (replace server_ip with the IP of the ubuntu machine where the manager is running):

```
sed -i 's/localhost/<server_ip>/g' ./OpenDaylight-Openflow-App/ofm/src/common/config/env.module.js
```

Install grunt-cli:

```
npm install -g grunt-cli
```

Run OpenDaylight lithium and install some needed features:

```
cd distribution-karaf-0.3.0-Lithium
./bin/karaf
feature:install odl-restconf-all odl-openflowplugin-all odl-l2switch-all odl-mdsal-all odl-yangtools-common
```

Then, in another terminal run the manager app:

```
cd OpenDaylight-Openflow-App
grunt
```

5. Evaluation

The evaluation will have two components, a demonstration and a report. Each group should submit the written report and any relevant files (code, configurations, etc) through the group exchange in blackboard. The report is to be submitted till 6 June. The demonstration will be made in the 7th.

The report should also include:

- Brief SND description;
- Brief OpenFlow Protocol description;
- Description of the interaction between the OpenvSwitches and the manager app;