SMART (Self MAnaged RouTing overlay)

Contents

Preamble

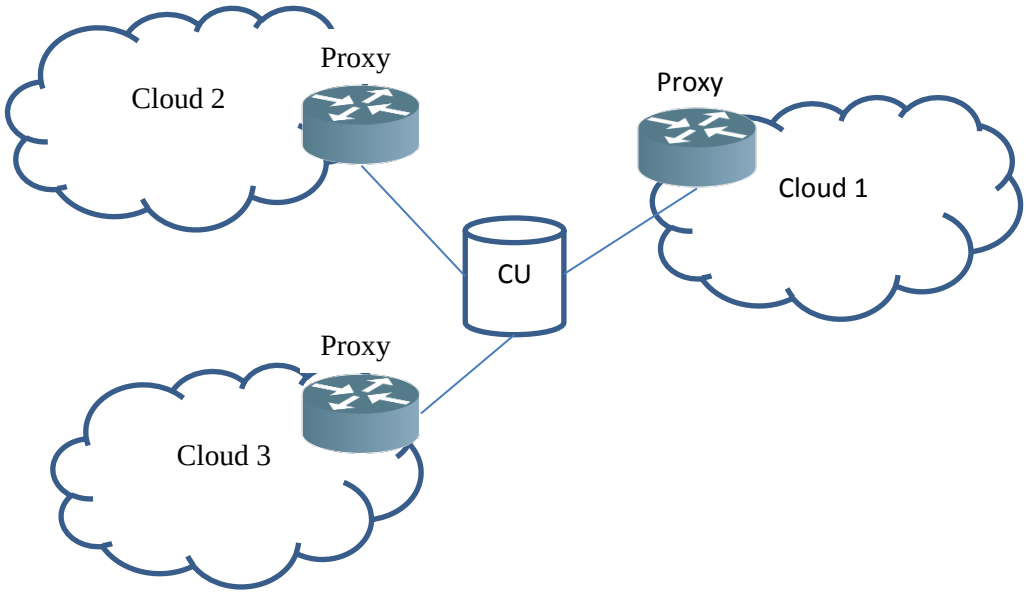
This document explains briefly the steps needed to deploy SMART on a real network. Deployment is specific to each application; meanwhile most of it is automatic. In order to illustrate the process we are going to present a scenario and detail its deployment. But before that, a brief introduction to SMART different components is necessary to understand its deployment.

SMART

SMART is an applicative system, **completely independent** of network infrastructure and client applications. It is designed to perform its own measurements, routing decisions, packet forwarding without interfering with applications or network equipment.

SMART components

SMART is composed of distributed nodes in different clouds. In order to make these nodes aware each one of the other a central unit (CU) is used to exchange configuration information. Overlay packets do not transit across the CU, they are forwarded from one proxy to another.



Communications with CU concerns only configuration information.

CU is the only inter-cloud component of SMART and it will be detailed later. In each cloud, SMART needs to operate a proxy. The proxy is composed of the following entities:

- The Router: it performs routing decisions according to intelligent algorithms in order to update the routing table of the forwarder
- The Forwarder: it receives routes updates from the router and performs packet forwarding.
- The Monitor: it performs, when asked by the router, network measurements according to specific metrics (latency).

Besides these three main components, the router contains a Daemon that communicates continuously with the CU. Its role is to provide the configuration information of the Router, Forwarder and Monitor, and receive configuration information about other clouds.

SMART does not need to interfere with applications using its services. However, on each virtual machine (VM) running an application two SMART agents should be deployed: TA and RA. Transmission Agent (TA) and Receiving Agent (RA) operate in a transparent way. They intercept packets when needed and deliver them to SMART without any modification to the application.

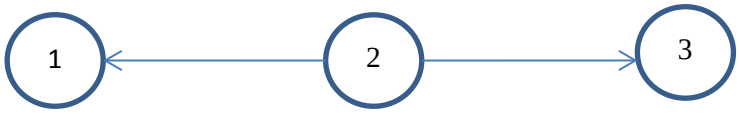
The Daemon running inside the Router communicates with TA and RA in order to send configuration information to CU and vice-versa. The Daemon is the only entity that communicates with CU in one cloud.

TA and RA should run on the same VM, the one that runs the application. On the other hand, Router, Forwarder and Monitor could be on separate VMs. Actually one Router can drive many Forwarders in order to enhance forwarding performances; however in the following example we will consider that the Router, Monitor and Forwarder run on the same VM for simplicity.

Proxy role and CU configuration

The IP address of the CU is hardcoded in the Router, so that the overlay topology will be discovered by exchanging configuration messages with CU. However, in order to define intelligent overlay network, proxies would not have the same role in all clouds. Otherwise, this will result in a full mesh network where every proxy communicates with all other proxies in the overlay network.

SMART introduces a role configuration file (graphCom) that should be provided to the CU. Roles are specific to applications. Using this file, very complex topologies could be created making it possible to define any distributed application topology. Let's take the following example.



In this example:

- Nodes of type 2 communicate with nodes with type 1 and nodes of type 3 (nodes could be simple tasks)
- Nodes of type 3 do not communicate with any other type of nodes
- Nodes of type 2 do not communicate with any other type of nodes

The graphCom file that should be provided to CU is the following:

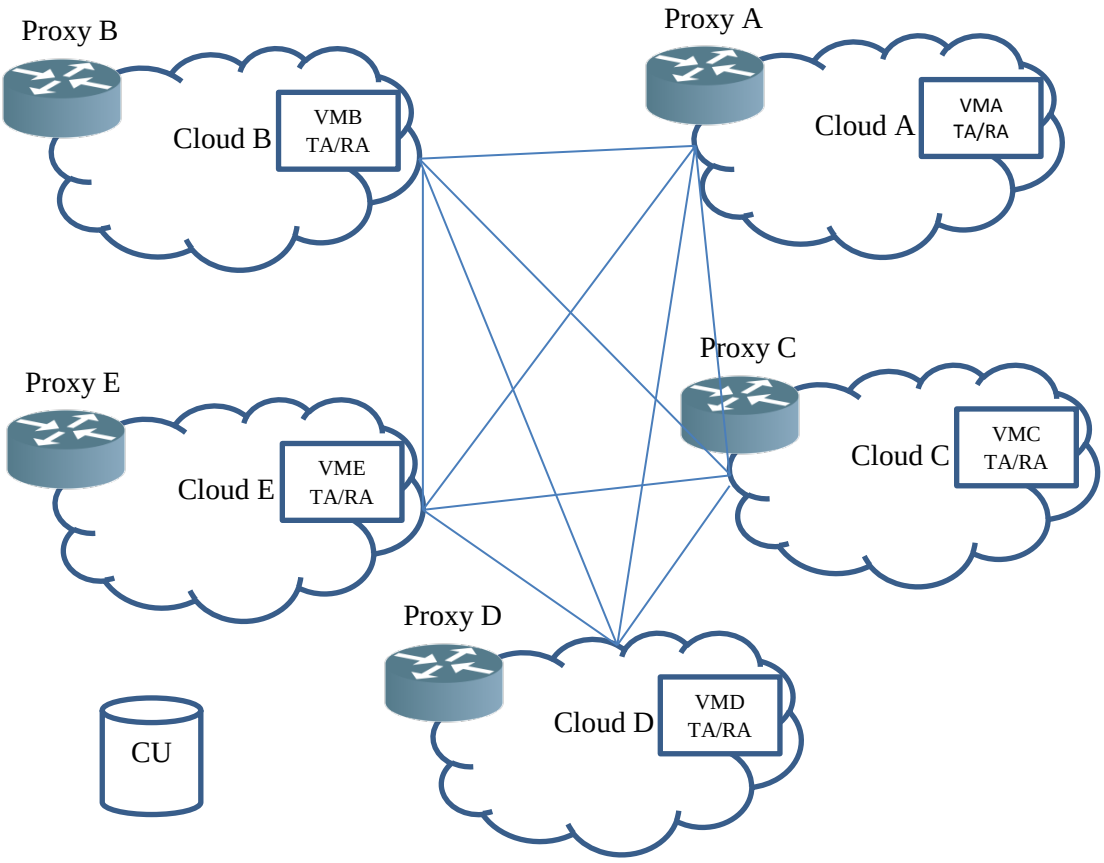
graphCom

```
1

2 1 3

3
```

As a result, in one application context, many proxies could be only intermediate nodes to provide alternative routes. They do not calculate routes neither do any monitoring. They are here just here to forward packets. Others may be source and destination, only source or only destination. The CU will receive for each proxy its IP address and its role and it should deliver this information to other nodes. Based on roles any kind of communication graphs can be described.

Scenario

Let's consider the following topology:



Using the same graphCom file seen before, we consider the following roles:

- Role 1 for clouds B and C
- Role 2 for clouds D
- Role 3 for clouds A and E

In this scenario D will try to discover alternative routes to other clouds but other clouds will only participate in the overlay as forwarders and destinations.

SMART code

The SMART code is structured as following:

- **src** folder: contains all the source files of SMART. It organized by module:
  - **common**: common libraries
  - **CU**
  - **router**
  - **forwarder**
  - **monitor**
  - **RA**
  - **TA**
  - **regret:** contains extra code for simulation result analysis
  - Doxyfile: used to generate Doxigen documentation
- **build**: where executables will be generated
- **utils**: contains a bash script to install all necessary libraries on Ubuntu VM. This script should be launched on each VM before deploying SMART
- CMakeList: to decide which executables to generate
- README.txt: explains how to use SMART

In each folder there is a README file that explains what it contains and how the code can be used.

Deploy SMART

VMs preparation

The code was compiled and tested on **Ubuntu 14.04 LTS**. All VMs should run this system. Before compiling code it is necessary to run the SMART/utils/install_dependencies_ubuntu.sh script on the VM in order to install required libraries. The script need to be launched with sudo privileges.

Compiling code

This step is explained in the SMART/build/README. However do the following:

1. cd SMART/build
2. cmake ../
3. make

All executables, unless errors were reported, will be generated in the SMART/build/exe folder.

Do not forget that the CU IP address is hardcoded in SMART/src/router/dameon.cpp. You need to change it to the public IP address of your CU before compiling.

Launching process

You need to copy executables to all VMs. You may copy only the executable needed (CU, router, forwarder…). But it is easier to copy the SMART/build/exe folder on all VMS and then launch the appropriate executable.

The launching process should respect the following order:

1. CU: the graphCom file should be located at the same place as CU executable
2. Routers: please respect arguments as explained in SMART/src/router/README
3. Monitors & Forwarders: please respect arguments as explained in SMART/src/forwarder(monitor)/README
4. Client VMs (TA & RA): do not forget to provide the configuration file mylocalip for RA if your VMs are behind a NAT.

CU will wait for router connections, gather configuration information, update all connected clouds. Each time a new VM joins the overlay the cloud router will announce its configuration to CU and CU will update all other clouds.

SMART is running…..

Detailed steps

CU VM

1- Make sure that graphCom configuration file is at the same location as CU executable
2- Launch CU
    a. ./CU

Router VMs

We suppose that router, monitor and forwarder are hosted on the same VM

1- On cloud A
    a. ./router exhaustive 3
        i. Exhaustive for the algorithm, 3 for the role
2- On cloud B
    a. ./router exhaustive 1
        i. Exhaustive for the algorithm, 1 for the role
3- On cloud C
    a. ./router exhaustive 1
        i. Exhaustive for the algorithm, 1 for the role
4- On cloud D
    a. ./router exhaustive 2
        i. Exhaustive for the algorithm, 2 for the role
5- On cloud E
    a. ./router exhaustive 3
        i. Exhaustive for the algorithm, 3 for the role

Monitor VMs

1- On each cloud

a. ./monitor routerip
    i. routerip is the IP address of router machine, in this case it is the IP address of the same VM (router, monitor and forwarder are on the same VM). It can be passed as an argument or in a file called routerip.

Forwarder VMs

1- On each cloud
    a. ./forwarder routerip
        i. routerip is the IP address of router machine, in this case it is the IP address of the same VM (router, monitor and forwarder are on the same VM). It can be passed as an argument or in a file called routerip.

Client VMs

1- On each client VM
    b. ./TA routerip
        i. routerip is the IP address of router machine. It can be passed as an argument or in a file called routerip.
    b. ./RA
        i. If the VM is behind a NAT provide its private IP address in the mylocalip configuration file. Otherwise it is the public IP address.