SDN controller clustering

Computer Networks module - SDN assignment

Michele Zanotti

Spring term 2018

Learning objectives

After finishing this lab activity you will be able to:

- Understand why clusters of controller are needed in SDN networks
- Understand which are the most common methods to build a cluster of controllers
- Implement a cluster of controllers using mininet
- Test the performance of a SDN network which uses a cluster of controllers

Overview

The most popular methods for creating a cluster of controllers are:

- by using the ovs-vsctl tool
- by python script

Cluster of controlers can be:

- physically centralized
- physically distributed

1 Task 1: build a cluster of local controllers

In this task we will create a simple network topology with multiple local controllers (that make up a cluster???).

The topology is the simple linear topology shown in figure *** and it will be created with a python script using the middle-level API provided by mininet. The topology includes two switches, each one connected to a different local controller.

The script shown in listing 1 will be used as reference in the following steps in order to progressively write the complete python script required for creating the proposed topology. The script simply creates a new empty mininet network and starts the mininet CLI: we will use it as a start point and in the next steps and we will add code for creating hosts, switches, links and controllers. In the first line of the body of the function `multiControllerNet` a new mininet network is created invoking the Mininet constructor: the parameters passed to the constructor are the Controller class and the OVSSwitch class, therefore the used controllers will be the Stanford/OpenFlow reference controllers while the used switches will be Open vSwitch switches. Note that these two classes are the default parameters in the Mininet constructor, so it is not really necessary to specify them.

Listing 1: task 1 reference python script

```
#!/usr/bin/python
from mininet.net import Mininet
from mininet.node import Controller, OVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info

def multiControllerNet():
    net = Mininet( controller=Controller, switch=OVSSwitch )

    "" Create hosts ""



    "" Create switches ""



    "" Create controllers ""



    info( "*** Running CLI\n" )
    CLI( net )

    info( "*** Stopping network\n" )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )  # for CLI output
    multiControllerNet()
```

Step 1: create a new python script

First step is creating a new text file and copying into it the content of the python script shown above. Save the file as *controllers-1.py* in your custom directory inside your mininet virtual machine.

Step 2: add host to the network

The second step is to create the hosts of the network. To do that, we need to add the following lines of code:

```
info( "*** Creating hosts\n" )
h1 = net.addHost('h3')
h2 = net.addHost('h4')
h3 = net.addHost('h5')
h4 = net.addHost('h6')
```

The first line prints to the console that hosts are being created while the others lines add four new hosts to the network respectively called h3, h4, h5 and h6.

Step 3: add switches to the network

Now we need to add the two switches to the network, therefore we have to add the following code to the respective section of the template script:

```
info( "*** Creating switches\n" )
s1 = net.addSwitch( 's1' )
s2 = net.addSwitch( 's2' )
```

As in the step 2, the first line print to the console a message to inform the user that the switches are being created, while the next two lines add two new switches to the network.

Step 4: create links between nodes

The next step is adding the links between hosts and switches and the link between the two switches. For doing that we add the following lines:

```
info( "*** Creating links\n" ) net.addLink( h3, s1 )
net.addLink( h4, s1 )
net.addLink( h5, s2 )
net.addLink( h6, s2 )
net.addLink( s1, s2 )
```

Step 5: create controllers

In this step we are going to create two local SDN controllers. The code we have to add to the script is the following:

```
info( "*** Creating (reference) controllers\n" )
c1 = net.addController( 'c1', port=6633 )
c2 = net.addController( 'c2', port=6634 )
```

Note that we specified a different TCP port for each controller (the controllers will listen on the specified port for switches that want to set up a connection).

Why did we specify a different port for each controller?

Step 6: start the network

After adding all the required nodes to the network we can finally start it. In order to do that, we have to build the mininet network and start the controllers and the switches:

```
info( "*** Starting network\n" )
net.build()
c1.start()
c2.start()
s1.start( [ c1 ] )
s2.start( [ c2 ] )
```

In the last two lines of code we used the function start() to start the two switches, passing as parameter

After this step, the python script required to build the topology for the task one is completed. The full script is shown in listing 2.

1.1 Step 7: test network connectivity and performance

The final step is execute the script and test the created topology: try to verify the network connectivity between all hosts and the bandwidth between `h3` and `h6`. Write in the lines below the commands you used and the results you obtained.

Listing 2: Task 1 complete python script

```
1  #!/usr/bin/python
2  from mininet.net import Mininet
3  from mininet.node import Controller, OVSSwitch
4  from mininet.cli import CLI
5  from mininet.log import setLogLevel, info
6
7  def multiControllerNet():
8      net = Mininet( controller=Controller, switch=OVSSwitch )
9
10     info( "*** Creating hosts\n" )
11     h1 = net.addHost('h3')
12     h2 = net.addHost('h4')
13     h3 = net.addHost('h5')
14     h4 = net.addHost('h6')
15
16     info( "*** Creating switches\n" )
17     s1 = net.addSwitch( 's1' )
```

```
18      s2 = net.addSwitch( 's2' )
19
20      info( "*** Creating links\n" )
21      net.addLink( h3, s1 )
22      net.addLink( h4, s1 )
23      net.addLink( h5, s2 )
24      net.addLink( h6, s2 )
25      net.addLink( s1, s2 )
26
27      info( "*** Creating (reference) controllers\n" )
28      c1 = net.addController( 'c1', port=6633 )
29      c2 = net.addController( 'c2', port=6634 )
30
31      info( "*** Starting network\n" )
32      net.build()
33      c1.start()
34      c2.start()
35      s1.start( [ c1 ] )
36      s2.start( [ c2 ] )
37
38      info( "*** Running CLI\n" )
39      CLI( net )
40
41      info( "*** Stopping network\n" )
42      net.stop()
43
44  if __name__ == '__main__':
45      setLogLevel( 'info' )  # for CLI output
46      multiControllerNet()
```