Final Report

Part-4

CS562_Applied Software Engineering (Alex Groce)

By: Rafid Almahdi

E-mail:mohammra@oregonstate.edu

ID: 932280933

System under Test:- Mininet

Library:- Mininet.net and Topo

File code:- Controller.py

Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking.

Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC.

Mininet provides an easy way to get correct system behavior (and, to the extent supported by your hardware, performance) and to experiment with topologies.

1- Accomplishment

I test many function in Mininet system.I found Mininet is a great system and helpful to design and implementation network. I test controller.py and test addhost (), addSwitch () function and addlink because I used these function a lot to build network topology.In this project I learn new idea about testing and how to use new beautiful language like (TSTL) .Also,I understand how to use Mininet tools and how to write network program by python and connect this program with Mininet .I choose controller.py because it heart of SDN (Software Define Network). By using TSTL I understand how testing work and what steps should use to get a perfect test. With this project I knew more about Mininet libraries, but I used Mininet and Topo.I hope I have more time to test many library and function from Mininet system.

2- The tester work

In this test I used random test test controller.py I wrote to create controller with switch and (2 host). Then I test controller.py by wrote controller.tstl test if all of topology component create or not. I try to test multicontroller.py but it’s very big file and very hard to understand the function that used in it. The coverage show me a good look about what happen when I test my controller.py .Following the topology I design and implementation.First,create controller .Second, create switch.Third,create hosts.

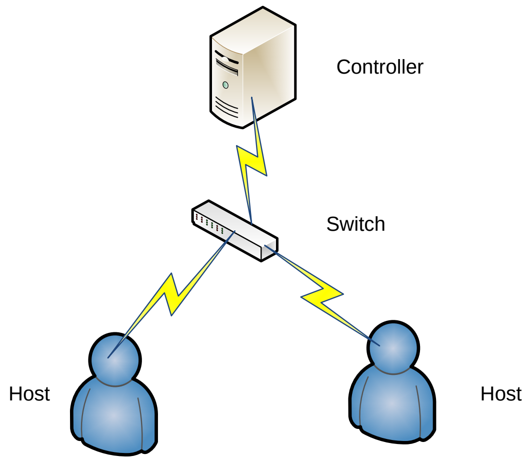


Fig. 1. Conventional SDN with a single controller, 1 switches, and 2 users.

3- Find bugs

I tested this systems a lot but there wasn’t any bugs. The system performance it works excellently. The proof of this, I have a special project to networks and I daily use the system and I didn’t find any bugs. The system is built a network and I can add pc’s and switches and others. I think, Mininet is stable and solid system, and there is no any bugs.

System work really good and researchers need it a lot. Researchers found this system helpful and they need it all the time to help them find result. These researchers worked on it really hard to find that there is no any problems or errors with the system but, there wasn’t any bugs in it. I was working on the controller.py file from Mininet system and it was really good. I didn’t find bugs in the software under test, so it doesn’t have any error or problems and nobody faced any problem with it so far.

4- Wrong with TSTL

I used to have difficulties using tstl, it was hard at first and now it easy for me because I started learning tstl grammar and understand it more. It still very hard to understand because there is nothing helpful like a book or video which can help to learn it faster and easier. I worked really hard to get along with it and it worked. Now the system is really easy for me and I don’t have any problems with it.

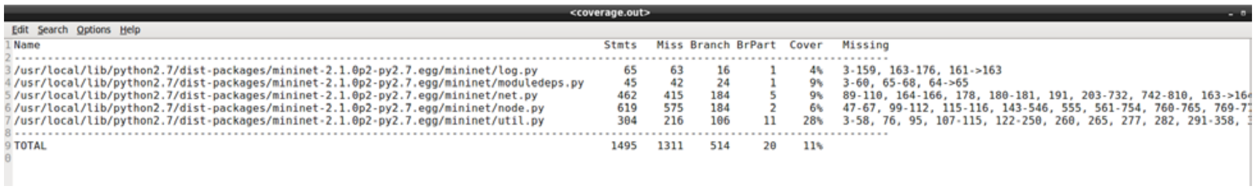
5- Good about TSTL

The good thing about TSTL, we can test large system in easy way and also it can coverage many functions in the code. TSTL can show what happen in the code in easy way. Also with TSTL, I can check if I write a perfect code. In my situation it’s helpful to create a perfect network topology by create controller and add switch with hosts. Controller.tstl test if controller controls the topology designed and implemented. When I work with TSTL, it looks great and fun and I encourage anyone to use TSTL to learn testing in perfect way.

6- Code Coverage

The total code coverage is 11%. In Fig. (2), show also that the percentage of mininet/log.py is 4%, the mininet/moduledeps.py is 9%, the mininet/net.py is 9%, the mininet/node.py is 6%, and the minimet/util.py is 28%.

I think python 2.7 and TSTL coverage the function add switch add host easily but there some libraries and functions is difficult to cover by TSTL file which I try to write.



```
Name                                                                Stmts   Miss Branch BrPart  Cover   Missing
----------------------------------------------------------------------------------------------------------------
/usr/local/lib/python2.7/dist-packages/mininet-2.1.0p2-py2.7.egg/mininet/log.py            65     63     16      1     4%   3-159, 163-176, 161->163
/usr/local/lib/python2.7/dist-packages/mininet-2.1.0p2-py2.7.egg/mininet/moduledeps.py     45     42     24      1     9%   3-60, 65-68, 64->65
/usr/local/lib/python2.7/dist-packages/mininet-2.1.0p2-py2.7.egg/mininet/net.py           462    415    184      5     9%   89-110, 164-166, 178, 180-181, 191, 203-732, 742-810, 163->16
/usr/local/lib/python2.7/dist-packages/mininet-2.1.0p2-py2.7.egg/mininet/node.py          619    575    184      2     6%   47-67, 99-112, 115-116, 143-546, 555, 561-754, 760-765, 769-7
/usr/local/lib/python2.7/dist-packages/mininet-2.1.0p2-py2.7.egg/mininet/util.py          304    216    106     11    28%   3-58, 76, 95, 107-115, 122-250, 260, 265, 277, 282, 291-358,
----------------------------------------------------------------------------------------------------------------
TOTAL                                                                1495   1311    514     20    11%
```

Fig (2):- code coverage

Name Stmts Miss Branch BrPart
Cover Missing

/usr /local /lib /python2.7 /dist-packages /mininet-2.1.0p2-py2.7.egg /mininet /log.py
65 63 16 1 4% 3-159, 163-176, 161->163

```
/usr/local/lib/python2.7/dist-packages/mininet-2.1.0p2-py2.7.egg/mininet/moduledeps.py
45    42    24     1    9%   3-60, 65-68, 64->65

/usr/local/lib/python2.7/dist-packages/mininet-2.1.0p2-py2.7.egg/mininet/net.py          462
415    184     5    9%   89-110, 164-166, 178, 180-181, 191, 203-732, 742-810, 163->164,
177->178, 179->180, 184->186, 200->202

/usr /local /lib /python2.7 /dist-packages /mininet-2.1.0p2-py2.7.egg/mininet/node.py
619    575    184     2    6%   47-67, 99-112, 115-116, 143-546, 555, 561-754, 760-765, 769-
771, 777-948, 954-957, 967-1244, 114->115, 966->967

/usr /local /lib /python2.7 /dist-packages/mininet-2.1.0p2-py2.7.egg/mininet/util.py          304
216    106     11    28%   3-58, 76, 95, 107-115, 122-250, 260, 265, 277, 282, 291-358, 369-
370, 376, 383, 405-432, 438-439, 442-510, 516-517, 65->69, 67->69, 74->76, 94->95, 100-
>91, 102->91, 286->289, 361->364, 368->369, 379->-376, 515->516

---------------------------------------------------------------------------------------------------------

TOTAL                                                                    1495   1311    514    20    11%
```

Function tested

build(): The method to override in your topology class. Constructor parameters (n) will be passed through to it automatically by Topo.__init__().

addSwitch(): adds a switch to a topology and returns the switch name

addHost(): adds a host to a topology and returns the host name

addLink(): adds a bidirectional link to a topology (and returns a link key, but this is not important). Links in Mininet are bidirectional unless noted otherwise.

start(): starts your network

pingAll(): tests connectivity by trying to have all nodes ping each other

stop(): stops your network

net.hosts: all the hosts in a network

dumpNodeConnections(): dumps connections to/from a set of nodes.

References:

www.mininet.org

www.python.org

www.github.com/agroce/tstl