

CS 562 – Applied Software Engineering - Part 4

Eman Almadhoun – 932909951

Prof. Alex Groce

My library: topo (from mininet)

Python code: topo.py

This system is exist on GitHub website under: <https://github.com/mininet/mininet>

I have tested topo functions from mininet system using TSTL automation tools for testing. I have chosen this system because I want to understand this system more and how it works, as well as the correction of this system. Also, I am a graduate student in Networking and Computer System Group, so this project will help me in my research as well. I am interesting so much on networking area.

Mininet is a system which using for emulating a network, add controllers, add hosts, links and switches virtually which we can use it to do you experiments and give you excellent results. Also, it gives you the idea of how to create the topology with hundreds of nodes and how it works on your computers. Mininet is a useful tool for whose using OpenFlow and Software-Defined Networking (SDN). They can create a network, testing it and make demos. Mininet has written in Python language and provide Python API libraries. It helps to emulate Linux-based network system very fast. Mininet startup a virtual system in a few second building the controller, switches, with a help of Linux operating system inside Virtual Machine by using commands and specify how many nodes or switches need to create and what the type of the topology you want to create. It is a very powerful system. It used widely in development, research and teaching especially in SDN, and OpenFlow. Mininet is included in Ubuntu as the Mininet package and it exists under a liberal BSD-style.

Functions have tested:

I have tested the following functions in Topo class. I listed them below with their description of what each function does:

1. addNode() : to add node to the graph
2. addHost() : to add host to the graph
3. addSwitch() : to add switch to the graph
4. addLink() : to link node together
5. nodes() : return nodes name in graph
6. isSwitch() : return true if node is a switch
7. switches() : return switches name in graph
8. hosts() : return hosts name in graph
9. links() : return links
10. addPort() : generate port mapping for new edge

11. port() : get port numbers
12. linkInfo() : return link metadata dictionary
13. setlinkInfo() : set link metadata dictionary
14. nodeInfo() : return metadata dictionary for node
15. setNodeInfo() : set metadata dictionary for node
16. iterLinks() : return links (iterator)
17. _linkEntry() : return link entry and key

Some Issues found:

I test the system by I checked that if I added a node to the graph, the graph is not empty. Also, I assert that if I added switched in the graph, the graph is not empty. I checked that if I added a host to the graph, the host is added. Also, when I add switches, I assert that the added node is switches. Most of these function work fine without any issues. But, I found some issue in addPort(), addLink(), nodeInfo(), iterLinks() and linkEntry()

1. addPort(): I cannot add port before adding switches. The tester specified that source and destination switches should be added first. After adding switches, addPort() function works perfectly. addLink() function has same issue of addPort() but after add nodes, switches or hosts addLink() function works fine.
2. nodeInfo() function has an error which is “str object has no attribute get”.

```

mininet@mininet-vm: ~/tstl/generators
File Edit Tabs Help
Information
['S3']
['N1']
['S3']
['N1']
Information
Reduced test has 6 steps
REDUCED IN 0.0065960121155 SECONDS
stringl0 = "S3" # STEP 0
string0 = "N1" # STEP 1
topo0 = topo.Topo() # STEP 2
topo0.addNode(string0); print topo0.nodes() # STEP 3
topo0.addNode('N'); topo0.setNodeInfo('N1', "Informantion"); print topo0.nodeInfo('N1') # STEP 4
topo0.addSwitch(stringl0); print topo0.switches() # STEP 5

FINAL VERSION OF TEST, WITH LOGGED REPLAY:
stringl0 = "S3" # STEP 0
string0 = "N1" # STEP 1
topo0 = topo.Topo() # STEP 2
topo0.addNode(string0); print topo0.nodes() # STEP 3
['N1']
topo0.addNode('N'); topo0.setNodeInfo('N1', "Informantion"); print topo0.nodeInfo('N1') # STEP 4
Information
topo0.addSwitch(stringl0); print topo0.switches() # STEP 5
ERROR: (<type 'exceptions.AttributeError'>, AttributeError("'str' object has no attribute 'get'",), <traceback object at 0x7f6ea5f936c8>)
TRACEBACK:
  File "/home/mininet/tstl/generators/sut.py", line 875, in safely
    act[2]()
  File "/home/mininet/tstl/generators/sut.py", line 309, in act11
    self.p.topo[0].addSwitch(self.p.stringl[0]); print self.p.topo[0].switches()
  File "build/bdist.linux-x86_64/egg/mininet/topo.py", line 140, in switches
    return [n for n in self.nodes(sort) if self.isSwitch(n)]
  File "build/bdist.linux-x86_64/egg/mininet/topo.py", line 133, in isSwitch
    return info and info.get('isSwitch', False)
STOPPING TESTING DUE TO FAILED TEST
0.0 PERCENT COVERED

```

3. When I tested the iterLinks() and linkEntry() function, TSTL complains that Topo object has no attribute 'iterLinks', '_linkEntry'. Please see screen shots below:

```

stutter=None, running=False, nocover=False, gendepth=None, logging=None, html=None, keep=False, depth=100, timeout=20, output=None)
UNCAUGHT EXCEPTION
ERROR: (<type 'exceptions.AttributeError'>, AttributeError("'Topo' object has no attribute 'iterLinks'",), <traceback object at 0x7ff2ea6a1560>)
TRACEBACK:
  File "/home/mininet/tstl/generators/sut.py", line 977, in safely
    act[2]()
  File "/home/mininet/tstl/generators/sut.py", line 529, in act19
    self.p.topo[0].iterLinks()
Original test has 6 steps
REDUCING...
Reduced test has 2 steps
REDUCED IN 0.00499796867371 SECONDS
topo0 = topo.Topo()                                # STEP 0
topo0.iterLinks()                                  # STEP 1

FINAL VERSION OF TEST, WITH LOGGED REPLAY:
topo0 = topo.Topo()                                # STEP 0
topo0.iterLinks()                                  # STEP 1
ERROR: (<type 'exceptions.AttributeError'>, AttributeError("'Topo' object has no attribute 'iterLinks'",), <traceback object at 0x7ff2ea6a12d8>)
TRACEBACK:
  File "/home/mininet/tstl/generators/sut.py", line 977, in safely
    act[2]()
  File "/home/mininet/tstl/generators/sut.py", line 529, in act19
    self.p.topo[0].iterLinks()

```

```

mininet@mininet-vm: ~/tstl/generators
File Edit Tabs Help
Random testing using config=Config(verbose=False, failedLogging=None, maxtests=-1, greedyStutter=False, seed=None, generalize=False, uncaught=False, speed='FAST', internal=False,
, normalize=False, replayable=False, essentials=False, quickTests=False, coveragefile='coverage.out', ignoreprops=False, total=False, noreassign=False, full=False, multiple=False,
stutter=None, running=False, nocover=False, gendepth=None, logging=None, html=None, keep=False, depth=100, timeout=15, output=None)
['S3']
Switches added
[('h1', 'h2')]
Links added
['H3']
['H3']
UNCAUGHT EXCEPTION
ERROR: (<type 'exceptions.AttributeError'>, AttributeError("'Topo' object has no attribute '_linkEntry'",), <traceback object at 0x7fd70878ea28>)
TRACEBACK:
  File "/home/mininet/tstl/generators/sut.py", line 873, in safely
    act[2]()
  File "/home/mininet/tstl/generators/sut.py", line 449, in act16
    self.p.topo[0]._linkEntry()
Original test has 13 steps
REDUCING...
['S3']
['S3']
Reduced test has 2 steps
REDUCED IN 0.006460045166 SECONDS
topo0 = topo.Topo()                                # STEP 0
topo0._linkEntry()                                  # STEP 1

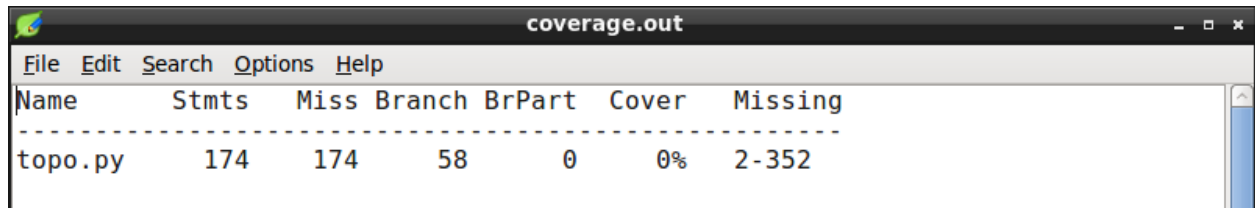
FINAL VERSION OF TEST, WITH LOGGED REPLAY:
topo0 = topo.Topo()                                # STEP 0
topo0._linkEntry()                                  # STEP 1
ERROR: (<type 'exceptions.AttributeError'>, AttributeError("'Topo' object has no attribute '_linkEntry'",), <traceback object at 0x7fd70878e878>)
TRACEBACK:
  File "/home/mininet/tstl/generators/sut.py", line 873, in safely
    act[2]()
  File "/home/mininet/tstl/generators/sut.py", line 449, in act16
    self.p.topo[0]._linkEntry()
STOPPING TESTING DUE TO FAILED TEST
0.0 PERCENT COVERED
0.0480909347534 TOTAL RUNTIME

```

Other than that, everything works perfectly, the topology is created. The tester works hard to fix any problem may the system face. It is a very important system and widely use to help researchers to do their experiments and get a perfect result.

Code coverage:

After all my tested I did, my code coverage still 0%. I think this happened because the python file I tested is so big 352 lines and has 6 classes on it and may be because the limitation of my test cases. Below the screen shot for my code coverage:



Name	Stmts	Miss	Branch	BrPart	Cover	Missing
topo.py	174	174	58	0	0%	2-352

How well des the tester work?

The tester works perfectly. Random-tester is amazing for finding bugs, analysis them and give us the location of the bugs with an explanation why these are bugs. It assigns every line a step number which helps us identify the bugs and find them easily. Also, random-tester can execute random test cases with random values assigned. It is a very powerful tester and well written.

Wrong with TSTL:

TSTL was so hard to understand at the start. This is because there is no textbook which explains its grammar and what is the meaning of them (log, pool, property, guard, REF) with some examples of how they use. But with the reading of Professor Alex' paper, reading his examples on GitHub and practice them, we can understand it more and get familiars of its syntax. Also, TSTL does not support other languages than Python which is better in the future if it will support Java or C for example.

Good with TSTL:

TSTL is a very simple tool to learn and easy to use because its grammar is so simple and basic. Using TSTL to test Software Under Test is so valuable and give us meaningful results. It can be reducing the number of steps and organized them through Normalization and Generalization.

During my project, I faced a problem which is my laptop hard drive corrupted before a week and a half from the project submission and needs to be repaired for about one week. So, after my laptop has fixed, I installed TSTL again. Some of my work have gone but I worked hard to complete as much as I can to submit my project on time.

At the end, I would like to thanks, Professor Alex for this helpful and amazing automated tool for testing which show us his creativity and intelligence. Also, I would like to thanks the TA, Hongyan for helping us a lot and give us a lot of her time. It was a new experience for me to learn testing with TSTL and I enjoyed learning and using it.

References:

- [1] <https://github.com/mininet/mininet>
- [2] <https://github.com/mininet/mininet/wiki/Ideas#background>
- [3] http://mininet.org/api/classmininet_1_1topo_1_1Topo.html