API Documentation

API Documentation

December 16, 2014

Contents

Contents 1

1 Package FlowSampTest 2
1.1 Modules 2
1.2 Variables 2

2 Package FlowSampTest.FlowSampRyu 3
2.1 Modules 3
2.2 Variables 3

3 Package FlowSampTest.FlowSampRyu.controller 4
3.1 Modules 4
3.2 Variables 4

4 Module FlowSampTest.FlowSampRyu.controller.feedback_analyser 5
4.1 Functions 5
4.2 Variables 5

5 Module FlowSampTest.FlowSampRyu.controller.flow_samp 6
5.1 Functions 6
5.2 Variables 6
5.3 Class FlowSamp 6
5.3.1 Methods 6
5.3.2 Class Variables 7

6 Module FlowSampTest.FlowSampRyu.controller.limit_parser 8
6.1 Functions 8
6.2 Variables 8

7 Package FlowSampTest.FlowSampRyu.monitor 9
7.1 Modules 9
7.2 Variables 9

8 Module FlowSampTest.FlowSampRyu.monitor.send_feedback 10
8.1 Functions 10
8.2 Variables 10

9 Module FlowSampTest.FlowSampRyu.monitor.utilisation 11

9.1 Functions 11
9.2 Variables 11

10 Module FlowSampTest.flow_samp_testbed 12
10.1 Functions 12

11 Module FlowSampTest.plotter 13
11.1 Functions 13
11.2 Variables 13

12 Module FlowSampTest.topology 14
12.1 Functions 14
12.2 Class TestTopo 14
12.2.1 Methods 14

1 Package FlowSampTest

1.1 Modules

- **FlowSampRyu** *(Section 2, p. 3)*
  - **controller** *(Section 3, p. 4)*
    - **feedback_analyser** *(Section 4, p. 5)*
    - **flow_samp** *(Section 5, p. 6)*
    - **limit_parser** *(Section 6, p. 8)*
  - **monitor** *(Section 7, p. 9)*
    - **send_feedback** *(Section 8, p. 10)*
    - **utilisation** *(Section 9, p. 11)*
- **flow_samp_testbed** *(Section 10, p. 12)*
- **plotter** *(Section 11, p. 13)*
- **topology** *(Section 12, p. 14)*

1.2 Variables

Name	Description
__package__	**Value:** `None`

2 Package FlowSampTest.FlowSampRyu

2.1 Modules

- **controller** *(Section 3, p. 4)*
  - **feedback_analyser** *(Section 4, p. 5)*
  - **flow_samp** *(Section 5, p. 6)*
  - **limit_parser** *(Section 6, p. 8)*
- **monitor** *(Section 7, p. 9)*
  - **send_feedback** *(Section 8, p. 10)*
  - **utilisation** *(Section 9, p. 11)*

2.2 Variables

Name	Description
__package__	**Value:** `None`

3 Package FlowSampTest.FlowSampRyu.controller

3.1 Modules

- **feedback_analyser** *(Section 4, p. 5)*
- **flow_samp** *(Section 5, p. 6)*
- **limit_parser** *(Section 6, p. 8)*

3.2 Variables

Name	Description
__package__	**Value:** `None`

4 Module FlowSampTest.FlowSampRyu.controller.feedback_analyser

4.1 Functions

adjust_accept_limit(*params*, *limits_config*=`'FlowSampRyu/controller/controller_config.ini'`, *soft_limit*=`0.9`)

Determines the accept limit for the flows to the monitor. Test for proposed idea. Algorithm Supplied in Adapdation.txt separately

4.2 Variables

Name	Description
HARD_MUL	**Value:** `2`
SOFT_MUL	**Value:** `1`
__package__	**Value:** `'FlowSampTest.FlowSampRyu.controller'`

5 Module FlowSampTest.FlowSampRyu.controller.flow_samp

5.1 Functions

hash_flow(*flow_string*)

Creates an MD5 hash for a particular flow string. Return only first 4 characters of the hash

5.2 Variables

Name	Description
PORT	**Value:** `12000`
ETHTYPE_IPV4	**Value:** `0x0800`
PLOT_LOG_FILE	**Value:** `'PlotLogs/values.log'`

5.3 Class FlowSamp

ryu.base.app_manager.RyuApp

FlowSampTest.FlowSampRyu.controller.flow_samp.FlowSamp

The Default Class For the Ryu Flow Samp Application Extends the simple learning switch provided in the Ryu Documentation https://github.com/osrg/ryu/blob/master/ryu/app/simple_switch.py Contains own extension for the Adaptaion in packet_in

5.3.1 Methods

__init__(*self*, **args*, ***kwargs*)

switch_features_handler(*self*, *ev*)

add_flow(*self*, *datapath*, *priority*, *match*, *actions*)

Add a particular flow @param datapath = router/switch @param priority = priority of the flow @param match = the rule differentiating the flow from the rest @action = usually decision if to be sent to monitor as well or not

build_flow_string(*self*, **args*)

Build a concatenated string from the various flow characteristics

flow_decision(*self*, *flow_string*)

Checks the new incoming flow and makes a decision based on last known monitor load.

update_accept_limit(*self*, *percentage*)

Change the monitor accept percentage to the argument

monitor_feedback_loop(*self*, *port*=`PORT`)

Listens to feedback from monitor Updates Accept Limit based on analysis

5.3.2 Class Variables

Name	Description
OFP_VERSIONS	**Value:** `[ofproto_v1_3.OFP_VERSION]`

6 Module FlowSampTest.FlowSampRyu.controller.limit_parser

6.1 Functions

limit_parser(*limits_file*)
Parse The Limits File and Return a list with the limits

6.2 Variables

Name	Description
__package__	**Value:** `'FlowSampTest.FlowSampRyu.controller'`

7 Package FlowSampTest.FlowSampRyu.monitor

7.1 Modules

- **send_feedback** *(Section 8, p. 10)*
- **utilisation** *(Section 9, p. 11)*

7.2 Variables

Name	Description
__package__	**Value:** `None`

8 Module FlowSampTest.FlowSampRyu.monitor.send_feedback

8.1 Functions

send_feedback(*sock*, *ip*, *port*, *interface*)

```
Build and Send Feedback to the Controller
@param sock = the socket (UDP) to use to send the feedback
@param ip = the ip of the controller
@param port = port on which the controller is listening
@param interface = the interface for which the stats should be
                   calculated
```

main()

The main function Add and parse the arguments. Create the UDP socket for connection with the controller Start the feedback loop

8.2 Variables

Name	Description
__package__	**Value:** `'FlowSampTest.FlowSampRyu.monitor'`

9 Module FlowSampTest.FlowSampRyu.monitor.utilisation

9.1 Functions

link_stats(*interface*)

Returns statistics about the interface utilization

9.2 Variables

Name	Description
__package__	**Value:** `'FlowSampTest.FlowSampRyu.monitor'`

10 Module FlowSampTest.flow_samp_testbed

10.1 Functions

launch()

```

Start The Main Testbed

Includes:
Starting mininet
Creating the topology
Provide initial configuration to nodes
Start actual testbed commands:
    Start the FlowSamp application on the Controller
    Start the Feedback loop on the monitor
    Replay a Pcap across the two nodes
    Start the Plotter
```

add_arguments(*parser*)

Add and Parse command Line Options

11 Module FlowSampTest.plotter

11.1 Functions

start_plotter(*plot_log_file*)

Starts an interactive plotter which plots figures for each parameter and the current accept limit

11.2 Variables

Name	Description
PARAM_LIST	**Value:** `['Bandwidth', 'Packet Count']`
PARAM_COUNT	**Value:** `3`
SOFT_LIMIT	**Value:** `0.9`

12 Module FlowSampTest.topology

12.1 Functions

configureRootConnection(*root*, *monitor*)

Configure Feedback link properly, different subnet Add Host Routes properly on both monitor and client

12.2 Class TestTopo

```
mininet.topo.Topo ─┐
                   FlowSampTest.topology.TestTopo
```

```
The Topology for the testbed:
Source------OFSwitch------Sink
             |    |
  Controller_|    |___Monitor
       |__________|
         Feedback Link
```

12.2.1 Methods

__init__(*self*)

Index

FlowSampTest *(package)*, 2
FlowSampTest.flow_samp_testbed *(module)*, 12
FlowSampTest.flow_samp_testbed.add_arguments *(function)*, 12
FlowSampTest.flow_samp_testbed.launch *(function)*, 12
FlowSampTest.FlowSampRyu *(package)*, 3
FlowSampTest.FlowSampRyu.controller *(package)*, 4
FlowSampTest.FlowSampRyu.monitor *(package)*, 9
FlowSampTest.plotter *(module)*, 13
FlowSampTest.plotter.start_plotter *(function)*, 13
FlowSampTest.topology *(module)*, 14
FlowSampTest.topology.configureRootConnection *(function)*, 14
FlowSampTest.topology.TestTopo *(class)*, 14