RDM Simulator User´s Guide

Requirements

This simulator uses the libraries *Matplotlib* and *SimPy*. Both can be installed with *pip* using, respectively:

```
python -m pip install -U matplotlib
python -m pip install simpy
```

Configuration

There are two .json files on the rdm_sim subpackage, these are used to configure the simulator according to the user's need. What follows is a quick overview of each file and the values on them.

- **configuration.json**: This .json file contains general configuration values for the simulation:

Parameter	Data type	Value range	Description
random_seed	null/int	All integers	Used to set a seed for the random values. Leave as *null* for the default random seed.
scenario	int	[0, 6]	The scenarios allow the simulator the emulate disturbances on the network. There are 7 selectable scenarios, which affect the simulator as below.
time_steps	int	Non negative integers	The amount of timesteps the program will run.
mirror_number	int	Non negative integers	The number of mirrors the simulated network will have.
mst_active_link	int	[0, 100]	Array that establishes a range from which the number of active links will be randomly taken when the MST topology is selected.
mst_bandwidth_consumption	int	[0, 100]	Establishes the range from which the bandwidth consumption will be randomly taken when the MST topology is selected.
mst_writing_time	int	[0, 100]	Establishes the range from which the time to write data will be randomly taken when the MST topology is selected.
rt_active_link	int	[0, 100]	Same as mst_active_links, for the RT topology.

- **Scenarios**:
    - 0: No disturbances, standard scenario
    - 1: Random changes on the execution of MST topology, adds a deviation on the active network links.
    - 2: Random changes on the execution of RT topology, adds deviation in the values of bandwidth consumption and time to write data.

    - 3: Simultaneous occurrence of scenario 1 and 2, meaning that when MST topology is selected there will be a deviation on active links and when RT is selected there will be a deviation on bandwidth consumption and writing times.
    - 4: Random changes on the execution of MST topology, adds deviation on all monitorable metrics.
    - 5: Random changes on the execution of RT topology, adds deviation on all monitorable metrics.
    - 6: Simultaneous occurrence of scenario 4 and 5, there will always be a random deviation on all monitorable metrics.
- **configuration_detrimental.json**: Contains configuration values specific for each of the scenarios. Each entrance on this file is made of an array of two elements, a lower and upper bound for the deviation that will be added

Architecture

rdm_sim subpackage

- **Config.py**: Here are defined functions that takes the data from the configuration *configuration.json* and *configuration_detrimental.json* files and loads it into the simulator.
- **Network Properties**: Here are defined the core characteristic of the network to be simulated, specifically the number of mirrors and the total number of links.
- **DetrimentalScenarios.py:** Here are defined functions that get a deviation for monitorable metrics, thus introducing some disturbance in the simulated network.
- **MonitorableMetrics.py**: Contains the functions that defines the values for the monitorable metrics: Active Links, Bandwidth Consumption and Time to write data.
- **Topology.py**: Contains the function that sets the Topology for a given timestep, and also functions that determine the impact of the topologies on the network by calling the functions in *MonitorableMetrics.py*.
- **NetworkManagement.py**: Defines the *probe()* and *effector()* functions. The *probe()* function gets the current topology of the network, as well as the monitorable metrics' values; and the *effector()* function, calls the Impact functions of *Topology.py*, based on the current topology.
- **Logger.py**: Defines the function used to log each run of the simulation in a .json file.

SimpleSimulationExample

- **MAPEKLoop.py**: Implements the Monitor-Analyze-Implement Loop, that actually enables the simulator to make decisions on runtime. Consists of four functions: *Monitor()* calls the *probe()* function to get the monitorable metrics values, *AnalyzeandPlan()* decides on a topology based on previous results and satisfaction thresholds for the monitorable metrics, *Execute()* calls the *effector()* function to set the topology, and *Run()* calls the previous three functions.
- **Main.py**: Comprised of two functions, *Simulator()* and *RunRDMSim()*. *Simulator()* runs the simulation by calling the *Run()* function of *MAPEKLoop.py*, prints the state for each timestep on console and gathers the data to plot after it has run. *RunRDMSim()* loads the configuration

using the *Config.py* functions, runs the simulation by calling Simulator*()*, calls the log function from *Logger.py* and finally displays the results on a plot.