Coordination Simulation Documentation

Release 0.9.1

Stefan Schneider

Jul 12, 2019

Contents:

CHAPTER 1

Introduction

This discrete-event flow-based simulator is a fast testbed for VNF coordination algorithms. It is designed to simulate small to medium sized networks with relatively close-to-reality accuracy. It can interact with coordination algorithms via an interface that can be adapted to by the developers of the algorithms.

The simulator is based on SimPy and is tested with Python 3.6.

Project owner: Stefan Schneider, MSc, University of Paderborn.

Developers: - Haydar Qarawlus, MSc. CS student, University of Paderborn. - Adnan Manzoor, MSc. CS student, University of Paderborn. - Sven Uthe, MSc, University of Paderbron.

CHAPTER 2

Application Structure

The file structure of the simulator is as follows: - docs (Folder): Contains the doumentation files of the project. - params (Folder): Contains sample parameter files (network file and VNF file) for testing purposes. - src (Folder): Contains the source code for the simulator and the interface. - coordsim (Folder): contains the source code of the simulator - metrics (Folder): contains the metrics module. - network (Folder): contains the network module. - reader (Folder): contains the params file reader module. - simulation (Folder): main simulator module. - main (Python): main executable for running the simulator from CLI - siminterface (Folder): contains the interface source code - tests (Folder): contains the unit tests for the simulator.

The main modules of the application are the following:

1. Simulation Module: This module contains FlowSimulator and SimulatorParams files. FlowSimulator is the main simulator that imports and utilizes the other modules. SimulatorParams is the class used to hold all the simulator parameters, this allows the parameters to be changed mid-simulation by the calling coordination algorithm.
2. Network Module: This module contains the Flow descriptor class. this class holds the main properties of the flow such as the data rate and the requested SFC, etc.
3. Reader Module: This module holds the network reader file that parses the parameter YAML and GraphML files that identify the network and the available SFCs and SFs.
4. Simulation Module: This module contains the main flow simulator. The flow simulator obtains its parameters from the SimulatotParams class. The params class is created by the calling coordination algorithm.

CHAPTER 3

Operation Flow

The simulator works as follows: The user (coordination algorithm or cli) provide two main inputs for the simulator: - Network file: GraphML file using the Zoo format. This file contains the network nodes and the edges. - VNF file: YAML file that includes the list of SFCs and the list of SFs under each SFC in an ordered manner. The file can also include a specifiecd placement that can be used as a default placement. The SFs must include a processing delay mean and standard deviation values so that processing delays can be calculated for each flow passing through that SF.

Once the parameters are provided, the flow of data through the simulator is as follows:

1. The input network and VNF files are parsed producing a NetworkX object containing the list of nodes and edges, and the shortest paths of the network (using Floyd-Warshall). The parsing also produces dictionaries that contain the list of SFCs and the list of SFs and their respective values. Additionally, the list of ingress nodes (nodes at which flows arrive) are also calculated from the GraphML file. These parameters are then passed to a SimulatorParams object, which holds all the parameters of the simulator, the simulator is then started using the FlowSimulator object's `start()` function
2. At each ingress node, the function `generate_flow()` is called as a SimPy process, this function creates `Flow` objects with exponentially distributed random inter arrival times. The flow's data rate and size are generated using normally distributed random variables. All of the inter arrival time, data rate, and flow size parameters are user configurable. The flow is also assigned a random SFC chosen from the list of available SFC given in the VNF file.
3. Once the flow is generated, `init_flow()` is called as a SimPy process which initializes the handling of the flow within the simulator. The function then calls `pass_flow()`, which then handles the scheduling of the flow according to the defined load balancing rules (flow schedule). Once the next node has been determined, the forwarding of the flow is simulated by halting the flow for the path delay duration using the `forward_flow()` function. Once that is done, the processing of the flow is simulated by calling `process_flow()` as a SimPy process. If the requested SF was not found at the next node, the flow is then dropped.
4. In `process_flow()`, the processing delay for that particular SF is generated using given mean and standard deviation values using a normal distribution. The simulator checks the node's remaining processing capacity to check if the node can handle the data rate requested by the SF, if there is not enough capacity, then the flow is dropped. For the duration that the flow is being processed by the SF, the flow's data rate is deducted from the node's capacity, and returned after the flow finished processing completely.

5. Once the flow was processed completely at each SF, `depart_flow()` is called to register the flow's departure from the network. If the flow still has other SFs to be processed at in the network, `process_flow()` calls `pass_flow()` again in a mutually recursive manner. This allows the flow to stay in the SF for processing, while the parts of the flow that were processed already to be sent to the next SF.

CHAPTER 4

Input Parameters

The available input parameters that are configurable by the user are: - d: The duration of the simulation (simulates milliseconds). - s: The seed to use for the random number generator. - n: The GraphML network file that specifies the nodes and edges of the network. - sf: VNF file which contains the SFCs and their respective SFs and their properties. - iam: Inter arrival mean of the flows' arrival at ingress nodes. - fdm: The mean value for the generation of data rate values for each flow. - fds: The standard deviation value for the generation of data rate values for each flow. - fss: The shape of the Pareto distribution for the generation of the flow size values.

CHAPTER 5

Running the simulator

The simulator application is called `coord-sim` To run the simulator, the following call may be executed:

```
coord-sim -d 20 -n params/networks/triangle.graphml -sf params/services/abc.yaml -c␣
↪params/config/sim_config.yaml
```

This will run the coord-sim simulator to simulate the given network and vnf parameter files for 10ms (environment time).

CHAPTER 6

Indices and tables

- genindex
- modindex
- search