BGPFSMSym WorkFlow plan

Mattia Milani*
*Dept. of Information Engineering and Computer Science, University of Trento, Italy
mattia.milani@studenti.unitn.it

I. Main Idea

The main idea of this tool is to emulate and log the FSM of BGP nodes described in [1]. The tool should have a double function, emulate a random evolution of the nodes grpah and the possibility to calculate all the possible states that a node can assume from an input message.

II. Goals

This tool should help us to model the fact that Minimum Route Advertisement Interval (MRAI) in Border Gateway Protocol (BGP) is essential to prevent the occurence of wedgies FiXme: insert citetion or persistant instability situations.

III. Inputs of the environment

The software will require two mandatory inputs:

- Graph, this file would describe the graph and how is structured. a more deep explanation of the graph is provided in Section VI;
- Environment descriptor, this file would describe the simulation environment, providing arguments to the simulator, this file is described more deeply in Section VII.

IV. Output of the environment

The software output would depend on the type of the simulation:

- Evolutional experiment this experiment would produce e simple CSV as output that can describe all the sates that a node assumes during the evolution.
- Compleate experiment, this type of experimetn goal is to describe all the possible states that nodes can assume, so the CSV presented as output will include all the states, even rare states that could happen.

V. External Libraries and their use

- ArgParse, used for the argument parsing;
- NetworkX, used for the graph/network handl;
- SimPy, could be used to simulate the entire environment.
- Pandas, data handling for easy manipulation of outputs in CSV
- ...

VI. Input Graph

The input of the graph is fundamental for the software, its mandatory. The graph is directed. The graph file format is GraphML, a markup language for graphs. Is possible to import and export graphml files from networks. This format should give us the maximum possible degree of freedom.

Arguments of nodes (in red mandatory arguments):

- NodeId: identifier of the node, would be treated as an integer, not integer Ids will raise an execption
- destination: in the future will be an ip prefix, for now it's a bool value that identify the node that will be reached from the others.

Arguments for the edges (in red mandatory arguments):

- delay: this argument will describe the delay that will be used for the communications on this link, is composed by two values min and max delay.

future arguments: implementation of destinations like ip address, with the possibility to have more ip addresses per node. Insertion of MRAI in the edge

VII. Input arguments

It's mandatory to give an environment descriptor. This file will be taken in consideration by the Argparse library to correctly set up all the simulation variables and inputs.

All the arguments that could be used are:

- graph, this argument describe the position of the graph file taken in consideration for the experiment
- experiment, this argument describe which type of experiment should be done, the possible values are "evolutional" "compleate"
- outputdir, this argument describe the directory that should be used as output for the CSV files, the if it does not exists will be created, if a directory with the same name already exists a new directory will be created
- verbose, this argument describe how much verbose should be the output of the program
- delaydistribution, this argument describe the distribution that should be taken in consideration for the delays on the edges
- defaultdelay, this argumnet set the default delay min and max will be overrided by the graph argument if present

VIII. Tests

Is possible to run the unit test that are present in the "test" folder to check if the software is working properly

IX. Templates

In the folder templates are present some templates experiments that could be useful.

X. Software main components

The main components of the software are:

- argument pareser, this component checks the arguments that are present in the environment file
- graph converter, this component is responsible to convert the graph passed as input in simpy components
- simulation environment, this is the main component for the simulation it takes as input the environment configuration and the components created for the simulation, is also responsible for the start and the stop of the simulation
- logger, this component is responsible for the inline output is possible to check the evolution of a simulation with the inline output
- CSV writer, this component is responsible for the output of the system, the output must be a CSV in the predefined format.

Parallel execution of experiments thanks to *parallel* [2].

References

[1] T. G. Griffin, "A Finite State Model Update Propagation for Hard-State Path-Vector Protocols."

[2] O. Tange, "Gnu parallel - the command-line power tool," *;login: The USENIX Magazine*, vol. 36, no. 1, pp. 42–47, Feb 2011. [Online]. Available: http://www.gnu.org/s/parallel