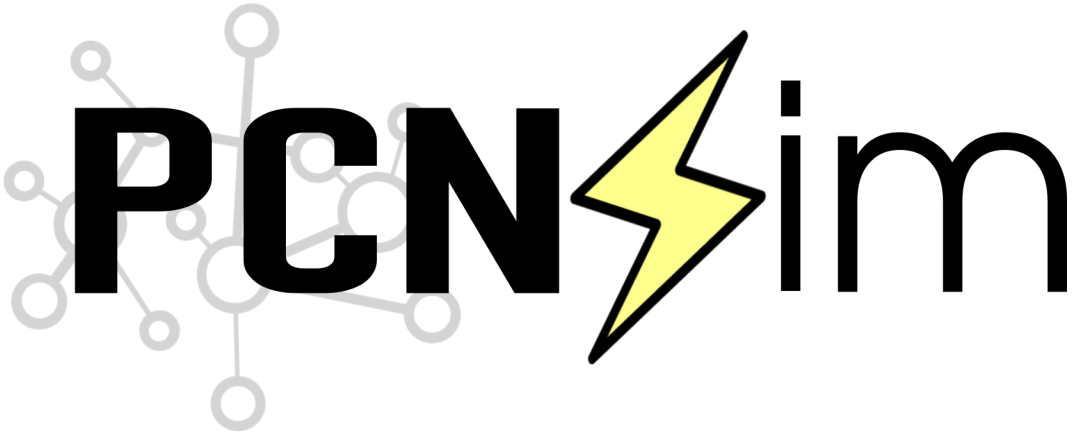
PCNsim

Gabriel Rebello and Gustavo Camilo

Mar 16, 2023

CONTENTS



A reliable and modular open-source payment channel network simulator based on the Lightning Network.

CHAPTER

ONE

GETTING STARTED

1.1 Prerequisites

Before using PCNsim simulator, you need the following programs installed. PCNsim is available for Linux OS and Windows.

Warning: Although PCNsim is available in Windows OS, this documentation follows the commands used in Linux. Some command might change depending on the operating system that you run.

1.1.1 Install Git

Make sure you have git installed on your device.

1.1.2 Install Python

PCNsim requires Python v3.8 to execute every module correctly. You can find every available Python version for download at the official Python website. If you already have Python 3 installed, you can verify the python version you have by executing the following command:

```
$ python3 --version
```

1.1.3 Install OMNET++

PCNsim runs over OMNET++ to deliver a user-friendly interface, modular, and reliable network simulation. You can find OMNET++ installation guide in the official OMNET++ website.

1.2 Installing PCNsim

After checking if you meet the correct requirements of PCNsim, you can install install PCNsim.

First, clone our Github respository:

```
$ git clone https://github.com/gfrebello/pcnsim
```

After cloning the repository, install the necessary libraries to run PCNsim:

```
$ cd pcnsim
$ pip install -r requirements.txt
```

You may want to create a environment variable that points to the pcnsim source directory:

```
$ export PCNSIM_DIR = $PWD
```

1.2.1 Installing the Dataset

As real-world transaction data about the Lightning Network is not available due to privacy, PCNsim uses a credit-card dataset to model transactions on the network simulator. As PCNs aim to offer a payment method as fast as current credit-card companies, we argue that credit-card transactions are a good fit to model transactions size. The credit-card dataset used to model transactions in PCNSim is available at Kaggle.

After downloading the creditcard dataset, move the csv file to the datasets folder:

```
$ mv creditcard.csv $PCNSIM_DIR/scripts/datasets
```

Note: If you are having problems installing PCNsim, you can contact our team. Our contact information is available at the official PCNsim website.

1.3 Testing Your PCNsim

After downloading and installing the required components to run PCNsim, you can test your simulator by following this documentation. Our test simulation delivers the following scenario:

- ten nodes disposed in a scale-free network topology;
- one hundred transactions issued in the network. It's worth mentioning that the transactions are only issued by the end-hosts to simulate the behavior of the Lightning Network accurately;
- the channel values follow real-world Lightning Network values;
- the transaction values follow credit-card transaction values collected from a dataset.

1.3.1 Creating Network Topology and Transaction Workload

The first step in running the simulation is determining which network topology PCNsim will use to run the payment channel network. PCNsim offers scale-free and small-world topology by default given researches show that the Lightning Network behaves as both. It is possible to implement other network topologies by implementing them with NetworkX or by defining them in the topology file. To build the scenario described in this documentation, from the *pcnsim* root directory, go to the scripts directory:

```
cd script
```

To generate the network topology described in this section of the documentation, run the *genTopo* command specifying *10* as the number of nodes and the channel modelling following the Lightning Network:

```
python3 generate_topology_workload.py genTopo -n 10 --lightning
```

This command will generate a file in the *topologies* directory. This file will be used by OMNET++ to establish the connections among the nodes and channel parameters. After generating the topology, you'll have to generate the transaction workload, which defines the characteristics of the transactions in the simulation. PCNsim offers transaction modelling following a real-world data from a credit-card company or an e-commerce sales dataset. You can also customize the workload by directly modifying the workload file. To generate the transaction workload of our scenario, run the `genWork`command specifying credit-card as the modelling reference and `100`as the number of transactions:

```
python3 generate_topology_workload.py genWork --n_payments 100 --credit-card
```

This command generates a file in the *workloads* directory.

Note: For more information on the commands and their options, check the *Commands Reference* section of the documentation.

1.3.2 Running the Simulation

After generating the topology and workload, you can run PCNsim by opening the project on OMNET++ and running the simulation.

Before starting to develop applications in PCNsim, check if you meet all the *Prerequisites* and programs installed on your device. You'll not be able to run PCNsim without the described requisites.

After installing all prerequisites, you can follow to *Testing Your PCNsim* and start working with PCNsim on your local device.

CHAPTER

TWO

COMMANDS REFERENCE

2.1 genTopo

2.1.1 Description

The genTopo command generates a network topology to be used in the payment channel network simulator. PCNsim offers network topology modeling following a small-world topology or a scale-free topology as the Lighnint Network presents behavior similar to both types.

2.1.2 genTopo

```
Usage: generate_topology_workload.py genTopo [OPTIONS]

  Generates a topology for the simulation

Options:
  -t, --topology [scale-free|barabasi-albert|watts-strogatz]
                                  Topology used in the simulation
  -n, --nodes INTEGER             Number of nodes in the topology
  --alpha FLOAT                   Alpha parameter for scale-free topology
  --beta FLOAT                    Beta parameter for scale-free topology
  --gamma FLOAT                   Gamma parameter for scale-free topology
  -k INTEGER                      K parameter for Watts-Strogatz graph
  -p FLOAT                        P parameter for Watts-Strogatz graph
  -m INTEGER                      M parameter for Barabasi-Albert graph
  --lightning                     Channel capacities are modeled following
                                  real-world lightning network channels

  --help                          Show this message and exit.
```

2.1.3 Default Values

- `-t, --topology`: Type of network topology that will be used to run the PCN simulation. We offer 3 types of network topology: scale-free, Barabasi-Albert, and Watts-Strogatz. It's worth noting that Barabasi-Albert and Watts-Strogatz graph model are small-world topology. The user may choose the network topology by selecting the topology option followed by the string `"scale-free"`, `"barabasi-albert"`, or `"watts-strogatz"`, depending on the type of topology the the user choses.
  - *Default value:* `"scale-free"`
- `-n, --nodes`: Number of nodes that make up the network topology. This value has to be an **integer**.
  - *Default value:* `10`
- `--alpha`: Parameter alpha used to generate a scale-free network topology. This parameter is only used if the `scale-free` topology is chosen to model the network. Alpha is a **float** type parameter.
  - *Default value:* `0.5`
- `--beta`: Parameter beta used to generate a scale-free network topology. This parameter is only used if the `scale-free` topology is chosen to model the network. Beta is a **float** type parameter.
  - *Default value:* `1e-05`
- `--gamma`: Parameter gamma used to generate a scale-free network topology. This parameter is only used if the `scale-free` topology is chosen to model the network. It's worth noting that `alpha + gamma = 1`. Gamma is a **float** type parameter.
  - *Default value:* `0.49999`
- `--k`: Parameter k used to generate a scale-free network topology. This parameter is only used if the `watts-strogatz` topology is chosen to model the network. K is an **integer** type parameter.
  - *Default value:* `2`

2.1.4 Example Usage

- Model channels following the real-world Lightning Network channel capacity. It's worth mentioning that this options **does not** model the network topology following the Lightning Network topology, only its channel parameters. The network topology follows a scale-free network model.

```
$ python3 generate_topology_workload.py genTopo --lightning

Setting topology to scale-free
Setting n to 10
Setting alpha to 0.5
Setting beta to 1e-05
Setting gamma to 0.49999
```

2.2 genWork

2.2.1 Description

The `genWork` command generates a transaction workload to be used in the payment channel network simulator. PCNsim offers genWork modeling following a real-world data from a credit-card and an e-commerce dataset as the Lighnint Network does not disclose transaction information due to privacy issues.

2.2.2 genWork

```
Usage: generate_topology_workload.py genWork [OPTIONS]

  Generates a payment workload for the simulation

Options:
  --n_payments INTEGER   Number of payments in the network simulation
  --min_payment FLOAT    Minimum value of a payment in the network
  --max_payment INTEGER  Maximum value of a payment in the network
  --any_node             Transactions are issued by any node in the network,
                         not only end hosts

  --credit_card          Transactions are modeled following a credit card
                         dataset

  --e_commerce           Transactions are modeled following a e-commerce
                         dataset

  --help                 Show this message and exit.
```

2.2.3 Default Values

- `--n_payments`: Number of payments in the PCN simulation. The selected number of payments will be issued by a randomly selected node to another randomly selected node. This value as to be an **integer**.
  - *Default value:* `1`
- `--min_payments`: Minimum value of a payment in the PCN simulation. This value has to be an **float**.
  - *Default value:* `0.1`
- `--max_payments`: Maximum value of a payment in the PCN simulation. This value has to be an **float**.
  - *Default value:* `1`

2.2.4 Flags

- `--any_node`: By default, PCNsim restricts the act of issuing a transaction to end-hosts. Therefore, core nodes act only as intermediary, forwarding trasactions, but not issuing. This flag removes this restriction and includes core nodes in the act of issuing transactions.
- `--credit_card`: This flag makes PCNsim model transaction values following a credit-card dataset.
- `--e_commerce`: This flag makes PCNsim model transaction values following an e-commerce dataset.

2.2.5 Example Usage

Note: If you still have questions about PCNsim not covered by this documentation, please visit https://gta.ufrj.br/pcnsim webpage to find contact information for additional help.
