Contents

1 Installation **1**
1.1 Getting the right files 1
1.2 Setting up dependencies 1

2 Navigating the code **1**
2.1 config.py 2
2.2 network.py 2
2.3 run.py 2

1 Installation

The following are the steps to successfuly install and run PyNet.

1.1 Getting the right files

Installing the git package requires git to be installed in your system. Firstly, clone the git repository from the URL provided below: `https://github.com/amirinia/pynet2.git`

Example: git clone <URL>

To ensure that you are working on the latest version of the software, please make sure you are using the separation branch. To do this, you may use the following command:

git checkout sepration

1.2 Setting up dependencies

Once the directory has been initialized, you may navigate within it. To ensure that your first run of the simulator proceeds smoothly, please install the latest version of python and all the dependencies listed in "requirements.txt".

Example: pip install <module>

2 Navigating the code

Once you have completed the aforementioned steps, you may proceed with execution of the code. Using your favourite editor or the terminal, run the "run.py" file. This will run PyNet with the default configurations. Let us take a look at how one can modify the default settings.

2.1 config.py

Below is an explanation of the various parameters defined in this file. Most of the macros are self explanatory and fall under different sections demarcated by the use of python comments.

1. Area Definition defines the area of the canvas that includes the different motes.
2. Superframe section defines the unique characteristics of the superframe which is closely tied with the 802.15.4 MAC. The CSMA and TDMA duration and therefore, their ratio within the active period of the superframe are defined here. The inactive duration is also defined.
3. Energy section defines the initial energy of the individual motes as well as the power characteristics of the transceiver module.
4. TX_{RANGE} defines the maximum transmission distance and is crucial for interference detection.

2.2 network.py

This file is used to add nodes into the simulation while working on the command line. Note that there is a GUI method to achieve the same result. The 'env' variable contains a reference to the running simpy environment which is used throughout the code. Note that there must be at least two nodes in a network, one being the base station and the other being a regular node. By default, the node with id = 0 is defined as the base station. For instance:

```
net1.add_node(Node(0, env, 4, (config.xsize)/2, (config.ysize)/2,
node_type='B' ,power_type=0,ieee802154 =net1))
```

The aforementioned method can be called to install further nodes. The arguments for the node constructor can be found in the file named 'node.py'. The main parameters the constructor takes are the node ID, the X and Y coordinates and the instance of the network they belong to. The lines below the definition of the individual nodes relate to the GUI and may be skipped for the purposes of this tutorial.

2.3 run.py

The code laid out in this file is mostly self explanatory. The variable start-static determines whether the simulator will use the previously configured

'network.py' or the GUI initialization. You may try out both and use the one suited for your application.

The next step is to select a clustering algorithm to organize the nodes. By default, LEACH-C is used whereas PyNet also supports the use of K-MEANS clustering. Information on both can be found in the developer docs.

Lastly, we can pass the macro $\text{MAX}_{\text{RUNTIME}}$ as a paramenter to the env.run() function. This is the function that starts the simpy simulator and is the most crucial step.