# API Documentation

API Documentation

May 11, 2012

## Contents

**Contents** **1**

**1 Module broadCast** **2**
1.1 Functions . . . 2
1.2 Variables . . . 2

**2 Module main** **3**
2.1 Variables . . . 3

**3 Module olsr** **5**
3.1 Functions . . . 5
3.2 Variables . . . 6

**Index** **7**

# 1 Module broadCast

```
def setup(node_id, addr, isMPR, mypeer):

    This method takes the arguments,
    1. addr: The address to which the calling process binds itself to.
    2. isMPR: It is a boolean value. It is True when the calling process [node] is a Multi Point Relay o
    3. node_id: Unique Identifier of the calling process[node] a.k.a receiver
    4. mypeer: It is a list of one-hop neighbors of the calling process[node].

def source_setup(node_id, addr, MPRset, oneHop):

    This method takes the arguments,
    1. node_id: Unique Identifier of the calling process[node] a.k.a receiver
    2. addr: The address to which the calling process binds itself to.
    3. MPRset: applicable only to the source node, its set of Multi Point relays.
    4. oneHop: applicable only to the source node, its set of one-hop neighbors who are not its Multi Po

def startBroadcast(MPRset, i, allNeighbor):

    This method takes the arguments,
    1. MPRset: multi point relay set of the source node
    2. i: index of the node
    3. allNeighbor: It is a 2-D list, where each 1-D list in this 2-D list represents the one-hop neighb
       e.g. allNeighbor[3] is a 1-D list of one-hop neighbor of the 4th[only in the list] one-hop neighb
```

## 1.1 Functions

**setup**(*node_id, addr, isMPR, mypeer*)

**source_setup**(*node_id, addr, MPRset, oneHop*)

**startBroadcast**(*MPRset, i, allNeighbor*)

## 1.2 Variables

| Name | Description |
|---|---|
| __package__ | **Value:** `None` |

# 2 Module main

This is the main program. upon executing ”python main.py” on the console, this program, reads the first line from the input file named ’input’. The first line must be the number of processes [nodes] in the network. This is needed to initialize the 2-D list of size n*n [n - number of nodes], which contains a mapping of links between different nodes. Next, ths control enters an infinite loop, in which the ’edge’ information between two nodes is read, the control breaks out of the loop when an EOF is encountered in the file ’input’. the link information is stored in the ’edges’ list. From here, the method getMPR is called , it exists in the olsr API file olsr.py.

sample input file —

7

0,1

0,2

0,3

0,4

1,5

1,6

2,4

3,5

As we can see, the first line contains 7, which is the number of nodes in the network. the following lines after this give the information about the links between nodes. e.g. take 0,1 means that there is a link between node0 and node1.

## 2.1 Variables

| Name | Description |
|---|---|
| thr | **Value:** `19` |
| nodes | **Value:** `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]` |
| edges | **Value:** `[[0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...` |
| f | **Value:** `open("input", ’r’)` |
| p | **Value:** `19` |
| neighborset | **Value:** `[[], [18, 7], [8, 9, 10], [11, 12], [13, 14], [16, 15], [...` |
| i | **Value:** `0` |
| Set | **Value:** `[[], [18, 7], [8, 9, 10], [11, 12], [13, 14], [16, 15], [...` |
| __package__ | **Value:** `None` |
| a | **Value:** `6` |
| b | **Value:** `18` |

*continued on next page*

| Name | Description |
|---|---|
| each | **Value:** `[0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]` |
| j | **Value:** `19` |
| line | **Value:** `''` |
| temp | **Value:** `['6', '18']` |
| temp1 | **Value:** `[1, 6]` |
| x | **Value:** `0` |

# 3 Module olsr

```
This is the API file for my OLSR protocol implementation. The methods defined in this file are as follow

def all_indices(value, qlist):

     This method takes the arguments value and a list qlist. it returns the list containing the indices

def receive(addr, isMPR, i, mypeer, source):

     This method takes the arguments,
    1. addr: The address to which the calling process binds itself to.
    2. isMPR: It is a boolean value. It is True when the calling process [node] is a Multi Point Relay o
    3. i: Unique Identifier of the calling process[node] a.k.a receiver
    4. mypeer: It is a list of one-hop neighbors of the calling process[node].
    5. source: It is a boolean value, True only for the source node.

def sendd(mesg, isMPR, worker_addr, me):

     This method takes the arguments,
    1. mesg: It is a string, representing the message to be sent.
    2. isMPR: It is a boolean value. It is True when the calling process [node] is a Multi Point Relay o
    3. worker_addr: It is the receiver[IP+Port number] to whom the message is to be sent.
    4. me: Unique Identifier of the calling process[node] a.k.a sender

def GetMPR(allNeighbor, thr, i):

     This method takes the arguments,
    1. allNeighbor: It is a 2-D list, where each 1-D list in this 2-D list represents the one-hop neighb
       e.g. allNeighbor[3] is a 1-D list of one-hop neighbor of the 4th[only in the list] one-hop neigh
    2. thr: It is the number of processes [nodes].
    3. i: the Identifier of the node whose Multi Point Relays are to be found.


zmq - zeromq, The socket library that provides a API framework for sending messages in IPC. for more inf
```

## 3.1 Functions

**all_indices**(*value*, *qlist*)

**receive**(*addr*, *isMPR*, *i*, *mypeer*, *source*)

**sendd**(*mesg*, *isMPR*, *worker_addr*, *me*)

**GetMPR**(*allNeighbor*, *thr*, *i*)

## 3.2 Variables

| Name | Description |
|---|---|
| __package__ | **Value:** `None` |

# Index

broadCast *(module)*, 2
    broadCast.setup *(function)*, 2
    broadCast.source_setup *(function)*, 2
    broadCast.startBroadcast *(function)*, 2

main *(module)*, 3–4

olsr *(module)*, 5–6
    olsr.all_indices *(function)*, 5
    olsr.GetMPR *(function)*, 5
    olsr.receive *(function)*, 5
    olsr.sendd *(function)*, 5