Bogazici University, Department of Computer Engineering,
CMPE 524 Fall 2016

Energy Efficient Routing in Wireless Sensor Networks
Code Manual

Nurefşan Sertbaş, 2016700132

December 14, 2016

Note that you can access the code using https://github.com/sertbasn1/WSNProject.git.

1 Platform and Tool Specification

The written codes should run over Matlab-R2015b otherwise, some of the functions or declarations may not be supported.

Note: We have used Mac OSX 2,9 GHz Intel Core i5, 16 GB 1867 MHz DDR3 machine as a processing hardware.

2 Basics

Put the project folder named "WSNProject" into the Matlab working directory and go that directory from the Matlab addressing field.
The code files are listed :

1. CheckCoverage.m
2. checkMatrix.m
3. checkNodeStatus.m

4. checktreshold.m
5. Consumedenergy.m
6. costbasedgraph.m
7. createtopology.m
8. dijkstra.m
9. listdijkstra.m
10. main.m
11. main2.m
12. printNet.m
13. setup.m
14. simNet.m
15. simulation.m
16. updateNodeStatus.m

3 Running an Example

- Run Setup.m file

  In order to set the system /algorithm variables you should edit Setup.m file. You should run it as following therefore system parameters are set and stored in Matlab work space.

```
>> [x,y,N,receiver,sense_range,topology,Einit,Etx,Erx,treshold,termRatio,deadTresh,out] = Setup();
>>
```

Figure 1: 1st step: set variables

- Run the routing algorithm

  In order to run the proposed heuristic use Fig. 2. Otherwise, use Fig. 3. to run default Dijkstra shortest path algorithm.

```
>> [EofNodes,fij,graph]=main(x,y,N,receiver,sense_range,topology,Einit,Etx,Erx,treshold,termRatio,deadTresh,out);
```

Figure 2: 2nd step(1): run the proposed algorithm

```
>> [EofNodes,fij,graph]=main2(x,y,N,receiver,sense_range,topology,Einit,Etx,Erx,treshold,termRatio,deadTresh,out);
```

Figure 3: 2nd step(2): run the Dijkstra algorithm

Note that in order to compare their performances follow above steps respectively. Do not use clear command use delete variables in Matlab work-space.

3.1 Sample Output

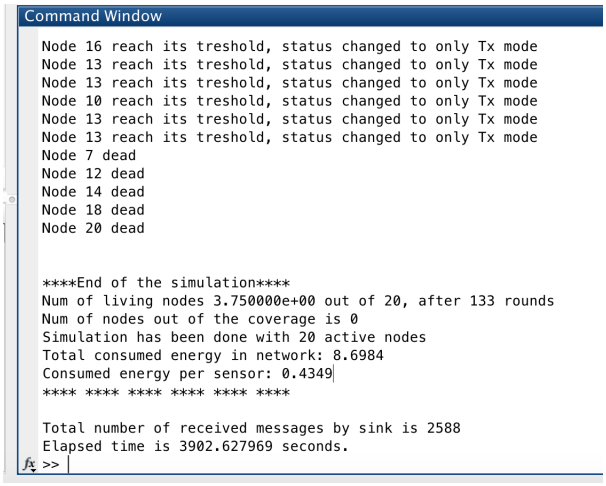

Figure 4: Sample execution of the algorithm-Output1

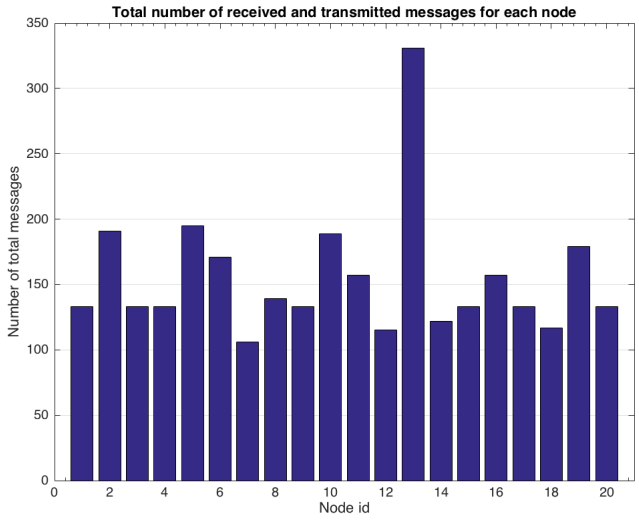

Figure 5: Sample execution of the algorithm-Output2

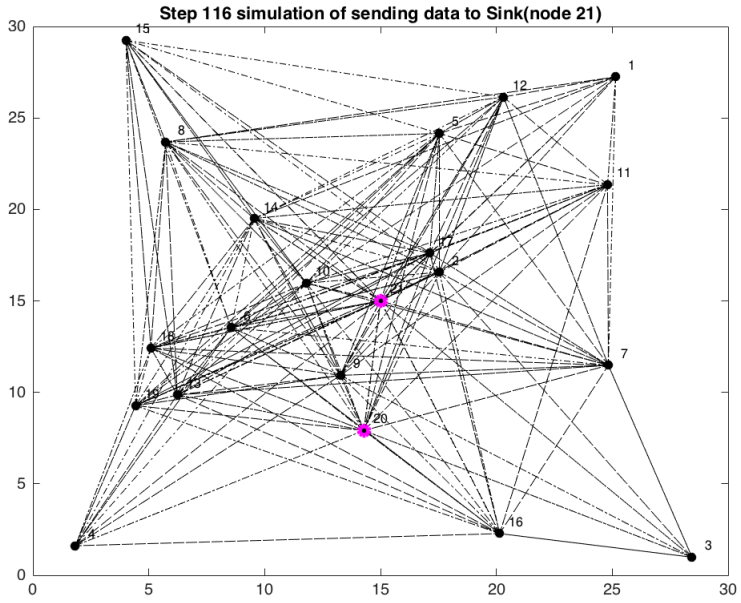

Figure 6: Sample execution of the algorithm-Output3

4 Explanation of the variables

```
N          %Number of sensor nodes+1(for sink)
receiver %id of the  sink, always N
x ,y %Grid lengths
sense_range      %Sensing range of the sensor in meter
topology %Holds the location information of the nodes and
    sink
Einit   %Initial energy of the sensor node
Etx  %Tx energy per  packet
Erx       %Rx energy per  packet
treshold %battery threshold to reject
termRatio%coverage constraint to finish the simulation
deadTresh% treshold to evaluate node as dead
out       %number of nodes that has no any neighbor
EofNodes          %Residual energy of the nodes
fij               %number of transmitted and received
   messages of each node and sink
dgraph %distance based graph for all  nodes
cgraph %energy related metric based graph for all  nodes
```

5 Explanation of the codes

5.1 Setup.m

```
function [x,y,N,receiver,sense_range,topology,Einit,Etx,
    Erx,treshold,termRatio,deadTresh,out] = Setup()
```

- It does not take any input.
- It specifies the parameters (N, x, y) and calls *createtopology*(N, x, y) function to create the topology.
- It initializes the system and algorithm parameters

```
%% parameter initialization
Einit=0.5;%joule
Etx=2*power(10,-5);%joule/packet
Erx=power(10,-5);%joule/packet
treshold=60;
termRatio=80;
deadTresh=3*power(10,-5);
```

- Returns with the parameters and topology (Saves them to the Matlab workspace.)

5.2 createtopology.m

```
function nodes = create_topology(N,x,y)
```

- It takes N,x and y as an input
- Creates x*y field
- Creates N-1 node randomly on that field
- Create sink and place it to the center of the field
- Returns with the location information of the nodes and sink

5.3 main.m

It is used for running energy efficient routing heuristic.

```
function [EofNodes,fij,graph]=main(x,y,N,receiver,
    sense_range,topology,Einit,Etx,Erx,treshold,termRatio,
    deadTresh,out)
```

- It takes outputs of the setup.m function as an input
- Creates fij and Eofnodes variables and initializes them
- Calls $printNet(sense_range, topology, x, y)$ function to visualize the network
- Determines the neighbors of all nodes and calculates the distance between them, creates variable named *dgraph*
- By using dgraph and $costbasedgraph(nofneighbors, cgraph, dgraph, EofNodes, N)$ function, creates variable named *cgraph*
- Starts the simulation, simulate rounds until the coverage criteria violated
- Calls $simulation(i, topology, cgraph, sense_range, x, y, receiver, adj, N, dgraph)$ function to simulate one round of network
- It updates cgraph (dynamically changing edges) and updates residual energy of the nodes
- Checks the termination criteria
- Prints the results to the screen at the end of the program

5.4 main2.m

It is used for running shortest path routing.

```
function [EofNodes, fij, graph]=main2(x,y,N,receiver,
    sense_range,topology,Einit,Etx,Erx,treshold,termRatio,
    deadTresh,out)
```

All of the operations same as main.m. The only difference is edge metric therefore, it uses dgraph instead of cgraph.

5.5 dijkstra.m, checkMatrix.m, listdijkstra.m

All of these functions are used to implement default shortest path algorithm.

```
function L = listdijkstra(L,W,s,d);
function [cost,path] = dijkstra(graph,s,d);
function [m,n,newE] = checkMatrix(adj) ;
```

5.6 CheckCoverage.m

```
function ratio = CheckCoverage(topology,N);
```

- It returns with the coverage ratio of the current network status by using number of nodes and number of dead nodes

5.7 checkNodeStatus.m

```
function topology=checkNodeStatus(deadTresh,topology,
    EofNodes,N)
```

- It finds the nodes reach to the deadTresh therefore, they can be marked as a dead node. It writes this information to the 4th row of the topology variable.

5.8 checktreshold.m

```
function [ status ] = checktreshold( Einit,EofNodes,nodes
    ,treshold )
```

- It returns 1 if the node reaches the treshold. Up to that point, it just sends its own data notaccept any data to forward. Otherwise, returns with 0.

5.9 ConsumedEnergy.m

```
function EofNodes = ConsumedEnergy(Einit,deadTresh,
    EofNodes,topology,N,active)
```

- It returns total consumed energy in the network by examining each node individually.

5.10 costbasedgraph.m

```
function cgraph=costbasedgraph(nofneighbors,cgraph,dgraph
    ,EofNodes,N)
```

- It returns with cgraph. It uses the new metric formulation given in the proposed algorithm to create graph.

5.11 printNet.m

```
function adj=printNet(range,nodes,x,y)
```

- It visualizes the network topology

5.12 simNet.m

```
1 function simNet(round,R,topology,x,y,N)
```

- It visualizes the network simulation by specifying which sensor send data through the which route

5.13 updateNodeStatus.m

```
1 function [cgraph]=updateNodeStatus(cgraph,topology,N)
```

- Dead nodes are already marked before. It just deleted the edges between dead nodes and their adjacent s. Therefore, it returns with the updated graph.