**ISTANBUL TECHNICAL UNIVERSITY**



**Computer Engineering Department**

**BLG 566E Digital Solutions for Smart Cities**

**Instructors:**

**Sema Fatma Oktuğ**

**Yusuf Yaslan**

**Spring 2018 Term Project Report**

**Cooperative Communication Topologies for Smart City IoT Applications**

|  |  |
| --- | --- |
| **Sinan Atan**  **504162306** | **Tuğrul Yatağan**  **504161551** |

Table of Contents

[1. Introduction 3](#_Toc516011267)

[2. Problem Description 3](#_Toc516011268)

[3. Proposed Framework 4](#_Toc516011269)

[4. Results 5](#_Toc516011270)

[5. Conclusion 6](#_Toc516011271)

[6. Development and Modelling 6](#_Toc516011272)

[6.1. Model Data Generation 6](#_Toc516011273)

[6.1.1. Example Generated Data 7](#_Toc516011274)

[6.1.2. Proposed Model Data Generator Source Code 8](#_Toc516011275)

[6.2. Weka Machine Learning Tool 8](#_Toc516011276)

[6.2.1. Weka Example Screenshot 8](#_Toc516011277)

[6.2.2. Weka arff File Explorer Screenshot 9](#_Toc516011278)

[6.2.3. Weka Input Selector Screenshot 10](#_Toc516011279)

[6.2.4. Weka Classification Result Screenshot 10](#_Toc516011280)

[7. References 11](#_Toc516011281)

[8. Appendix 11](#_Toc516011282)

[Matlab Codes 11](#_Toc516011283)

# Introduction

Smart cities are expected to provide their residents a variety of innovative services and applications using advanced technologies. These technologies are heavily influenced by recent communication methods. Wireless communication can provide flexibility and mobility to any smart city IoT application thus wireless connectivity has become more important to provide smart city IoT applications. Example smart city IoT applications which extensively need wireless communication can be; waste management, water/electricity/gas metering, environment/animal tracking, water/gas/fire detecting, and so on.

Wireless connection is expected to be reliable, efficient, scalable, fair, high throughput, low delay and green power in case of system design principles. It can be difficult to satisfy all these expectations due to the nature of wireless communication and behavior of wireless network entities. Mobile node which has low signal strength to base station may face drastic communication issues. To cope with this issue, relay nodes can be used to provide alternative communication path. This scheme is called as cooperative communication.

In this work, we propose a cooperative communication framework for smart city IoT applications which utilize machine learning methods to optimize relay node allocation policy.

# Problem Description

Wireless communication reliability is severely affected by propagation medium and congestion. Either wireless communication throughput will decrease, or power consumption will increase if a network entity does not have a good communication path. If a wireless network entity tries to communicate through a longer path or through a heavily loaded (congested) node, not only its performance but the overall network performance will decrease due to the half-duplex nature of wireless communication.

Most of the wireless communication capable nodes are mobile, this makes it even more difficult to provide reliable wireless link between end nodes and base station. Thus, unpredictable channel state of mobility is the trade-off for wireless systems against flexibility.

In a common wireless smart city IoT application characteristics are;

* Most of the entities are low power or mobile end devices: animal trackers, waste management devices, water/gas/fire sensors, water/electricity/gas meters, etc.
* There are decent amount of fixed end devices which connected to power: traffic lights, street lamps, surveillance cameras, etc.
* There is at least one base station which provides internet access to other entities.

Using these fixed end devices as relay nodes can improve overall network connectivity, reliability, efficiency, throughput, delay and jitter. Mobile end nodes are not aware of the network topology thus they tend to select nearest relay node to reach base station. But selecting nearest relay node may not be the optimum solution for a mobile end node.

# Proposed Framework

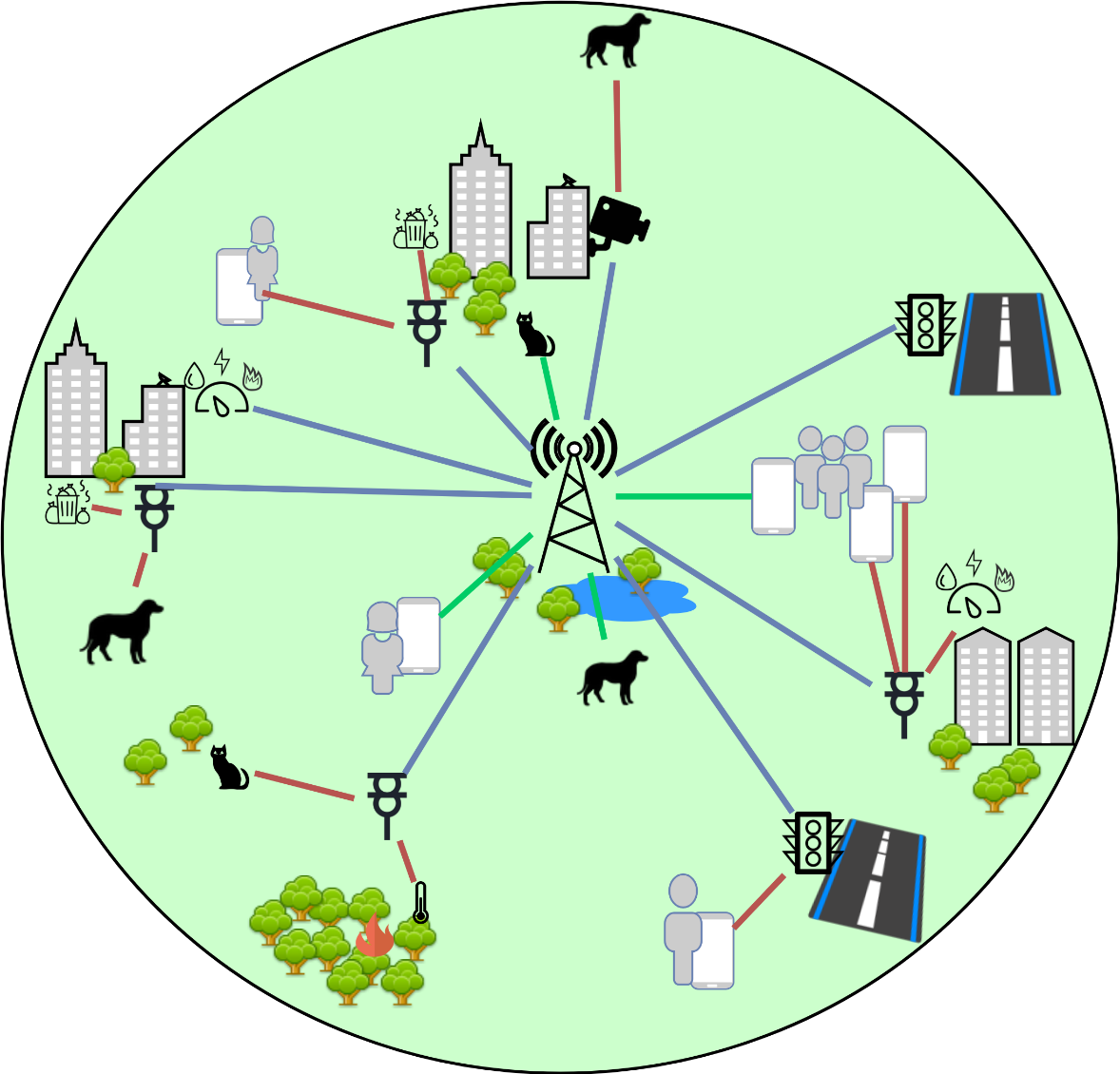
In our framework, we introduce a cooperation framework which use historic data as input and by the help of machine learning, assignment of cooperation relay is done for end nodes. This achieves better utilization of power by avoiding extra wireless signaling for resource allocation. Furthermore, when an end node is not in the coverage area of the base station, assignment of cooperation fails in classical methods. Our scheme eliminates those issues as well.

We utilize fixed relaying nodes for cooperative communication. Scale of fixed nodes in a common smart city application should be sufficient to cover most of the mobile end nodes. Furthermore, mobile end nodes should not waste their limited energy resources since they wouldn't continuously listen medium to participate wireless cooperation. We create a simple model for a common smart city application containing; one base station, N fixed relays, MUE end-user and MIoT sensor mobile end nodes. Proposed smart city application framework topology is illustrated in Figure 1.

In this architecture, we modelled every wireless node with following parameters;

* Time [0-23]: Time information is used as input, since usage and mobility characteristics of nodes change during the day.
* Node ID [0-(N+MUE+MIoT+1)]: Unique identity of each node in the system. Cooperation pattern of each individual node in the network is tracked by node ID.
* Type {base, relay, end-user, IoT device}: Node type illustrates the category of each device to do a better classification for each node in the network in terms of mobility, usage, etc.
* Capacity [0-100]: Maximum throughput, towards base station, can be achieved by each node in a certain state. This is derived from underlying radio access technology capacity, but it reflects instantaneous value in specified condition.
* Rate [0-100]: Utilized throughput by each node. Rate is less than or equal to capacity for any node in the network.
* Distance to base station [0-5000]: Channel quality and signal strength are measured with distance factor from base station. This provides rough estimation of instantaneous capacity for end points.
* Distance to relay nodes [0-5000]: Similar to distance to base station, rather gives channel status information for each relay.
* Selected relay [0-1000]: Output parameter of classification, which gives the selection for each end node to cooperate. Each end node tries to select best relay for itself.

Base station and relay nodes can be able to obtain this data during daily operation. From this obtained data, base station can train a model for every end node. Base station should share this trained model with end nodes daily. Then, every end node can be able to select the best possible relay for cooperative communication. End nodes can select relay nodes offline. They don’t need to know current network topology, or they don’t require any external information.



**Figure 1.** Architectural decomposition of proposed cooperative communication topology.

# Results

In this work, we use Weka tool to analyze data set and performance of three different machine learning algorithms are surveyed; NaiveBayes, IBk and J48 (C4.5). We try to select the best algorithm. Processing power is not a constraint as calculations are done in base station. Results are fetched daily to end nodes as trained model to use for their future selections. Any failure in relay selection would lead to duplicated signaling which impacts the system performance.

We applied some assumptions to simplify our model;

* Only relay nodes are considered to cooperate with end points; cooperation between end points is not considered in the scope of this work.
* Only end points with lower capacity than required throughput is expected to cooperate.
* Clients are served as first in first out policy; no end nodes do a relay selection for speculated utilization by different end nodes in the future. End nodes only utilize single relay at a time.

Data set includes 1 base station, 5 fixed relaying nodes, 50 mobile end-user nodes and 50 mobile IoT sensor nodes. Technology-wise capacity of end-users and IoT sensors are 20 Mbit/s and 4 Mbit/s respectively. Similarly, relays have 200 Mbit/s capacity which does not vary due to fixed location and base station is considered to have unlimited capacity. Throughput of IoT sensors are fixed and 2 Mbit/s. End user throughput patterns vary during the day. Coverage area of the system is considered to be 10km in diameter, base station is located in the center of the area, and all end-users and IoT sensors reside within this range during the day, neither any endpoint leaves nor enters the system. Similarly, relays are geographically distributed in the coverage range.

# Conclusion

In this work, we proposed a cooperative communication framework for a common smart city IoT application. Proposed framework uses machine learning methods by training data obtained from base station and fixed nodes, then framework uses this trained model to provide optimum relay node selection to mobile end nodes with reduced complexity.

Proposed framework shows that machine learning methods can classify the data well, which can be used as an input for future selections. This would improve overall reliability, efficiency, throughput, delay and jitter of the wireless network. Numerical model results are shown the effectiveness of the proposed scheme.

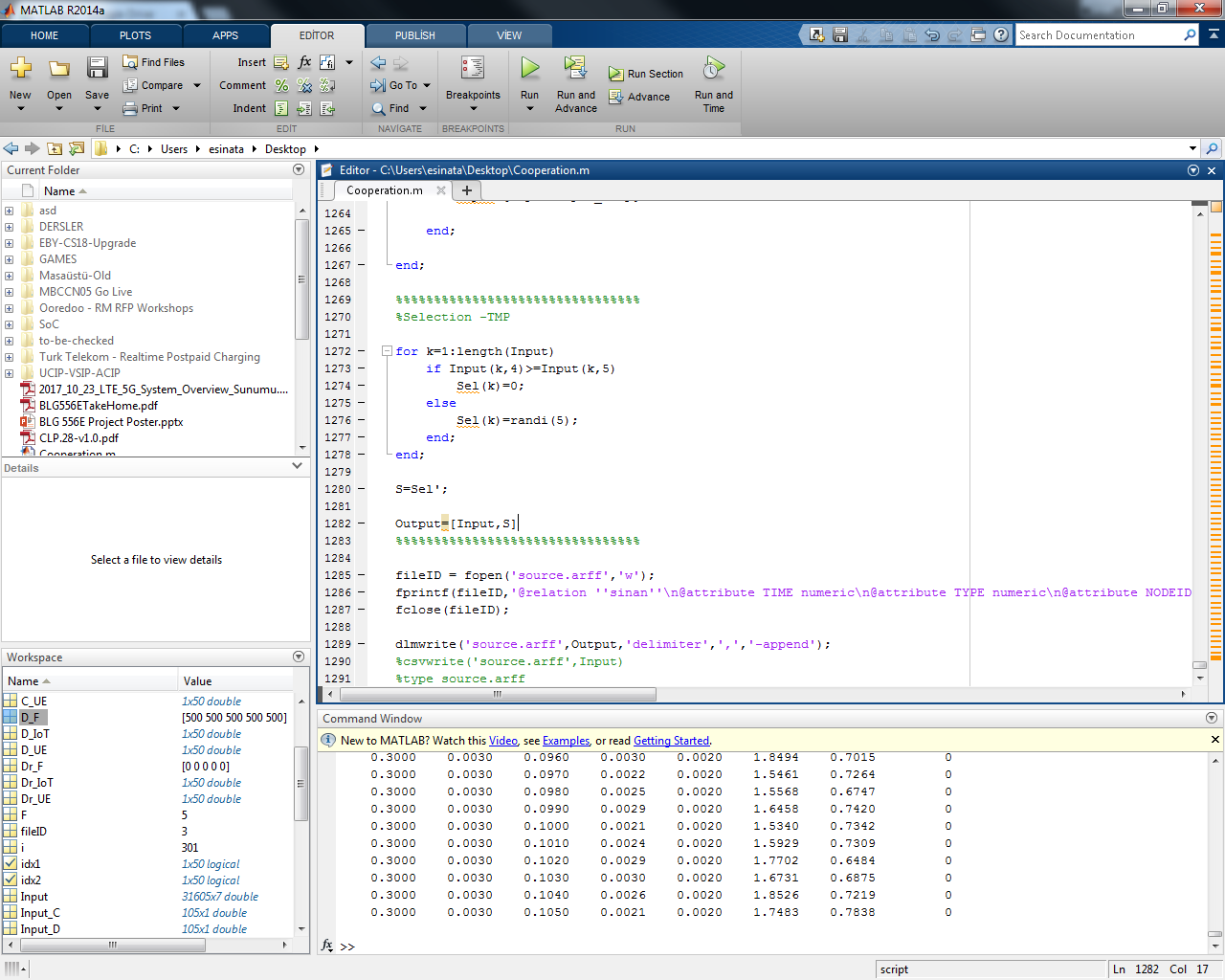
Obtaining real world data is important for future studies. As well as scale of the environment and cooperation between mobile nodes are some of the further studies in this topic.

# Development and Modelling

Dataset for proposed scheme is generated by a Matlab code. Then, this dataset is processed trained and tested by Weka machine learning tool.

# Model Data Generation

Matlab R2014 is used for generating data according to proposed framework data mode. Data generation process is done considering wireless communication attributes. Matlab code which we use to generate data can be found in “Proposed Model Data Generator Source Code” section.



# Example Generated Data

Example arff file generated by our data set generator is shown below. Weka tool can process this arff file for machine learning methods.

|  |
| --- |
| @relation 'cooperative'  @attribute TIME numeric  @attribute TYPE numeric  @attribute NODEID numeric  @attribute CAPACITY numeric  @attribute RATE numeric  @attribute BSDISTANCE numeric  @attribute R1DISTANCE numeric  @attribute R2DISTANCE numeric  @attribute R3DISTANCE numeric  @attribute R4DISTANCE numeric  @attribute R5DISTANCE numeric  @attribute selected {0, 1, 2, 3, 4, 5}  @data  0,0,1,50000,0,0,0,0,0,0,0,0  0,1,2,500,0,0,0,0,0,0,0,0  0,1,3,500,0,0,0,0,0,0,0,0  0,1,4,500,0,0,0,0,0,0,0,0  0,1,5,500,0,0,0,0,0,0,0,0  0,1,6,500,0,0,0,0,0,0,0,0  0,2,7,15.716,5.0776,1365.6,22.667,327.51,382.76,0,176.35,0  0,2,8,15.953,0,633.3,449.79,314.86,279.24,0,0,0  0,2,9,16.639,22.78,955.67,480.28,0,192.37,604.84,78.539,2  0,2,10,16.592,7.0863,1246.1,568.11,815.95,240.48,703.28,305.53,0  0,2,11,13.166,0,844.13,399.96,178.51,188.77,90.971,211.52,0  0,2,12,19.095,13.494,1617.6,79.366,362.21,53.925,454.06,66.315,0  0,2,13,18.034,14.95,878.99,223.96,264.21,0,95.039,93.067,0  0,2,14,9.0707,7.9433,1260.8,393.91,161.89,612.82,503.73,395.89,0  0,2,15,9.0488,21.357,1168.7,86.572,623.58,362.4,545.28,247.85,1  0,2,16,8.8092,11.212,1062.5,0,21.377,287.01,153.86,112.01,1  0,2,17,16.01,4.6789,816.11,224.66,867.62,386.23,512.62,557.92,0  0,2,18,18.415,10.904,1244.4,305.57,131.11,615.8,86.362,694.07,0  0,2,19,17.505,37.016,698.33,346.45,180.67,163.76,686.41,439.58,3  0,2,20,8.5343,8.1767,811.21,288.77,530.4,497.17,316.07,84.733,0  0,2,21,19.391,8.6493,1052.5,0,0,587.59,94.028,314.56,0  0,2,22,19.413,28.458,1287.6,618.24,422.71,587.55,440.98,495.88,2  0,2,23,11.359,10.922,740.31,125,268.9,663.35,49.169,159.04,0  0,2,24,15.729,26.754,1438.6,123.36,365.49,488.5,320.98,0,5  0,2,25,17.165,22.863,1067.4,218.54,237.05,398.47,0,483.85,4  0,2,26,14.486,31.776,774.58,0,439.35,322.36,101.76,0,5  … |

# Proposed Model Data Generator Source Code

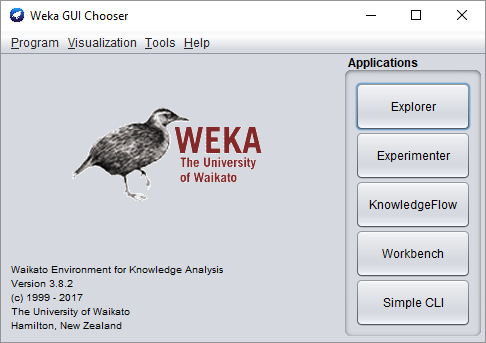
Matlab source code of proposed model data generator:

|  |
| --- |
| ADD MATLAB CODE HERE !!! |

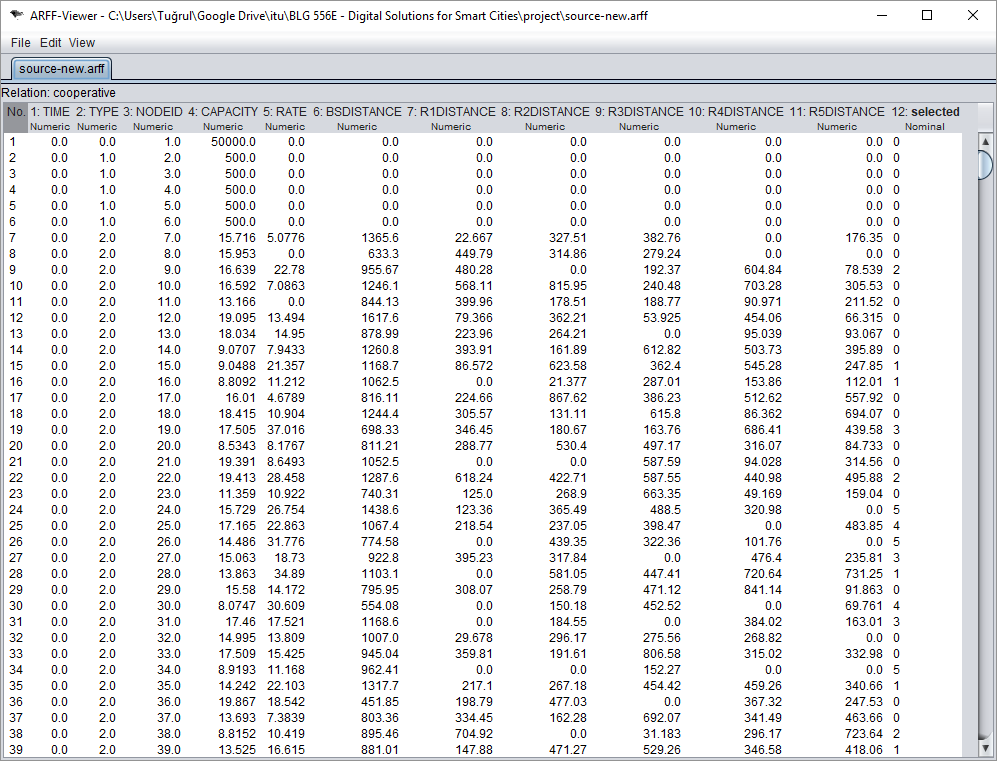
# Weka Machine Learning Tool

Weka is a powerful open source tool for machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

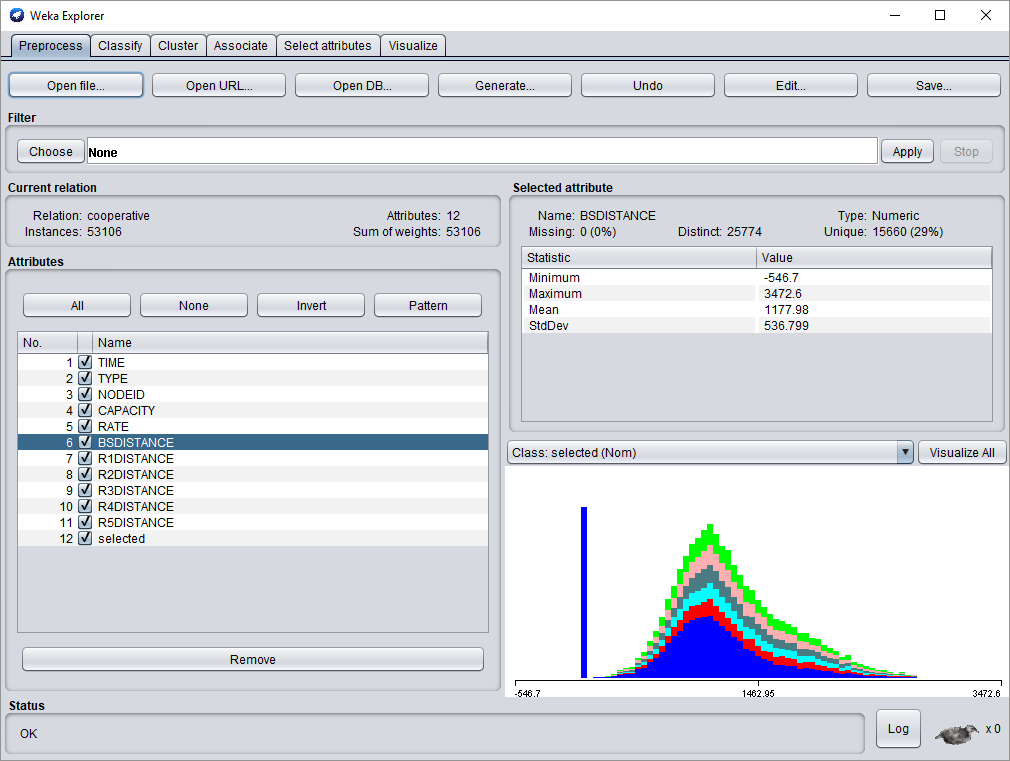
# Weka Example Screenshot



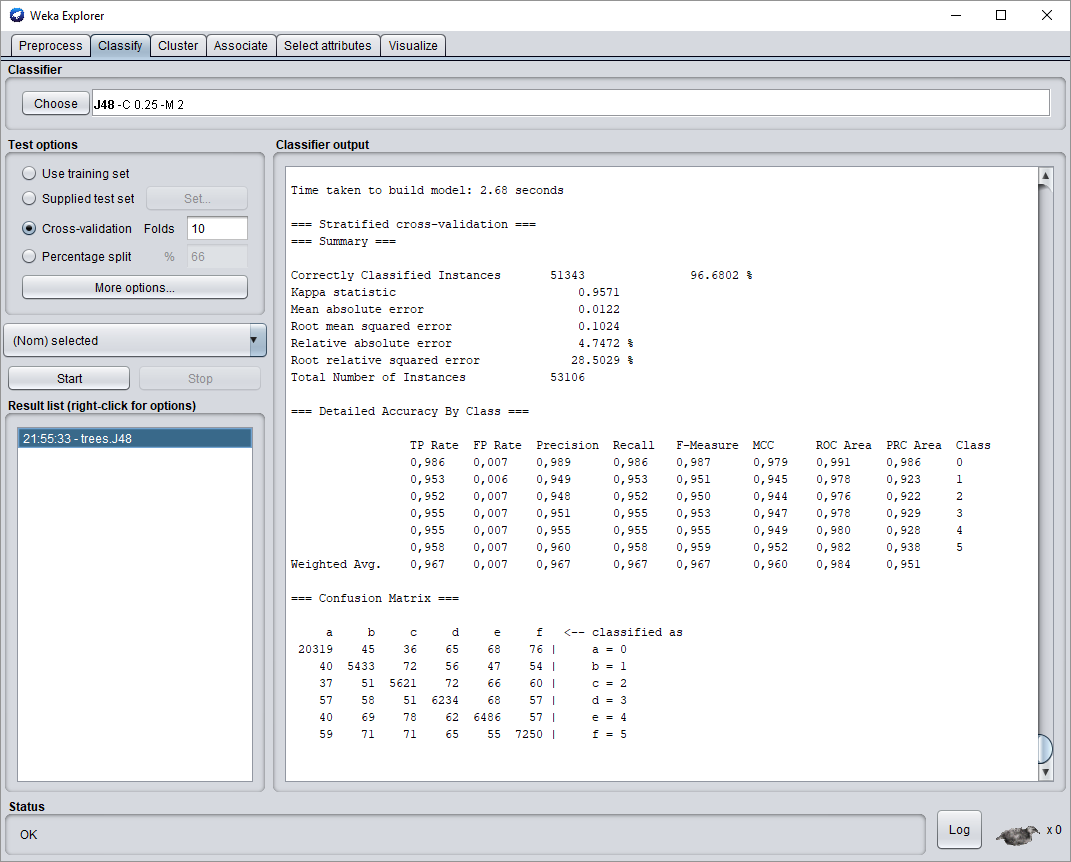
# Weka arff File Explorer Screenshot



# Weka Input Selector Screenshot



# Weka Classification Result Screenshot



# References

[1] U. Raza, P. Kulkarni and M. Sooriyabandara, "Low Power Wide Area Networks: An Overview", IEEE Communications Surveys & Tutorials, vol. 19, no. 2, pp. 855-873, second quarter 2017

[2] “Cellular networks for massive IoT: Enabling low power wide area applications,” Ericsson, Stockholm, Sweden, Tech. Rep. UEN 284 23-3278, Jan. 2016.

[3] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung and H. Yin, "Software-Defined Networks with Mobile Edge Computing and Caching for Smart Cities: A Big Data Deep Reinforcement Learning Approach," in IEEE Communications Magazine, vol. 55, no. 12, pp. 31-37, Dec. 2017.

[4] A. Al-Fuqaha et al., “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” IEEE Commun. Surveys & Tutorials, vol. 17, 4th qtr. 2015, pp. 2347–76.

[5] T. Taleb et al., “Mobile Edge Computing Potential in Making Cities Smarter,” IEEE Commun. Mag., vol. 55, no. 3, Mar. 2017, pp. 38–43.

[6] G. Han et al., “Analysis of Energy-Efficient Connected Target Coverage Algorithms for Industrial Wireless Sensor Networks,” IEEE Trans. Industrial Informatics, vol. 13, Feb. 2017, pp. 135–43.

[7] P. Pandey and R. Prabhakar, "An analysis of machine learning techniques (J48 & AdaBoost)-for classification," 2016 1st India International Conference on Information Processing (IICIP), Delhi, 2016, pp. 1-6.

[8] C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993

# Appendix

# Matlab Codes

clear all; clc;

s=RandStream.create('mt19937ar','seed',sum(100\*clock));

RandStream.setGlobalStream(s);

%Input=[Time, Type, NodeID, Capacity, Rate, BaseDistance, RelayDistances]

%Output=[Selection]

UE=50;

IoT=50;

F=5;

B=1;

C\_avg\_UE=20;

C\_avg\_F=50;

C\_avg\_IoT=2;

%TimexDay (2159: 90 Day Data Used)

ToD=0:1:2159;

%Preallocations

R\_UE=zeros(1,UE);

R\_IoT=zeros(1,IoT);

R\_F=zeros(1,F);

Input=[];

%Types

Type\_UE=zeros(1,UE)+2;

Type\_IoT=zeros(1,IoT)+3;

Type\_F=zeros(1,F)+1;

Type\_B=zeros(1,B);

Input\_Type=[Type\_B';Type\_F';Type\_UE';Type\_IoT'];

for x=1:1:length(ToD)

if mod(ToD(x),24)==0

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=250.\*randn(1,UE)+1000;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1500;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_IoT<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=10.\*randn(1,UE)+13;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=250.\*randn(5,UE)+250;

Dr\_IoT=500.\*randn(5,IoT)+350;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==1

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=200.\*randn(1,UE)+900;

D\_IoT(1:IoT)=400.\*randn(1,IoT)+1200;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=8.\*randn(1,UE)+12;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=200.\*randn(5,UE)+200;

Dr\_IoT=400.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==2

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=180.\*randn(1,UE)+800;

D\_IoT(1:IoT)=360.\*randn(1,IoT)+1000;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=4.\*randn(1,UE)+10;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=180.\*randn(5,UE)+150;

Dr\_IoT=360.\*randn(5,IoT)+300;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==3

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=180.\*randn(1,UE)+800;

D\_IoT(1:IoT)=360.\*randn(1,IoT)+1000;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=3.\*randn(1,UE)+7;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=180.\*randn(5,UE)+150;

Dr\_IoT=360.\*randn(5,IoT)+300;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==4

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=180.\*randn(1,UE)+800;

D\_IoT(1:IoT)=360.\*randn(1,IoT)+1000;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=2.\*randn(1,UE)+6;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=180.\*randn(5,UE)+150;

Dr\_IoT=300.\*randn(5,IoT)+300;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==5

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=250.\*randn(1,UE)+1000;

D\_IoT(1:IoT)=600.\*randn(1,IoT)+1700;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=6.\*randn(1,UE)+10;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=200.\*randn(5,UE)+250;

Dr\_IoT=600.\*randn(5,IoT)+600;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==6

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=800.\*randn(1,UE)+1600;

D\_IoT(1:IoT)=600.\*randn(1,IoT)+2000;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=10.\*randn(1,UE)+12;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=600.\*randn(5,UE)+400;

Dr\_IoT=800.\*randn(5,IoT)+750;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==7

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=1000.\*randn(1,UE)+1700;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1500;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=15.\*randn(1,UE)+14;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=800.\*randn(5,UE)+600;

Dr\_IoT=400.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==8

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=800.\*randn(1,UE)+1500;

D\_IoT(1:IoT)=600.\*randn(1,IoT)+1500;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=4.\*randn(1,UE)+12;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=400.\*randn(5,UE)+500;

Dr\_IoT=500.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==9

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=250.\*randn(1,UE)+750;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1200;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=4.\*randn(1,UE)+10;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=100.\*randn(5,UE)+150;

Dr\_IoT=300.\*randn(5,IoT)+400;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==10

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=300.\*randn(1,UE)+750;

D\_IoT(1:IoT)=600.\*randn(1,IoT)+1300;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=8.\*randn(1,UE)+12;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=300.\*randn(5,UE)+200;

Dr\_IoT=500.\*randn(5,IoT)+600;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==11

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=500.\*randn(1,UE)+1200;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1000;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=12.\*randn(1,UE)+15;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=400.\*randn(5,UE)+350;

Dr\_IoT=500.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==12

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=700.\*randn(1,UE)+2000;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1500;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=15.\*randn(1,UE)+18;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=400.\*randn(5,UE)+450;

Dr\_IoT=500.\*randn(5,IoT)+600;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==13

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=300.\*randn(1,UE)+1500;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1000;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=10.\*randn(1,UE)+15;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=250.\*randn(5,UE)+350;

Dr\_IoT=500.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==14

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=450.\*randn(1,UE)+1000;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1500;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=7.\*randn(1,UE)+10;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=250.\*randn(5,UE)+250;

Dr\_IoT=500.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==15

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=250.\*randn(1,UE)+1000;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1500;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=12.\*randn(1,UE)+9;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=250.\*randn(5,UE)+250;

Dr\_IoT=500.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==16

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=300.\*randn(1,UE)+1200;

D\_IoT(1:IoT)=400.\*randn(1,IoT)+1300;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=8.\*randn(1,UE)+12;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=350.\*randn(5,UE)+350;

Dr\_IoT=500.\*randn(5,IoT)+400;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==17

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=400.\*randn(1,UE)+1400;

D\_IoT(1:IoT)=400.\*randn(1,IoT)+1200;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=10.\*randn(1,UE)+14;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=350.\*randn(5,UE)+450;

Dr\_IoT=500.\*randn(5,IoT)+400;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==18

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=600.\*randn(1,UE)+2000;

D\_IoT(1:IoT)=400.\*randn(1,IoT)+1200;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=12.\*randn(1,UE)+16;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=400.\*randn(5,UE)+600;

Dr\_IoT=200.\*randn(5,IoT)+400;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==19

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=350.\*randn(1,UE)+1400;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1200;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=9.\*randn(1,UE)+15;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=250.\*randn(5,UE)+450;

Dr\_IoT=500.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==20

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=250.\*randn(1,UE)+1200;

D\_IoT(1:IoT)=500.\*randn(1,IoT)+1600;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=10.\*randn(1,UE)+15;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=250.\*randn(5,UE)+250;

Dr\_IoT=500.\*randn(5,IoT)+600;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==21

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=350.\*randn(1,UE)+1100;

D\_IoT(1:IoT)=400.\*randn(1,IoT)+1600;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=12.\*randn(1,UE)+16;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=200.\*randn(5,UE)+300;

Dr\_IoT=400.\*randn(5,IoT)+800;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

elseif mod(ToD(x),24)==22

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=400.\*randn(1,UE)+1000;

D\_IoT(1:IoT)=600.\*randn(1,IoT)+1500;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=12.\*randn(1,UE)+15;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=250.\*randn(5,UE)+250;

Dr\_IoT=500.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

else

%In general, you can generate N random numbers in the interval (a,b)

%with the formula r = a + (b-a).\*rand(N,1).

%%%%%%%%%%%%%%%%%%%

%Distance to BS

D\_UE=300.\*randn(1,UE)+1000;

D\_IoT(1:IoT)=450.\*randn(1,IoT)+1500;

D\_F(1:F)=0;

D\_B(1:B)=0;

%End-Node Distance Normalization

idx1=D\_UE<0;

idx2=D\_UE<0;

D\_UE(idx1)=0;

D\_IoT(idx2)=0;

idx1=D\_UE>5000;

idx2=D\_IoT>5000;

D\_UE(idx1)=5000;

D\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Capacity

C\_UE=20-real(log(D\_UE).\*abs(randn(1,UE)));

C\_IoT(1:IoT)=4-real(log(D\_IoT).\*abs(randn(1,IoT)));

C\_F(1:F)=500;

C\_B(1:B)=50000;

%End-Node Capacity Normalization

idx1=C\_UE<0;

idx2=C\_IoT<0;

C\_UE(idx1)=0;

C\_IoT(idx2)=0;

idx1=C\_UE>20;

idx2=C\_IoT>4;

C\_UE(idx1)=20;

C\_IoT(idx2)=4;

%%%%%%%%%%%%%%%%%%%

%Rate

R\_UE=6.\*randn(1,UE)+15;

R\_IoT(1:IoT)=2;

R\_F(1:F)=0;

R\_B(1:B)=0;

%End-Node Rate Normalization

idx1=R\_UE<0;

R\_UE(idx1)=0;

%idx2=R\_UE>C\_UE;

%R\_UE(idx2)=0;

%R\_UE=R\_UE+C\_UE.\*idx2;

%%%%%%%%%%%%%%%%%%%

%Relay Distances

Dr\_UE=300.\*randn(5,UE)+250;

Dr\_IoT=600.\*randn(5,IoT)+500;

Dr\_F=zeros(5,F);

Dr\_B=zeros(5,B);

%End-Node Relay Distance Normalization

idx1=Dr\_UE<0;

idx2=Dr\_UE<0;

Dr\_UE(idx1)=0;

Dr\_IoT(idx2)=0;

idx1=Dr\_UE>5000;

idx2=Dr\_IoT>5000;

Dr\_UE(idx1)=5000;

Dr\_IoT(idx2)=5000;

%%%%%%%%%%%%%%%%%%%

%Generate Data Set

Input\_C=[C\_B';C\_F';C\_UE';C\_IoT'];

Input\_R=[R\_B';R\_F';R\_UE';R\_IoT'];

Input\_D=[D\_B;D\_F';D\_UE';D\_IoT'];

Input\_ID=(1:(B+F+UE+IoT))';

Input\_Dr=[Dr\_B';Dr\_F';Dr\_UE';Dr\_IoT'];

%Time Input

Input\_T=(zeros(1,length(Input\_C))+ToD(x))';

Input\_temp=[Input\_T,Input\_Type,Input\_ID,Input\_C,Input\_R,Input\_D,Input\_Dr];

Input=[Input;Input\_temp];

end;

end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Selection -TMP

for k=1:length(Input)

if Input(k,4)>=Input(k,5)

Sel(k)=0;

else

tmp=min(Input(k,7:11));

for a=1:1:F

if tmp==Input(k,6+a)

Sel(k)=a;

end;

end;

end;

end;

S=Sel';

Output=[Input,S];

%Output2=[mod(Output(:,1),24),Output(:,2:12)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fileID = fopen('source-new.arff','w');

fprintf(fileID,'@relation ''sinan''\n@attribute TIME numeric\n@attribute TYPE numeric\n@attribute NODEID numeric\n@attribute CAPACITY numeric\n@attribute RATE numeric\n@attribute BSDISTANCE numeric\n@attribute R1DISTANCE numeric\n@attribute R2DISTANCE numeric\n@attribute R3DISTANCE numeric\n@attribute R4DISTANCE numeric\n@attribute R5DISTANCE numeric\n@attribute selected {0, 1, 2, 3, 4, 5}\n@data\n');

fclose(fileID);

dlmwrite('source-new.arff',Output,'delimiter',',','-append');

%csvwrite('source.arff',Input)

%type source.arff

%dlmwrite('source-new.txt',Output2,'delimiter',',','-append');

count=zeros(1,24);

for x=1:1:length(Output2)

if Output2(x,1)==0 && Output2(x,5)>Output2(x,4)

count(1)=count(1)+1;

elseif Output2(x,1)==1 && Output2(x,5)>Output2(x,4)

count(2)=count(2)+1;

elseif Output2(x,1)==2 && Output2(x,5)>Output2(x,4)

count(3)=count(3)+1;

elseif Output2(x,1)==3 && Output2(x,5)>Output2(x,4)

count(4)=count(4)+1;

elseif Output2(x,1)==4 && Output2(x,5)>Output2(x,4)

count(5)=count(5)+1;

elseif Output2(x,1)==5 && Output2(x,5)>Output2(x,4)

count(6)=count(6)+1;

elseif Output2(x,1)==7 && Output2(x,5)>Output2(x,4)

count(8)=count(8)+1;

elseif Output2(x,1)==8 && Output2(x,5)>Output2(x,4)

count(9)=count(9)+1;

elseif Output2(x,1)==9 && Output2(x,5)>Output2(x,4)

count(10)=count(10)+1;

elseif Output2(x,1)==10 && Output2(x,5)>Output2(x,4)

count(11)=count(11)+1;

elseif Output2(x,1)==11 && Output2(x,5)>Output2(x,4)

count(12)=count(12)+1;

elseif Output2(x,1)==12 && Output2(x,5)>Output2(x,4)

count(13)=count(13)+1;

elseif Output2(x,1)==13 && Output2(x,5)>Output2(x,4)

count(14)=count(14)+1;

elseif Output2(x,1)==14 && Output2(x,5)>Output2(x,4)

count(15)=count(15)+1;

elseif Output2(x,1)==15 && Output2(x,5)>Output2(x,4)

count(16)=count(16)+1;

elseif Output2(x,1)==16 && Output2(x,5)>Output2(x,4)

count(17)=count(17)+1;

elseif Output2(x,1)==17 && Output2(x,5)>Output2(x,4)

count(18)=count(18)+1;

elseif Output2(x,1)==18 && Output2(x,5)>Output2(x,4)

count(19)=count(19)+1;

elseif Output2(x,1)==19 && Output2(x,5)>Output2(x,4)

count(20)=count(20)+1;

elseif Output2(x,1)==20 && Output2(x,5)>Output2(x,4)

count(21)=count(21)+1;

elseif Output2(x,1)==21 && Output2(x,5)>Output2(x,4)

count(22)=count(22)+1;

elseif Output2(x,1)==22 && Output2(x,5)>Output2(x,4)

count(23)=count(23)+1;

elseif Output2(x,1)==23 && Output2(x,5)>Output2(x,4)

count(24)=count(24)+1;

end;

end;

percentage=count./(length(Output)/24);

figure;

grid on; hold on;

plot(1:24,percentage,'r-o','MarkerSize',6,'LineWidth',2);

title('Outage Percentage of End Nodes in Base Station');

xlabel('Time in Hour'); ylabel('Outage Percentage'); grid on;

legend('Hourly Mean Outage');