PROJECT PROBLEM

Computer Networks Lab (CSE 303 L)

Harika

AP19110010499

CSE - G

Random distribution of sensors in a network using Poisson distribution

Objective : The objective is to execute the Random distribution of sensors in a network using Poisson distribution, computing the mean and standard deviation of Poisson random variables.

Problem Statement : The problem statement is to execute the Random distribution of sensors in a network using Poisson distribution A poisson distribution Like how many sensor nodes are going to sent to the base station or to be in dead state it is a probability distribution of sensor node and dead state that likely in a particular instance of time, with various energy energy sensory nodes in the network a square deployment area with same length L and width W, Number of data packets to be sent or base station by each node as Numpkts 5 bits, except base station or sink. Each node in the network can communicate to every node in the network. Sink is located at the centre of the network distribution. Distance between the sending a data packet and the base station is d, E is energy consumed,

* Energy spent on sending one frame is directly to L and d square.

E (ni) = L*d^2

Algorithm

1. Let us set the length and width of the square deployment area to 1000.
2. Here the variable is Numdeadnodes, in the network there are 100 nodes and each node in the network sends 100 packets and the base station will be receiving 10,000 packets.
3. Select the centre coordinate of this square deployment area as (500,500) as per the position of sink in the network.
4. Each sensor node is having some initial energy Ei(ni) = 0.5
5. We used python numpy modules to create a Poisson distributed X and Y coordinates for 99 sensor nodes except sink, for a lambda value of 500.
6. Use python numpy modules to create a graph and add attributes for 100 nodes in the network :
    - ID = range from 1 to 100 where 1 is the ID of the sink node.
    - Pos = Poission distributed X and Y coordinates.
    - Initial Energy is 0.5 , Energy Transfer and initialize with 50*0.000000001.

- Destination ID = ID of sinkNode that is 1 (in 1 byte), initially always 0
- Data Field = 2 bytes of randomly generated by the Binary Bits
- CRC bytes = CRC Bits Results by the CRC Algorithm on the Data Field bits.

7. Base station is receiving the packet, Base Station is resource rich it has an infinite amount of energy, Number of packets to be sent to the sink or base station each node as Numpkts = 5, Data packets size sent by the node to sink data pktsize = 6bits.
8. Displaying the following details about the packets of sink side
    - Source ID = ID of origin of the packet
    - Data field in the packet
    - Packet Arriving time in "YY:MM:DD HH:M:SS" format

Source Code :

```
 import networkx as nx

from math import *

import numpy as np

import random

import datetime


class PACKET():

#instanting the pckt class with the attributes of packet

    def __init__(self,srcID,destID,crc,dataField):

        self.srcID="{0:08b}".format(srcID)

        self.destID="{0:08b}".format(1)

        self.crc=crc

        self.dataField=dataField
```

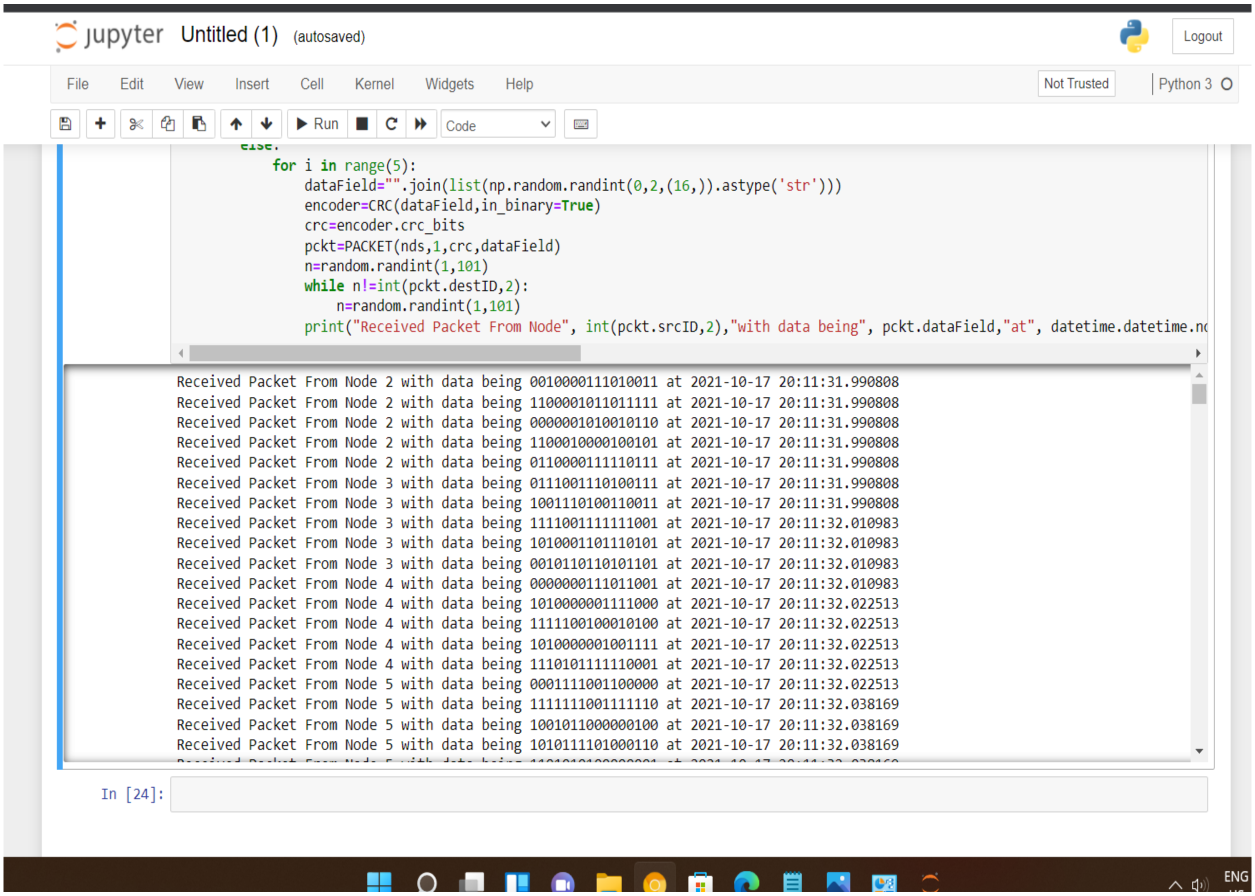
```
class CRC():

#divisor generated using the polynomial coefficient , and initial error status to be -1.

    def __init__(self,msg,in_binary=False):

        self.sent_message=msg

        self.sent_message_encoded=""

        self.sent_message_binary=in_binary

        self.received_message_encoded=""

        self.message_len=0

        self.crc_bits=""

        self.crc_bits_len=16

        self.divisor="10001000000100001"

        self.message_error_status=-1

   #Recursive method to perform binary division with CRC16 generated binary as divisor which takes dividend to be accurate at each recursion step.

    def BinaryDivision(self,partial_divident):

        did1=partial_divident[:len(self.divisor)]

        did2=partial_divident[len(self.divisor):]

        div=self.divisor

       #if the first bit that is left to right of dividend is to perform with divisor.

        if did1[0]=="0":

            div = "0"*len(self.divisor)

        y=int(did1,2)^int(div,2)

        pad_d=str(len(div))

        format_str="{0:0" +pad_d+ "b}"

 #if any part of dividend is not
```

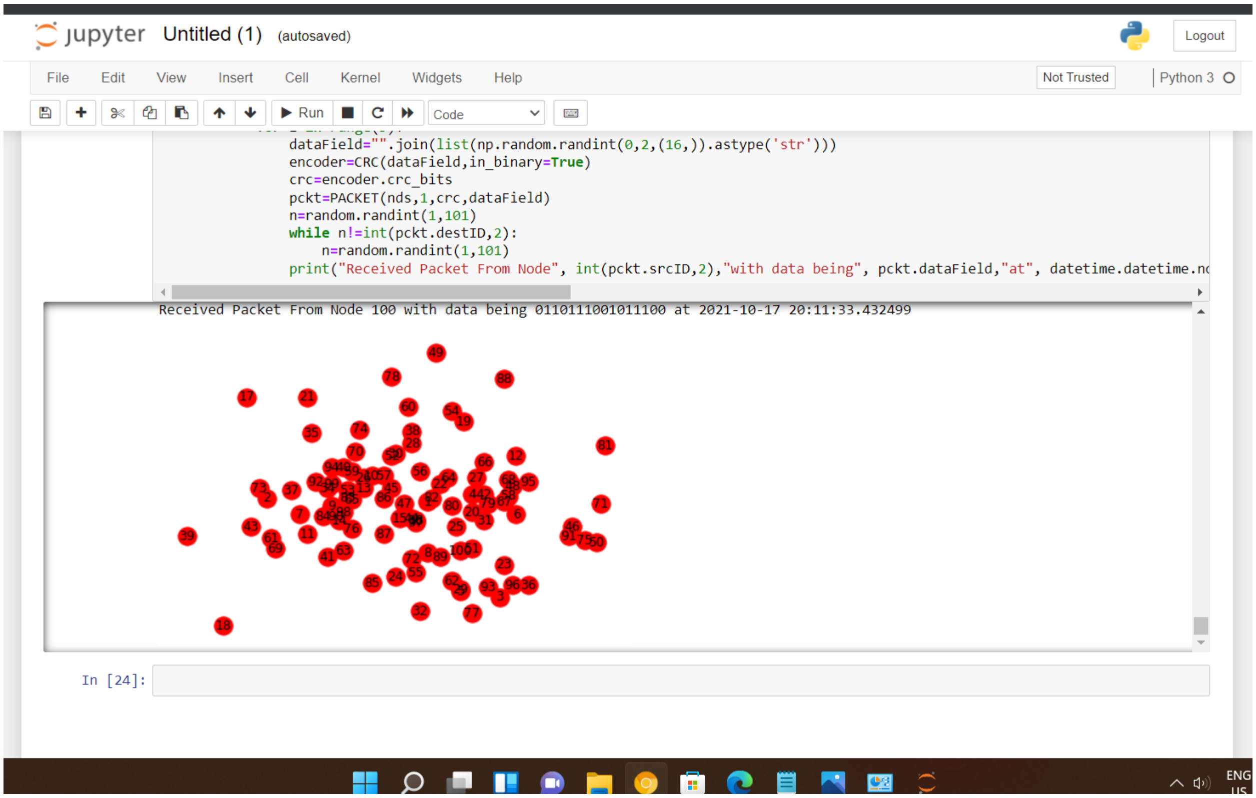
```
        if did2=="":
            next_partial_divident=format_str.format(y)
            return next_partial_divident
        else:
            next_partial_divident=format_str.format(y)[1:]+did2
            return self.BinaryDivision(next_partial_divident)
    def Encode(self):
        if self.sent_message_binary:
            res=self.sent_message
        else:
            res=''.join(format(ord(i),'08b') for i in self.sent_message)
        data=res
        m=len(data)
        self.message_len=m
        self.crc_bits=self.BinaryDivision(data+"".join(["0" for i in
range(self.crc_bits_len)]))[1:]
        self.sent_message_encoded=data+self.crc_bits
        return self.sent_message_encoded

    def CheckMessage(self, rmsg):
        self.received_message_encoded=rmsg
        if set(self.BinaryDivision(rmsg)[1:])==set(['0']):
            self.message_error_status=1
        else:
            self.message_error_status=0
        return
    netG=nx.Graph()
```

```
    L,W,C=1000,1000,500

    CtrlPktSize=200

    node_xs=np.concatenate([np.array([500]),
np.random.poisson(lam=500,size=99)])

    node_ys=np.concatenate([np.array([500]),
np.random.poisson(lam=500,size=99)])

#instanting all the parameters of nodes in the network

    InitEnergy=0.5

    EnergyTransfer=50*0.000000001

    EnergyReceive=50*0.000000001

    EnergyFreeSpace=10*0.000000000001

    EnergyMultiPath=0.0013*0.000000000001

    EnergyAgg=5*0.000000001

    Dead=0

    NumPkts=5

    DataPktSize=6

    POSs=dict()


    for i in range(100):

        POSs[i+1]=(node_xs[i],node_ys[i])


netG.add_nodes_from([(i,{"ID":i,"pos":(node_xs[i-1],node_ys[i-1]),"InitEne
rgy":InitEnergy,
"EnergyTransfer":EnergyTransfer,"EnergyReceive":EnergyReceive,"EnergyFreeS
pace":EnergyFreeSpace,"EnergyMultiPath":EnergyMultiPath,"EnergyAgg":Energy
Agg,"Dead":Dead, "NumPkts":NumPkts, "DataPktSize":DataPktSize})for i in
range(1,101)])

    nx.draw(netG,POSs,with_labels=True,node_color="Red")

    for nds in range(1,101):
```

```
#executing a packet for data transmission
        if nds==1:
            continue
        else:
            for i in range(5):
   #creating a packet for data transmission
dataField="".join(list(np.random.randint(0,2,(16,)).astype('str')))
                encoder=CRC(dataField,in_binary=True)
                crc=encoder.crc_bits
                pckt=PACKET(nds,1,crc,dataField)
                n=random.randint(1,101)
                while n!=int(pckt.destID,2):
                    n=random.randint(1,101)
                print("Received Packet From Node", int(pckt.srcID,2),"with
data being", pckt.dataField,"at", datetime.datetime.now())
```

<u>**Output:**</u>


jupyter Untitled (1) (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
else:
for i in range(5):
dataField="".join(list(np.random.randint(0,2,(16,)).astype('str')))
encoder=CRC(dataField,in_binary=True)
crc=encoder.crc_bits
pckt=PACKET(nds,1,crc,dataField)
n=random.randint(1,101)
while n!=int(pckt.destID,2):
n=random.randint(1,101)
print("Received Packet From Node", int(pckt.srcID,2),"with data being", pckt.dataField,"at", datetime.datetime.n
Received Packet From Node 2 with data being 0010000111010011 at 2021-10-17 20:11:31.990808
Received Packet From Node 2 with data being 1100001011011111 at 2021-10-17 20:11:31.990808
Received Packet From Node 2 with data being 0000001010010110 at 2021-10-17 20:11:31.990808
Received Packet From Node 2 with data being 1100010000100101 at 2021-10-17 20:11:31.990808
Received Packet From Node 2 with data being 0110000111110111 at 2021-10-17 20:11:31.990808
Received Packet From Node 3 with data being 0111001110100111 at 2021-10-17 20:11:31.990808
Received Packet From Node 3 with data being 1001110100110011 at 2021-10-17 20:11:31.990808
Received Packet From Node 3 with data being 1111001111111001 at 2021-10-17 20:11:32.010983
Received Packet From Node 3 with data being 1010001101110101 at 2021-10-17 20:11:32.010983
Received Packet From Node 3 with data being 0010110110101101 at 2021-10-17 20:11:32.010983
Received Packet From Node 4 with data being 0000000111011001 at 2021-10-17 20:11:32.010983
Received Packet From Node 4 with data being 1010000001111000 at 2021-10-17 20:11:32.022513
Received Packet From Node 4 with data being 1111100100010100 at 2021-10-17 20:11:32.022513
Received Packet From Node 4 with data being 1010000001001111 at 2021-10-17 20:11:32.022513
Received Packet From Node 4 with data being 1110101111110001 at 2021-10-17 20:11:32.022513
Received Packet From Node 5 with data being 0001111001100000 at 2021-10-17 20:11:32.022513
Received Packet From Node 5 with data being 1111111001111110 at 2021-10-17 20:11:32.038169
Received Packet From Node 5 with data being 1001011000000100 at 2021-10-17 20:11:32.038169
Received Packet From Node 5 with data being 1010111101000110 at 2021-10-17 20:11:32.038169
In [24]:
ENG



Challenges Faced :

1. I found it difficult to implement the simulation concept at first. But, after referring to a few sources I was able to implement it using python programming.
2. I even faced a bit of a problem in understanding the poisson distribution concept.
3. Searching for popper implementation python modules for networks and graphs delayed my progress by a tiny bit. Previous experience in an IOT research led by the University faculty came in handy and didn't let this stage consume much of my time.

Conclusion :

1. We have implemented Random distribution of sensors in a network using Poisson distribution. Through this assignment we were able to learn and practice some cognitive skills and also could be faced before even implementing the algorithm.

2. We were able to gain the Knowledge our skills using OOPS concepts in implementing some algorithms,usage of packages, exploring documentation of the third party modules in python and optimally coding the methods to increase the time complexity.

Skills:

While working on this project, I was able to understand and implement different concepts of python and its libraries. This helped me with my coding abilities and I was able to understand and implement the concept more clearly.

Attitudinal:

My group and I discussed and worked on the coding and algorithm. Each one of us contributed to better the idea and the code. Since we were able to implement it on our own, we got a better understanding of how this concept works in real life too.