Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN2053 - Communication Networks

Assignment on Wireless Communication

Group Members:

- Rathnayaka R.G.H.V. - 180529E
- Thilakarathna G.D.O.L. - 180642T
- Udara A.W.T. - 180650P

This is submitted as a partial fulfillment for the module
EN2053 - Communication Networks
Department of Electronic and Telecommunication Engineering
University of Moratuwa

13th September 2020

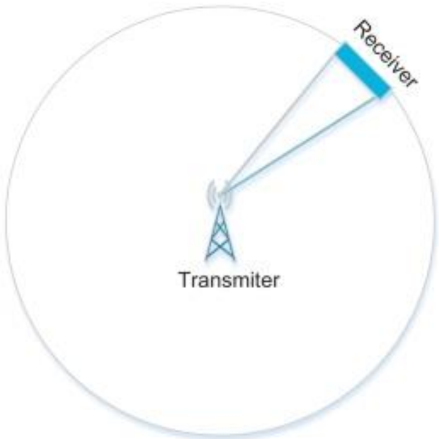
Introduction

Wireless communication has made a revolutionary change in modern communication field. The main objective of this assignment on Wireless communication is to analyze various factors of the propagation model and to understand different routing protocols found in ADHOC networks. This assignment is comprised of two tasks. We will be able to model a wireless communication network with various propagation models through task 1. We will be able to understand various routing protocols used in mobile Ad hoc networks (MANETS)

Task 1 – Modeling the RF propagation model using Matlab

Radio wave propagation in wireless networks is a result of spread out of energy as it travels farther away from a point of origin. This model assumes that both transmitter antenna and the receiver antenna in an otherwise empty environment. No absorption by obstacles and no reflection by surfaces are considered. Simply there is no additional losses between transmit and receive antennas. For antennas here we assume that they are isotropic antennas which shows homogeneous propagation in all directions. Here the influence of the earth surface is also omitted.

In a propagation of a distance d which is larger than the size of antenna, the radiating antenna is considered as a point source with negligible dimensions. Energy is radiated homogeneously in all directions over a surface of a hypothetical sphere. As the surface area of a sphere of radius d is $4\pi d^2$, according to this model the power density P_D at a distance d away from the transmitting antenna with power P_T and antenna gain G_T is,



$P_D = (P_T.G_T) / (4\pi d^2)$ --------- 1

Here the power density at the wave front inversely proportional to d^2. The available power at a receive antenna P_R depends on the effective aperture of receiving antenna where A_{eff} is the effective area or the aperture of the antenna.

$P_R = P_D.A_{eff}$ --------- 2

As the receiving antenna is assumed to be hypothetically isotropic the receiver gain G_R can be defined as,

$G_R = (4\pi.A_{eff}) / \lambda^2$ --------- 3

From 2 & 3, $P_R = P_D.(\lambda^2. G_R) / 4\pi$ --------- 4

From 4 & 1, $P_R = (P_T.G_T) .(\lambda^2. G_R) / (4\pi d^2).(4\pi)$

$$P_R = (P_T.G_T) .(\lambda^2. G_R) / (4\pi d)^2 \quad \text{--------- 5}$$

Since the wavelength $\lambda = (C / f)$ where C is the velocity of light and the f is the carrier frequency,

$$P_R = (P_T.G_T.G_R. C^2) / (4\pi df)^2 \quad \text{--------- 6}$$

The product $P_T.G_T$ is called the Effective Radiated Power (ERP) of the transmitter.

The Free Space Path Loss (FSL) can be defined as,

$$\text{FSL} = 10 \log(P_T / P_R) \quad \text{--------- 7}$$

$$\text{FSL} = 10 \log\{(4\pi df)^2 / (G_T.G_R. C^2)\} \quad \text{--------- 8}$$

By taking the transmitter gain G_T and receiver gain G_R as unity and taking the distance in km and the frequency in GHz,

$$\text{FSL} = 10 \log\{(4\pi*10^3*10^6*df)^2 / (C^2)\} \quad \text{--------- 9}$$

As the $C = 3\text{x}10^8 \text{ ms}^{-1}$,

$$\text{FSL} = 10 \log\{(4\pi*10^3*10^9*df)^2 / (3\text{x}10^8)^2\}$$

$$\text{FSL} = 10 \log\{(4\pi*10^{12}*df)^2 / (3\text{x}10^8)^2\}$$

$$\text{FSL} = 10 \log\{(4\pi*10^{12}*df / (3\text{x}10^8))^2\}$$

$$\text{FSL} = 20 \log(4\pi*10^{12} / (3\text{x}10^8)) + 20 \log(d) + 20 \log(f)$$

$$\mathbf{FSL = 92.4418 + 20 \log(\mathit{d}) + 20 \log(\mathit{f})} \quad \text{--------- 10}$$

Here d should be in km and f should be in GHz. Therefore, the loss related to this mechanism of propagation is known as free space path loss depends on frequency and the distance of propagation. This is also named as spreading loss. Radio frequency signals propagates at a constant velocity of light behave like follow.

$$\text{FSL} = 10 \log\{(4\pi df)^2 / (C^2)\}$$

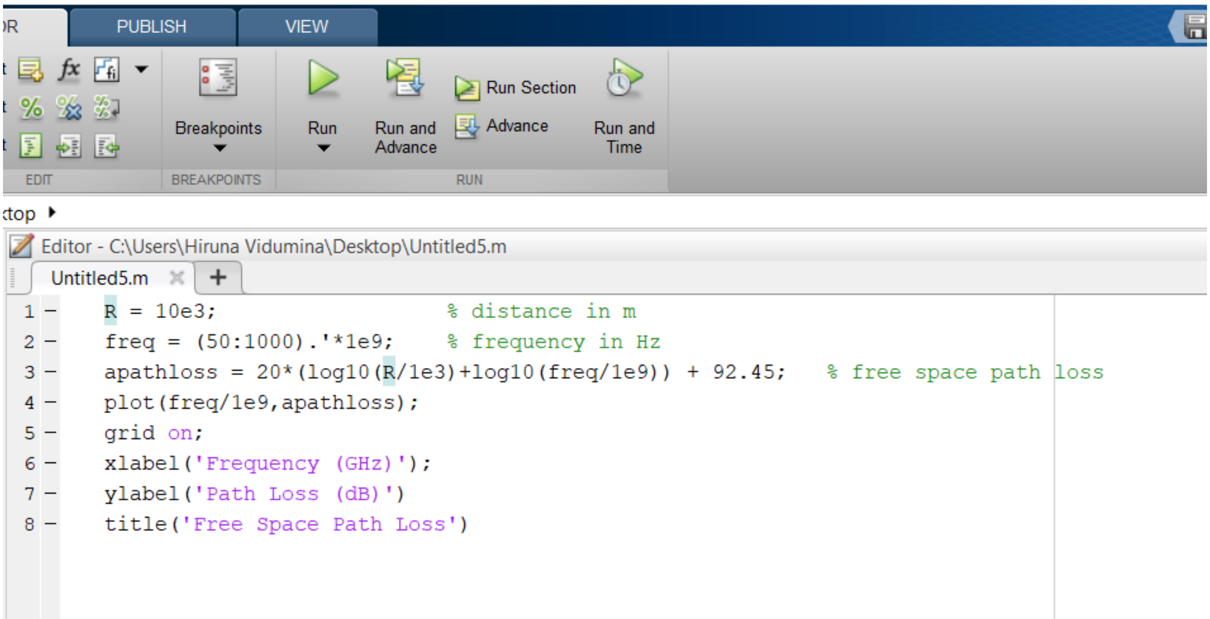
$$\text{FSL} = 20 \log (4\pi df / C) \quad \text{dB}$$

Two-way free space path loss can be calculated by doubling the one-way free space path loss.

Variation of free space path loss with frequency

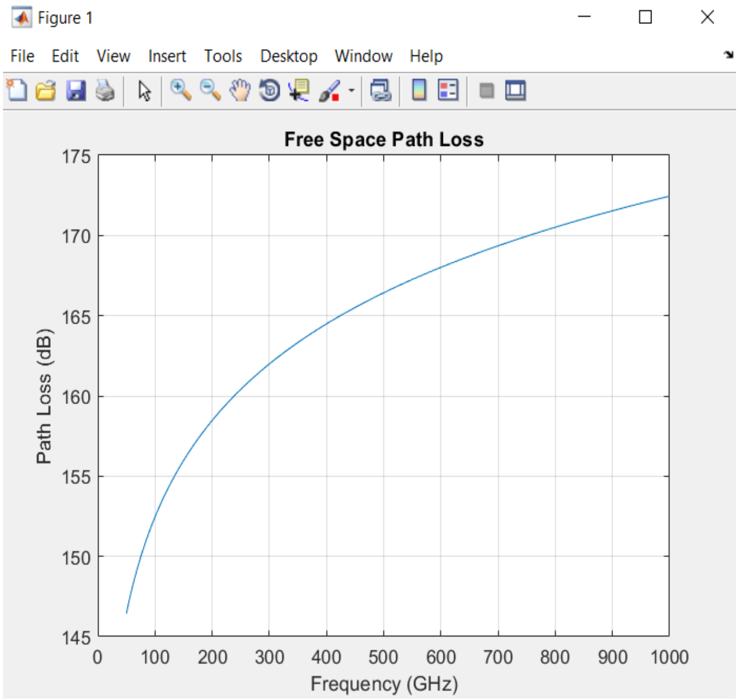
In this assignment a transmitter is used to transmit voice signal over to a receiver located 10km away. The variation of free space path loss with frequency can be illustrated from the graph plotted by using Matlab.

Matlab Code:



```
R = 10e3;                 % distance in m
freq = (50:1000).'*1e9;   % frequency in Hz
apathloss = 20*(log10(R/1e3)+log10(freq/1e9)) + 92.45;   % free space path loss
plot(freq/1e9,apathloss);
grid on;
xlabel('Frequency (GHz)');
ylabel('Path Loss (dB)')
title('Free Space Path Loss')
```

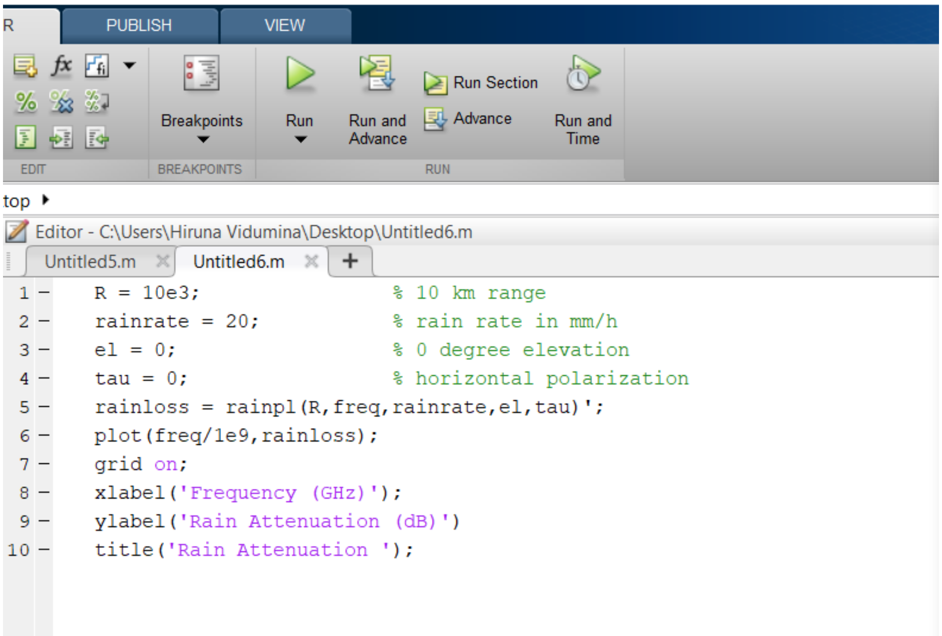
Variation of free space path loss with frequency:



Variation of rain Attenuation with frequency

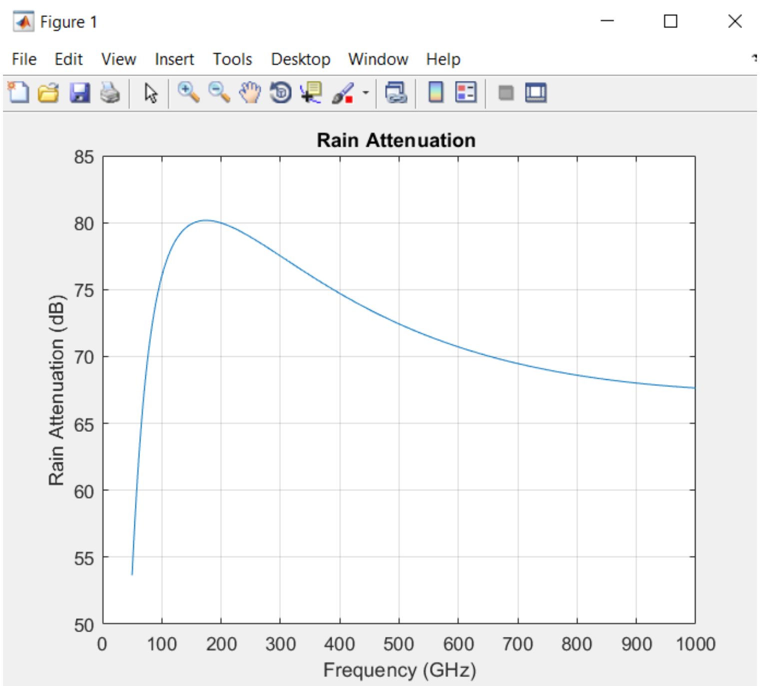
During the process of wireless communication, radio frequency signals released from the transmitter get interact with air particles. As a result of this energy loss may occur in the propagation path. Rain is considered as a major factor in reducing the energy. In plotting the behavior of rain attenuation with frequency we have assumed that polarization is horizontal which means tilt angle = 0 and the signal is propagating parallel to the ground which means elevation angle = 0. Horizontal polarization represents the maximum rain attenuation hence it is known as the worst case in propagation loss due to rain attenuation.

Matlab Code:




```
R = 10e3;                    % 10 km range
rainrate = 20;               % rain rate in mm/h
el = 0;                      % 0 degree elevation
tau = 0;                     % horizontal polarization
rainloss = rainpl(R,freq,rainrate,el,tau)';
plot(freq/1e9,rainloss);
grid on;
xlabel('Frequency (GHz)');
ylabel('Rain Attenuation (dB)')
title('Rain Attenuation ');
```

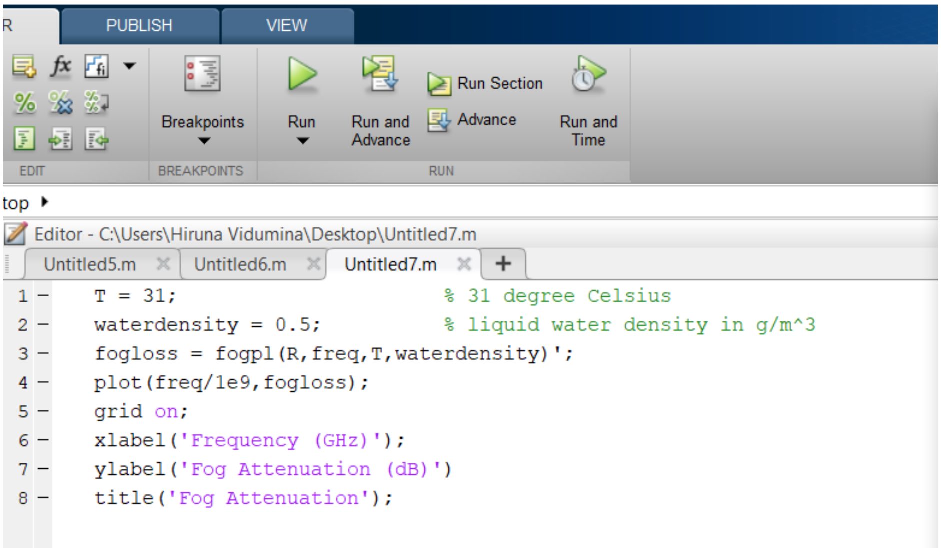
Variation of rain attenuation with frequency:



Variation of fog Attenuation with frequency

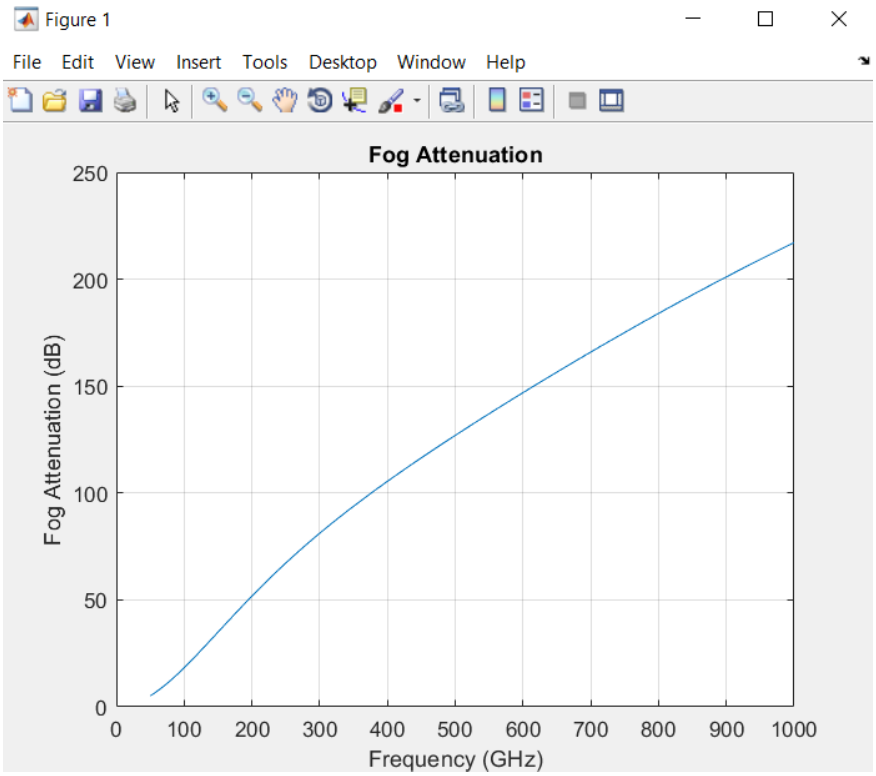
Although fog and cloud are very smaller relative to water drops, they are also formed by water droplets. The size of a fog droplet may even be less than 0.01cm. this fog is characterized by liquid water density. The following plot depicts the behavior of fog attenuation with the change of frequency. Atmospheric temperature may also influence in this variation.

Matlab Code:




```
T = 31;                    % 31 degree Celsius
waterdensity = 0.5;        % liquid water density in g/m^3
fogloss = fogpl(R,freq,T,waterdensity)';
plot(freq/1e9,fogloss);
grid on;
xlabel('Frequency (GHz)');
ylabel('Fog Attenuation (dB)')
title('Fog Attenuation');
```

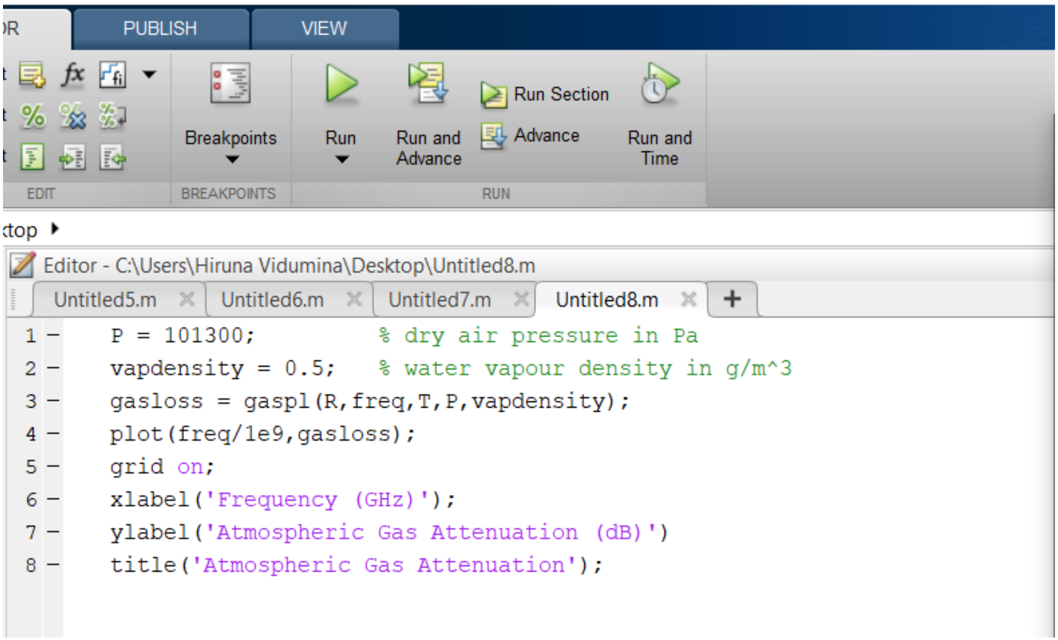
Variation of fog attenuation with frequency:



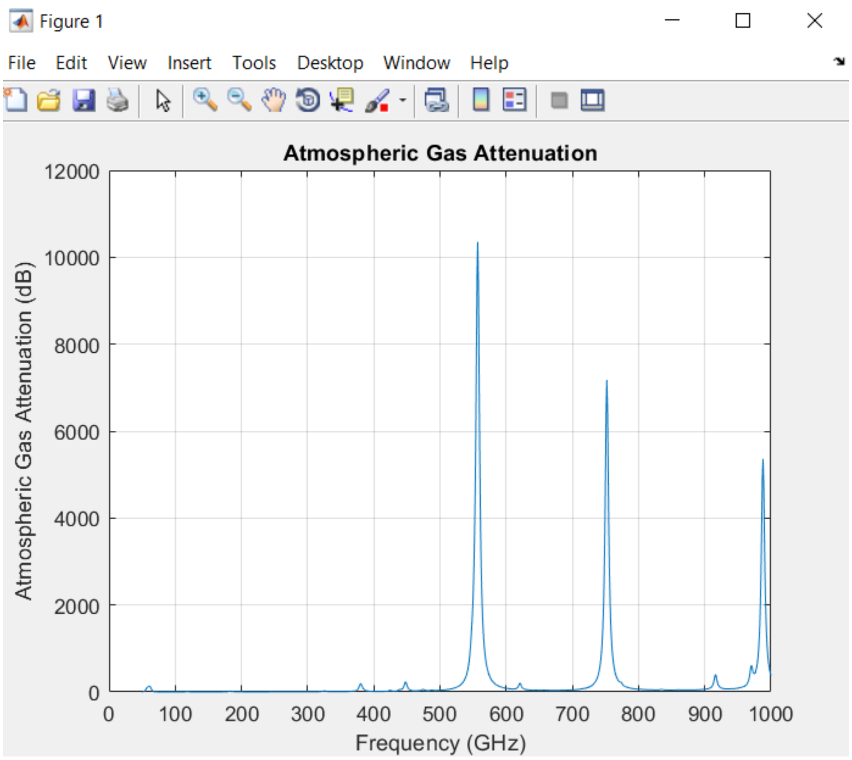
Variation of Atmospheric gas Attenuation with frequency

Atmospheric gas attenuation is a function of both dry air pressure from gases like oxygen and water vapour density. The graph given below will depict the change of atmospheric gas attenuation with the change of frequency.

Matlab Code:

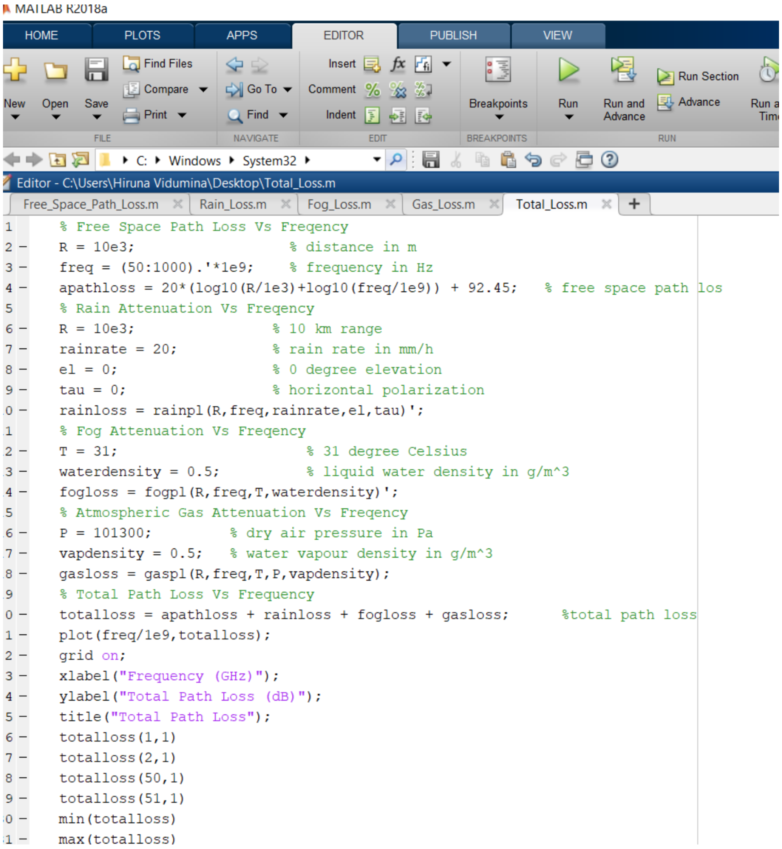


Variation of atmospheric gas attenuation with frequency:



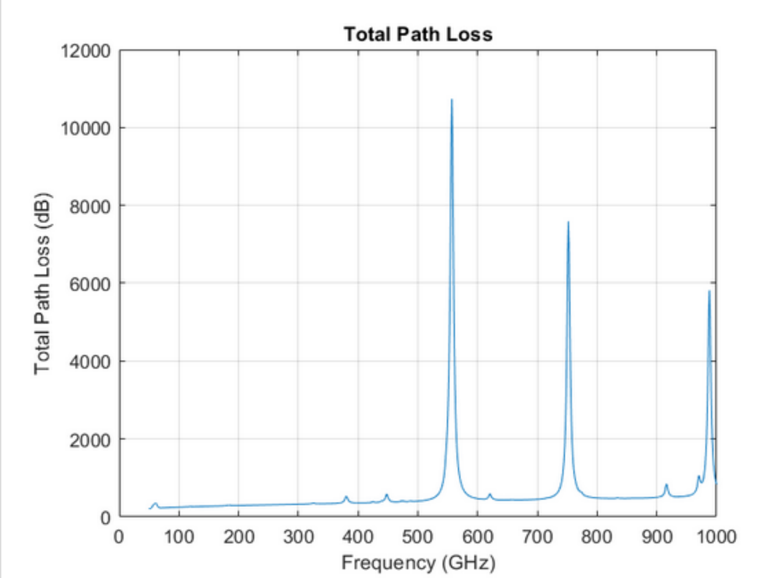
Variation of Total Path Loss with frequency

Matlab Code:



```
% Free Space Path Loss Vs Freqency
R = 10e3;                     % distance in m
freq = (50:1000).'*1e9;      % frequency in Hz
apathloss = 20*(log10(R/1e3)+log10(freq/1e9)) + 92.45;    % free space path los
% Rain Attenuation Vs Freqency
R = 10e3;                   % 10 km range
rainrate = 20;              % rain rate in mm/h
el = 0;                     % 0 degree elevation
tau = 0;                    % horizontal polarization
rainloss = rainpl(R,freq,rainrate,el,tau)';
% Fog Attenuation Vs Freqency
T = 31;                         % 31 degree Celsius
waterdensity = 0.5;             % liquid water density in g/m^3
fogloss = fogpl(R,freq,T,waterdensity)';
% Atmospheric Gas Attenuation Vs Freqency
P = 101300;          % dry air pressure in Pa
vapdensity = 0.5;    % water vapour density in g/m^3
gasloss = gaspl(R,freq,T,P,vapdensity);
% Total Path Loss Vs Frequency
totalloss = apathloss + rainloss + fogloss + gasloss;      %total path loss
plot(freq/1e9,totalloss);
grid on;
xlabel("Frequency (GHz)");
ylabel("Total Path Loss (dB)");
title("Total Path Loss");
totalloss(1,1)
totalloss(2,1)
totalloss(50,1)
totalloss(51,1)
min(totalloss)
max(totalloss)
```

Variation of Total Path Loss with frequency:




```
ans = 207.6179
ans = 209.9244
ans = 246.6193
ans = 246.0388
ans = 247.1964
ans = 207.6179
ans = 1.0731e+04
```

According to the details depicted in the graph the minimum path loss can be expected at **50 GHz.** It is 207.6179 dB. So, 50 GHz can be taken as a suitable frequency for transmission.

Receiver Sensitivity

Tx power AP	= 76.99 dBm
Antenna gain AP	= 30 dB
Cable Loss AP	= -3 dB
Antenna gain Client	= 24.77 dB
Cable Loss Client	= -4 dB

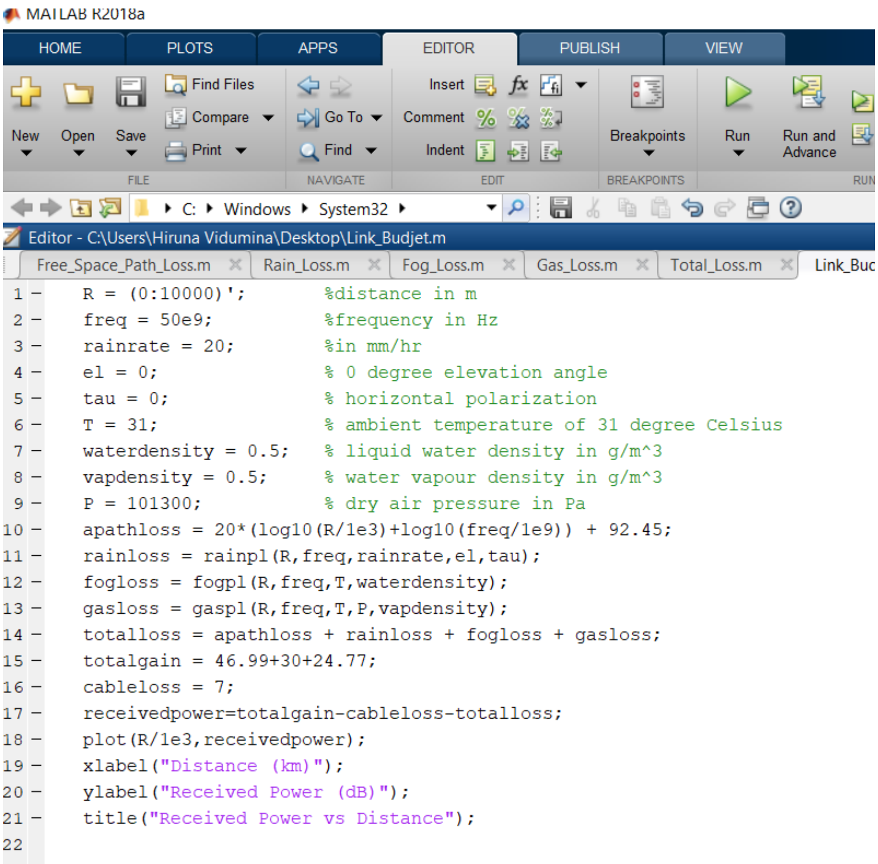
Total Gain	= 124.76 dB
Free Space Loss (10km)	= -207.62 dB

Expected Received signal level	= -82.86 dBm
	(-)
Receiver Sensitivity	**= ?**

Link Margin	= 11 dB

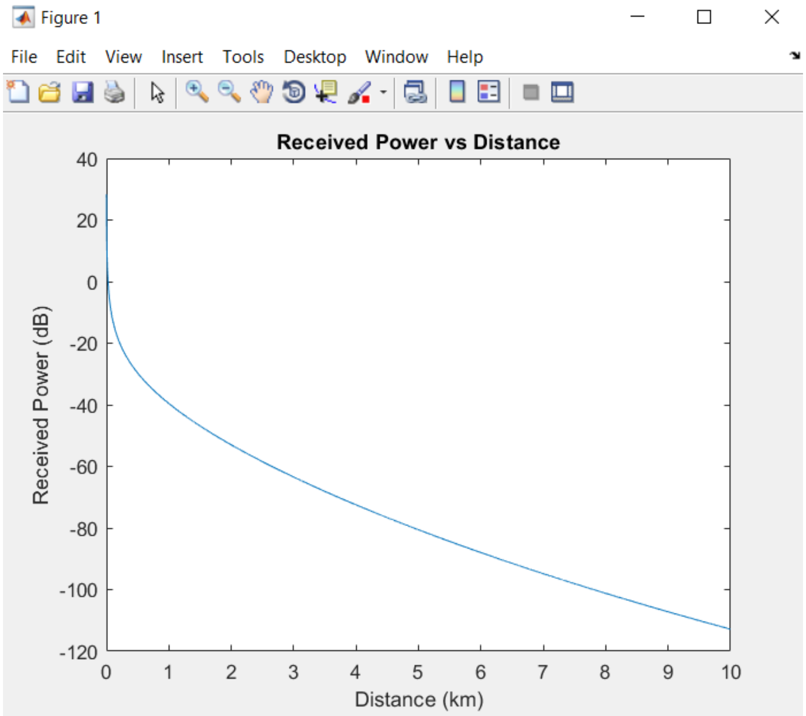
Therefore, Receiver Sensitivity = Expected Received signal level - Link Margin
= (-82.86 dBm) – (11 dB)
= **-93.86 dBm**

Matlab Code for Link Budget Calculation:

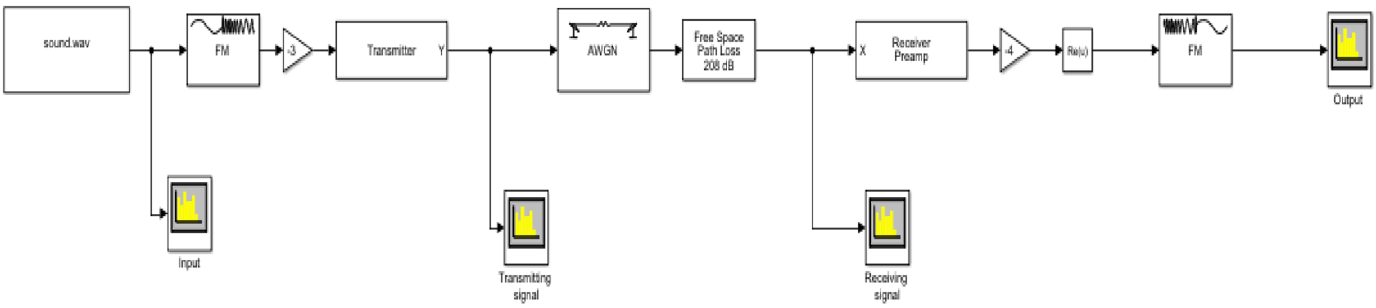



```
R = (0:10000)';         %distance in m
freq = 50e9;            %frequency in Hz
rainrate = 20;          %in mm/hr
el = 0;                 % 0 degree elevation angle
tau = 0;                % horizontal polarization
T = 31;                 % ambient temperature of 31 degree Celsius
waterdensity = 0.5;     % liquid water density in g/m^3
vapdensity = 0.5;       % water vapour density in g/m^3
P = 101300;             % dry air pressure in Pa
apathloss = 20*(log10(R/1e3)+log10(freq/1e9)) + 92.45;
rainloss = rainpl(R,freq,rainrate,el,tau);
fogloss = fogpl(R,freq,T,waterdensity);
gasloss = gaspl(R,freq,T,P,vapdensity);
totalloss = apathloss + rainloss + fogloss + gasloss;
totalgain = 46.99+30+24.77;
cableloss = 7;
receivedpower=totalgain-cableloss-totalloss;
plot(R/1e3,receivedpower);
xlabel("Distance (km)");
ylabel("Received Power (dB)");
title("Received Power vs Distance");
```

Link Budget



Transmitting a voice signal over a noisy channel



Task 2 – Implementing a simplified version of the DSR(Dynamic source routing) routing protocol using python

Python Code:

```
from .packet import PKT_TYPE
from .packet import Packet


class Node:
    def __init__(self, id, x, y, range):
        """
            Attributes:
                id: Node address
                x:  x-coordinate
                y:  y-coordinate
                range: Maximum transmission range
                queue_in: Input queue. Packets entering will be present here with the earliest one at index 0
                queue_out: Output queue of node. Packets forwarded to by this node must be appended to
this list
                adjacent_nodes: A dictionary of neighboring nodes which will be updated every time step
                routing_cache: A dictionary containing the routes discovered so far. Implemented with a
expiration duration if not used recently.
                expire_time: expire time of a route in cache
                recent: A list of recently recieved packets (RREQ only)
                count : Used to generate a unique id  for data packets originating from this node
                buffer: A dictionary of Buffered DATA packets
                recieved: A list of data packets sent to this node( i.e packet target == node.id)
        """
        self.id = id
        self.x = x
        self.y = y
        self.range = range
        self.queue_in = []
        self.queue_out = []
        self.adjacent_nodes = {}
        self.routing_cache = {}
        self.expire_time = 30
        self.recent = []
        self.count = 1
        self.buffer = {}
        self.received = []

    def generate_pkt_id(self):
        """
            Generates a unique packet id
```

```
        """
        return self.id + str(self.count+1)

    def forward(self):
        """
            Packet forwarding
        """
        pkt = None
        if self.queue_in != []:
            pkt = self.queue_in.pop(0)
        if pkt is not None:
            self.route(pkt)

    def check_in_recent(self, pkt):
        """
            Checks whether the given pkt is in the nodes recently forwarded packets.
            For RREQ packets
        """
        assert pkt.type == PKT_TYPE.RREQ,"Invalid packet type"
        if (pkt.source, pkt.target, pkt.id) in self.recent:
            return True
        return False

    def add_to_recent(self, pkt):
        """
            Adds the pkt to the recent history of the node
            For RREQ packets
        """
        assert pkt.type == PKT_TYPE.RREQ,"Invalid packet type"
        self.recent.append((pkt.source, pkt.target, pkt.id))

    def add_to_cache(self, pkt):
        """
            Add the packets source_route to the route cache
            For RREP packets
        """
        self.routing_cache[pkt.target] = [pkt.source_route,self.expire_time]

    def check_in_cache(self, pkt):
        """
            Check whether a route has been already discovered from the node to the target
        """
        if pkt.target in self.routing_cache.keys():
            return True
        return False

    def add_path_from_cache(self, pkt):
        """
```

```
            Add route from cache to the source route of the packet
        """
        pkt.source_route = self.routing_cache[pkt.target][0]
        self.routing_cache[pkt.target][1] = self.expire_time

    def check_in_buffer(self, pkt):
        if pkt.id in self.buffer.keys():
            return True
        return False

    def add_to_buffer(self, pkt):
        self.buffer[pkt.id] = pkt

    def retrieve_from_buffer(self, pkt):
        DATA_pkt = self.buffer[pkt.id]
        del self.buffer[pkt.id]
        DATA_pkt.source_route = pkt.source_route
        DATA_pkt.next_hop += 1
        return DATA_pkt

    def generate_RREP(self, pkt):
        assert pkt.type == PKT_TYPE.RREQ, "RREP can be generated only for RREQ pkts. pkt
recieved {}".format(pkt.type)
        pkt.type = PKT_TYPE.RREP
        pkt.source_route.append(self.id)
        return pkt

    def generate_RREQ(self, pkt):
        RREQ_pkt = Packet(pkt.id, PKT_TYPE.RREQ)
        RREQ_pkt.source = pkt.source
        RREQ_pkt.target = pkt.target
        RREQ_pkt.source_route.append(self.id)
        return RREQ_pkt

    def add_to_queue_out(self, pkt):
        if pkt.type == PKT_TYPE.RREQ and pkt.source != self.id:
            pkt.next_hop += 1
        elif pkt.type == PKT_TYPE.DPKT and pkt.source != self.id:
            pkt.next_hop += 1
        elif pkt.type == PKT_TYPE.RREP and pkt.target != self.id:
            pkt.next_hop -= 1
        self.queue_out.append(pkt)

    def route(self, pkt):
        """
            A packet can be RREQ (Route Request), RREP(Route reply) or a DPKT(Data packet)
            A data packet originating from this node will have an empty list as the pkt.source_route)
            Your task is complete the routing algorithm using the helper functions given. Feel free to add
```

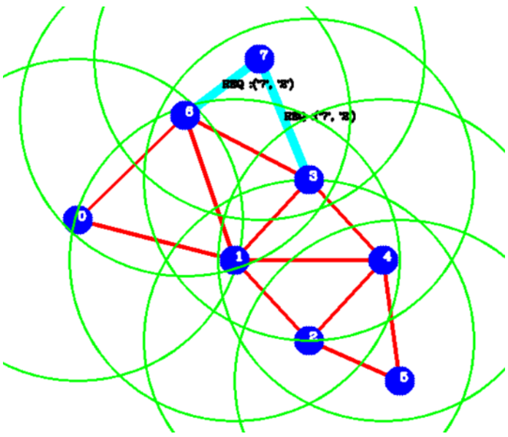
```
your own
            functions and make sure you add comments appropriately.

            If a packet is to be broadcasted or to be forwarded to another node it should be appended to the
queue_out.
            Take note next hop should give the index of the next node it must be forwarded in the source
route. Make sure you update
            the pkt.next_hop before appending to queue_out.
        """
        if pkt.type == PKT_TYPE.DPKT:
            if pkt.source_route!=[]:
                self.add_to_cache(pkt)
                self.add_to_queue_out(pkt)
            else:
                self.add_to_buffer(pkt)
                RREQ_pkt = self.generate_RREQ(pkt)
                self.add_to_queue_out(RREQ_pkt)


        elif pkt.type == PKT_TYPE.RREQ:
            if (self.check_in_recent(pkt)) or (pkt.check_id(self.id)):
                self.forward()
            elif pkt.target == self.id:
                if (self.check_in_recent(pkt)) or (pkt.check_id(self.id)):
                    self.forward()
                else:
                    self.add_to_recent(pkt)
                    RREP_pkt = self.generate_RREP(pkt)
                    self.add_to_queue_out(RREP_pkt)

            else:
                pkt.add_id(self.id)
                self.add_to_recent(pkt)
                self.add_to_queue_out(pkt)

        else:
            if self.id == pkt.source:
                self.add_to_cache(pkt)
                if self.check_in_buffer(pkt):
                    DATA_pkt = self.retrieve_from_buffer(pkt)
                    self.add_path_from_cache(DATA_pkt)
                    self.add_to_queue_out(DATA_pkt)

            else:
                self.add_to_queue_out(pkt)
```

Simulation Results

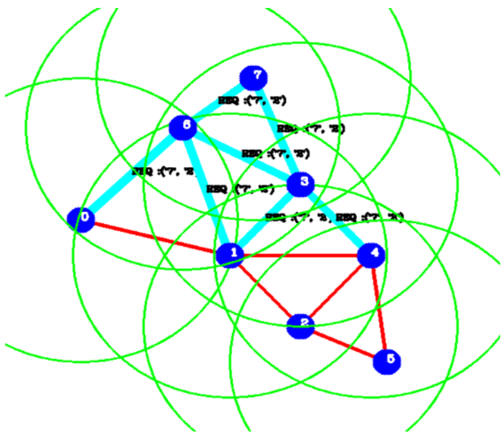
Number of nodes - 8 (0 to 7)

1) manet.send(7, 2, 1, 'Test 1') # send a data packet from 7 to 2 with time step 1

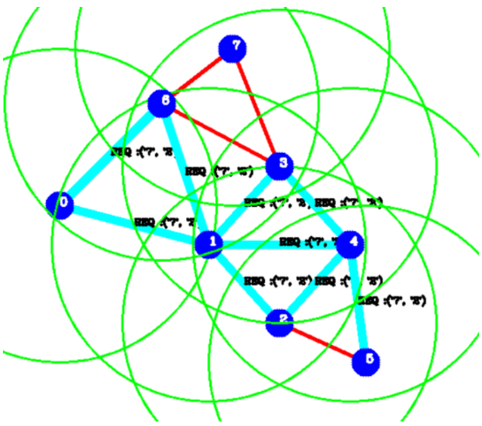
```
step 0:  []
step 1:  [('7', '3', 'REQ', ('7', '2')), ('7', '6', 'REQ', ('7', '2'))]
step 2:  [('3', '1', 'REQ', ('7', '2')), ('3', '4', 'REQ', ('7', '2')), ('3', '6', 'REQ', ('7', '2')),
('3', '7', 'REQ', ('7', '2')), ('6', '0', 'REQ', ('7', '2')), ('6', '1', 'REQ', ('7', '2')), ('6', '3',
'REQ', ('7', '2')), ('6', '7', 'REQ', ('7', '2'))]
step 3:  [('0', '1', 'REQ', ('7', '2')), ('0', '6', 'REQ', ('7', '2')), ('1', '0', 'REQ', ('7', '2')),
('1', '2', 'REQ', ('7', '2')), ('1', '3', 'REQ', ('7', '2')), ('1', '4', 'REQ', ('7', '2')), ('1', '6',
'REQ', ('7', '2')), ('4', '1', 'REQ', ('7', '2')), ('4', '2', 'REQ', ('7', '2')), ('4', '3', 'REQ', ('7
', '2')), ('4', '5', 'REQ', ('7', '2'))]
step 4:  [('2', '1', 'REP', ('7', '2')), ('5', '2', 'REQ', ('7', '2')), ('5', '4', 'REQ', ('7', '2'))]
step 5:  [('1', '3', 'REP', ('7', '2'))]
step 6:  [('3', '7', 'REP', ('7', '2'))]
step 7:  [('7', '3', 'DATA', ('7', '2'))]
step 8:  [('3', '1', 'DATA', ('7', '2'))]
step 9:  [('1', '2', 'DATA', ('7', '2'))]
step 10:  []
```



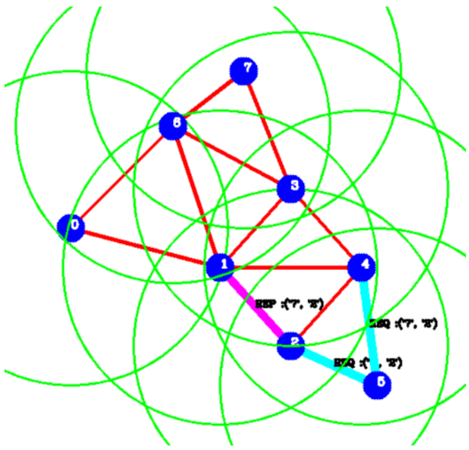
Step 1



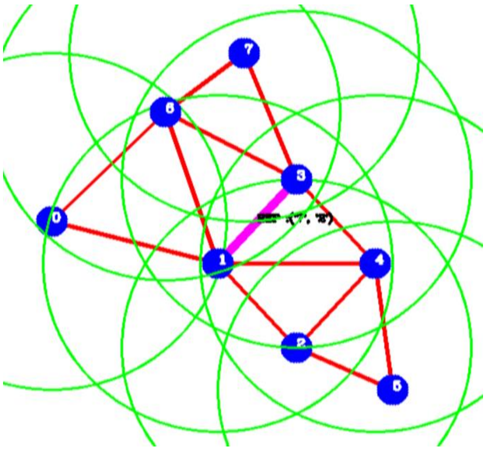
Step 2



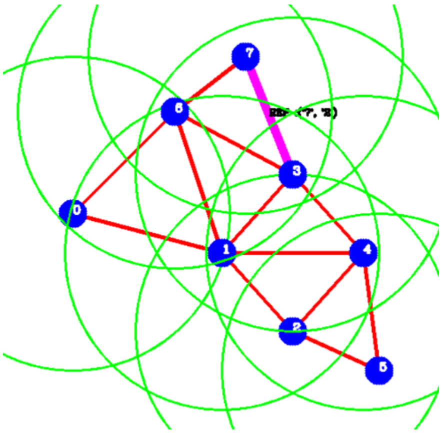
Step 3



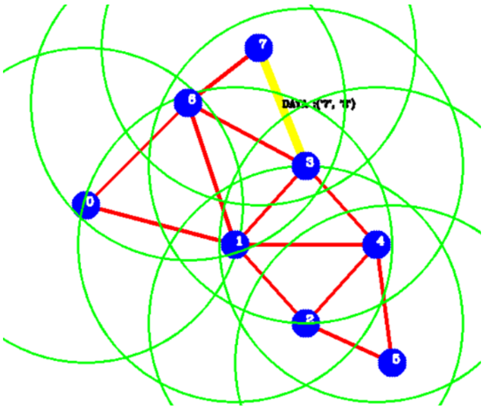
Step 4



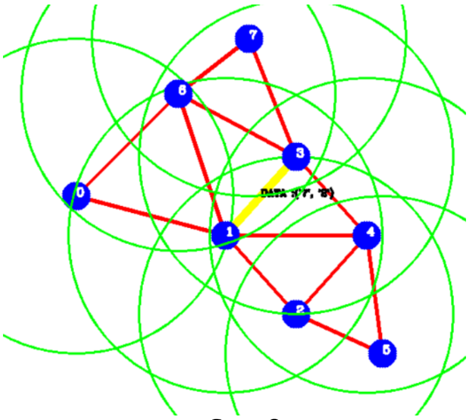
Step 5



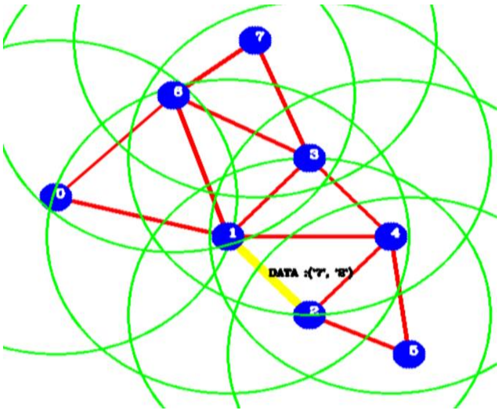
Step 6



Step 7



Step 8

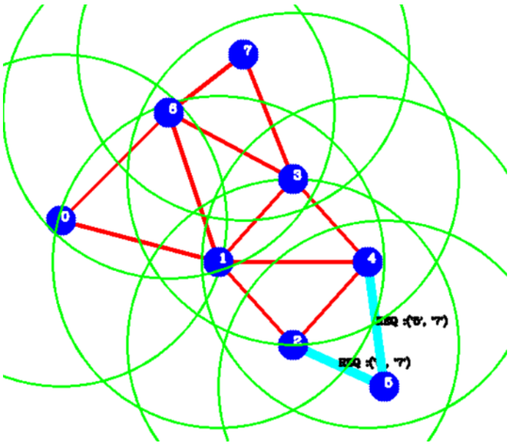


Step 9

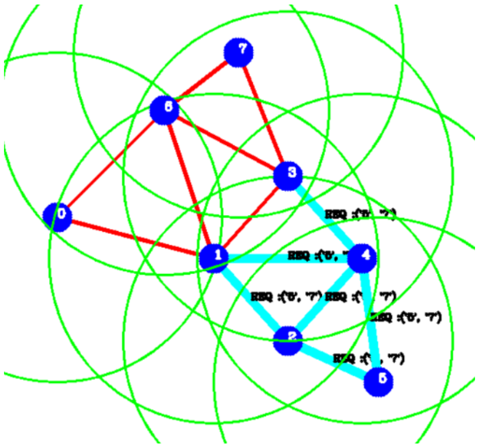
- Hops that passes the data packet = 3,1
- Number of hops for the data packet = 2

2) manet.send(5, 7, 7, 'Test 2') # send a data packet from 5 to 7 with time step 7

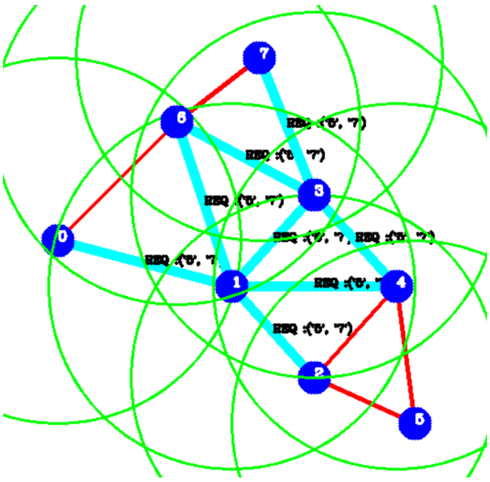
```
step 6:  []
step 7:  [('5', '2', 'REQ', ('5', '7')), ('5', '4', 'REQ', ('5', '7'))]
step 8:  [('2', '1', 'REQ', ('5', '7')), ('2', '4', 'REQ', ('5', '7')), ('2', '5', 'REQ', ('5', '7')), ('4',
'1', 'REQ', ('5', '7')), ('4', '2', 'REQ', ('5', '7')), ('4', '3', 'REQ', ('5', '7')), ('4', '5', 'REQ', ('5'
, '7'))]
step 9:  [('1', '0', 'REQ', ('5', '7')), ('1', '2', 'REQ', ('5', '7')), ('1', '3', 'REQ', ('5', '7')), ('1',
'4', 'REQ', ('5', '7')), ('1', '6', 'REQ', ('5', '7')), ('3', '1', 'REQ', ('5', '7')), ('3', '4', 'REQ', ('5'
, '7')), ('3', '6', 'REQ', ('5', '7')), ('3', '7', 'REQ', ('5', '7'))]
step 10:  [('0', '1', 'REQ', ('5', '7')), ('0', '6', 'REQ', ('5', '7')), ('6', '0', 'REQ', ('5', '7')), ('6',
'1', 'REQ', ('5', '7')), ('6', '3', 'REQ', ('5', '7')), ('6', '7', 'REQ', ('5', '7')), ('7', '3', 'REP', ('5'
, '7'))]
step 11:  [('3', '4', 'REP', ('5', '7'))]
step 12:  [('4', '5', 'REP', ('5', '7'))]
step 13:  [('5', '4', 'DATA', ('5', '7'))]
step 14:  [('4', '3', 'DATA', ('5', '7'))]
step 15:  [('3', '7', 'DATA', ('5', '7'))]
step 16:  []
```



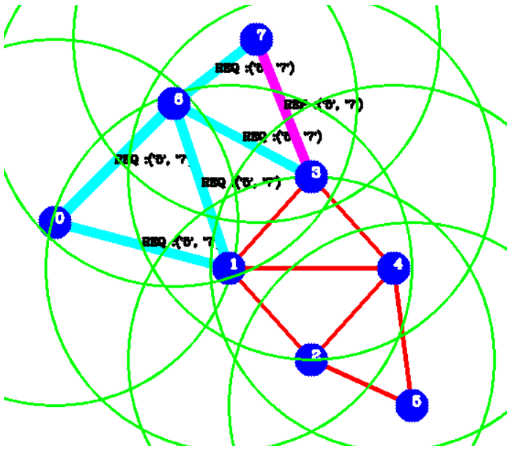
Step 7



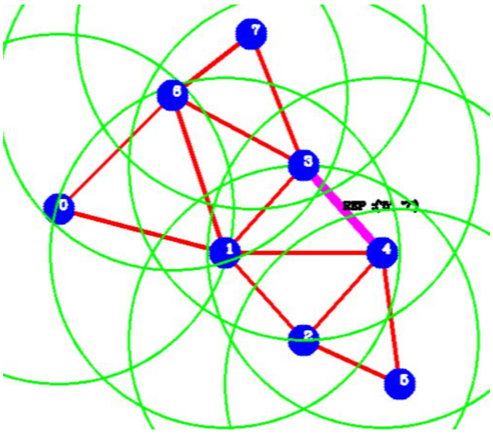
Step 8



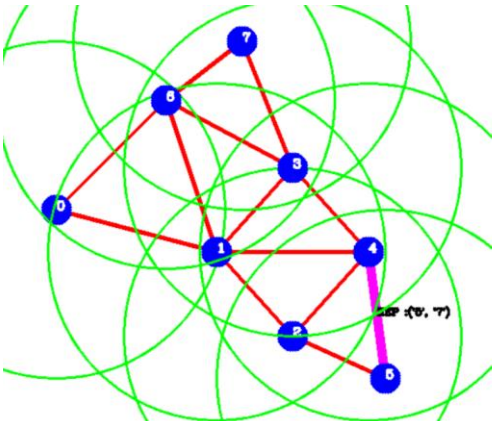
Step 9



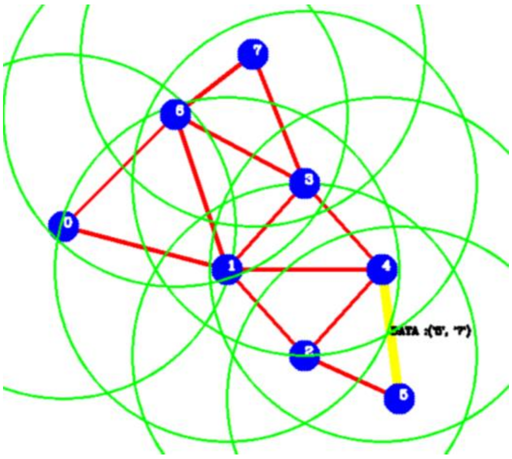
Step 10



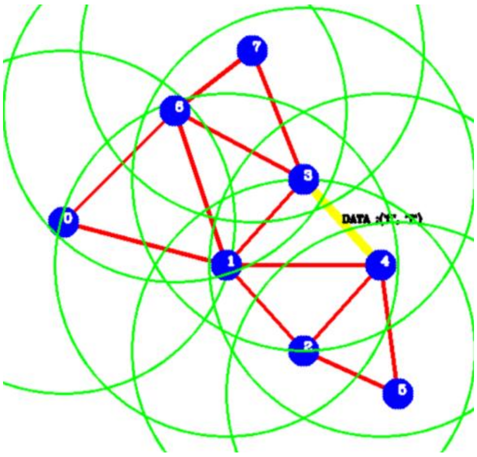
Step 11



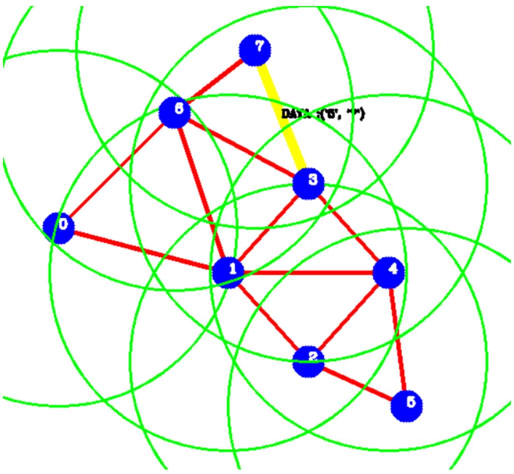
Step 12



Step 13



Step 14

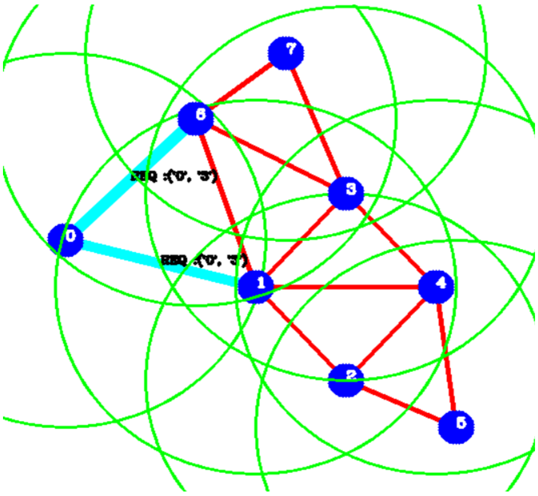


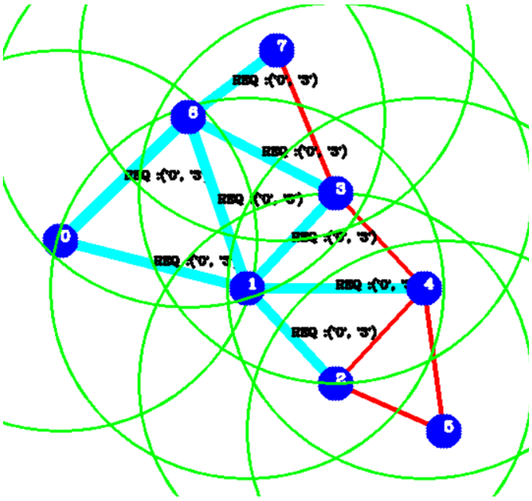
Step 15

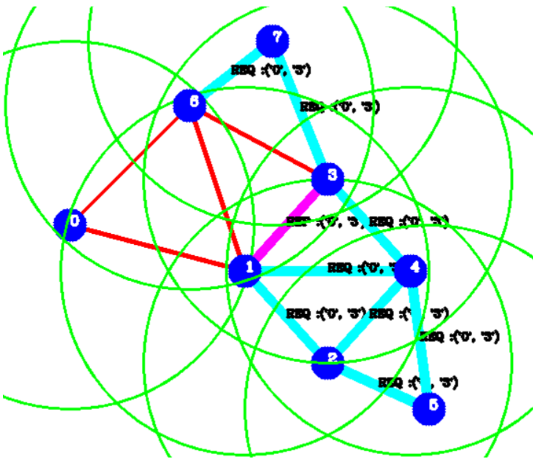
- Hops that passes the data packet = 4,3
- Number of hops for the data packet = 2

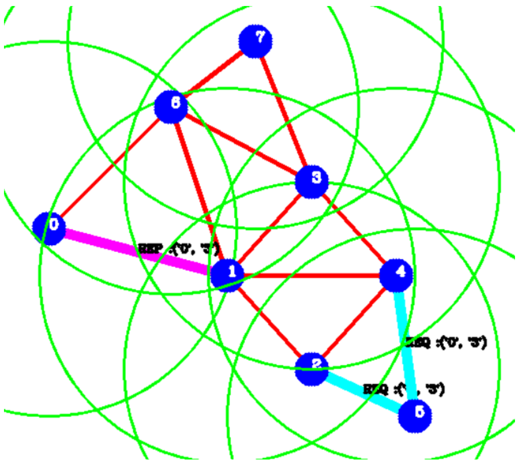
3) manet.send(0, 3, 15, 'Test 3') # send a data packet from 0 to 3 with time step 15

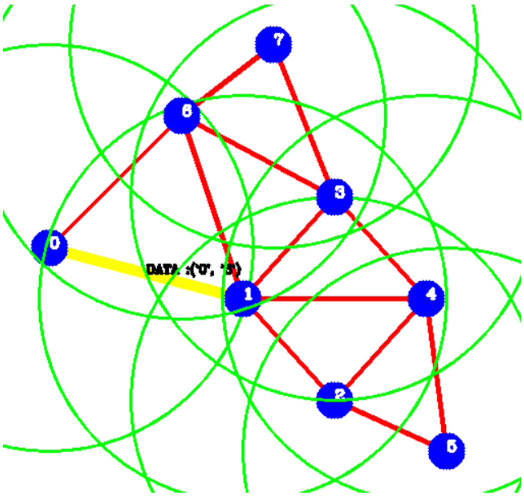
```
step 11:  []
step 12:  [('0', '1', 'REQ', ('0', '3')), ('0', '6', 'REQ', ('0', '3'))]
step 13:  [('1', '0', 'REQ', ('0', '3')), ('1', '2', 'REQ', ('0', '3')), ('1', '3', 'REQ', ('0',
'3')), ('1', '4', 'REQ', ('0', '3')), ('1', '6', 'REQ', ('0', '3')), ('6', '0', 'REQ', ('0', '3')
), ('6', '1', 'REQ', ('0', '3')), ('6', '3', 'REQ', ('0', '3')), ('6', '7', 'REQ', ('0', '3'))]
step 14:  [('2', '1', 'REQ', ('0', '3')), ('2', '4', 'REQ', ('0', '3')), ('2', '5', 'REQ', ('0',
'3')), ('3', '1', 'REP', ('0', '3')), ('4', '1', 'REQ', ('0', '3')), ('4', '2', 'REQ', ('0', '3')
), ('4', '3', 'REQ', ('0', '3')), ('4', '5', 'REQ', ('0', '3')), ('7', '3', 'REQ', ('0', '3')), (
'7', '6', 'REQ', ('0', '3'))]
step 15:  [('1', '0', 'REP', ('0', '3')), ('5', '2', 'REQ', ('0', '3')), ('5', '4', 'REQ', ('0',
'3'))]
step 16:  [('0', '1', 'DATA', ('0', '3'))]
step 17:  [('1', '3', 'DATA', ('0', '3'))]
step 18:  []
```


Step 12

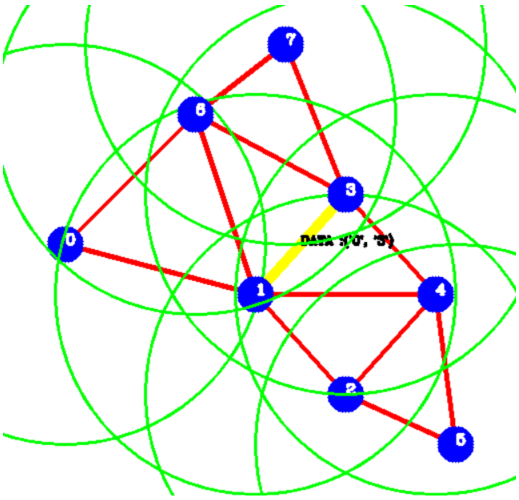

Step 13


Step 14


Step 15



Step 16



Step 17

 Hops that passes the data packet = 1
 Number of hops for the data packet = 1

Average Number of Hops per data packet = (2 + 2 + 1) / 3 ≈ 2

The way of improving the efficiency by exploiting the route cache

New routes can be identified by discovered source routes. As the nodes in discovered source routes have the ability of seeking the entire source route of a packet, they can identify new routes easily. These nodes are capable in adding the new routes to the local cache also.

The receiving hops can contain a route for the target in their cache for route requests. Instead of re-broadcasting they can append that route to the source route which in terms help in saving the time.

We can get rid of lengthy routes by limiting the number of hops that a packet can be propagated.

The way of handling the disconnections during the transmission

Routing overheads can be minimized by re-transmitting the same packet repeatedly until a route is discovered and limiting the rate of initiating new requests for the same packet.

Use of route error packets with promiscuous receive mode will help in handling the disconnections during transmission. In this process, if a node is unable to find a route, it can initiate a route error packet and send it backward. These route error packets contain both ends of the errored hop. Every node which receives this error packet can see this. So, they are capable in finding alternative routes and truncating the route from errored hop.

Use of passive acknowledgement will also help in handling the disconnections in transmission. Here, neighboring nodes can hear the transmission of other nodes. So, we are capable in maintaining a continuous path.

Dynamic Source Routing (DSR) Vs Distance Vector Routing (DVR)

Dynamic Source Routing (DSR)	**Distance Vector Routing (DVR)**
High Resource requirement	Minimum resource requirement
Multiple paths can be discovered.	A Single path is discovered.
Low power consumption: battery power is conservative. Host able to put their selves into sleep or standby modes when there is no task take place with them.	High power consumption because each and every host has to be waited for receptions every time in sending and receiving advertisements.
Packet sender will determine the complete list of nodes that the packet should be forwarded. It contains the route in the header of packets.	Each and every router will broadcast its view of the distance to all hosts and its neighboring routers. Each router will compute the shortest path to each host based on the information advertised by each of its neighboring routers.
Information on routes are kept and stored in the source.	Information on routes are kept and stored in the intermediate nodes.
Quick adaptation to routing changes when frequent host movement take places	Slow adaptation to routing changes when frequent host movement take places
No unnecessary bandwidth consumption.	Unnecessary bandwidth consumption due to periodic beaconing.

References

- https://www.mathworks.com/help/phased/examples/modeling-the-propagation-of-rf-signals.html
- www.wirelesscommunication.nl. (n.d.). *Free Space Propagation*. [online] Available at: http://www.wirelesscommunication.nl/reference/chaptr03/fsl.htm
- www.electronics-notes.com. (n.d.). *Radio Signal Path Loss » Electronics Notes*. [online] Available at: https://www.electronics-notes.com/articles/antennas-propagation/propagation-overview/radio-signal-path-loss.php.
- Kestwal, M.C., Joshi, S. and Garia, L.S. (2014). *Prediction of Rain Attenuation and Impact of Rain in Wave Propagation at Microwave Frequency for Tropical Region (Uttarakhand, India)*. [online] International Journal of Microwave Science and Technology. Available at: https://www.hindawi.com/journals/ijmst/2014/958498/.
- Wikipedia. (2019). *Dynamic Source Routing*. [online] Available at: https://en.wikipedia.org/wiki/Dynamic_Source_Routing
  [Accessed 12 Sep. 2020]
- Singh, M. and Thakur, S. (2014). Comparison of DSDV, DSR and ZRP Routing Protocols in MANETs. *International Journal of Computer Applications*, [online] 108(13), pp.975–8887. Available at: https://research.ijcaonline.org/volume108/number13/pxc3899548.pdf#:~:text=The%20study%20of%20these%20routing
  [Accessed 12 Sep. 2020]