Department of Electronic and Telecommunication Engineering

University of Moratuwa, Sri Lanka

EN 2053 - Communication Systems and Networks

Modelling Propagation Losses and MANETs

Assignment 2

Submitted by

Thalagala B.P.	180631J
Sauranga H.W.C.	180574K

Submitted on

September 13, 2020

Contents

**** PDF is clickable***

Note:

*All the executable codes, a simulink model and a voice sample are included in a separate folder named as **EN2053-Assignment 2-180574K-180631J.rar** which was submitted along with this report.*

*Before run the **VoiceTransmissionBaseband.slx** simulink simulation related to the part 5 of the Task 1 ,(included in a separate folder named as **Task1-RFPM** inside the above folder) change the directories related to the voice input and output. Voice sample named as **voice.wav** is also included in the same folder.*

Additionally all the materials related to Task 1 can also be found at https://github.com/bimalka98/RF-Propagation-Model

1 Modeling the RF propagation Using Matlab

1.1 Relationship between Free Space Path Loss and Frequency

Consider following meanings for the parameters

P_{RX}	= Received Power at the Receiving Antenna
P_{TX}	= Transmitted Power at the Transmitting Antenna
f	= Frequency of the wave in Hz
f_{GHz}	= Frequency of the wave in GHz
d	= Distance between the antennas in m
d_{km}	= Distance between the antennas in km
G_{TX}	= Directive gain of the Transmitter
G_{RX}	= Directive gain of the Receiver
c	= Velocity of the electromagnetic waves in a vacuum

The relationship between above parameters can be given as follows

$$P_{RX} = P_{TX} \cdot \frac{c^2}{(4\pi.f.d)^2} .G_{TX}.G_{RX}$$

From the above equation, free space path loss, say L

$$L = \frac{(4\pi.f.d)^2}{c^2}$$

By considering $10.log_{10}()$ in both sides, Free Space Path Loss in dB, say L_{dB}

$$\begin{aligned}
10.\log_{10}(L) &= 10.\log_{10}(\frac{(4\pi.f.d)^2}{c^2}) \\
L_{dB} &= 10.\log_{10}((4\pi.f.d)^2) - 10.\log_{10}(c^2) \\
&= 20.\log_{10}(4\pi.f.d) - 20.\log_{10}(c) \\
&= 20.\log_{10}(4\pi) - 20.\log_{10}(c) + 20.\log_{10}(f) + 20.\log_{10}(d) \\
&= 20.\log_{10}(\frac{4\pi}{c}) + 20.\log_{10}(f) + 20.\log_{10}(d) \\
&= -147.5522168 + 20.\log_{10}(f_{GHz}.10^9) + 20.\log_{10}(d_{km}.10^3) \\
&= -147.5522168 + 20.\log_{10}(10^9) + 20.\log_{10}(f_{GHz}) + 20.\log_{10}(10^3) + 20.\log_{10}(d_{km}) \\
&= -147.5522168 + 180 + 20.\log_{10}(f_{GHz}) + 60 + 20.\log_{10}(d_{km}) \\
&= -147.5522168 + 240 + 20.\log_{10}(f_{GHz}) + 20.\log_{10}(d_{km}) \\
&= +92.44778322 + 20.\log_{10}(f_{GHz}) + 20.\log_{10}(d_{km})
\end{aligned}$$

Since transmitter and receiver are located at distance of 10km apart, by substituting $d_{km} = 10$.

Free Space Path Loss in dB, L_{dB} as a function of frequency in Giga Hertz

$$L_{dB}(f_{GHz}) = +112.44778322 + 20.\log_{10}(f_{GHz})$$

Note : Axes of the following plots are given in the logarithmic scale and range of frequency was chosen from 50 GHz to 1000 GHz since some of the ITU-R models are only defined in the 10 GHz-1000 GHz range.

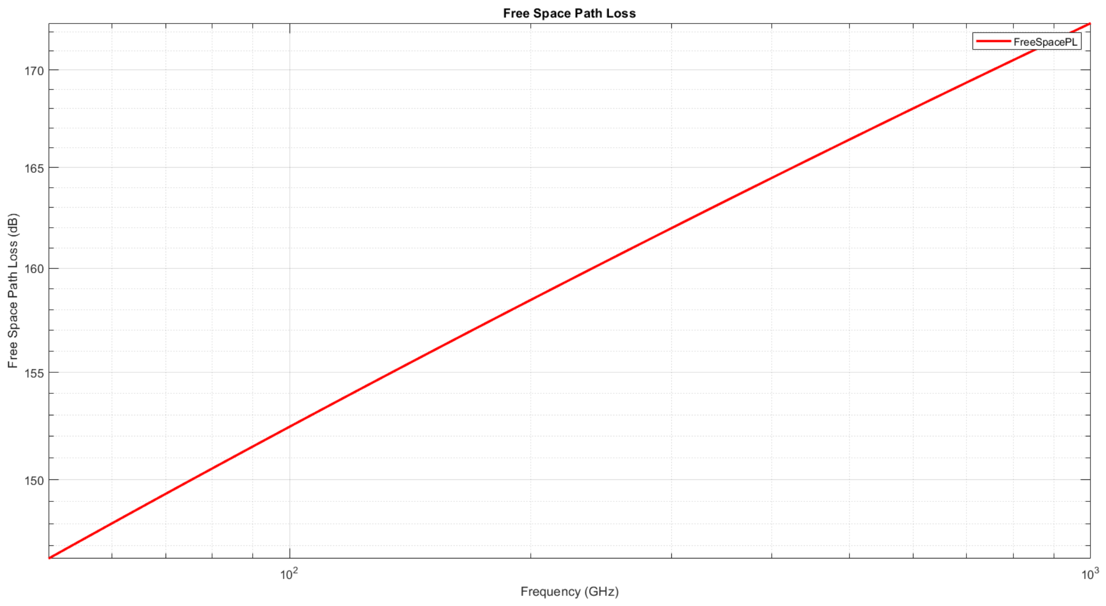


Figure 1: Relationship between Free Space Path Loss and Frequency

1.2 Rain attenuation, Fog attenuation and Atmospheric gas attenuation with Frequency

*Note : For the generation of following plots three of the Matlab built-in functions, namely **rainpl()**[3], **gaspl()**[3], **fogpl()**[3] which are developed according to the ITU-R P Series recommendations were used and links for their documentations are given at the Reference section.*

1.2.1 Rain attenuation - Recommendation ITU-R P.838-3, 2005[5]

The following plot shows how losses due to rain varies with frequency. The plot assumes the followings in addition to the provided information in the Task 1.

Elevation angle of the propagation path	$= 0$
Polarization tilt angle of the signal	$= 0$

In general, horizontal polarization represents the worse case for propagation loss due to rain.

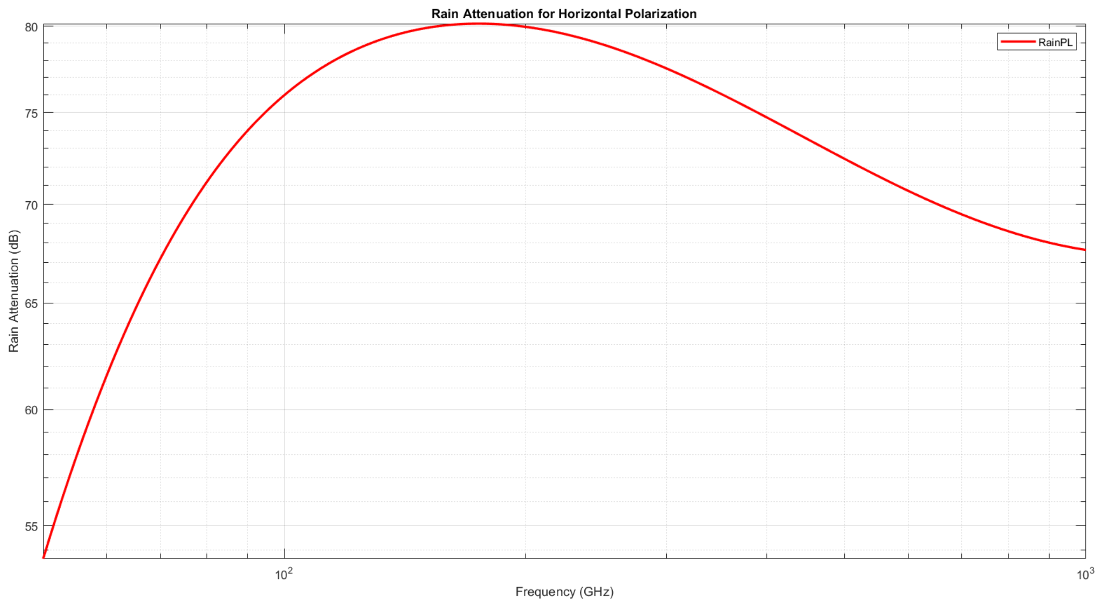


Figure 2: Relationship between Rain attenuation and Frequency

1.2.2 Fog attenuation - Recommendation ITU-R P.840-3, 2013[6]

The following plot shows how losses due to fog/cloud varies with frequency. The plot assumes the following provided information in the Task 1.

Ambient Temperature in Celsius	= 31
Liquid Water Density in g/m^3	= 0.5

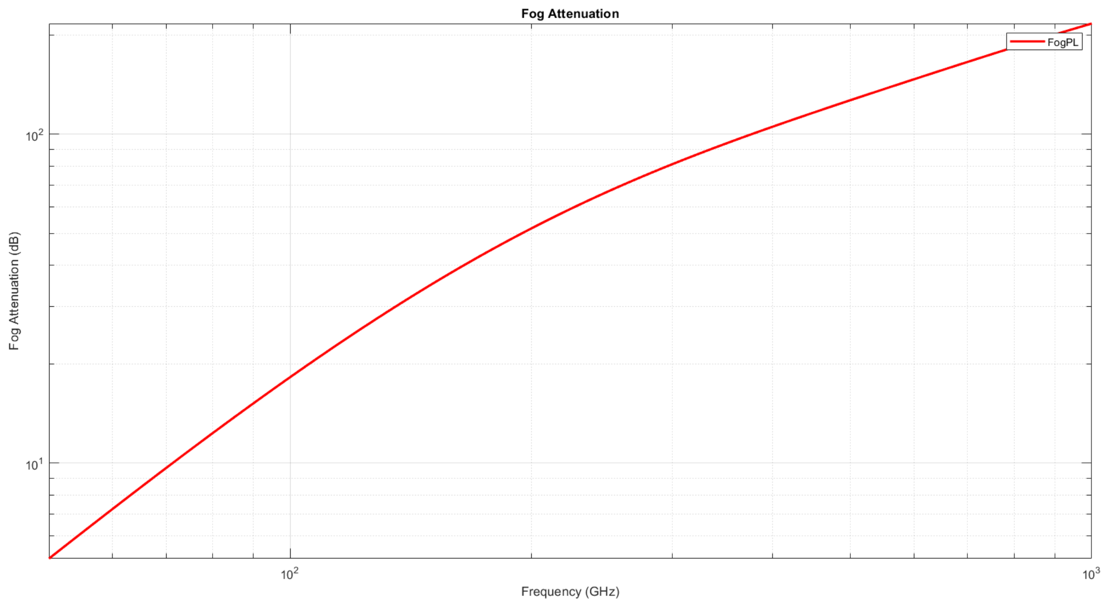


Figure 3: Relationship between Fog attenuation and Frequency

1.2.3 Atmospheric gas attenuation - Recommendation ITU-R P.676-10, 2013[4]

The plot below shows how the propagation loss due to atmospheric gases varies with the frequency. The plot assumes the followings in addition to the provided information in the Task 1.

Dry air pressure in Pa	= 101325
Water Vapor Density in g/m^3	= 30.4[7]

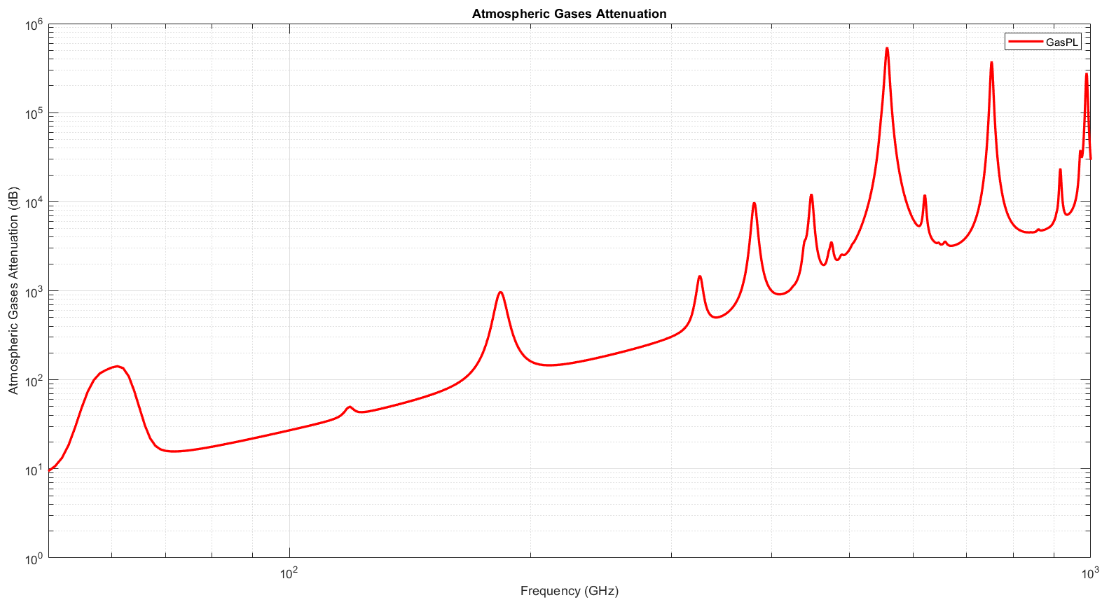


Figure 4: Relationship between Atmospheric gas attenuation and Frequency

1.3 Total Path Loss with Frequency

Note : Range of frequency was chosen from 50 GHz to 1000 GHz since some of the ITU-R models are only defined in 10 GHz - 1000 GHz range.

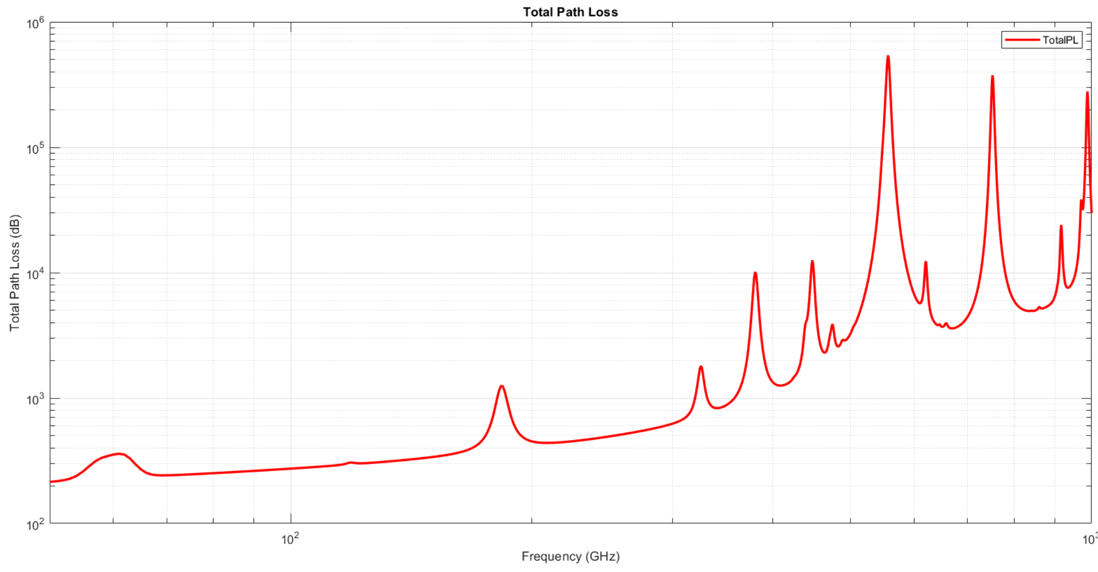


Figure 5: Relationship between Total Path Loss and Frequency

By inspecting the figure we can conclude that the minimum propagation loss is given at the frequency of 50 GHz in the given range. Therefore from this point onward, for the calculations it will be the frequency for transmission.

Minimum Propagation Loss = 214.624 dB
Corresponding Frequency = 50 GHz

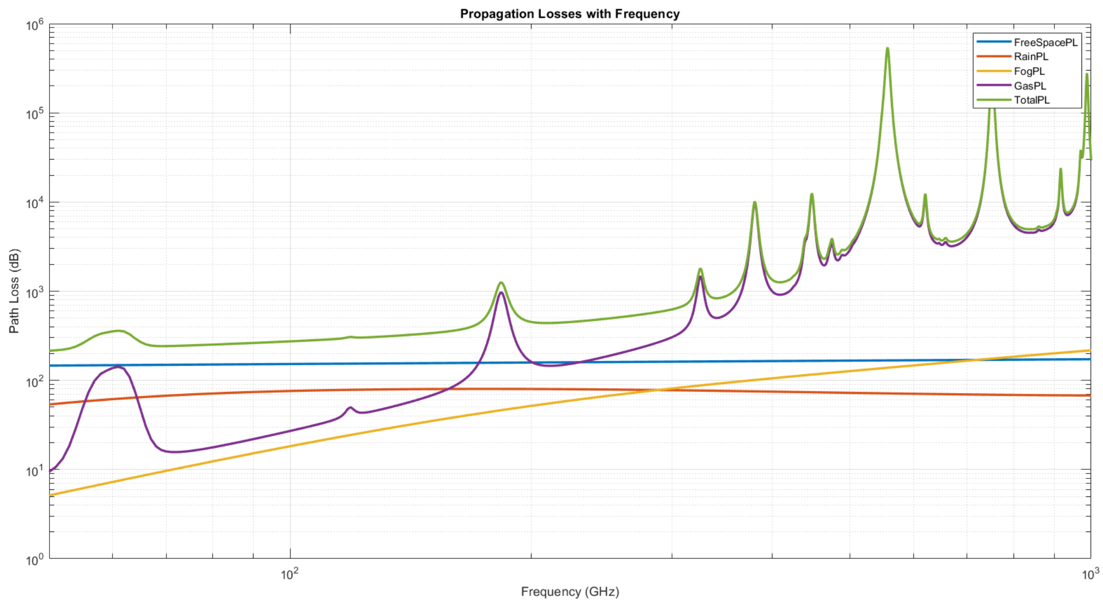


Figure 6: Relationship between Various Path Losses and Frequency - All in One

1.4 Variation of the Signal Power with the Distance

Parameters For the propagation model

Chosen Carrier frequency	50 GHz
Transmission power	50 kW or 47 dB
Cable loss at Transmitter	3 dB
Transmitter Gain	30 dB
Receiver Gain	24.77 dB
Cable loss at Receiver	4 dB
Total Path Loss	Varies with Distance

According to above values, Let's calculate the Power of the signal when leaving the Transmission antenna, say $P_{dB}(0\ km)$,

$$\begin{aligned} P_{dB}(0\ km) &= Transmission\ power - Cable\ loss\ at\ Transmitter + Transmitter\ Gain \\ &= 47 - 3 + 30 \\ &= 74\ dB \end{aligned}$$

Free Space Path Loss in dB, L_{dB} as a function of distance in kilo meters. By substituting $f_{GHz} = 50$ to the equation derived in part 1.

$$\begin{aligned} L_{dB}(d_{km}) &= +92.44778322 + 20.\log_{10}(50) + 20.\log_{10}(d_{km}) \\ &= +92.44778322 + 33.97940009 + 20.\log_{10}(d_{km}) \\ &= +126.4271833 + 20.\log_{10}(d_{km}) \end{aligned}$$

Therefore,

$$Total\ Path\ Loss = L_{dB}(d_{km}) + Rain\ Attenuation + Fog\ Attenuation + Atmospheric\ Gas\ Attenuation$$

Therefore the Signal Power when reaching the Receiving Antenna at d_{km} distance,

$$P_{dB}(d_{km}) = 74\ dB - Total\ Path\ Loss$$

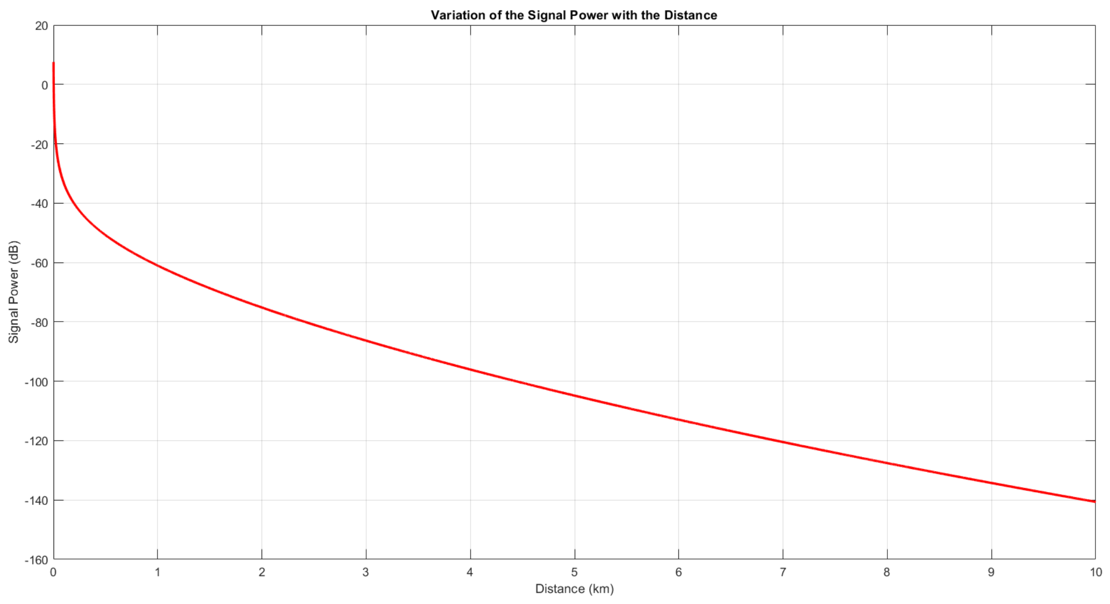


Figure 7: Variation of the Signal Power with the Distance

1.5 Transmitting a voice signal over a noisy channel using the above Transmission frequency and the Propagation model.

1.5.1 Logic and Assumptions Used to implement the Propagation Model

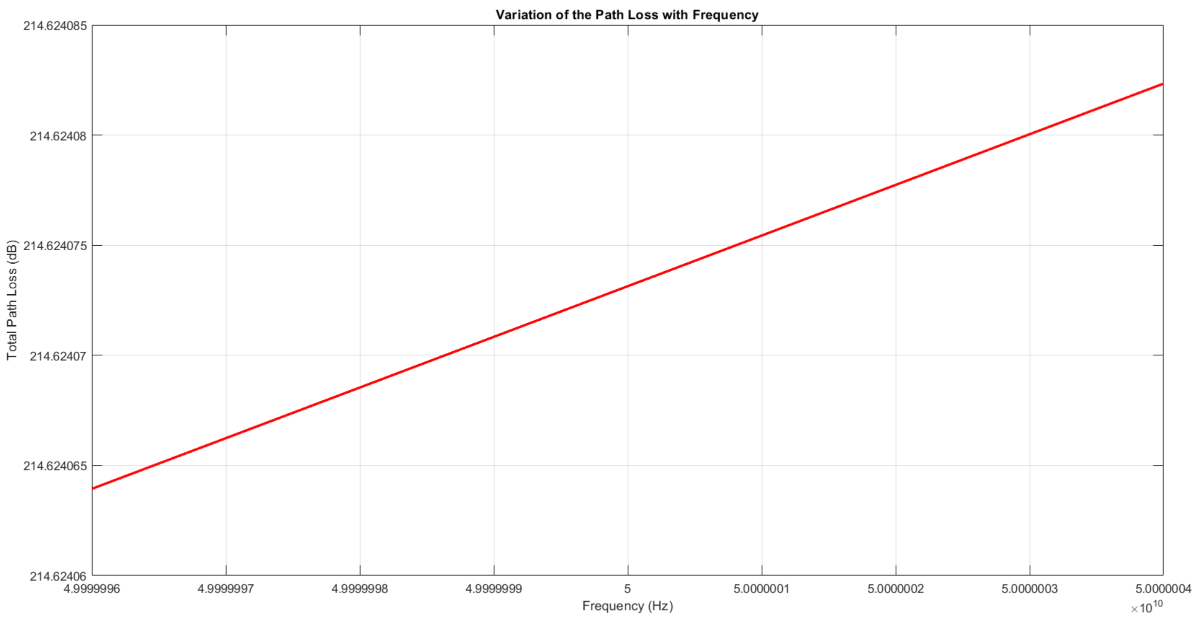


Figure 8: Variation of the Total Path Loss with the Frequency in modulated Voice Signal

By inspecting the above figure, it can be observed that the total Path Loss of the **Frequency Modulated Voice Signal**, over its frequency range(*from 50e9 - 4e3 to 50e9 + 4e3*)Hz changes only by a value less than 0.000025 dB which is almost Zero. Therefore in a real world application it can be assumed that the total path loss of the modulated voice signal is as same as that of the carrier wave(50 GHz) alone. That is, path loss variation due to the frequency variation in the above range can be neglected and can be assumed as a constant of 214.6240 dB which is the total path loss corresponding to the 50 GHz wave. Therefore for the following model, total path loss of the signal was taken as 214.6240 dB and it was included in the ***Free Space Loss block*** .

In addition to that since the attenuation affects only to the amplitude of the signal, actual scenario of the Frequency Modulation of the voice signal was not considered and instead the voice signal was FM modulated using the ***FM Modulator Baseband block*** which is an ideal option for simulation purposes.

1.5.2 RF Propagation Model - Simulink

Steps of the simulation are as follows:

1. Input the voice.wav file of 8000Hz sampling rate using ***From Multimedia File block***.

2. Normalization of the voice signal using ***Normalization block***.

3. Frequency modulating the signal using ***FM Modulator Baseband block***.

4. Addition of the -3 dB Cable loss at Transmitter using ***dB Gain block***.

5. Addition of the 30 dB transmitter gain using the ***Transmitter block***.

6. Addition of the effect of noise using ***AWGN channel block***.

7. Addition of the -214.6240 dB propagation loss using ***Free Space Path Loss block***.
8. Addition of the 24.77 dB receiver gain using ***Receiver Preamp block***.
9. Addition of the -4 dB Cable loss at the Receiver using ***dB Gain block***.
10. Demodulating the signal using ***FM Demodulator Baseband block***.
11. Saving the output voice signal using ***To Multimedia File block***.
12. convert the output signal into audio using ***Audio Device Writer block***.

Before run the simulation change the directories related to the voice input and output as mentioned in the page 1.

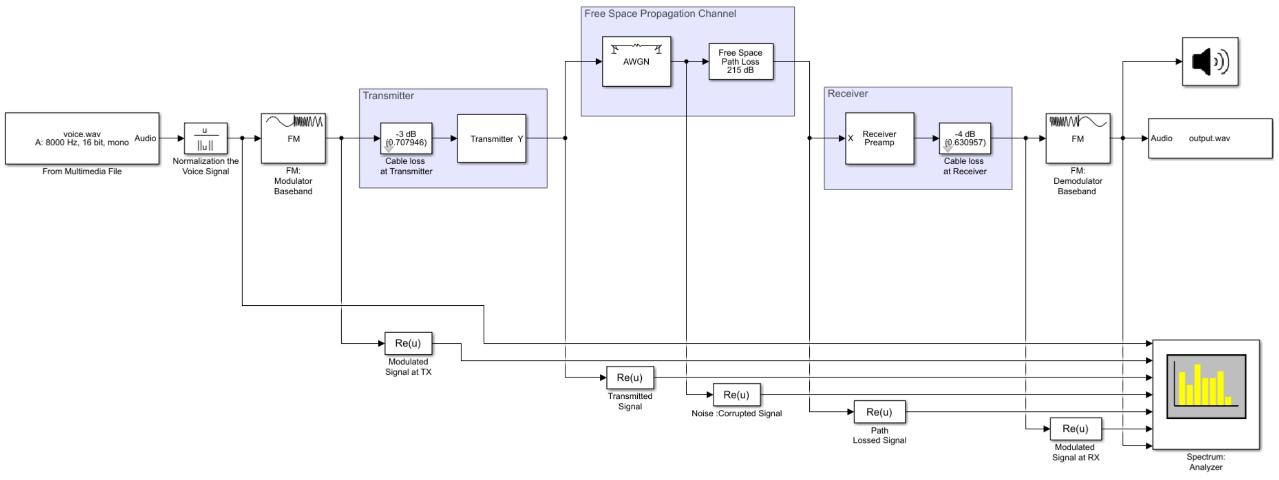


Figure 9: RF Propagation Model - Simulink

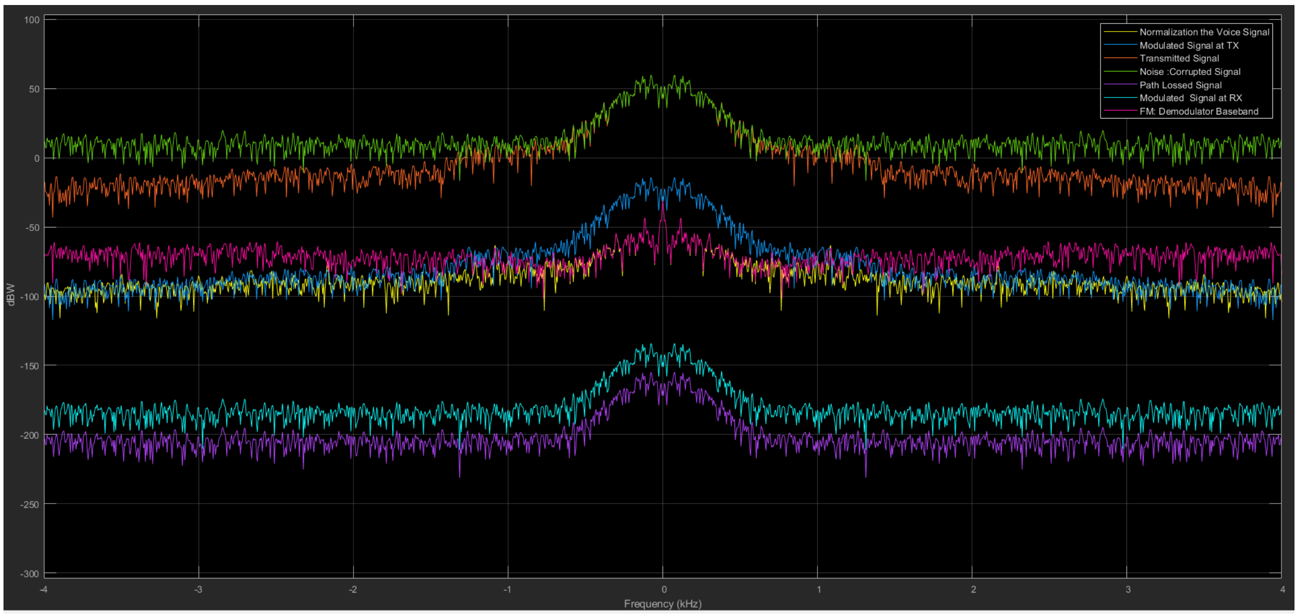


Figure 10: Frequency Spectrum of the signal at Various States

1.6 Codes for Task 1

```
%% Initialization
clear; close all; clc
%% ========= Free Space Propagation Loss with Frequency =========

%Defining the frequency range in GigaHertz
f_GHz = 50:1000;
%Free Space Path Loss Model obtained through calculations
freeSpaceLoss1 = 112.44778322 + 20*log10(f_GHz);

% Plotting Data
plotCurve(freeSpaceLoss1, 'FreeSpacePL')

%% == Rain, Fog, Atmospheric Gases Attenuations with Frequency ==

freq = f_GHz*1e9;% Defining the frequency range in Hertz
range = 10e3;    % Distance between transceivers in m
rainrate = 20;   % Rain rate in mm/h
elev = 0;        % Elevation angle of the propagation path
tau = 0;         % Polarization tilt angle of the signal
temp = 31;       % Ambient Temperature in celcious
dens = 0.5;      % Liquid Water Density in g/m^3
rou = 30.4;      % Water Vapor Density in g/m^3
p =  101325;     % Atmospheric Pressure in Pa at sea level

% Calculating Attenuations
rainAttenuation = rainpl(range,freq,rainrate,elev,tau);
fogAttenuation = fogpl(range,freq,temp,dens);
gasAttenuation = gaspl(range,freq,temp, p, rou);

% Plotting Data
plotCurve(rainAttenuation, 'RainPL');
plotCurve(fogAttenuation, 'FogPL');
plotCurve(gasAttenuation, 'GasPL');

%% ============ Total Propagation Loss with Frequency ============

% Calculating Total Attenuation
Totalpathloss = freeSpaceLoss1 + rainAttenuation + ...
                                fogAttenuation +gasAttenuation;
% Plotting Data
plotCurve(Totalpathloss, 'TotalPL');

%% ======= Variation of the Signal Power with the Distance =======

distance = 0:10e3; % Distance between transceivers in m
freq = 50*1e9;     % Choosen frequency value in Hertz

% Calculating Attenuations with Distance
freeSpaceLoss2 = 126.4271833 + 20*log10(distance/(10e2));
rainAttenuation = rainpl(distance,freq,rainrate,elev,tau);
fogAttenuation = fogpl(distance,freq,temp,dens);
gasAttenuation = gaspl(distance,freq,temp, p, rou);

% Total Path Loss with Distancce
```

```matlab
TotalLosswithDistance = freeSpaceLoss2' + rainAttenuation + ...
                                fogAttenuation +gasAttenuation;

% Calculating the signal Power with the distance
signalPower =  74 - TotalLosswithDistance;

% Plotting Data
figure;
plot(distance/10e2, signalPower, 'r','LineWidth', 2);
grid on;
xlabel('Distance (km)');
ylabel('Signal Power (dB)');
title('Variation of the Signal Power with the Distance');

fprintf('Program paused. Press enter to continue.\n');
pause;

%% ==Sending Voice Signal Over a Noisy Channel - Associated Logic==

freqDeviation = 4000; % Frequency Deviation of the Voice signal
CarrierFreq = 50e9;   % Carrier Frequency

% Frequency range of the Transmitted Signal
freqRange = CarrierFreq - freqDeviation :...
                    CarrierFreq + freqDeviation;

% Calculating Losses
freeSpaceLoss3 = 112.44778322 + 20*log10(freqRange/(1e9));
rainAttenuation = rainpl(range,freqRange,rainrate,elev,tau);
fogAttenuation = fogpl(range,freqRange,temp,dens);
gasAttenuation = gaspl(range,freqRange,temp, p, rou);

% Total Path Loss in the given Frequency Range
TotalPathLoss = freeSpaceLoss3 + rainAttenuation + ...
                                fogAttenuation +gasAttenuation;
% Plotting Data
figure;
plot(freqRange, TotalPathLoss, 'r', 'LineWidth', 2);
grid on;
xlabel('Frequency (Hz)');
ylabel('Total Path Loss (dB)');
title('Variation of the Path Loss with Frequency');
```

```matlab
function [] = plotCurve(inputArg1,inputArg2)
% Function to plot the Curves

f_GHz = 50:1000;%Defining the frequency range in GigaHertz
figure;
loglog(f_GHz, inputArg1, 'r', 'LineWidth', 2);
grid on;
xlabel('Frequency (GHz)');

if strcmp(inputArg2, 'FreeSpacePL')
    ylabel('Free Space Path Loss (dB)');
    title('Free Space Path Loss');

elseif strcmp(inputArg2, 'RainPL')
     ylabel('Rain Attenuation (dB)');
     title('Rain Attenuation for Horizontal Polarization');

elseif strcmp(inputArg2, 'FogPL')
     ylabel('Fog Attenuation (dB)');
     title('Fog Attenuation');

elseif strcmp(inputArg2, 'GasPL')
     ylabel('Atmospheric Gases Attenuation (dB)');
     title('Atmospheric Gases Attenuation');

elseif strcmp(inputArg2, 'TotalPL')
     ylabel('Total Path Loss (dB)');
     title('Total Path Loss');
end

legend(inputArg2);
%saveas(gcf,strcat(inputArg2,'.png'));
fprintf('Program paused. Press enter to continue.\n');
pause;
end
```

2 Implementing a Simplified Version of the Dynamic Source Routing(DSR) Protocol in Ad Hoc Wireless Networks

2.1 Simulation of a MANET with 6 nodes

For the simulation of the Dynamic Source Routing(DSR) Protocol, we have implemented a MANET with the aid of the code base provided in the assignment.

2.1.1 Codes for Task 2

We were tasked with completing the routing function in 'node.py' and the completed routing function is as follows.

```
"""
    ------------------------------------------------------------------------------
    node.py
    ------------------------------------------------------------------------------

"""
def route(self, pkt):

    if pkt.type == PKT_TYPE.RREQ:     #checking whether the packet is a
                                      #route request
        if (self.check_in_recent(pkt)) or (pkt.check_id(self.id)):
         #checking whether the packet has already passed through the node
            self.disregard += 1  #if it has already been processed,
                                 #it is counted as a disregarded packet
            self.forward()   #proceeding onto the next packet
        else:    #if the packet is received for the first time
            self.add_to_recent(pkt)      #adding to the recent packet list
                                         #of the node
            if pkt.target == self.id:    #checking whether this node
                                         #is the destination of the packet
                rrepPkt = self.generate_RREP(pkt)    #creating
                                                     #a route reply packet
                self.add_to_queue_out(rrepPkt)   #sending the packet
            else:    #if the packet is not addressed to the considered node
                pkt.add_id(self.id)      #appending the id of this node
                                         #to the packet route
                self.add_to_queue_out(pkt)   #sending the packet

    elif pkt.type == PKT_TYPE.RREP:   #checking whether the packet is a
                                      #route reply packet
        if pkt.source == self.id:     #checking whether the packet
                                      #has reached the source
            self.add_to_cache(pkt)    #getting the route from the packet
                                      #into the cache
            if self.check_in_buffer(pkt):    #checking whether the data
                                             #packet is available in buffer
                dataPkt = self.retrieve_from_buffer(pkt)#getting the data
                                                        #packet from the
                                                        #buffer(as dataPkt)
                self.add_path_from_cache(dataPkt)    #adding the route to
                                                     #the data packet
                self.add_to_queue_out(dataPkt)   #sending the packet
        else:
            self.add_to_queue_out(pkt)    #forwarding the route reply packet
                                          #until it reaches the source

    elif pkt.type == PKT_TYPE.DPKT:   #checking whether the packet is a
                                      #data packet
        if self.id == pkt.source:     #checking whether the packet is
                                      #origintaing from the considered node
            if self.check_in_cache(pkt): #checking for the path for
                                         #the packet in the node cache
                self.add_path_from_cache(pkt)    #adding the path from
```

```
                                                      #the node cache
                self.add_to_queue_out(pkt)    #sending the packet
            else:    #if the path for the packet is not available in cache
                self.add_to_buffer(pkt)  #keep the data packet in a buffer
                rreqPkt = self.generate_RREQ(pkt)     #creating a route
                                                      #request packet
                self.add_to_recent(rreqPkt)  #adding the route request
                                             #packet to the recents list
                                             #of the node
                self.add_to_queue_out(rreqPkt)    #sending the packet
        elif self.id == pkt.target:  #checking whether the data packet
                                     #has reached its destination
            self.received.append(pkt)    #saving the received data packet
        else:
             self.add_to_queue_out(pkt)
```

In addition, we have also created 'results.py' which we have used to interpret the final evaluation results of the simulation.

```
"""
    ------------------------------------------------------------------------------
    results.py
    ------------------------------------------------------------------------------

"""

from .packet import PKT_TYPE

class Results:
    def __init__(self):
        self.dpkt_count = 0
        self.total_hops = 0
        self.counted_dpkts = []

    def counter(self,pkt):
        """
        Checking the ID of packets to avoid duplicates
        and counting the number of data packets
        """
        self.total_hops += 1
        if (pkt.id) not in (self.counted_dpkts):
            self.counted_dpkts.append(pkt.id)
            self.dpkt_count += 1

    def print_results(self):
        """
        Printing the results
        """
        print ("Number of data packets :",self.dpkt_count)
        print("Total hops :",self.total_hops)
        print(
            "Average no. of hops per data packet :",
            (self.total_hops/self.dpkt_count)
            )
```

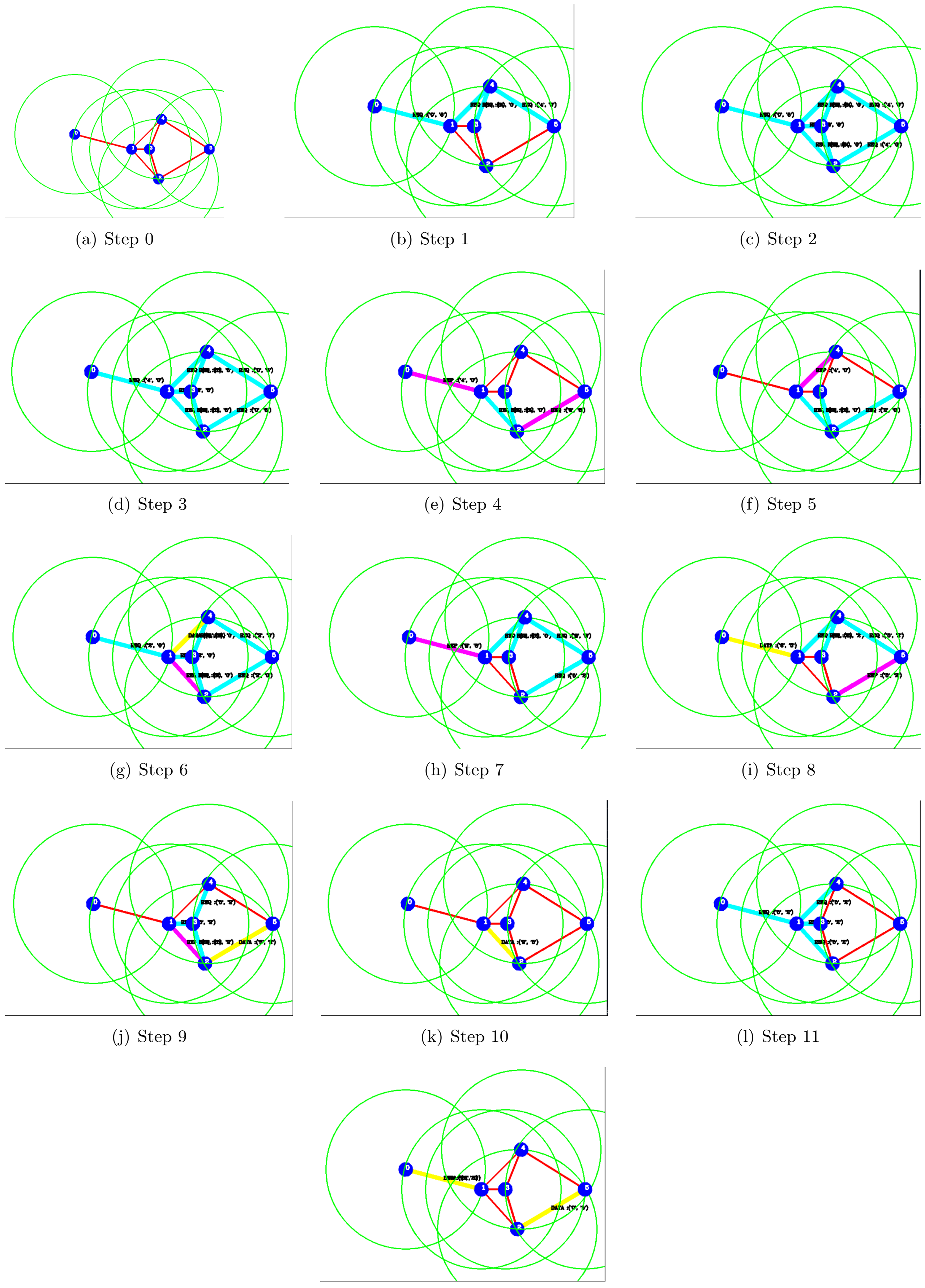
Finally, the ‘demo.py’ file which was used to run the simulation is given below. We have used a MANET consisting of 6 nodes We have simulated for the transmission of 4 data packets.

```
"""
    ------------------------------------------------------------------------------
    demo.py
    ------------------------------------------------------------------------------

"""

from simulator.graph import Graph
from simulator.visualizer import step
import cv2

manet = Graph()

# initialize network

tx_range = 200
dynamic = False  from simulator.graph import Graph
from simulator.visualizer import step
import cv2

manet = Graph()

# initialize network


tx_range = 200
dynamic = False  # True to use mobile model

no_of_hops =0

manet.add_node(500, 550, tx_range)  # "Add node 0 at 500,500"
manet.add_node(690, 600, tx_range)  # "Add node 1 at 690,600"
manet.add_node(780, 700, tx_range)  # "Add node 2 at 780,700"
manet.add_node(750, 600, tx_range)  # "Add node 3 at 750,600"
manet.add_node(790, 500, tx_range)  # "Add node 4 at 790,500"
manet.add_node(950, 600, tx_range)  # "Add node 5 at 950,600"

manet.send(0, 5, 1, 'Test')  # send a data packet at time step 1
                             # from node 0 to 5
manet.send(4, 0, 1, 'Test')  # send a data packet at time step 1
                             # from node 4 to 1
manet.send(2, 0, 3, 'Test')  # send a data packet at time step 3
                             # from node 2 to 0
manet.send(4, 2, 7, 'Test')  # send a data packet at time step 7
                             # from node 4 to 2
for t in range(15):  # Simulate for 15 time steps
    step(manet, t, dynamic)

manet.results.print_results()

cv2.destroyAllWindows()
```

2.1.2 Terminal Output of the Simulation

```
step 00:  []

step 01:  [('0', '1', 'REQ', ('0', '5')), ('4', '1', 'REQ', ('4', '0')),
  ('4', '3', 'REQ', ('4', '0')), ('4', '5', 'REQ', ('4', '0'))]

step 02:  [('1', '0', 'REQ', ('0', '5')), ('1', '2', 'REQ', ('0', '5')),
  ('1', '3', 'REQ', ('0', '5')), ('1', '4', 'REQ', ('0', '5')),
  ('3', '1', 'REQ', ('4', '0')), ('3', '2', 'REQ', ('4', '0')),
  ('3', '4', 'REQ', ('4', '0')), ('5', '2', 'REQ', ('4', '0')),
  ('5', '4', 'REQ', ('4', '0'))]

step 03:  [('1', '0', 'REQ', ('4', '0')), ('1', '2', 'REQ', ('4', '0')),
  ('1', '3', 'REQ', ('4', '0')), ('1', '4', 'REQ', ('4', '0')),
  ('2', '1', 'REQ', ('0', '5')), ('2', '3', 'REQ', ('0', '5')),
  ('2', '5', 'REQ', ('0', '5')), ('3', '1', 'REQ', ('0', '5')),
  ('3', '2', 'REQ', ('0', '5')), ('3', '4', 'REQ', ('0', '5')),
  ('4', '1', 'REQ', ('0', '5')), ('4', '3', 'REQ', ('0', '5')),
  ('4', '5', 'REQ', ('0', '5'))]

step 04:  [('0', '1', 'REP', ('4', '0')), ('2', '1', 'REQ', ('4', '0')),
  ('2', '3', 'REQ', ('4', '0')), ('2', '5', 'REQ', ('4', '0')),
  ('5', '2', 'REP', ('0', '5'))]

step 05:  [('1', '4', 'REP', ('4', '0')), ('2', '1', 'REQ', ('2', '0')),
  ('2', '3', 'REQ', ('2', '0')), ('2', '5', 'REQ', ('2', '0'))]

step 06:  [('1', '0', 'REQ', ('2', '0')), ('1', '2', 'REQ', ('2', '0')),
  ('1', '3', 'REQ', ('2', '0')), ('1', '4', 'REQ', ('2', '0')),
  ('2', '1', 'REP', ('0', '5')), ('3', '1', 'REQ', ('2', '0')),
  ('3', '2', 'REQ', ('2', '0')), ('3', '4', 'REQ', ('2', '0')),
  ('4', '1', 'DATA', ('4', '0')), ('5', '2', 'REQ', ('2', '0')),
  ('5', '4', 'REQ', ('2', '0'))]

step 07:  [('0', '1', 'REP', ('2', '0')), ('1', '0', 'REP', ('0', '5')),
  ('4', '1', 'REQ', ('2', '0')), ('4', '3', 'REQ', ('2', '0')),
  ('4', '5', 'REQ', ('2', '0')), ('5', '2', 'REQ', ('5', '2')),
  ('5', '4', 'REQ', ('5', '2'))]

step 08:  [('0', '1', 'DATA', ('0', '5')), ('1', '0', 'DATA', ('4', '0')),
  ('2', '5', 'REP', ('5', '2')), ('4', '1', 'REQ', ('5', '2')),
  ('4', '3', 'REQ', ('5', '2')), ('4', '5', 'REQ', ('5', '2'))]

step 09:  [('1', '2', 'REP', ('2', '0')), ('3', '1', 'REQ', ('5', '2')),
  ('3', '2', 'REQ', ('5', '2')), ('3', '4', 'REQ', ('5', '2')),
  ('5', '2', 'DATA', ('5', '2'))]

step 10:  [('1', '2', 'DATA', ('0', '5')), ('2', '1', 'DATA', ('2', '0'))]

step 11:  [('1', '0', 'REQ', ('5', '2')), ('1', '2', 'REQ', ('5', '2')),
 ('1', '3', 'REQ', ('5', '2')), ('1', '4', 'REQ', ('5', '2'))]

step 12:  [('0', '1', 'REQ', ('5', '2')), ('1', '0', 'DATA', ('2', '0')),
   ('2', '5', 'DATA', ('0', '5'))]

step 13:  []
step 14:  []
```

2.1.3 Visualizer Outputs


(a) Step 0 (b) Step 1 (c) Step 2

(d) Step 3 (e) Step 4 (f) Step 5

(g) Step 6 (h) Step 7 (i) Step 8

(j) Step 9 (k) Step 10 (l) Step 11

(m) Step 12

Figure 11: The visualizer outputs for each time step

The results of the final evaluation is as follows.

Number of data packets : 4
Total hops : 8
Average no. of hops per data packet : 2.0

2.2 Improving the Efficiency of Protocol by further Exploiting the Route Cache

2.2.1 Derestricting route format

In the existing system, the entries in the route cache contain a specific format as [(target : route), (target : route) ..]. This generates the following inefficiency.

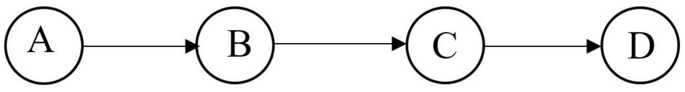


Figure 12: A Route which is already in the Route Cache

Consider an instance where a packet has to be sent from node A to node C and also assume that the route to node D also exists in the route cache of A as (D : B, C, D) The node A is unable to extract the route to node C from this entry as node C is not listed as the target node. Instead another route reply packet has to be initiated to find the path to node C. If this format is derestricted and used in a manner such that the path to a node inside any entry can be extracted, the number of route request transmissions can be minimized, hence propagation time as well as CPU overhead required to process those packets is reduced.

2.2.2 Intermediate nodes resolving a route request using their route caches

In the existing system, once a node receives a route request, it checks to see if it is the target node for that packet and if not, it broadcasts the route request packet again. As an alternative, once a node receives a route request, the following algorithm can be implemented.

```
IF packet ID already processed
   THEN skip packet
ELSE
   add packet ID to the recents list
   IF current node is target node
      THEN append current node ID to path and initiate RREP
   ELSE IF the path to destination exists in the routing cache of current node
      THEN append that route to path and initiate RREP
   ELSE
      append current node ID to path and broadcast
```

This will enable the system to minimize route request transmissions. However, steps have to be taken to avoid processing multiple route reply packets from multiple nodes and to select the shortest path among them.

2.3 Handling Disconnections During Transmission

To make the process of handling disconnections more robust, we propose that an acknowledgement message be sent during each transmission. For example, in a situation where a data packet is to be sent from node A to node D using the path A-B-C-D,

1. Node A send the packet to node B but still keeps the packet in a buffer

2. Node B receives the packet and sends an acknowledgement packet back to node A

3. Node A receives the acknowledgement packet and clears the packet from its buffer

4. The same process is repeated when the packet is sent from node B to node C and so on.

A description on disconnections handling using this system follows.

Scenario : A data packet is to be sent from node A to node D along the pre-discovered path A-B-C-D
Error : Node C disconnects

As per the above sequence of steps, node B sends the data packet to node C but does not receive an acknowledgment packet. Hence, the data packet is retained in the buffer and more attempts are made to resend the data packet to node C in an exponentially decreasing rate. In order to completely terminate the attempt to send the packet to node C, a maximum number of attempts is also specified.If node C does not respond after the maximum number of attempts, node C will initiate the error handling procedure.

Node B will initially look into its own route cache and if there are any entries through C present in it, they will be removed. Secondly, a node error packet indicating the unavailability of node C will be transmitted to the source i.e. node A. Upon receiving this packet, node A will check its route cache for any entries containing node C. If such entries are found, they will be truncated at node C.

2.4 Differences between DSR protocol and Distance Vector Routing protocol

Distance vector protocols rely on the information provided by the neighboring nodes. Each node periodically broadcasts the distance to every node within its transmission range. Based on this data, the source of a data packet computes the shortest path to the target by virtue of these distance values. The direction in this 'vector' routing protocol is embedded in the form of the nodes it passes[1][2].

Example : 'The target 192.168.1.0/24 is 4 hops away in direction of next-hop B'

In dynamic source routing protocol, the sender itself decides the complete path that the packet has to take through a route discovery process and embeds the discovered path into the packet and hence the consequent nodes will merely have to check for the next hop and forward it accordingly.

Advantages of Dynamic source routing

- In a time interval where no data packets are transmitted, a distance vector protocol will continuously broadcast routing advertisement messages whereas in dynamic source routing, route discovery procedure takes place only when data packets are in buffer.

- These route advertisements employed in distance vector protocol also causes the nodes to process a large number of redundant messages which requires CPU overhead.

- In dynamic source routing, each node possesses a route cache which contains pre discovered routes to multiple nodes. Therefore in a scenario where there is little to no movement of the nodes, these routes can be easily utilized without a route discovery process.

- Even in a situation where node movement is significant, the route discovery process employed in dynamic source routing can yield in an effective route much faster than in the case of a distance vector routing protocol.

Bibliography

[1] Cisco CCNA – Distance Vector Routing Protocols – CertificationKits.com. https://www.certificationkits.com/cisco-certification/ccna-articles/cisco-ccna-intro-to-routing-basics/cisco-ccna-distance-vector-routing-protocols/.

[2] IMPLEMENTATION OF DYNAMIC SOURCE ROUTING. http://www.cse.iitd.ac.in/~mcs142144/documents/DSR_thesis.pdf.

[3] Modeling the Propagation of RF Signals - MATLAB & Simulink - MathWorks India. https://in.mathworks.com/help/phased/examples/modeling-the-propagation-of-rf-signals.html.

[4] P.676 : Attenuation by atmospheric gases. https://www.itu.int/rec/R-REC-P.676-10-201309-S/en.

[5] P.838 : Specific attenuation model for rain for use in prediction methods. https://www.itu.int/rec/R-REC-P.838-3-200503-I/en.

[6] P.840 : Attenuation due to clouds and fog. https://www.itu.int/rec/R-REC-P.840-6-201309-S/en.

[7] Water Vapor and Vapor Pressure. http://hyperphysics.phy-astr.gsu.edu/hbase/Kinetic/watvap.html.