CS425A: Computer Networks
Homework 1

Kuldeep Singh Chouhan
200530

January 29, 2023

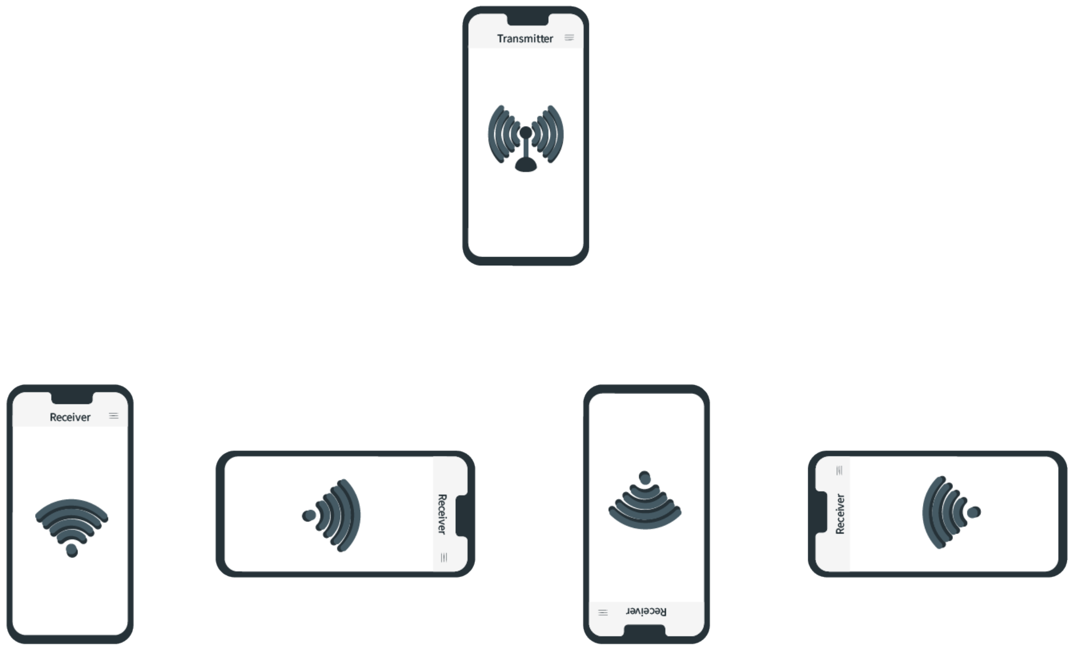
1 Finding the Path Loss Exponent

Path Loss exponent is a parameter which can determine the rate of change in the strength of the received signal with respect to distance. The required relation is given by the following equation,

$$[P_r(d)]dBm = [P_r(d_0)]dBm - 10n\log_{10}(\frac{d}{d_0}) + \chi \tag{1}$$

Procedure

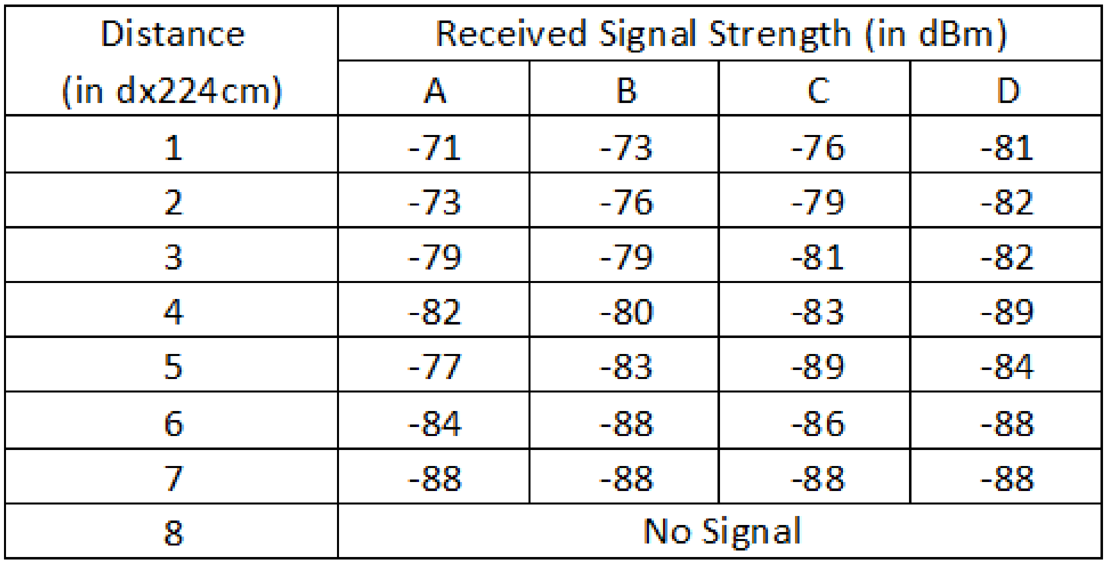
1. For this experiment, we will use two mobiles, one for transmitting WiFi signals and one for receiving them using WiFi Analyzer app.
2. For distance variation, we will use a set of four-floor tiles as a unit. Each tile is of 56cm x 56cm dimension. So, we will take readings at every 4x56cm = 224cm.
3. The WiFi signal used was of 5GHz bandwidth. The signal was received at a maximum distance of 15.68m.
4. Reciever mobile was kept at 4 different orientations as shown below.



(A) (B) (C) (D)

Data Collected

The following data was collected, at distances 1x224cm, 2x224cm, 3x224cm, etc from the transmitter for different orientations A, B, C and D respectively (in -dBm).

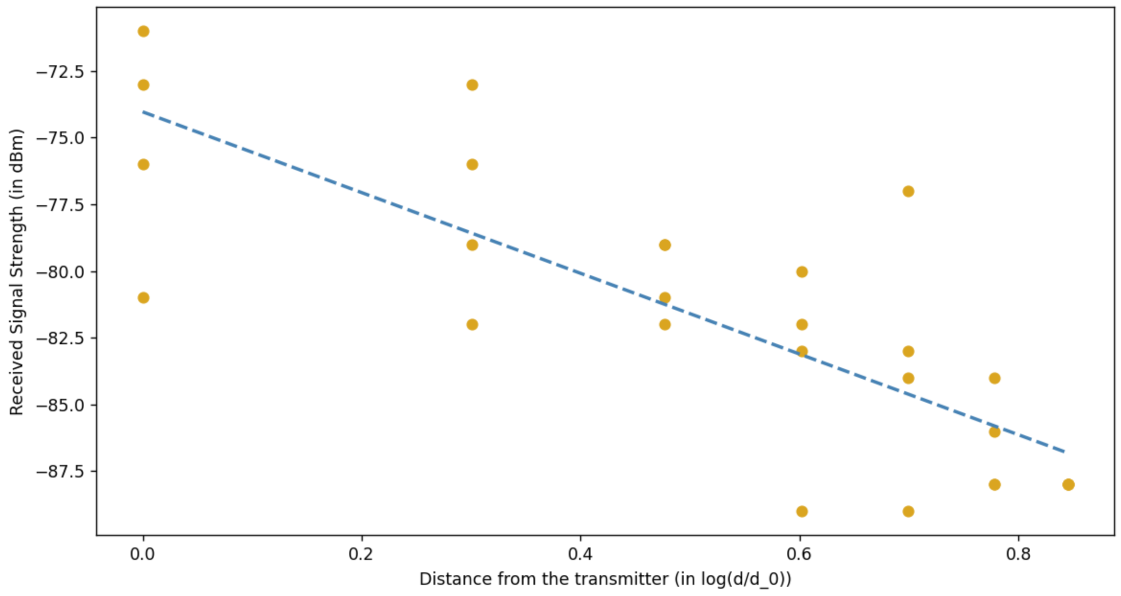


Distance (in dx224cm)	Received Signal Strength (in dBm)			
	A	B	C	D
1	-71	-73	-76	-81
2	-73	-76	-79	-82
3	-79	-79	-81	-82
4	-82	-80	-83	-89
5	-77	-83	-89	-84
6	-84	-88	-86	-88
7	-88	-88	-88	-88
8	No Signal			

Best Fit line

On plotting the above-mentioned data where the RSSI (received signal strength) values are in y-axis (dBm), and the distances are in x-axis (in log scale), we get the following line equation as the best fit

$$y = -74.04 - 15.13x \tag{2}$$



On comparing eq2 with eq1, we get the Path Loss Exponent **n = 1.513**.
The variance of the sample data with respect to the best fit line is **9.90**.

2 Range Estimation

On placing the value of n, and $d_0 = 1$, we get the following equation,

$$[P_r(d)]dBm = [P_r(d_0 = 1)]dBm - 15.13 \log_{10} d \tag{3}$$

$$[P_r(d_0 = 1)]dBm = -68dBm$$

$$[P_r(d)]dBm = -68dBm - 15.13 \log_{10} d \tag{4}$$

The below table shows the predicted and actual distance (in m) and the error in this calculation.

Received Signal Strength (in dBm)	Predicted distance (in m)	Actual Distance (in m)	Error (in m)
-71	1.57886948	2.24	0.66113052
-76	3.380051203	4.48	1.0999488
-81	7.236029501	6.72	0.5160295
-82	8.4258891	8.96	0.5341109
-84	11.42474613	11.2	0.22474613
		Average Error	0.60719317

The average error in range/distance detection comes out to be **0.61m**.

3 Code

```
import numpy as np
import matplotlib.pyplot as plt

# for question 1
#define data of distances and RSSI
x = np.array([0.0, 0.0, 0.0, 0.0,
    0.3010299956639812, 0.3010299956639812, 0.3010299956639812, 0.3010299956639812,
    0.47712125471966244, 0.47712125471966244, 0.47712125471966244, 0.47712125471966244,
    0.6020599913279624, 0.6020599913279624, 0.6020599913279624, 0.6020599913279624,
    0.6989700043360189, 0.6989700043360189, 0.6989700043360189, 0.6989700043360189,
    0.7781512503836436, 0.7781512503836436, 0.7781512503836436, 0.7781512503836436,
    0.8450980400142568, 0.8450980400142568, 0.8450980400142568, 0.8450980400142568])
y = np.array([-71,-73,-76,-81,
              -73,-76,-79,-82,
              -79,-79,-81,-82,
              -82,-80,-83,-89,
              -77,-83,-89,-84,
              -84,-88,-86,-88,
              -88,-88,-88,-88])

#find line of best fit
a, b = np.polyfit(x, y, 1)

#add points to plot
plt.scatter(x, y, color='goldenrod')

#add line of best fit to plot
plt.plot(x, a*x+b, color='steelblue', linestyle='--', linewidth=2)
plt.xlabel("Distance from the transmitter (in log(d/d_0))")
plt.ylabel("Received Signal Strength (in dBm)")
```

```python
# for question 2
# sample for RSSI values
collectedStrength = np.array([-71,-76,-81,-82,-84])

predictedDistance = []
predictionError = []
i=1

strength_d0 = 68  # signal strength at d=1m
refernceDistance = 2.24

for elem in collectedStrength:
    predictedDistance.append(10**((elem+strength_d0)/a))
    predictionError.append(abs(10**((elem+strength_d0)/a) - i*refernceDistance))
    i=i+1


print("Best Fit Line equation: y = " + "{:.2f}".format(b) + " + {:.2f}".format(a) + "x")

z=[]
for i in x:
    z.append(-74.04 -15.13*i)

sum=0

for i in range(0,len(x)):
    sum = sum + (y[i]-z[i])*(y[i]-z[i])
print("Variance: ", sum/(len(x)-1))

print("Predicted Distance (in m): ",predictedDistance)
print("Error in Predicted Distance (in m): ",predictionError)
print("Average Error (in m): ", np.average(predictionError))

#add fitted regression equation to plot
plt.show()
```

Output

Best Fit Line equation: $y = -74.04 + -15.13x$

Variance: 9.900562925598662

Predicted Distance (in m):
[1.578869479795241, 3.380051202929519, 7.236029501252332, 8.425889099832768, 11.424746134425286]

Error in Predicted Distance (in m):
[0.6611305202047593, 1.0999487970704815, 0.5160295012523317, 0.5341109001672333, 0.2247461344252848]

Average Error (in m): 0.6071931706240181