18-748 Wireless Sensor Networks Lab 1

Emily Rupple, Iljoo Baek, Mengwen He

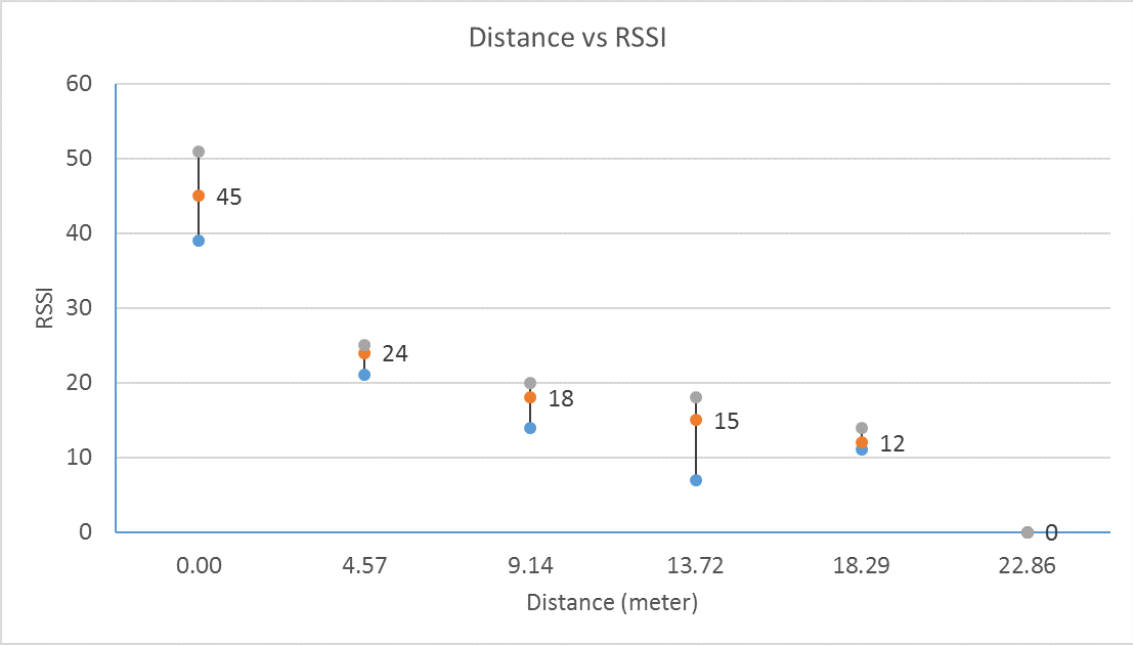
February 2, 2017

1 Assignment 2

1.1 Data Measurement

Table 1: The Measurements of Distance and RSSI

Distance		RSSI		
Yard	Meter	Min	Max	Ave.
0	0	39	51	45
5	4.57	21	25	24
10	9.14	14	20	18
15	13.72	7	18	15
20	18.29	11	14	12
25	22.86	0	0	0



1.2 Compute Path Loss Exponent n

Path loss exponent is given by:

$$L = 10n \log_{10} d + C$$

- L: path loss
- n: path loss exponent
- d: distance in meters
- C: system constant

From two different data points:

$$(L2 - L1) = 10n \log_{10}(d2/d1) \Rightarrow n = (L2 - L1)/(10 \log_{10}(d2/d1))$$

As RSSI is linearly related to path loss

$$(L2 - L1) = (RSSI2 - RSSI1)$$

Fix $d1 = 4.57$, $L1 = 24$ and get the estimation as Tab.2:

Table 2: The Estimation of Path Loss Exponent

$d2$	$L2$	n
9.14	18	-1.993
13.72	15	-1.885
18.29	12	-1.992
22.86	0	-3.433 (invalid)

Because we cannot get RSSI value when $d2 = 22.86$, we cannot trust this measurement that just represents the margin of the signal coverage. Therefore, the estimated Path Loss Exponent is:

$$\mathbb{E}[n] = \frac{-1.993 - 1.885 - 1.992}{3} = -1.957$$

$$Var[n] = \frac{(1.957 - 1.993)^2 + (1.957 - 1.885)^2 + (1.957 - 1.992)^2}{3} = 0.0039$$

$$\sigma = \sqrt{Var[n]} = 0.0621$$

2 Assignment 3

2.1 Data Collection

Task1:	0	351562680			
Task1:	1	298828278			
Task1:	2	601562808			
Task1:	3	296875152			
Task1:	4	599609682			
Task1:	5	294922026			
Task1:	6	597656556			
Task1:	7	292968900			
Task1:	8	595703430			
Task1:	9	291015774			
Task1:	10	593750304			
Task1:	11	289062648			
Task1:	12	591797178			
Task1:	13	287109522			
Task1:	14	589844052			
Task1:	15	285156396			
Task1:	16	587890926			
Task1:	17	283203270			
Task1:	18	585937800			
Task1:	19	281250144	Task2:	0	47851587
Task1:	20	583984674	Task2:	2	299804841
Task1:	21	279297018	Task2:	4	297851715
Task1:	22	582031548	Task2:	6	295898589
Task1:	23	277343892	Task2:	8	293945463
Task1:	24	580078422	Task2:	10	291992337
Task1:	25	275390766	Task2:	12	290039211
Task1:	26	578125296	Task2:	14	288086085
Task1:	27	273437640	Task2:	16	286132959
Task1:	28	576172170	Task2:	18	284179833
Task1:	29	271484514	Task2:	20	282226707
Task1:	30	574219044	Task2:	22	280273581
Task1:	31	269531388	Task2:	24	278320455
Task1:	32	572265918	Task2:	26	276367329
Task1:	33	267578262	Task2:	28	274414203
Task1:	34	570312792	Task2:	30	272461077
Task1:	35	265625136	Task2:	32	270507951
Task1:	36	568359666	Task2:	34	268554825
Task1:	37	263672010	Task2:	36	266601699
Task1:	38	566406540	Task2:	38	264648573

(a) Case1: Task1 has lower priority.

Task1:	0	183593844			
Task1:	1	298828278			
Task1:	2	297851715			
Task1:	3	296875152			
Task1:	4	295898589			
Task1:	5	294922026			
Task1:	6	293945463			
Task1:	7	292968900			
Task1:	8	291992337			
Task1:	9	291015774			
Task1:	10	290039211			
Task1:	11	289062648			
Task1:	12	288086085			
Task1:	13	287109522			
Task1:	14	286132959			
Task1:	15	285156396			
Task1:	16	284179833			
Task1:	17	283203270			
Task1:	18	282226707			
Task1:	19	281250144	Task2:	0	486328374
Task1:	20	280273581	Task2:	2	601562808
Task1:	21	279297018	Task2:	4	599609682
Task1:	22	278320455	Task2:	6	597656556
Task1:	23	277343892	Task2:	8	595703430
Task1:	24	276367329	Task2:	10	593750304
Task1:	25	275390766	Task2:	12	591797178
Task1:	26	274414203	Task2:	14	589844052
Task1:	27	273437640	Task2:	16	587890926
Task1:	28	272461077	Task2:	18	585937800
Task1:	29	271484514	Task2:	20	583984674
Task1:	30	270507951	Task2:	22	582031548
Task1:	31	269531388	Task2:	24	580078422
Task1:	32	268554825	Task2:	26	578125296
Task1:	33	267578262	Task2:	28	576172170
Task1:	34	266601699	Task2:	30	574219044
Task1:	35	265625136	Task2:	32	572265918
Task1:	36	264648573	Task2:	34	570312792
Task1:	37	263672010	Task2:	36	568359666
Task1:	38	262695447	Task2:	38	566406540

(b) Case2: Task2 has lower priority.

2.2 Worst-case Response Time Estimation

Because the task1 and task2 are released every 1s and 2s, we can directly get their response time from the nanosecond part, and thus easily get the worst-case response time as below:

- Case 1:
  - Task1: 601562808
  - Task2: 299804841
- Case 2:
  - Task1: 298828278
  - Task2: 601562808

2.3 Explanation of The Differences

2.3.1 Response Time

For case1, because task1 has lower priority, it will be preempted by task2 every 2 seconds; therefore, the response time of task1 is not stable. For case2, because task1 has higher priority, it will not be preempted by task2; therefore, the response time of task1 is stable.

For task2, because its period interval is longer than task1's, and they form a harmonized taskset; therefore, task2's response time is stable for both cases.

2.3.2 Worst-case Response Time

Comparing the worst-case response time of task1 and task2 from two cases, we found that the task with higher priority will have smaller worst-case response time.

2.4 Bonus1: Find a tight CPU reserve budget for each task.

From the experiment, we know that if task1 with shorter period interval has higher priority, both tasks can have stable response time; therefore, our method is first to use the RMS(Rate-monotonic scheduling) to assign the priority to each task, and then assign each task's reserve budget equals to its worst-case response time.

2.5 Bonus2: Plot graphs for both basks to show their response times.