Fall 2018

Simulation of Any-to-One Communication Protocol for WSN in Cooja Simulator of Contiki OS

Nirupana Naregudam
Iowa State University

Simulation of Any-to-one Communication Protocol for WSN in Cooja Simulator of Contiki OS

By

Nirupana Naregudam

A Creative Component submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:

Dr. Ahmed Kamal

Iowa State University

Ames, Iowa

2018

Acknowledgements

I would sincerely like to thank **Dr. Ahmed Kamal** (Director of Graduate Education and Professor for the Department of Computer Engineering) for guiding me through the course of this work. I would also like to thank Ala'Eddin Masadeh (PhD student in the department of Computer Engineering) who helped me understand the concepts and was available to clear my doubts. Finally, I would like to thank my family and friends who showed their support all the way.

Abstract

Increase in the usage Internet of Things has driven lot of importance to wireless sensor networks. Wireless sensor network consists of sensor nodes with low power and low transmission range. Sensor power is the crucial part because if the power goes down, the sensors die out and will not be available for communication. This project deals with sensor nodes which are deployed in an area and there is an external source available for harvesting power. The harvested energy keeps the sensor nodes powered and the communication in the network can sustain for longer time.

This work talks about the simulation methods of a multi-agent reinforcement algorithm that aims at minimizing the energy consumption in a wireless sensor network. The simulation is performed in Cooja simulator of Contiki Operating System. Contiki OS is an open source Real Time Operating System (RTOS) for low power wireless devices. Internet of Things has many features including power awareness which is providing mechanisms for the estimation of power consumption in the network. Developing and debugging large wireless networks is really difficult and this is made much easier by providing environment to develop applications on fully emulated devices using Cooja, the Contiki network simulator.

The simulation is carried out for two scenarios on a network where are two sensor nodes and one sink node. The results are studied for both the cases and seen that overall energy consumption on the network is reduced for the case where the q-learning algorithm is used.

Contents

Introduction

Internet of Things has gained a widespread attention in the research fields of communication in the past few years. One of the key areas of focus in Internet of Things (IoT) is the Wireless Sensor Network which is used many fields like military operations, factory automations, real-time monitoring. A Wireless Sensor Network (WSN) is a network which is formed by large number of sensor nodes in an area which are deployed to monitor the physical characteristics and to collect information regarding the environment. The information collected by these sensor nodes can be light, heat, temperature, pressure, etc., which will have to be communicated to a coordinator node which then sends to another system for subsequent usage. These sensor nodes are low power devices with very low battery time and throughput.

The energy wastage in WSNs is mainly due to reasons like collisions producing many retransmissions and increasing the latency, idle listening where radio of the sensor nodes will be kept on for longer periods of time to monitor the channel for the packets, overhearing where the nodes will pick up the packets which are not intended for them.

These sensor nodes have their battery life very low and when present along with any energy harvester, these can show better lifetimes. For the sensor nodes which must send the same data to the collector, the network throughput can be increased if the sensor nodes can be managed to send data from only one of them to the collector node [1]. In [1] a new MAC protocol called Flip MAC is proposed which uses any-to-one communication protocol.

In [2] the reinforcement learning is used to determine the best possible node to transmit to the collector node. Q-learning is used to train the model and the next best state for a current state of the network is selected.

Any-to-one Communication Protocol

When there are many sensor nodes deployed in an area, to save the energy consumption of the sensor nodes, it will be beneficial to send data from only one sensor to the sink node [2]. To know which the best node is to send data, a machine learning model can be trained on the sensor node which has information about all the other sensors in the network. Then best node can be determined according to the current state. This sensor node can then send the data and all the other sensor nodes can sleep till the next cycle. At the next cycle, again the machine learning model will be trained on the deciding sensor node. The process repeats.

Contiki OS

Contiki OS is an open source Real-time Operating System that has a network simulator called Cooja [5]. This network simulator makes it easy to develop and run user defined applications very smoothly and provide deep insights into the results using various forms like mote output window, network traffic window, power trackers, etc.

Implementation

The algorithm is simulated using Cooja simulator of Contiki Operating system. The current simulation is performed in a network where there are two sensor nodes and one sink node. To understand the effect of using the Q-learning algorithm on the network, two cases are simulated for the considered network and then the results are analyzed.

Case 1 – No ML case

The first case is simulated with two sensor nodes sending messages to the sink node at an interval of 30 clock seconds. Sky motes are used in the simulation. It does not involve any Q-learning training. The simulation is performed for 1 hour.

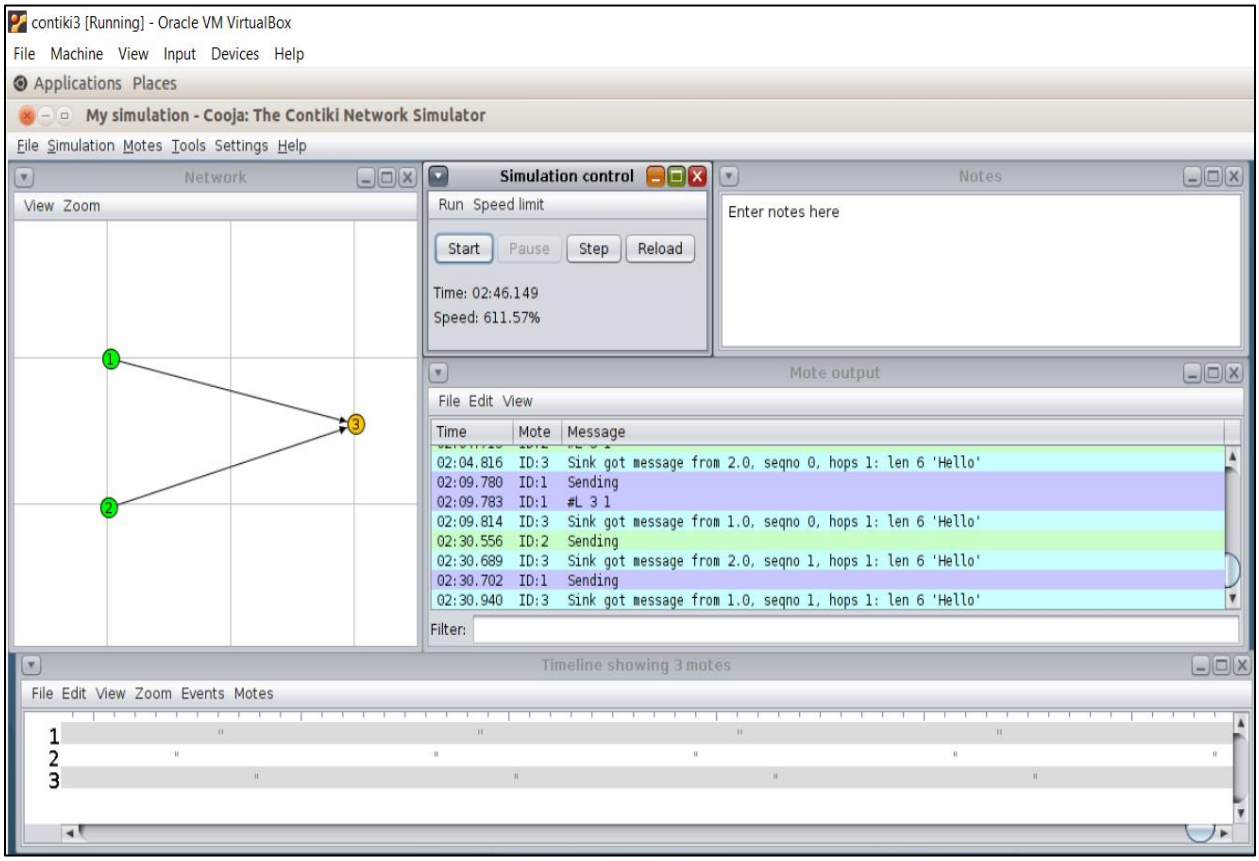

Figure 1: Cooja Simulator GUI for Non-ML case

Case 2 – ML case

The second case consists of two sensor nodes and a sink node. Sky motes are used in the simulation. Let us consider the two sensor nodes to be node 1 and node 2 and the sink node to be node 3. The simulation is performed for 1 hour. The steps in this firmware are as described below.

Step 1: node 1 will broadcast its state to node 2.

Step 2: Node 2 is the deciding node here. After it receives the states of the network sensor nodes, it will feed all the details to the q-learning function, i.e., it will send the transmission level of node 1 and transmission level of node 2.

Step 3: Model will be trained and give output which is the sensor node id and the transmission power with which it has to send data to the sink node.

Step 4: Node 2 after receiving the output from the q-learning function, it will broadcast the result to other sensor nodes in the network, which is node 1 in this case.

Step 5: If node 2 is the sensor which must send data to the sink node, then it will send the unicast message to the sink node.

Step 6: After receiving the broadcast message from the node 2, node 1 will check if it is the senor node that has to send unicast message to the sink node. If yes, it will send the unicast message to the sink node. If no, it will sleep till the next cycle when the q-learning function will be called.

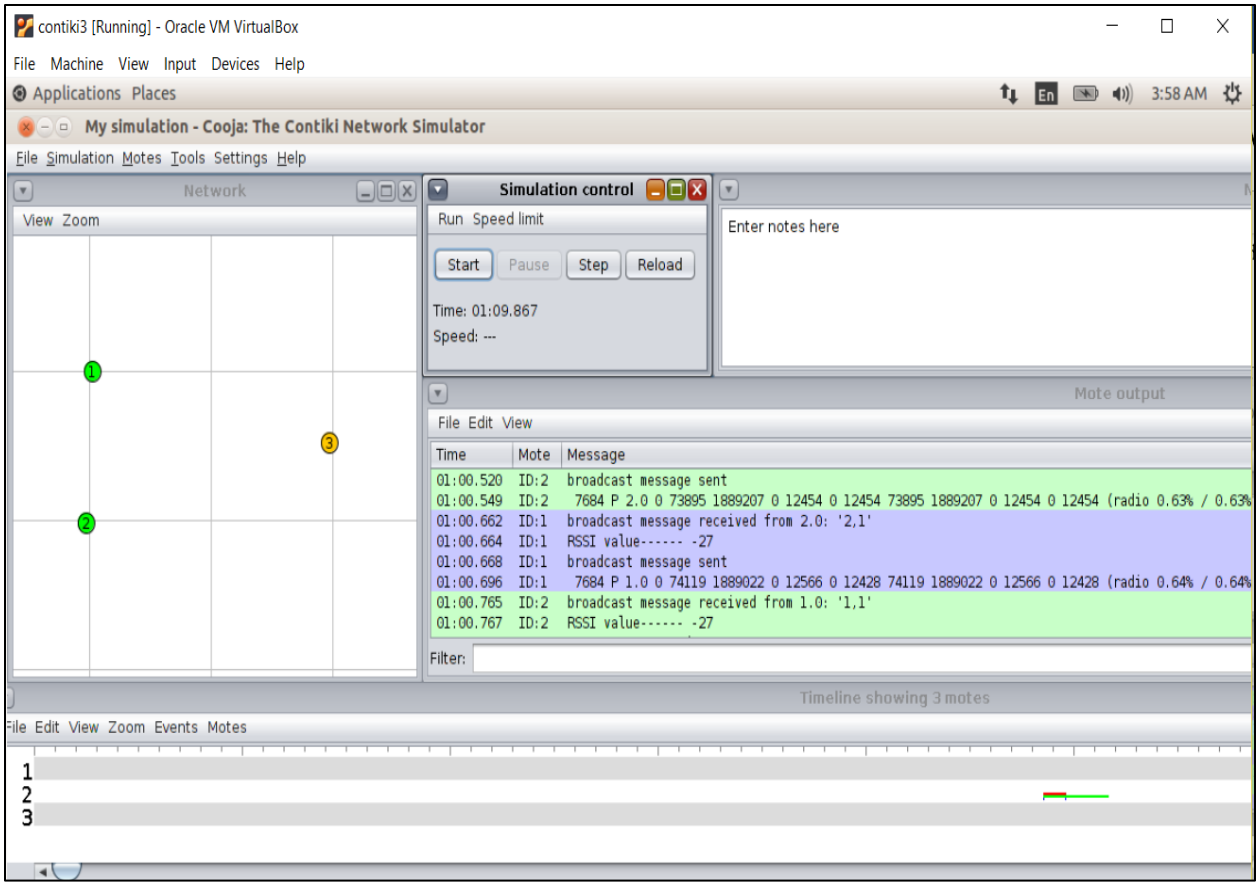

Figure 2: Cooja Simulator GUI for ML Case

Adding required libraries for ML function in Cooja

For the reinforcement learning functions some math libraries need to be included in Contiki system [10]. This is done by changing the Makefile.include file. So, at the end of LDFLGS, -lm should be added as shown below.

```
$(Q)$(LD) $(LDFLAGS) $(TARGET_STARTFILES) ${filter-out %.a,$^} \
    ${filter %.a,$^} $(TARGET_LIBFILES) -o $@ -lm
```

Adding the powertrace app to application

Powertrace app of the Contiki system will show the power values of the sensor nodes at different interval of time. To use it in the firmware, firstly add the powertrace app in the makefile of the "rime" folder by adding the below shown line.

APPS+=powertrace

Then the powertrace app can be referenced in the firmware file by including the powertrace.h header file. Open the firmware file and add the below include line.

#include "powertrace.h"

Powertrace can be called providing number of clockseconds it has to repeat. In this project, the power values of the sensor nodes will be collected for every 60 seconds of the Contiki clock.

powertrace_start (CLOCK_SECOND * 60);

Cooja Simulation Procedure

After logging into the Contiki OS, run the terminal to open the Cooja simulator by entering the following

> cd contiki/tools/cooja

> ant run

Cooja simulator will be opened. Now start a new simulation by selecting File->New Simulation. Following window will appear. Provide a name to the simulation. Then click on create button.

Simulation window will appear.

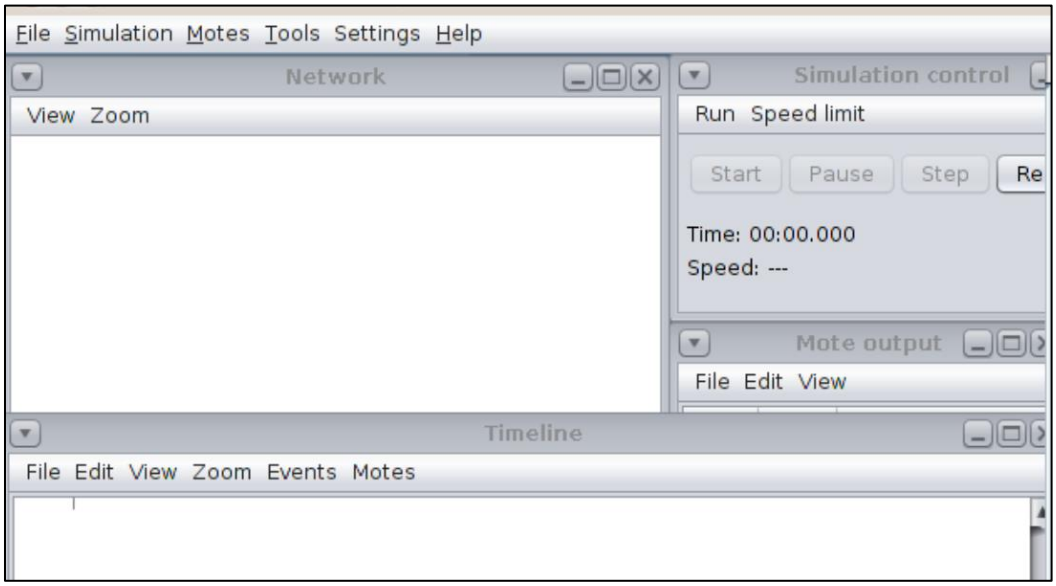

Figure 3: Cooja Simulation window

Select **Motes->Add motes->Create a new mote->sky mote…**

In the window that appears, provide a description if needed and then select the respected firmware file which has to be run in each of the cases that are being studied in this project. Then click on compile tab.

After the file is successfully compiled, create button will be enabled.

Click on create button to add 3 sky motes to the simulation window. Motes will be placed randomly by default. Their positions can be changed anytime even during the simulation is running.

After the nodes appear in the network window, select the “view” button on the top left of the network window. This option will enable to view details of the motes during the simulation. In the dropdown shown, “node id”, “radio traffic”, “10m background”, “node radio” can be selected.

After the “10m background grid” is seen in the network window, adjust the locations of the nodes in triangular model according to the below image.

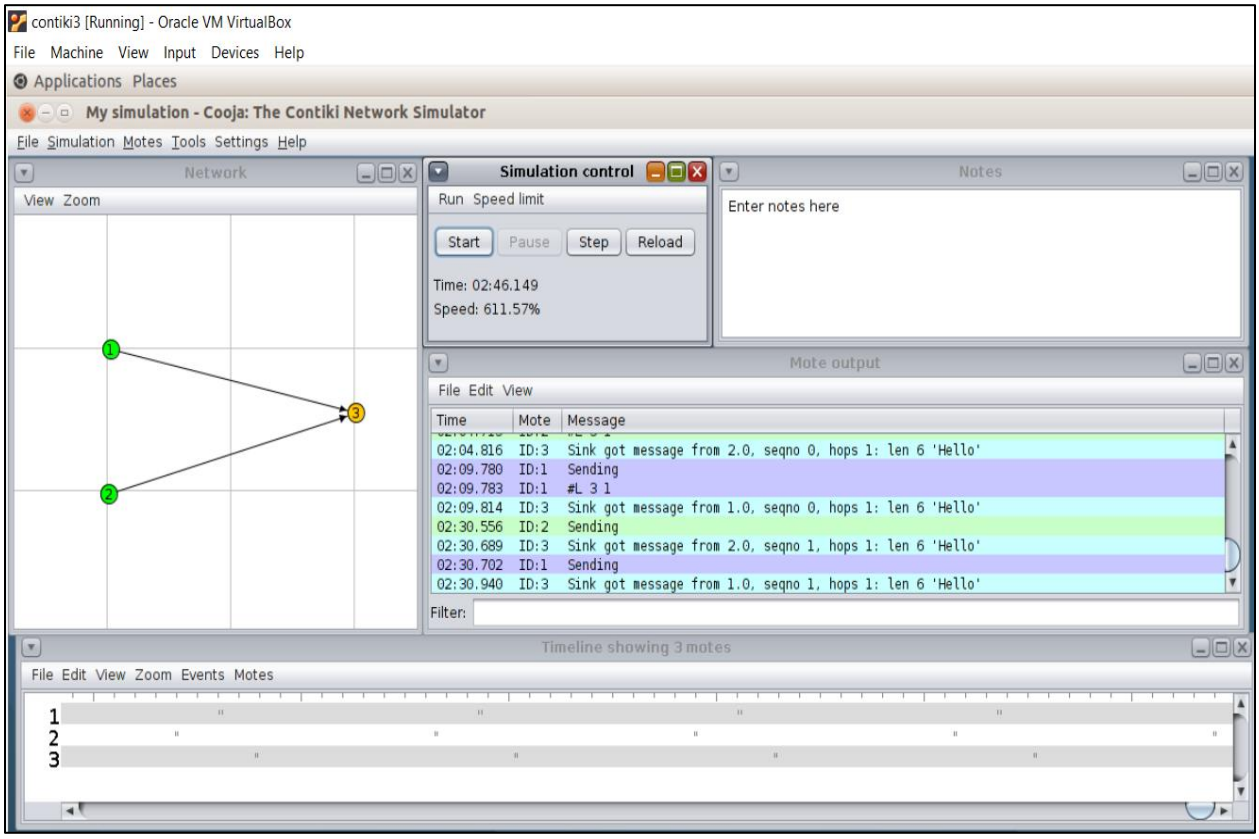

Figure 4: Cooja Simulator GUI and different windows

The Simulation control window provides buttons to start, pause, stop and reload the simulation. It also shows the time and speed of the simulation running. Start the simulation and run it for 1 hour. During the simulation, the “Mote output” window will show the outputs for each sky mote sensor node according to the timeline. It displays the message details, the time at which the message was sent or received, and the node ID details. The power trace values of each sensor node will also be displayed in the form of number of clock ticks. The output can also be saved to a file through the menu options present in it which will be used for energy calculations later. The ‘Timeline showing 3 motes’ window will provide a visual understanding of what happens at each node as time moves.

Energy Calculations

The values produced by the powertrace app needs to be processed to understand the energy consumption of the sensor nodes. So, the output from the “Mote output” window is saved to a text file by clicking on the File -> Save to file option. A sample powertrace output is show below. This is a part of the complete log file that is saved.

```
02:00.694   ID:1     15364 P 1.0 1 144762 3780546 2211 25098 0 24882 70640 1891524 2211 12532 0
12454 (radio 0.69% / 0.75% tx 0.05% / 0.11% listen 0.63% / 0.63%)
```

Figure 5: Example of Powertrace results for a node

The above image shows the powertrace output for node 1. The values of interest for the energy consumption calculations are 144762, 3780546, 2211, 25098. These numbers refer to ALL_CPU, ALL_LPM, ALL_TX and ALL_RX values. ALL_CPU is the total number of clock ticks when the CPU is high or in active mode. ALL_LPM is the total of clock ticks in the Low Power Mode state. ALL_TX is the total number of clock ticks in the Transmit state. ALL_RX is the total number of clock ticks in the Receive state.

Below is the sample data for a sensor node 2.

	Raw Data			
	ALL CPU	ALL LPM	ALL TX	ALL RX
0	117399	1845633	11095	14374
1	773878	12962717	46147	121389
2	1429725	24070951	85368	233493
3	2073853	35188582	124144	347658
4	2736429	46288033	164281	461354
5	3353600	57436808	198173	572342
6	3999490	68553826	233882	684488
7	4649956	79665563	270720	796309
8	5275348	90804023	307697	909236
9	5897996	1.02E+08	341821	1019995
10	6522574	1.13E+08	374719	1130772

Figure 6: Sample Powertrace data for 10 intervals

The Power consumption is then calculated by using the bellow formula [6]:

Power Consumption = (Energest_Value *Current * Voltage) / (RTIMER_SECOND * Runtime)

- Energest_Value will be obtained by taking the difference between number of ticks between a time interval and its previous time interval.
- According to the Skymote Datasheet [7], the values used are
    - Voltage = 3V
    - Current = 330 uA, 1.1 uA, 18.8 mA, and 17.4 mA for CPU, LPM, TX, and RX
    - RTIMER_SECOND = 32768
    - Runtime = 360

The Power consumption in mW will be as shown in table

	Energy Consumption (Power - mW)				
	CPU	LPM	TX	RX	Total
1	0.005509391	0.000310994	0.01551068	0.05116481	0.07249587
2	0.005504087	0.000310747	0.01735548	0.0535979	0.07676822
3	0.005405737	0.00031101	0.01715857	0.05458328	0.0774586
4	0.005560559	0.000310501	0.01776082	0.05435905	0.07799093
5	0.005179505	0.000311881	0.01499738	0.05306433	0.07355309
6	0.005420525	0.000310992	0.01580141	0.05361798	0.0751509
7	0.005458928	0.000310845	0.01630099	0.0534626	0.07553336
8	0.005248499	0.000311592	0.0163625	0.05399138	0.07591398
9	0.00522547	0.000311619	0.01510004	0.05295484	0.07359197
10	0.005241667	0.000311478	0.01455753	0.05296345	0.07307412

Figure 7: Sample Power Calculations for 10 intervals

The average energy consumption over the entire interval is 0.0751531 mW.

Results and analysis

The energy consumption values are calculated for all the three nodes in the network using the method described in the above section for both ML case and non-ML case.

NO ML case: 2 sensor nodes sending data to sink node

1) Node 3 (Sink Node in this case)

	Raw Data				Energy Consumption (Power - mW)					
S.No	ALL CPU	ALL LPM	ALL TX	ALL RX	S.No	CPU	LPM	TX	RX	Total
0	107038	1855924	10780	15507						
1	777147	12962842	97518	153444	1	0.005623779	0.00031071	0.03838199	0.065948883	0.11026536
2	1498982	24015123	198660	305740	2	0.00605788	0.000309181	0.04475583	0.072814046	0.12393694
3	2200170	35088043	298015	457863	3	0.005884604	0.000309759	0.04396507	0.072731333	0.12289077
4	2920533	46141854	399525	609962	4	0.006045527	0.000309224	0.04491867	0.072719859	0.12399328
5	3606906	57229668	498742	761666	5	0.005760271	0.000310175	0.04390401	0.072531006	0.12250546
6	4315721	68294954	597738	913259	6	0.005948612	0.000309545	0.04380621	0.072477936	0.12254231
7	5036444	79348396	698882	1065040	7	0.006048548	0.000309214	0.04475671	0.07256782	0.1236823
8	5723699	90435304	797902	1216634	8	0.005767673	0.00031015	0.04381683	0.072478414	0.12237307
9	6406995	101526137	894772	1366549	9	0.005734448	0.00031026	0.04286545	0.071675669	0.12058583
10	7100571	112606717	996809	1520401	10	0.005820721	0.000309973	0.04515187	0.073557983	0.12484055

Figure 8: Power Calculations for No ML case - Node 3 (Sink Node)

Average of the total energy consumption over entire interval for Sink Node = 0.12176159 mW

2) Node 2

	Raw Data				Energy Consumption (Power - mW)					
S.No	ALL CPU	ALL LPM	ALL TX	ALL RX	S.No	CPU	LPM	TX	RX	Total
0	117399	1845633	11095	14374						
1	773878	12962717	46147	121389	1	0.005509391	0.000310994	0.01551068	0.051164805	0.07249587
2	1429725	24070951	85368	233493	2	0.005504087	0.000310747	0.01735548	0.0535979	0.07676822
3	2073853	35188582	124144	347658	3	0.005405737	0.00031101	0.01715857	0.054583282	0.0774586
4	2736429	46288033	164281	461354	4	0.005560559	0.000310501	0.01776082	0.054359049	0.07799093
5	3353600	57436808	198173	572342	5	0.005179505	0.000311881	0.01499738	0.053064331	0.07355309
6	3999490	68553826	233882	684488	6	0.005420525	0.000310992	0.01580141	0.053617981	0.0751509
7	4649956	79665563	270720	796309	7	0.005458928	0.000310845	0.01630099	0.053462596	0.07553336
8	5275348	90804023	307697	909236	8	0.005248499	0.000311592	0.0163625	0.053991384	0.07591398
9	5897996	101943451	341821	1019995	9	0.00522547	0.000311619	0.01510004	0.052954844	0.07359197
10	6522574	113077833	374719	1130772	10	0.005241667	0.000311478	0.01455753	0.05296345	0.07307412

Figure 9: Power Calculations for No ML case - Node 2

Average of the total energy consumption over entire interval for Sensor Node 2 = 0.0751531 mW

3) Node 1

	Raw Data				Energy Consumption (Power - mW)					
S.No	ALL CPU	ALL LPM	ALL TX	ALL RX	S.No	CPU	LPM	TX	RX	Total
0	114086	1848941	9029	15788						
1	756976	12979737	39205	110242	1	0.005395348	0.000311378	0.01335303	0.045159281	0.06421903
2	1410292	24088697	69433	211501	2	0.005482846	0.000310767	0.01337604	0.048412811	0.06758246
3	2040588	35219861	98308	375671	3	0.005289655	0.000311388	0.01277733	0.078491109	0.09686948
4	2692858	46332021	124813	539038	4	0.005474068	0.000310857	0.01172859	0.078107188	0.0956207
5	3307695	57481791	156273	640522	5	0.005159917	0.000311909	0.0139212	0.048520386	0.06791342
6	3939320	68612474	182919	738958	6	0.005300808	0.000311375	0.01179099	0.04706311	0.06446628
7	4601369	79713325	215122	839706	7	0.005556136	0.00031054	0.01424998	0.048168498	0.06828516
8	5217833	90855920	243693	939114	8	0.005173572	0.000311708	0.01264281	0.047527832	0.06565592
9	5819983	102018627	267009	1035539	9	0.005053444	0.000312271	0.01031744	0.046101634	0.06178479
10	6425381	113175223	290245	1132188	10	0.005080702	0.0003121	0.01028204	0.04620873	0.06188358

Figure 10: Power Calculations for No ML case - Node 1

Average of the total energy consumption over entire interval for Sensor Node 3 = 0.07142808 mW

ML case: 2 sensor nodes with only one sending data to sink node

1) Node 3 (Sink Node)

	Raw Data				Energy Consumption (Power - mW)					
S.No	ALL CPU	ALL LPM	ALL TX	ALL RX	S.No	CPU	LPM	TX	RX	Total
0	73976	1889050	0	12778						
1	433509	13303284	0	88082	1	0.003017321	0.000319307	0	0.036003499	0.03934013
2	785470	24725131	0	162962	2	0.002953774	0.00031952	0	0.035800781	0.03907408
3	1137870	36146542	0	237842	3	0.002957458	0.000319508	0	0.035800781	0.03907775
4	1490628	47567594	0	312722	4	0.002960463	0.000319498	0	0.035800781	0.03908074
5	1843670	58988358	0	387602	5	0.002962846	0.00031949	0	0.035800781	0.03908312
6	2196977	70408856	0	462482	6	0.00296507	0.000319482	0	0.035800781	0.03908533
7	2550566	81829070	0	537362	7	0.002967437	0.000319474	0	0.035800781	0.03908769
8	2904472	93248968	0	612242	8	0.002970097	0.000319465	0	0.035800781	0.03909034
9	3258689	104668555	0	687122	9	0.002972707	0.000319457	0	0.035800781	0.03909295
10	3613234	116087815	0	762002	10	0.00297546	0.000319447	0	0.035800781	0.03909569

Figure 11: Power Calculations for ML case - Node 3 (Sink Node)

Average of the total energy consumption over entire interval for Sink Node = 0.03911078 mW

2) Node 2 (ML node and sender to Sink Node)

	Raw Data				Energy Consumption (Power - mW)					
S.No	ALL CPU	ALL LPM	ALL TX	ALL RX	S.No	CPU	LPM	TX	RX	Total
0	73895	1889207	0	12454						
1	4709021	9036252	6925	89440	1	0.038899525	0.000199935	0.00306435	0.036807678	0.07897148
2	5061234	20457849	6925	164320	2	0.002955889	0.000319513	0	0.035800781	0.03907618
3	5413897	31878997	6925	239200	3	0.002959666	0.0003195	0	0.035800781	0.03907995
4	5766858	43299847	6925	314080	4	0.002962167	0.000319492	0	0.035800781	0.03908244
5	6120105	54720408	6925	388960	5	0.002964567	0.000319484	0	0.035800781	0.03908483
6	6473622	66140696	6925	463840	6	0.002966833	0.000319476	0	0.035800781	0.03908709
7	6827420	77560701	6925	538720	7	0.002969191	0.000319468	0	0.035800781	0.03908944
8	7181526	88980397	6925	613600	8	0.002971776	0.00031946	0	0.035800781	0.03909202
9	7535923	100399803	6925	688480	9	0.002974218	0.000319452	0	0.035800781	0.03909445
10	7890652	111818879	6925	763360	10	0.002977004	0.000319442	0	0.035800781	0.03909723

Figure 12: Power Calculations for ML case - Node 2 (ML node and data sender to sink node)

Average of the total energy consumption over entire interval for Node 2 = 0.04307551 mW

3) Node 1 (Sleep Node)

	Raw Data				Energy Consumption (Power - mW)					
S.No	ALL CPU	ALL LPM	ALL TX	ALL RX	S.No	CPU	LPM	TX	RX	Total
0	74119	1889022	0	12566						
1	441727	13295090	4049	54246	1	0.003085089	0.000319078	0.01791702	0.019927572	0.04124876
2	793203	24717418	4049	54246	2	0.002949704	0.000319533	0.01791702	0	0.02118626
3	1145085	36139342	4049	54246	3	0.002953111	0.000319522	0.01791702	0	0.02118966
4	1497326	47560909	4049	54246	4	0.002956124	0.000319512	0.01791702	0	0.02119266
5	1849840	58982200	4049	54246	5	0.002958415	0.000319504	0.01791702	0	0.02119494
6	2202615	70403227	4049	54246	6	0.002960606	0.000319497	0.01791702	0	0.02119713
7	2555678	81823966	4049	54246	7	0.002963023	0.000319489	0.01791702	0	0.02119953
8	2909028	93244414	4049	54246	8	0.002965431	0.000319481	0.01791702	0	0.02120193
9	3262691	104664554	4049	54246	9	0.002968058	0.000319472	0.01791702	0	0.02120455
10	3616684	116084357	4049	54246	10	0.002970827	0.000319463	0.01791702	0	0.02120731

Figure 13: Power Calculations for ML case - Node 1

Average of the total energy consumption over entire interval for Node 1 = 0.02320227 mW

Graphs Comparison of Power Consumptions (mW) for ML and Non-ML Cases:

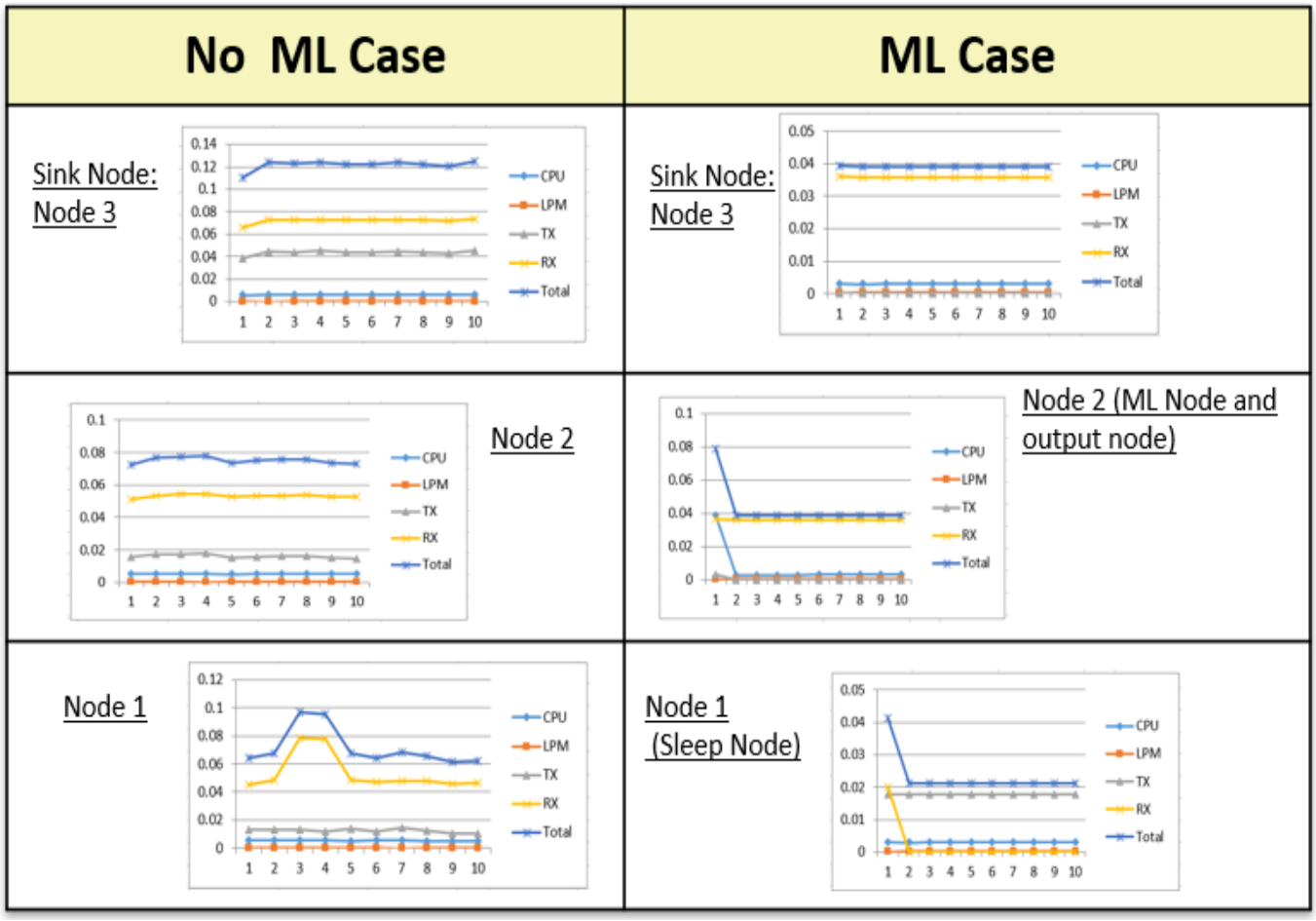

Figure 14: Power Consumption graphs (mW)

Total Average Power Consumption in the Network:

The average energy consumption values for both cases for a duration of 1 hour of simulation timer is as shown below for the ease of comparison. It is seen from the below table that the average energy consumption is higher for individual sensors as well as overall network for the network without using the q-learning algorithm when compared to the one using q-learning algorithm. It is because only one sensor node gets to send the data to the sink/collector node and the possibility of

collisions will not be present in such case. On the other hand, when two sensor nodes send data to the sink/collector node, the possibility of collisions is high on comparison. Other thing to note is that in case of the network using ML, other than the node sending data to sink node, rest all of the sensors will be in sleep mode until the next cycle, which is not happening in the No ML case because of which the overall average energy consumption is less in the ML case.

Time: 60 mins		
Average Energy Consumption in the network(mW)		
	NO ML	ML
Node 1	0.121	0.039
Node 2	0.075	0.043
Node 3	0.071	0.023
Network	0.267	0.105
Sink Node		
ML deciding Node		

Figure 15: Average Energy Consumption (mW) Values

Drawbacks of Cooja Simulator

Larger network was difficult to simulate using the Sky mote emulated sensors present in the Cooja Simulator. When the number of sensor nodes increases in the network, arrays (of type double) used in the reinforcement learning functions like the transition probability array will be larger in dimension. As the size of these arrays increases the sky mote is throwing “Array size too large” error while compiling the firmware onto these sensor nodes. The memory size of 10 KB RAM is not enough for increased number of sensor nodes.

Conclusion and Future Recommendations

The simulation of the any-to-one protocol is performed using two sensor nodes and one sin node for 1 hour of duration. It is seen that the average energy consumption reduces when q-learning algorithm is used in deciding the best sensor node to send the data to sink. As seen that the Cooja Sky motes are not suitable for larger sensor networks, the future works can extend to implement the algorithm for a greater number of nodes in the network using various network simulation softwares.

References

[1] D. Carlson and A. Terzis, “Flip-mac: A density-adaptive contention reduction protocol for efficient any-to-one communication,” *in Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on. IEEE*, 2011, pp. 1–8.

[2] Ala’eddin Masadeh, Ahmed E. Kamal and Zhengdao Wang, “Any-to-One Communication for Internet of Things Networks Using Reinforcement Learning” (unpublished)

[3] Noor Zuriatunadhirah binti Zubir, Aizat Faiz Ramli and Hafiz Basarudin, “Optimization of Wireless sensor Networks MAC protocols Using Machine Learning; A Survey”, *International Conference on Engineering Technology and Technopreneurship (ICE2T)*, 2017.

[4] A. Velinov and A. Mileva, “Running and Testing Applications for Contiki OS Using Cooja Simulator”, *International Conference on Information Technology and Development of Education*, 2016.

[5] http://www.contiki-os.org/index.html

[6] http://thingschat.blogspot.com/2015/04/contiki-os-using-powertrace-and.html

[7] http://www.snm.ethz.ch/snmwiki/pub/uploads/Projects/tmote_sky_datasheet.pdf

[8] https://github.com/contiki-os/contiki/blob/bc2e445817aa546c0bb93a9900093ec276005e2a/apps/powertrace/powertrace.c

[9] https://en.wikipedia.org/wiki/Wireless_sensor_network

[10] https://sourceforge.net/p/contiki/mailman/message/27784437/