

DTEK2011  
WIRELESS SENSOR  
NETWORKS

PREPARED BY  
SAFAYAT BIN HAKIM  
SAMUEL LINDQVIST

# MAC PROTOCOLS AND POWER CONSUMPTION ANALYSIS

CONTIKI NETWORK  
STACK

Network

MAC

Medium Access Control

responsible for sending and receiving packets in the wireless connection.

RDC

Radio-Duty Cycle

reduce the energy consumption by allowing a node to keep it's radio-transceiver off most of the time

Framer

Physical

6-Nov-17

# MAC Protocols in Contiki

6-Nov-17

## NullMAC

- Used for testing purpose
- Does not provide processing, forwards to RDC

## CSMA

- Collision happens
- Retransmission in the next wake-up call

## LPP

- Tries to keep the radio in sleep mode

## TSCH

- Provides deterministic access
- increases network capacity

Also  
Contains  
RDC

## ContikiMAC : RDC protocol in Contiki

6-Nov-17

Default Contiki RDC  
mechanism

**Transmission phase-lock:**  
After a successful  
transmission, the sender  
learns the wake-up phase of  
the receiver results in fewer X-  
missions

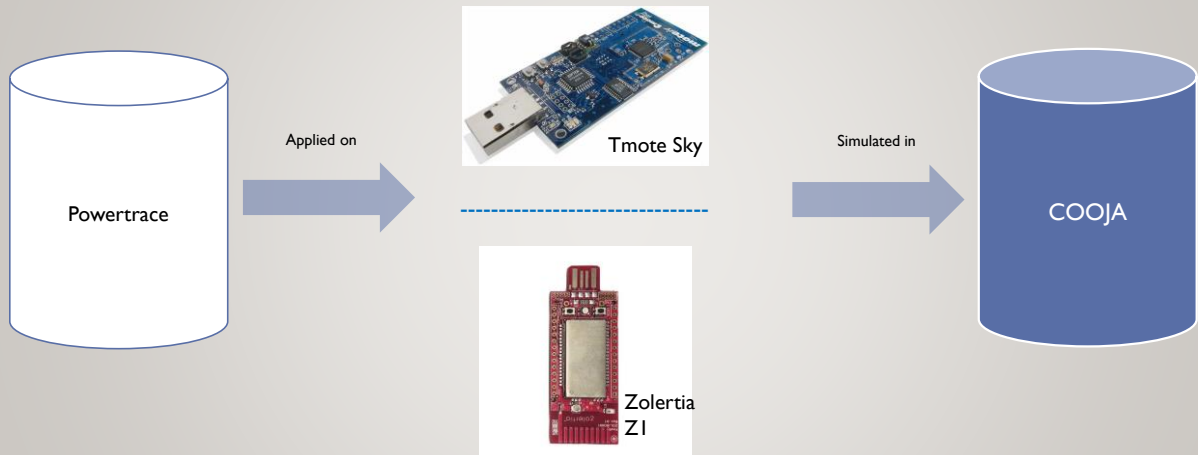
Can keep radio off 99% of  
the time

Based on low-power-  
listening

Enhanced version of  
X-MAC protocol

## Power Analysis Tools

6-Nov-17



## RPL-TSCH Simulation

6-Nov-17

- TSCH stands for **T**ime **S**lotted **C**hannel **H**opping
- MAC layer specified as **2.4 GHz IEEE std. 802.15.4e** platforms
- Nodes may join the network after receiving a beacon from the coordinator.
- Time is divided in time slots, All nodes are synchronized to a given slot-frame
- Nodes update their synchronization relative to their time source parent every time they **RECEIVE** a data or **ACK** frame from it

source:  
 [1] Duquennoy, Simon, et al. "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation." *IEEE DCSS* (2017)  
 [2] Github -> <https://github.com/contiki-os/contiki/tree/master/core/net/mac/tsch>

## Continued ...

6-Nov-17

- ➡ ZI mote is used for TSCH simulation
- ➡ One coordinator, coordinated 20 other nodes
- ➡ noRDC and nullRDC has been used in the RDC layer

```

/***** Enable TSCH *****/
/***** Enable TSCH *****/
/***** Enable TSCH *****/

/* Netstack layers */
#undef NETSTACK_CONF_MAC
#define NETSTACK_CONF_MAC    tschmac_driver
#undef NETSTACK_CONF_RDC
#define NETSTACK_CONF_RDC    nordc_driver
#undef NETSTACK_CONF_FRAMER
#define NETSTACK_CONF_FRAMER framer_802154

/* IEEE802.15.4 frame version */
#undef FRAME802154_CONF_VERSION
#define FRAME802154_CONF_VERSION FRAME802154_IEEE802154E_2012

```

changes in `project-conf.h` file

## Necessary Changes

6-Nov-17

Makefile

```

CONTIKI_PROJECT = my-project

all: $(CONTIKI_PROJECT)

CONTIKI= ../../

APPS += powertrace

CFLAGS += -DPROJECT_CONF_H=\"project-conf.h\"
include $(CONTIKI)/Makefile.include

```

project-conf.h

```

#ifndef __PROJECT_CONF_H__
#define __PROJECT_CONF_H__

/* For RDC driver */

#undef NETSTACK_CONF_RDC
#define NETSTACK_CONF_RDC contikimac_driver
// #define NETSTACK_CONF_RDC nullrdc_driver

/* For MAC driver */

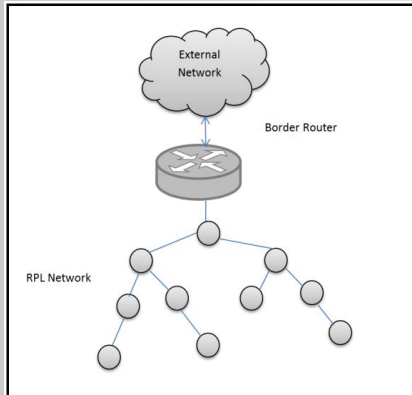
#undef NETSTACK_CONF_MAC
#define NETSTACK_CONF_MAC    nullmac_driver
// #define NETSTACK_CONF_MAC csma_driver
// #define NETSTACK_CONF_MAC tscjmac_driver

#endif /* __PROJECT_CONF_H__ */

```

## Source and Sink Nodes

6-Nov-17



### Border Router

- => Connects one network to another
- => Resides at the edge of the network

### Sky-websense

- => Generates sensing data
- => Provides access to the latest data via built-in webservice.

source: [http://anrg.usc.edu/contiki/index.php/RPL\\_Border\\_Router](http://anrg.usc.edu/contiki/index.php/RPL_Border_Router)

## Testing in Cooja

6-Nov-17

Applications Places

My simulation - Cooja: The Contiki Network Simulator

File Simulation Notes Tools Settings Help

Network

View Zoom

Simulation control

Run Speed limit

Start Pause Step Reload

Time: 00:32.147

Speed: 29.30%

Serial Socket (SERVER) [Sky 1]

Listen port: 60001

Stop

socket -> mote: 159 bytes

mote -> socket: 653 bytes

Status: Client /127.0.0.1:54650 connected.

Mote output

Time	Mote	Message
00:30.791	ID:14	48141 936422 11780 9609 18973 308559 5720 3875
00:30.791	ID:7	108258 877835 33516 29284 50764 278579 16642 12627
00:30.813	ID:20	67130 917355 22618 17089 28357 299170 9156 6963
00:30.814	ID:16	85143 899430 13964 19681 31505 295796 3870 6322
00:30.843	ID:15	110726 875059 20043 27063 38267 290526 6992 8354
00:30.813	ID:10	79838 905126 11496 22648 30389 297651 1840 7295
00:30.935	ID:12	70518 913630 11669 26624 31488 296035 5394 7224
00:31.041	ID:9	51254 932982 12833 15591 18826 308696 3709 5727
00:31.060	ID:19	51099 933381 12379 13719 25112 302395 6508 5426
00:31.080	ID:5	122428 862890 34506 30368 36428 291642 6162 8692
00:31.150	ID:17	61357 923136 16704 16918 25833 301697 6091 6544
00:31.233	ID:13	104716 880011 26528 26277 38447 286527 7123 9263
00:31.233	ID:3	106564 877785 29556 30900 40759 285532 4323 10496
00:31.246	ID:21	60100 924441 9157 16195 20541 306499 0 5067

Filter:

Power calculation is done by the formula used in the previous slide where powertrace introduced

```

00:00:11.55 [D-17] Jailing 'SERVE WEB DEMO' Web server
00:00:11.55 [D-18] Rime started with MAC 01:12:16:10:13:13
00:00:11.64 [D-19] Rime started with MAC 01:12:16:10:13:13
00:00:11.70 [D-20] Rime started with MAC 01:12:16:10:13:13
00:00:11.72 [D-21] nullsec CSPSA ConfigMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
00:00:11.73 [D-22] Rime started with MAC 01:12:16:10:13:13
00:00:11.79 [D-23] MAC 0012:74:03:08:03:03:03 Config2= 2-2450-848000000001:7403:0803:03
00:00:11.83 [D-24] Tentative Link-Link IPv6 address fe80::f000:0000:0000:0212:7403:0000:0000
00:00:11.83 [D-25] Starting 'Sense Web Demo' Web server
00:00:11.87 [D-26] nullsec CSPSA ConfigMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
00:00:11.87 [D-27] MAC 01:12:74:15:00:15:15:15 Config2= 2-2450-848000000001:74:15:00:15:15
00:00:11.90 [D-28] nullsec CSPSA ConfigMAC, channel check rate 8 Hz, radio channel 26, CCA threshold -45
00:00:11.90 [D-29] Tentative Link-Link IPv6 address fe80::f000:0000:0000:0212:7403:0000:0303
00:00:12.00 [D-30] Starting 'Sense Web Demo' Web server
00:00:12.06 [D-31] Tentative Link-Link IPv6 address fe80::0000:0000:0000:0212:7415:0015:1515
00:00:12.06 [D-32] Starting 'Sense Web Demo' Web server
00:00:12.06 [D-33] Server IPv6 addresses:
00:00:12.06 [D-34] aaaa::121:7401:1:1
00:00:12.06 [D-35] fe80::121:7401:1:101
00:00:12.06 [D-36] 8612 320585 0 3572 8612 320585 0 3572
00:00:12.08 [D-37] 9382 320672 0 3307 9382 320672 0 3307
00:00:12.08 [D-38] 13486 316029 2614 3427 13486 316029 2614 3427
00:00:12.08 [D-39] 12805 316392 2615 2755 12805 316392 2615 2755
00:00:12.08 [D-40] 3268 320578 0 3387 3268 320578 0 3387
00:00:12.08 [D-41] 28761 204453 1326 281607 28761 204453 1326 281607
00:00:12.08 [D-42] 9484 319173 0 3428 9484 319173 0 3428
00:00:12.08 [D-43] 15137 314359 3007 2666 15137 314359 3007 2666
00:00:12.08 [D-44] 12722 320823 3167 2763 12722 320823 3167 2763
00:00:12.08 [D-45] 16145 313344 3083 2420 16145 313344 3083 2420
00:00:12.08 [D-46] 13046 316480 3111 2595 13046 316480 3111 2595
00:00:12.08 [D-47] 14447 315059 3397 3363 14447 315059 3397 3363
00:00:12.08 [D-48] 12818 315332 2615 3552 12818 315332 2615 3552
00:00:12.08 [D-49] 9386 320403 0 3775 9386 320403 0 3775
00:00:12.08 [D-50] 12803 316397 2614 2676 12803 316397 2614 2676
00:00:12.08 [D-51] 94131 310797 5336 3844 94131 310797 5336 3844
00:00:12.08 [D-52] 1014 312419 0 2731 1014 312419 0 2731
00:00:12.08 [D-53] 12236 316526 2800 2772 12236 316526 2800 2772
00:00:12.08 [D-54] 13436 316029 2615 3798 13436 316029 2615 3798
00:00:12.08 [D-55] 1959 319408 0 4079 1959 319408 0 4079
00:00:12.29 [D-56] 13728 315005 4616 3626 13728 315005 4616 3626
00:00:12.29 [D-57] 24647 614245 8456 4666 24647 614245 8456 4666

```

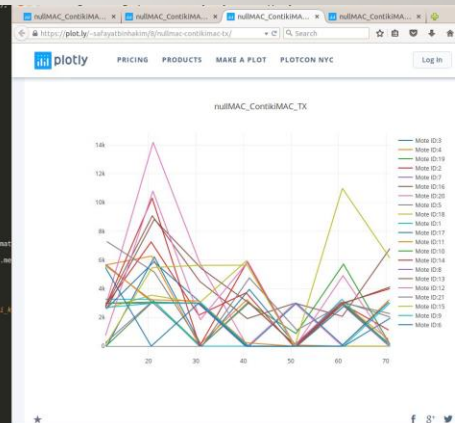
## 6-Nov-17

NumPy  
Plotly

```

100 print('Found %s > strip(count) < PowerTrace measurements')
101
102 print('Generating traces:')
103
104 title = input('Graph title: ')
105
106 # Remove previous average output
107
108 os.remove('averages.txt')
109
110 # Remove previous filebrowser output
111
112 import
113
114 def create_graph(fig, name):
115     full_title = title + ' - %s' % name
116
117     write_full_title = full_title + '*****'
118
119     total = []
120     data = []
121     fig = plt.figure()
122     x_list = []
123     y_list = []
124
125     for single_won in range(1, 1000000):
126         x_list.append(single_won*int(10))
127         y_list.append(single_won*int(10))
128     plt.title(full_title)
129     data.append('create_trace: %s' % x_list, y_list)
130     plt.xlabel('Time - %s' % name)
131     plt.ylabel('Power - %s' % name)
132
133     write_full_title = full_title + '*****'
134
135     layout = fig.layout(title=full_title, width=800, height=400)
136
137     print('Generating graph - %s' % full_title)
138
139     # plotly tools set config file name to FILENAME_USERNAME_and
140     # plotly.tools.set_config_file(username=USERNAME,
141
142     fig = fig.figure(title=full_title, layout=layout)
143
144     # plotly tools plot the graph, filename=full_title.lower()
145
146     def write_name(averages, x):
147         os.write('averages.txt', '%s' % x)
148         x = x + 1
149         if x == 1000000:
150             return
151         else:
152             return
153
154     create_graph(fig, 'CPU')
155     create_graph(fig, 'GPU')
156     create_graph(fig, 'RAM')
157     create_graph(fig, 'SSD')
158     create_graph(fig, 'XX')

```



## Data from PowerTrace

6-Nov-17

ALL_CPU	ALL_LPM	ALL_TX	ALL_RX
11382	317831	80	4088
18715	310759	5709	4207
13845	315358	2611	3295
19531	309690	5634	4123
30892	296605	10311	7202


⇒ Printed values in several states of the Sky mote in the form of the number of **CLOCK TICKS**

⇒ **ALL\_CPU**, values in active mode of CPU

⇒ **ALL\_LPM**, values in Low Power Mode

⇒ **ALL\_TX**, values in Transmit state

⇒ **ALL\_RX**, values in Receive state

source:

[1] Velinov, Aleksandar, and Aleksandra Mileva. "Running and Testing Applications for Contiki OS Using Cooja Simulator." (2016): 279-285.

## Power Consumption Calculation

6-Nov-17

⇒ To estimate power consumption in the simulation.

⇒ Code snippet to exploit the feature:

```
#include "powertrace.h"
powertrace_start(CLOCK_SECOND * 10);
```

$$\text{Power consumption} = (\text{Energy\_Value} * \text{Current} * \text{Voltage}) / (\text{RTIMER\_SECOND} * \text{Runtime})$$

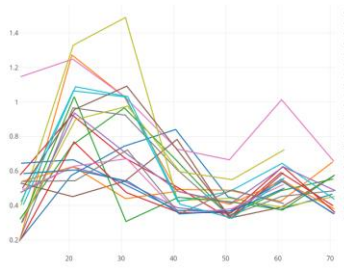
Difference between the number of ticks in two time intervals

Mote voltage and current (from datasheet)

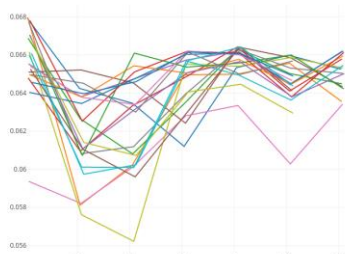
RTimer ticks in one second



CSMA\_ContikiMAC\_CPU



CSMA\_ContikiMAC\_LPM



6-Nov-17

System

Avg Power  
(mW)

MCU

0.61

LPM

0.064

TX

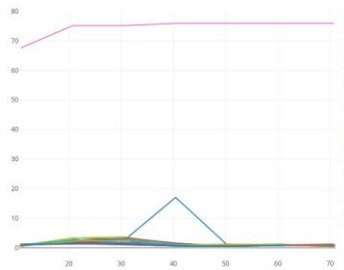
1.23

RX

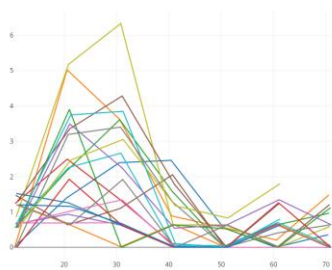
5.2

CSMA +  
ContikiMAC

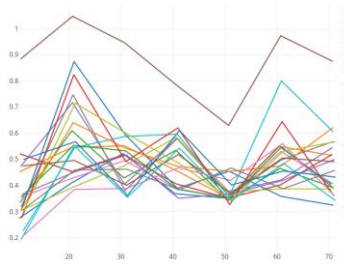
CSMA\_ContikiMAC\_RX



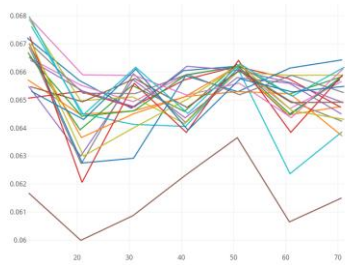
CSMA\_ContikiMAC\_TX



NullMAC\_ContikiMAC\_CPU



NullMAC\_ContikiMAC\_LPM



6-Nov-17

System

Avg Power  
(mW)

MCU

0.47

LPM

0.065

TX

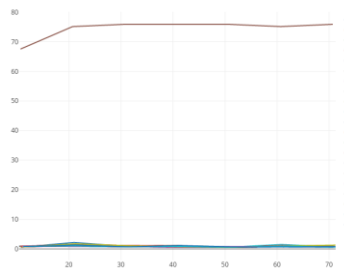
0.63

RX

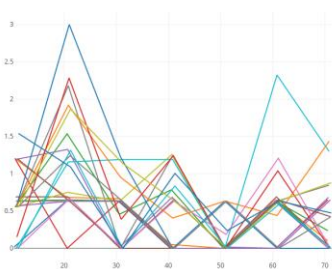
4.37

NullMAC +  
ContikiMAC

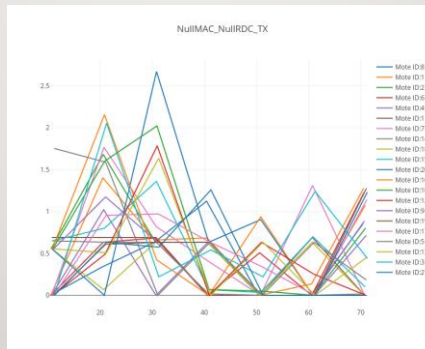
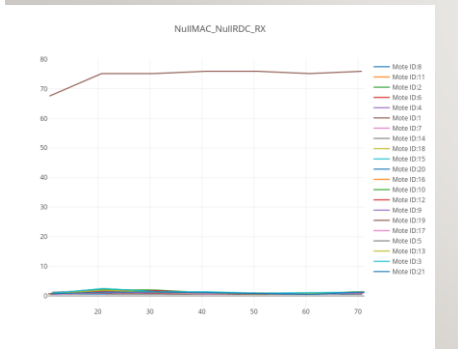
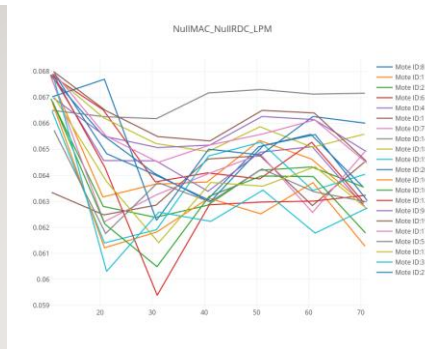
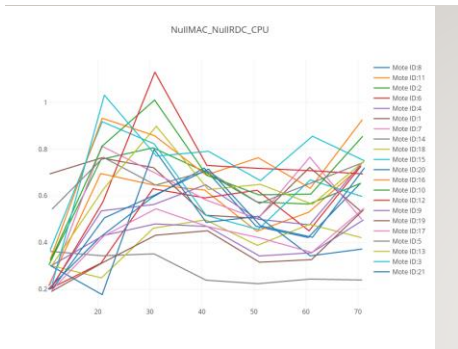
NullMAC\_ContikiMAC\_RX



NullMAC\_ContikiMAC\_TX



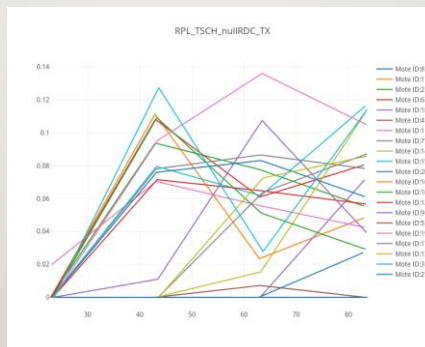
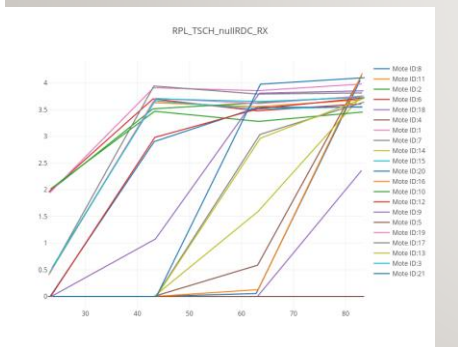
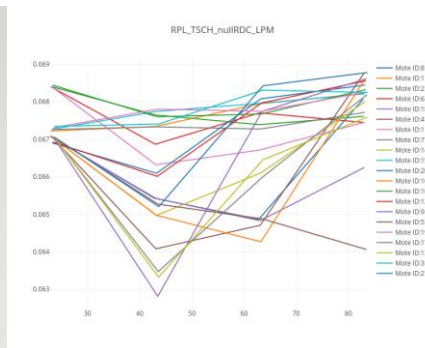
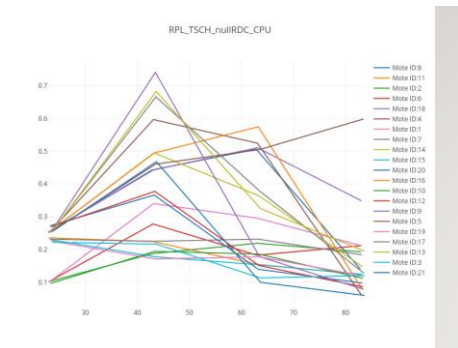




6-Nov-17

System	Avg Power (mW)
MCU	0.52
LPM	0.065
TX	0.58
RX	4.5

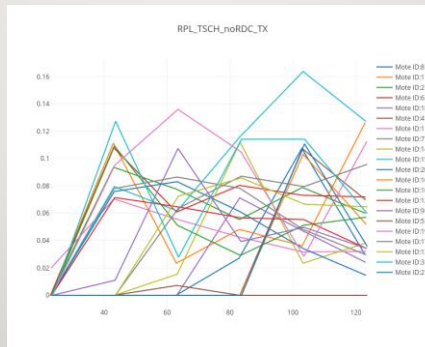
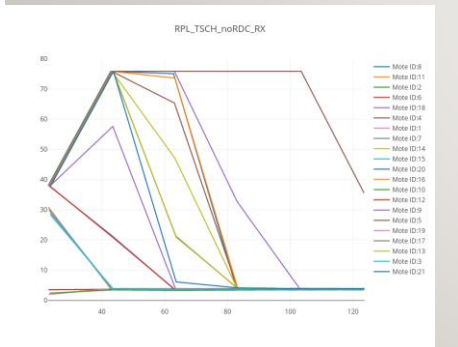
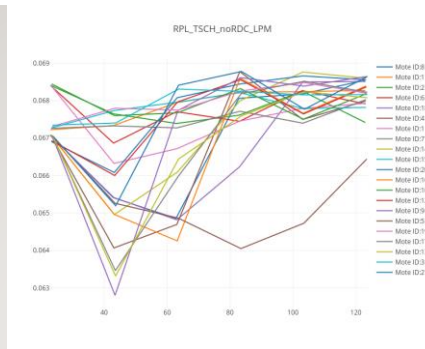
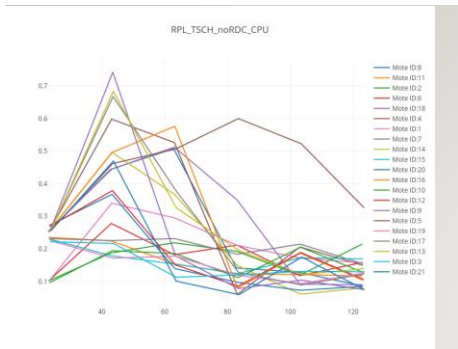
**NullMAC + NullRDC**



6-Nov-17

System	Avg Power (mW)
MCU	0.27
LPM	0.06
TX	0.03
RX	1.65

**TSCH + nullRDC**



6-Nov-17

System	Avg Power (mW)
MCU	0.25
LPM	0.06
TX	0.038
RX	2.5

**TSCH + noRDC**

## Comments on Results...

- NullMAC is not really comparable to CSMA or TSCH as it does not care about whether transmission was successful, addressing or anything else
- We see a lower radio power usage in NullMAC because of this, but it's not viable in practice
- Between CSMA and TSCH there seems to be a clear difference in power usage which would favor TSCH over CSMA
- CSMA is competition based protocol and TSCH represents the time-slotted protocols
  - When looking for a fitting protocol for you project other constraints such as scalability and memory usage and such must also be considered. This analysis was purely from the perspective of power usage.

