References

- B. Krishnamachari, "Autonomous Networks Research Group", *Anrg.usc.edu*, 2016. [Online]. Available: https://anrg.usc.edu/www/. [Accessed: 18- Aug- 2019].
- JNHuaMao Technology Company, "HM Bluetooth module datasheet," Bluetooth 4.0 BLE module Datasheet, Mar. 2015.
- "GATT Services", *Bluetooth.com*, 2019. [Online]. Available: https://www.bluetooth.com/specifications/gatt/services/.
- M. Currey, "HM-10 Bluetooth 4 BLE Modules", *Martyncurrey.com*, 2017. [Online]. Available: http://www.martyncurrey.com/hm-10-bluetooth-4ble-modules/.
- Micrium Embedded Software, *https://www.micrium.com/wp-content/uploads/2014/03/wireless-sensor-network.png.* 2019.

A Appendix

A.1 Wireless Gateway, Contiki: example-broadcast.c

```
/*
 * Copyright (c) 2007, Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the Institute nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * This file is part of the Contiki operating system.
 *
 */

/**
 * \file
 *         Testing the broadcast layer in Rime
 * \author
 *         Adam Dunkels <adam@sics.se>
 */


#include <stdlib.h>
#include <stdio.h>
#include "dev/light-sensor.h"
#include "dev/sht11/sht11-sensor.h"
#include <math.h>
#include "contiki.h"
#include "net/rime/rime.h"
#include "random.h"
#include "dev/button-sensor.h"
#include "dev/leds.h"
#include "cfs/cfs.h"

char message [90]; // Global message, going from sensor to transmit
int timer_duration = 20; // Seconds

// Header is located in contiki/core/sys/process.h
/*---------------------------------------------------------------------------*/
PROCESS(example_broadcast_process, "Broadcast example");
PROCESS(sensor_acq_process,"Sensor Acquisition"); // Adding sensor Process
AUTOSTART_PROCESSES(&example_broadcast_process, &sensor_acq_process); // Adding sensor Process
/*---------------------------------------------------------------------------*/
```

```
static void
broadcast_recv(struct broadcast_conn *c, const linkaddr_t *from)
{

// using the terminal command "sudo make login > filename.txt" to write output to file
  printf("%d.%d %s\n", from->u8[0], from->u8[1], (char *)packetbuf_dataptr());

}
static const struct broadcast_callbacks broadcast_call = {broadcast_recv};
static struct broadcast_conn broadcast;
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(example_broadcast_process, ev, data)
{
  static struct etimer et;

  PROCESS_EXITHANDLER(broadcast_close(&broadcast);)

  PROCESS_BEGIN();

  broadcast_open(&broadcast, 129, &broadcast_call);

  while(1) {

    /* Delay 2-4 seconds */
    //etimer_set(&et, CLOCK_SECOND * 4 + random_rand() % (CLOCK_SECOND * 4));
    etimer_set(&et, CLOCK_SECOND * timer_duration);

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));

    packetbuf_copyfrom(message, 91); // making size one more than message
    broadcast_send(&broadcast);
  }

  PROCESS_END();
}
/*---------------------------------------------------------------------------*/

PROCESS_THREAD(sensor_acq_process,ev,data)
{
      static struct etimer et;
      static int val;
      static float s = 0;
      static int dec;
      static float frac;

    // Placeholders to gather data before going into message
      char message_Temp [30];
      char message_Hum [30];
      char message_Light [30];

      PROCESS_BEGIN();

      printf("Starting Sensor Example.\n");

      while(1)
      {
       etimer_set(&et, CLOCK_SECOND * 2);
       SENSORS_ACTIVATE(light_sensor);
           SENSORS_ACTIVATE(sht11_sensor);

       PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));

       // Temperature
           val = sht11_sensor.value(SHT11_SENSOR_TEMP);
       if(val != -1)
           {
        // s= ((0.01*val) - 39.60); // This one is for Celsius
        s= (((0.01*val) - 39.60)*1.8)+32; // This one is for Fahrenheit
```

```
            dec = s;
            frac = s - dec;

         // Stores the data in the placeholder
         sprintf(message_Temp,"%d.%02u", dec, (unsigned int)(frac * 100));

            }

        // Humidity
        val=sht11_sensor.value(SHT11_SENSOR_HUMIDITY);
        if(val != -1)
            {
         s= (((0.0405*val) - 4) + ((-2.8 * 0.000001)*(pow(val,2))));
            dec = s;
            frac = s - dec;

         // Stores the data in the placeholder
         sprintf(message_Hum,"%d.%02u", dec, (unsigned int)(frac * 100));

            }

        // Light
            val = light_sensor.value(LIGHT_SENSOR_TOTAL_SOLAR);
            if(val != -1)
            {
             s = (float)(val * 0.4071);
             dec = s;
             frac = s - dec;
         // Stores the data in the placeholder
         sprintf(message_Light,"%d.%02u", dec, (unsigned int)(frac * 100));

            }

         // Concatenates the placeholders into the message
        sprintf(message, "%s %s %s", message_Temp, message_Hum, message_Light);

        etimer_reset(&et);
            SENSORS_DEACTIVATE(light_sensor);
            SENSORS_DEACTIVATE(sht11_sensor);


      } //end of while

      PROCESS_END();
}
```

A.2 Wireless Gateway: gatewayProg.py

```
# Implenting the gateway programs in a single program using threading

import threading
import time
import os
import mysql.connector
import datetime

def firstProg():
    # Changes to proper directory
    os.chdir("contiki/examples/rime")

    # Normal command to run program, though saving the output to a text file
    os.system("sudo make TARGET=sky example-broadcast.upload login > /home/pi/Desktop/Gatewaydata.txt")

def secondProg():
    # Need to wait 30 seconds for firstProg to get set up
    time.sleep(30)
    # Needed to change directory where text file is
    #os.system("pwd") # Use for debugging
    os.chdir("/home/pi/Desktop/")

    # Connect to Database
    # Fill in database information below
    mydb = mysql.connector.connect(
        host ="xxxx",
        user="xxxx",
        passwd="xxxx",
        database="xxxx"
    )

    mycursor = mydb.cursor()


    while(1):
        if (os.path.getsize("Gatewaydata.txt") > 0):
            x = datetime.datetime.now() # Get the time
            with open("Gatewaydata.txt", "r+") as file:

                for line in file:
                    if line.strip():
                        # print line
                        split_line = line.split()
                        if len(split_line) == 4:
                            mote_id, temp, hum, light = line.split(" ", 4)
                            print ("id: " + str(mote_id) + ", Temp=: " + str(temp) + ", Hum: " + str(hum) \\
                                + ",Light: " + str(light))
                file.truncate(0) # clears file
            time.sleep(3)
            # sql= "DELETE FROM pi WHERE id = %s"
            # val = (mote_id,)
            # mycursor.execute(sql, val)
            sql2 = "INSERT INTO pi (id, date, temperature, humidity, light) VALUES (%s, %s, %s, %s, %s)"
            val2 = (mote_id, x.strftime("%Y-%m-%d %H:%M:%S"), temp, hum, light)
            mycursor.execute(sql2, val2)
            mydb.commit()
            print(mycursor.rowcount, "record inserted...")
            print(x.strftime("%Y-%m-%d %H:%M:%S"))


t1=threading.Thread(target = firstProg, name = 'thread1')
t2=threading.Thread(target = secondProg, name = 'thread2')

t1.start()
t2.start()
```

A.3 Wireless Gateway: index.php

```
<?php
    session_start();
    ?>
<!DOCTYPE HTML>
<html>
<head>
   <title> Mote Sensor Database</title>
   <center><h1>Mote Sensor Database</h1></center>
   <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
</head>
<body>

<!--PHP code section-->
<?php

    require 'databaseaccess.php';
    include 'getid.php';
    $entries3;
    $v_temp3;


    # For the initial Graph
    $sql_init = "SELECT id FROM pi ORDER BY date DESC LIMIT 1";
    $result_init = $mysqli->query($sql_init);
    $row_init = $result_init->fetch_assoc(); // Just need to get one

    $sql_init2 = "SELECT * FROM pi WHERE id = '".$row_init["id"] ."'";

    $result_init2 = $mysqli->query($sql_init2);

    $entries_init = $result_init2->num_rows;

    $date_init = $temp_init = $hum_init = $light_init = array();
    if ($entries_init > 0) {
        // output data of each row
        while($row = $result_init2->fetch_assoc()) {
            array_push($date_init, $row["date"]);
            array_push($temp_init, $row["temperature"]);
            array_push($hum_init, $row["humidity"]);
            array_push($light_init, $row["light"]);

        }

    }
    else{
        echo "<br>NO RESULTS<br>";
    }



    # Gathers the mote IDs for the dropdown list
    $sql = "SELECT * FROM pi";
    $result = $mysqli->query($sql);

    $v_date2 = $v_temp2 = $v_hum2 = $v_light2 = $v_id = array();
    $entries = $result->num_rows;

    if ($entries > 0) {
        // output data of each row
        while($row = $result->fetch_assoc()) {
            if(!in_array($row["id"], $v_id))
                {array_push($v_id, $row["id"]);}
        }

    }
```

```
    else {
        echo "<br>0 results<br>";
    }



?>
<!---End PHP code section--->

    <style type="text/css">
        .table_titles, .table_cells_odd, .table_cells_even {
                padding-right: 20px;
                padding-left: 20px;
                color: #000;
        }
        .table_titles {
            color: #FFF;
            background-color: #666;
        }
        .table_cells_odd {
            background-color: #CCC;
        }
        .table_cells_even {
            background-color: #FAFAFA;
        }
        table {
            border: 2px solid #333;
        }
        body { font-family: "Trebuchet MS", Arial; }
    </style>

    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['line']});
      google.charts.setOnLoadCallback(load_data);


      // Moving array data from PHP to JS
        var chart;

        function load_data(id){
            if(id == null){
              $.ajax({
                method:"POST",
                dataType:"text",
                success:function()
                {
                    drawChartInitial();
                }
                });
            }
            else {
                $.ajax({
                    url:"fetch.php",
                    method:"POST",
                    data:{id:id},
                    dataType:"json",
                    success:function(data)
                    {
                        drawChart(data);
                    }
                });
            }
        }

        function drawChart(chart_data){
```

```
            var jsonData = chart_data ;
            var data = new google.visualization.DataTable();
                data.addColumn('datetime', 'Time');
                data.addColumn('number', 'Temperature');
                data.addColumn('number', 'Humidity');
                data.addColumn('number', 'Light');

                var Mote_string ='Mote: '; // add id when working
                var options = {
                chart: {
                  title: 'Gateway: 0.1'
                  ,subtitle: Mote_string},
                width: 900,
                height: 500
              };

                var limit = jsonData[0].length;
                var counter =0;
                while (counter < limit){
          data.addRows([
            [new Date(jsonData[0][counter]), Number(jsonData[1][counter]), Number(jsonData[2][counter]), Number(jsonData[3][cour

            counter = counter +1;
            }



                var chart = new google.charts.Line(document.getElementById('chart_loc'));

                chart.draw(data, google.charts.Line.convertOptions(options));
        }

    function drawChartInitial(){
        var counter = 0;
              // Initial values for chart
      var limit = <?php echo $entries_init; ?>;
      var temp = <?php echo json_encode($temp_init); ?>;
      var hum =  <?php echo json_encode($hum_init); ?>;
      var light = <?php echo json_encode($light_init); ?>;
      var Entry_Dates = <?php echo json_encode($date_init); ?>;

        var data = new google.visualization.DataTable();
            data.addColumn('datetime', 'Time');
            data.addColumn('number', 'Temperature');
            data.addColumn('number', 'Humidity');
            data.addColumn('number', 'Light');
            var Mote_string ='Mote: '+ <?php echo $row_init["id"]; ?>;
            var options = {
            chart: {
              title: 'Gateway: 0.1'
              ,subtitle: Mote_string
            },
            width: 900,
            height: 500
          };

        while (counter < limit){
          data.addRows([
            [new Date(Entry_Dates[counter]), Number(temp[counter]), Number(hum[counter]), Number(light[counter])]]);

            counter = counter +1;
            }
            var chart = new google.charts.Line(document.getElementById('chart_loc'));

            chart.draw(data, google.charts.Line.convertOptions(options));
    }

</script>
```

```
<script>

$(document).ready(function(){

    $('#id').change(function(){
        var id = $(this).val();
        if(id != '')
        {
            load_data(id);
        }
    });

});

</script>

<!--Div that will hold the Line chart-->
<div id="chart_loc"></div>

</center>

<!-----------Drop down list of Motes------------>
<!-- Code from w3-->
<script>
function showId(str) {
  if (str=="") {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  if (window.XMLHttpRequest) {
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  } else { // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange=function() {
    if (this.readyState==4 && this.status==200) {
      document.getElementById("txtHint").innerHTML=this.responseText;
    }
  }


// Multiply str, DOUBLE->INT, before passing
  xmlhttp.open("GET","getid.php?q="+(str*1000),true);
  xmlhttp.send();

}
</script>
<!-- END Code from w3-->


<!-- Try to draw chart here -->
<center>
<h4>Mote: </h4>
<select id="select" onchange="showId(this.value); load_data(this.value)">
    <option value=0>Select a Mote</option>
</select>

<script>
    var select = document.getElementById("select"),
        id = <?php echo json_encode($v_id); ?>;

    for(var i =0; i<id.length; i++) {
        var option = document.createElement("OPTION"),
            txt = document.createTextNode(id[i]);
        option.appendChild(txt);
        option.setAttribute("value",id[i]);
        select.insertBefore(option,select.lastChild);
```

```
    }

</script>

<div id="txtHint"><b>Mote info will be listed here.</b></div>

<input type="button" value="Redraw Chart" onClick="load_data()"> </input>

</center>
<!--END Drop down list of Motes-->

</body>

</html>
```

A.4 Wireless Gateway, Web Server: fetch.php

```
<?php

//fetch.php

include('databaseaccess.php');


if(isset($_POST["id"])){

 $query = "SELECT * FROM pi WHERE id = '".$_POST["id"]."' ";
 // ORDER BY date DESC

$statement = $mysqli->query($query);

$date = $temp = $hum = $light = array();

while($row = $statement->fetch_assoc()) {
    array_push($date, $row["date"]);
    array_push($temp, $row["temperature"]);
    array_push($hum, $row["humidity"]);
    array_push($light, $row["light"]);
}

 $output = array($date, $temp, $hum, $light);


}
 echo json_encode($output);

?>
```

A.5 Wireless Gateway, Web Server: databaseaccess.php

```
<?php
    # Insert database information below
    $host = "localhost";
    $username = "xxxx";
    $user_pass = "xxxx";
    $database_in_use = "xxxx";

    $mysqli = mysqli_connect($host, $username, $user_pass, $database_in_use);
    if ($mysqli->connect_errno) {
        echo "Failed to connect to MySQL: (" . $mysqli->connect_errno . ") " . $mysqli->connect_error;
    }

?>
```

A.6 Wireless Gateway, Web Server: getid.php

```
<?php
    session_start();
    ?>
<!DOCTYPE html>
<html>
<head>
<style>
table {
    width: 900;
    border-collapse: collapse;
}

table, td, th {
    border: 1px solid black;
    padding: 5px;
}

th {text-align: left;}
</style>
</head>
<body>
<center>
<?php

require 'databaseaccess.php'; // Connects to database

//Divide $q, INT->DOUBLE, to get the correct mote ID
$q = intval($_GET['q'])/1000;

if($q != 0){

    # The name of the columns and the table theyre from
    $sql = "SELECT * FROM pi WHERE id = '".$q."'";

    $result2 = $mysqli->query($sql);

    $entries2 = $result2->num_rows;
    echo "Entries:" . $entries2 . "<br>";


    echo "<table>
    <tr>
    <th>Mote ID</th>
    <th>Date</th>
    <th>Temperature</th>
    <th>Humidity</th>
    <th>Light</th>
    </tr>";


    if ($entries2 > 0) {
        // output data of each row
        while($row = $result2->fetch_assoc()) {
            echo "<tr>";
            echo "<td>" . $row['id'] . "</td>";
            echo "<td>" . $row['date'] . "</td>";
            echo "<td>" . $row['temperature'] . "</td>";
            echo "<td>" . $row['humidity'] . "</td>";
            echo "<td>" . $row['light'] . "</td>";
            echo "</tr>";
            array_push($v_date2, $row["date"]);
            array_push($v_temp2, $row["temperature"]);
            array_push($v_hum2, $row["humidity"]);
            array_push($v_light2, $row["light"]);
        }
```

```
        echo "</table>";
        }

    $_SESSION['entries3'] = $entries2;
    $_SESSION['v_date3'] = $v_date2;
    $_SESSION['v_temp3'] = $v_temp2;
    $_SESSION['v_hum3'] = $v_hum2;
    $_SESSION['v_light3'] = $v_light2;

} // end of if statement

?>


</center>

</body>
</html>
```

A.7 One-Hop Algorithm: sensor-network-leaf.c

```
#include "contiki.h"
#include "lib/list.h"
#include "lib/memb.h"
#include "lib/random.h"
#include "net/rime/rime.h"
#include "dev/light-sensor.h"
#include "dev/sht11/sht11-sensor.h"
#include <stdio.h>
#include <math.h>

// Struct used to hold the broadcast message
struct broadcast_message {
    uint8_t seqno;
    uint8_t type;
};

// Struct used to hold the type of unicast message
struct unicast_message {
    uint8_t type;
};

// defines the types of unicast messages
enum {
    UNICAST_TYPE_PING,
    UNICAST_TYPE_PONG,
    UNICAST_PARENT_ACK
};

enum {
    BROADCAST_TYPE_LEAF_TO_PARENT
};

struct neighbor {
    // next pointer needed for Contiki lists
    struct neighbor *next;
    // address of the neighbor
    linkaddr_t addr;
    // RSSI = Received Signal Strength Indicator
    // LQI = CC2420 Link Quality Indicator
    uint16_t last_rssi,last_lqi;
    // last sequence number we saw from this neighbor
    uint8_t last_seqno;
    // contains the average seqno gap from this neighbor
    uint32_t avg_seqno_gap;
};

struct parent {
    // next pointer needed for Contiki lists
    struct parent *next;
    // address of the neighbor
    linkaddr_t addr;
    // RSSI = Received Signal Strength Indicator
    // LQI = CC2420 Link Quality Indicator
    uint16_t last_rssi,last_lqi;
    // last sequence number we saw from this neighbor
    uint8_t last_seqno;
    // contains the average seqno gap from this neighbor
    uint32_t avg_seqno_gap;
};


#define MAX_PARENTS 16
MEMB(parents_memb, struct parent, MAX_PARENTS);
LIST(parents_list);
```

```
static struct broadcast_conn broadcast;
static struct unicast_conn unicast;

#define SEQNO_EWMA_UNITY 0X100
#define SEQNO_EWMA_ALPHA 0X040

PROCESS(broadcast_process,"Broadcast process");
PROCESS(unicast_process,"Unicast process");

AUTOSTART_PROCESSES(&broadcast_process,&unicast_process);

static void broadcast_recv(struct broadcast_conn *c, const linkaddr_t *from) {
    //neighbor *n;
    struct broadcast_message *m;
    //uint8_t seqno_gap;

    m=packetbuf_dataptr();
    // ignore leaf to parent broadcasts from other leaf nodes
    if (m->type == BROADCAST_TYPE_LEAF_TO_PARENT) { return; }
    else {;}
}

static const struct broadcast_callbacks broadcast_call = {broadcast_recv};

static void recv_uc(struct unicast_conn *c, const linkaddr_t *from) {
    struct unicast_message *msg;
    struct parent *p;
    msg = packetbuf_dataptr();

    if (msg->type == UNICAST_PARENT_ACK) {
        printf("unicast parent ACK received from %d.%d\n",
                from->u8[0],from->u8[1]);
        for (p=list_head(parents_list); p != NULL; p = list_item_next(p)) {
            if (linkaddr_cmp(&p->addr,from)) {
                return;
            }
        }
        if (p == NULL) {
            p = memb_alloc(&parents_memb);
            if (p == NULL) {
                return; // cant allocate new neighbor in memory so give up
            }
        }
        linkaddr_copy(&p->addr,from);

        list_add(parents_list,p);
        printf("Parent %d.%d added to list\n",from->u8[0],from->u8[1]);
    }
}

static const struct unicast_callbacks unicast_call = {recv_uc};

PROCESS_THREAD(broadcast_process, ev, data) {
    static struct etimer et;
    static uint8_t seqno;
    struct broadcast_message msg;

    PROCESS_EXITHANDLER(broadcast_close(&broadcast););

    PROCESS_BEGIN();

    broadcast_open(&broadcast,129,&broadcast_call);

    // only use broadcast to find this node's parent node
    while (1) {
        // send message every 1-2 seconds
        etimer_set(&et,CLOCK_SECOND*16 + random_rand() % (CLOCK_SECOND*16));
        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
```

```
        if (list_length(parents_list) == 0) {
            msg.seqno = seqno;
            msg.type = BROADCAST_TYPE_LEAF_TO_PARENT;
            packetbuf_copyfrom(&msg,sizeof(struct broadcast_message));
            broadcast_send(&broadcast);
            printf("Sending broadcast for parent\n");
            seqno++;
        }
    }
    PROCESS_END();
}

PROCESS_THREAD(unicast_process,ev,data) {
    PROCESS_EXITHANDLER(unicast_close(&unicast);)
        PROCESS_BEGIN();
    unicast_open(&unicast,146,&unicast_call);

    while(1) {
        static struct etimer et;
        struct unicast_message msg;
        struct parent *p;

        etimer_set(&et,CLOCK_SECOND*16 + random_rand() % (CLOCK_SECOND*16));

        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));

        if(list_length(parents_list) > 0) {
            p = list_head(parents_list);
            printf("sending unicast to %d.%d\n",p->addr.u8[0],p->addr.u8[1]);

            msg.type = UNICAST_TYPE_PING;
            packetbuf_copyfrom(&msg,sizeof(msg));
            unicast_send(&unicast,&p->addr);
        }
    }
    PROCESS_END();
}
```

A.8 One-Hop Algorithm: sensor-network-root.c

```
#include "contiki.h"
#include "lib/list.h"
#include "lib/memb.h"
#include "lib/random.h"
#include "net/rime/rime.h"

#include <stdio.h>

// Struct used to hold the broadcast message
struct broadcast_message {
    uint8_t seqno;
    uint8_t type;
};

// Struct used to hold the type of unicast message
struct unicast_message {
    uint8_t type;
};

// defines the types of unicast messages
enum {
    UNICAST_TYPE_PING,
    UNICAST_TYPE_PONG,
    UNICAST_TYPE_ACK
};

enum {
    BROADCAST_TYPE_LEAF_TO_PARENT
};

struct child {
    // next pointer needed for Contiki lists
    struct child *c;
    // address of the neighbor
    linkaddr_t addr;
    // RSSI = Received Signal Strength Indicator
    // LQI = CC2420 Link Quality Indicator
    uint16_t last_rssi,last_lqi;
    // last sequence number we saw from this neighbor
    uint8_t last_seqno;
    // contains the average seqno gap from this neighbor
    uint32_t avg_seqno_gap;
};

#define MAX_CHILDREN 16

MEMB(children_memb, struct child, MAX_CHILDREN);

LIST(children_list);

static struct broadcast_conn broadcast;
static struct unicast_conn unicast;

#define SEQNO_EWMA_UNITY 0X100
#define SEQNO_EWMA_ALPHA 0X040

PROCESS(broadcast_process,"Broadcast process");
PROCESS(unicast_process,"Unicast process");

AUTOSTART_PROCESSES(&broadcast_process,&unicast_process);

static void broadcast_recv(struct broadcast_conn *con, const linkaddr_t *from) {
    struct child *c;
    struct broadcast_message *m;
    uint8_t seqno_gap;
```

```
    m=packetbuf_dataptr();
    for (c=list_head(children_list); c != NULL; c = list_item_next(c)) {
        if (linkaddr_cmp(&c->addr,from)) {
            break;
        }
    }
    if (c == NULL) {
        c = memb_alloc(&children_memb);
        if (c == NULL) {
            return; // cant allocate new neighbor in memory so give up
        }
    }
    linkaddr_copy(&c->addr,from);
    c->last_seqno = m->seqno - 1;
    c->avg_seqno_gap = SEQNO_EWMA_UNITY;

    list_add(children_list,c);

    c->last_rssi = packetbuf_attr(PACKETBUF_ATTR_RSSI);
    c->last_lqi = packetbuf_attr(PACKETBUF_ATTR_LINK_QUALITY);

    seqno_gap = m->seqno - c->last_seqno;
    c->avg_seqno_gap = (((uint32_t)seqno_gap * SEQNO_EWMA_UNITY) *
            SEQNO_EWMA_ALPHA)/ SEQNO_EWMA_UNITY +
        ((uint32_t)c->avg_seqno_gap * (SEQNO_EWMA_UNITY -
            SEQNO_EWMA_ALPHA)) / SEQNO_EWMA_UNITY;

    c->last_seqno = m->seqno;
    printf("broadcast message received from %d.%d with seqno %d, RSSI %u, LQI %u, avg seqno gap %d.%02d\n",
            from->u8[0],from->u8[1],
            m->seqno,
            packetbuf_attr(PACKETBUF_ATTR_RSSI),
            packetbuf_attr(PACKETBUF_ATTR_LINK_QUALITY),
            (int)(c->avg_seqno_gap / SEQNO_EWMA_UNITY),
            (int)(((100UL * c->avg_seqno_gap)/SEQNO_EWMA_UNITY)%100));

    struct unicast_message *ack;
    ack = packetbuf_dataptr();
    ack->type = UNICAST_TYPE_ACK;
    packetbuf_copyfrom(ack,sizeof(struct unicast_message));
    unicast_send(&unicast,from);
}

static const struct broadcast_callbacks broadcast_call = {broadcast_recv};

static void recv_uc(struct unicast_conn *con, const linkaddr_t *from) {
    struct unicast_message *msg;

    msg = packetbuf_dataptr();

    if (msg->type == UNICAST_TYPE_PING) {
        printf("unicast ping received from %d.%d\n",
                from->u8[0],from->u8[1]);
        msg->type = UNICAST_TYPE_ACK;
        packetbuf_copyfrom(msg,sizeof(struct unicast_message));
        unicast_send(con,from);
    }
}

static const struct unicast_callbacks unicast_call = {recv_uc};

PROCESS_THREAD(broadcast_process,ev,data) {
    static struct etimer et;
    PROCESS_EXITHANDLER(broadcast_close(&broadcast);)

        PROCESS_BEGIN();
```

```
    broadcast_open(&broadcast,129,&broadcast_call);

    while(1) {
        etimer_set(&et,CLOCK_SECOND*8 + random_rand() % (CLOCK_SECOND*8));
        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
    }
    PROCESS_END();
}

PROCESS_THREAD(unicast_process,ev,data) {
    static struct etimer et;
    //struct unicast_message msg;
    //struct child *c;
    PROCESS_EXITHANDLER(unicast_close(&unicast);)
        PROCESS_BEGIN();
    unicast_open(&unicast,146,&unicast_call);

    while(1) {
        etimer_set(&et,CLOCK_SECOND*8 + random_rand() % (CLOCK_SECOND*8));

        PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
    }
    PROCESS_END();
}
```