Application Source Code:

- This is outside project source to build and deploy to the devices.

AP (Orion Border Router)

Makefile

- contain project configurations, such as which module will be used and the config parameter.
- CONTIKI_PROJECT : project name of source.c file
- MODULES: Add module for project (Ex: Orchestra, CoAP, …)
- MAKE_ROUTING: Config for Routing Protocol
- MAKE_NET: Config for Network Layer
- MAKE_MAC: Config for Mac Layer
- CONTIKI: path to source code of Contiki Project

```
1  CONTIKI_PROJECT = ap
2  all:$(CONTIKI_PROJECT)
3  MODULES += os/services/orchestra
4
5  DEFINES += CNLAB_CONF_PROTOCOL_ENABLE
6  MAKE_ROUTING = MAKE_ROUTING_R_RPT
7  MAKE_NET = MAKE_NET_NULLNET
8  MAKE_MAC = MAKE_MAC_TSCH
9
10 CONTIKI = /home/huynn/CNLAB/CNLAB_PROTOCOL_DOWNLINK/contiki-ng
11 include $(CONTIKI)/Makefile.include
```

ap.c

- contain AP Application source code.

```c
#include "contiki.h"
#include "net/mac/tsch/tsch.h"
#include "net/netstack.h"
#include "node-id.h"
#include "dev/button-hal.h"
#include "leds.h"
#include "dev/serial-line.h"
#include "sys/log.h"
#include <string.h>

#define LOG_MODULE "CNLAB_NODE"
#define LOG_LEVEL LOG_LEVEL_INFO

static button_hal_button_t *button;
char uart_buffer[128];
PROCESS(node_process, "Node Process");
AUTOSTART_PROCESSES(&node_process);

PROCESS_THREAD(node_process, ev, data)
{
    PROCESS_BEGIN();
    while (1)
    {
        PROCESS_WAIT_EVENT();
        if(ev == button_hal_press_event) {
            button = (button_hal_button_t *)data;
            LOG_INFO("%s Pressed\r\n",BUTTON_HAL_GET_DESCRIPTION(button));
            leds_toggle(LEDS_GREEN);
            NETSTACK_MAC.on();
        }else if(ev == button_hal_release_event){
            button = (button_hal_button_t *)data;
            LOG_INFO("%s Released\r\n",BUTTON_HAL_GET_DESCRIPTION(button));
            break;
        }
    }
    LOG_INFO("WAITING 'START' CMD FROM SERVER!!!\r\n");
    memset(&uart_buffer, 0, sizeof(uart_buffer));
    while (1)
    {
        PROCESS_YIELD();
        if(ev == serial_line_event_message) {
            int size = snprintf(uart_buffer, sizeof(uart_buffer),"%s", (char *)data);
            LOG_INFO("Message with size %d received from UART: '%s'\r\n",size, (char *)data);
            if(strstr(uart_buffer, "START") != NULL){
                LOG_INFO("Node %d run as Coordinator\r\n", node_id);
                tsch_set_coordinator(1);
                PROCESS_EXIT();
            }
        }
    }
    PROCESS_END();
}
```

Sensor Node (Zolertia RE Mote)

Makefile

- contain project configurations, such as which module will be used and the config parameter.

```
CONTIKI_PROJECT = node node-sht
all:$(CONTIKI_PROJECT)
MODULES += os/services/orchestra
CONTIKI_TARGET_SOURCEFILES += sht25.c
DEFINES += CNLAB_CONF_PROTOCOL_ENABLE
DEFINES += SD_CARD_ENABLE
MAKE_ROUTING = MAKE_ROUTING_R_RPT
MAKE_NET = MAKE_NET_NULLNET
MAKE_MAC = MAKE_MAC_TSCH

CONTIKI = /home/huynn/CNLAB/CNLAB_PROTOCOL_DOWNLINK/contiki-ng
include $(CONTIKI)/Makefile.include
```

node.c

- contain Sensor Node Application source code with DTPS-I2C sensor.

```
#include "contiki.h"
#include "net/mac/tsch/tsch.h"
#include "net/netstack.h"
#include "node-id.h"
#include "dev/button-hal.h"
#include "leds.h"
#include "dev/serial-line.h"
#include "dev/i2c.h"
#include "sys/log.h"
#include <string.h>

#define LOG_MODULE "CNLAB_NODE"
#define LOG_LEVEL LOG_LEVEL_INFO
#define BUTTON_PRESSED 1

static button_hal_button_t *button;
PROCESS(node_process, "Node Process");
PROCESS(node_send_event_process, "send Event Packet");
AUTOSTART_PROCESSES(&node_process);

PROCESS_THREAD(node_send_event_process, ev, data)
{
    PROCESS_BEGIN();
    while (1)
    {
        PROCESS_WAIT_EVENT();
        if(ev == button_hal_press_event) {
```

```
            button = (button_hal_button_t *)data;
            LOG_INFO("%s Pressed\r\n",BUTTON_HAL_GET_DESCRIPTION(button));
            if(cnlab_current_state == CNLAB_PROTOCOL_SEND_DATA){
                cnlab_aperidic_data_output(BUTTON_PRESSED);
            }
        }
    }
    PROCESS_END();
}

PROCESS_THREAD(node_process, ev, data)
{
    float SensorTemp = 0;
    //float ObjTemp = 0;
    uint8_t buf[6];
    uint16_t raw_temp[2];
    PROCESS_BEGIN();
    clock_delay(1000000);
    i2c_init(I2C_SDA_PORT, I2C_SDA_PIN, I2C_SCL_PORT, I2C_SCL_PIN, I2C_SCL_NORMAL_BUS_SPEED);
    while (1)
    {
        PROCESS_WAIT_EVENT();
        if(ev == button_hal_press_event) {
            button = (button_hal_button_t *)data;
            LOG_INFO("%s Pressed\r\n",BUTTON_HAL_GET_DESCRIPTION(button));
            leds_toggle(LEDS_GREEN);
            NETSTACK_MAC.on();
        }else if(ev == button_hal_release_event){
            button = (button_hal_button_t *)data;
            LOG_INFO("%s Released\r\n",BUTTON_HAL_GET_DESCRIPTION(button));
            break;
        }
    }
    process_start(&node_send_event_process, NULL);
    while (1){
        PROCESS_WAIT_EVENT_UNTIL(ev == send_data_event && data != NULL);
        if(i2c_single_send(0x10, 0x80) == I2C_MASTER_ERR_NONE) {
            if(i2c_burst_receive(0x10, buf, 6) == I2C_MASTER_ERR_NONE) {
                raw_temp[0] = buf[0] + buf[1]*256 + buf[2]*65536;
                raw_temp[1] = buf[3] + buf[4]*256 + buf[5]*65536;
                if(buf[2] < 0x80){
                    SensorTemp = (float)raw_temp[0] / 200.0;
                }else{
                    SensorTemp = ( (float)(raw_temp[0]-16777216) ) / 200.0;
                }
                /*if(buf[5] < 0x80){
                    ObjTemp = (float)raw_temp[1] / 200.0;
                }else{
                    ObjTemp = ( (float)(raw_temp[1]-16777216) ) / 200.0;
                }*/
            }
        }
        uint16_t *seqno = (uint16_t *)data;
        cnlab_periodic_data_output(*seqno, (uint8_t)SensorTemp);
    }
    PROCESS_END();
}
```

node-sht.c

- contain Sensor Node Application source code with SHT sensor.

```c
#include "contiki.h"
#include "net/mac/tsch/tsch.h"
#include "net/netstack.h"
#include "node-id.h"
#include "dev/button-hal.h"
#include "leds.h"
#include "dev/serial-line.h"
#include "dev/sht25.h"
#include "sys/log.h"
#include <string.h>

#define LOG_MODULE "CNLAB_NODE"
#define LOG_LEVEL LOG_LEVEL_INFO
#define BUTTON_PRESSED 1

static button_hal_button_t *button;
PROCESS(node_process, "Node Process");
PROCESS(node_send_event_process, "send Event Packet");
AUTOSTART_PROCESSES(&node_process);


PROCESS_THREAD(node_send_event_process, ev, data)
{
    PROCESS_BEGIN();
    while (1)
    {
        PROCESS_WAIT_EVENT();
        if(ev == button_hal_press_event) {
            button = (button_hal_button_t *)data;
            LOG_INFO("%s Pressed\r\n",BUTTON_HAL_GET_DESCRIPTION(button));
            if(cnlab_current_state == CNLAB_PROTOCOL_SEND_DATA){
                cnlab_aperidic_data_output(BUTTON_PRESSED);
            }
        }
    }
    PROCESS_END();
}

PROCESS_THREAD(node_process, ev, data)
{
    int16_t temperature;
    PROCESS_BEGIN();
    SENSORS_ACTIVATE(sht25);
    while (1)
    {
        PROCESS_WAIT_EVENT();
        if(ev == button_hal_press_event) {
            button = (button_hal_button_t *)data;
            LOG_INFO("%s Pressed\r\n",BUTTON_HAL_GET_DESCRIPTION(button));
            leds_toggle(LEDS_GREEN);
            NETSTACK_MAC.on();
        }else if(ev == button_hal_release_event){
            button = (button_hal_button_t *)data;
            LOG_INFO("%s Released\r\n",BUTTON_HAL_GET_DESCRIPTION(button));
            break;
        }
    }
    process_start(&node_send_event_process, NULL);
```

```
60      if(sht25.value(SHT25_VOLTAGE_ALARM)) {
61          LOG_INFO("Voltage is lower than recommended for the sensor operation\n");
62          PROCESS_EXIT();
63      }
64      sht25.configure(SHT25_RESOLUTION, SHT2X_RES_14T_12RH);
65      while (1){
66          PROCESS_WAIT_EVENT_UNTIL(ev == send_data_event && data != NULL);
67          temperature = sht25.value(SHT25_VAL_TEMP);
68          uint16_t *seqno = (uint16_t *)data;
69          cnlab_periodic_data_output(*seqno, (uint8_t)(temperature / 100));
70      }
71      PROCESS_END();
72  }
```

Contiki-NG CNLAB Protocol Implementation Source Code:

- Implement CNLAB Protocol base on current Contiki-NG open source code.
- Supported standard protocols, such as IPv6/6LoWPAN, 6TiSCH, RPL,CoAP, and Orchestra services, …

Handle NodeID to make unique node id.

contiki-ng/os/sys/node-id.h

- Contain node_id global variables to handle the node ID

```
46  #ifndef NODE_ID_H_
47  #define NODE_ID_H_
48
49  /* Total number of nodes
50   *
51   * Author: Huy
52   * Date: 2020/05/15
53   */
54  #define NUMBER_OF_NODES 5
55  /* Type of nodes
56   *
57   * Author: Huy
58   * Date: 2020/08/08
59   */
60  enum CNLAB_NODE_TYPES {
61    AP = 1,
62    SENSOR = 2,
63    ACTUATOR = 3,
64    ROUTER = 4
65  };
66
67  /* A global variable that hosts the node ID */
68  extern uint8_t node_id;
69  extern uint8_t cnlab_node_type;
70  /**
71   * Initialize the node ID. Must be called after initialized of linkaddr
72   */
73  void node_id_init(void);
```

```
/* Initialize the node ID. Set up manually
 *
 * Author: Huy
 * Date: 2019/09/20
 */
void set_node_id(uint8_t id);
/* Set the node types
 *
 * Author: Huy
 * Date: 2020/08/08
 */
void set_node_type(uint8_t type);
#endif /* NODE_ID_H_ */
/**
 * @}
 * @}
 */

```

contiki-ng/os/sys/node-id.c

```
#include "contiki.h"
#include "sys/node-id.h"
#include "net/linkaddr.h"
#include "services/deployment/deployment.h"

uint8_t node_id = 0;
uint8_t cnlab_node_type = 0;

void
node_id_init(void) {
#if BUILD_WITH_DEPLOYMENT
  deployment_init();
#else /* BUILD_WITH_DEPLOYMENT */
  /* Initialize with a default value derived from linkaddr */
  node_id = linkaddr_node_addr.u8[LINKADDR_SIZE - 1]
          + (linkaddr_node_addr.u8[LINKADDR_SIZE - 2] << 8);
#endif /* BUILD_WITH_DEPLOYMENT */
}
/* Initialize the node ID. Set up manually
 *
 * Author: Huy
 * Date: 2019/09/20
 */
void
set_node_id(uint8_t id) {
  node_id = id;
}

/* Initialize the node type. Set up manually
 *
 * Author: Huy
 * Date: 2020/08/08
 */
void
set_node_type(uint8_t type) {
  cnlab_node_type = type;
}
```

contiki-ng/os/contiki-main.c

- Contain init Contiki-NG OS libraries, init platform, and handle main loop of Contiki-NG processes.

```c
#include <stdio.h>
#include <stdint.h>
/*---------------------------------------------------------------------------*/
/* Configuration for each node.
 * Node ID is integer number start from 1.
 * Node Type is AP, SENSOR, ACTUATOR, ROUTER
 * Date: 2020/08/09
 * Author: Huy
 */
#define NODE_ID 1
#define NODE_TYPE AP
/* Log configuration */
#include "sys/log.h"
#define LOG_MODULE "Main"
#define LOG_LEVEL LOG_LEVEL_MAIN
/*---------------------------------------------------------------------------*/
int
#if PLATFORM_MAIN_ACCEPTS_ARGS
main(int argc, char **argv)
{
  platform_process_args(argc, argv);
#else
main(void)
{
#endif
  platform_init_stage_one();

  clock_init();
  rtimer_init();
  process_init();
  process_start(&etimer_process, NULL);
  ctimer_init();
  watchdog_init();

  energest_init();

#if STACK_CHECK_ENABLED
  stack_check_init();
#endif

  platform_init_stage_two();

#if QUEUEBUF_ENABLED
  queuebuf_init();
#endif /* QUEUEBUF_ENABLED */

/* Comment out unused init node id from lladdr.
 * Set up manually in #define NODE_ID, NODE_TYPE.
 *
 * Author: Huy
 * Date: 2020/08/08
 */
  // node_id_init();
  set_node_id(NODE_ID);
  set_node_type(NODE_TYPE);
  netstack_init();

  LOG_INFO("Starting " CONTIKI_VERSION_STRING "\r\n");
  LOG_INFO("- Routing: %s\r\n", NETSTACK_ROUTING.name);
```

```
118     LOG_INFO("- Net: %s\r\n", NETSTACK_NETWORK.name);
119     LOG_INFO("- PHY: IEEE 802.15.4\r\n");
120     LOG_INFO("- MAC: %s\r\n", NETSTACK_MAC.name);
121     LOG_INFO("- 802.15.4 PANID: 0x%04x\r\n", IEEE802154_PANID);
122   #if MAC_CONF_WITH_TSCH
123     LOG_INFO("- 802.15.4 TSCH channel hopping sequence length: %u\r\n", (unsigned)sizeof(TSCH_D
124     LOG_INFO("- 802.15.4 TSCH default hopping sequence: [ ");
125     for(uint8_t hop_index = 0; hop_index < (unsigned)sizeof(TSCH_DEFAULT_HOPPING_SEQUENCE); hop
126       LOG_INFO_("%u ", TSCH_DEFAULT_HOPPING_SEQUENCE[hop_index]);
127     }
128     LOG_INFO_("]\r\n");
129   #else /* MAC_CONF_WITH_TSCH */
130     LOG_INFO("- 802.15.4 Default channel: %u\r\n", IEEE802154_DEFAULT_CHANNEL);
131   #endif /* MAC_CONF_WITH_TSCH */
132     LOG_INFO("Node ID: %u Node Type: %u\r\n", node_id, cnlab_node_type);
133     LOG_INFO("Link-layer address: ");
134     LOG_INFO_LLADDR(&linkaddr_node_addr);
135     LOG_INFO_("\r\n");
136
137   #if NETSTACK_CONF_WITH_IPV6
138     {
139       uip_ds6_addr_t *lladdr;
140       memcpy(&uip_lladdr.addr, &linkaddr_node_addr, sizeof(uip_lladdr.addr));
141       process_start(&tcpip_process, NULL);
142
143       lladdr = uip_ds6_get_link_local(-1);
144       LOG_INFO("Tentative link-local IPv6 address: ");
145       LOG_INFO_6ADDR(lladdr != NULL ? &lladdr->ipaddr : NULL);
146       LOG_INFO_("\r\n");
150     platform_init_stage_three();
151
152   #if BUILD_WITH_RPL_BORDER_ROUTER
153     rpl_border_router_init();
154     LOG_DBG("With RPL Border Router\n");
155   #endif /* BUILD_WITH_RPL_BORDER_ROUTER */
156
157   #if BUILD_WITH_ORCHESTRA
158     orchestra_init();
159     LOG_DBG("With Orchestra\n");
160   #endif /* BUILD_WITH_ORCHESTRA */
161
162   #if BUILD_WITH_SHELL
163     serial_shell_init();
164     LOG_DBG("With Shell\n");
165   #endif /* BUILD_WITH_SHELL */
166
167   #if BUILD_WITH_COAP
168     coap_engine_init();
169     LOG_DBG("With CoAP\n");
170   #endif /* BUILD_WITH_SHELL */
171
172   #if BUILD_WITH_SIMPLE_ENERGEST
173     simple_energest_init();
174   #endif /* BUILD_WITH_SIMPLE_ENERGEST */
175
176   #if BUILD_WITH_TSCH_CS
177     /* Initialize the channel selection module */
178     tsch_cs_adaptations_init();
179   #endif /* BUILD_WITH_TSCH_CS */
```

```
  autostart_start(autostart_processes);

  watchdog_start();

#if PLATFORM_PROVIDES_MAIN_LOOP
  platform_main_loop();
#else
  while(1) {
    uint8_t r;
    do {
      r = process_run();
      watchdog_periodic();
    } while(r > 0);

    platform_idle();
  }
#endif

  return 0;
}
```

Orchestra Services implementation.

contiki-ng/os/services/orchestra/orchestra-conf.h

```
/* Comment out unused ORCHESTRA_RULES
 * Define our rule : cnlab_eb_period
 *
 * Author: Huy
 * Date: 2019/09/20
 */
#define ORCHESTRA_RULES {&cnlab_eb_period}
//#define ORCHESTRA_RULES { &eb_per_time_source, &unicast_per_neighbor_rpl_ns, &default_commor
/* Example configuration for RPL non-storing mode: */
/* #define ORCHESTRA_RULES { &eb_per_time_source, &unicast_per_neighbor_rpl_ns, &default_commo

#endif /* ORCHESTRA_CONF_RULES */

/* Define config of Slotframe size. Prime number was preferred.
 *
 * Author: Huy
 * Date: 2020/05/18
 */
#define CNLAB_SLOTFRAME_SIZE 17
/* Length of the various slotframes. Tune to balance network capacity,
 * contention, energy, latency. */
#ifdef ORCHESTRA_CONF_EBSF_PERIOD
#define ORCHESTRA_EBSF_PERIOD                    ORCHESTRA_CONF_EBSF_PERIOD
#else /* ORCHESTRA_CONF_EBSF_PERIOD */
#define ORCHESTRA_EBSF_PERIOD                    397
#endif /* ORCHESTRA_CONF_EBSF_PERIOD */
```

contiki-ng/os/services/orchestra/orchestra.h

- contain the define structure, functions name of Orchestra.

```
37
38  #ifndef __ORCHESTRA_H__
39  #define __ORCHESTRA_H__
40
41  #include "net/mac/tsch/tsch.h"
42  #include "orchestra-conf.h"
43
44  /* The structure of an Orchestra rule */
45  struct orchestra_rule {
46    void (* init)(uint16_t slotframe_handle);
47    void (* new_time_source)(const struct tsch_neighbor *old, const struct tsch_neighbor *new);
48    int  (* select_packet)(uint16_t *slotframe, uint16_t *timeslot);
49    void (* child_added)(const linkaddr_t *addr);
50    void (* child_removed)(const linkaddr_t *addr);
51  };
52  /* Comment out orchestra rules because we don't use
53   * their rules.
54   * Define our rule : cnlab_eb_period
55   *
56   * Author: Huy
57   * Date: 2019/09/20
58   */
59  // struct orchestra_rule eb_per_time_source;
60  // struct orchestra_rule unicast_per_neighbor_rpl_storing;
61  // struct orchestra_rule unicast_per_neighbor_rpl_ns;
62  // struct orchestra_rule default_common;
63  #if CNLAB_PROTOCOL_ENABLE
64  struct orchestra_rule cnlab_eb_period;
65  #endif /* CNLAB_PROTOCOL_ENABLE */
66  // extern linkaddr_t orchestra_parent_linkaddr;
67  // extern int orchestra_parent_knows_us;
68
69  /* Call from application to start Orchestra */
70  void orchestra_init(void);
71  /* Callbacks requied for Orchestra to operate */
72  /* Set with #define TSCH_CALLBACK_PACKET_READY orchestra_callback_packet_ready */
73  void orchestra_callback_packet_ready(void);
74  /* Set with #define TSCH_CALLBACK_NEW_TIME_SOURCE orchestra_callback_new_time_source */
75  void orchestra_callback_new_time_source(const struct tsch_neighbor *old, const struct tsch_neighbor *new);
76  /* Set with #define NETSTACK_CONF_ROUTING_NEIGHBOR_ADDED_CALLBACK orchestra_callback_child_added */
77  void orchestra_callback_child_added(const linkaddr_t *addr);
78  /* Set with #define NETSTACK_CONF_ROUTING_NEIGHBOR_REMOVED_CALLBACK orchestra_callback_child_removed */
79  void orchestra_callback_child_removed(const linkaddr_t *addr);
80
```

contiki-ng/os/services/orchestra/orchestra.c

- contain the implementation of Orchestra: an autonomous scheduler.

```c
#include "contiki.h"
#include "orchestra.h"
#include "net/packetbuf.h"
/* Comment out unused code : IPv6, RPL, Network Sniffer
 *
 * Author: Huy
 * Date: 2019/09/20
 */
// #include "net/ipv6/uip-icmp6.h"
// #include "net/routing/routing.h"
// #if ROUTING_CONF_RPL_LITE
// #include "net/routing/rpl-lite/rpl.h"
// #elif ROUTING_CONF_RPL_CLASSIC
// #include "net/routing/rpl-classic/rpl.h"
// #endif

// #define DEBUG DEBUG_PRINT
// #include "net/ipv6/uip-debug.h"

/* A net-layer sniffer for packets sent and received */
// static void orchestra_packet_received(void);
// static void orchestra_packet_sent(int mac_status);
// NETSTACK_SNIFFER(orchestra_sniffer, orchestra_packet_received, orchestra_packet_sent);

/* The current RPL preferred parent's link-layer address */
// linkaddr_t orchestra_parent_linkaddr;
/* Set to one only after getting an ACK for a DAO sent to our preferred parent */
// int orchestra_parent_knows_us = 0;

/* The set of Orchestra rules in use */
const struct orchestra_rule *all_rules[] = ORCHESTRA_RULES;
#define NUM_RULES (sizeof(all_rules) / sizeof(struct orchestra_rule *))
/* Comment out unused functions
 *
 * Author: Huy
 * Date: 2019/09/20
 */
/*---------------------------------------------------------------------------*/
// static void
// orchestra_packet_received(void)
// {
// }
/*---------------------------------------------------------------------------*/
// static void
// orchestra_packet_sent(int mac_status)
// {
//   /* Check if our parent just ACKed a DAO */
//   if(orchestra_parent_knows_us == 0
//      && mac_status == MAC_TX_OK
//      && packetbuf_attr(PACKETBUF_ATTR_NETWORK_ID) == UIP_PROTO_ICMP6
//      && packetbuf_attr(PACKETBUF_ATTR_CHANNEL) == (ICMP6_RPL << 8 | RPL_CODE_DAO)) {
//     if(!linkaddr_cmp(&orchestra_parent_linkaddr, &linkaddr_null)
//        && linkaddr_cmp(&orchestra_parent_linkaddr, packetbuf_addr(PACKETBUF_ADDR_RECEIVER))
//       orchestra_parent_knows_us = 1;
//     }
//   }
// }
/*---------------------------------------------------------------------------*/
```

```
164  orchestra_init(void)
165  {
166    int i;
167    /* Snoop on packet transmission to know if our parent knows about us
168     * (i.e. has ACKed at one of our DAOs since we decided to use it as a parent) */
169  /* Comment out unused functions
170   *
171   * Author: Huy
172   * Date: 2019/09/20
173   */
174    // netstack_sniffer_add(&orchestra_sniffer);
175    // linkaddr_copy(&orchestra_parent_linkaddr, &linkaddr_null);
176    /* Initialize all Orchestra rules */
177    for(i = 0; i < NUM_RULES; i++) {
178      if(all_rules[i]->init != NULL) {
179        printf("Orchestra: initializing rule %u\r\n", i);
180        all_rules[i]->init(i);
181      }
182    }
183    printf("Orchestra: initialization done\r\n");
184  }
```

contiki-ng/os/services/orchestra/cnlab-orchestra-rule.c

- contain the implmentation of CNLAB Orchestra rule (Tx at its node id and Rx at all the other timeslots).

```
1   /* cnlab_eb_period rule implementation
2    *
3    * Author: Huy
4    * Date: 2019/09/20
5    */
6   #include "contiki.h"
7   #include "orchestra.h"
8   #include "net/packetbuf.h"
9   #include "sys/node-id.h"
10
11  static uint16_t slotframe_handle = 0;
12  static struct tsch_slotframe *sf_eb;
13  static uint8_t timeslot_space = 1;
14  /*---------------------------------------------------------------------------*/
15  static uint16_t
16  cnlab_get_timeslot()
17  {
18    return (node_id - 1)*timeslot_space;
19  }
20  /*---------------------------------------------------------------------------*/
21  static int
22  select_packet(uint16_t *slotframe, uint16_t *timeslot)
23  {
24    if(slotframe != NULL) {
25      *slotframe = slotframe_handle;
26    }
27    if(timeslot != NULL) {
28      *timeslot = cnlab_get_timeslot();
29    }
30    return 1;
31  }
```

```
/*---------------------------------------------------------------------------*/
static void
init(uint16_t sf_handle)
{
  int i;
  uint16_t tx_timeslot;
  slotframe_handle = sf_handle;
  timeslot_space = CNLAB_SLOTFRAME_SIZE/NUMBER_OF_NODES;
  uint16_t sf_size = CNLAB_SLOTFRAME_SIZE;
  sf_eb = tsch_schedule_add_slotframe(slotframe_handle, sf_size);
  tx_timeslot = cnlab_get_timeslot();
  /*
   * Add a Tx link at node id
   * Add a Rx link at each available timeslot */
  for(i = 0; i < sf_size; i+=timeslot_space) {
    tsch_schedule_add_link(sf_eb,
                           ((i == tx_timeslot) ? LINK_OPTION_TX : LINK_OPTION_RX),
                           LINK_TYPE_ADVERTISING, &tsch_broadcast_address,
                           i, 0);
  }
}
/*---------------------------------------------------------------------------*/
struct orchestra_rule cnlab_eb_period = {
  init,
  NULL,
  select_packet,
  NULL,
  NULL,
};
```

Protocol Configuration:

contiki-ng/os/net/mac/tsch/tsch-const.h

- contain the define constant of TSCH.

```
  /* 4 channels, sequence length 4 */
  #define TSCH_HOPPING_SEQUENCE_4_4 (uint8_t[]){ 20, 22, 23, 25 }
```

contiki-ng/os/net/mac/tsch/tsch-conf.h

- contain the configuration params of TSCH.

```

/* With TSCH_ADAPTIVE_TIMESYNC enabled: keep-alive timeout used after reaching
 * accurate drift compensation. */
/* Original Code
#ifdef TSCH_CONF_MAX_KEEPALIVE_TIMEOUT
#define TSCH_MAX_KEEPALIVE_TIMEOUT TSCH_CONF_MAX_KEEPALIVE_TIMEOUT
#else
#define TSCH_MAX_KEEPALIVE_TIMEOUT (60 * CLOCK_SECOND)
#endif*/
/*Change config
 * Date: 06/11/2020
 * Author: Huy
 */
#ifdef TSCH_CONF_MAX_KEEPALIVE_TIMEOUT
#define TSCH_MAX_KEEPALIVE_TIMEOUT TSCH_CONF_MAX_KEEPALIVE_TIMEOUT
#else
#define TSCH_MAX_KEEPALIVE_TIMEOUT (12 * CLOCK_SECOND)
#endif

/* Max time without synchronization before leaving the PAN */
#ifdef TSCH_CONF_DESYNC_THRESHOLD
#define TSCH_DESYNC_THRESHOLD TSCH_CONF_DESYNC_THRESHOLD
#else
#define TSCH_DESYNC_THRESHOLD (2 * TSCH_MAX_KEEPALIVE_TIMEOUT)
#endif
```

```
/* CNLab Protocol Enable
 *
 * Author: Huy
 * Date: 2020/05/15
 */
#ifdef CNLAB_CONF_PROTOCOL_ENABLE
#define CNLAB_PROTOCOL_ENABLE 1
#else
#define CNLAB_PROTOCOL_ENABLE 0
#endif

/******** Configuration: channel hopping *******/

/* Default hopping sequence, used in case hopping sequence ID == 0 */
/* Use TSCH_HOPPING_SEQUENCE_4_4.
 *
 * Author: Huy
 * Date: 2019/09/20
 */
#ifdef TSCH_CONF_DEFAULT_HOPPING_SEQUENCE
#define TSCH_DEFAULT_HOPPING_SEQUENCE TSCH_CONF_DEFAULT_HOPPING_SEQUENCE
#else
#define TSCH_DEFAULT_HOPPING_SEQUENCE TSCH_HOPPING_SEQUENCE_4_4
#endif
```

```
/******** Configuration: association *******/

/* Start TSCH automatically after init? If not, the upper layers
 * must call NETSTACK_MAC.on() to start it. Useful when the
 * application needs to control when the nodes are to start
 * scanning or advertising.*/
/* Turn off TSCH Auto start, we call NETSTACK_MAC.on() in our application.
 *
 * Author: Huy
 * Date: 2019/09/20
 */
#ifdef TSCH_CONF_AUTOSTART
#define TSCH_AUTOSTART TSCH_CONF_AUTOSTART
#else
#define TSCH_AUTOSTART 0
#endif
/* By default: initialize schedule from EB when associating, using the
 * slotframe and links Information Element */
/* Turn off TSCH init schedule from EB. We use PC-LLF in our application.
 *
 * Author: Huy
 * Date: 2019/09/20
 */
#ifdef TSCH_CONF_INIT_SCHEDULE_FROM_EB
#define TSCH_INIT_SCHEDULE_FROM_EB TSCH_CONF_INIT_SCHEDULE_FROM_EB
#else
#define TSCH_INIT_SCHEDULE_FROM_EB 0
#endif
```

```
/* Size of the ring buffer storing incoming packets.
 * Must be power of two */
#ifdef TSCH_CONF_MAX_INCOMING_PACKETS
#define TSCH_MAX_INCOMING_PACKETS TSCH_CONF_MAX_INCOMING_PACKETS
#else
#define TSCH_MAX_INCOMING_PACKETS 32
#endif
```

```
370  /* Max number of links */
371  #ifdef TSCH_SCHEDULE_CONF_MAX_LINKS
372  #define TSCH_SCHEDULE_MAX_LINKS TSCH_SCHEDULE_CONF_MAX_LINKS
373  #else
374  #define TSCH_SCHEDULE_MAX_LINKS 256
375  #endif
```

```
455  /* TSCH timeslot timing template */
456  #ifdef TSCH_CONF_DEFAULT_TIMESLOT_TIMING
457  #define TSCH_DEFAULT_TIMESLOT_TIMING TSCH_CONF_DEFAULT_TIMESLOT_TIMING
458  #else
459  #define TSCH_DEFAULT_TIMESLOT_TIMING tsch_timeslot_timing_us_15000
460  #endif
```

contiki-ng/os/net/mac/tsch/tsch-timeslot-timing.c

- contain the define constant of TSCH timeslot timing.
- Different between 10ms and 15ms config as below

10ms

```
64  const tsch_timeslot_timing_us
65      1800, /* CCAOffset */
66       128, /* CCA */
67      2120, /* TxOffset */
68     (2120 - (TSCH_CONF_RX_WAIT
69       800, /* RxAckDelay */
70      1000, /* TxAckDelay */
71     TSCH_CONF_RX_WAIT, /* RxWai
72       400, /* AckWait */
73       192, /* RxTx */
74      2400, /* MaxAck */
75      4256, /* MaxTx */
76     10000, /* TimeslotLength */
77  };
```

15ms

```
84  const tsch_timeslot_timing_us
85      1800, /* CCAOffset */
86       128, /* CCA */
87      4000, /* TxOffset */
88     (4000 - (TSCH_CONF_RX_WAIT
89      3600, /* RxAckDelay */
90      4000, /* TxAckDelay */
91     TSCH_CONF_RX_WAIT, /* RxWai
92       800, /* AckWait */
93      2072, /* RxTx */
94      2400, /* MaxAck */
95      4256, /* MaxTx */
96     15000, /* TimeslotLength */
97  };
```

Protocol Implementation:

contiki-ng/os/net/mac/tsch/cnlab-protocol.h

- contain the define constant, data structure, declare of processes and functions of CNLAB Protocol.

```c
/* CNLAB PROTOCOL implementation header.
 *
 * Author: Huy
 * Date: 2019/09/20
 */
#ifdef CNLAB_CONF_PROTOCOL_ENABLE
#ifndef __CNLAB_PROTOCOL__
#define __CNLAB_PROTOCOL__

/********** Includes **********/
#include "contiki.h"
#include "sys/node-id.h"
#include "net/nbr-table.h"
#include "net/linkaddr.h"

#define NUMBER_OF_PROBE_RSSI 30
#define NUMBER_OF_PROBE_PRR 500
#define MAX_TREE_DEPTH 16
#define MIN_PRR_LINK 0.85f
#define MAX_PRR_LINK 0.99f

/* Protocol states */
enum CNLAB_PROTOCOL_STATES {
  CNLAB_PROTOCOL_DEFAULT,
  CNLAB_PROTOCOL_INITIALIZE,
  CNLAB_PROTOCOL_MEASURE_LINK_QUALITY,
  CNLAB_PROTOCOL_MAKE_SCHEDULE,
  CNLAB_PROTOCOL_SEND_DATA,
  CNLAB_PROTOCOL_RESET
};

  /* Uart Packet Types */
  enum CNLAB_UART_PACKET_TYPES {
    ROUTING_UART_PACKET,
    DOWNSCHEDULE_UART_PACKET,
    UPSCHEDULE_UART_PACKET
  };

  /* Handle fixed point */
  typedef uint32_t fixed_point_t;

  /* RSSI Array */
  struct rssi_array
  {
    uint8_t id;
    uint8_t count;
    int16_t rssi_avg;
  }__attribute__((__packed__));
  typedef struct rssi_array rssi_array_t;
```

```c
/* Node Information */
struct node_info {
  uint8_t id;
  uint8_t type;
  uint8_t level;
  fixed_point_t rank;
  uint8_t degree;
  uint8_t primary_parent;
  uint8_t reserve_parent;
  uint8_t next_hop_child;
  uint8_t number_of_nbr;
}__attribute__((__packed__));
typedef struct node_info node_info_t;

/* Neighbor Infomation */
struct nbr_info {
  uint8_t id;
  uint8_t level;
  uint16_t rx_probe_count;
  int16_t avg_rssi;
  fixed_point_t prr_incoming; //Neighbor->Node
  fixed_point_t prr_outgoing; //Node->Neighbor
}__attribute__((__packed__));
typedef struct nbr_info nbr_info_t;

/* Uptree Information */
struct uptree_info {
  uint8_t node;
  fixed_point_t rank;
  uint8_t primary_parent;
  uint8_t reserve_parent;
}__attribute__((__packed__));
typedef struct uptree_info uptree_info_t;

/* Schedule Cell Structure */
struct cell {
  uint16_t timeslot;
  uint8_t channel;
  linkaddr_t lladdr;
}__attribute__((__packed__));
typedef struct cell cell_t;

/* Scheduling Information */
struct schedule_data {
  uint16_t sf_size;
  uint16_t num_transmit_cells;
  uint16_t num_receive_cells;
  cell_t *transmit_cells;
  cell_t *receive_cells;
}__attribute__((__packed__));
typedef struct schedule_data schedule_data_t;
```

```
/* Data Packet */
struct data {
  char cmd[2];
  char value1[3];
  char value2[6];
  char value3[6];
}__attribute__((__packed__));
typedef struct data data_t;

/***** Variables *****/
process_event_t send_data_event;

/***** CNLAB Processes *****/

/* Process Measure Link Quality */
PROCESS_NAME(cnlab_measure_link_process);

/* Process Receive Network Information */
PROCESS_NAME(cnlab_receive_network_info);

/* Process send data packet */
PROCESS_NAME(cnlab_send_data_process);

/* Process send command packet */
PROCESS_NAME(cnlab_send_command_process);

/********** Functions *********/

/* Write log to SD Card files */
void cnlab_write_log_to_file(const unsigned char *data, unsigned int len);

/* Init protocol */
void cnlab_protocol_init();

/* Start protocol */
void cnlab_process_start();

/* Send periodic data packet */
void cnlab_periodic_data_output(uint16_t seq_no, uint8_t temperature);

/* Send command packet */
void cnlab_aperidic_data_output(uint8_t type);

/* Processing received packet */
void cnlab_packet_input();

/* Reset protocol */
void cnlab_protocol_reset();

#endif /* __CNLAB_PROTOCOL__ */
#endif
```