Module: Internet of Things **2019**

Lab Session 4: Sensor Communications

Lecturers: Dr Payam Barnaghi, Dr Chuan H Foh

Demonstrators: Hamidreza Bagheri, Honglin Li, Roonak Rezvani, Narges Pourshahrokhi, Rob Pell

4.1 Introduction

The goal of this lab session is to run and extend a network protocol similar to the Ping protocol.

We will develop a very simple broadcast communication protocol in which each sensor node sends a Ping packet to other nodes via broadcast.

Once a sensor node receives a Ping packet, it replies with a Pong packet on the broadcast channel.

4.2 Exercise

Read and deploy the sensor code, which is provided (next page), on at least two sensor nodes. Note that we have already deployed one sensor node, you only need to deploy another one to achieve the communication.

Try to answer the following questions:

- The broadcast protocol is inefficient. Why? (think about scenarios > 2 nodes)
- How could the protocol be made more efficient?

Try the following extensions:

- Adjust the broadcast time interval.
- Work in a pair and extend the server-side code to provide more meaningful services (such as alarm reporting service).

Sample code:

```
/*
 * Broadcast PingPong
 */

#include "contiki.h"
#include "lib/random.h"
#include "net/rime.h"
#include "etimer.h"
#include <stdio.h>

#define BROADCAST_PORT 1234

#define SEND_INTERVAL (20 * CLOCK_SECOND)

static struct broadcast_conn broadcast;

PROCESS(pingpong, "Ping Pong Process");
AUTOSTART_PROCESSES(&pingpong);

/*
 * Broadcast callback
 * This method gets called in case that a broadcast is received
 */
static void
broadcast_recv(struct broadcast_conn *c, const rimeaddr_t *from)
{
  printf("Broadcast received from %d.%d: '%s'\n",
         from->u8[0], from->u8[1], (char *) packetbuf_dataptr());
}
static const struct broadcast_callbacks broadcast_call = {broadcast_recv};


PROCESS_THREAD(pingpong, ev, data)
{
  static struct etimer et;
  PROCESS_EXITHANDLER(broadcast_close(&broadcast);)
  PROCESS_BEGIN();
  broadcast_open(&broadcast, 129, &broadcast_call);

  while(1) {
    etimer_set(&et, SEND_INTERVAL);
    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&et));
    packetbuf_copyfrom("Pong",5);
    broadcast_send(&broadcast);
    printf("Ping\n");
  }

  PROCESS_END();
}
```