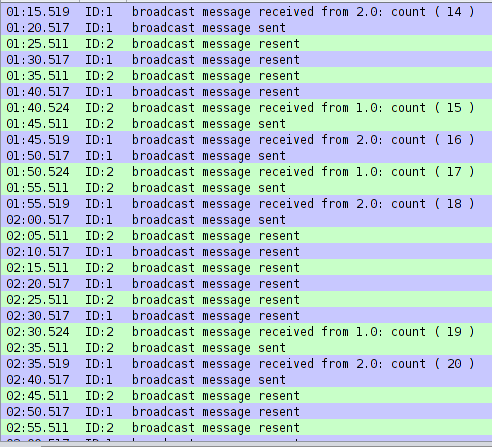
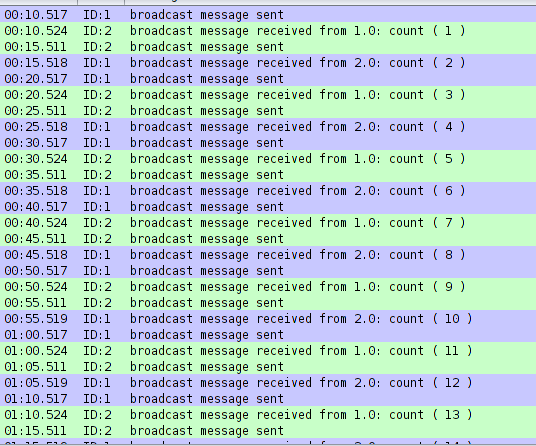
**네트워크 프로토콜 설계 및 실습 과제 2**

전기정보공학부 2016-10769 이권형

**1. 출력화면과 설명**



**그림1.** 시뮬레이션 실행 결과 캡처

그림 1을 통해 알 수 있듯, ID 1의 node가 먼저 packet을 전송하기 시작하면 이후 count를 1씩 증가시키며 packet을 주고받는다. packet count 14에 대해서, 재전송이 일어나는 것을 확인할 수 있는데, 양 쪽에서 packet을 보낸 뒤 10초 후에 재전송을 하는 것을 확인할 수 있다.

#include "contiki.h"

#include "net/rime.h"

#include "random.h"

#include "dev/button-sensor.h"

#include "dev/leds.h"

#include "dev/light-sensor.h"

#include "dev/cc2420.h"

#include <stdio.h>

#include <string.h>

*struct* message {

*uint8\_t* seqno;

};

/\*---------------------------------------------------------------------------\*/

PROCESS(example\_broadcast\_process, "Broadcast example");

AUTOSTART\_PROCESSES(&example\_broadcast\_process);

/\*---------------------------------------------------------------------------\*/

*struct* message msg;

static *void*

broadcast\_recv(*struct* broadcast\_conn \*c, const *rimeaddr\_t* \*from)

{

memcpy(&msg, packetbuf\_dataptr(), sizeof(*struct* message));

printf("broadcast message received from %d.%d: count ( %u )\n", from->u8[0], from->u8[1], msg.seqno);

    //increase global sequence number

    msg.seqno++;

    // Continue broadcast sending

    process\_post(&example\_broadcast\_process, PROCESS\_EVENT\_CONTINUE, NULL);

}

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\_resend, et\_delay;

PROCESS\_EXITHANDLER(broadcast\_close(&broadcast);)

PROCESS\_BEGIN();

broadcast\_open(&broadcast, 129, &broadcast\_call);

while(1) {

        // delay for 5 seconds

        etimer\_set(&et\_delay, CLOCK\_SECOND \* 5);

        PROCESS\_WAIT\_EVENT\_UNTIL(ev==PROCESS\_EVENT\_TIMER);

        // send broadcast packet

        packetbuf\_copyfrom(&msg, sizeof(*struct* message));

        broadcast\_send(&broadcast);

        printf("broadcast message sent\n");

        // Resend packet until callback continue me

        etimer\_set(&et\_delay, CLOCK\_SECOND \* 10);

        // Timer expired or callback function wake up me!

        PROCESS\_WAIT\_EVENT();

        // while callback function send me event, resned packet

        while (!(ev==PROCESS\_EVENT\_CONTINUE)) {

            if (ev == PROCESS\_EVENT\_TIMER) {

                packetbuf\_copyfrom(&msg, sizeof(*struct* message));

                broadcast\_send(&broadcast);

                printf("broadcast message resent\n");

                // reset timer for resend

                etimer\_reset(&et\_delay);

            }

            PROCESS\_WAIT\_EVENT();

        }

}

PROCESS\_END();

}

/\*---------------------------------------------------------------------------\*/

**코드1.** sender node의 source code

sender의 경우 처음 packet을 보내기 시작하는 node이다. delay를 위해 5초간 timer를 설정하고, 이후 broadcast message를 send한다. 이후에 resend를 위해 10초의 timer를 setting하고, event를 기다린다. event가 발생하면 계속하는데, PROCESS\_EVENT\_CONTINUE이면 callback함수에 의해 발생한 event이므로 loop의 처음으로 돌아가 5초간 기다린 뒤 packet을 sending한다. PROCESS\_EVENT\_TIMER가 발생하면 resend timer에 의해 발생한 것이므로, packet을 resend한뒤 다시 timer를 setting하도록 loop를 만들었다.

receiver의 경우 가장 처음 while loop에 들어가기 전에 PROCESS\_WAIT\_EVENT()를 만들어 packet을 받았을 때 loop에 진입하도록 하였다.

callback 함수에서, packet을 성공적으로 받은 후에는 PROCESS\_EVENT\_CONTINUE를 전달해 packet을 보낼 수 있도록 구현하였다.

sequence number는 global 변수로 구현되어 있다. 따라서 callback과 process가 공유하여 사용하고, callback함수에서 sequence number를 받아 저장하고 1 증가시킨다. process는 이를 packet으로 보낸다.