

```

/* -----
(c) Riduan M ABID, eMail: R.Abid@aui.ma
This code was implemented as a partial requirement for
a Ph.D degree in Computer Science at Auburn University
Supervisor: Dr. Biaz - eMail: biazsaa@auburn.edu,
Shelby Center for Engineering Technology- Auburn Unveristy - 2009,
----- */

#include <sys/socket.h>
#include <linux/if_packet.h>
#include <linux/if_ether.h>
#include <linux/if.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <asm/types.h>
#include <sys/time.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define MP_COUNT 10
#define R_SINRs_SHARED_MEMORY_SIZE (MP_COUNT+1) * sizeof(struct reverse_SINR)
#define KEY_R_SINR 1331131

struct PREQ {
    __u8 eltID;
    __u8 hop_count;
    __u8 MPP_mac_addr[6];
    __u32 MPP_seq_no;
    double metric;
};

struct reverse_SINR {
    __u8 mac_addr[6];
    double R_SINR, avg_rssi, avg_rate, avg_length;
    double interference_RSSI_sum;
};

void print_R_SINRs(struct reverse_SINR *table);
void print_mac_addr(__u8 * mac);

int main(int argc, char* argv[]) {

    int raw_sock, fd, i, PREQs_BROADCAST_PERIOD, PREQ_size = sizeof(struct PREQ);
    __u8 dest_mac_addr[6] = {0xff, 0xff, 0xff, 0xff, 0xff, 0xff},
MPP_mac_addr[6], buffer[4096], *shm;
    struct sockaddr_ll sll;
    struct ifreq ifr;
    struct PREQ PREQ;
    struct reverse_SINR R_SINRs[MP_COUNT+1];
    int shmid_R_SINRs;
    key_t key_R_SINRs;
    double CCI;

```

```

// Checking for Arguments,
if (argc < 3) {
    perror("\n Inussufficient Arguments ");
    perror("\n Usage: <executable> <Interface_Name>
<PREQ_BROADCAST_PERIOD>");
    exit(-1);
}
PREQs_BROADCAST_PERIOD = atoi(argv[2]); // in Seconds,

// Getting Local MAC Address,
fd = socket(AF_INET, SOCK_DGRAM, 0);
ifr.ifr_addr.sa_family = AF_INET;
strncpy(ifr.ifr_name, argv[1], IFNAMSIZ-1);
ioctl(fd, SIOCGIFHWADDR, &ifr);
close(fd);
memcpy(MPP_mac_addr, ifr.ifr_hwaddr.sa_data, 6);

// Finding Local Wireless Interface Index,
if ((raw_sock = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL))) < 0 ) {
    perror("socket");
    exit(-1);
}
memset(&ifr, 0, sizeof(ifr));
strcpy(ifr.ifr_name, argv[1]);
if ( ioctl(raw_sock, SIOCGIFINDEX, &ifr) < 0) {
    perror("ioctl (SIOCGIFINDEX) ");
    exit(-1);
}

// Bind the socket to the interface,
memset(&sll, 0, sizeof(sll));
sll.sll_family = AF_PACKET;
sll.sll_protocol = ETH_P_ALL;
sll.sll_ifindex = ifr.ifr_ifindex;
if ( bind(raw_sock, (struct sockaddr *) &sll, sizeof(sll)) < 0) {
    perror("Error Binding to ETH_P_ALL");
    exit(-1);
}

/* Set the MAC address of source and destination */
memcpy(sll.sll_addr, dest_mac_addr, 6);
sll.sll_addr[6] = 0x00;
sll.sll_addr[7] = 0x00;
memcpy(buffer, dest_mac_addr, 6);
memcpy(buffer+6, MPP_mac_addr, 6);

// Locating R_SINRs Shared Memory,
key_R_SINRs = KEY_R_SINR;
if ((shmid_R_SINRs = shmget(key_R_SINRs, R_SINRs_SHARED_MEMORY_SIZE,
IPC_CREAT)) < 0) {
    perror("shmget");
    exit(1);
}
if ((shm = shmat(shmid_R_SINRs, NULL, 0)) == NULL) {

```

```

        perror("shmat");
        exit(1);
    }

    // Filling-Initial PREQ,
    PREQ.eltID = 13;
    PREQ.hop_count = 0;
    memcpy(PREQ.MPP_mac_addr, MPP_mac_addr, 6);
    PREQ.MPP_seq_no = 0;
    PREQ.metric = 0;
    while (1) {
        PREQ.MPP_seq_no++;
        memcpy(buffer+12, &PREQ, PREQ_size);
        // Reading R_SINRs Entries,
        memcpy(R_SINRs, shm, R_SINRs_SHARED_MEMORY_SIZE);
        // Adding R_SINRs Entries to PREQ,
        memcpy(buffer+12+PREQ_size, R_SINRs, R_SINRs_SHARED_MEMORY_SIZE);
        // Sending,
        if ((i = sendto(raw_sock, buffer,
12+PREQ_size+R_SINRs_SHARED_MEMORY_SIZE, 0x00, (struct sockaddr *)&sll,
sizeof(sll))) < 0 ) {
            perror("sendto");
            exit(-1);
        }
        printf("\n -----");
        printf("\n --- Sent %d Bytes - Seq_no: %d, \n", i, PREQ.MPP_seq_no);
        print_R_SINRs(R_SINRs);
        printf("\n CCI: %.2f", R_SINRs[MP_COUNT].R_SINR);
        sleep(PREQs_BROADCAST_PERIOD);
    }

    return 0;
}

void print_R_SINRs(struct reverse_SINR *R_SINRs) {
    int i;
    for (i = 0; i < MP_COUNT; i++) {
        printf("\n *"); print_mac_addr(R_SINRs[i].mac_addr);
        printf(" - R_SINR: %.2f", R_SINRs[i].R_SINR);
    }
}

void print_mac_addr(__u8 * mac) {
    int i;
    for (i = 0; i < 6; i++)
        printf("%2x:", mac[i]);
}

```