

```

/* -----
   (c) Riduan M. ABID - eMail: R.Abid@aui.ma, abidmoh@auburn.edu,
   This Work is relevant to a Ph.D Dissertation,
   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 <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <asm/types.h>
#include <sys/ioctl.h>
#include <sys/time.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define PROBE_FRAME_SIZE 1024
#define MP_COUNT 10
#define SHARED_MEMORY_SIZE MP_COUNT * sizeof(struct Received_Probe)
#define KEY 55555

struct Received_Probe { // Received Probe Frames Per MP,
    __u8 mac_addr[6];
    __u8 value;
};

void print_Received_Probes (struct Received_Probe *Ri);
void print_mac_addr(__u8 * mac) ;

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

    int raw_sock, fd, i, sec, shmid;
    struct sockaddr_ll sll;
    struct ifreq ifr;
    struct Received_Probe Ri[MP_COUNT];
    __u8 probe_frame[PROBE_FRAME_SIZE], dest_mac_addr[6] = {0xff, 0xff, 0xff,
0xff, 0xff, 0xff}, local_mac_addr[6], null_addr[6] =
{0x00,0x00,0x00,0x00,0x00,0x00};
    __u8 *shm, eltID, mp_count = MP_COUNT, cursor;
    key_t key = KEY;

    // Checking for Arguments,
    if (argc < 2) {
        perror("\n Inussufficient Arguments ");
        perror("\n Usage: <executable> <Interface_Name>");
        exit(-1);
    }
}

```

```

}
// 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(local_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(probe_frame, dest_mac_addr, 6);
memcpy(probe_frame+6, local_mac_addr, 6);
eltID = 77; // Arbitrary,
memcpy(probe_frame+12, &eltID, 1);
memcpy(probe_frame+13, &mp_count, 1);

// Locating Shared Memory,
if ((shmid = shmget(key, SHARED_MEMORY_SIZE, IPC_CREAT)) < 0) {
    perror("shmget");
    exit(1);
}

// Initializing Ri - Received Probes - for initial broadcast,
for (i = 0; i < MP_COUNT; i++) {
    Ri[i].value = 0;
    memcpy(&Ri[i].mac_addr, null_addr, 6);
}

sec = 1;

```

```

while (1) {
    // Filling Probe,
    cursor = 14;
    for (i = 0; i < MP_COUNT; i++) {
        memcpy(probe_frame+cursor, Ri[i].mac_addr, 6);
        memcpy(probe_frame+cursor+6, &Ri[i].value, 1);
        cursor += 7;
    }

    // Broadcast the probe, - Including Ri,
    if ((i = sendto(raw_sock, probe_frame, PROBE_FRAME_SIZE, 0x00, (struct
sockaddr *)&sll, sizeof(sll))) < 0 ) {
        perror("sendto");
        exit(-1);
    }
    // Feedback,
    printf("\n Sent Bytes: %d --- Probe Frame Second: %d , \n", i, sec);
    // Refereshing Ri after a period of 10 Seconds,
    if (sec % 10 == 0) {
        printf("\n ----- Refereshing Ri ----- ");
        // READ Shared Memory - Ri
        if ((shm = shmat(shmid, NULL, 0)) == NULL) {
            perror("shmat");
            exit(1);
        }
        memcpy(Ri, shm, SHARED_MEMORY_SIZE);
        print_Received_Probes(Ri);
    }
    sleep(1);
    sec++; // Next Second,
}
return 0;
}

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

void print_Received_Probes (struct Received_Probe *Ri) {
    int i;

    for (i = 0; i < MP_COUNT; i++) {
        printf("\n-----");
        print_mac_addr(Ri[i].mac_addr);
        printf(" --- Probes: %d", Ri[i].value);
    }
}

```