

```
/* -----  
    (c) Riduan M. ABID - eMail: R.Abid@auburn.edu, 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 <stdio.h>  
#include <stdlib.h>  
#include <sys/socket.h>  
#include <linux/if_packet.h>  
#include <linux/if_ether.h>  
#include <linux/if.h>  
#include <netinet/in.h>  
#include <asm/types.h>  
#include <sys/ioctl.h>  
#include <string.h>  
#include <sys/ipc.h>  
#include <sys/shm.h>
```

```
#define MAP_COUNT 10  
#define KEY 12345  
#define SHARED_MEMORY_SIZE  sizeof(struct MPP_Proxying_Table)  
#define RREP_SIZE  sizeof(struct RREP)
```

```
struct RREP {  
    __u8 eltID;  
    __u8 MPP_mac_addr[6];  
    __u8 MAP_mac_addr[6];  
    __u32 MAP_seq_no;  
    __u8 hop_count;  
    double metric;  
};
```

```
struct MAP_Proxy_Entry {  
    __u8 MAP_mac_addr[6];  
    __u8 next_hop[6];  
    __u32 seq_no;  
    __u8 hop_count;  
    double metric;  
};
```

```
struct MPP_Proxying_Table {  
    struct MAP_Proxy_Entry MAP_Entries[MAP_COUNT];  
    int visible_MAPs;  
};
```

```
__u8 local_mac_addr[6], src_mac_addr[6], dest_mac_addr[6], MAP_mac_addr[6],  
NULL_addr[6] = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0};
```

```
int get_MAP_Entry(struct MPP_Proxying_Table table, __u8 *mac_addr);  
void print_MPP_Proxying_Table(struct MPP_Proxying_Table table);  
void print_mac_addr(__u8 * mac);  
int comp_mac_addr(__u8 * mac1, __u8 * mac2);
```

```

int main(int argc, char* argv[]) {
    __u8 buffer[4096], eltID, *shm;
    struct sockaddr dest;
    int dlen, i, recievedBytes, currentMAP, len, cnt = 0;
    int shmid, raw_sock, fd;
    key_t key;
    struct RREP rrep;

    struct ifreq ifr;
    struct sockaddr_ll sll;
    struct MPP_Proxying_Table MPP_proxying_table, MPP_proxying_table2;
    struct MAP_Proxy_Entry MAP_Entry;

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

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

    // Finding Ad-Hoc-ifr Index,
    if ((raw_sock = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL))) < 0 ) {
        perror("socket error");
        exit(-1);
    }
    memset(&ifr, 0, sizeof(ifr));
    strncpy(ifr.ifr_name, argv[1], IFNAMSIZ);
    if ( ioctl(raw_sock, SIOCGIFINDEX, &ifr) < 0) {
        perror("ioctl (SIOCGIFINDEX) ");
        exit(-1);
    }

    // Bind "sll" to the wireless ifr,
    memset(&sll, 0, sizeof(sll));
    sll.sll_family = AF_PACKET;
    sll.sll_protocol = htons(ETH_P_ALL);
    sll.sll_ifindex = ifr.ifr_ifindex;

    if (bind(raw_sock, (struct sockaddr *)&sll, sizeof(sll)) != 0) {
        perror("Error Binding Raw_Sock");
        exit(-1);
    }

    // Creating Shared Memory for MPP_Proxying_Table,
    key = KEY;
    if ((shmid = shmget(key, SHARED_MEMORY_SIZE, IPC_CREAT)) < 0) {

```

```

        perror("shmget");
        exit(1);
    }
    if ((shm = shmat(shmid, NULL, 0)) == NULL) {
        perror("shmat");
        exit(1);
    }
    // Initializing the MAP_Proxying_Table,
    for (i = 0; i < MAP_COUNT; i++) {
        memcpy(MPP_proxying_table.MAP_Entries[i].MAP_mac_addr, NULL_addr, 6);
        memcpy(MPP_proxying_table.MAP_Entries[i].next_hop, NULL_addr, 6);
        MPP_proxying_table.MAP_Entries[i].seq_no = 0;
        MPP_proxying_table.MAP_Entries[i].hop_count = 0;
        MPP_proxying_table.MAP_Entries[i].metric = 0;
    }
    MPP_proxying_table.visible_MAPs = 0;

    while(1) {
        recievedBytes = recvfrom(raw_sock,buffer,4096,0,&dest,&dlen);
        memcpy(src_mac_addr, buffer+6, 6);
        if (comp_mac_addr(src_mac_addr, local_mac_addr))
            continue; // Avoiding local loops,

        // Getting Frame eltID,
        memcpy(&eltID, buffer+12, 1);

        // Checking if it is a RREP frame,
        if (eltID != 5)
            continue;
        // Feedback,
        memcpy(dest_mac_addr, buffer, 6);
        if (!comp_mac_addr(local_mac_addr, dest_mac_addr)) // In case ath is in
monitor mode,
            continue;
        printf("\n --- %d --- RREP received from: ", cnt++);
        print_mac_addr(src_mac_addr);

        // Reading RREP,
        memcpy(&rrep, buffer+12, RREP_SIZE);

        // Getting the MAP MAC address,
        memcpy(MAP_mac_addr, rrep.MAP_mac_addr, 6);

        // Getting corresponding MAP Entry from MPP_Forwarding_Table,
        currentMAP = get_MAP_Entry(MPP_proxying_table, MAP_mac_addr);
        if (currentMAP == -1) { // There is NO entry, create one
            currentMAP = MPP_proxying_table.visible_MAPs;
            MPP_proxying_table.visible_MAPs++;
            memcpy(MPP_proxying_table.MAP_Entries[currentMAP].MAP_mac_addr,
MAP_mac_addr, 6);
            memcpy(MPP_proxying_table.MAP_Entries[currentMAP].next_hop,
src_mac_addr, 6);
            MPP_proxying_table.MAP_Entries[currentMAP].seq_no = 0;
            MPP_proxying_table.MAP_Entries[currentMAP].metric = 0;

```

```

    }
    memcpy(MPP_proxying_table.MAP_Entries[currentMAP].next_hop,
src_mac_addr, 6);
    MPP_proxying_table.MAP_Entries[currentMAP].hop_count = ++rrep.hop_count;
    MPP_proxying_table.MAP_Entries[currentMAP].metric = rrep.metric;
    // Writing next_hop to shared memory,
    print_MPP_Proxying_Table(MPP_proxying_table);
    memcpy(shm, &MPP_proxying_table, SHARED_MEMORY_SIZE);
}
}
int get_MAP_Entry(struct MPP_Proxying_Table table, __u8 *mac_addr) {
    int i;
    for (i = 0; i < MAP_COUNT; i++)
        if (comp_mac_addr(table.MAP_Entries[i].MAP_mac_addr, mac_addr))
            return i;
    return -1;
}

void print_mac_addr(__u8 * mac) {
    int i;
    for (i = 0; i < 6; i++)
        printf("%2x:", mac[i]);
}
int comp_mac_addr(__u8 * mac1, __u8 * mac2) {
    int i;
    for (i = 0; i < 6; i++)
        if (mac1[i] != mac2[i])
            return 0;
    return 1;
}
void print_MPP_Proxying_Table(struct MPP_Proxying_Table table) {
    int i;
    printf("\n ----- MPP Proxying Table ----- ");
    printf("\n - Visible MAPs: %d", table.visible_MAPs);
    for (i = 0; i < table.visible_MAPs; i++) {
        printf("\n ***** MAP: %d", i+1);
        printf("\n ----- MAC addr: ");
print_mac_addr(table.MAP_Entries[i].MAP_mac_addr);
        printf("\n ----- Next hop: ");
print_mac_addr(table.MAP_Entries[i].next_hop);
        printf("\n ----- Metric: %.2f",table.MAP_Entries[i].metric);
        printf("\n ----- Hop count: %d",table.MAP_Entries[i].hop_count);
    }
    printf("\n ----- ");
}
}

```