/\*

\* This program is free software: you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation, either version 3 of the License, or

\* (at your option) any later version.

\*/

#include <arpa/inet.h>

#include <linux/if\_packet.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/ioctl.h>

#include <sys/socket.h>

#include <net/if.h>

#include <netinet/ether.h>

#define MY\_DEST\_MAC0 0x00

#define MY\_DEST\_MAC1 0x00

#define MY\_DEST\_MAC2 0x00

#define MY\_DEST\_MAC3 0x00

#define MY\_DEST\_MAC4 0x00

#define MY\_DEST\_MAC5 0x02

#define DEFAULT\_IF "tap0"

#define BUF\_SIZ 1024

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

{

int sockfd;

struct ifreq if\_idx;

struct ifreq if\_mac;

int tx\_len = 0;

char sendbuf[BUF\_SIZ];

struct ether\_header \*eh = (struct ether\_header \*) sendbuf;

struct iphdr \*iph = (struct iphdr \*) (sendbuf + sizeof(struct ether\_header));

struct sockaddr\_ll socket\_address;

char ifName[IFNAMSIZ];

/\* Get interface name \*/

if (argc > 1)

strcpy(ifName, argv[1]);

else

strcpy(ifName, DEFAULT\_IF);

/\* Open RAW socket to send on \*/

if ((sockfd = socket(AF\_PACKET, SOCK\_RAW, IPPROTO\_RAW)) == -1) {

perror("socket");

}

/\* Get the index of the interface to send on \*/

memset(&if\_idx, 0, sizeof(struct ifreq));

strncpy(if\_idx.ifr\_name, ifName, IFNAMSIZ-1);

if (ioctl(sockfd, SIOCGIFINDEX, &if\_idx) < 0)

perror("SIOCGIFINDEX");

/\* Get the MAC address of the interface to send on \*/

memset(&if\_mac, 0, sizeof(struct ifreq));

strncpy(if\_mac.ifr\_name, ifName, IFNAMSIZ-1);

if (ioctl(sockfd, SIOCGIFHWADDR, &if\_mac) < 0)

perror("SIOCGIFHWADDR");

/\* Construct the Ethernet header \*/

memset(sendbuf, 0, BUF\_SIZ);

/\* Ethernet header \*/

eh->ether\_shost[0] = ((uint8\_t \*)&if\_mac.ifr\_hwaddr.sa\_data)[0];

eh->ether\_shost[1] = ((uint8\_t \*)&if\_mac.ifr\_hwaddr.sa\_data)[1];

eh->ether\_shost[2] = ((uint8\_t \*)&if\_mac.ifr\_hwaddr.sa\_data)[2];

eh->ether\_shost[3] = ((uint8\_t \*)&if\_mac.ifr\_hwaddr.sa\_data)[3];

eh->ether\_shost[4] = ((uint8\_t \*)&if\_mac.ifr\_hwaddr.sa\_data)[4];

eh->ether\_shost[5] = ((uint8\_t \*)&if\_mac.ifr\_hwaddr.sa\_data)[5];

eh->ether\_dhost[0] = MY\_DEST\_MAC0;

eh->ether\_dhost[1] = MY\_DEST\_MAC1;

eh->ether\_dhost[2] = MY\_DEST\_MAC2;

eh->ether\_dhost[3] = MY\_DEST\_MAC3;

eh->ether\_dhost[4] = MY\_DEST\_MAC4;

eh->ether\_dhost[5] = MY\_DEST\_MAC5;

/\* Ethertype field \*/

//eh->ether\_type = htons(ETH\_P\_IP);

eh->ether\_type = 0x0400;

tx\_len += sizeof(struct ether\_header);

/\* Packet data \*/

sendbuf[tx\_len++] = 0xde;

sendbuf[tx\_len++] = 0xad;

sendbuf[tx\_len++] = 0xbe;

sendbuf[tx\_len++] = 0xef;

/\* Index of the network device \*/

socket\_address.sll\_ifindex = if\_idx.ifr\_ifindex;

/\* Address length\*/

socket\_address.sll\_halen = ETH\_ALEN;

/\* Destination MAC \*/

socket\_address.sll\_addr[0] = MY\_DEST\_MAC0;

socket\_address.sll\_addr[1] = MY\_DEST\_MAC1;

socket\_address.sll\_addr[2] = MY\_DEST\_MAC2;

socket\_address.sll\_addr[3] = MY\_DEST\_MAC3;

socket\_address.sll\_addr[4] = MY\_DEST\_MAC4;

socket\_address.sll\_addr[5] = MY\_DEST\_MAC5;

/\* Send packet \*/

if (sendto(sockfd, sendbuf, tx\_len, 0, (struct sockaddr\*)&socket\_address, sizeof(struct sockaddr\_ll)) < 0)

printf("Send failed\n");

return 0;

}