```c
#include <arpa/inet.h>
#include<net/if.h>
#include <sys/types.h>          /* See NOTES */
#include <stdio.h>
#include <sys/socket.h>
#include <linux/if_packet.h>
#include <net/ethernet.h> /* the L2 protocols */
#include <string.h>
#include <strings.h>


unsigned char broadcast[6]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};//MAC con tutti 1, pacchetto
indirizzato a tutti
unsigned char mymac[6]={0xf2,0x3c,0x91,0xdb,0xc2,0x98}; //ifconfig
unsigned char myip[4]={88,80,187,84};
unsigned char gateway[4]={88,80,187,1};//comando linux: route -n
unsigned int netmask = 0x00FFFFFF;

// Frame Ethernet
struct eth_frame {
  unsigned char dst[6];
  unsigned char src[6];
  unsigned short type;  //2byte
  char payload[1500]; //ARP o IP
};

//Pacchetto ICMP ECHO
struct icmp_packet{
  unsigned char type;
  unsigned char code;
  unsigned short checksum;
  unsigned short id;
  unsigned short seq;
  char payload[128]; //Dati
};

// Datagramma IP
struct ip_datagram{
  unsigned char ver_ihl;
  unsigned char tos;
  unsigned short totlen;
  unsigned short id;
  unsigned short flags_offs;
  unsigned char ttl;
  unsigned char protocol;
  unsigned short checksum;
  unsigned int src;
  unsigned int dst;
  unsigned char payload[1500];//ICMP o solo dati
};

// Pacchetto ARP
struct arp_packet{
  unsigned short htype;//HardwareType: Ethernet=1
  unsigned short ptype;//ProtocolType IPv4=0x0800
  unsigned char hlen; //LunghezzaIndirizzoHardware = 6 byte
  unsigned char plen; //LunghezzaIndirizzoProtocol = 4 byte
```

```
  unsigned short opcode;//Operazione: 1=Richiesta, 2=Risposta
  unsigned char hsrc[6];//IndirizzoHardware di chi ha fatto la richiesta
  unsigned char psrc[4];//IndirizzoIP di chi ha fatto la richiesta
  unsigned char hdst[6];//IndirizzoHardware del destinatario della richiesta(in una
  richiesta è nullo)
  unsigned char pdst[4];//IndirizzoIP del destinatario della richiesta
};

//Prototipi di funzioni
void crea_eth(struct eth_frame *e, unsigned char * src, unsigned char*dst,unsigned short
type);
void crea_arp(struct arp_packet* a,unsigned char *mymac,unsigned char *myip,unsigned char
*dstip);
void crea_ip( struct ip_datagram* ip, int payloadlen,unsigned char payloadtype,unsigned int
ipdest );
void crea_icmp_echo(struct icmp_packet * icmp);
unsigned short checksum(unsigned char *buffer,int len);
void stampabytes(unsigned char * buffer, int quanti);
int trovamac(unsigned char *mac_incognito, unsigned char*dstip);



int main(){
  struct sockaddr_ll sll; //Indirizzo ethernet mio
  unsigned char mac_incognito[6];
  unsigned char buffer[1500]; //Conterrà i pacchetti di inviare
  unsigned char * nexthop;  //prossimo router
  struct eth_frame * eth; //Pacchetto ethernet che contiene anche ARP
  struct ip_datagram * ip;// Datagramma IP che contiene anche ICMP
  struct icmp_packet * icmp;// pachetto ICMP
  //unsigned char targetip[4]={88,80,187,2};
  unsigned char targetip[4]={147,162,2,100};//Padova
  int m,i,s,len,t;  //Variabili di supporto

//APRIRE COMUNICAZIONE
  /*socket restituisce un INT che è un file descriptor
    ovvero l'indice della tabella con tutto ciò che serve per gestire la comunicazione*/
  s = socket(AF_PACKET, SOCK_RAW,htons(ETH_P_ALL));//AF_INET=IPv4, SOCK_RAW=non elaborato,
  htons(ETH_P_ALL)

  //Se l'host è nella mia stessa sotto rete tengo il suo IP per cercalre l'indirizzo ethernet
  //altrimenti uso l'indirizzo del mio gateway.
  if ((*((unsigned int *)targetip)&netmask) == (*((unsigned int*)myip)&netmask))
    nexthop = targetip;
  else nexthop = gateway;

  //Trovo l'indirizzo mac di destinazione avendo l'IP
  if(!trovamac(mac_incognito,nexthop)){
    printf("MAC incognito:");
    stampabytes(mac_incognito,6);
  }
  else printf("trovamac fallita\n");

  //Costruisco un frame ethernet con pacchetto IP-ICMP all'interno
  eth = (struct eth_frame *) buffer;
  ip = (struct ip_datagram *) eth->payload;
  icmp = (struct icmp_packet*) ip->payload;
```

```
  crea_eth(eth,mymac,mac_incognito,0x0800);
  crea_ip(ip,28,1,*(unsigned int*)targetip);
  crea_icmp_echo(icmp);

  //Visualizzo a schermo il pacchetto creato
  printf("\nPacchetto ETH+IP+ICMP da inviare:");
  stampabytes(buffer,14+20+28);//+14 header ethernet, +20 IP +28 ICMP(8+20 di payload)

  //Preparo l'indirizzo
  bzero(&sll,sizeof(struct sockaddr_ll));//Metto tutto a 0
  sll.sll_ifindex = if_nametoindex("eth0");
  printf("ifindex = %d\n",sll.sll_ifindex);
  len = sizeof(sll);

  //Invio il pacchetto ethernet che contiene IP
  t = sendto(s, buffer, 14+20+28, 0, (struct sockaddr *) &sll, len);
  if (t == -1 ){
    perror("sendto fallita");
    return 1;
  }

  //Ricevo la risposta
  for(m=0;m<1000;m++){
    t = recvfrom(s, buffer, 1500, 0, (struct sockaddr *) &sll, &len);
    if (t == -1 ){
      perror("recvfrom fallita");
      return 1;
    }

    //Se il frame Ethernet ricevuto contiene un pacchetto IP
    if (eth->type == htons(0x0800)){

      //Se il protocollo all'interno del payload IP è ICMP=1
      if((ip->protocol == 1 )){

        //Se il tipo di ICMP è 0: Echo Reply
        if((icmp->type == 0)){
          printf("\nPacchetto ICMP ECHO REPLY ricevuto:");
          stampabytes(buffer,t);
          return 0;
        }

      }

    }

  }//Fine for

}//Fine main
```

```c
//Invia un Pacchetto Ethernet con dentro il pacchetto ARP e ricevo in risposta il MAC del destinatario
//Ritorna 0 se tutto ok altrimenti 1.
int trovamac(unsigned char *mac_incognito, unsigned char*dstip){
  struct sockaddr_ll sll;//Struttura indirizzo per il sendto
  int len,i,t,m,s;
  char ifname[50];
  unsigned char buffer[1500];
  struct arp_packet * arp;
  struct eth_frame * eth;
  eth = (struct eth_frame *) buffer;
  arp = (struct arp_packet *) eth->payload;
  crea_eth(eth,mymac,broadcast,0x0806);
  crea_arp(arp,mymac,myip,dstip);

  //Visualizzo a schermo il pacchetto creato
  printf("Pacchetto ETH+ARP creato:");
  stampabytes(buffer,14+sizeof(struct arp_packet));//+14 è il frame Ethernet

//APRIRE COMUNICAZIONE
  /*socket restituisce un INT che è un file descriptor
    ovvero l'indice della tabella con tutto ciò che serve per gestire la comunicazione*/
  s = socket(AF_PACKET, SOCK_RAW,htons(ETH_P_ALL));

  //Prepara l'indirizzo
  bzero(&sll,sizeof(struct sockaddr_ll));//Mette tutti i byte a 0
  sll.sll_family = AF_PACKET;//Livello ethernet8(NO IP)
  //for(i=1;i<4;i++)
  //      printf("%s\n",if_indextoname(i,ifname));

  printf("\n");
  sll.sll_ifindex = if_nametoindex("eth0");
  len = sizeof(sll);//Grandezza della struttura indirizzo

  //Invia il pacchetto ethernet
  t = sendto(s, buffer, 14+sizeof(struct arp_packet), 0, (struct sockaddr *) &sll, len);
  if (t == -1 ){
    perror("sendto fallita");
    return 1;
  }

  //Ricevo il pacchetto ethernet di risposta che contiene un ARP
  for(m=0;m<1000;m++){
    t = recvfrom(s, buffer, 1500, 0, (struct sockaddr *) &sll, &len);
    if (t == -1 ){
      perror("recvfrom fallita");
      return 1;
    }

    //Se il frame Ethernet ricevuto contiene un ARP
    if (eth->type == htons(0x0806)){
      if(arp->opcode == htons(2))//Se l'ARP è di risposta=2

        //Se i primi 4 byte di dell'IP di sorgente ricevuto sono quelli di destinazione precedenti
        if(!memcmp(arp->psrc,dstip,4)){
          memcpy(mac_incognito,arp->hsrc,6);
```

```
          return 0;
        }
      }
  }

  return 1;
}

//Crea pacchetto ethernet
void crea_eth(struct eth_frame * e, unsigned char * src, unsigned char*dst,unsigned short
type){
  int i;
  e->type = htons(type);
  for(i=0;i<6;i++) e->src[i]=src[i];
  for(i=0;i<6;i++) e->dst[i]=dst[i];
}

//Crea pacchetto ARP
void crea_arp(struct arp_packet* a,unsigned char *mymac,unsigned char *myip,unsigned char
*dstip){
  int i;
  a-> htype = htons(1);//Ethernet=1
  a-> ptype = htons(0x0800);//IPv4=0x0800
  a-> hlen = 6;//Lunghezza indirizzo hardware(Ethernet)=6byte
  a-> plen = 4;//Lunghezza indirizzo di rete (IP)=4byte
  a-> opcode =htons(1);//Richiesta=1, risposta=2
  for(i=0;i<6;i++) a-> hsrc[i]=mymac[i];
  for(i=0;i<4;i++) a-> psrc[i]=myip[i];
  for(i=0;i<6;i++) a-> hdst[i]=0;
  for(i=0;i<4;i++) a-> pdst[i]=dstip[i];
}

//Stampa un numero di byte da un buffer
void stampabytes(unsigned char * buffer, int quanti){
  int i;
  for(i=0;i<quanti;i++){
    if (!(i&0x3)) printf("\n");
    printf("%.2X(%.3d) ",buffer[i],buffer[i]);
  }
  printf("\n");
}

//Crea pacchetto IP
void crea_ip( struct ip_datagram* ip, int payloadlen,unsigned char payloadtype,unsigned int
ipdest ){
  ip->ver_ihl = 0x45; //primi 4 bit: versione=4, ultimi 4 bit:IHL=5word(32bit*5=20byte)
  ip->tos= 0;
  ip->totlen=htons(20+payloadlen);
  ip->id=htons(0xABCD);
  ip->flags_offs=htons(0);
  ip->ttl=128;  //Può attraversare massimo 128 router
  ip->protocol=payloadtype;
  ip->src=*((unsigned int*)myip);
  ip->dst=ipdest;
  ip->checksum=htons(0);
  ip->checksum=htons(checksum((unsigned char*) ip, 20));
};
```

```c
//header Checksum del pacchetto IP
unsigned short checksum( unsigned char * buffer, int len){
  int i;
  unsigned short *p;
  unsigned int tot=0;
  p = (unsigned short *) buffer;
  for(i=0;i<len/2;i++){
    tot = tot + htons(p[i]);
    if (tot&0x10000) tot = (tot&0xFFFF)+1;
  }
  return (unsigned short)0xFFFF-tot;
}

//Crea pachetto ICMP ECHO
void crea_icmp_echo(struct icmp_packet * icmp){
  int i;
  icmp->type=8; //Echo (request)
  icmp->code=0;
  icmp->checksum=htons(0);
  icmp->id=htons(0x1234);
  icmp->seq = htons(1);
  for(i=0;i<20;i++) icmp->payload[i]=i;//numeri da 0a 19
  icmp->checksum = htons(checksum((unsigned char*)icmp,28));
}
```