```c
#include <stdio.h>
#include <strings.h>

#include <sys/types.h>          /* See NOTES */
#include <sys/time.h>
#include <sys/socket.h>

#include <linux/if_packet.h>
#include <net/ethernet.h> /* the L2 protocols */

#include <arpa/inet.h>


struct timeval tcptime, timezero; //???
struct sockaddr_ll sll; //Indirizzo ethernet mio
unsigned int delta_sec, delta_usec; //???
int primo;  //???

unsigned char dstip[4]={88 ,80 ,187 ,80 };//IP destinazione
unsigned char miomac[6]={0xf2,0x3c,0x91,0xdb,0xc2,0x98};//ifconfig
unsigned char mioip[4]={88 ,80 ,187 ,84 };
unsigned char gateway[4]={88 ,80 ,187 ,1};//comando linux: route -n
unsigned char netmask[4]={255 ,255 ,255 ,0 };
unsigned char broadcastmac[6]={0xff,0xff,0xff,0xff,0xff,0xff};//MAC con tutti 1, pacchetto
indirizzato a tutti

int progr=0;//???

//Segmento TCP
struct tcp_segment{
  unsigned short s_port;
  unsigned short d_port;
  unsigned int seq;
  unsigned int ack;
  unsigned char d_offs_res;
  unsigned char flags;
  unsigned short win;
  unsigned short checksum;
  unsigned short urgp;
  unsigned char payload[1000];
};

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

// Datagramma IP
struct ip_datagram{
  unsigned char ver_ihl;
  unsigned char tos;
  unsigned short totlen;
  unsigned short id;
  unsigned short flag_offs;
```

```
  unsigned char ttl;
  unsigned char proto;
  unsigned short checksum;
  unsigned int saddr;
  unsigned int daddr;
  unsigned char payload[1];//???
};

//Funzione checksum per pacchetto IP
unsigned short int checksum(char *b, int l){
  unsigned short *p ;

  int i;
  unsigned short int tot=0;
  unsigned short int prec;
  /*Assunzione: l pari */
  p=(unsigned short *)b;

  for(i=0; i < l/2 ;i++){
    prec = tot;
    tot += htons(p[i]);
    if (tot<prec) tot=tot+1;
  }
  return (0xFFFF-tot);
}

//Port 0-65536
struct tcp_status{
  unsigned int forwzero, backzero;
  struct timeval timezero;
}st[65536];

//Crea pacchetto IP
void creaip(struct ip_datagram *ip, unsigned int dstip,int payload_len, unsigned char proto){
  ip->ver_ihl= 0x45 ;//primi 4 bit: versione=4, ultimi 4 bit:IHL=5word(32bit*5=20byte)
  ip->tos = 0x0;
  ip->totlen = htons(payload_len+20);
  ip->id=progr++;//???
  ip->flag_offs=htons(0);
  ip->ttl=64; //Può attraversare massimo 64 router
  ip->proto=proto;
  ip->checksum=0;
  ip->saddr=*(unsigned int *)mioip;
  ip->daddr=dstip;
  ip->checksum = htons(checksum((unsigned char*)ip,20));
};
```

```
// MAIN
int main(int argc, char ** argv){
  unsigned int ackzero, seqzero, forwzero,backzero;//Contatori per
  unsigned char buffer[1600];
  unsigned short other_port;// Variabile di supporto per salvare la porta di sorgente/dest
  int i,l,s,lunghezza;//???

  struct eth_frame * eth; //Frame ethernet
  struct ip_datagram *ip;// Datagramma IP che contiene anche TCP
  struct tcp_segment *tcp;//Segmento TCP

  //Costruisco un frame ethernet con pacchetto IP-TCP all'interno
  eth = (struct eth_frame *) buffer;
  ip = (struct ip_datagram*) eth->payload;
  tcp = (struct tcp_segment*) ip->payload;

//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)

  //Preparo l'indirizzo
  lunghezza = sizeof(struct sockaddr_ll);
  bzero(&sll,sizeof(struct sockaddr_ll));
  sll.sll_ifindex=3;


  primo=1;
  while(1){
    //Ricevo
    l=recvfrom(s,buffer,1600,0,(struct sockaddr *) &sll, &lunghezza);

    //Inserisco in tcptime quando ho ricevuto il pacchetto
    gettimeofday(&tcptime,NULL);

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

      //Se il protocollo all'interno del payload IP è TCP=6
      if(ip->proto==6){
              // if (tcp->d_port == htons(80) && primo){
       //             other_port=htons(tcp->s_port);
      //              timezero=tcptime;
                    // primo=0;
                    // }

        //Se hanno come sorgente o destinazione il port80
        if((tcp->s_port == htons(80)) || (tcp->d_port == htons(80))){

          //Se la portSorgente=80 in other_port inserisco la portDestinazione sennò il
          contrario
          other_port = (tcp->s_port==htons(80))? htons(tcp->d_port):htons(tcp->s_port);

          //Se il SYN è a 1
          if(tcp->flags&0x02){
            st[other_port].timezero=tcptime;//Salvo quando ho visto pasare il primo SYN
```

```
            //Salvo il contatore dei byte del segmento in base alla direzione (entrata/uscita)
            if(tcp->d_port==htons(80)) st[other_port].forwzero=htonl(tcp->seq);
            if(tcp->s_port==htons(80)) st[other_port].backzero=htonl(tcp->seq);
          }

          //Se la portDestinazine=80 in seqzero inserisco forwzero dell'altra porta
          altrimenti backzero
          seqzero = (tcp->d_port==htons(80))?st[other_port].forwzero:st[other_port].backzero;
          //Se la portSorgente=80 in ackzero inserisco forwzero dell'altra porta altrimenti
          backzero
          ackzero = (tcp->s_port==htons(80))?st[other_port].forwzero:st[other_port].backzero;

          //Vedo la differenza in secondi tra il tempo 0(primo pacchetto) e tempo attuale
          delta_sec = tcptime.tv_sec - st[other_port].timezero.tv_sec;

          //Differenza microsecondi
          if (tcptime.tv_usec < st[other_port].timezero.tv_usec) {
            delta_sec --;
            delta_usec = 1000000+tcptime.tv_usec - st[other_port].timezero.tv_usec;
          }
          else delta_usec = tcptime.tv_usec - st[other_port].timezero.tv_usec;

          //Per ogni pacchetto stampo una riga che dice i dati, stma della banda,...
          printf("%.4d.%.6d %.5d->%.5d %.2x %.10u %.10u %.5u %4.2f\n",delta_sec, delta_usec,
          htons(tcp->s_port),htons(tcp->d_port),tcp->flags, htonl(tcp->seq)-seqzero,
          htonl(tcp->ack)-ackzero,
          htons(tcp->win),(htonl(tcp->ack)-ackzero)/(double)(delta_sec*1000000+delta_usec));
        }

      }

  }//Fine while

}//Fine main
```