

Università di Catania

Laboratorio di Reti di Calcolatori

Reti di Calcolatori 1

1

Università di Catania

Socket

Un socket indica un'astrazione software progettata per utilizzare delle API standard e condivise per la trasmissione e la ricezione di dati attraverso una rete oppure come meccanismo di IPC (inter-Process Communication).

Reti di Calcolatori 2

2

Università di Catania

Socket di Berkeley

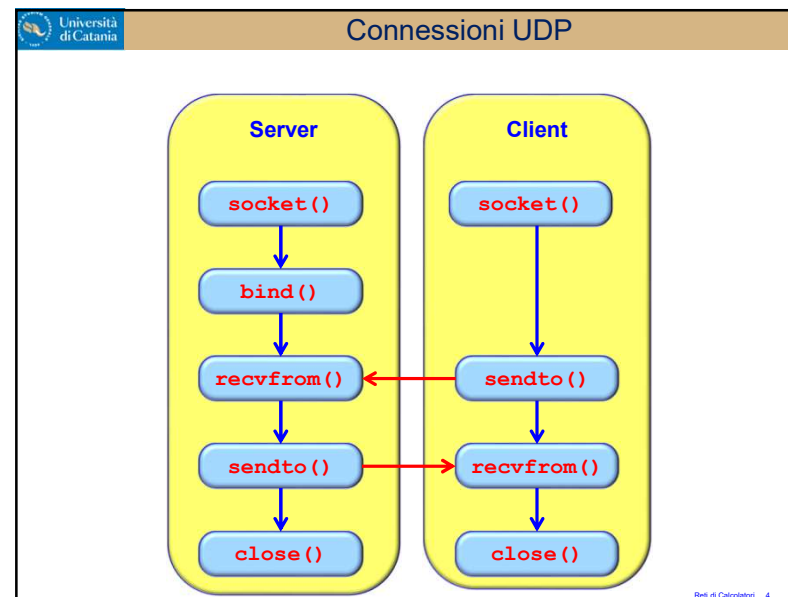
I socket sono definiti in modo univoco da un insieme di 5 elementi:

- il protocollo utilizzato
- l'indirizzo IP locale
- la porta locale
- l'indirizzo IP remoto
- la porta remota

Basta che uno solo di questi elementi sia differente per rendere diverso il collegamento.

Reti di Calcolatori 3

3



4

Definizione di un socket

```
struct sockaddr
{ u_short sa_family;
  char    sa_data[14];
}
```

Gestisce varie famiglie di socket (anche IPv4 e IPv6).

La famiglia `sa_family` può essere di vario tipo:

`AF_UNIX, AF_NS, AF_IMPLINK, AF_INET, AF_INET6,
AF_IPX, AF_IRDA, AF_BLUETOOTH`

Linux supporta 29 tipi differenti di famiglie.

Definizione di un socket AF_INET

Tipo `AF_INET`

```
struct sockaddr_in
{ short int      sin_family; //AF_INET
  unsigned short int sin_port;
  struct in_addr sin_addr;
  unsigned char   sin_zero[8];
};

struct in_addr
{ u_int32_t s_addr;
};
```

Definizione di un socket AF_INET6

Tipo `AF_INET6`

```
struct sockaddr_in6
{ u_int16_t sin6_family; //AF_INET6
  u_int16_t sin6_port;
  u_int32_t sin6_flowinfo; //IPv6 flow info
  struct in6_addr sin6_addr;
  u_int32_t sin6_scope_id; //scope ID
};

struct in6_addr
{ unsigned char s6_addr[16];
};
```

Conversione degli indirizzi

`pton` => Presentation to Network
`ntop` => Network to Presentation

Queste funzioni consentono di trasformare gli indirizzi IPv4 o IPv6 dal formato compatto (interno) a quello testo leggibile, e viceversa.

```
const char *inet_ntop(int af, const void *src ,
                      char *dst, socklen_t size);

int inet_ntop(int af, const char *src , void *dst);
```

Università di Catania

Conversione degli indirizzi

Esempi

```
struct sockaddr_in sa; // IPv4
char ip4[16];

inet_ntop(AF_INET, &(sa.sin_addr), ip4, INET_ADDRSTRLEN);

inet_pton(AF_INET, "192.0.2.1", &(sa.sin_addr));

// INET_ADDRSTRLEN = 16
// 255.255.255.255
```

Reti di Calcolatori 9

9

Università di Catania

Conversione degli indirizzi

Esempi

```
struct sockaddr_in6 sa6; // IPv6
char ip6[46];

inet_ntop(AF_INET6, &(sa6.sin6_addr), ip6, INET6_ADDRSTRLEN);

inet_pton(AF_INET6, "2001:d8:b3:1::3490", &(sa6.sin6_addr));

// INET6_ADDRSTRLEN = 46
// FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:255.255.255.255
```

Reti di Calcolatori 10

10

Università di Catania

Conversione degli indirizzi

```
int inet_aton(const char *name, struct in_addr *addr)
```

converte un indirizzo IP in formato "xx.xx.xx.xx" nel corrispondente numero esadecimale e lo inserisce nella struttura *addr*.

```
unsigned long inet_addr(const char *name)
```

converte un indirizzo IP in un *unsigned long*

```
char *inet_ntoa(struct in_addr addr)
```

fa l'operazione inversa di *inet_aton()*

Sono funzioni deprecate

Reti di Calcolatori 11

11

Università di Catania

Conversione degli indirizzi


```
unsigned short htons(unsigned short int n)
unsigned short ntohs(unsigned short int n)
unsigned long htonl(unsigned long int n)
unsigned long ntohl(unsigned long int n)
```

The **htons()** function converts the unsigned short integer *hostshort* from host byte order to network byte order.
 The **ntohs()** function converts the unsigned short integer *netshort* from network byte order to host byte order.
 The **htonl()** function converts the unsigned integer *hostlong* from host byte order to network byte order.
 The **ntohl()** function converts the unsigned integer *netlong* from network byte order to host byte order.

On the i386 the host byte order is Least Significant Byte first, whereas the network byte order, as used on the Internet, is Most Significant Byte first.

Reti di Calcolatori 12

12

Università di Catania

Socket

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```


La chiamata `socket()` restituisce un identificatore di *socket*.

Il *socket* viene creato e viene definito sia il suo tipo che il protocollo utilizzato.

L'utilizzo dei descrittori di socket è simile a quello dei file.

Reti di Calcolatori 13

13

Università di Catania

Socket

Domain

`PF_INET PF_INET6`

Type (solo per `PF_INET`):

`SOCK_DGRAM, SOCK_STREAM`

Il protocollo dipende dalla famiglia (di seguito per `AF_INET`):

`IPPROTO_UDP, IPPROTO_TCP, IPPROTO_ICMP, IPPROTO_RAW`

Reti di Calcolatori 14

14

Università di Catania

Socket

Domain:


spesso, al posto di `PF_INET` si usa `AF_INET` (hanno lo stesso valore)

Protocol:

Si può usare `0`, (verrà scelto il protocollo più adatto al tipo indicato)

Reti di Calcolatori 15

15

Università di Catania

Bind

```
int bind(int socket,
         struct sockaddr *addr,
         int addrlen);
```

La chiamata `bind()` restituisce zero in caso di successo. La *bind* serve per inserire i dati locali (indirizzo e porta) nel *socket*.

L'effetto della `bind` è duplice: per il traffico in *input* serve per dire al sistema a chi deve consegnare pacchetti entranti, mentre per il traffico in *output* serve per inserire il mittente nell'intestazione dei pacchetti.

Reti di Calcolatori 16

16

Università di Catania

Bind

Nella definizione dell'indirizzo Internet, è possibile usare alcuni valori definiti tramite macro:

- INADDR_LOOPBACK**
indica l'host stesso (localhost 127.0.0.1).
- INADDR_ANY**
serve per accettare le connessioni da qualunque indirizzo
- INADDR_BROADCAST**
serve per mandare messaggi in broadcast
- INADDR_NONE**
viene restituito da alcune funzioni in caso di errore.

Reti di Calcolatori 17

17

Università di Catania

sendto - recvfrom

```
int sendto(int socket, void *buffer, size_t size
           int flags, struct sockaddr *addr, size_t length);

int recvfrom(int socket, void *buffer, size_t size
             int flags, struct sockaddr *addr, size_t *length);
```

Queste due funzioni operano come la `send()` e la `recv()` ma sono utilizzate per le trasmissioni senza connessione. Nella `sendto()` deve essere specificato l'indirizzo di destinazione, mentre nella `recvfrom()` il campo indirizzo viene riempito con quello del mittente.

```
int close(int socket);
```

Chiude un socket aperto.

Reti di Calcolatori 18

18

Università di Catania

Esempio UDP – sender to receiver

Il seguente programma è il più semplice immaginabile. È composto da un `sender` che spedisce pacchetti e da un `receiver` che li riceve. La comunicazione è unidirezionale. Il protocollo usato è UDP.

Reti di Calcolatori 19

19

Università di Catania

Connessioni UDP

```

graph TD
    subgraph Sender
        S_socket(socket()) --> S_sendto(sendto())
        S_sendto --> S_close(close())
    end
    subgraph Receiver
        R_socket(socket()) --> R_bind(bind())
        R_bind --> R_recvfrom(recvfrom())
        R_recvfrom --> R_close(close())
    end
    S_sendto -- red arrow --> R_recvfrom
  
```

Reti di Calcolatori 20

20

Università

di Catania

Esempio UDP – Sender

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>

int main(int argc, char* argv[])
{
    int sockfd;
    struct sockaddr_in dest_addr;
    extern int errno;
    int i,n;
    char buffer[100];

    if(argc < 3)
    {
        printf("Use: sender IP_dest PORT_dest");
        return 0;
    }
    if((sockfd=socket(PF_INET,SOCK_DGRAM,0)) <0)
    {
        printf("\nError in socket opening ... exit!");
        return -1;
    }
    memset(&dest_addr, 0, sizeof(dest_addr));
    dest_addr.sin_family = AF_INET;
    inet_pton(AF_INET, argv[1], &(dest_addr.sin_addr));
    dest_addr.sin_port = htons(atoi(argv[2]));

    printf("\nInsert an integer: ");
    scanf("%d",&n);

    for(i=0; i<10; ++i)
    {
        sprintf(buffer,"%d",i+n);
        printf("sending %s\n",buffer);
        sendto(sockfd, buffer, strlen(buffer)+1, 0, (struct sockaddr *) &dest_addr, sizeof(dest_addr));
    }
}

```

21

Università

di Catania

Esempio UDP – Receiver

```

* UDP Receiver - Server*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <errno.h>

int main(int argc, char* argv[])
{
    int sockfd;
    struct sockaddr_in local_addr, remote_addr;
    socklen_t len = sizeof(struct sockaddr_in);
    char buffer[256];
    if(argc < 2)
    {
        printf("Use: receiver listening_PORT");
        return 0;
    }
    if((sockfd=socket(PF_INET,SOCK_DGRAM,0)) <0)
    {
        printf("\nError in socket opening ... exit!");
        return -1;
    }
    memset((char *) &local_addr,0,len);
    local_addr.sin_family = AF_INET;
    //local_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    local_addr.sin_port = htons(atoi(argv[1]));

    if(bind(sockfd, (struct sockaddr *) &local_addr, sizeof(local_addr))<0)
    {
        printf("\nBinding error! Error code n.%d \n",errno);
        return -1;
    }
    for(;;)
    {
        recvfrom(sockfd, buffer, 99,0, (struct sockaddr *) &remote_addr, &len);
        printf("Packet from IP:%s Port:%d msg:%s \n", inet_ntoa(remote_addr.sin_addr),
        ntohs(remote_addr.sin_port), buffer);
    }
}

```

22

Università

di Catania

Esempio UDP bidirezionale

Il seguente esempio mostra invece una connessione UDP *bidirezionale*. Un modulo (**transponder**) rimane residente su un host e restituisce al mittente tutto ciò che arriva alla sua porta d'ascolto.


23

Università

di Catania

Connessioni UDP

24



Università

di Catania

Esempio UDP bidirezionale

```

/* Simple UDP server with bidirectional connection */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <errno.h>

int main(int argc, char**argv)
{
    int sockfd, n;
    extern int errno;
    struct sockaddr_in local_addr, remote_addr;
    socklen_t len = sizeof(struct sockaddr_in);
    char msg[1000];

    if (argc < 2)
    {
        printf("Use: server listening_PORT");
        return 0;
    }

    /* Simple UDP server with bidirectional connection */
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <arpa/inet.h>
    #include <errno.h>

    int main(int argc, char**argv)
    {
        int sockfd, n;
        extern int errno;
        struct sockaddr_in local_addr, remote_addr;
        socklen_t len = sizeof(struct sockaddr_in);
        char msg[1000];

        if (argc < 2)
        {
            printf("Use: server listening_PORT");
            return 0;
        }

        if ((sockfd=socket(PF_INET, SOCK_DGRAM, 0)) < 0)
        {
            printf("Error in socket opening ... exit!");
            return -1;
        }

        memset(&local_addr, 0, sizeof(local_addr));
        local_addr.sin_family = AF_INET;
        local_addr.sin_port = htons(atoi(argv[1]));
        if (bind(sockfd, (struct sockaddr *)&local_addr, len) < 0)
        {
            printf("Binding error! Error code %d\n", errno);
            return -1;
        }


        for (;;)
        {
            n = recvfrom(sockfd, msg, 999, 0, (struct sockaddr *)&remote_addr, &len);
            msg[n] = 0;
            printf("From IP: %s Port: %d msg: %s\n",
                inet_ntoa(remote_addr.sin_addr),
                ntohs(remote_addr.sin_port), msg);
            sendto(sockfd, msg, n, 0, (struct sockaddr *)&remote_addr, len);
        }
    }

```

Ret di Calcolatori

25

25



Università

di Catania

Esempio UDP bidirezionale

```

if ((sockfd=socket(PF_INET, SOCK_DGRAM, 0)) < 0)
{
    printf("\nError in socket opening ... exit!");
    return -1;
}

memset(&local_addr, 0, sizeof(local_addr));
local_addr.sin_family = AF_INET;
local_addr.sin_port = htons(atoi(argv[1]));
if (bind(sockfd, (struct sockaddr *)&local_addr, len) < 0)
{
    printf("\nBinding error! Error code %d\n", errno);
    return -1;
}


for (;;)
{
    n = recvfrom(sockfd, msg, 999, 0, (struct sockaddr *)&remote_addr, &len);
    msg[n] = 0;
    printf("From IP: %s Port: %d msg: %s\n",
        inet_ntoa(remote_addr.sin_addr),
        ntohs(remote_addr.sin_port), msg);
    sendto(sockfd, msg, n, 0, (struct sockaddr *)&remote_addr, len);
}

```

Ret di Calcolatori

26

26



Università

di Catania

Esempio UDP bidirezionale

```

/* Simple UDP client */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>

int main(int argc, char**argv)
{
    int sockfd, n;
    struct sockaddr_in remote_addr;
    char sendline[1000];
    char recvline[1000];
    socklen_t len = sizeof(struct sockaddr_in);

    if (argc != 3)
    {
        printf("usage: UDPclient <remote_IP remote_port>\n");
        return 1;
    }

    if ((sockfd=socket(AF_INET, SOCK_DGRAM, 0)) < 0)
    {
        printf("Error in socket opening ... exit!");
        return -1;
    }

    memset(&remote_addr, 0, len);
    remote_addr.sin_family = AF_INET;
    inet_pton(AF_INET, argv[1], &remote_addr.sin_addr);
    remote_addr.sin_port = htons(atoi(argv[2]));


    while (fgets(sendline, 1000, stdin) != NULL)
    {
        sendto(sockfd, sendline, strlen(sendline), 0,
            (struct sockaddr *)&remote_addr, len);
        n = recvfrom(sockfd, recvline, 999, 0, (struct sockaddr *)&remote_addr, &len);
        recvline[n] = 0;
        printf("From IP: %s Port: %d msg: %s\n",
            inet_ntoa(remote_addr.sin_addr),
            ntohs(remote_addr.sin_port), recvline);
    }
}

```

Ret di Calcolatori

27

27



Università

di Catania

Esempio UDP bidirezionale

```

if ((sockfd=socket(AF_INET, SOCK_DGRAM, 0)) < 0)
{
    printf("\nError in socket opening ... exit!");
    return -1;
}

memset(&remote_addr, 0, len);
remote_addr.sin_family = AF_INET;
inet_pton(AF_INET, argv[1], &remote_addr.sin_addr);
remote_addr.sin_port = htons(atoi(argv[2]));

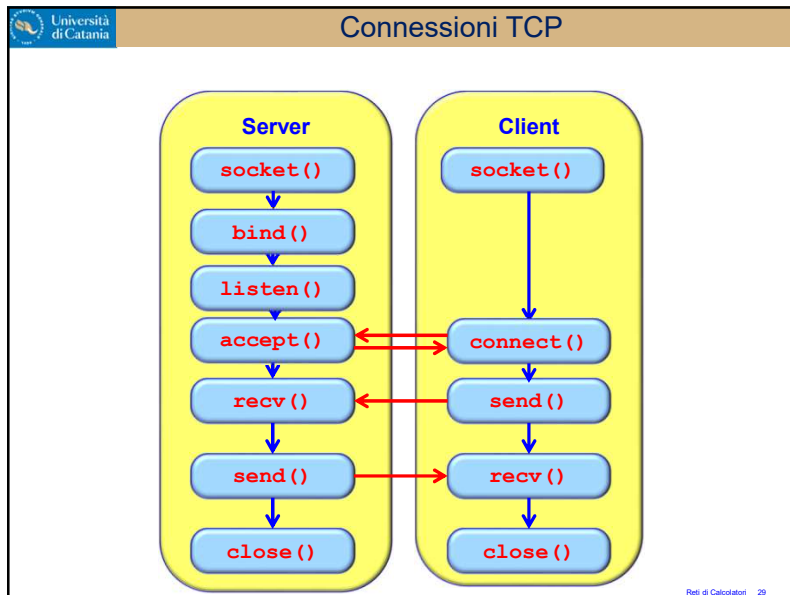
while (fgets(sendline, 1000, stdin) != NULL)
{
    sendto(sockfd, sendline, strlen(sendline), 0,
        (struct sockaddr *)&remote_addr, len);
    n = recvfrom(sockfd, recvline, 999, 0, (struct sockaddr *)&remote_addr, &len);
    recvline[n] = 0;
    printf("From IP: %s Port: %d msg: %s\n",
        inet_ntoa(remote_addr.sin_addr),
        ntohs(remote_addr.sin_port),
        recvline);
}

```

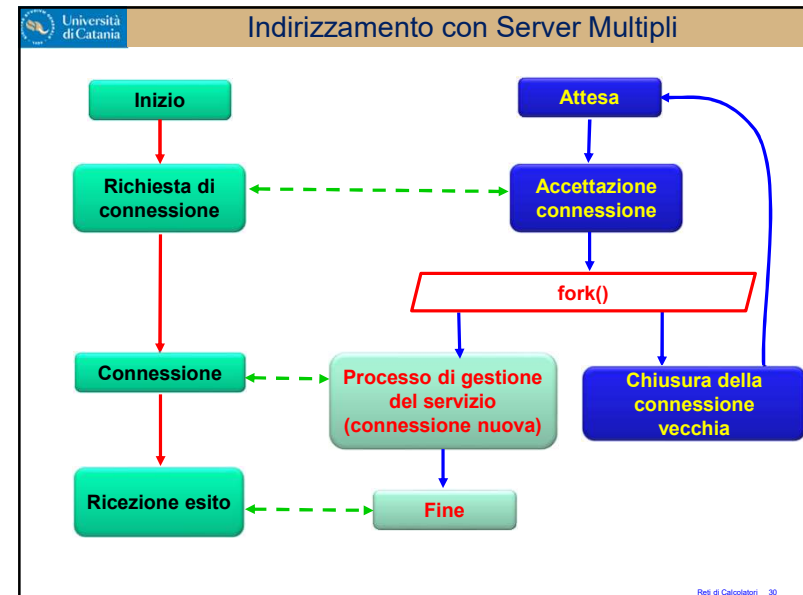
Ret di Calcolatori

28

28



29



30

Università di Catania

Esempio fork()

```

/* fork */
#include <stdio.h>
#include <unistd.h>

int main(int argc, char**argv)
{ int n=0, pid;

  for(; n<5; ++n) printf("%d \n",n);

  pid = fork();

  if (pid == 0)
  { for(; n<10; ++n) printf("%d %d \n", pid, n*2);
    return 0;
  }
  else
  { for(; n<10; ++n) printf("%d %d \n", pid, n*3);
    return 0;
  }
}
  
```

Reti di Calcolatori 31

31 La funzione fork viene utilizzata per creare un nuovo processo che rappresenta la duplicazione del processo chiamante

Università di Catania

Lato server – listen() – accept()

```

int listen(int socket, int backlog);
  
```

La chiamata `listen()` abilita il socket a ricevere connessioni, rendendolo quindi un server socket. Il parametro `n` indica quante richieste in sospeso devono essere accodate.

```

int accept(int socket, struct sockaddr *addr, socklen_t *addrlen);
  
```

La chiamata `accept()` è bloccante. Non appena arriva una richiesta di connessione, crea un nuovo socket e ne restituisce il descrittore. Il vecchio socket rimane aperto e non connesso. L'indirizzo restituito è quello di chi ha effettuato la `connect()`

Reti di Calcolatori 32

32


```
int connect(int socket, struct sockaddr *addr,
            int addrlen);
```

La chiamata `connect()` inizializza una connessione con un socket remoto. L'indirizzo che viene passato è relativo ad un host remoto.

La `connect()` è bloccante, finché non vengono negoziati i parametri della trasmissione (il protocollo è il TCP).
La funzione viene richiamata da un client che vuol connettersi ad un server (il cui socket deve già essere aperto).

```
int send(int socket, void *buffer, size_t size,
         int flags);

int recv(int socket, void *buffer, size_t size,
         int flags);
```

Queste due funzioni operano come la `write()` e la `read()` per i file normali. Sono utilizzate per trasmissioni con connessione.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <errno.h>

int main(int argc, char**argv)
{ int sockfd,newsockfd,n;
  struct sockaddr_in local_addr,remote_addr;
  socklen_t len;
  char msg[1000];

  if(argc < 2)
  { printf("Use: server listeing_PORT");
    return 0;
  }
```

```
if((sockfd=socket(AF_INET,SOCK_STREAM,0)) <0)
{ printf("\nErrore nell'apertura del socket");
  return -1;
}

memset((char *) &local_addr,0,sizeof(local_addr));
local_addr.sin_family = AF_INET;
local_addr.sin_addr.s_addr = htonl(INADDR_ANY);
local_addr.sin_port = htons(atoi(argv[1]));

if(bind(sockfd, (struct sockaddr *) &local_addr,
sizeof(local_addr))<0)
{ printf("\nErrore nel binding. Errore %d \n", errno);
  return -1;
}

listen(sockfd,5);
```

Università di Catania

Client Server TCP - Modulo server

```

for(;;)
{ len = sizeof(remote_addr);
  newsockfd = accept(sockfd, (struct sockaddr *) &remote_addr, &len);

  if (fork() == 0)
  { close(sockfd);
    for(;;)
    { n = recv(newsockfd, msg, 999, 0);
      if(n==0) return 0;
      msg[n] = 0;
      printf("\nPid=%d: received from %s:%d the following: %s\n",
getpid(), inet_ntoa(remote_addr.sin_addr),
ntohs(remote_addr.sin_port), msg );
      send(newsockfd, msg, n, 0);
    }
    return 0;
  }
  else
    close(newsockfd);
}
}

```

Reti di Calcolatori 37

37

Università di Catania

Client Server TCP - Modulo client

```

/* Simple TCP client */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <errno.h>

int main(int argc, char**argv)
{ int sockfd, n;
  struct sockaddr_in local_addr, dest_addr;
  char sendline[1000];
  char recvline[1000];

  if (argc != 3)
  { printf("usage: client IP_address <Port\n");
    return 1;
  }
}

```

Reti di Calcolatori 38

38

Università di Catania

Client Server TCP - Modulo client

```

sockfd=socket(AF_INET, SOCK_STREAM, 0);

memset(&dest_addr, 0, sizeof(dest_addr));
dest_addr.sin_family = AF_INET;
dest_addr.sin_addr.s_addr = inet_addr(argv[1]);
dest_addr.sin_port = htons(atoi(argv[2]));

connect(sockfd, (struct sockaddr *) &dest_addr,
sizeof(dest_addr));

while (fgets(sendline, 999, stdin) != NULL)
{ send(sockfd, sendline, strlen(sendline), 0);
  n=recv(sockfd, recvline, 999, 0);
  recvline[n]=0;
  printf("\nPid=%d: received from %s:%d the following: %s\n",
getpid(), inet_ntoa(dest_addr.sin_addr),
ntohs(dest_addr.sin_port), recvline );
}
}

```

Reti di Calcolatori 39

39

Università di Catania

RAW socket

I socket di tipo **SOCK_DGRAM**, **SOCK_STREAM** eliminano le intestazioni, passando soli i dati di livello di trasporto.

I socket di tipo **SOCK_RAW** permettono invece di accedere direttamente alle frame ethernet ricevute.

```

int sockfd;
sockfd = socket(AF_PACKET, SOCK_RAW, ETH_P_ALL);
// all frame

int sockfd;
sockfd = socket(AF_PACKET, SOCK_RAW, ETH_P_IP);
// only IP packets

```

Reti di Calcolatori 40

40

Struttura dell'intestazione ethernet

```
struct ethhdr {
    unsigned char h_dest[ETH_ALEN]; /* destination eth addr */
    unsigned char h_source[ETH_ALEN]; /* source ether addr */
    __be16 h_proto; /* packet type ID field */
} __attribute__((packed));
```

41

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
    __u8  ihl:4,
        version:4;
#elif defined(__BIG_ENDIAN_BITFIELD)
    __u8  version:4,
        ihl:4;
#else
#error "Please fix <asm/byteorder.h>"
#endif
    __u8  tos;
    __u16 tot_len;
    __u16 id;
    __u16 frag_off;
    __u8  ttl;
    __u8  protocol;
    __u16 check;
    __u32 saddr;
    __u32 daddr;
    /*The options start here. */
};
```

42

```
struct udphdr {
    __u16 uh_sport; /* source port */
    __u16 uh_dport; /* destination port */
    __u16 uh_ulen; /* udp length */
    __u16 uh_sum; /* udp checksum */
};
```

43

```
struct tcphdr {
    __u16 source;
    __u16 dest;
    __u32 seq;
    __u32 ack_seq;
#if defined(__LITTLE_ENDIAN_BITFIELD)
    __u16 res1:4,
        doff:4,
        fin:1,
        syn:1,
        rst:1,
        psh:1,
        ack:1,
        urg:1,
        ece:1,
        cwr:1;
...
};
```

44

```
...
#elif defined(__BIG_ENDIAN_BITFIELD)
    __u16  doff:4,
        res1:4,
        cwr:1,
        ece:1,
        urg:1,
        ack:1,
        psh:1,
        rst:1,
        syn:1,
        fin:1;
#else
#error "Adjust your <asm/byteorder.h> defines"
#endif
    __u16  window;
    __u16  check;
    __u16  urg_ptr;
};
```