Reti

(già "Reti di Calcolatori")

Cenni di Socket Programming

Renato Lo Cigno

http://disi.unitn.it/locigno/index.php/teaching-duties/computer-networks

UNIVERSITY OF TRENTO
Department of Information Engineering and Computer Science

Obiettivo: imparare a costruire un protocollo di livello applicativo usando le API socket

Socket API

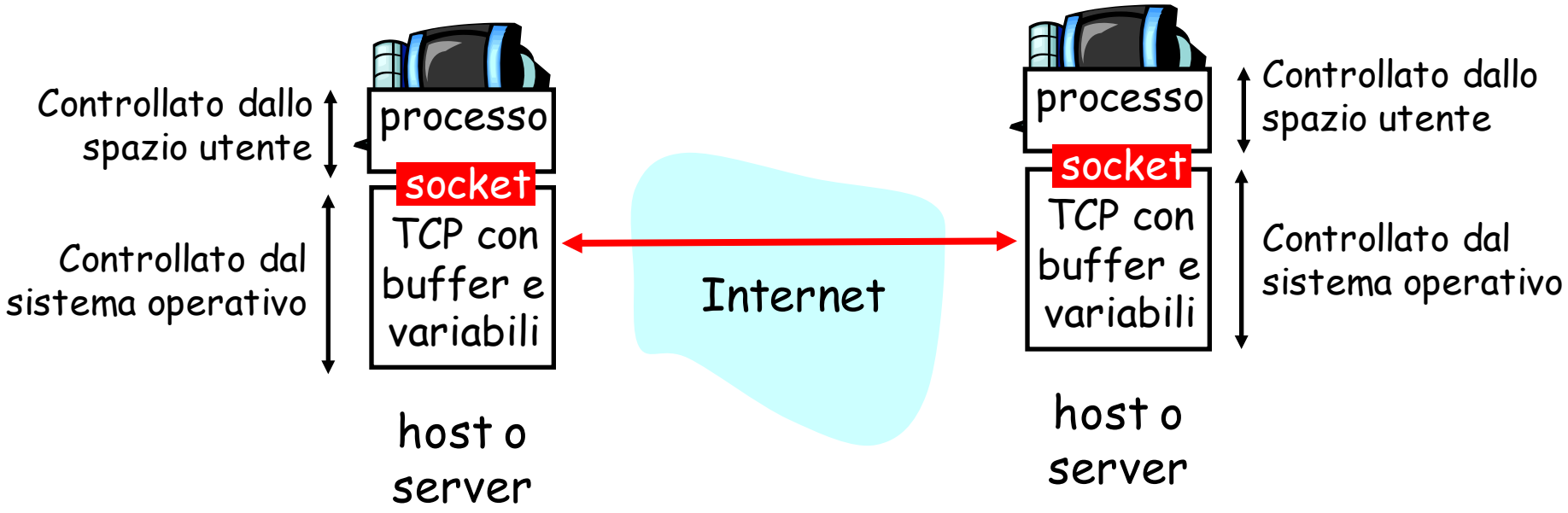
- Introdotte in BSD4.1 UNIX, nel 1981
- Esplicitamente creata, usata, distribuita dalle applicazioni
- Paradigma client/server
- Due tipi di servizio di trasporto tramite una socket API:
  - Datagramma inaffidabile (UDP)
  - Affidabile, orientata ai byte (TCP)
- Esistono anche socket per accedere direttamente a IP e anche a Ethernet

socket

Interfaccia di un
host locale,
creata dal protocollo applicativo,
controllata dal SO
(una "porta") in cui
il processo di un'applicazione può
inviare e ricevere
messaggi al/dal processo di
un'altra applicazione

Socket: una porta tra il processo di un'applicazione e il protocollo di trasporto end-end (UCP o TCP)

Servizio TCP: trasferimento affidabile di **byte** da un processo all'altro, detto anche stream socket (SOCK_STREAM)



TCP socket

Il client deve contattare il server

- Il processo server deve essere in corso di esecuzione
- Il server deve avere creato una socket (porta) che dà il benvenuto al contatto con il client

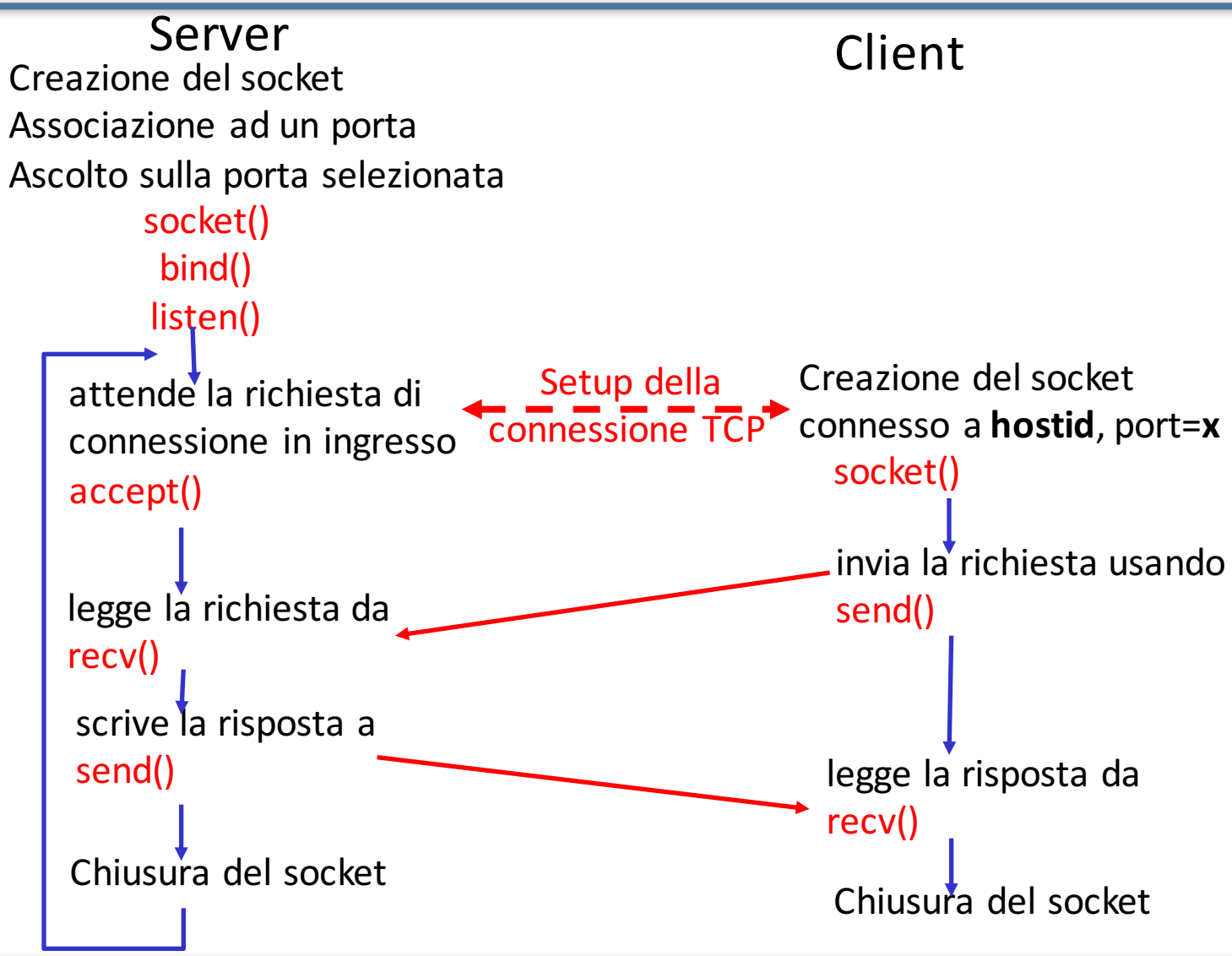
Il client contatta il server:

- Creando una socket TCP
- Specificando l'indirizzo IP, il numero di porta del processo server
- Quando il client crea la socket: il client TCP stabilisce una connessione con il server TCP

- Quando viene contattato dal client, il server TCP crea una nuova socket per il processo server per comunicare con il client
  - consente al server di comunicare con più client
  - numeri di porta origine usati per distinguere i client

TCP fornisce un trasferimento di byte affidabile e ordinato ("pipe") tra client e server

Interazione client/server: TCP



Esempi di manipolazione socket Client/Server TCP in Linguaggio C

Esempi di base

Socket Programming viene sviluppato a reti avanzate

Indirizzamento (i)

- Questa struct permette di manipolare in maniera semplice i Socket Address
  - "in" stands for Internet
  - "sin_family": AF_INET → IPv4; AF_INET6 → IPv6; AF_UNIX → a local file

```
struct sockaddr_in {
   short int sin_family; // Address family
   unsigned short int sin_port; // Port number
   struct in_addr sin_addr; // Internet address
   unsigned char sin_zero[8];
};
```

Indirizzamento (ii)

- Esistono diverse funzioni per manipolare gli indirizzi
  - inet_aton("10.12.110.57", &(my_addr.sin_addr));
  - ina.sin_addr.s_addr = inet_addr("10.12.110.57");
  - ina.sin_addr.s_addr = INADDR_ANY;

```
struct sockaddr_in my_addr;
my_addr.sin_family = AF_INET;
my_addr.sin_port = htons(MYPORT);
inet_aton("10.12.110.57", &(my_addr.sin_addr));
memset(&(my_addr.sin_zero), '\0', 8);
```

Creare un socket

- Un socket è creato utilizzando la seguente system call:
  - int socket(int domain, int type, int protocol);
- Dove:
  - Domain: PF_INET
  - Type: SOCK_STREAM or SOCK_DGRAM or SOCK_RAW
  - Protocol: IPPROTO_TCP, o altri protocolli

Associare una porta

- Se stiamo implementando un server dobbiamo attendere le nuove connessioni su una specifica porta
  - int bind(int sockfd, struct sockaddr *my_addr, int addrlen);
- Dove:
  - Sockfd: file descriptor restituito dalla funzione socket()
  - my_addr: puntatore ad una struct sockaddr che contiene le informazioni relative al socket address(porta ed indirizzo IP
  - addrlen: deve essere impostato a sizeof(struct sockaddr)

Ascoltare su una porta

- Dopo essersi associato su una porta il processo server deve rimanare in attesa di nuove connessioni
  - int listen(int sockfd, int backlog);
- Dove:
  - Sockfd: socket file descriptor ottenuto da socket()
  - Backlog: numero massimo di connessione che possono essere accettate

Accettare una connessione (i)

- Quando qualcuno cerca di connettersi al server sulla porta dove si sta ascoltando, la loro connessione viene messa in cosa in attesa di essere servita
- Nel momento in cui la connessione viene accettata il sistema operativo crea un nuovo file descriptor
- Il file descriptor originale resta ancora in attesa sulla porta originale

Accettare una connessione (ii)

- Sintassi:
  - int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
- Dove:
  - Sockfd: file descriptor del socket in ascolto
  - Addr: puntatore ad una struct utilizzata per memorizzare le informazioni sulla connessione in arrivo
  - Addrlen: da imposta a sizeof(struct sockaddr_in)

Trasmettere dati

- Sintassi:
  - int send(int sockfd, const void *msg, int len, int flags);
- Dove:
  - Sockfd: file descriptor relativo al socket da utilizzare per la trasmissione
  - Msg: puntatore ai dati da trasmettere
  - Len: lunghezza della trasmissione in byte
  - Flags: NON sono i Flag TCP, ma controlli locali per definire il funzionamento (es. consentire l'invio solo su rete locale e non su Internet)

Ricevere dati

- Sintassi:
  - int recv(int sockfd, void *buf, int len, unsigned int flags);
- Dove:
  - Sockfd: file descriptor del socket da utilizzare per la ricezione
  - buf: buffer di memoria dove scrivere I dati ricevuti
  - len: dimensione massima del buffer
  - Flags: da impostare a 0