

Telematica

L'interfaccia Socket

• Telematica

Che cos'è

- E' una API per la suite TCP/IP
 - una API (Application Program Interface) è il set di funzioni che i programmatori usano per sviluppare applicazioni in un determinato ambiente.
 - Nell'ambiente di rete per la suite TCP/IP questo set di funzioni è definito dall'interfaccia socket.

- Telematica

Le origini

- Inizialmente nasce in ambiente unix
 - Negli anni '80 la Advanced Research Project Agency finanziò l'università di Berkeley per implementare la suite TCP/IP nel sistema operativo Unix.
 - I ricercatori di Berkeley svilupparono il set originario di funzioni che fu chiamato *interfaccia socket*.

- Telematica

Le origini

- In un primo tempo si tentò di implementare le socket API come estensione delle API di UNIX per la gestione dell'I/O su periferica standard (files su disco, stampanti, etc)

- Telematica

Difficoltà di implementazione

- L' I/O di rete è molto più complesso dell'I/O delle periferiche standard:
 - Stabilire una connessione VERSO un server era semplice, ma il sistema operativo non aveva supporto per l'I/O passivo
 - UNIX identifica le periferiche con degli ID fissi, mentre la comunicazione sulla rete utilizza degli ID che indicano la DESTINAZIONE del pacchetto, ma non il prossimo dispositivo a cui inviarlo lungo il cammino (non e' orientata alla connessione)
 - A livello di sistema operativo erano previsti solo dei flussi di byte, e non la pacchettizzazione.

- Telematica

Difficoltà di implementazione

- A causa di queste difficoltà si scelse di creare delle API aggiuntive anziché modificare quelle già esistenti; l'interfaccia socket fu, tuttavia, modellata sulle chiamate di sistema unix, e ne mantiene ancora il 'sapore'

- Telematica

Nel modello TCP/IP

- Possiamo pensare al socket come al punto di arrivo delle comunicazioni di rete. Esso è il Service Access Point del protocollo TCP, e rappresenta l'astrazione di un estremo di una comunicazione di rete.

- Telematica

Nasconde le complessità del protocollo

- L'interfaccia socket svolge per noi tutte le operazioni necessarie per stabilire una connessione TCP:
 - Si occupa di creare i pacchetti TCP
 - Stabilisce una connessione con il three way handshaking
 - Gestisce in generale tutte le problematiche della trasmissione dei pacchetti (ACK / NACK / ritrasmissioni etc)

• Telematica

Nasconde le complessità del protocollo

- L'interfaccia socket gestisce anche altri protocolli di rete, permettendo un approccio abbastanza uniforme alla programmazione di rete a prescindere dall'infrastruttura protocollare sottostante.

Telematica

Programmare con i Socket

- Telematica

Descrittori di file, handles & co

- Nel paradigma di programmazione UNIX ogni periferica è utilizzata come sorgente o destinazione di un flusso di byte, detto *file*
- Poiché i socket inizialmente erano stati modellati su questo paradigma, il sistema operativo associa le strutture dati che mantengono lo stato interno del socket ad un descrittore analogo a quello utilizzato per le periferiche 'standard'.

- Telematica

Descrittori di file, handles & co

- Un descrittore di file, o un handle, e' un numero, unico a livello di applicazione e /o di sistema, che individua in una apposita tabella le strutture dati che mantengono le informazioni sullo stato dell'*oggetto* associato (socket o file)

• Telematica

Le funzioni: **socket()**

- La funzione **socket()** fornisce un'interfaccia flessibile che gestisce anche le reti diverse da TCP/IP.

```
socket_handle=socket(protocol_family,  
                     socket_type,  
                     protocol)
```

• Telematica

socket(): i parametri

- **Protocol_family**: come già detto esistono diverse *famiglie* di protocolli gestiti dalle socket API:
 - PF_INET -> protocolli per internet TCP UDP
 - PF_UNIX -> protocolli per uso interno di UNIX
 - PF_NS -> Network Services di Xerox

- Telematica

socket(): i parametri

- **socket_type**: identifica che tipo di protocollo usare all'interno della famiglia.
 - Per PF_INET sono definiti SOCK_DGRAM e SOCK_STREAM.
 - SOCKET_RAW che permette di scavalcare il livello di trasporto
 - SOCK_RDM (Reliably-Delivered Message) e SOCK_SEQPACKET (Sequenced Packet Stream) definiti ma non implementati

- Telematica

socket(): i parametri

- **protocol**: all'interno di TCP/IP esistono diversi tipi di protocollo: TCP,UDP,ICMP e IP.
 - IPPROTO_TCP
 - IPPROTO_UDP
- **chiamata tipica**:

```
socket_handle = socket(PF_INET,
                       SOCK_STREAM,
                       IPPROTO_TCP);
```

- Telematica

Identificare una connessione

- Informazioni che identificano una connessione (mantenute dal socket)
 - porta di protocollo locale
 - indirizzo locale
 - porta di protocollo remota
 - indirizzo remoto
 - tipo di protocollo

- Telematica

creare una connessione: **connect()**

```
result = connect(socket_handle,  
                 remote_socket_address,  
                 address_length);
```

socket_handle: l'handle ottenuto da una precedente chiamata a *socket()*

remote_socket_address: l'indirizzo al quale connettersi (identifica univocamente indirizzo IP e port)

address_length: lunghezza in bytes della struttura di *remote_socket_address*.

- Telematica

connect()

- Effettuando la connessione indico solo l'indirizzo della macchina remota. Non mi preoccupo dell'indirizzo e del port locali poiché si occuperà la socket API di sceglierne una coppia valida ed utilizzabile.
- E' possibile indicare esplicitamente a che indirizzo e a che port deve essere un socket.

- Telematica

Specificare un indirizzo locale: bind()

```
result = bind (socket_handle,  
              local_address,  
              address_length);
```

- I parametri sono analoghi a quelli di *connect()*.
- Anziché connettersi ad un dispositivo remoto, però, assegna l'indirizzo locale *local_address* al socket.
- Utile per
 - Server
 - Multi homed hosts se desideriamo usare un'interfaccia fisica in particolare.
 - Socket non connessi

- Telematica

Mettersi in ascolto con **listen()**

```
result = listen (socket_handle,  
                backlog);
```

- *listen()* ci permette creare un server che aspetta connessioni su un dato socket.
- *socket_handle*: restituitoci da una precedente chiamata a *socket()*. E' ovviamente necessario legare il socket ad una coppia port/indirizzo locali ben precisi con *bind()*.
- *backlog*: il numero di connessioni in attesa di essere accettate.

- Telematica

Accettare una connessione: **accept()**

```
new_socket_handle = accept (socket_handle,  
                            remote_address,  
                            address_length);
```

- accetta una connessione sul *socket_handle* usato da *listen()*.
- *remote_address*: riceve l'indirizzo del peer remoto
- *new_socket_handle*: un **nuovo** socket, restituito da *accept()*, che rappresenta la nuova connessione.

- Telematica

Inviare dati su una connessione: **send()**

```
result = send (socket_handle,  
              buffer,  
              buffer_length,  
              flags);
```

- `socket_handle`: il descrittore di socket di una connessione **già stabilita**
- `buffer`, `buffer_length`: i dati da mandare e la lunghezza (in bytes) dei dati.
- Non viene specificata la destinazione dei dati, perchè `send()` si effettua solo su un socket già connesso, in cui pertanto l'altra parte è già nota.

- Telematica

Inviare dati senza una connessione: **sendto()**

```
result = sendto (socket_handle,  
                buffer,  
                buffer_length,  
                flags,  
                remote_address,  
                remote_address_length);
```

- Abbiamo gli stessi parametri di `send()`; tuttavia, non essendoci una connessione ad identificare in maniera univoca la destinazione dei dati, occorre specificare esplicitamente l'indirizzo di destinazione `remote_address`.

- Telematica

Ricevere dati con una connessione: **recv()**

```
result = recv (socket_handle,  
              buffer,  
              buffer_length,  
              flags);
```

- `socket_handle`: il descrittore di socket di una connessione **già stabilita**
- `buffer`, `buffer_length`: il buffer dove ricevere i dati e la dimensione del buffer.

- Telematica

Inviare dati senza una connessione: **sendto()**

```
result = recvfrom(socket_handle,  
                  buffer,  
                  buffer_length,  
                  flags,  
                  remote_address,  
                  remote_address_length);
```

- `remote_address` riceverà le informazioni relative all'indirizzo del peer che ha inviato i dati

- Telematica

In sintesi: lato client

- Si crea un *socket*
- Usa *connect()* per connettersi al server
 - Se la connessione è andata a buon fine
 - *send()* mi permette di inviare dati
 - *recv()* mi permette di ricevere
- Se preferisco scambiare i dati senza stabilire una connessione non uso *connect()* e subito dopo la creazione del socket
 - *sendto()*
 - *recvfrom()*

- Telematica

In sintesi: lato server

- Si crea un *socket*
- Con *bind()* associo il socket ad un indirizzo locale.
- Se voglio un circuito virtuale uso le API per gestire la connessione:
 - Mi metto in ascolto con *listen()*
 - Accetto le connessioni con *accept()*
 - Scambio i dati con *send()* e *recv()*
- Se non voglio usare una connessione mi pongo in ascolto con *recvfrom()* e continuo poi a scambiare dati con
 - *sendto()*
 - *recvfrom()*