Corso di Laurea in Ingegneria Informatica

Corso di Reti di Calcolatori

Antonio Pescapè (pescape@unina.it)
Roberto Canonico (roberto.canonico@unina.it)

La comunicazione client/server tramite socket API: concetti generali

Nota di Copyright

Socket: cosa sono? (1)

- Le socket rappresentano un'astrazione di canale di comunicazione tra processi (locali o remoti).
- Attraverso di esse un'applicazione può ricevere o trasmettere dati.
- I meccanismi restano (quasi) indipendenti dal supporto fisico su cui le informazioni viaggiano.
- Inizialmente nascono 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*.
  - Esse originariamente apparvero nella versione 4.1cBSD di Unix.

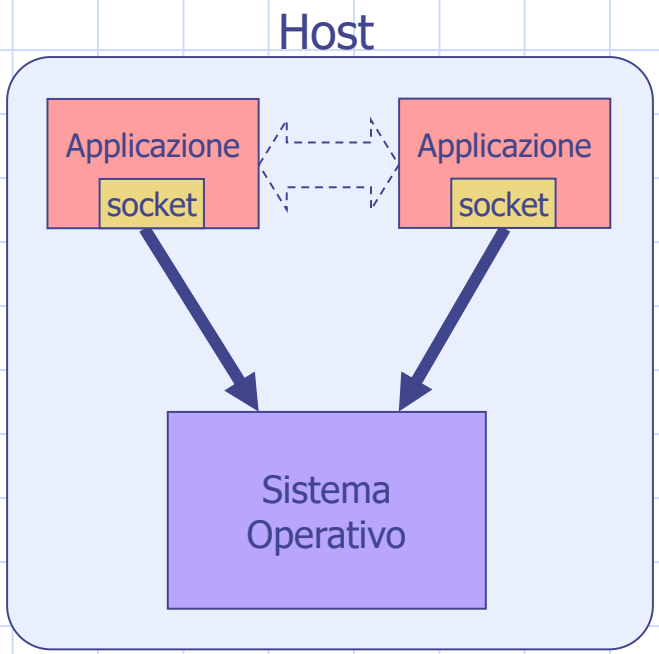
Socket: cosa sono? (2)

- Si presentano sotto la forma di un'API (Application Programming Interface), cioè un insieme di funzioni che le applicazioni possono invocare per ricevere il servizio desiderato.
- Rappresentano una estensione delle API di UNIX per la gestione dell'I/O su periferica standard (files su disco, stampanti, etc.).
- Questa API è poi divenuta uno standard *de facto*, ed oggi è diffusa nell'ambito di tutti i maggiori sistemi operativi (Linux, FreeBSD, Solaris, Windows… etc.).

Interazione tra Applicazione e SO

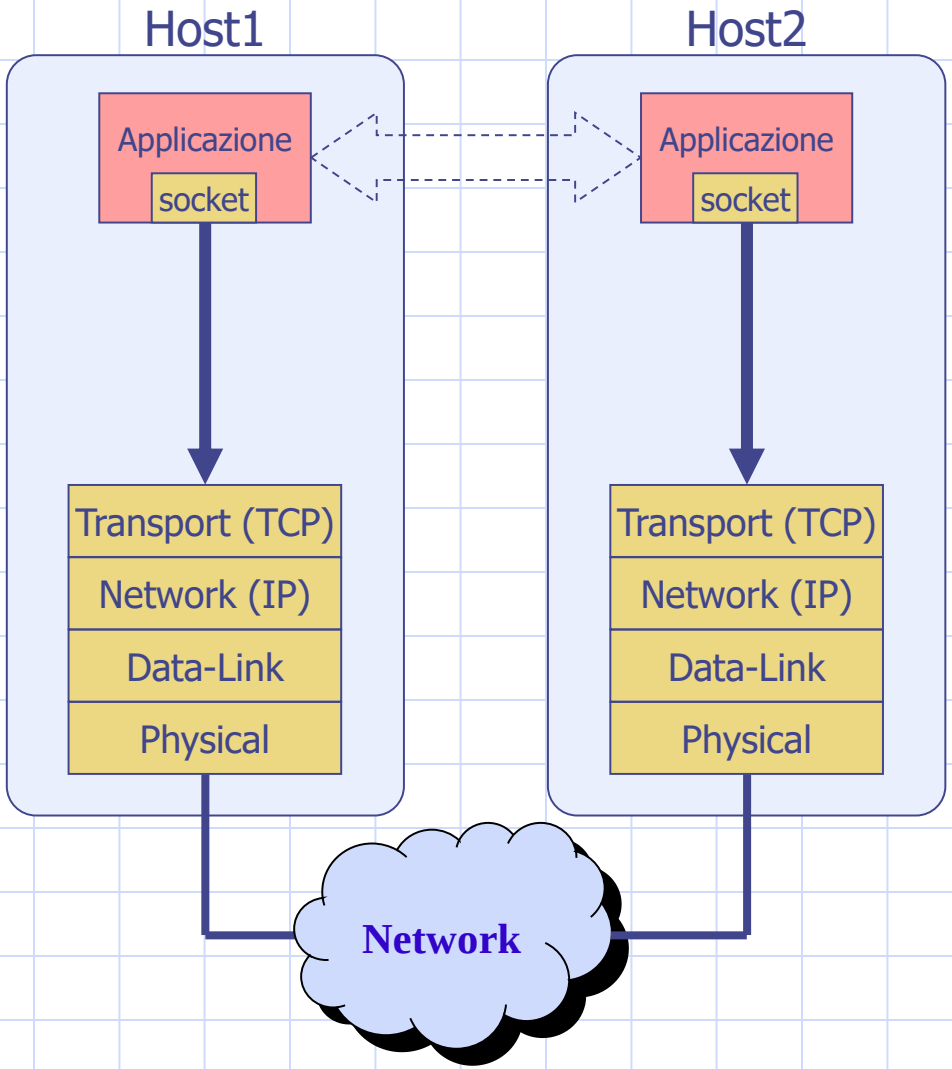
- L'applicazione chiede al sistema operativo di utilizzare i servizi di rete
- Il sistema operativo crea una socket e la restituisce all'applicazione
  - restituito un socket descriptor
- L'applicazione utilizza la socket
  - Open, Read, Write, Close.
- L'applicazione chiude la socket e la restituisce al sistema operativo

Comunicazione locale



- Due applicazioni in esecuzione sulla stessa macchina scambiano dati utilizzando l'interfaccia delle socket ed un meccanismo di *Interprocess Communication* (IPC)
- Due diversi tipi di socket:
  - socket Unix-domain (PF_UNIX)
  - socket TCP/IP (PF_INET)

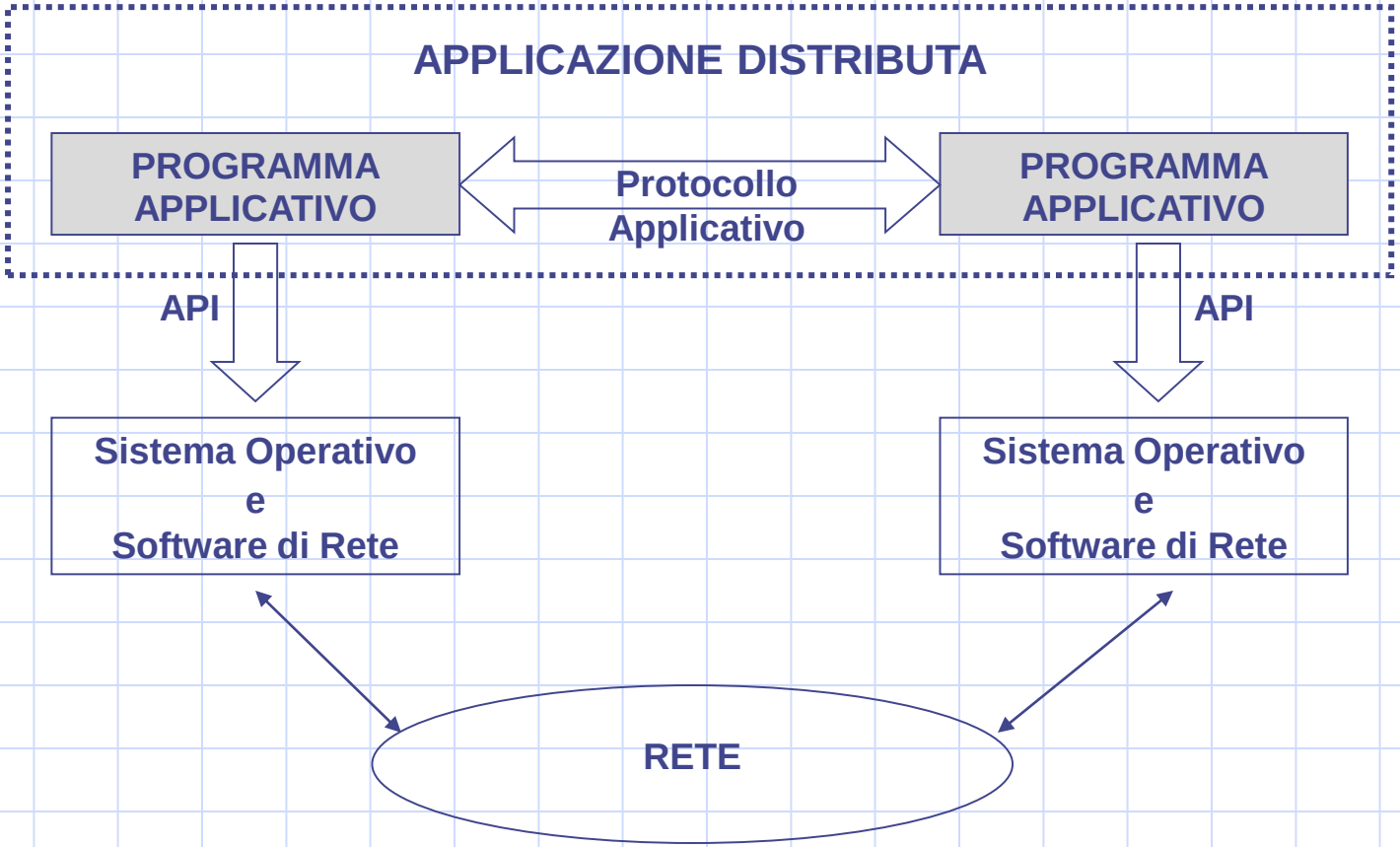
Comunicazione remota via TCP/IP



- Socket PF_INET
- Comunicazione end-to-end realizzata tramite un protocollo di trasporto (TCP o UDP)

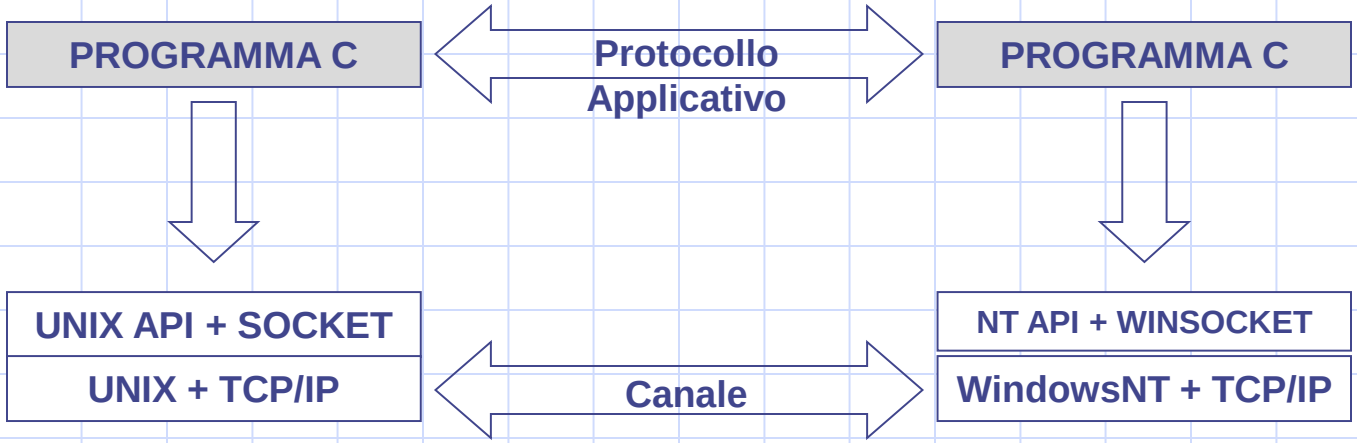
Interfacce e protocolli

- Esempio di applicazione distribuita:



Interfacce e protocolli

- La configurazione di riferimento di una applicazione distribuita basata su TCP/IP e socket è il seguente:



Il paradigma Client-Server (C/S)

- L'entità che prende l'iniziativa di comunicare è il client:
  - conosce l'indirizzo del server
  - chiede al server un determinato servizio tramite una richiesta

- L'entità che risponde a richieste di servizio è il server:
  - deve aver divulgato il proprio indirizzo
  - è in attesa di richieste di servizio

Il concetto di indirizzo

- Un terminale di comunicazione (*end-point*) è identificato univocamente da una coppia {address, process}
- Una comunicazione end-to-end è identificata attraverso la quintupla:

  {protocol, local-address, local-process, foreign-address, foreign-process}

- Per una comunicazione basata sui protocolli TCP/IP:
  - protocol è TCP oppure UDP
  - local-address e foreign-address sono indirizzi IP
  - local-process e foreign-process sono numeri di porto TCP o UDP

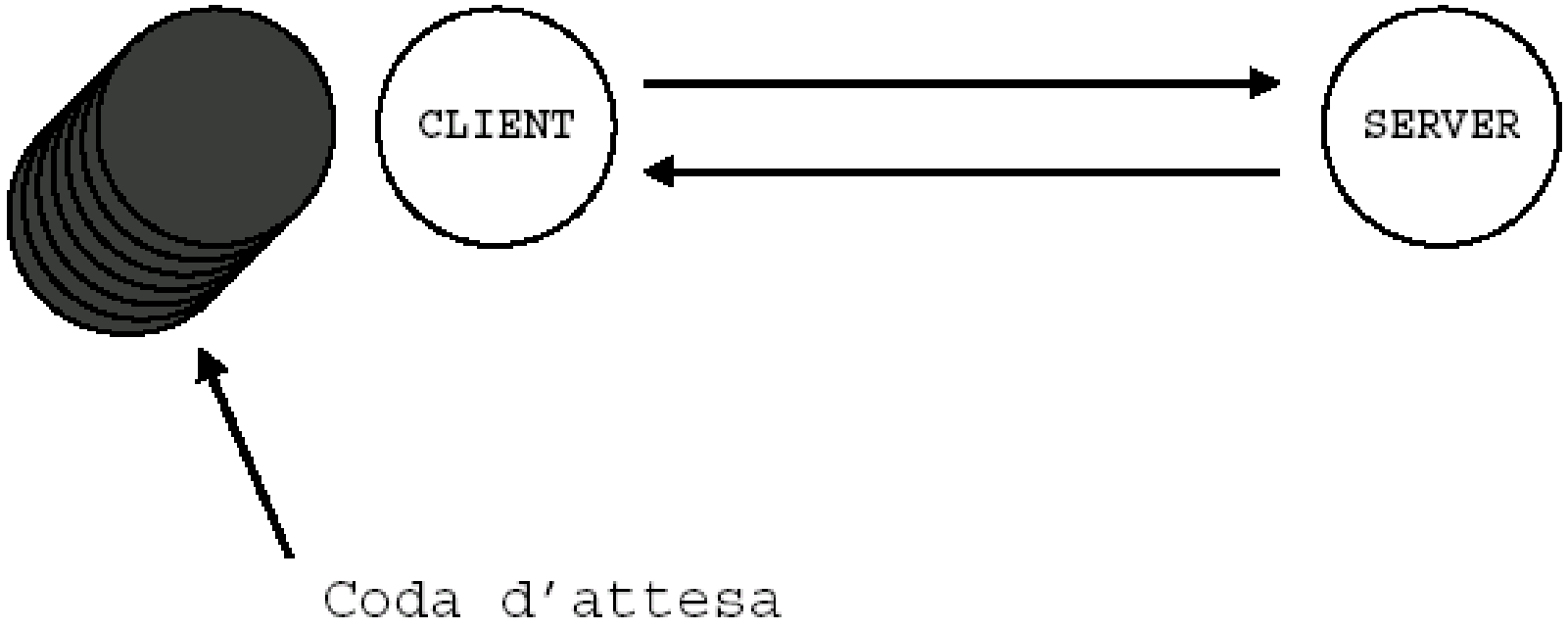
Server concorrente e iterativo

- Un server può ricevere chiamate anche da più client diversi.
- Ogni comunicazione richiederà un certo tempo prima di potersi considerare conclusa.

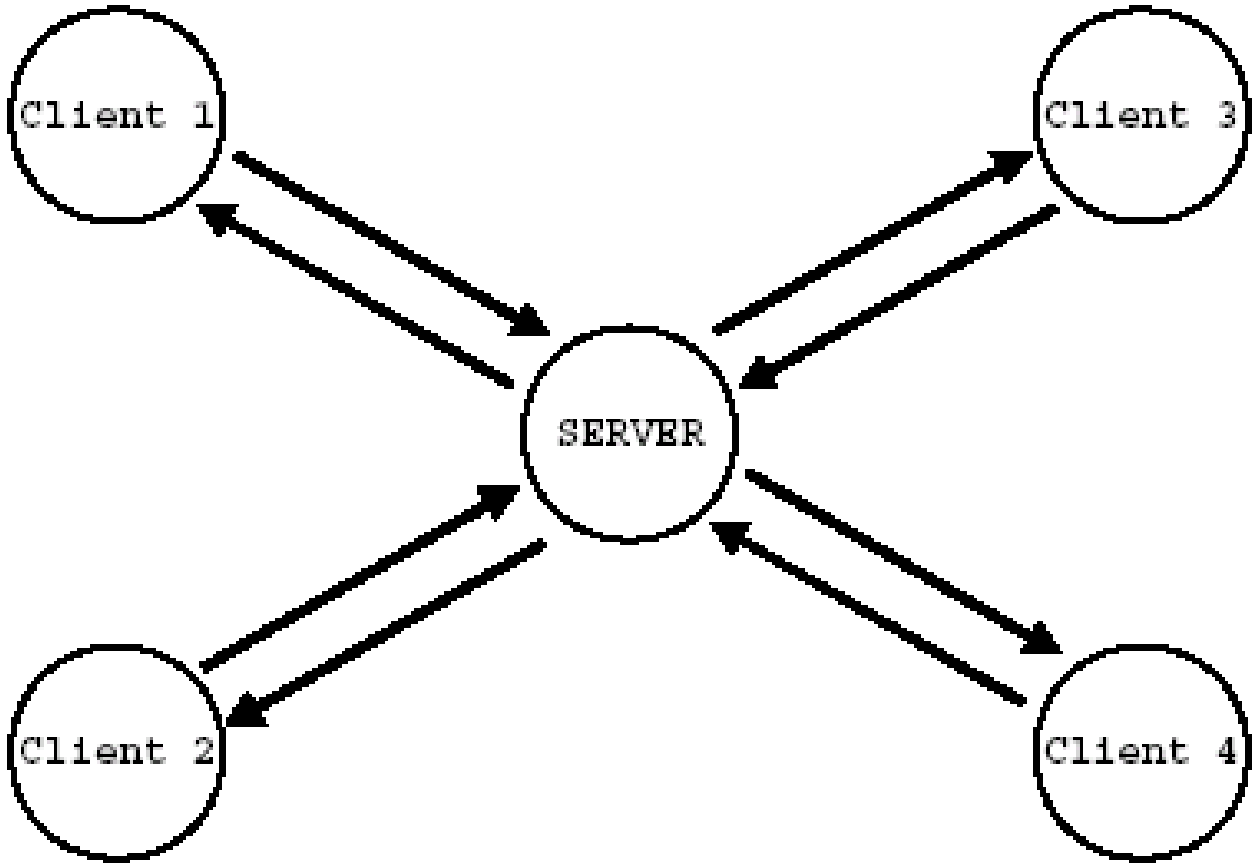
E se una chiamata arriva mentre il server è già impegnato in una comunicazione?

- Un server che accetti più comunicazioni contemporaneamente si definisce concorrente.
- Un server che accetti una sola comunicazione alla volta è detto iterativo.
  - In questo ultimo caso una richiesta può essere servita solo quando la precedente si è già conclusa.
  - Questo è il paradigma applicato nel modello di comunicazione telefonica di base.
  - E l'avviso di chiamata?...

Server Iterativo



Server Concorrente



Il paradigma di comunicazione "Connection-Oriented"

- In una comunicazione dati Connection-Oriented, i due end-point dispongono di un canale di comunicazione che:
  - trasporta flussi
  - è affidabile
  - è dedicato
  - preserva l'ordine delle informazioni
- Il canale si comporta cioè come una sorta di "tubo": tutto quello che viene inserito al suo interno, arriverà inalterato dall'altro lato e nello stesso ordine con cui è stato immesso.
- Non è detto che vengano però mantenuti i limiti dei messaggi.
- La comunicazione telefonica è più simile ad una comunicazione connection-oriented.

Il paradigma di comunicazione "Datagram"

- In una comunicazione Datagram (anche detta connectionless), il canale
  - trasporta messaggi
  - non è affidabile
  - è condiviso
  - non preserva l'ordine delle informazioni
- Se si inviano dieci messaggi dall'altro lato essi possono anche arrivare mescolati tra di loro e tra i messaggi appartenenti ad altre comunicazioni. I limiti dei messaggi vengono comunque preservati.
- La posta ordinaria è un esempio di comunicazione a datagramma.

Creazione di una socket: Byte Stream o Datagram

- Nel momento in cui si crea una socket, è necessario specificare la modalità di comunicazione che si desidera utilizzare
- Nel caso delle socket abbiamo due opzioni:
  - **byte-stream**: i dati vengono trasferiti come una sequenza ordinata di byte e consegnati in modo affidabile (`SOCK_STREAM`)
  - **datagram**: i dati vengono inviati come messaggi indipendenti senza garanzie di consegna (`SOCK_DGRAM`)
- Quando la socket è associata allo stack TCP/IP:
  - la modalità byte-stream implica il protocollo TCP
  - La modalità datagram implica il protocollo UDP

Naming / Binding

- È l'operazione con cui ad una socket già creata si associa un endpoint di trasporto specificando:

local-address, local-process

Progettazione di un Server TCP

- Creazione di un endpoint
  - Richiesta al sitema operativo
- Collegamento dell'endpoint ad una porta
  - Ascolto sulla porta
    - Processo sospeso in attesa
- Accettazione della richiesta di un client
- Letture e scritture sulla connessione
- Chiusura della connessione

Progettazione di un Client TCP

- Creazione di un endpoint
  - Richiesta al sistema operativo
- Creazione della connessione
  - Implementa open di TCP (3-way handshake)
- Lettura e scrittura sulla connessione
  - Analogo a operazioni su file in Unix
- Chiusura della connessione
  - Implementa close di TCP (4-way handshake)

Progettazione di un Server UDP

- Creazione di un endpoint
  - Richiesta al sistema operativo
- Collegamento dell'endpoint ad una porta
  - open passiva in attesa di ricevere datagram
- Ricezione ed invio di datagram
- Chiusura dell'endpoint

Progettazione di un Client UDP

- Creazione di un endpoint
  - Richiesta al sistema operativo
- Invio e ricezione di datagram
- Chiusura dell'endpoint

Comunicazione a Datagram: primitive