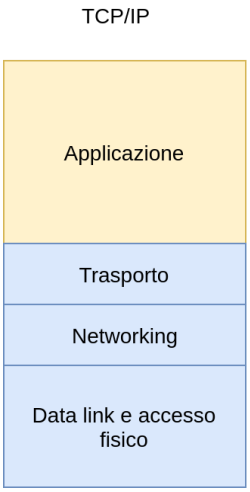
Appunti sul livello Applicativo

Livello 5 (Applicazione)



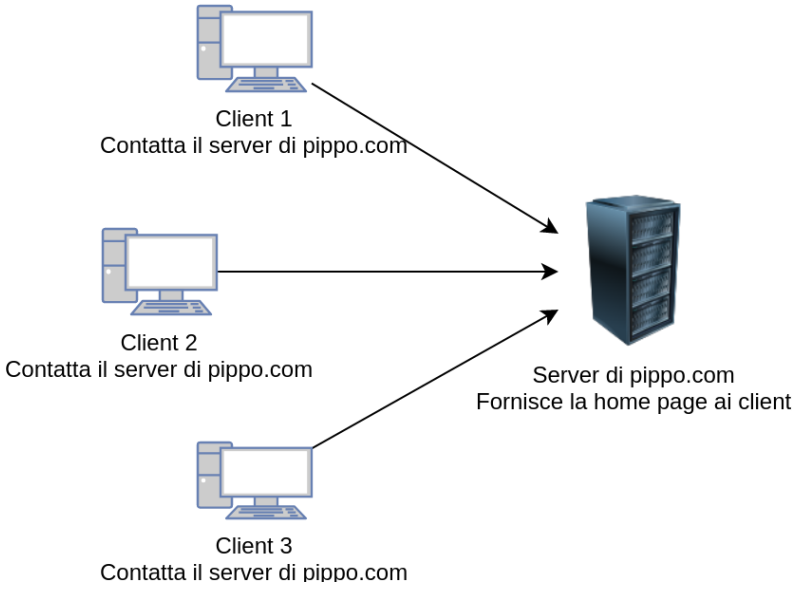
Questo è il livello più alto dello stack TCP/IP, dove l'applicazione genera il messaggio da inoltrare per l'invio ai livelli sottostanti, o riceve dai livelli sottostanti un messaggio destinatole.

Alla base dei servizi online e delle comunicazioni in rete è il **paradigma client-server**, usato dallo scenario più semplice come la semplice visita di un sito **web**, alle **e-mail**, la **messaggistica** istantanea e molto molto altro.

Paradigma client-server

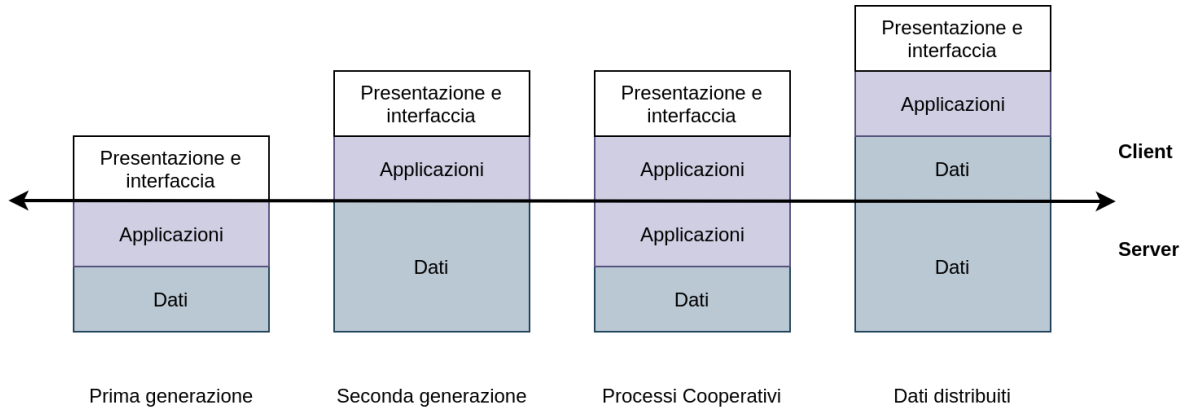
Prevede:

- Un host **server** sempre in attesa di essere "contattato" per fornire un servizio.
- Uno o più **client** che contatta il server per usufruire del servizio.



Modelli client-server

Nel corso dell'evoluzione tecnologica, sono nate diverse "generazioni" del modello client-server, anche se non necessariamente una variante ha sostituito l'altra.



- Prima generazione:
  - Il client esegue solo una interfaccia e si collega al server
  - Il server esegue la logica applicativa e mantiene i dati
  - ES: Videoterminale remoto (Desktop remoto)

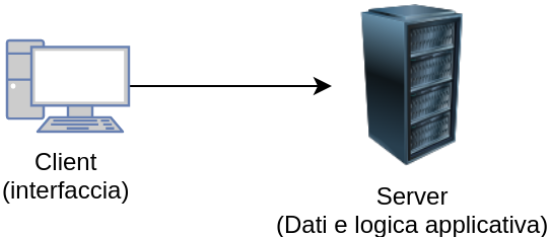
- Seconda generazione:
  - Il client esegue interfaccia e logica applicativa dell'applicazione
  - Il server viene interpellato per attingere ai dati
  - ES: Browser web

- Processi cooperativi:
  - una parte della logica applicativa viene eseguita sul client
  - una parte sul server
  - Il server mantiene i dati anche in questo caso
  - ES: Browser web (con javascript)

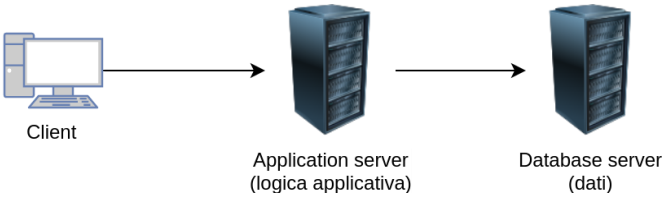
- Dati distribuiti:
  - Il client esegue la logica applicativa e mantiene una parte dei dati
  - Il server mantiene la restante parte dei dati
  - ES: Un software gestionale con database condiviso in rete

Nel caso dei server che eseguono anche la logica applicativa poi, possiamo fare la distinzione tra due tipi di architetture:

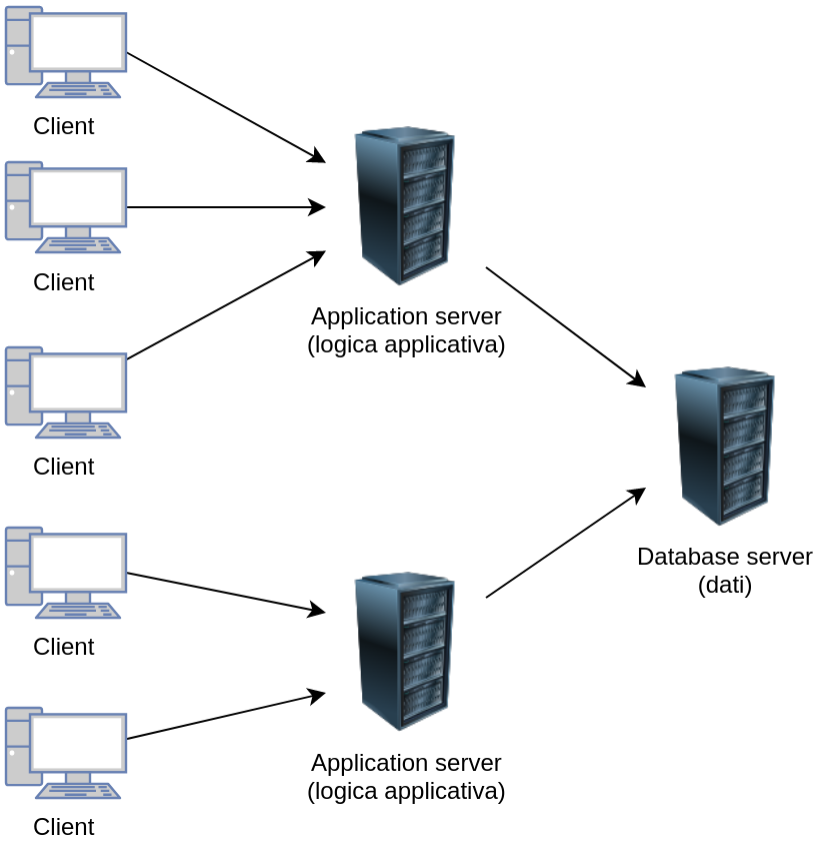
2-tier:



3-tier:



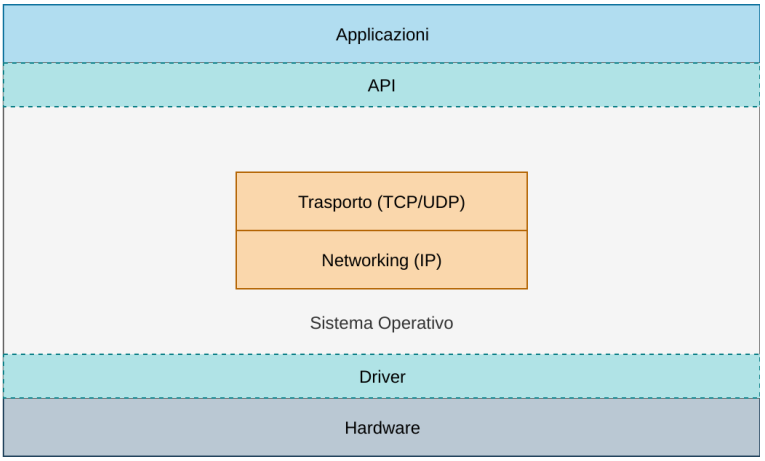
Con l'architettura 3-tier è anche possibile aggiungere più application server per gestire un maggior numero di richieste dai client.



NB: In questo scenario è generalmente abbinato un sistema di **load balancing** per smistare le richieste dei client sugli Application server.

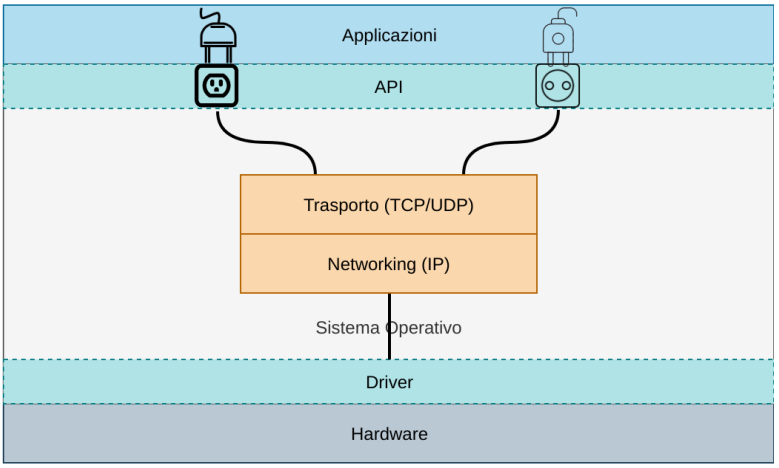
Socket API

Il meccanismo che permette alle applicazioni di “richiedere” al sistema operativo delle risorse da lui gestite, si chiama API (**A**pplication **P**rogramming **I**nterface). Possiamo immaginarlo come un sottile strato interposto tra il livello applicativo ed il sistema operativo:



I livelli 3 e 4 dello stack TCP/IP sono implementati all’interno del sistema operativo, quindi il processo di livello Applicativo (5), per inviare il messaggio dovrà passarlo al livello sottostante (4, Trasporto), usando le API del sistema operativo.

Per le applicazioni di rete queste API si chiamano “**Socket**” - dall’inglese **presa** (di corrente)



Così l’applicazione deve solo chiedere al sistema operativo un socket a cui “agganciarsi” e “scrivere” i messaggi su quel socket.

Ricapitolando:

- Due processi comunicano inviando dati e leggendo dati da un socket
- Il socket è una “interfaccia” tra il processo applicativo e il protocollo di trasporto (TCP o UDP a seconda di che tipo di socket viene richiesto).
- Ogni processo applicativo può aprire uno o più socket per contattare simultaneamente uno o più host remoti

Come si usa un socket?

È molto semplice:

- Creiamo il socket
- Lo usiamo per collegarci ad un host (remoto o locale)
- Inviamo i dati al suo interno

main.py

```
1  from socket import *
2
3  s = socket(AF_INET, SOCK_STREAM)
4  s.connect((127.0.0.1, 65432))
5  s.sendall(b'Hello, world')
6
```

Un esempio di socket in python

Qui una guida completa allo sviluppo con le socket API in Python.