Sistema client/server

Da Wikipedia, l'enciclopedia libera.

Un'applicazione ***client-server*** (letteralmente *cliente-servente*) è un tipo di applicazione di rete nel quale un computer *client* istanzia l'interfaccia utente di un'applicazione connettendosi ad una *server application* o ad un sistema di *database*.

Più semplicemente, i sistemi *client/server* sono un'evoluzione dei sistemi basati sulla condivisione semplice delle risorse.

La presenza di un *server* permette ad un certo numero di *client* di condividerne le risorse, lasciando che sia il *server* a gestire gli accessi alle risorse per evitare conflitti tipici dei primi sistemi informatici.

Un sistema *client/server* non è altro che la trasposizione informatica dei sistemi convenzionali.

Indice

[nascondi]

client

Il software *client* in genere è di limitata complessità, limitandosi normalmente ad operare come interfaccia verso il *server*. In generale nel campo informatico il termine *client* indica una componente che accede ai servizi o alle risorse di un'altra componente, detta *server*. In questo contesto si può quindi parlare di *client* riferendosi all'hardware o al software.

Un computer collegato ad un *server* tramite rete locale o geografica, ed al quale richiede uno o più servizi, utilizzando uno o più protocolli di rete è un esempio di *client* hardware.

Un programma di posta elettronica è un esempio di *client* software.

Sono sempre di più i software, come il web, l'e-mail, i database, che sono divisi in una parte *client* (residente ed in esecuzione sul pc *client*) ed una parte *server* (residente ed in esecuzione sul *server*).

Il termine *client* indica anche il software usato sul computer *client* per accedere alle funzionalità offerte dal *server*.

Ad esempio, nel web il software *client* è il browser, e parla con un *server* web attraverso il protocollo HTTP; per l'e-mail il *client* è detto in gergo mail user agent o MUA (ad esempio, Outlook, Mozilla Thunderbird, Eudora, ...), e parla con il *server* attraverso i protocollo SMTP e POP o IMAP; il *client* per la consultazione o la modifica del database (spesso costituito da librerie software utilizzate da una applicazione) parla con il DBMS, che gestisce il database e risponde alle interrogazioni del *client*.

server

Il software *server*, oltre alla gestione logica del sistema, deve implementare tutte le tecniche di gestione degli accessi, allocazione e rilascio delle risorse, condivisione e sicurezza dei dati o delle risorse.

Ad esempio un *server* di posta elettronica è paragonabile ad un qualunque ufficio postale. Gli utilizzatori per accedere via *client* alla loro cassetta di posta elettronica devono essere stati autorizzati. In modo analogo un utente deve possedere la chiave della cassetta sita presso un ufficio postale dalla quale vuole prelevare la corrispondenza.

Interazioni *client/server*

Quando un *computer client* si connette direttamente ad un sistema di *database* o a una *server application* standard, questa viene chiamata **2-tier architecture** (architettura a 2 livelli).

Recentemente, è più usuale per *computer client*, chiamati *thin client* che non incorporano *business logic*, ma solo elementi di interfaccia, connettersi ad una *server application* che implementa una *business logic* nella quale transitivamente (ossia successivamente) comunica con il database del *server*, il quale memorizza i dati utilizzati dall'applicazione. Tale architettura è chiamata **3-tier architecture** (architettura a 3 livelli).

In generale architetture ad *n* livelli possono impiegare un certo numero di servizi distinti, comprese relazioni transitive tra *application server* che implementano differenti funzioni di *business logic*, ognuna delle quali può impiegare o meno un sistema di database condiviso o distinto.

Collegamento

I *client* ed il *server* sono in collegamento tramite un protocollo di comunicazione attraverso una rete di comunicazione. Il protocollo può essere in chiaro o in certi casi crittografato. Nell'ambito delle telecomunicazioni, due o più macchine o host (computer, telefono, stampante, ecc...) possono comunicare tra loro rispettando norme che sono dette protocolli di rete. L'aderenza ai protocolli garantisce che due macchine possano comunicare correttamente, anche se sono state realizzate indipendentemente.

Servizio con e senza connessione

Quando un *client* e un *server* iniziano a comunicare si possono scambiare pacchetti di controllo prima di spedire i dati reali.

Queste procedure, dette di *handshaking* preparano le due componenti alla comunicazione. Tali procedure sono alla base, ad esempio, del TCP.

Tuttavia possono anche esserci servizi che inviano direttamente i dati come nel caso dell'UDP.

La maggior parte delle applicazioni, tuttavia, ha bisogno di inviare i dati in maniera sicura e affidabile per cui l'handshake serve proprio a questo compito. Si comprende come la connessione con TCP, ad esempio, sia più sicura ma anche più lenta perché scambia non solo dati reali, ma anche dati di servizio.

I livelli

Ciascun protocollo regola normalmente solo una parte degli aspetti di una comunicazione. I diversi protocolli sono organizzati con un sistema detto "a livelli" : a ciascun livello viene usato uno specifico protocollo.

La divisione in livelli è fatta in modo che ciascun livello utilizzi i servizi offerti dal livello inferiore, e fornisca servizi più "ricchi" al livello superiore. I diversi livelli in un host comunicano tra loro tramite le interfacce. Ogni livello parla solo con quello immediatamente superiore e con quello immediatamente inferiore. I protocolli regolano invece la comunicazione tra due entità dello stesso livello, che serve a fornire servizi al livello superiore.

I vari livelli sono organizzati in pile di protocolli. Le pile di protocolli sono un modo flessibile per combinare componenti per realizzare un servizio.

Un esempio reale di una organizzazione a livelli protocollari, classico nelle trattazioni inerenti le reti di calcolatori, è quello del percorso di una valigia in un viaggio aereo partendo dalla casa di origine all'hotel di destinazione. Il primo livello che notiamo è quello della preparazione della valigia: il turista prende i vestiti e ve li ripone per poi chiuderla, come ciò viene fatto è definito dal protocollo del primo livello 1°. Il livello 2° è quello dell'addetta alla valigie all'aeroporto di partenza, il turista gli consegna la valigia (passaggio dal primo al secondo livello) e l'addetta attacca alla valigia le informazioni relative al volo e alla destinazione. Qui notiamo l'aspetto fondamentale dell'organizzazione a livelli protocollari, cioè che per l'addetta non è necessario conoscere come i vestiti sono stati riposti nella valigia, altresì non è necessario per il turista conoscere le operazioni che deve effettuare l'addetta, infatti il turista otterra cioè che vuole (avere i vestiti all'hotel d'arrivo) senza che ciò influisca affatto come gli altri protocolli debbano lavorare, a patto che lo facciano correttamente.

La struttura serve ad adempiere ad alcuni compiti:

- controllo dell'errore;
- controllo del flusso;
- frammentazione e riassemblaggio;
- *multiplexing*, in modo che sessioni dello strato più alto possano condividere una singola connessione dello strato più basso;
- instaurazione della connessione.

Tale architettura presenta vantaggi concettuali e strutturali anche se alcuni si sono opposti in maniera decisa in quanto uno strato spesso duplica le funzionalità di un altro strato in maniera ripetitiva.

Ad esempio, il servizio di ADSL viene fornito con diverse modalità, le più comuni sono chiamate PPP over ATM (ovvero il protocollo Point to Point usa i servizi forniti dal protocollo ATM) e PPP over Ethernet.

Il livello più basso è detto "livello fisico" e si occupa di gestire la trasmissione dei segnali attraverso il mezzo di trasporto (cavo, fibra ottica, infrarossi, ecc...). Il livello più elevato è chiamato "livello applicativo" ed è quello che permette all'utente di creare il messaggio da comunicare.

La divisione in livelli è piuttosto rigida a livello di specifica dei protocolli, mentre nell'implementazione spesso diversi livelli vengono implementati insieme in uno stesso modulo software.

Non è detto che due macchine che comunicano usino la stessa pila di protocolli. Ad esempio, se vi connettete ad internet attraverso un modem voi appoggiate il livello di rete IP su una connessione ppp, mentre il *server* a cui vi collegate probabilmente appoggia la rete IP su una connessione ethernet.

In una rete a pacchetto ciascun livello della "pila protocollare" aggiunge ai pacchetti una intestazione, attraverso una operazione detta imbustamento. Il termine si applica anche ad alcune reti a commutazione di circuito, come SDH, dove l'imbustamento è un circuito dedicato a trasmettere informazioni di controllo.

L'OSI

L'International Organisation for Standardisation (ISO) nel 1979 ha stabilito il protocollo Open Systems Interconnection (OSI), con l'intenzione di creare uno standard per le telecomunicazioni da usare nelle reti di tutto il mondo. All'atto pratico però, lo standard de facto che viene comunemente usato nella maggior parte delle reti, è il TCP/IP, definito nella RFC 1155. Le differenze fondamentali dei due standard sono semplici: il primo è stato definito a tavolino da un'organizzazione super partes, mentre il secondo è opera di chi costruì materialmente le prime reti, sviluppandolo sul campo. Inoltre, lo standard ISO/OSI assegna un determinato compito ad ogni livello, mentre il TCP/IP è più "elastico" e permette di sviluppare protocolli che svolgono più di un compito-base.

Elenco di protocolli di rete secondo ISO/OSI Nella seguente suddivisione, si segue lo standard ISO/OSI.

- Livello 1: fisico
  - Bluetooth
  - DSL Digital Subscriber Line
  - FDDI
  - RS-232
  - Wi-Fi
  - Ultra Wide Band (UWB)
- Livello 2: link
  - Ethernet
  - Point-to-Point Protocol (PPP)
  - Token ring
- Livello 3: rete
  - ATM Asynchronous Transfer Mode
  - IP Internet Protocol
  - X.25
  - IPX
- Livello 4: trasporto
  - TCP e UDP (usati su IP)
  - SPX (usato su IPX)
  - NetBIOS
- Livello 5: sessione
- Livello 6: presentazione
- Livello 7: applicazione

Protocolli di servizio:

- DHCP Dynamic Host Configuration Protocol
- DNS Domain Name System
- Finger Servizio unix che fornisce informazioni sugli utenti
- NTP Network Time Protocol
- SNMP Simple Network Management Protocol
- LDAP Lightweight Directory Access Protocol

Protocolli di accesso a terminali remoti:

- Telnet
- SSH Secure shell

Protocolli usati per realizzare il servizio di posta elettronica e newsgroup:

- SMTP Simple Mail Transfer Protocol
- POP Post Office Protocol
- IMAP Internet Message Access Protocol
- NNTP Network News Transfer Protocol

Protocolli di trasferimento file:

- FTP File Transfer Protocol
- HTTP HyperText Transport Protocol
- IRC Internet Relay Chat
- Gnutella Condivisione file peer-to-peer

I livelli visti precedentemente sono una suddivisione concettuale, ma la loro implementazione non è uniforme, infatti il livello fisico e quello di collegamento di solito sono presenti sulla scheda di interfaccia della rete, mentre il livello di rete ha un'implementazione mista hardware-software.

Esempi

Esempi di sistemi *client/server*:

- Database server: per la gestione di grandi moli di dati
- File server: per la condivisione dei file
- FTP server: per la gestione dell'upload/download dei file
- Groupware: per la gestione d'informazioni riguardanti gruppi di lavoro
- Print server: per la condivisione delle stampanti
- Web server: per la gestione dell'interazione via web tra *server* e *client*

Client-server in un sistema locale

Quasi tutti i sistemi operativi utilizzano, per il funzionamento dei vari processi, dei meccanismi basati sul modello client-server. Lo stesso kernel si comporta come server quando gestisce le chiamate alle primitive di sistema da parte dei processi in esecuzione.

Più in generale in un sistema operativo, per alcuni tipi di servizi, sono espressamente previsti dei **processi server**, gli unici in grado di eseguire una certa operazione. Spesso questi processi hanno accesso esclusivo ad una risorsa e devono, appunto, *servire* le richieste dei processi client.

Ad esempio, nei sistemi Windows è presente uno spooler, unico processo in tutto il sistema a poter utilizzare la stampante. Per poter stampare, un processo non deve interfacciarsi con il driver della stampante (ed eseguire la sequenza del driver virtualizzato *acquisizione-uso-rilascio*) ma deve inviare i propri dati, attraverso i servizi offerti dal sistema, al processo spooler, il quale, tra l'altro, effettua lo scheduling dei documenti da stampare. Una volta inviati i dati allo spooler, il processo saprà non che la stampa è stata eseguita, ma che lo sarà certamente (a meno di intoppi). Avendo accesso esclusivo a tutte le stampanti di sistema, il driver della stampante non prevede procedure di acquisizione e rilascio.