

Parte Quinta: Sistemi Operativi e Applicazioni Software

Fondamenti di informatica



Sistemi Operativi

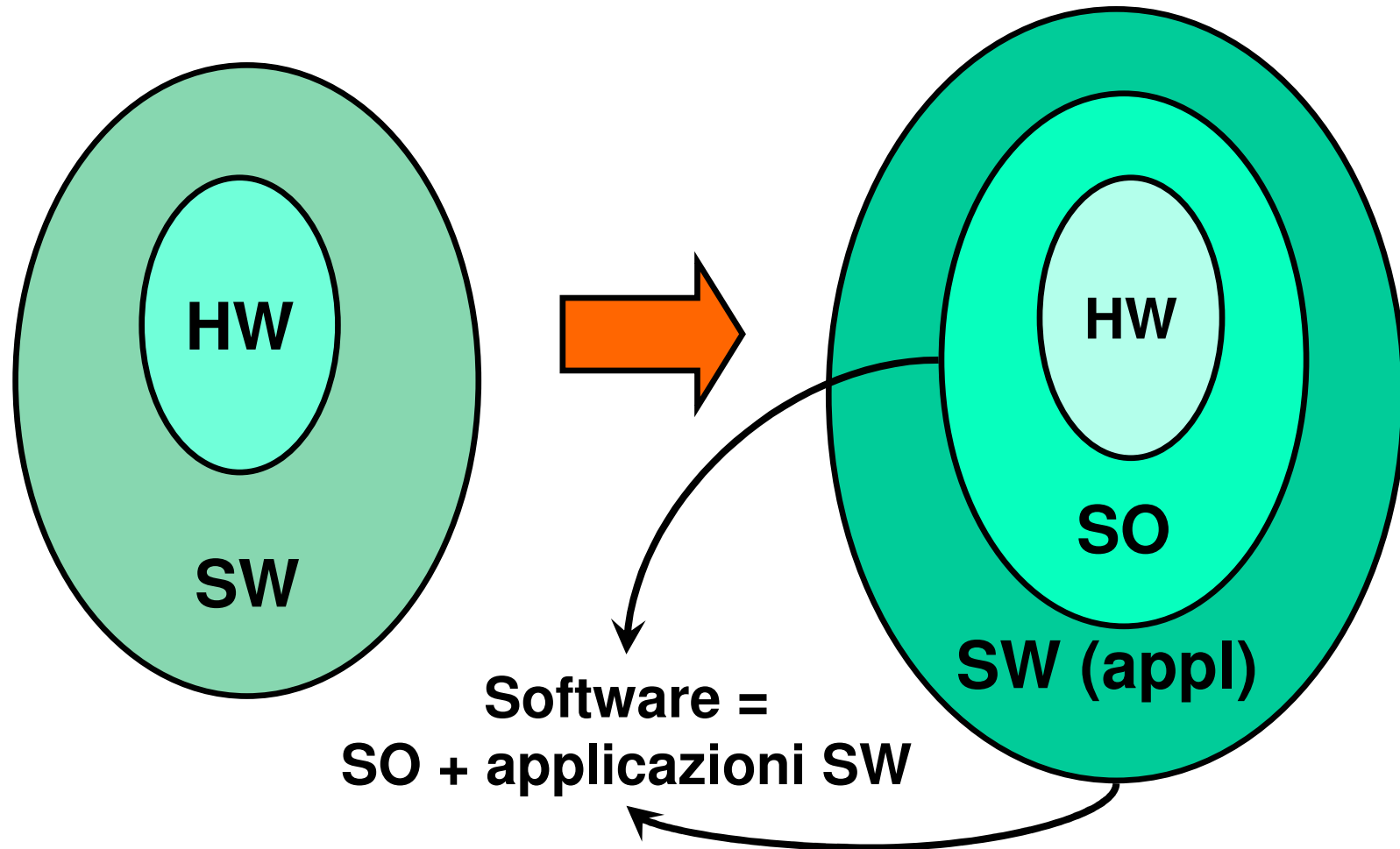
I Sistemi Operativi

- I sistemi operativi permettono di **gestire le risorse** efficientemente
 - tengono traccia di chi accede alle risorse
 - accettano e soddisfano le richieste di uso di risorse
 - risolvono i conflitti tra più risorse
- Possono essere visti come una **macchina di calcolo estesa**
 - rappresentano la base su cui è possibile scrivere programmi applicativi in modo più semplice che utilizzando direttamente l'HW.

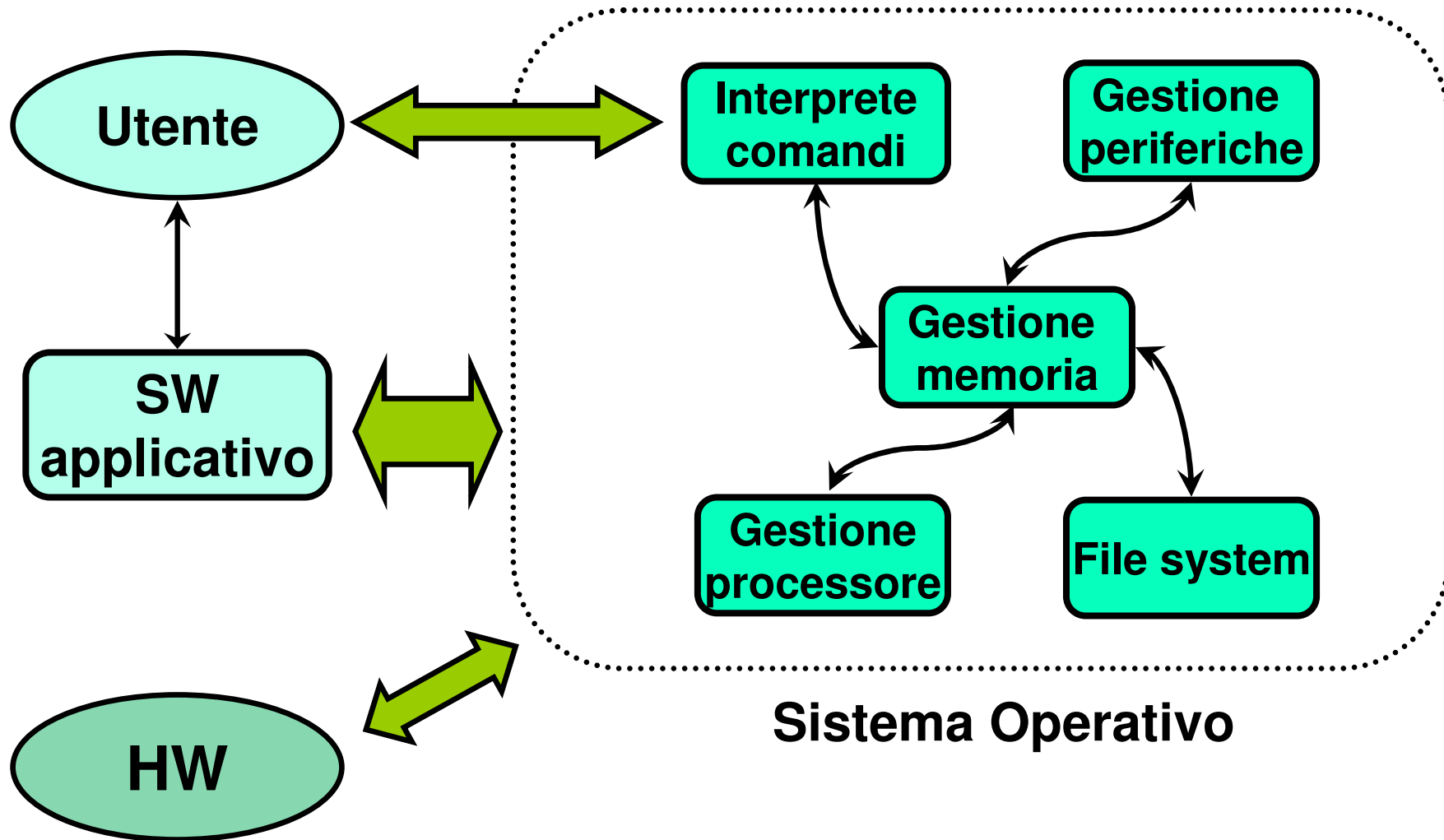
Vantaggi

- I sistemi operativi permettono definire uno standard per interfacciare i dispositivi fisici, per cui:
 - lo sviluppo dei programmi risulti più semplice ed indipendente dal calcolatore che si utilizza
 - l'aggiornamento del SW di base e dell'HW sia trasparente all'utente ed alle applicazioni.

Il SO come intermediario tra HW e SW



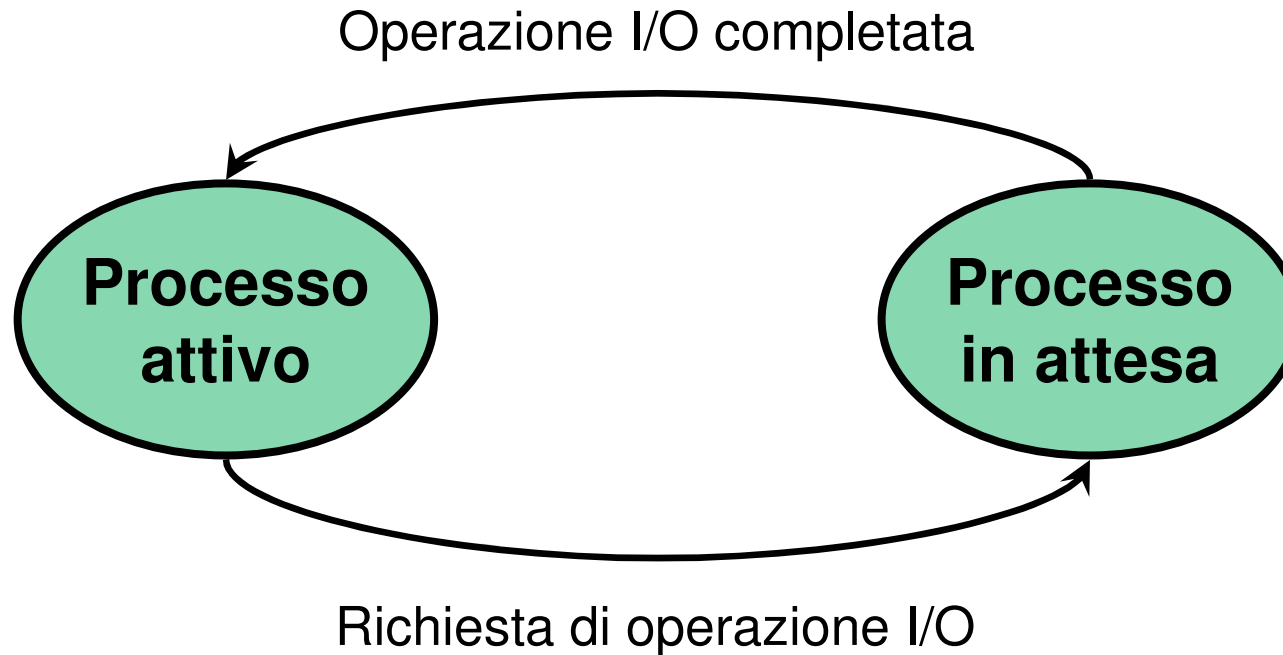
Composizione di un SO



Processi e programmi

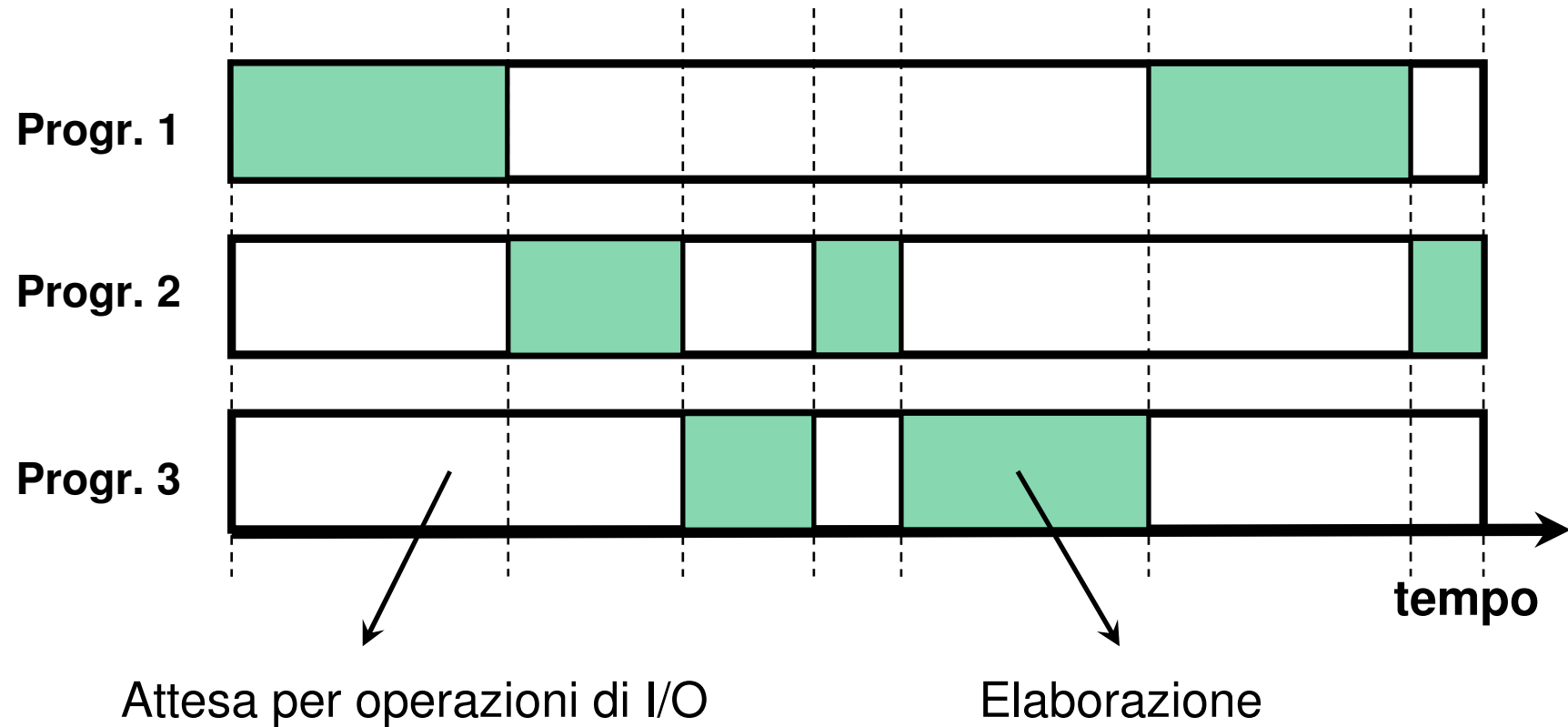
- Un **programma** è una entità statica composta dal codice eseguibile del processore.
- Un **processo** è una entità dinamica relativa al programma *in esecuzione*, ed è composto da:
 - codice del programma
 - dati necessari all'esecuzione del programma
 - stato dell'esecuzione

Esecuzione di un processo



Ogni operazione di I/O consiste in una chiamata al SO e successiva sospensione del processo utente per attendere l'esecuzione dell'operazione di I/O

La Multiprogrammazione



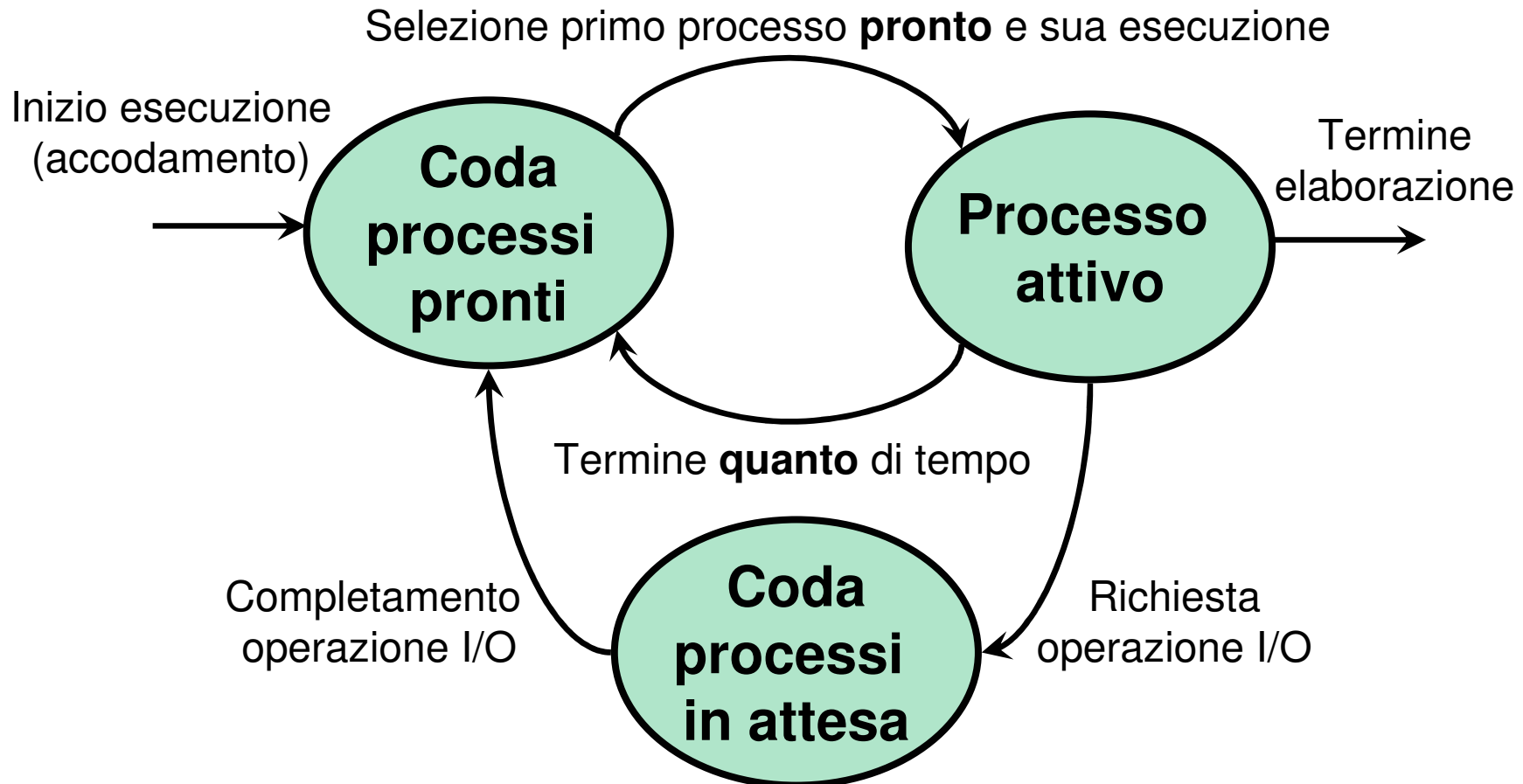
Time Sharing

- È possibile condividere la CPU tra più processi interattivi, **suddividendo** il tempo di esecuzione del processore tra più utenti
- Ogni processo utilizza periodicamente un intervallo di tempo prestabilito (**quanto**)
- Durante il quanto di esecuzione di un processo, tutti gli altri processi sono sospesi
- Al termine di ogni quanto (**context switch**), il processo in esecuzione viene sospeso e si assegna la CPU ad un altro processo.

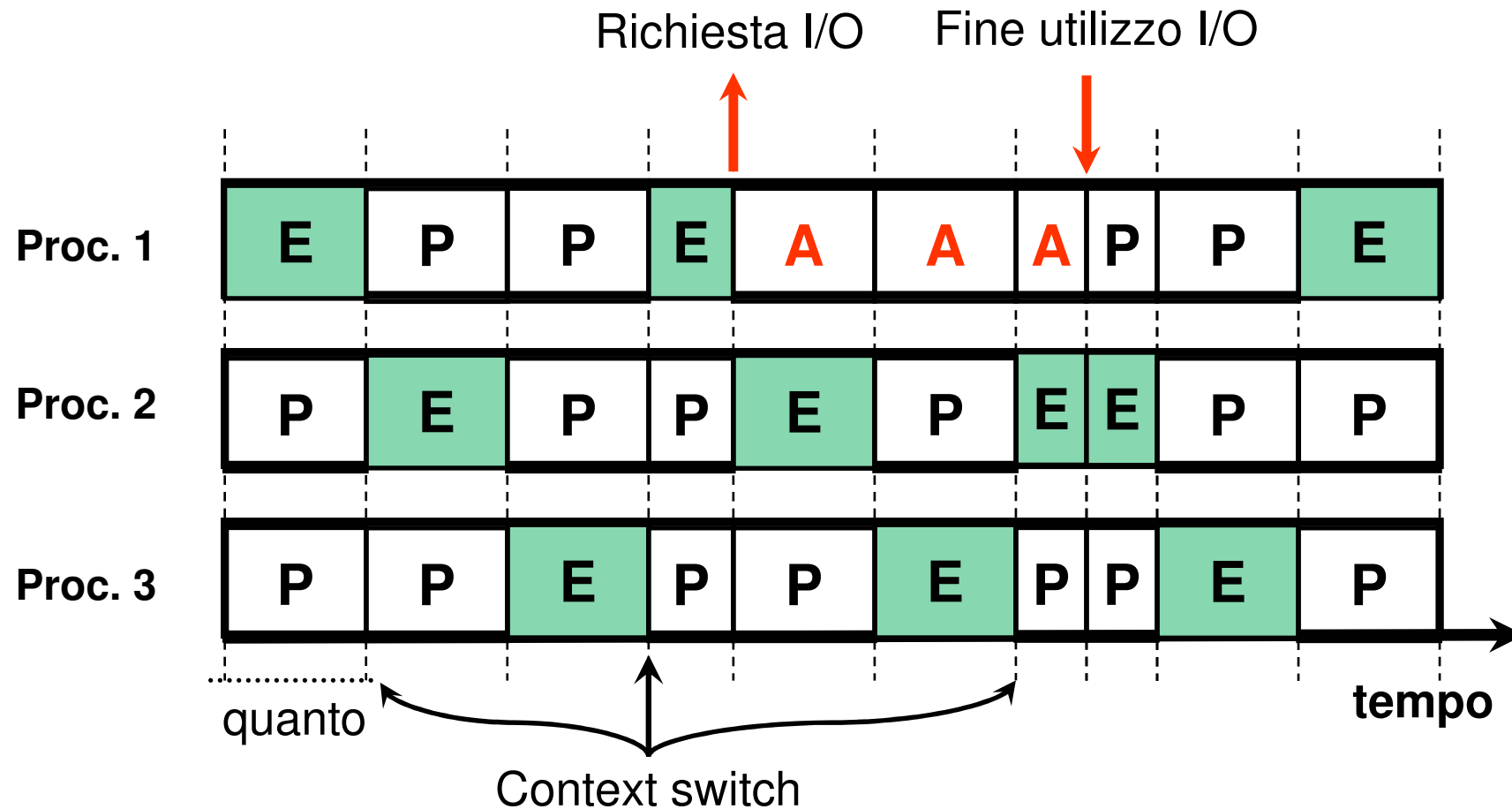
Processi “pronti” ed “in attesa”

- Quando un processo **non è in esecuzione** può assumere due diversi stati:
 - **attesa**: il processo è sospeso finché un determinato evento esterno non si verifica (i.e., I/O)
 - **pronto**: il processo è sospeso finché non gli viene concesso l'uso della CPU
 - in ogni istante di tempo vi è **un solo** processo attivo, e tutti gli altri sono *o in attesa o pronti*, e memorizzati in apposite **code** che ne indicano l'ordine di esecuzione.

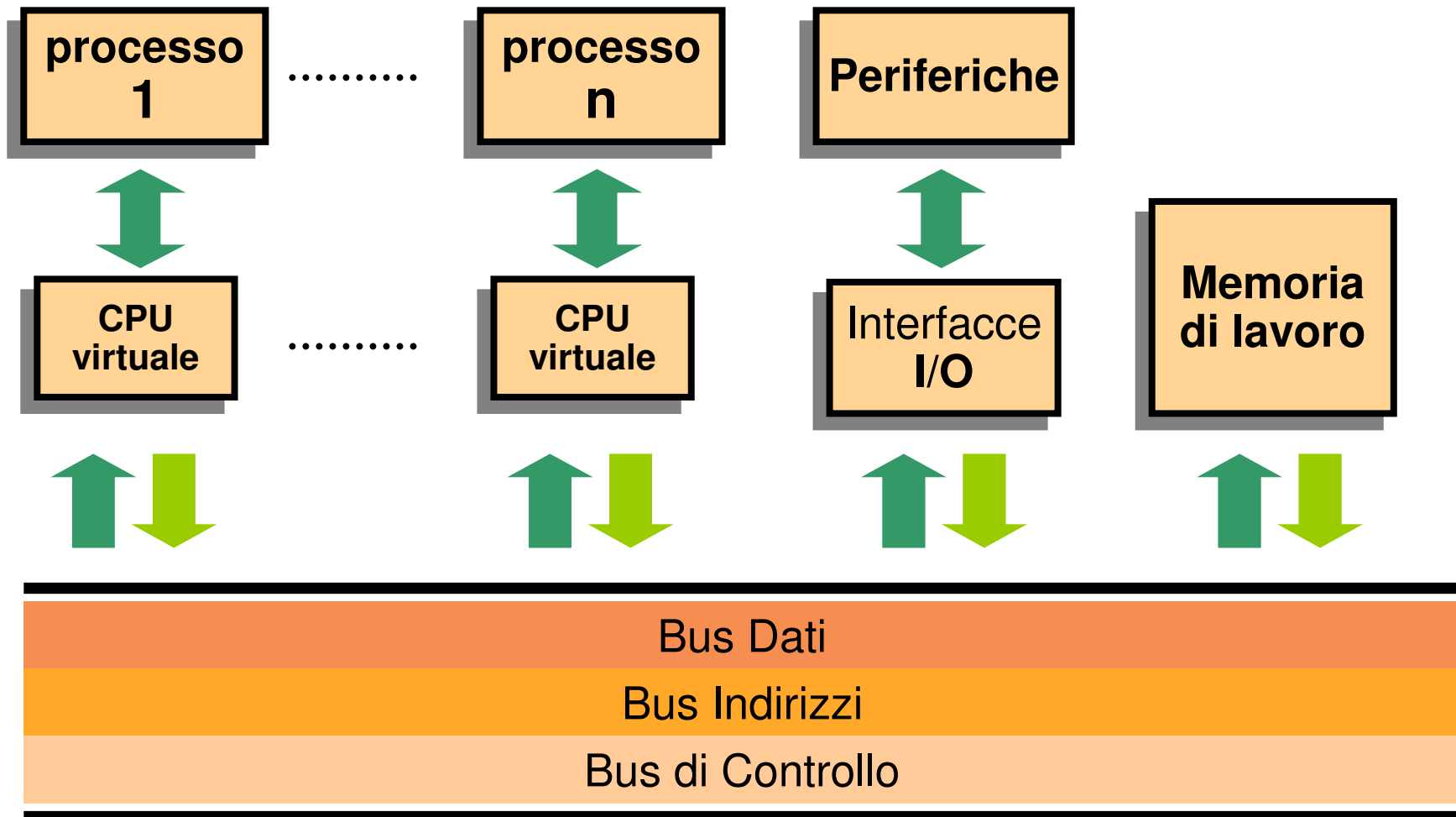
Diagramma di esecuzione



Round Robin



Macchina astratta del Kernel



Gestione della memoria

- Ogni processo necessita di una certa quantità di memoria (ad esempio per immagazzinare il codice ed i dati utilizzati)
- Spesso l'effettiva memoria **fisica** non è sufficiente a contenere tutto lo spazio richiesto per **n** processi
- Il **gestore della memoria** risolve i conflitti garantendo uno **spazio di memoria virtuale anche** superiore alla capacità della memoria fisica.

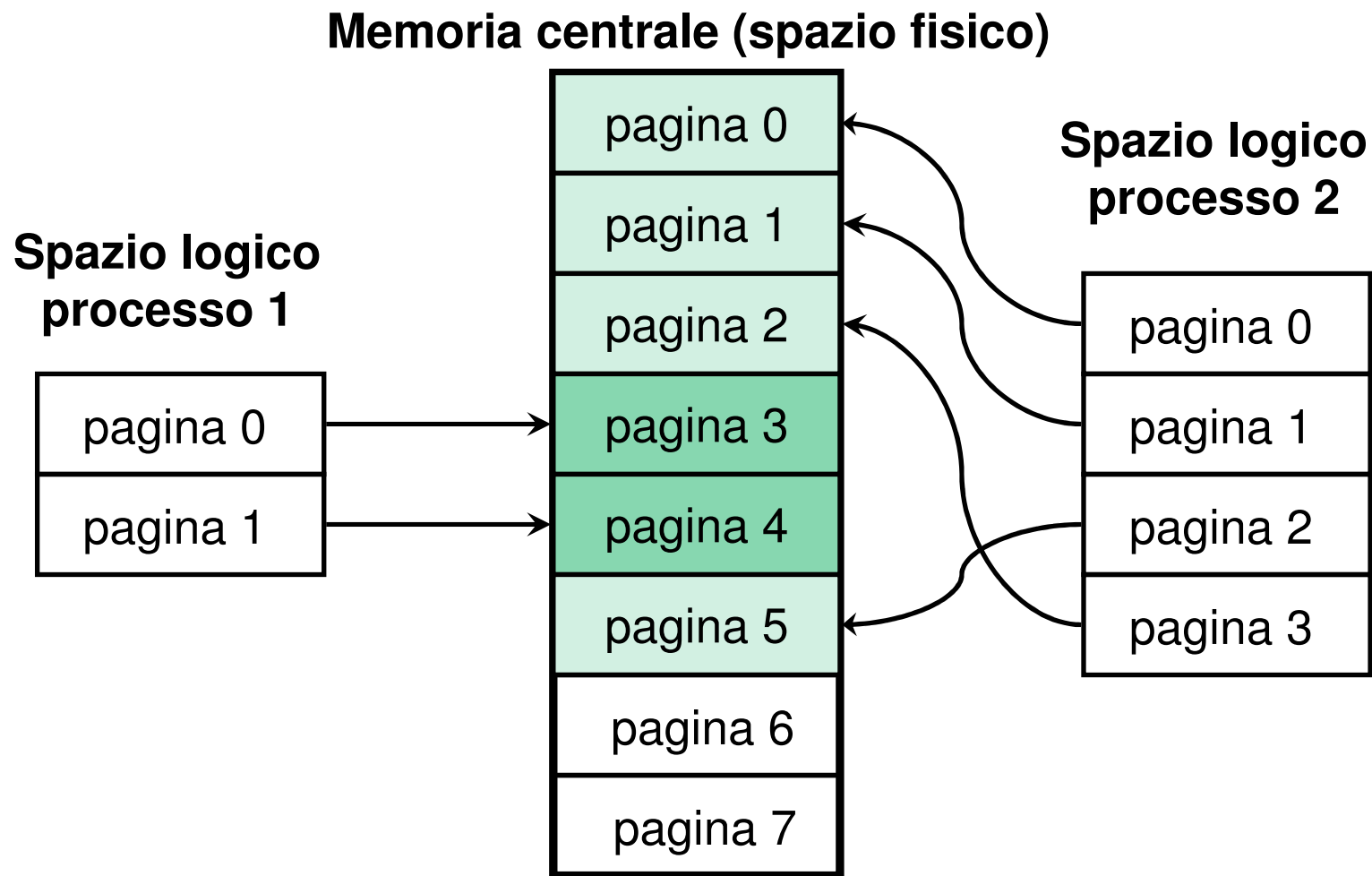
Swapping

- Nonostante le diverse politiche di gestione della memoria, spesso la memoria centrale non è sufficientemente estesa per contenere tutti i programmi concorrenti
- Una soluzione consiste nel trasferire il contenuto di un'area di memoria centrale in un'area della memoria di massa (area di **swap**).
- La memoria di massa è molto più lenta della memoria centrale, quindi lo swap è utile per processi *in attesa* ma non per processi *pronti*.

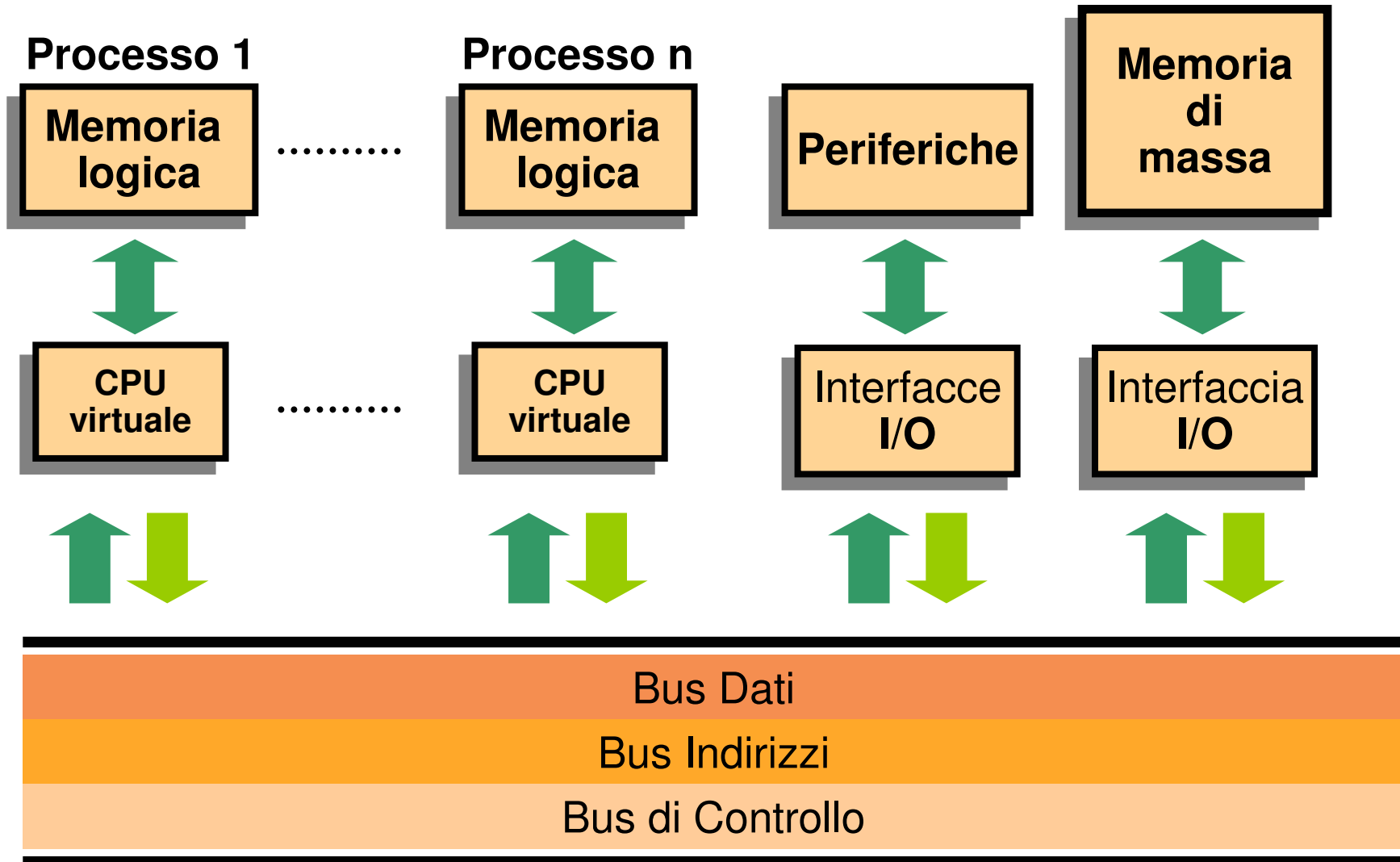
Paginazione

- Un miglioramento nell'efficienza dell'uso della memoria si ha grazie alla **paginazione**, ovvero la suddivisione della memoria in sezioni di dimensioni fisse (**pagine**)
- Si basa sul principio di *località spaziale e temporale*: si possono utilizzare zone di memoria non fisicamente contigue e tenere in memoria centrale solo la porzione di codice che si sta eseguendo.

Pagine logiche e pagine fisiche



Macchina astratta della memoria



Gestione delle periferiche

- Il **gestore delle periferiche** permette la comunicazione tra il calcolatore e tutti i dispositivi esterni ad esso collegati (video, tastiera, stampanti, mouse, ecc.); inoltre:
 - garantisce un comportamento **asincrono** dell'ambiente rispetto al calcolatore e gestisce di accessi **contemporanei** da parte di più periferiche
 - nasconde ai processi il numero (spesso limitato) di risorse HW disponibili (i.e. più stampe su un'unica stampante)
 - non permette ai processi di distinguere tra differenti risorse dello stesso tipo.

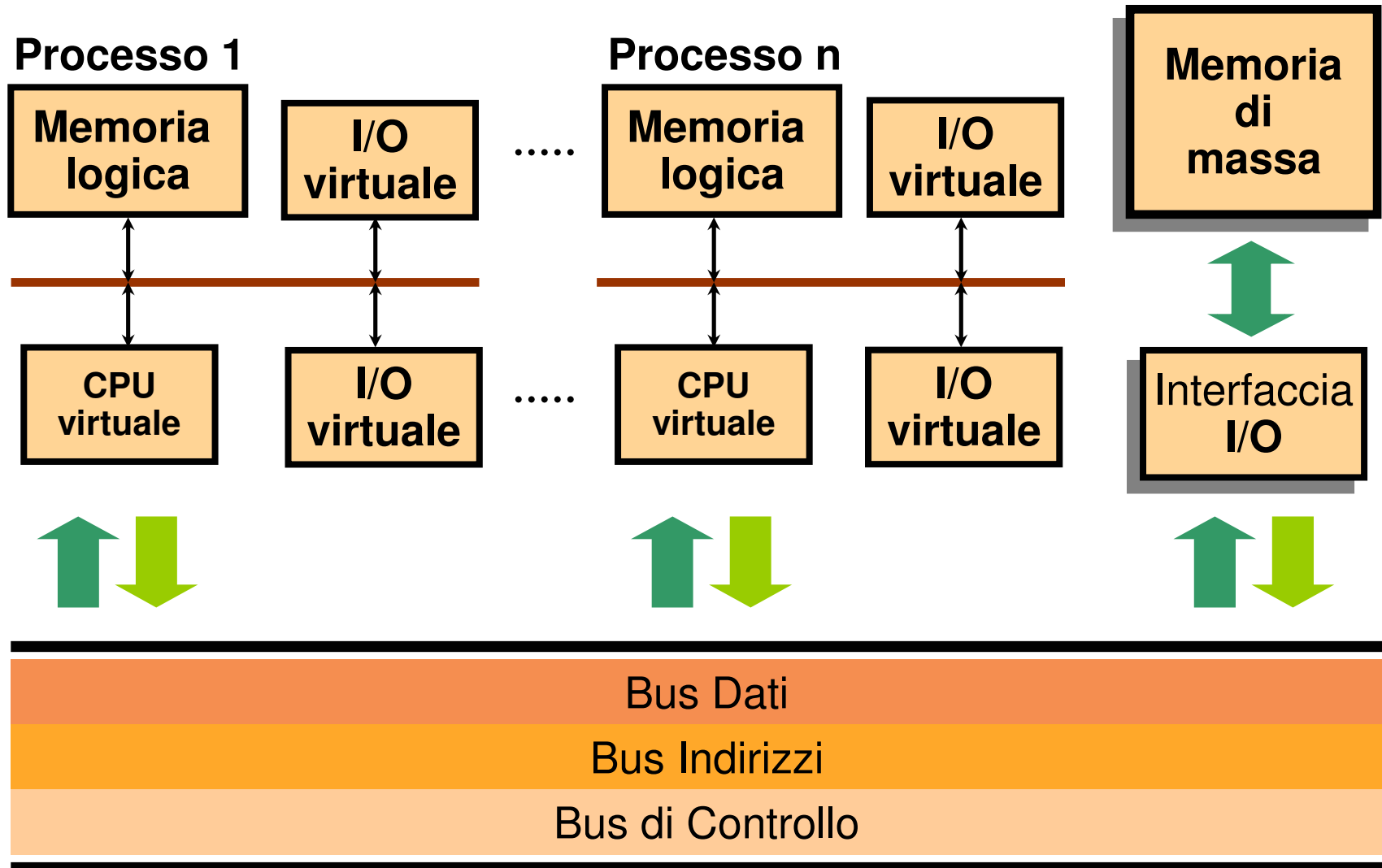
Sistemi Plug & Play

- Nelle versioni più recenti dei sistemi operativi, la necessità di configurare “manualmente” ogni periferica tramite appositi driver viene sostituita da funzioni **Plug&Play**:
 - ad ogni accensione del calcolatore il SO scandisce tutte le risorse HW rilevando quelle non ancora configurate
 - ogni periferica comunica al SO i driver di cui necessita ed il SO installa gli appositi driver senza l'intervento dell'utente.

Lo Spooling

- La tecnica di **spooling** è utilizzata dai driver per rendere **virtuali** più periferiche non condivisibili; ad esempio, nel caso di *una stampante e più processi* che intendono stampare:
 - ogni processo invia il file da stampare al driver della stampante, che lo mette **in coda** nella **directory di spooling**
 - i file in coda vengono stampati secondo l'ordine di arrivo
 - a directory di spooling vuota il driver rimane in memoria in attesa di una nuova richiesta di stampa.
- In questo modo i processi evitano lunghe attese ed operano indipendentemente dalla periferica.

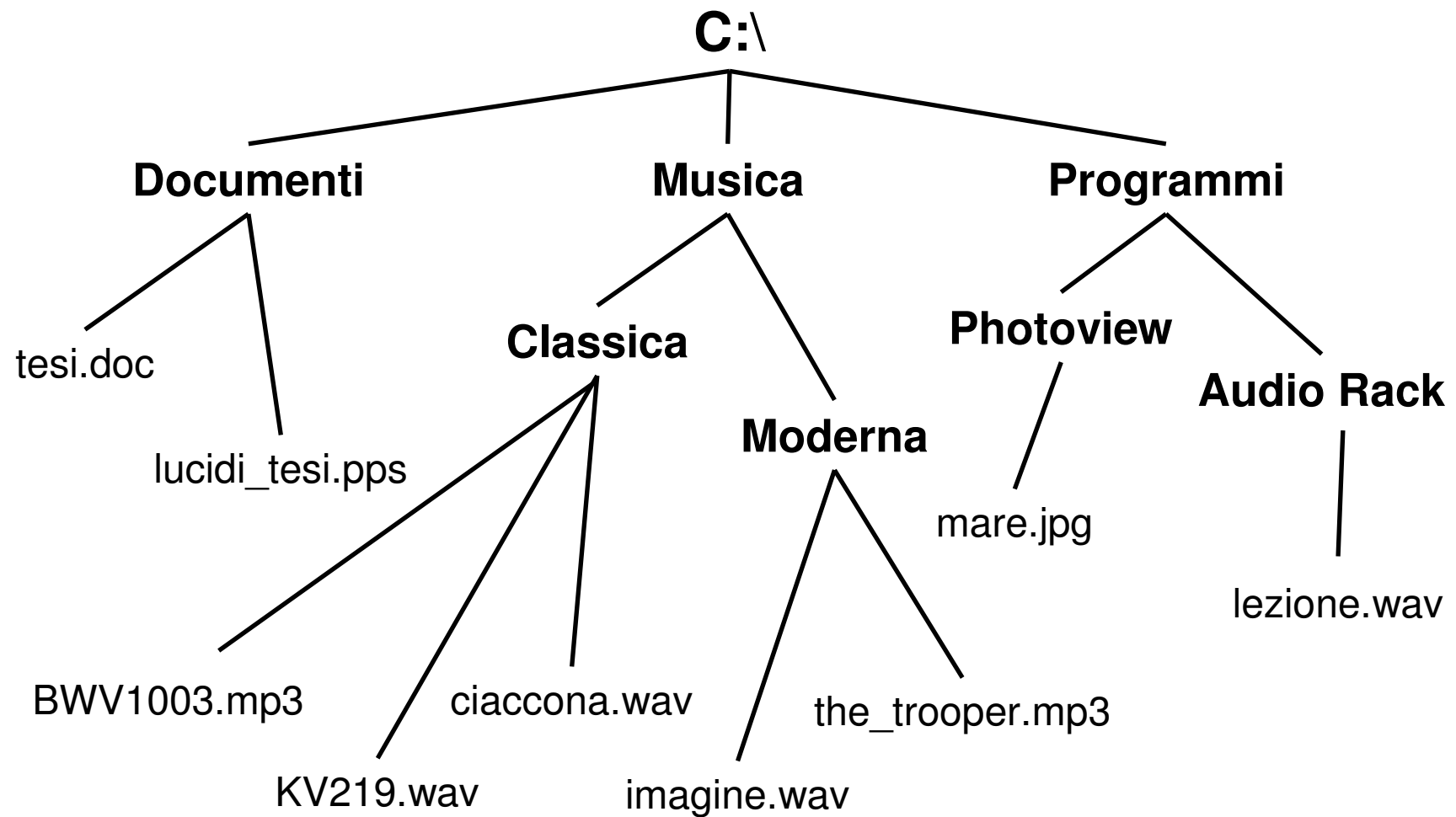
Macchina astratta dell'I/O



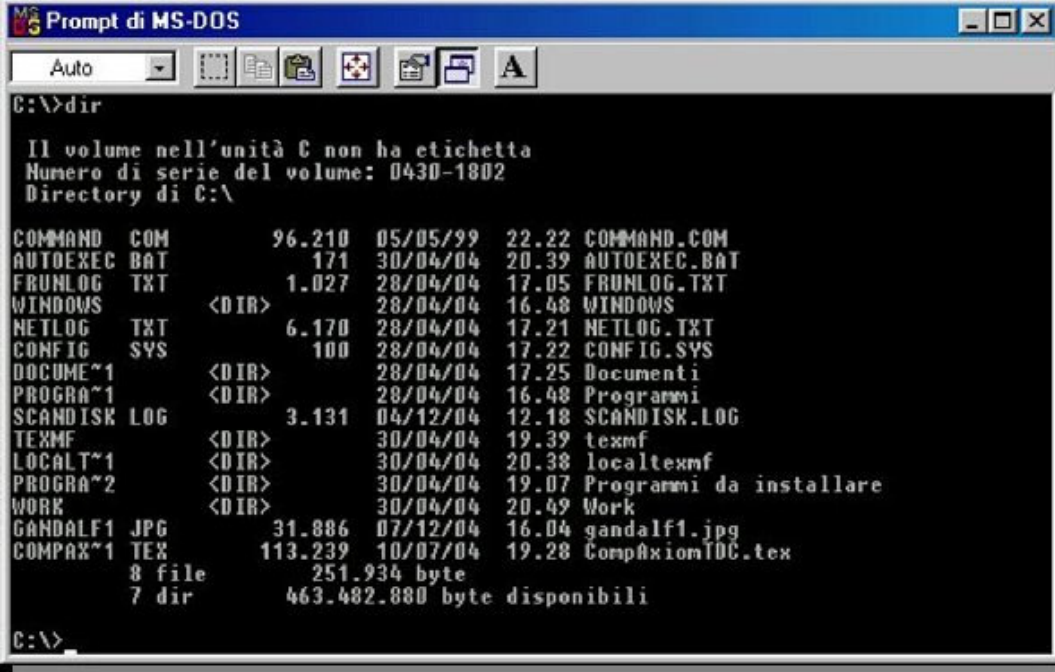
Memoria di massa e file system

- Gestire la memoria di massa significa **organizzare logicamente** i dati e le possibili operazioni su di essi:
 - **recupero, cancellazione o modifica** di dati memorizzati
 - **copia** di dati per backup o **trasferimento** da supporto a supporto.
- Il **File System** rappresenta l'organizzazione logica dei dati in memoria di massa (stabile)
- Entità atomica costituente ogni file system è il **file**, costituito da identificatore (nome.estensione), indicatore di posizione (i.e. C:\Documenti**tesi.doc**), data di creazione, dimensione ecc.
- File e **directory** sono organizzati secondo una struttura ad **albero** che ne rende semplice la localizzazione.

Struttura logica di un file system



File system: Interfaccia di testo



```
MS-DOS Prompt di MS-DOS
Auto
C:\>dir
Il volume nell'unit  C non ha etichetta
Numero di serie del volume: 0430-1802
Directory di C:\
COMMAND COM          96.210  05/05/99  22.22  COMMAND.COM
AUTOEXEC BAT         171    30/04/04  20.39  AUTOEXEC.BAT
FRUNLOG TXT          1.027  28/04/04  17.05  FRUNLOG.TXT
WINDOWS <DIR>          28/04/04  16.48  WINDOWS
NETLOG TXT           6.170  28/04/04  17.21  NETLOG.TXT
CONFIG SYS           100    28/04/04  17.22  CONFIG.SYS
DOCUME~1 <DIR>      28/04/04  17.25  Documenti
PROGRA~1 <DIR>      28/04/04  16.48  Programmi
SCANDISK LOG         3.131  04/12/04  12.18  SCANDISK.LOG
TEXMF <DIR>         30/04/04  19.39  texmf
LOCALT~1 <DIR>      30/04/04  20.38  localtexmf
PROGRA~2 <DIR>      30/04/04  19.07  Programmi da installare
WORK <DIR>          30/04/04  20.49  Work
GANDALF1 JPG         31.886  07/12/04  16.04  gandalf1.jpg
COMPAX~1 TEX        113.239  10/07/04  19.28  CompaxiomTDC.tex
      8 file          251.934 byte
      7 dir           463.482.880 byte disponibili
C:\>
```

File system: Interfaccia grafica



Evoluzione dei sistemi operativi

- I primi calcolatori non prevedevano l'uso di sistemi operativi, ed erano direttamente programmati in linguaggio macchina.
- Con l'aumento della complessità degli elaboratori e del codice da implementare si è reso necessario introdurre un intermediario tra HW e SW che facilitasse la gestione del lavoro.
- Uno dei primi SO fu **OS/360**.
- Con l'introduzione della multiprogrammazione furono progettati sistemi come **CTSS** e **MULTICS**, che spianarono la strada alla nascita di **UNIX** uno dei più popolari ed efficienti sistemi operativi.

Evoluzione dei sistemi operativi

- In seguito nacque l'**MS-DOS**, fin troppo “ispirato” a UNIX, ma particolarmente semplice ed adatto ai personal computer.
- Una nota particolare merita **Linux**, un discendente diretto di UNIX ed apparso per la prima volta nel 1991.
- Uno dei suoi pregi è di essere completamente gratuito ed “open source”, ovvero qualsiasi utente può modificarne il codice sorgente.
- Tuttavia i sistemi che hanno registrato il maggior successo negli ultimi anni sono **Macintosh** e **Windows**, grazie alla loro estrema semplicità e chiarezza dell'interfaccia grafica.

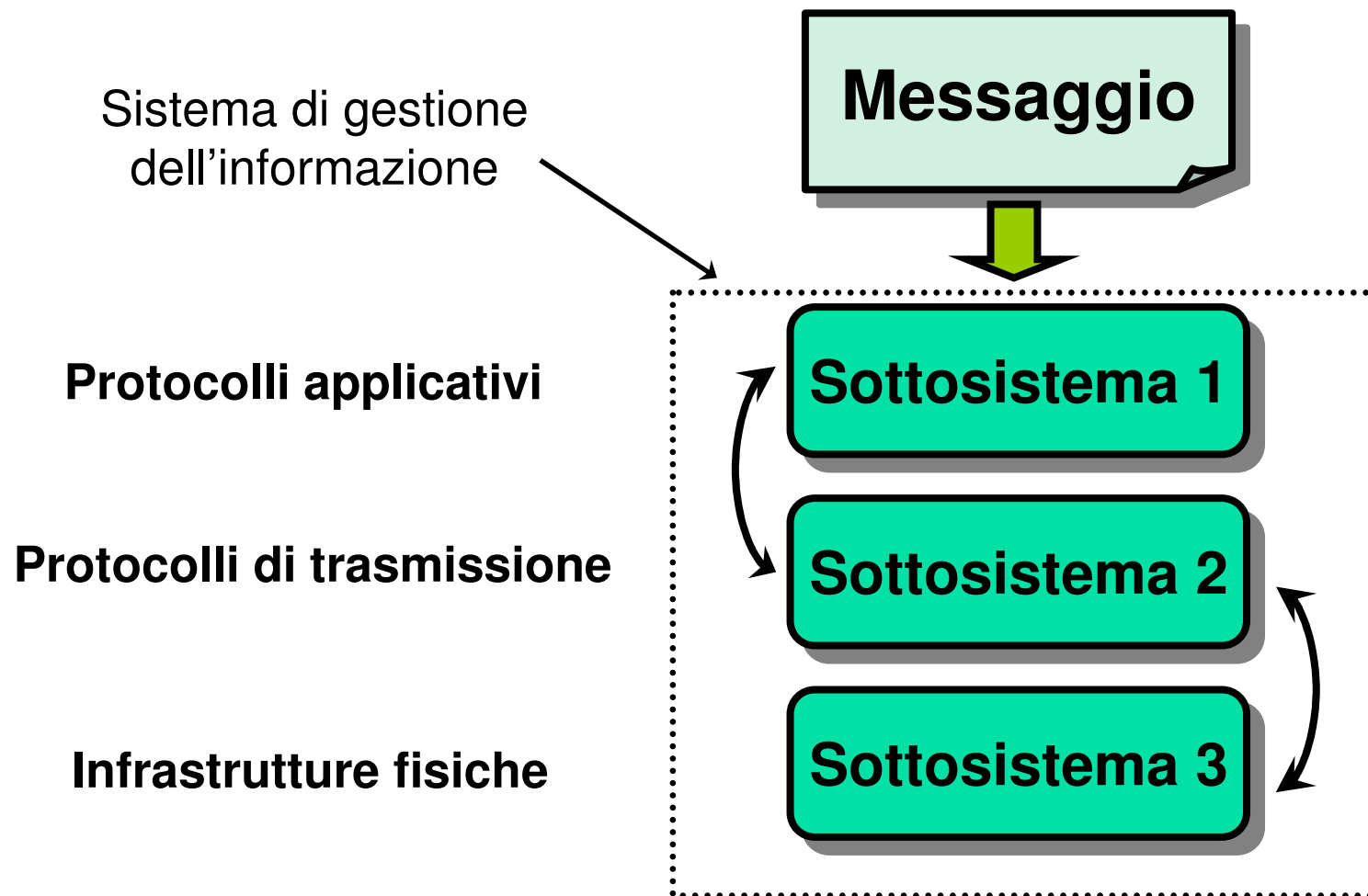


La comunicazione

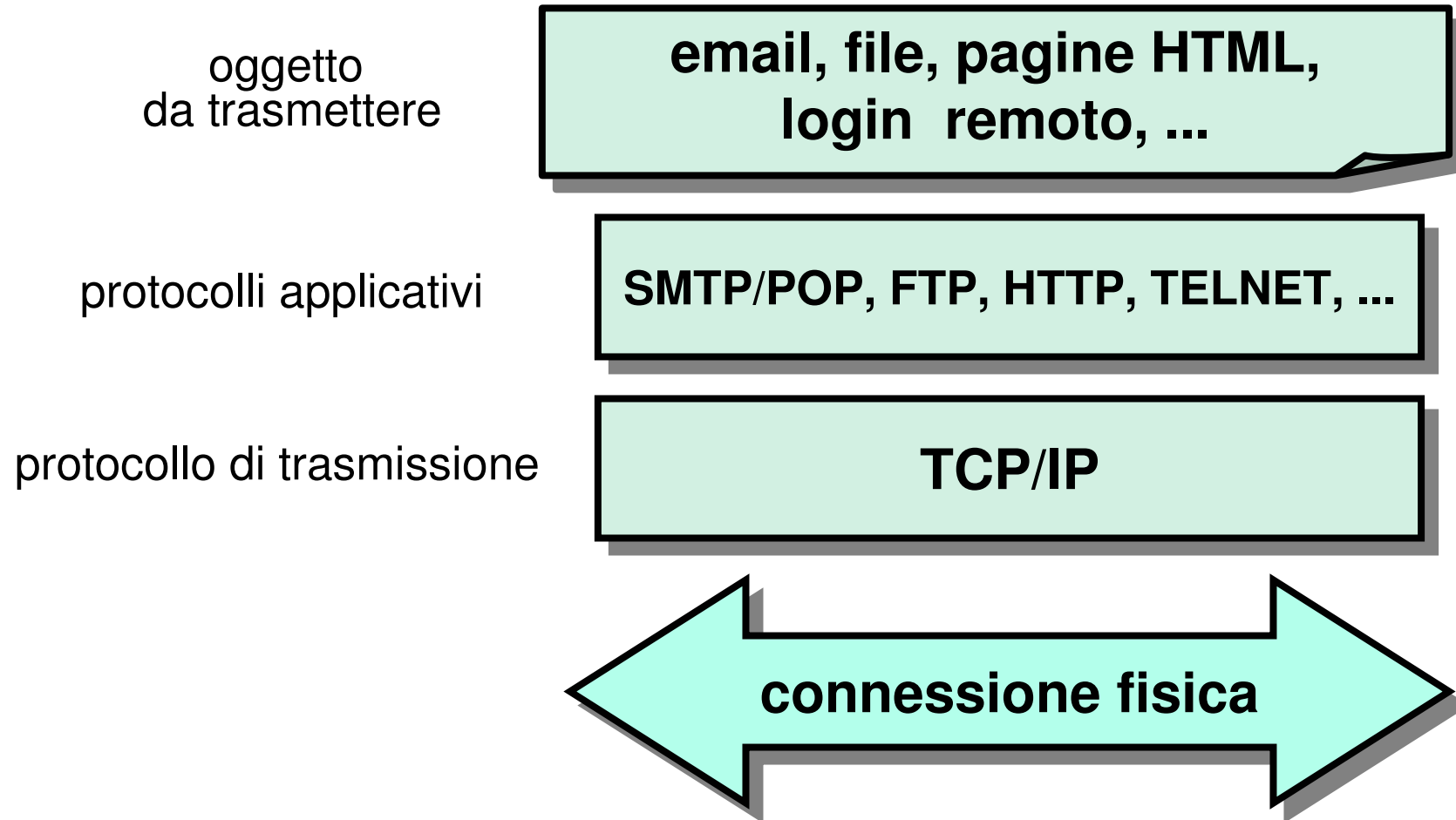
Comunicazione e protocolli

- Oltre all'elaborazione e memorizzazione di dati, un sistema di calcolo deve essere in grado di **comunicare** con altri sistemi
- Le **reti** rappresentano *il mezzo* di scambio di informazioni, ma devono esistere procedure (**protocolli**) che gestiscano le interazioni tra sistemi collegati tra loro
- Ogni protocollo deve indicare il formato dei dati da trasferire, la struttura dei **pacchetti**, la velocità ecc., ed è dedicato ad un particolare aspetto della comunicazione
- I protocolli sono organizzati in differenti **livelli** di astrazione della comunicazione indipendenti e comunicanti.

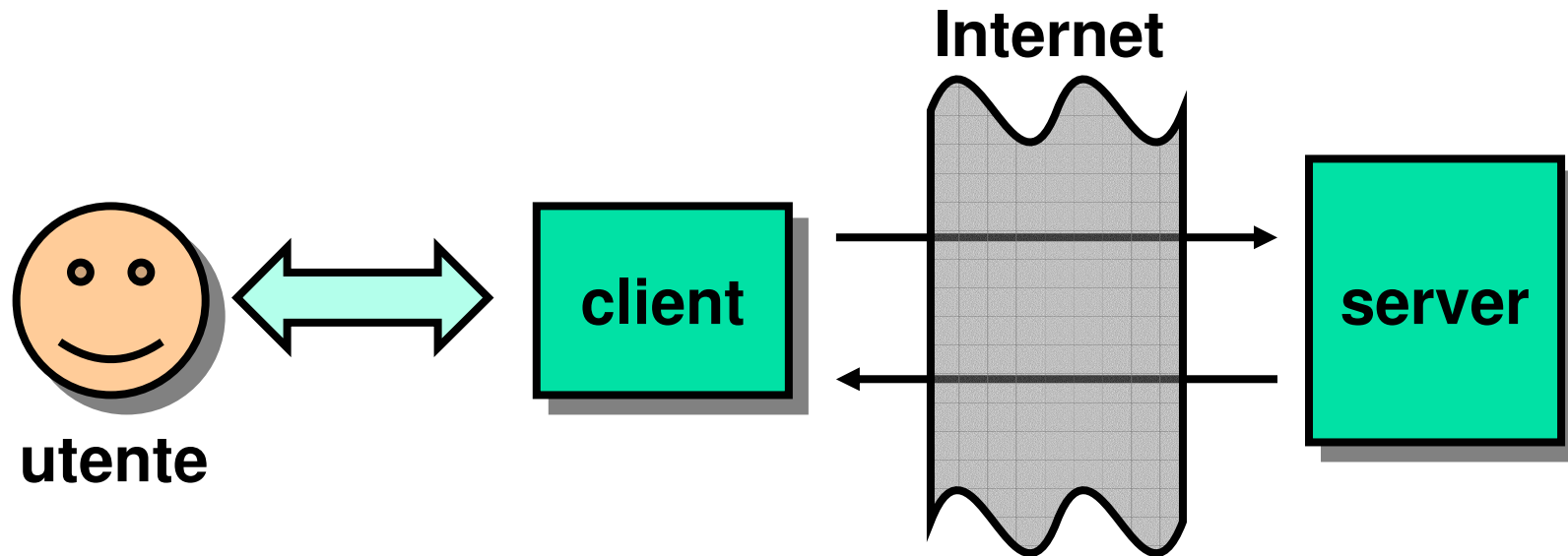
Sistema di comunicazione



Livelli di Internet



Comunicazione client-server



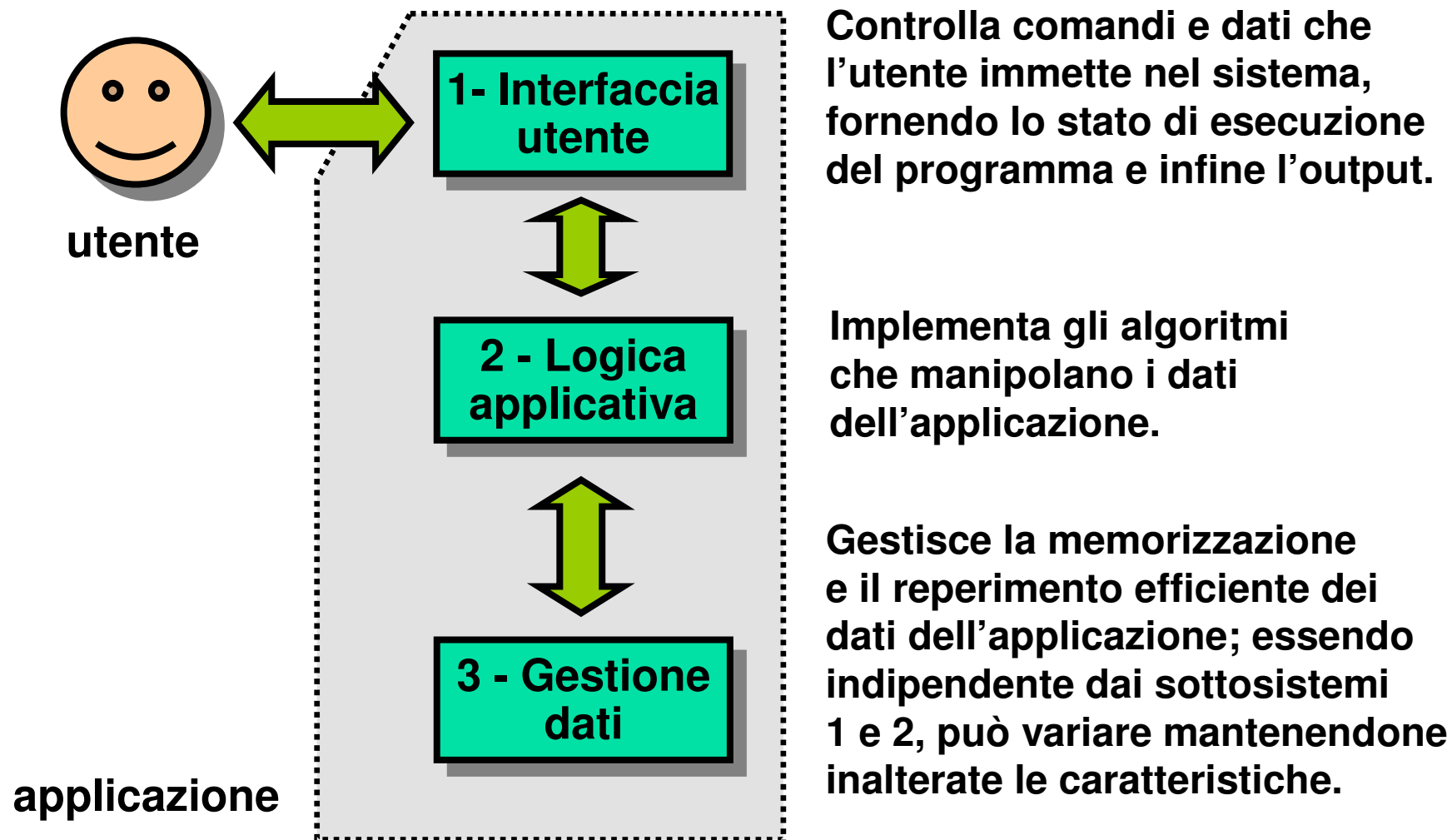
- L'utente comunica la sua richiesta al client
- Il client si collega al server e trasmette la richiesta
- Il server risponde al client
- Il client fornisce la risposta all'utente

Indirizzi IP e DNS

- In Internet, ogni **host** e **router** è *univocamente rappresentato* da un codice di 32 bit, detto **indirizzo IP**
- L'IP non è di semplice comprensione da parte dell'utente, ed è quindi uso comune assegnare ad ogni IP un **nome** simbolico
- Per fare questo si utilizza il **Domain Name System (DNS)**, che associa uno o più nomi ad ogni IP, e gestisce la conversione tra le due codifiche.

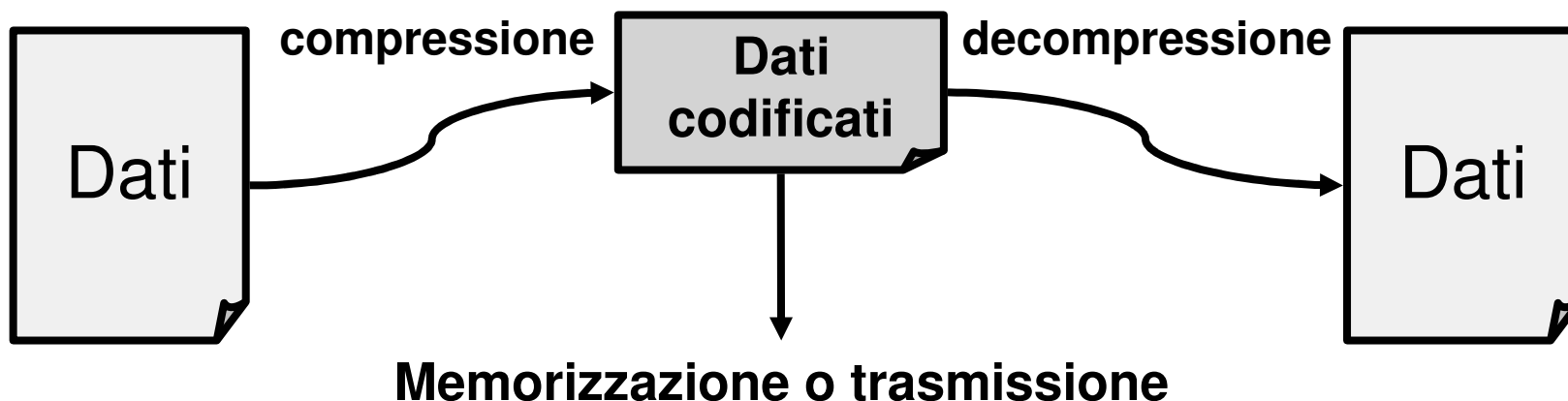


Applicazioni informatiche



Compressione dei dati

- Spesso i dati devono essere memorizzati su supporti di dimensione non sufficiente oppure trasmessi attraverso mezzi di capacità ridotta.
- È necessario **comprimerli** (e.g. prima di essere trasmessi via modem, o su supporti stabili tramite **zip** o altro).



Sicurezza



La sicurezza in rete

- Lo sviluppo delle reti su scala mondiale rende necessario studiare la **sicurezza** dei dati che vi circolano
- Le quattro principali aree in ambito di sicurezza in rete sono:
 - **segretezza dei dati:** garantire che utenti non autorizzati possano *leggere o modificare* dati privati (e.g. **crittografia**)
 - **autenticazione:** accertamento dell'identità del proprio interlocutore in rete, per evitare di fornire informazioni riservate ad estranei
 - **firme elettroniche:** accertamento dell'autenticità dell'autore di un determinato documento "firmato"
 - **controllo di integrità:** accertamento che un messaggio ricevuto non sia stato modificato durante la trasmissione.

Crittografia

- La **crittografia** è una tecnica antica, che ha assunto piena connotazione *scientifica* nel secolo scorso.
- Essa consiste nel **codificare** un “testo” **in chiaro** utilizzando un **chiave**, che descrive una certa *funzione di codifica*.
- Questa tecnica risulta necessaria ovunque si voglia archiviare o trasmettere dati riservati, rendendo impossibile (o meglio **molto difficile**) l’accesso a chi non dispone della **chiave**.
- “Molto difficile” risulta più corretto che “impossibile”, in quanto la maggior parte dei sistemi di crittografia ritenuti **sicuri** sono stati violati.

Chiavi pubbliche e simmetriche

- Le chiavi **simmetriche** (**segrete**) vengono utilizzate sia per la cifratura che per la decifratura dei dati:

$$\text{Cod}_{\text{Key}}(\text{Msg}) = \text{Cript_Msg} \qquad \text{Decod}_{\text{Key}}(\text{Cript_Msg}) = \text{Msg}$$

- spesso l'imprudente condivisione delle chiavi o il loro "furto" ha compromesso anche i sistemi più sicuri.

- Un metodo di cifratura più sicuro utilizza una unica chiave **pubblica** di codifica, ed una chiave **privata** di decodifica:

$$\begin{aligned} \text{Cod}_{\text{PubKey}}(\text{Msg}) &= \text{Cript_Msg} \\ \text{Decod}_{\text{PvtKey}}(\text{Cript_Msg}) &= \text{Msg}. \end{aligned}$$

I Virus

- In informatica il **virus** è un frammento di codice inserito all'inizio di un normale programma con lo scopo di alterare o distruggere dati, rallentare le prestazioni di sistemi o bloccarli del tutto
- Lo sviluppo di Internet ha favorito ed aumentato la diffusione di migliaia di tipologie di virus
- Per ogni virus “progettato” viene immediatamente codificato un nuovo **antivirus**:
 - programma che contiene grandi liste di virus regolarmente aggiornate e che ispeziona i file sospetti “riconoscendo” ed eliminando l'eventuale codice pericoloso.
- Essendo spesso causa di distruzione o manomissione di dati importanti, la generazione di virus è ritenuta un reato.

Lo Spam



- La SPAM (Spiced Ham) era la carne in scatola fornita ai soldati dell'esercito americano, e si guadagnò una fama negativa.
- In Internet lo "Spamming" consiste nell'invio di messaggi pubblicitari tramite posta elettronica **in nessun modo sollecitati** ("Junk Mail").
- Lo Spamming danneggia il ricevente facendogli perdere tempo e denaro per scaricare posta inutile, e danneggia il gestore del server di posta con uno smisurato aumento di "traffico" nelle sue linee.

Evitare lo Spam

- Evitare messaggi di spam è molto difficile: alcuni metodi sono troppo blandi mentre altri troppo restrittivi (fermano anche posta “legittima”).
- Tuttavia esistono norme pratiche che consentono di ridurre questo problema:
 - evitare di comunicare **pubblicamente** il proprio indirizzo email attraverso siti, guestbook, chat, messenger, ICQ, ecc.
 - spesso gli spammers reperiscono **automaticamente** gli indirizzi, cercando stringhe della forma “tizio@caio.ecc” ; quindi, se desideriamo comunicare la nostra email in un contesto **pubblico** è buona norma scriverla nella forma:

tizioNOSPAM@caio.ecc