

ARCHITETTURA SOFTWARE DEI SISTEMI DISTRIBUITI

Testi di riferimento:

Amjad Umar, "Client/Server Internet Environments", Prentice Hall, 1997

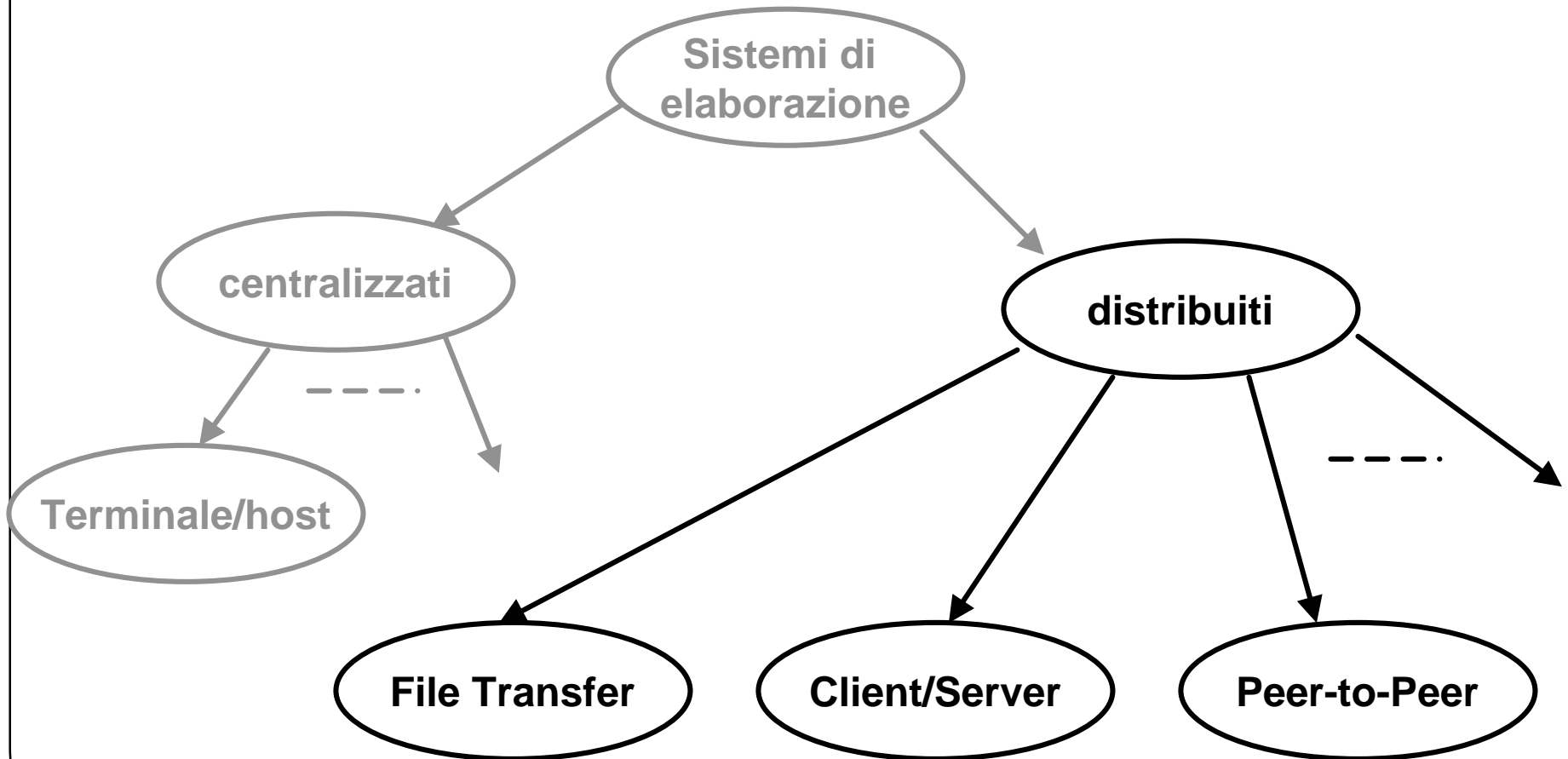
Definizione di Sistema Distribuito (DCS)

- **Sistema di elaborazione distribuito (DCS - distributed computing system):**
 - sistema costituito fisicamente da una collezione di calcolatori **autonomi** interconnessi tramite una rete
- ⇒ L'elaborazione è distribuita
- ⇒ I sistemi basati sul modello Host-Terminale (con terminali non intelligenti) non rientrano in questa categoria

Architetture Software

- L'architettura software di un sistema distribuito
 - è basata sul modello di riferimento a livelli (OSI)
 - prevede un insieme di processi
 - » **eseguiti fisicamente sui diversi calcolatori del sistema**
 - » **che interagiscono tra loro secondo un certo modello**
- I livelli di interesse per il programmatore sono quelli applicativi
- È possibile classificare le architetture in base ai **modelli di interazione** usati

Classificazione dei Sistemi Distribuiti (in base al modello di interazione)

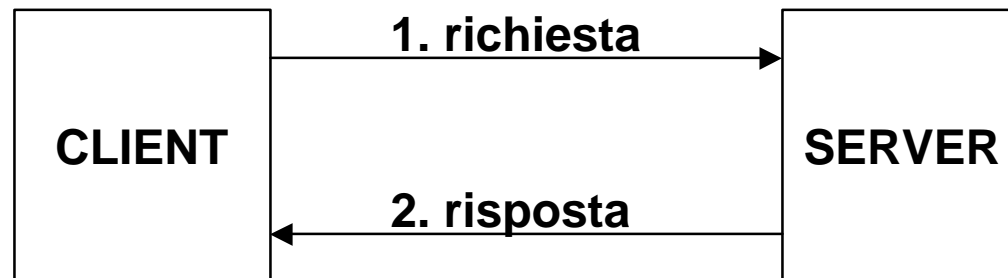


Il Modello File-Transfer

- Le uniche interazioni tra i calcolatori del sistema sono trasferimenti di file
- L'accoppiamento tra calcolatori diversi è minimo
- Esempi:
 - sistemi di e-mail

Il Modello Client-Server (C/S)

- È il modello attualmente più usato
- Ogni interazione avviene tra 2 processi: uno svolge il ruolo di **cliente**, l'altro quello di **servitore**.
- L'interazione è basata su uno scambio di **messaggi**: il client invia la **richiesta** ed il server la **risposta**



Il Modello Client-Server (C/S)

- Il ruolo (client o server) si riferisce alla singola interazione, non al processo:
 - Un processo può anche svolgere in alcune interazioni ruolo client e in altre ruolo server

Il Modello Peer-to-peer (P2P)

- Ogni interazione avviene tra 2 processi, ma in modo simmetrico
- L'interazione si basa su uno scambio di messaggi, ma ciascuno dei due processi può prendere l'iniziativa di inviarli

Principali Particolarità del Software Distribuito

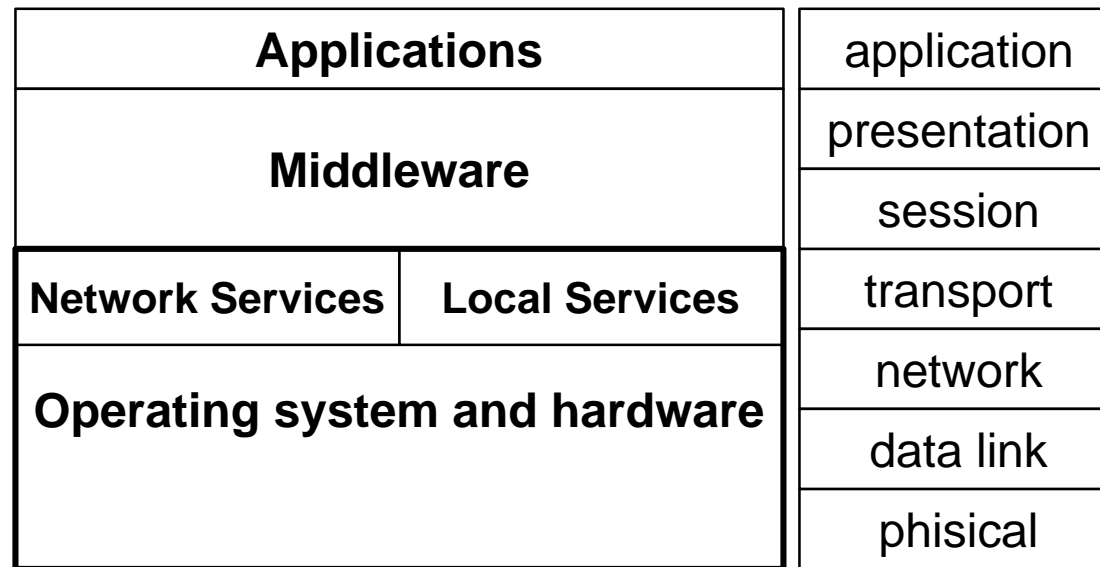
- Il software distribuito è sempre **concorrente** (i processi girano su CPU diverse)
 - occorre risolvere i problemi tipici della concorrenza (sincronizzazione, coordinamento, ecc.)
- **La comunicazione** tra componenti software residenti su host diversi **incide sulle prestazioni e può fallire.**
 - occorre tenerne conto nello sviluppo del software
- È anche possibile il **crash parziale** (di singoli calcolatori del sistema) o la **caduta delle connessioni** di rete
 - occorre adottare strategie di fault tolerance

Principali Particolarità del Software Distribuito

- Il software distribuito gira su **piattaforme eterogenee**
 - occorre risolvere i problemi dell'eterogeneità di hw e s.o.
- Il software distribuito è per sua natura più esposto ad attacchi di **security**
- Il software distribuito può avere la necessità di **localizzare** processi all'interno del sistema

Il Middleware

- I problemi di base comuni al software distribuito vengono risolti da uno strato detto middleware, che sta tra le applicazioni vere e proprie e il S.O.:



Il Middleware

- fornisce servizi **business-unaware** per il coordinamento di e la comunicazione tra processi (o utenti) remoti
- maschera la presenza della rete, l'eterogeneità dei computer, i problemi di security, ecc.
- Esempi di software classificabile come middleware:
 - web browsers
 - database drivers
- Esempi di software non classificabile come middleware:
 - airline reservation system (business aware)

Tipici Servizi del Middleware

- **Servizi di interazione:**

- scambio di informazioni, gestione delle connessioni, meccanismi per evitare i deadlock, ecc.

- **Servizi per specifiche categorie di applicazioni:**

- accesso a database, transaction processing, distributed object management

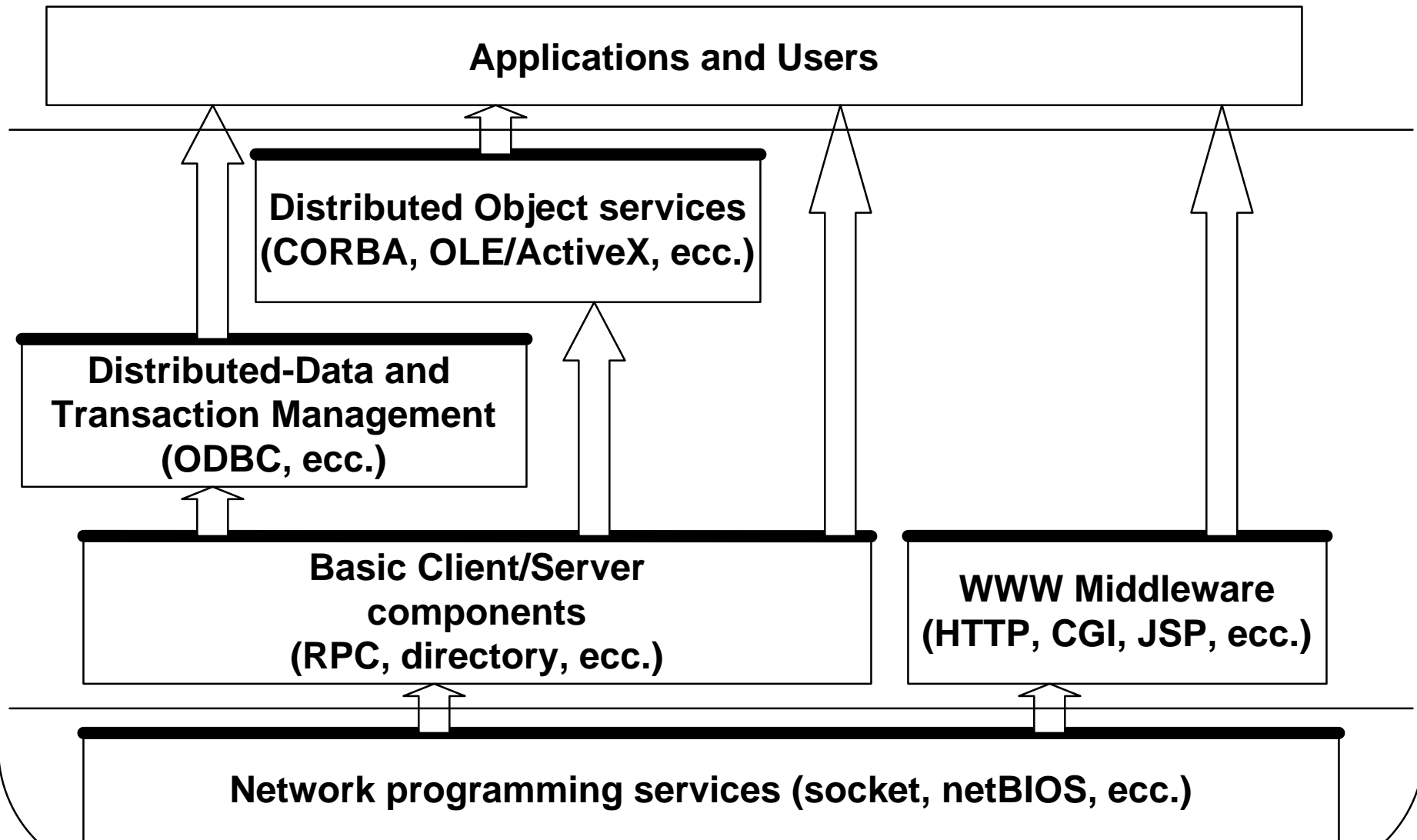
- **Servizi di gestione, controllo, amministrazione:**

- directory, security, monitoraggio delle prestazioni, ecc.

Organizzazione del Middleware: i Componenti

- Il middleware può essere organizzato come un insieme di **componenti**, ciascuno dei quali fornisce determinati servizi ed è accessibile attraverso determinate **API**
- I servizi di un componente possono essere usati da altri componenti, per costruire servizi più complessi
- I servizi base usati da tutto il middleware sono i servizi di connettività forniti dai livelli rete e trasporto.
- Per loro natura i componenti si prestano bene ad essere trattati secondo il paradigma ad oggetti.

Esempi di Componenti Middleware



Organizzazione del Middleware: gli Ambienti

- Componenti di vario tipo possono essere combinati in **ambienti** middleware, che
 - forniscono una molteplicità di servizi di vario tipo (tutti i principali tipi)
 - possono includere strumenti di sviluppo appositi
- Esempi di ambienti:
 - OSF DCE
 - Sun J2EE
 - Microsoft .NET

Apertura del Middleware

- Il middleware può essere più o meno **open** (aperto all'integrazione di componenti di produttori diversi)
- Si possono distinguere due aspetti indipendenti:
 - l'apertura dei protocolli utilizzati
 - l'apertura delle API

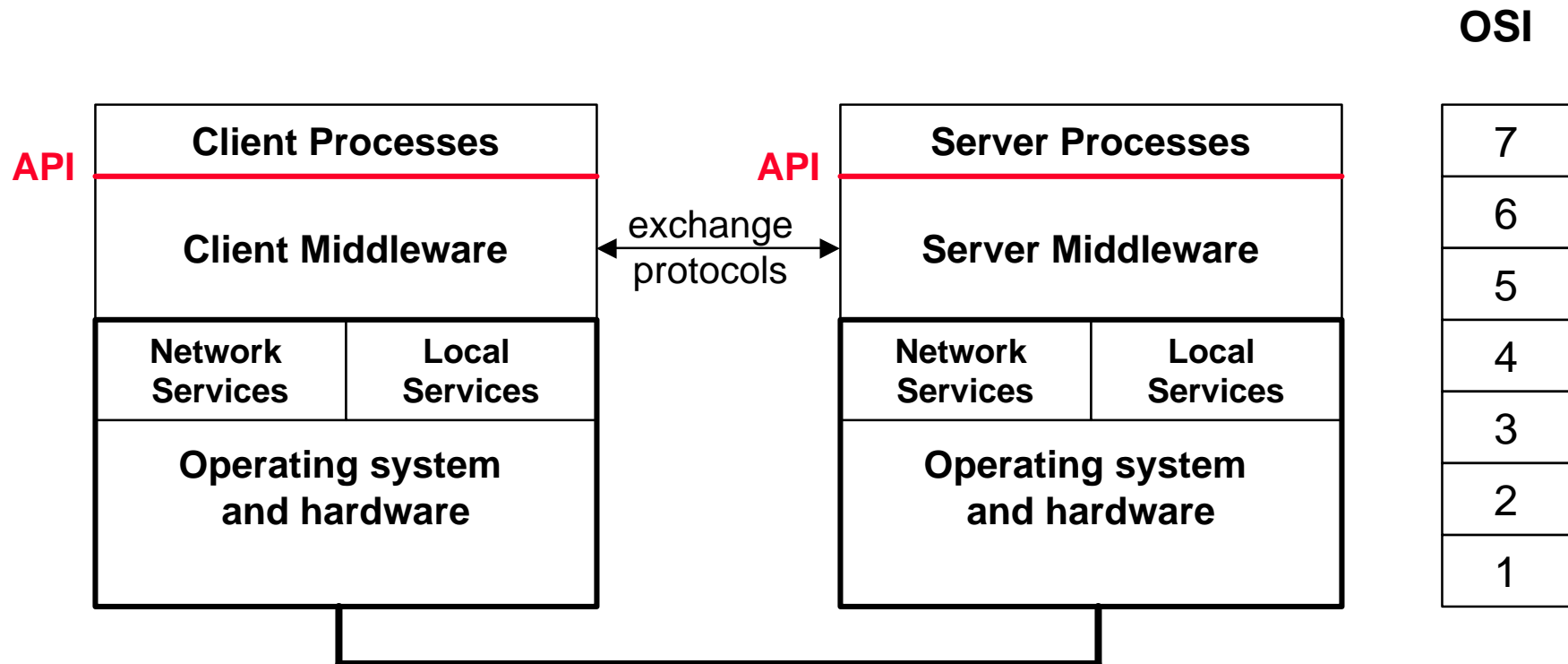
Evoluzione e Tendenze Attuali

- I sistemi distribuiti attuali (quelli che studieremo) sono i cosiddetti sistemi OCSI:
 - object-oriented
 - client/server
 - internet based

Le Generazioni di Sistemi C/S

- I generazione (1G) – anni 80-90
 - singole applicazioni per fornire servizi di stampa, accesso a file condivisi, database, ecc (print/file/DB-server). Middleware quasi inesistente.
- II generazione (2G) – metà anni 90
 - sistemi OCSI, maggior interoperabilità, componenti
- III generazione (3G) – anni 2000
 - ulteriore sviluppo dei componenti e degli ambienti (che includono strumenti per lo sviluppo di applicazioni)

Anatomia del Software C/S



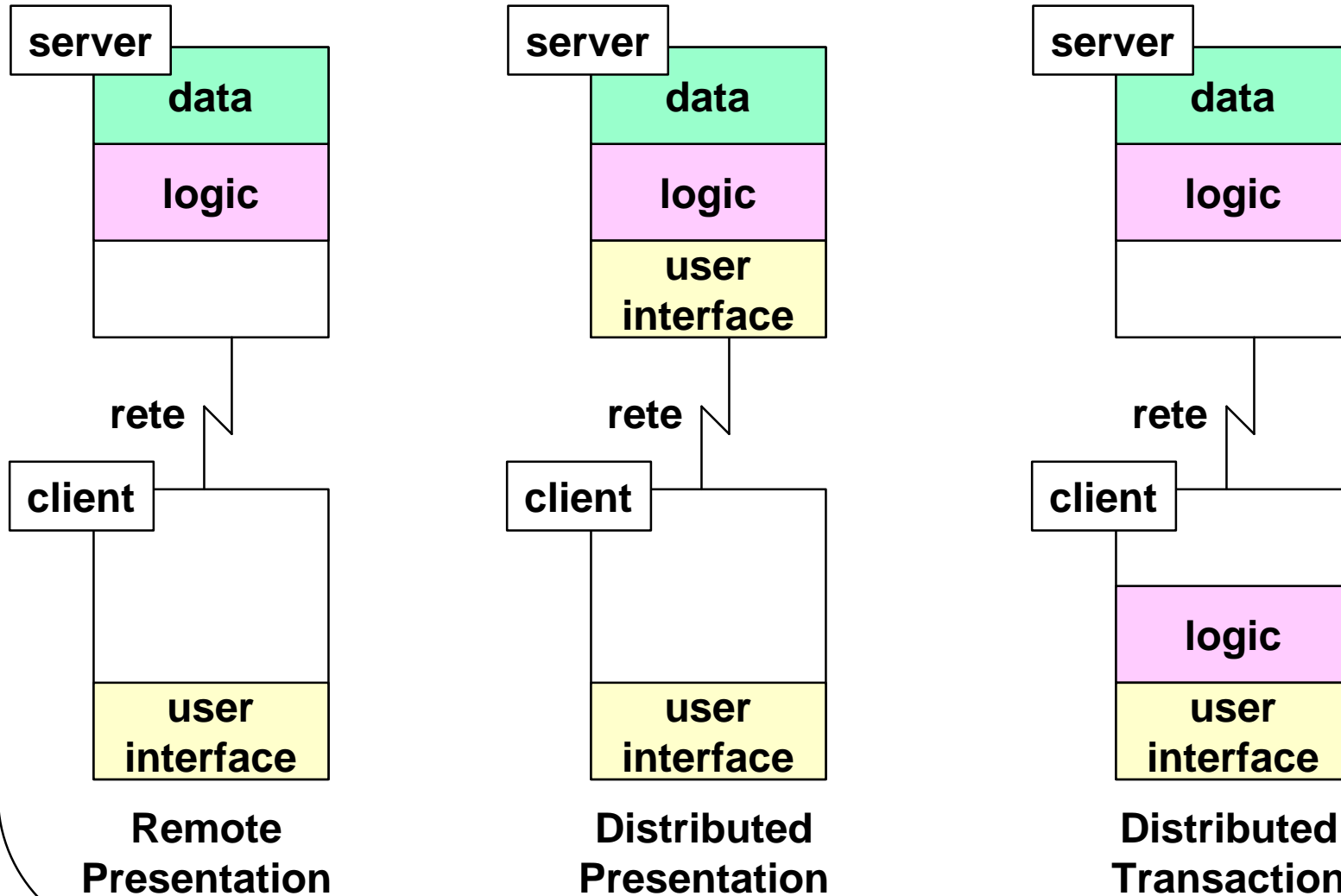
Application Tier e Physical Tier

- Ogni applicazione può essere divisa **logicamente** in tre componenti principali (tier logici):
 - interfaccia utente
 - elaborazione
 - dati
- In un sistema distribuito i tier logici dell'applicazione vengono mappati su più processi che girano sui calcolatori del sistema (tier fisici)
- La mappatura tier logici/ tier fisici può essere fatta in diversi modi

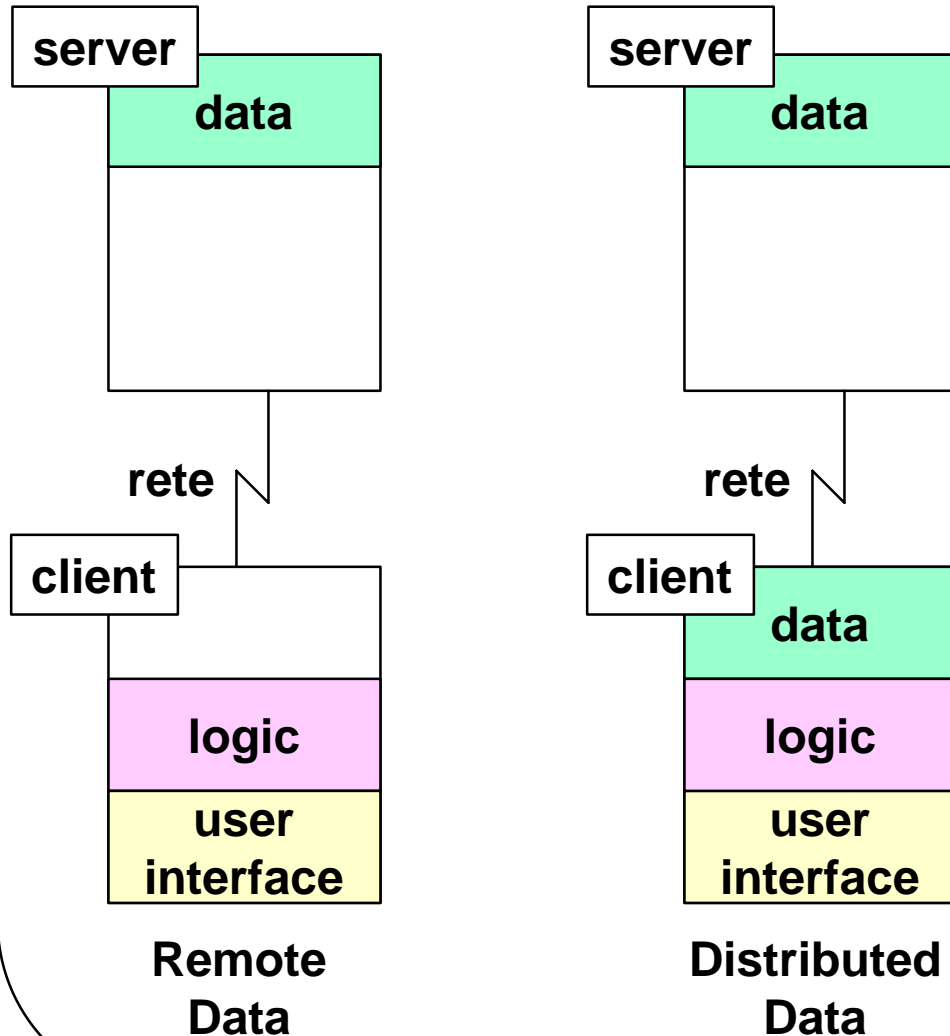
Architetture C/S 2-tier

- Prevedono solo 2 tier fisici: uno client (detto anche front end) ed uno server (detto anche back end)
- È l'architettura classica, tipica della prima generazione di sistemi C/S.
- A seconda di come si distribuiscono i tier logici su quelli fisici, si hanno diverse configurazioni

Classificazione delle Architetture C/S 2-tier (Gartner Group)



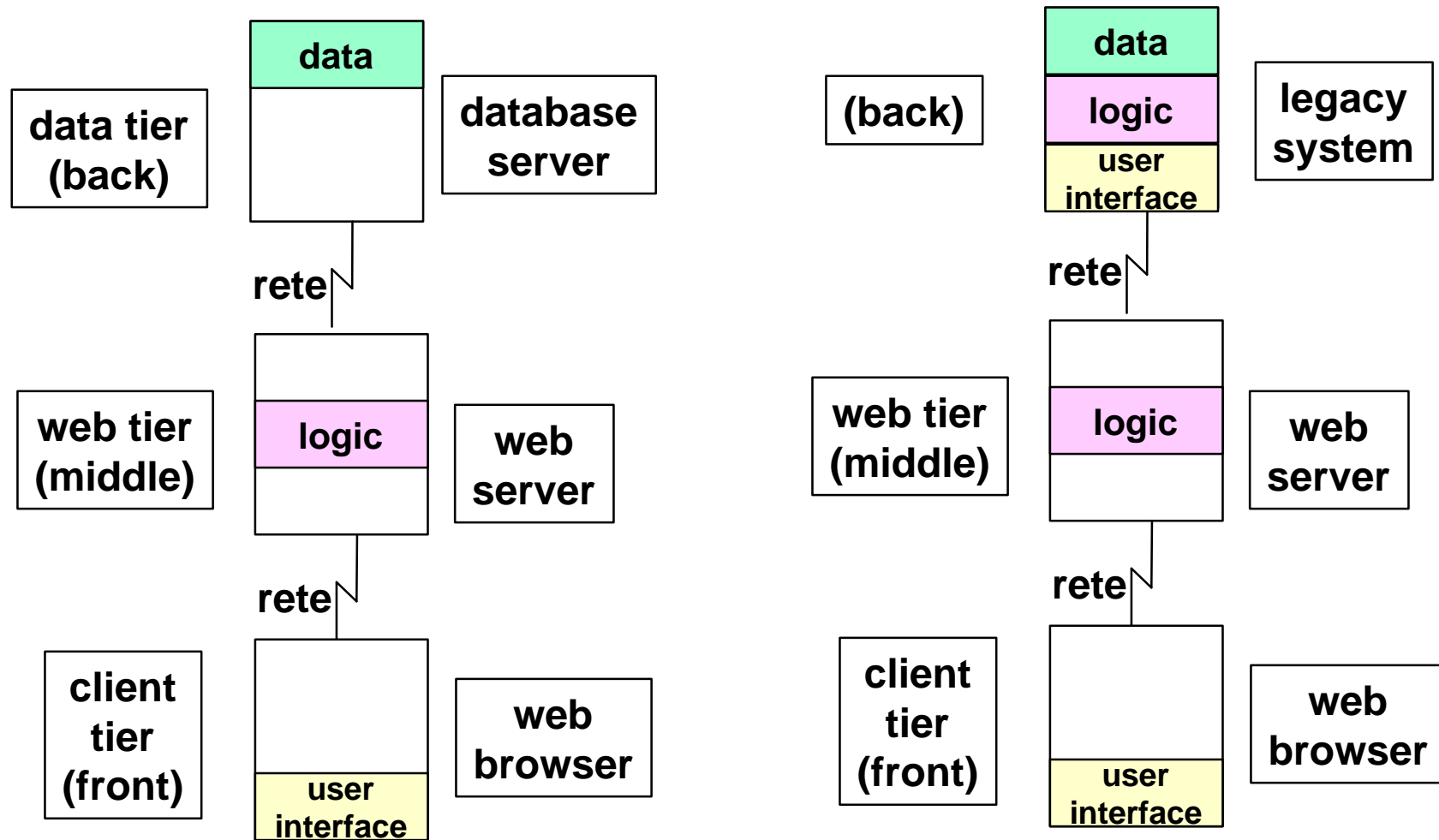
Classificazione delle Architetture C/S 2-tier (Gartner Group)



Architetture C/S 3-tier

- Le architetture C/S 2-tier hanno limiti di **scalabilità** e di **flessibilità**
- Per superare questi limiti si può inserire un tier fisico intermedio (middle machine), con varie possibili funzioni:
 - distribuzione del carico su più back end
 - filtro (p. es. firewall)
 - conversione di protocollo, accesso a legacy systems (gateway)
 - accesso a più server simultaneamente, con servizi a valore aggiunto

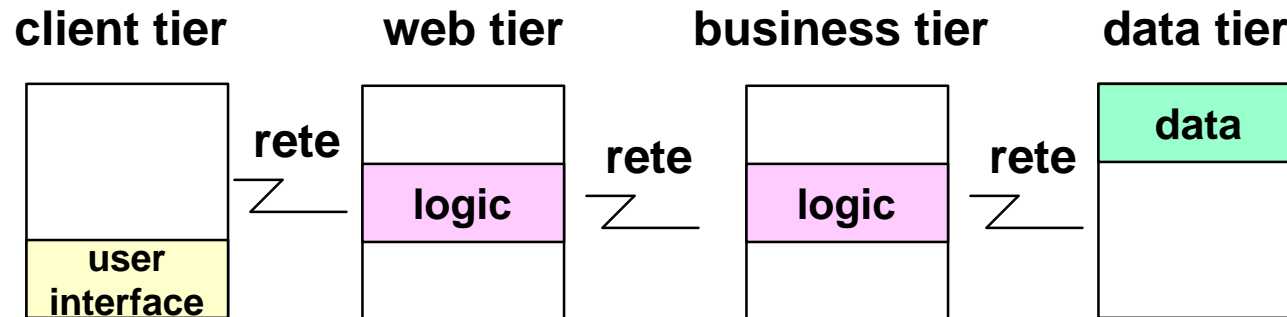
Esempi di architetture C/S 3-tier



Web gateway

Architetture C/S multi-tier

- Per aumentare ulteriormente la flessibilità e per sistemi molto complessi, può essere utile aumentare ulteriormente il numero dei tier fisici
- Esempio di architettura 4-tier:



Diversi tipi di API

- **Interfaccia procedurale:** è costituita da un insieme di procedure che realizzano le funzionalità richieste (per esempio aprire una connessione). L'interazione è tipicamente sincrona.
- **Interfaccia basata su messaggi:** è costituita da un insieme di messaggi che fruitore e fornitore si scambiano secondo un certo protocollo (interazione asincrona)
- **Interfaccia basata su meccanismi specifici** (p. es. interrupt software)

Cosa Studieremo?

- Alcuni esempi di componenti middleware, le relative API, il loro uso per creare applicazioni distribuite
- I principi dell'object-orientation e la loro applicazione al software distribuito