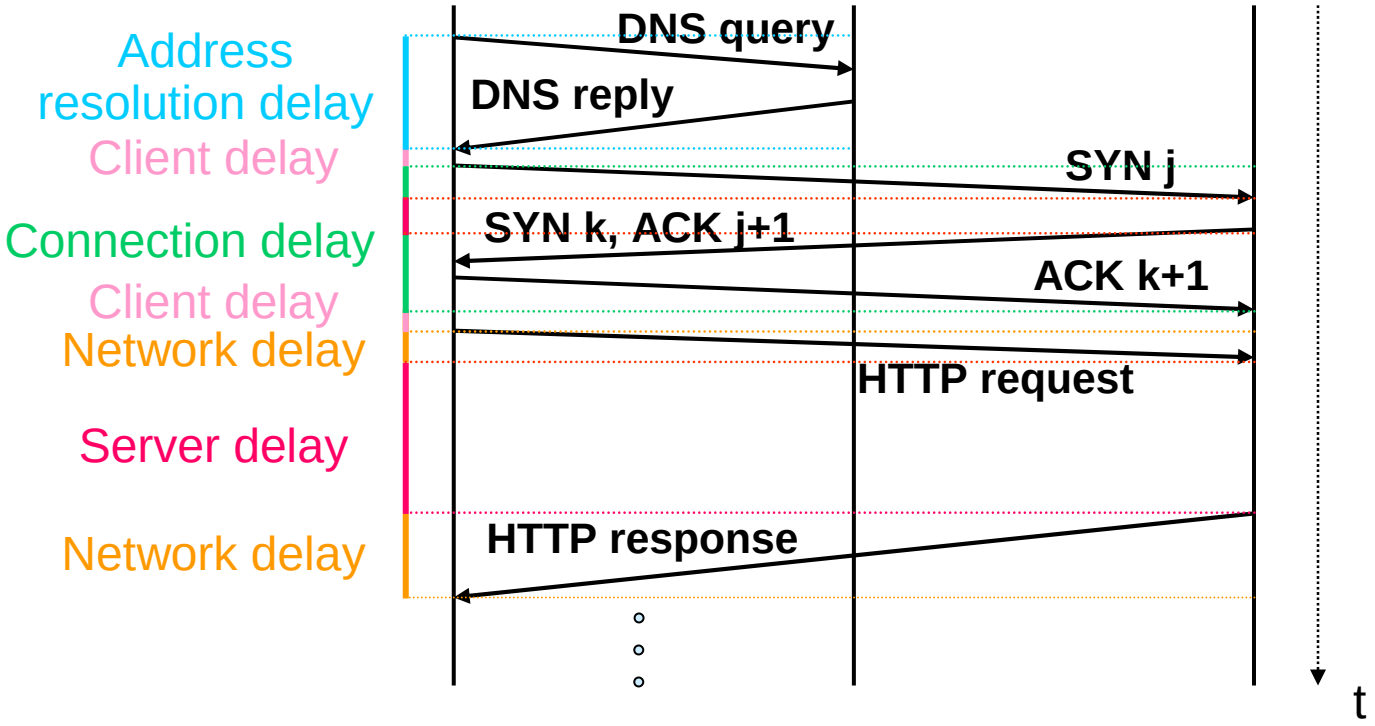
# Sistemi Web distribuiti localmente

Valeria Cardellini
Università di Roma Tor Vergata

# Motivazioni

- Il successo del Web
  - Siti Web popolari sono soggetti a milioni di hit al giorno
  - Es.: il sito Web delle Olimpiadi del 2000 ha ricevuto:
    - 875 milioni di hit nel giorno di picco
    - 1,2 milioni di hit nel minuto di picco
- L'evoluzione dei servizi basati sul Web
  - Servizi sempre più complessi e che richiedono generazione di contenuti dinamici e sicuri
  - Maggiori aspettative da parte degli utenti
    - Regola degli 8 secondi

# Componenti del ritardo Web




*Dov'è il collo di bottiglia?*

| | |
|---|---|
| (1) DNS? | (2) Client/connessione? |
| (3) Rete? | (4) Web server? |

# Potenziamenti del Web

- Azioni a livello del CLIENT
  - Possibilità di intervento limitato
- Azioni a livello di RETE
  - “Network guys are doing an excellent job”
- Azioni da parte del CONTENT/SERVICE PROVIDER
  - Replicazione dei server 
  - Caching
- Azioni da parte di INTERMEDIARI
  - Caching cooperativo (istituzionale o ISP)
  - Content Distribution Networks (commerciale)

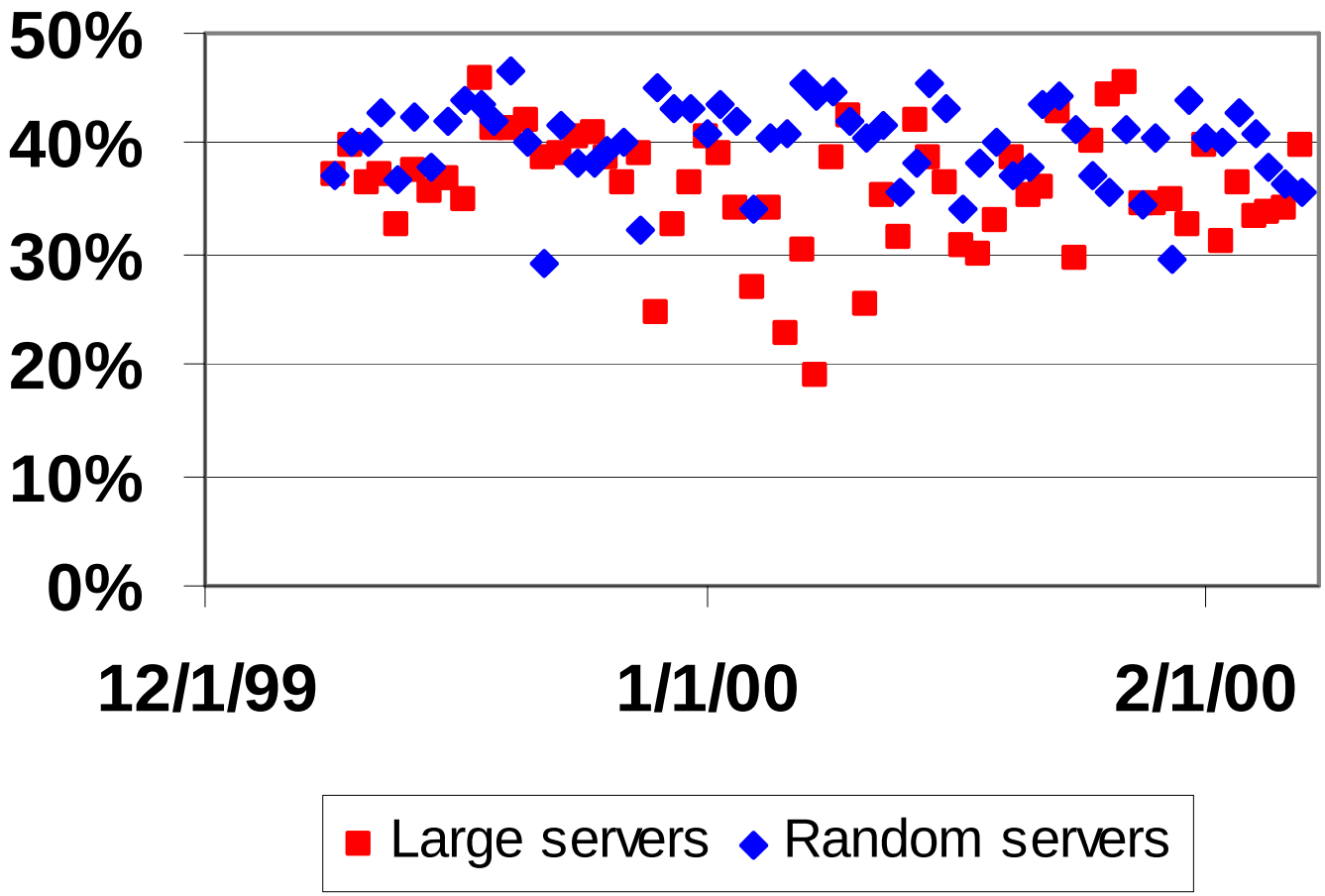
# Bottleneck: rete?

- La bandwidth tende ad aumentare
- Il round trip time (RTT) tende a diminuire
- L'infrastruttura di rete fissa tende a migliorare

- Ma ci sono ancora molti problemi aperti
  - Peering points
  - Routers
  - QoS
  - Protocolli
  - Wireless, ad-hoc networks
  - …

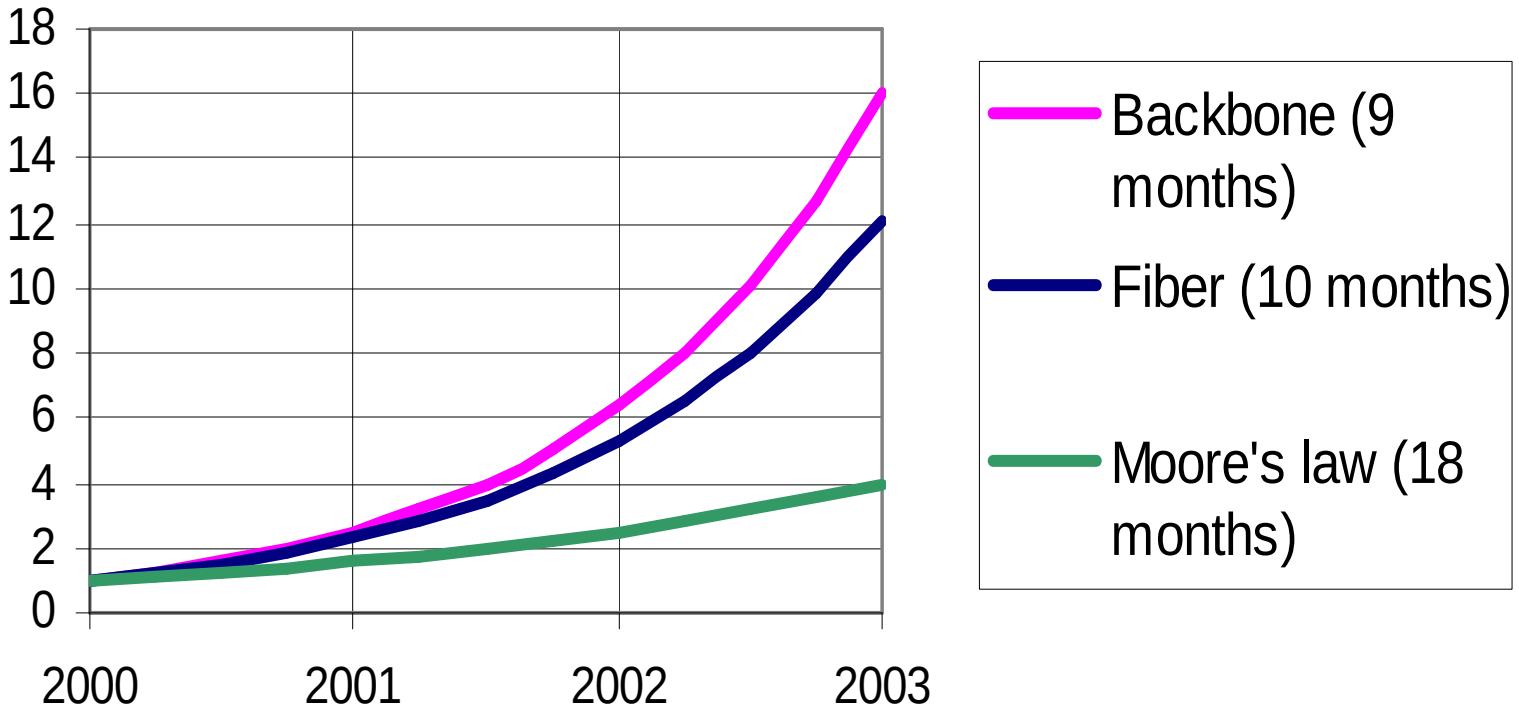
# Bottleneck: server?

## Ritardi (percentuali) dovuti al server

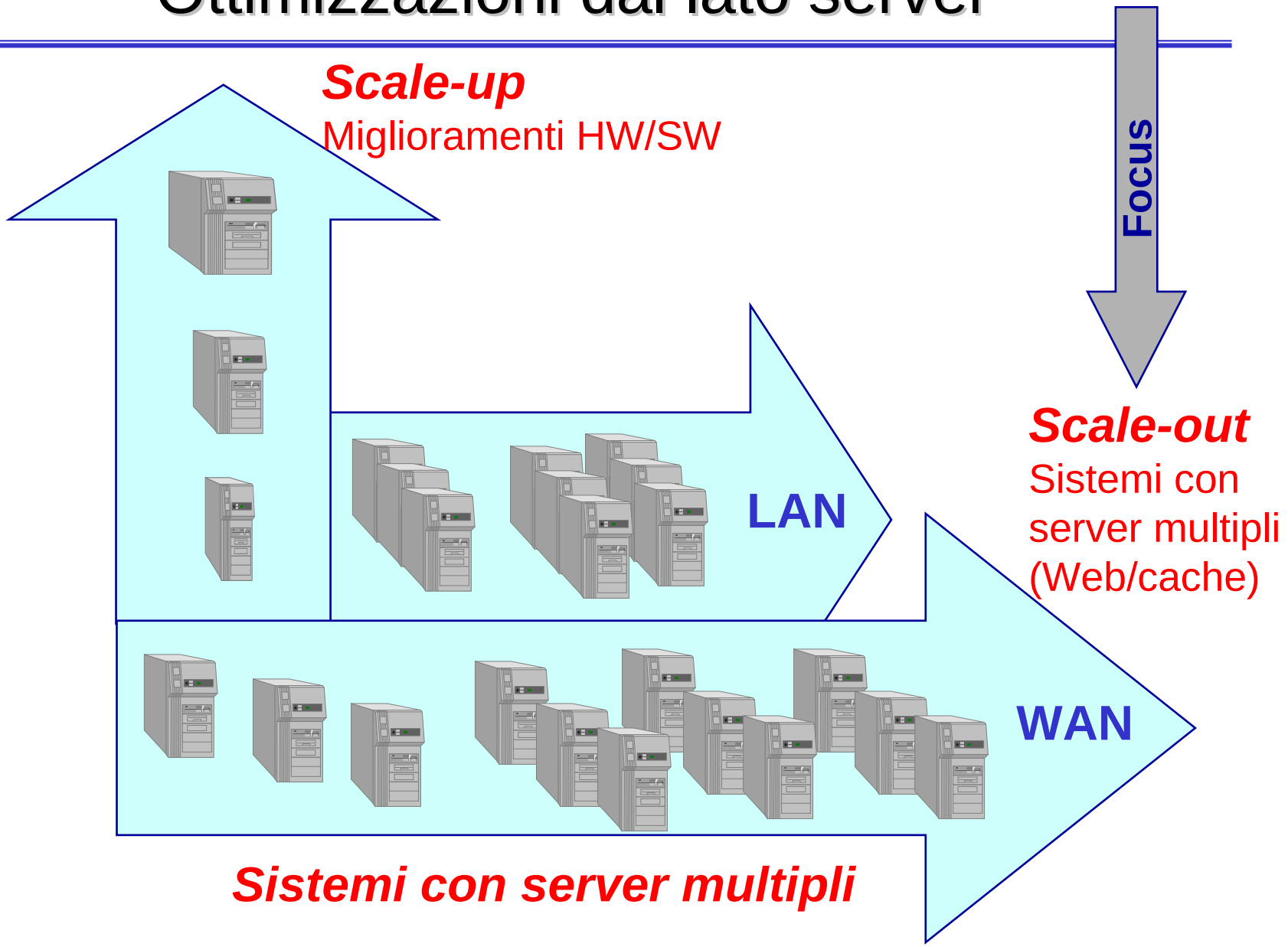


# La sfida della rete a 10 Gigabit

(ovvero *i limiti della legge di Moore*)



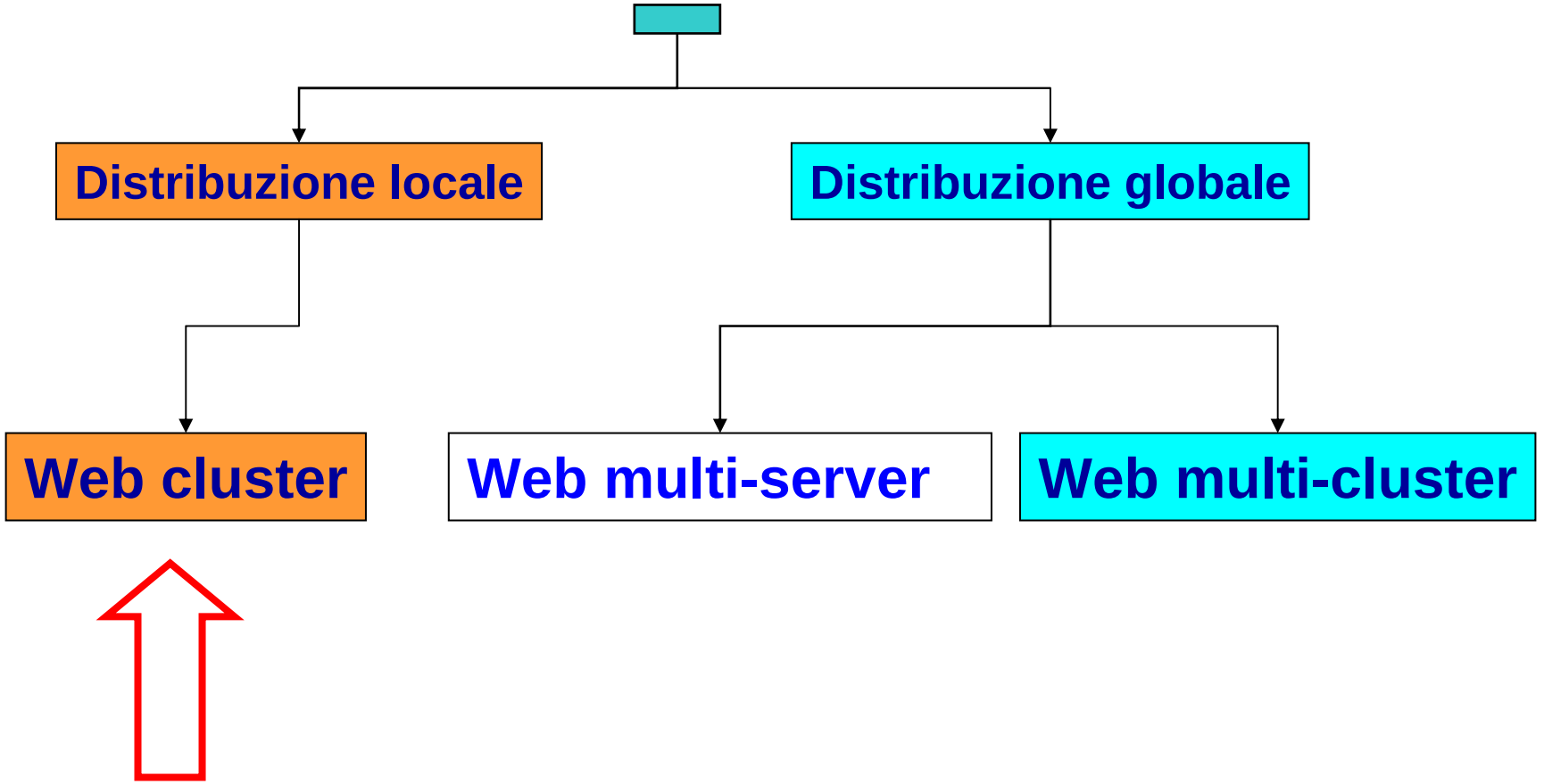
# Ottimizzazioni dal lato server



# Scale-up e scale-out

- Scale-up
  - Interventi a livello di SO
    - Evitare copie multiple dello stesso oggetto, politiche di scheduling diverse da round-robin (es., SRTF)
  - Modifiche del software del server Web
    - Apache 2.2, Flash, Zeus
- Scale-out
  - A livello di content/service provider
    - Distribuzione locale dei server
    - Distribuzione globale dei server
    - Integrazione con meccanismi di caching
  - A livello di intermediari
    - Caching cooperativo
    - Content Delivery Networks

# Sistemi Web distribuiti

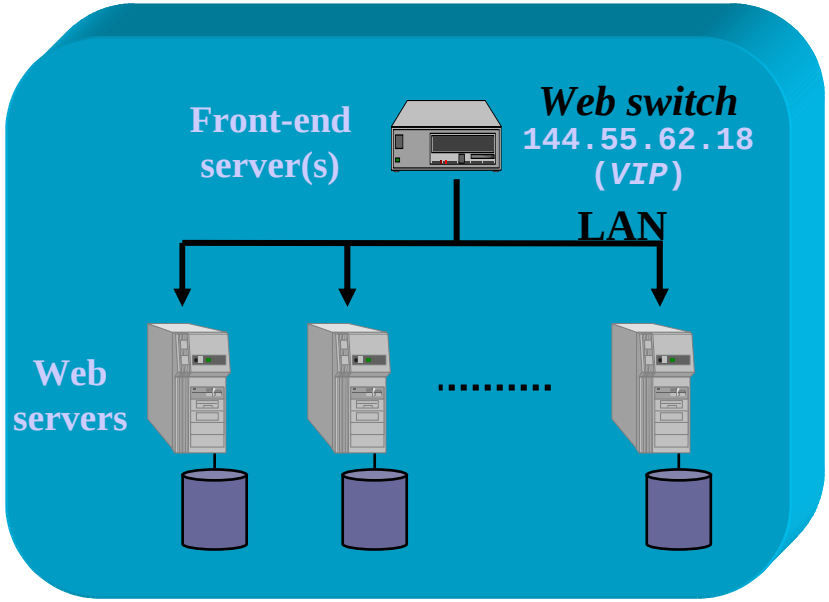


# Sistemi Web distribuiti (2)

## Sistemi Web scalabili basati su piattaforme con server multipli

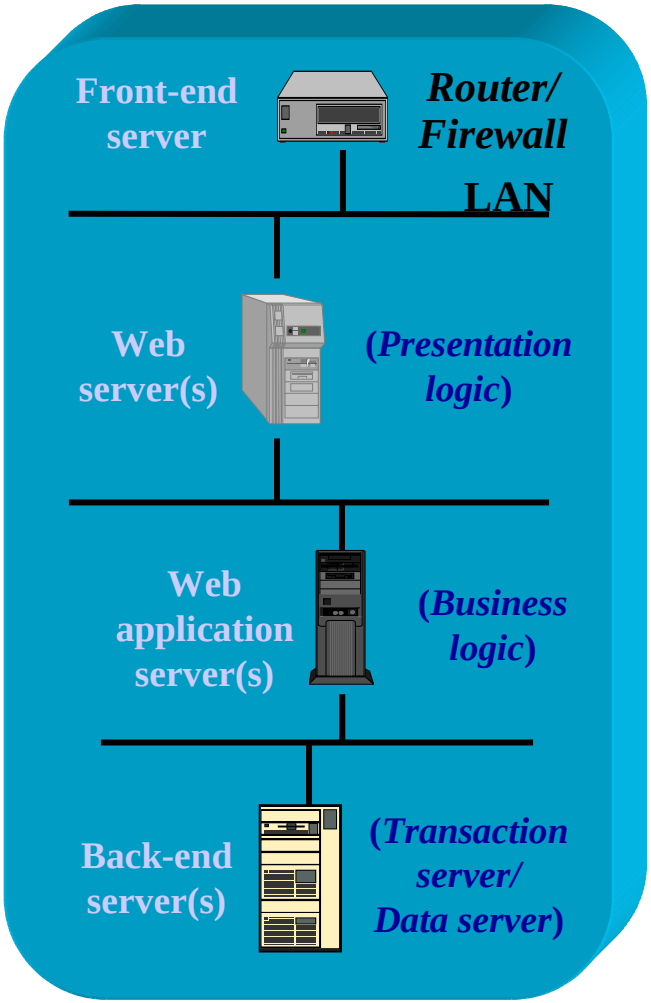
- Un **meccanismo di routing** per indirizzare le richieste client al nodo “migliore”
- Un **algoritmo di distribuzione** (*dispatching*) per individuare il nodo “migliore”
- Un componente **esecutore** per eseguire l’algoritmo di distribuzione utilizzando il relativo meccanismo di routing

# Architetture per Web cluster




**Replicazione orizzontale**

**Replicazione verticale**



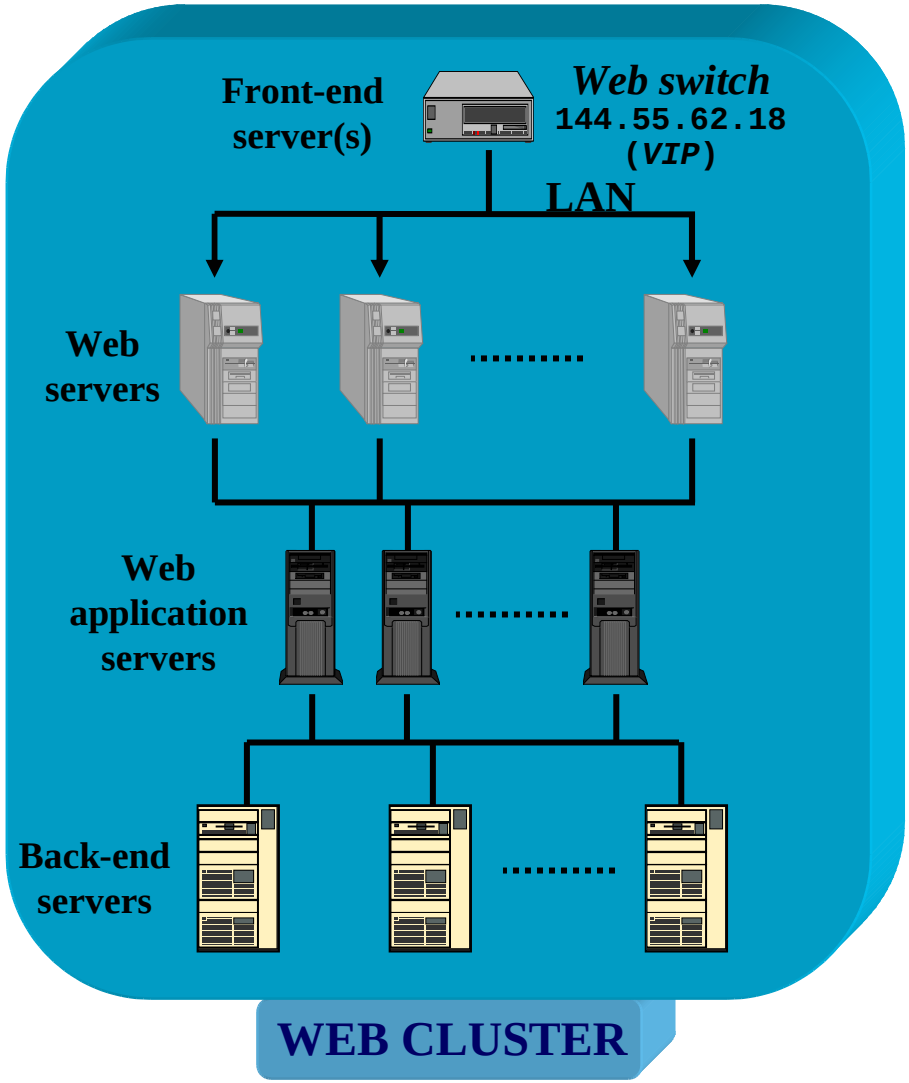
# Architetture per Web cluster:
## una miriade di tecnologie (*complesse*)

**Network/OS technology**

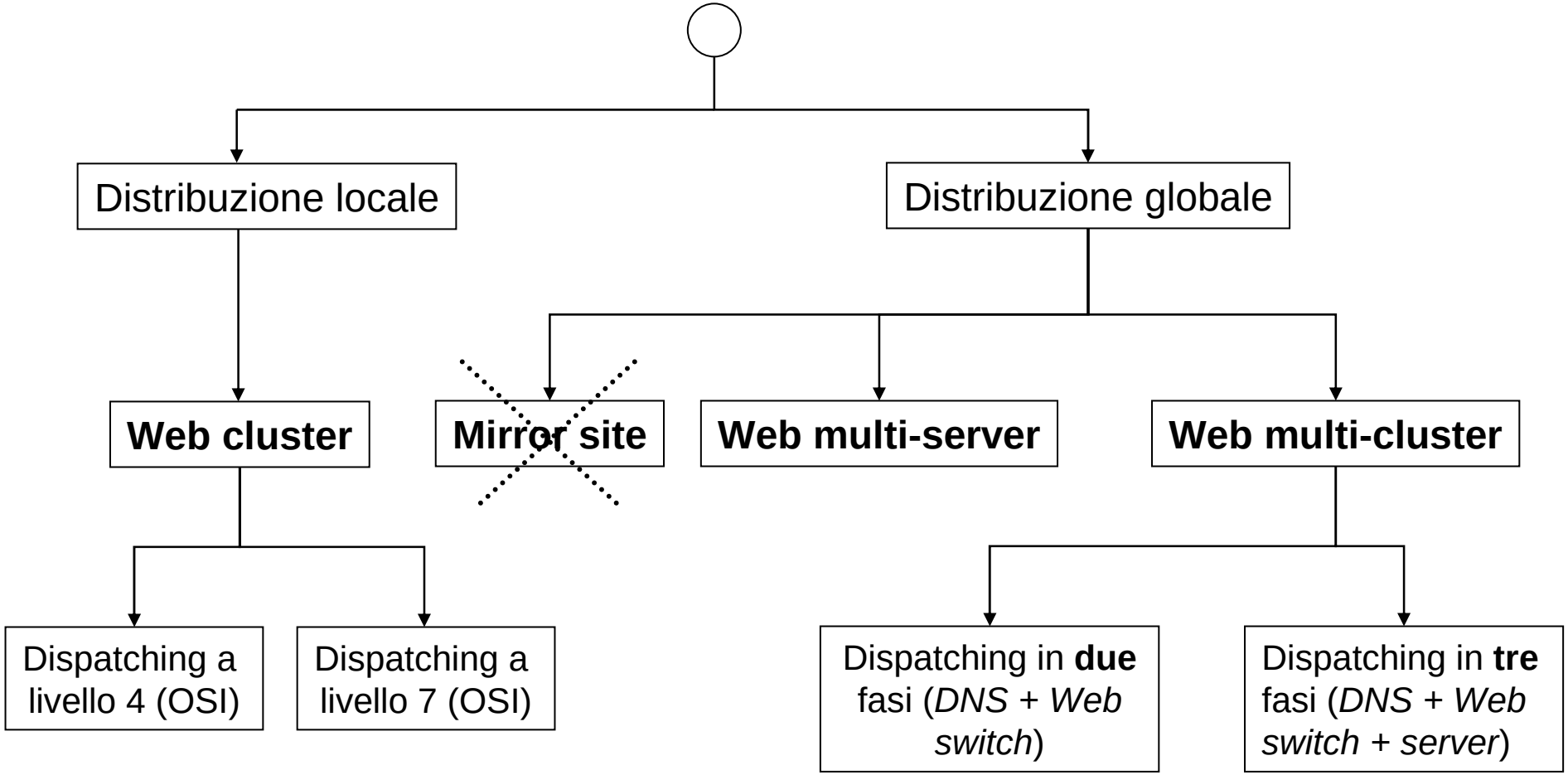
**Web server technology**

**Middleware technology**

**Database technology**



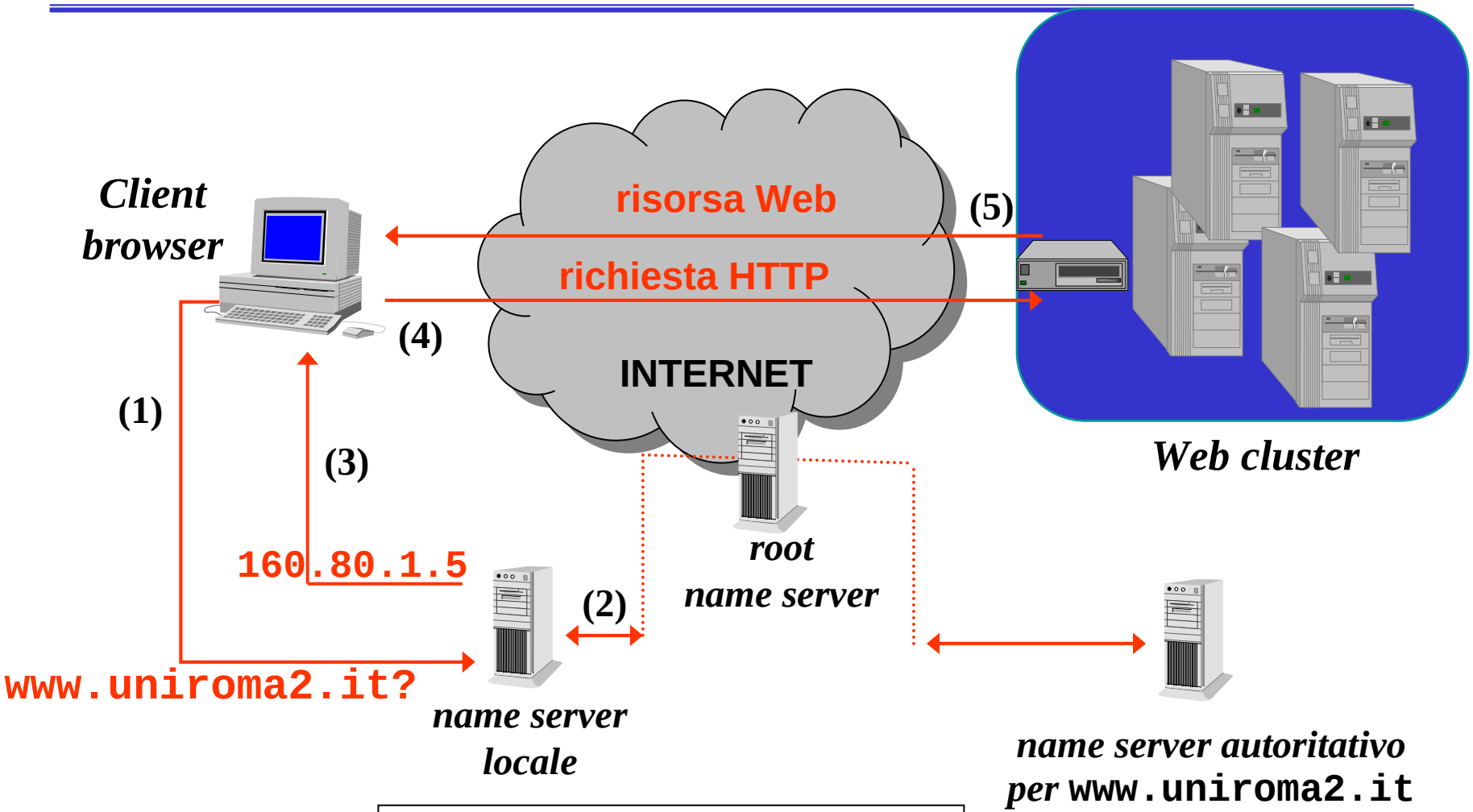
# Sistemi Web distribuiti



# Web cluster

- Sito Web implementato su di un'architettura parallela o distribuita localmente
- Indirizzi sito Web
  - Un solo hostname (es., "www.uniroma2.it")
  - Un solo indirizzo IP (***virtual IP address*** *o VIP*)
- Web server con indirizzi IP "mascherati" all'esterno
- ***Web switch*** (*dispatcher*) il cui indirizzo IP è l'indirizzo IP del sito Web (VIP)

# Richiesta HTTP ad un Web cluster



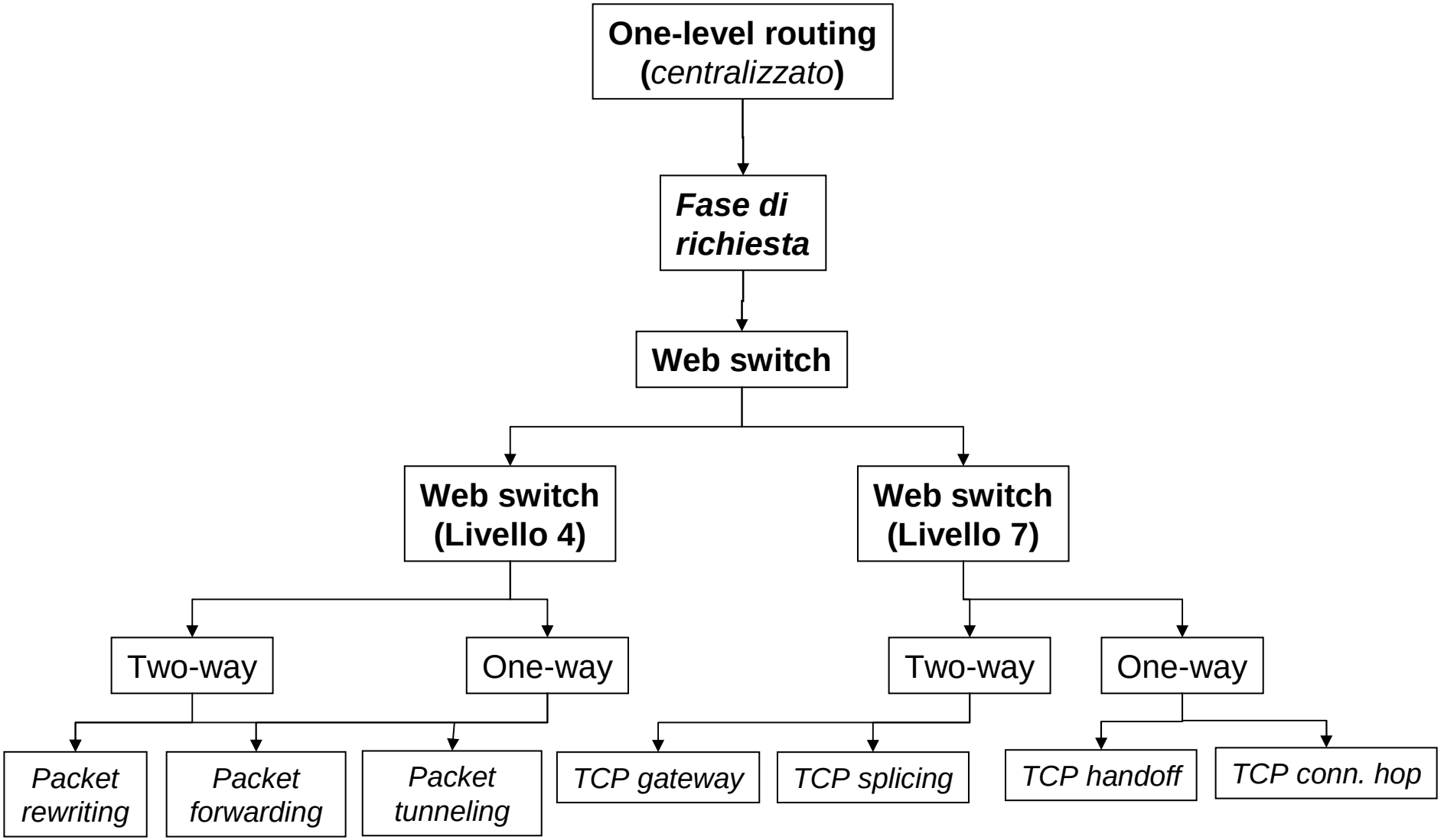

Due fattori da considerare:

- Caratteristiche del Web switch
- Flusso dei pacchetti di risposta

# Web switch del cluster

- Componente di rete con ruolo di dispatcher
  - Mapping da VIP ad indirizzo IP di un server
  - Distribuzione delle richieste a granularità fine (i pacchetti entranti con indirizzo VIP sono indirizzati dal Web switch)
    - dispositivo hardware special-purpose
    - modulo software eseguito a livello kernel (SO special-purpose)
    - modulo software eseguito a livello applicativo (SO general-purpose)
- Architetture alternative:
  - **Web switch di livello 4** (*content information blind*)
    - sorgente e destinazionazione indirizzo IP, numeri di porta TCP, SYN/FIN bit nell'header TCP
  - **Web switch di livello 7** (*content information aware*)
    - contenuto URL, cookie, SSL id

# Architetture per Web cluster



# Architetture per Web cluster (2)

Classificazione delle architetture basata su

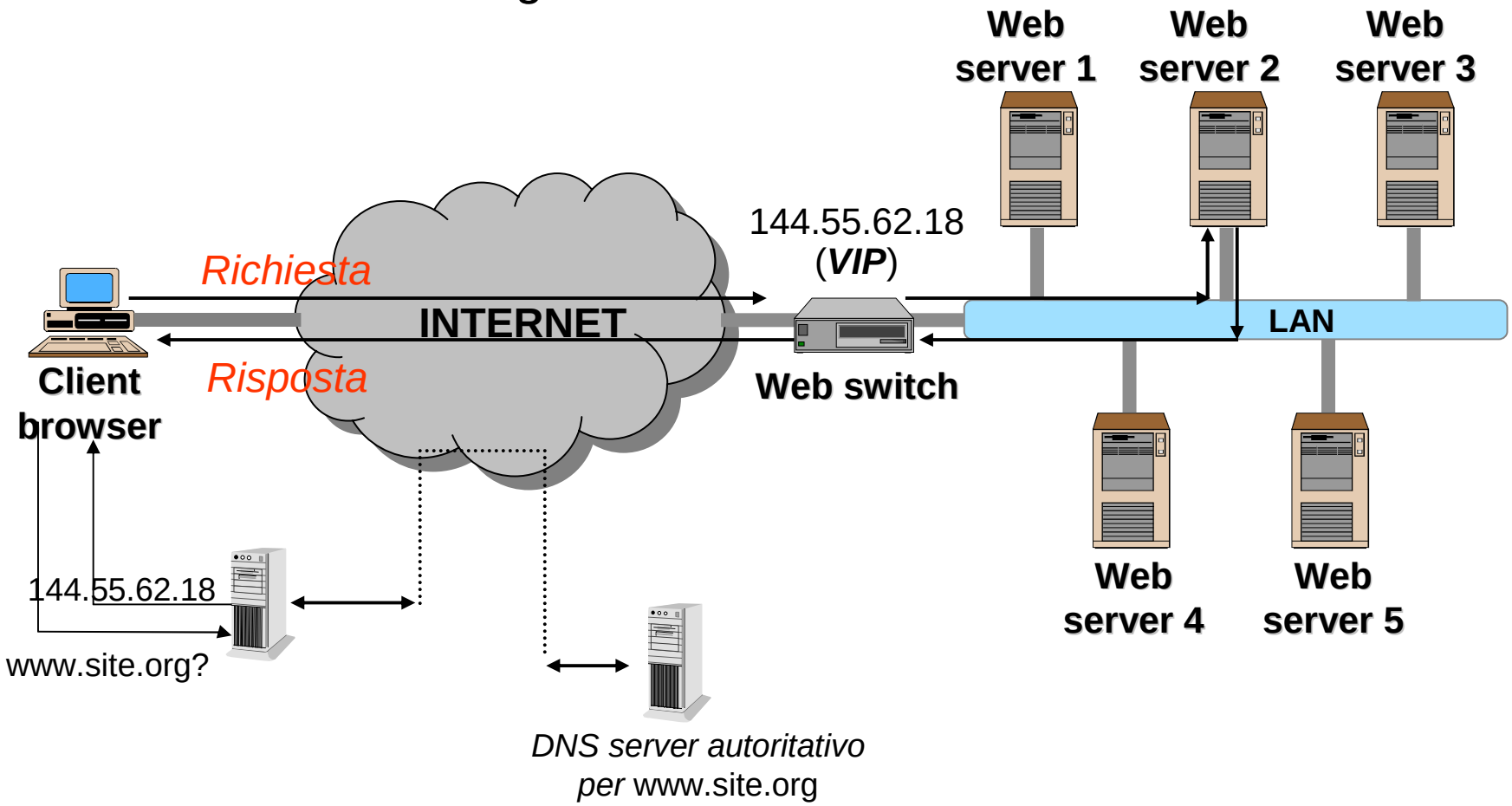
- ***Livello*** dello stack OSI a cui opera il Web switch
- ***Percorso*** seguito dai pacchetti
  - I pacchetti in ingresso (*inbound*) passano sempre dallo switch
  - I pacchetti in uscita (*outbound*)
    - Passano anche dallo switch: **architetture two-way**
    - Transitano attraverso un'altra connessione: **architetture one-way**
- ***Meccanismo*** di routing utilizzato dal Web switch per re-indirizzare i pacchetti inbound verso i server
    - Ad esempio, packet rewriting, packet forwarding, TCP handoff

# Web switch di livello 4 (L4)

- Opera a livello TCP/IP
- TCP session management
  - Pacchetti appartenenti alla stessa connessione TCP devono essere assegnati allo stesso server
  - Il Web switch utilizza una **binding table** per la gestione delle connessioni TCP attive
  - Il Web switch esamina l'header di ogni pacchetto:
    - nuova connessione *(SYN bit)* ➡ *assegnamento del server*
    - connessione esistente ➡ *ricerca nella binding table*

# Architetture di livello 4 two-way

- Packet double-rewriting



# Architetture di livello 4 two-way (2)

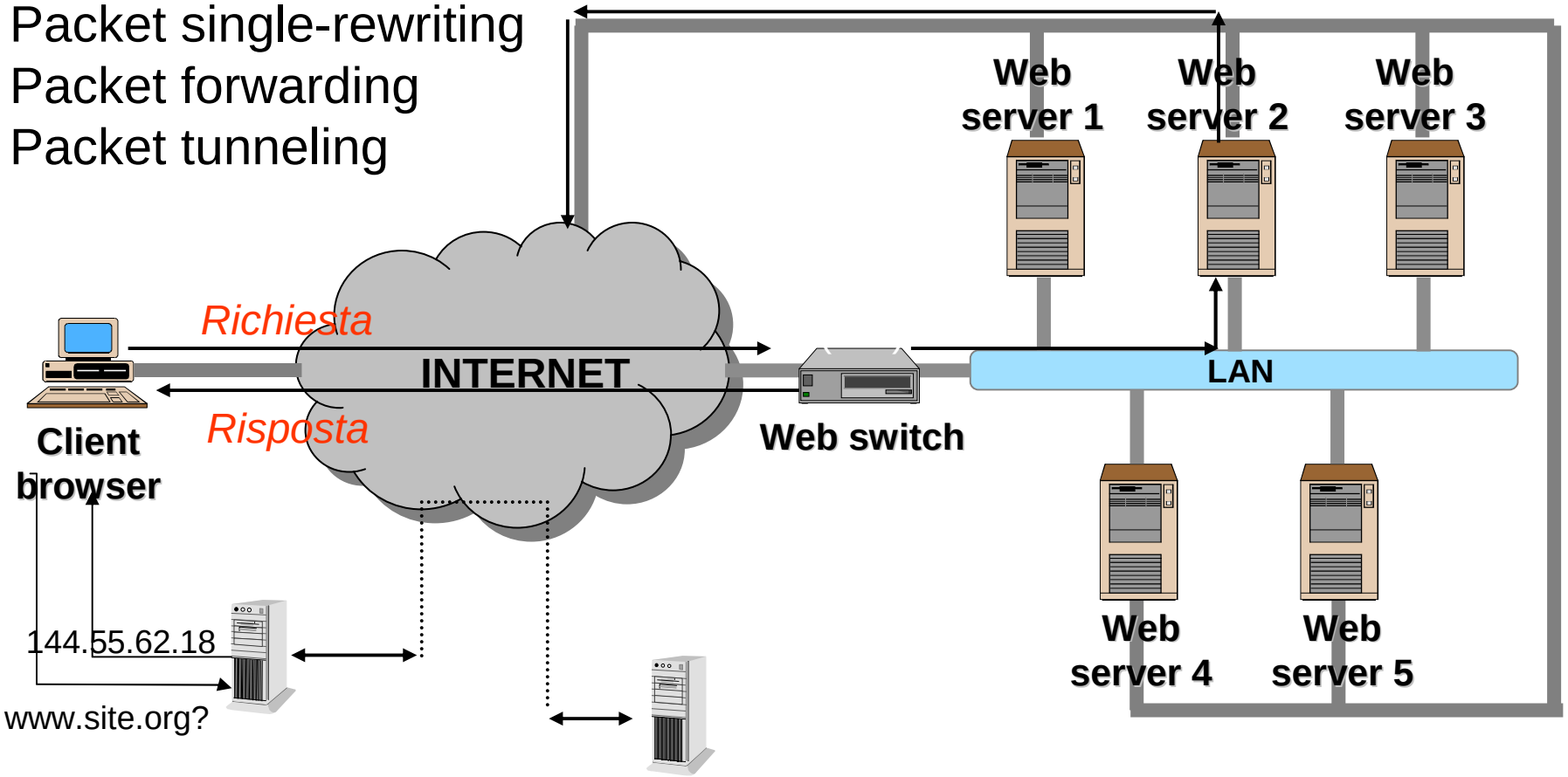
Ogni server ha il suo indirizzo IP privato

- I pacchetti in uscita devono riattraversare il Web switch
- Web switch modifica dinamicamente sia i pacchetti entranti sia quelli uscenti
  - Indirizzo IP *destinazione* dei pacchetti entranti (***VIP → IP server***)
  - Indirizzo IP *sorgente* dei pacchetti uscenti (***IP server → VIP***)
  - Ricalcolo dei checksum IP e TCP

Tecnica basata sul meccanismo di Network Address Translation (NAT)

# Architetture di livello 4 one-way

- Packet single-rewriting
- Packet forwarding
- Packet tunneling



# Architetture di livello 4 one-way (2)

## Packet single-rewriting

- Lo switch modifica solo i pacchetti IP entranti
- Il server modifica i pacchetti IP in uscita (IP server $\rightarrow$ VIP)

## Packet forwarding

- Non c'è modifica dei pacchetti IP entranti ed uscenti: i pacchetti sono inoltrati a livello MAC (***ri-indirizzamento del frame MAC***)
- *PRO*: minor overhead sullo switch per pacchetto
- *CONTRO*: Web switch e server devono trovarsi sulla stessa sottorete fisica

## Packet tunneling

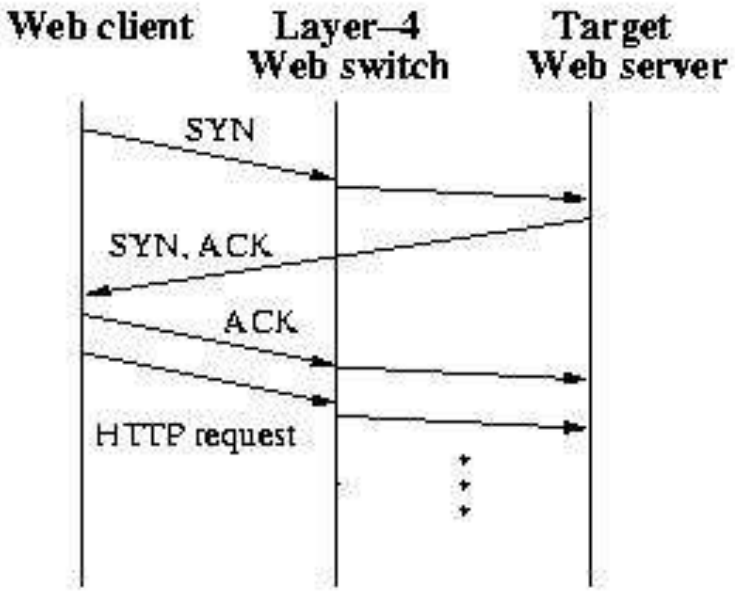
- Il pacchetto IP originale è incapsulato dallo switch in un altro pacchetto IP, il cui header contiene: VIP come indirizzo IP sorgente e indirizzo server come indirizzo IP destinatario

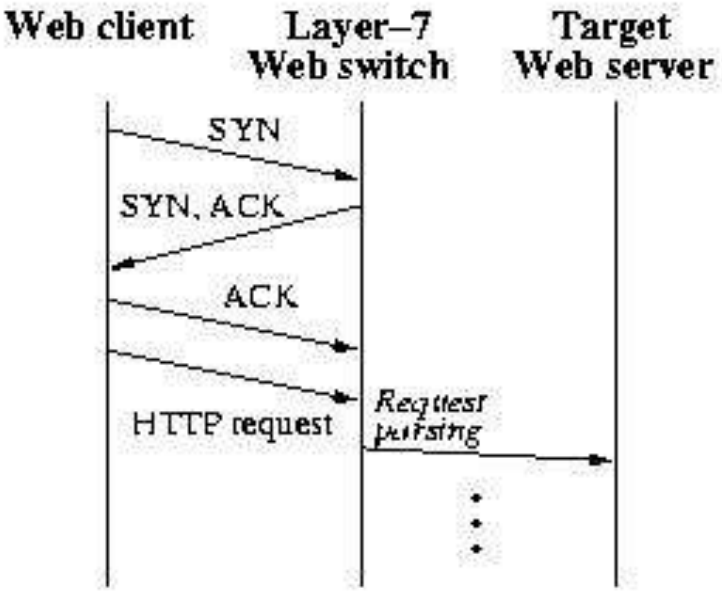
# Web switch di livello 7

- Il Web switch opera a livello applicativo
- Il Web switch deve stabilire la connessione TCP con il client ed attendere la richiesta HTTP
- Ispeziona il contenuto della richiesta HTTP per decidere a quale server inoltrarla
  - Parsing dell'header HTTP
  - Gestione dei pacchetti inbound (ACKs)

- Principali caratteristiche del content-based routing
  - Consente il partizionamento dei contenuti/servizi del sito Web tra diversi server (eventualmente specializzati)
  - Favorisce l'utilizzo di meccanismi di caching
  - Supporta il dispatching a granularità fine delle richieste HTTP effettuate tramite connessioni persistenti

# Decisione sull'assegnamento a livello 4 e 7



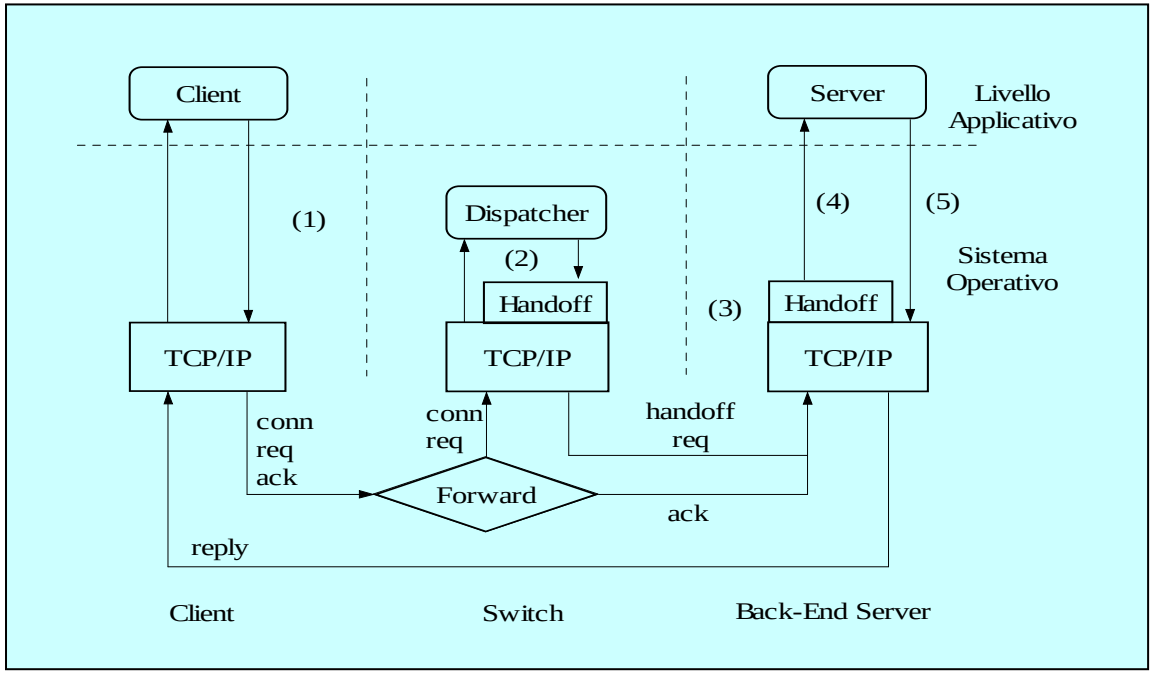
# Architetture di livello 7 two-way

- TCP gateway (livello applicativo)
  - Il Web switch è realizzato mediante un proxy
    - Il forwarding dei dati è realizzato a livello applicativo
  - Overhead elevato: ogni pacchetto deve attraversare tutto lo stack di protocolli (data link <-> application <-> data link)

- TCP splicing (livello di SO)
  - Ottimizzazione del TCP gateway
    - Il forwarding dei dati è realizzato a livello TCP
  - Richiede modifiche del kernel del SO sul Web switch

# Architetture di livello 7 one-way

- TCP handoff (livello di SO)
  - La connessione viene stabilita con il Web switch; il Web switch passa (handoff) la connessione al server che gestisce il servizio ed invia direttamente la risposta al client
  - Richiede modifiche del kernel sia sul Web switch che sui server



# Algoritmi di distribuzione

– **Statici** (stateless)

– **Dinamici** (state aware)
  - client info aware
  - server state aware
  - client info & server state aware

# Algoritmi statici *vs.* dinamici

## Statici

- Facile implementazione
- Overhead trascurabile (sullo switch)
- Possibili situazioni di sbilanciamento del carico

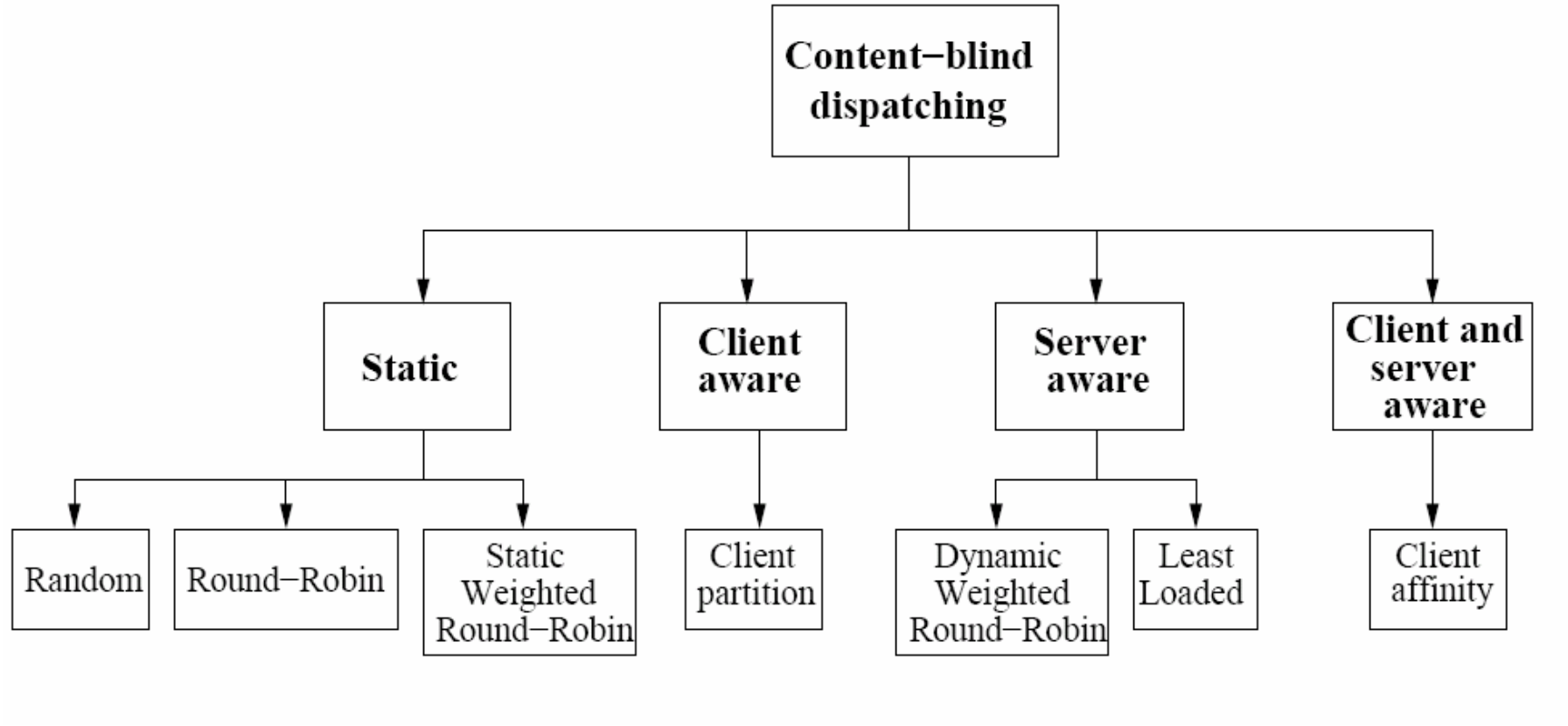
## Dinamici

- Implementazione più complessa
- Overhead di comunicazione (tra switch e server) e di computazione (sullo switch)
- Miglior bilanciamento del carico a parità di politica adottata

# Confronto meccanismi di distribuzione

- Livello 4
  - Livello connessione TCP
  - Algoritmi di distribuzione *content blind*
  - Algoritmi statici e dinamici
- Livello 7
  - Livello connessione applicativa (HTTP)
  - Algoritmi di distribuzione *content aware*
  - Algoritmi dinamici
    - almeno client info aware (occorre usare l'informazione contenuta nella richiesta del client!)

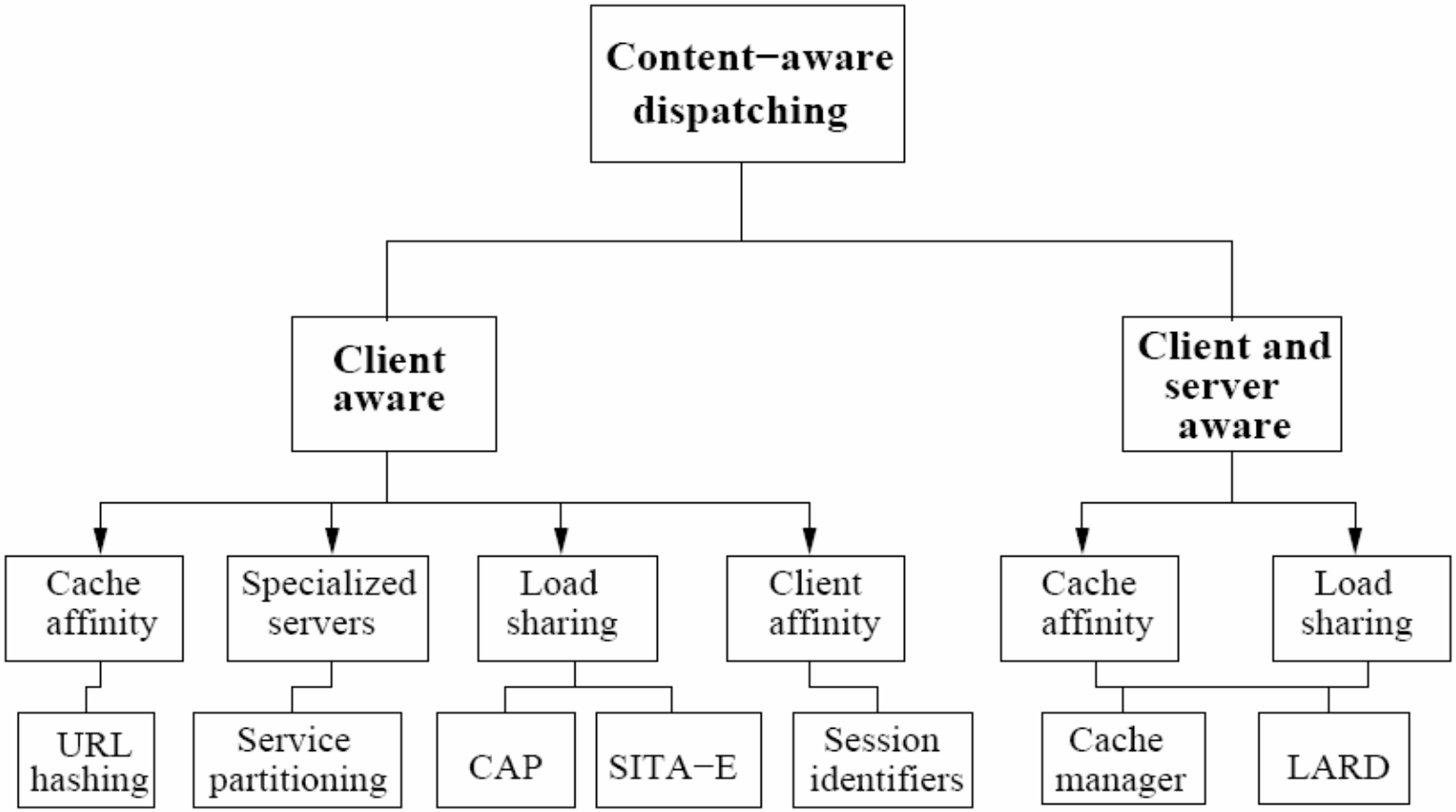
# Content blind dispatching



# Distribuzione a livello 4

- Non servono algoritmi particolarmente sofisticati
- Esempi di algoritmi statici:
  - Random, Round Robin (assegnamento circolare)
- Esempi di algoritmi dinamici server state-aware:
  - Attuano una distribuzione delle richieste in base allo stato di carico dei server
  - Least loaded, Weighted Round Robin
- Gli algoritmi statici forniscono prestazioni confrontabili a quelle di algoritmi dinamici nel caso di servizi Web con tempi di servizio che rientrano in intervalli temporali di 2 ordini di grandezza
- Oltre i 2 ordini di grandezza, è opportuno utilizzare algoritmi dinamici (client o preferibilmente server state aware)

# Content Aware Dispatching



# Distribuzione a livello 7

- Algoritmi di distribuzione client info aware
  - Esempi: Session ID, Content Partition, CAP
- Algoritmi di distribuzione client info & server state aware
  - Esempio: LARD

# Algoritmi L7 client info aware

- Identificatori di sessione
  - Richieste HTTP con stesso *SSL id* o stesso *cookie* assegnate allo stesso server
- Partizione del contenuto (statico)
  - Contenuto partizionato tra i server rispetto al *tipo di file* (HTML, immagini, contenuto dinamico, audio, video, …)
    - Scopo: utilizzare server specializzati per contenuti differenti
  - Contenuto partizionato tra i server rispetto alla *dimensione dei file* (soglie dinamiche)
    - Scopo: aumentare load balancing in presenza di contenuti eterogenei
  - Insieme dei file partizionato tra i server mediante una *funzione hash*
    - Scopo: aumentare il *cache hit rate* nei server Web

# Algoritmi L7 client info aware (2)

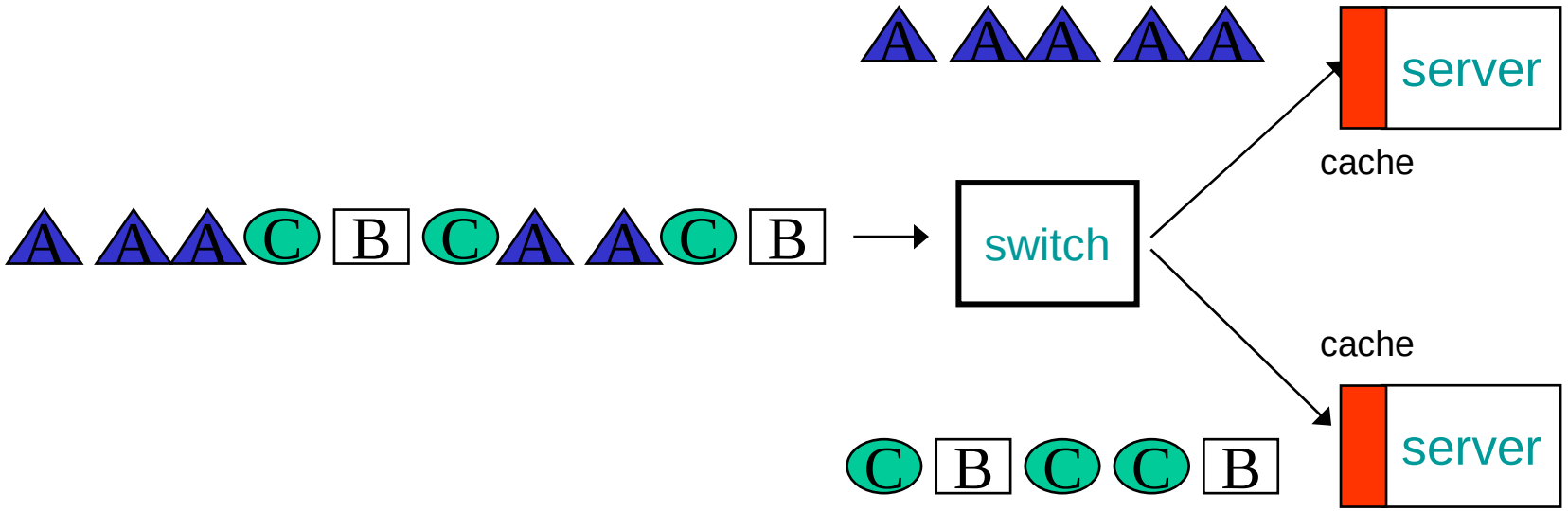
Content Aware Policy (CAP)

- Sfrutta informazioni relative alla tipologia della richiesta da assegnare
- Richiede un meccanismo di classificazione dinamica delle richieste effettuabile in base al tipo di servizio (URL)
  - *CPU-bound* (es., crittografia)
  - *Disk-bound* (query a database)
  - *Network-bound* (download di file di grandi dimensioni)
- Obiettivo: ripartire le richieste *CPU/disk/network-bound* tra tutti i server

# Algoritmi L7 client info & server state aware

Locality Aware Request Distribution (LARD)

- considera sia il tipo di richiesta/servizio sia lo stato di carico dei server Web
- ha l'ulteriore obiettivo di aumentare il *cache hit rate* dei server Web

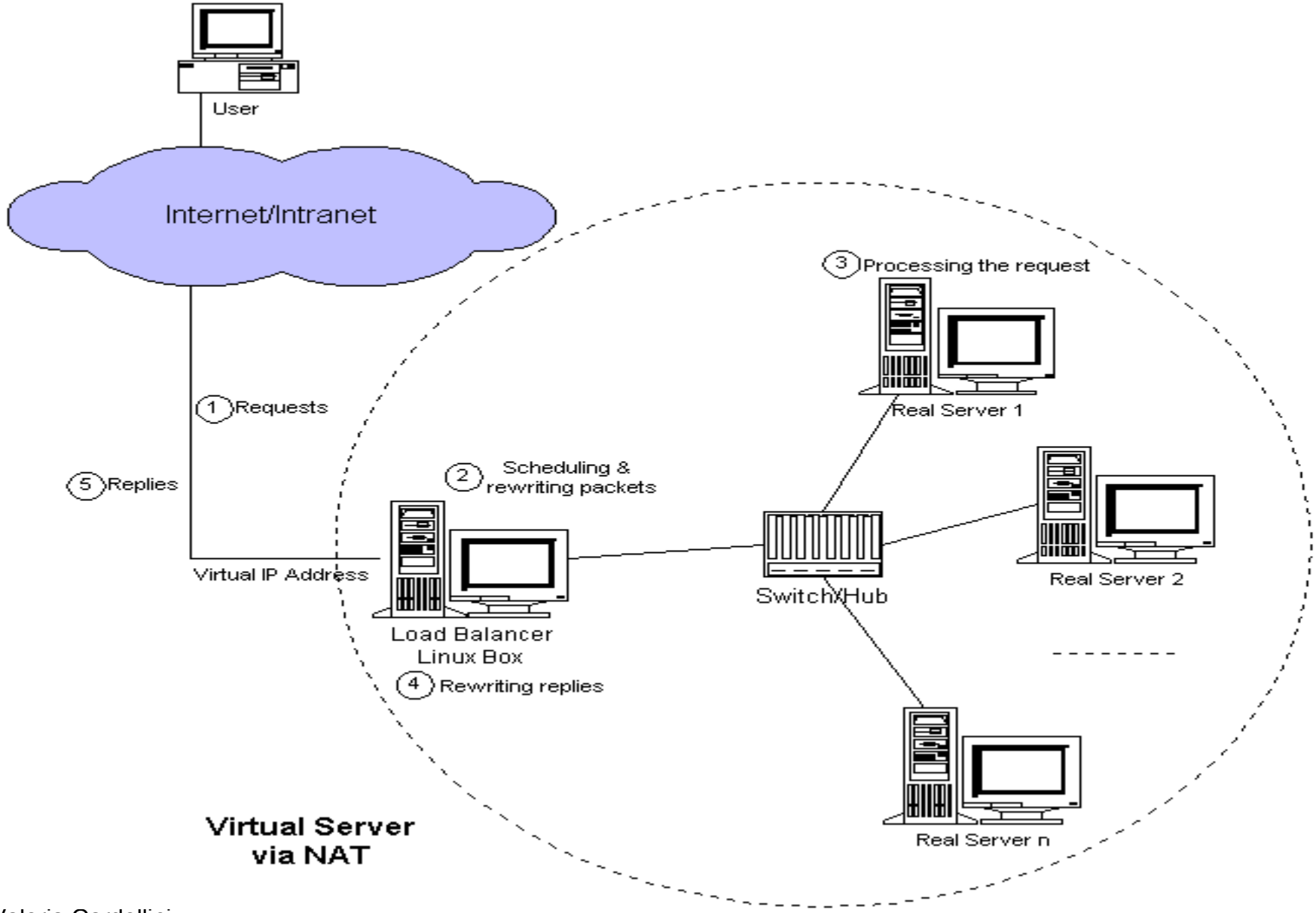


# Linux Virtual Server (LVS)

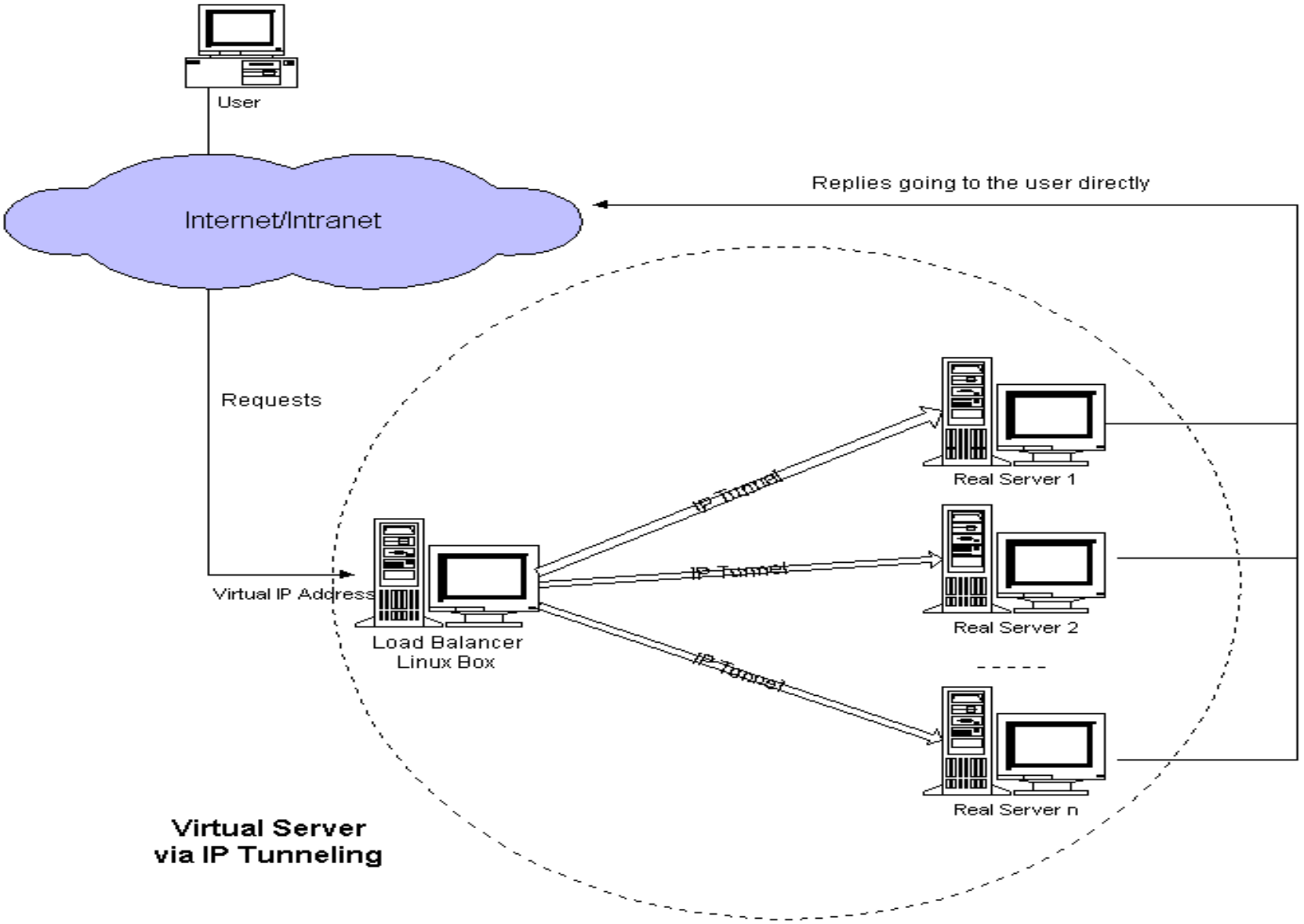
- www.linuxvirtualserver.org
- Sistema per la realizzazione di Web cluster con Web switch operante a livello 4
  - Per switching a livello 7 vedere i sottoprogetti
    - KTCPVS (Kernel TCP Virtual Server): livello applicativo nel kernel
    - TCPHA: TCP handoff
    - TCPSP: TCP splicing

- Caratteristiche salienti
  - Scalabilità
    - Possibilità di aggiungere e rimuovere dinamicamente i nodi dal cluster
  - Elevata disponibilità
    - Meccanismo per la riconfigurazione dinamica del sistema e per il riconoscimento di failure dei nodi (Ultra Monkey Project)
  - Configurazione LAN e WAN

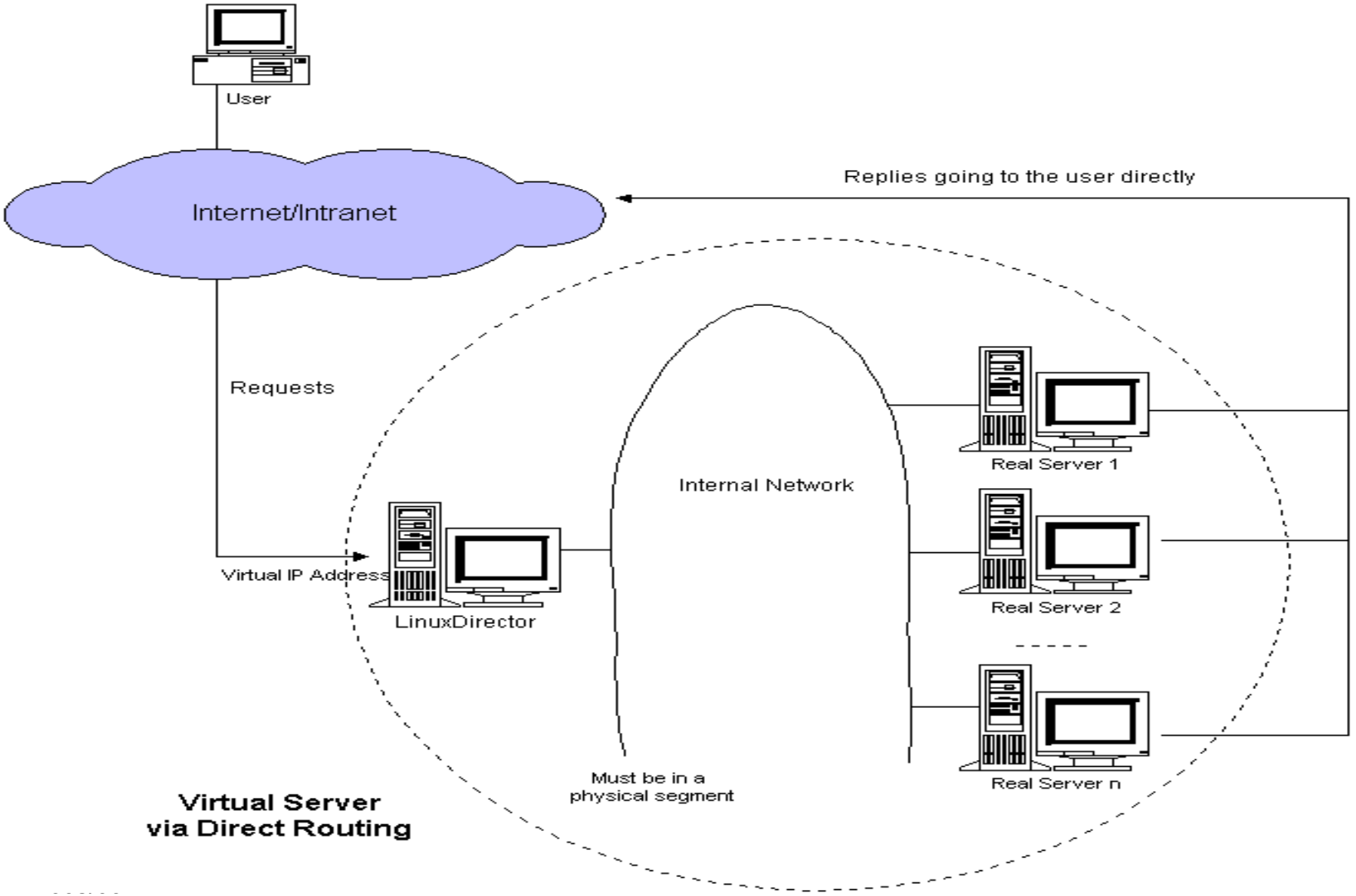
# LVS/NAT: Network Address Translation



# LVS/IP: IP tunneling



# LVS/DR: Direct Routing



# Sommario caratteristiche Web cluster

- Architetture alternative
  - Web switch livello 4 vs. Web switch livello 7
  - One-way vs. two-way
- Principali vantaggi
  - Controllo a granularità fine sull'assegnamento delle richieste
  - Elevata affidabilità (*availability*, *sicurezza*)
- Principali svantaggi
  - Single point of failure
  - Scalabilità limitata dal Web switch
  - Scalabilità limitata dalla banda di accesso ad Internet (es., T3 ≈ 45 Mbps)