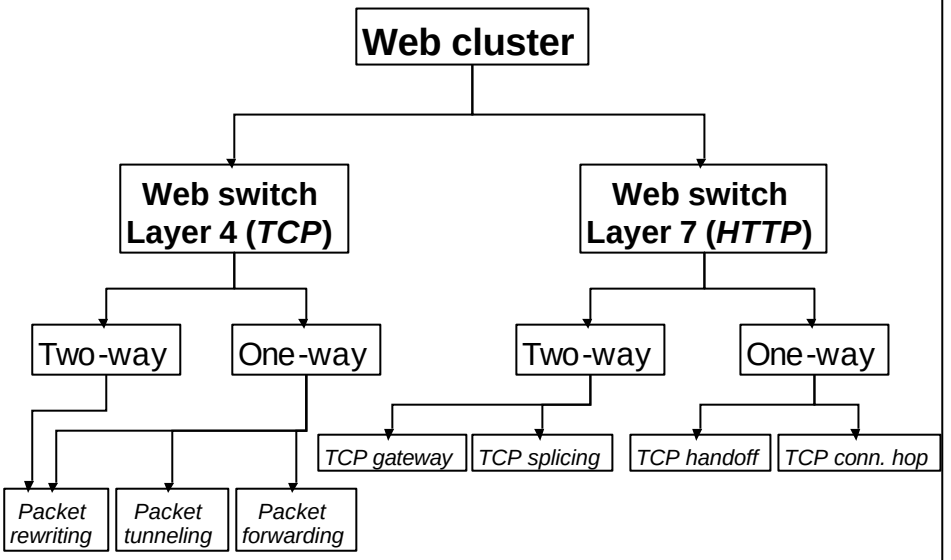
Web cluster alternatives

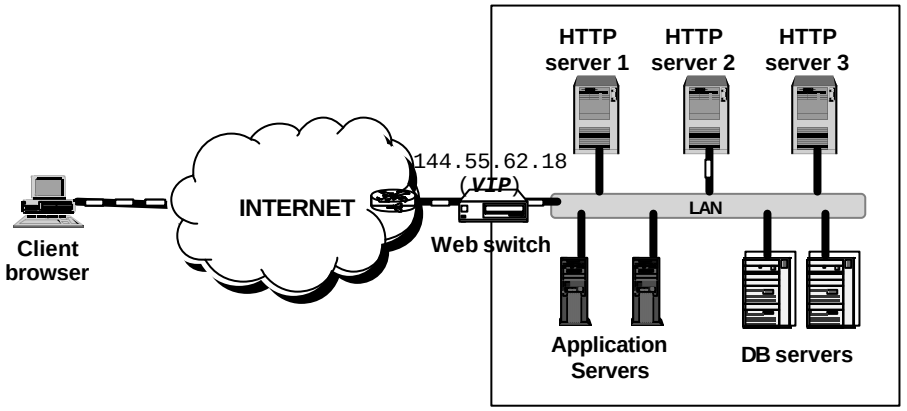


Layer 4 Web switch

- **Layer 4 Web switch works at** TCP/IP level
- **Mapping on a per-connection basis**
  - Packets pertaining to the same TCP connection must be assigned to the same server machine
  - **Binding table** maintained by the Web switch to associate each active connection with the assigned server
    - The Web switch examines the header of each incoming packet
      - new connection (**SYN bit**) → *new server assignment*
      - existing connection → *lookup in the binding table*
    - Each connection requires about 32 bytes of information in the binding table

Two-way architecture

- *Packet rewriting* technique



Layer 4 – Two way – *Packet rewriting*

- **Packet rewriting is based on the IP Network Address Translation (NAT) technique**

  [K. Egevang, P. Francis, "The IP Network Address Translator (NAT)", RFC 1631, May 1994]

  – Each internal server has its own private IP address

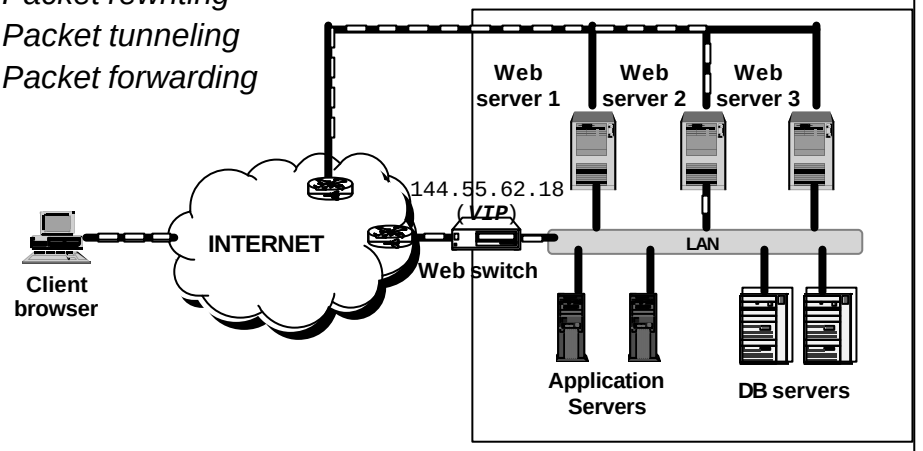
  – Outbound packets must pass back through the Web switch

  – The Web switch dynamically modifies **both** inbound and outbound IP packets

  - IP destination address in inbound packet (VIP → IP server address)
  - IP source address in outbound packet (IP server → VIP)
  - IP and TCP checksum recalculation

One-way architecture

- *Packet rewriting*
- *Packet tunneling*
- *Packet forwarding*



Layer 4 – One way – *Packet rewriting*

- **Outbound packets do not need to pass back through the Web switch**
  - A separate high-bandwidth connection can be used for outbound packets
- **Each internal server has its own unique IP address**
- **The Web switch modifies** only **inbound IP packets**
  - IP destination address in inbound packet (VIP → IP server)
  - IP and TCP checksum recalculation
- **The HTTP server modifies outbound IP packets**
  - IP source address in outbound packet (IP server → VIP)
  - IP and TCP checksum recalculation
  - **Modifications of the server kernel (TCP/IP stack)**

Layer 4 – One way – *Packet tunneling*

- **IP tunneling (or *IP encapsulation*) is a technique to encapsulate IP datagrams within IP datagrams. The effect is to transform the old headers and data into the payload of the new packet**
- **The Web switch tunnels the inbound packet destined to the VIP address to the HTTP server by encapsulating it within an IP datagram**
- **When the target server receives the encapsulated packet**
  - it strips the IP header off and finds that the inside packet is destined to the VIP address
  - it processes the request and returns the response directly to the client by using VIP as the source address

Layer 4 – One way – *Packet forwarding*

- **Clever idea!**
- **PRO: Little overhead per packet**
- **CON: Web switch and HTTP servers must be on the same physical network segment**
- **There is no modification of inbound/outbound packets at TCP/IP layers**
- **Packets are forwarded from the Web switch to an HTTP server and vice versa at the MAC level (*through a redirection of the MAC frames*)**

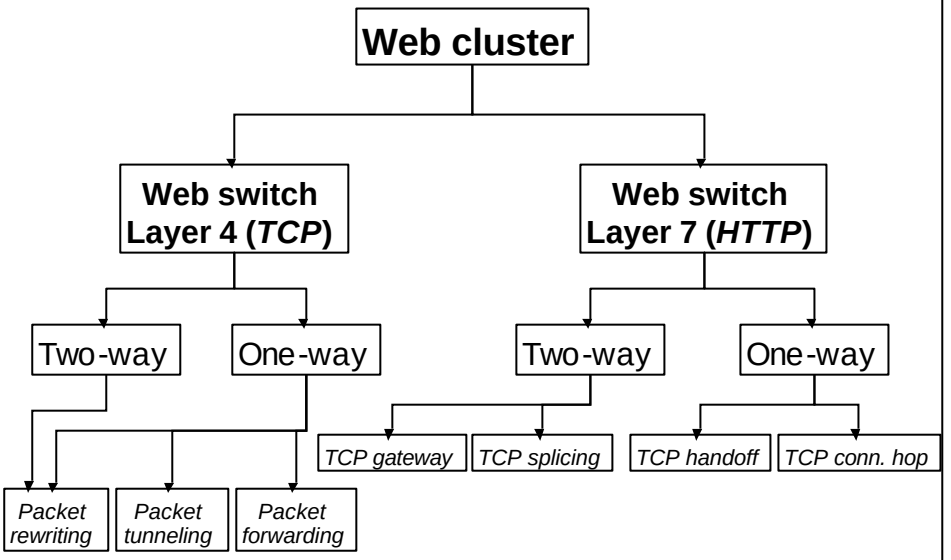
Layer 4 – One way – *Packet forwarding*

- **The same VIP address is shared by the Web switch and the servers through the use of primary (→ switch) and secondary IP addresses (→ HTTP servers)**
- **Even if all nodes share the VIP address, the inbound packets reach the Web switch because, to avoid collisions, the server nodes disable ARP**
- **The server private addresses are now at the MAC layer (*packet forwarding = MAC address translation*)**
- **The Web switch forwards an inbound packet to a server by writing the server MAC address in the layer 2 destination address and re-transmitting the frame on the shared LAN segment**
- **Since all nodes share the same VIP address, the server with the right MAC address can recognize itself as a destination and can respond directly to the client**

Prototypes/Products (*layer 4 clusters*)

Two-way		One-way	
Packet rewriting	*Packet rewriting*	*Packet tunneling*	*Packet forwarding*
• **Cisco's** LocalDirector	• **IBM** TCP router	• **Linux** Virtual server (**LVS**)	• **IBM** Network Dispatcher
• **Magicrouter**			• **ONE-IP**
• **Foundry Networks'** ServerIron			• **LSMAC Dispatcher**
• **Alteon WebSystems**			• **BIG-IP**
• **LSNAT**			• **F5 Networks**
• **Linux Virtual Server**			• **LSMAC**
• **F5 Networks** BIG/ip			• **NetStructure**
• **HydraWeb Techs**			• **Intel Traffic Director**
• **Coyote Point Systems** Equalizer			• **Alteon 180**
• **Radware's** WSD			• **Nortel Networks 2002**
			• **Radware**
			• **Foundry Networks' Server Iron**

Web cluster alternatives



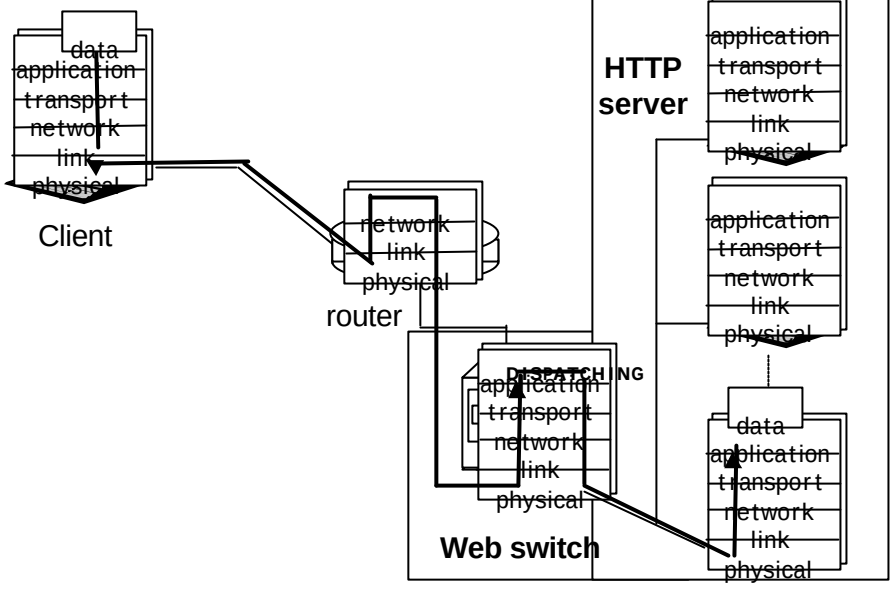
Layer 7 Web switch

- **Level 7 Web switch works at** application **level**
- **The Web switch must establish a connection with the client, and inspects the HTTP request content to decide about dispatching**
  - The switch parses HTTP header (URL, cookie)
  - The switch manages inbound packets (ACK packets)

Layer 7 – Two way – *TCP gateway*

- **On the Web switch is executed an *application level proxy* that mediates all communications between a client and a server**
- **The Web switch:**
  - maintains a permanent TCP connection with each HTTP server (for efficiency reasons)
  - issues the same request to the selected HTTP server
  - Packets are forwarded by the Web switch at application level
- **TCP gateway technique is affected by serious overhead**
  - Two TCP connections per HTTP request
  - Way up and down through the protocol stack from/to the application layer

Communication through layer-7 switch



Layer 7 – Two way – *TCP splicing*

- **It is quite similar to the TCP gateway technique, but data are forwarded by the switch at network level**
- **Once the TCP connection between the client and the Web switch has been established and the persistent TCP connection between the switch and the target server has been chosen, the two connections are *spliced* together ("joined")**
  - IP packets are forwarded from one endpoint to the others without traversing the transport layer up to the application layer on the Web switch
  - The Web switch kernel handles the subsequent packets by changing the IP and TCP packet headers so that both the client and HTTP server can recognize these packets as destined to them
- **It requires modifications at the kernel level**

Layer 7 – One way – *TCP handoff*

- **Permanent TCP connection between the Web switch and each HTTP server**
- **The Web switch "passes" (*handoff*) the TCP connection established by the client with the Web switch to the HTTP server, which can communicate directly with the client**
- **The TCP hand-off mechanism remains transparent to the client, as packets sent by the servers appear to be coming from the Web switch**
- **Incoming traffic on already established connections (i.e., any ack packet sent by the client to the switch) is forwarded to the target server by an efficient module of the Web switch**
- **The TCP hand-off mechanism requires consistent modifications to the switch and server kernels**

Layer 7 – One way – *TCP connection hop*

- **This a software-based proprietary solution implemented by Resonate as a TCP-based encapsulation protocol**
- **It is executed at the network layer between the *network interface card* (NIC) driver and the server's native TCP/IP stack**

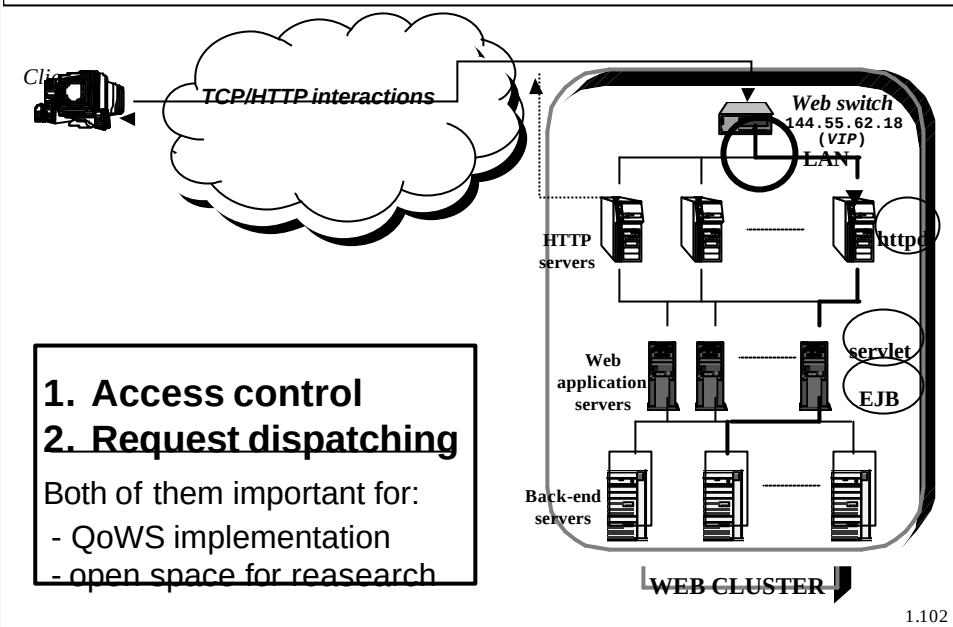
  – Proprietary protocol → Not enough information

Prototypes/Products (*layer 7 clusters*)

Two-way		One-way	
TCP gateway	***TCP splicing***	***TCP handoff***	***TCP conn. hop***
• **IBM** Network Dispatcher • HACC	• Nortel's Alteon Web Systems • F5 BIG-IP • Foundry Nets' ServerIron • IBM Network Dispatcher • Cisco CSS • Radware WSD • Zeus Load Balancer	• **ScalaServer** • **ClubWeb*** [by Weblab]	• **Resonate's** Central Dispatcher

DISPATCHING ALGORITHMS

Decisions at the Web switch



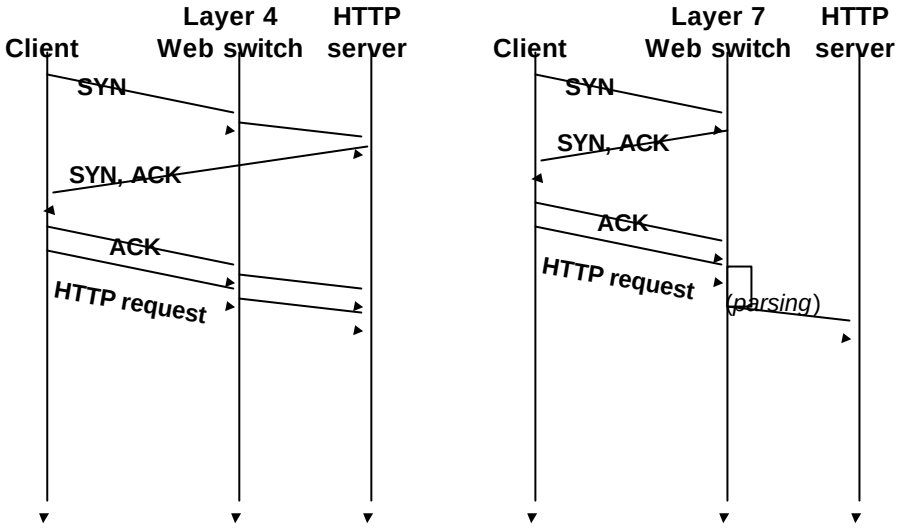
1. **Access control**
2. **Request dispatching**

Both of them important for:
- QoWS implementation
- open space for reasearch

Web switch dispatching

- **Mapping from VIP to actual server address**
- **Hit/Page request distribution**
- *Fine grain* **control on request assignment**
- *Centralized* **control**

Layer 4 *vs*. Layer 7 operations



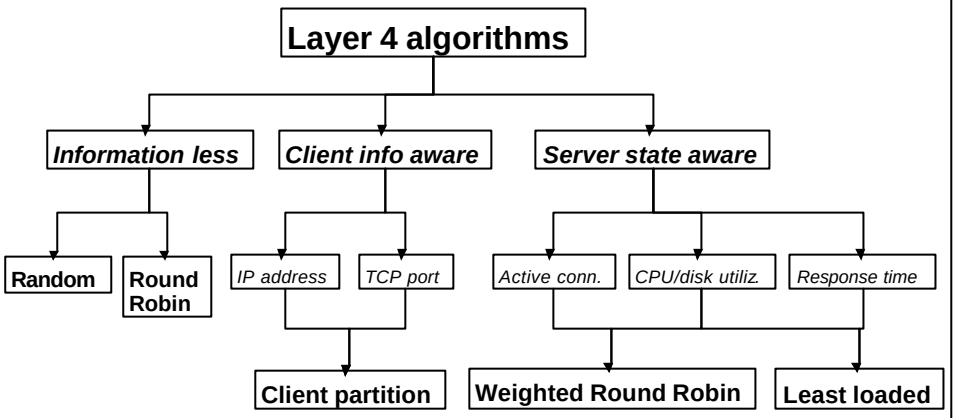
Consequences on dispatching

- **Layer 4**
  - TCP connection
  - ***Content blind*** dispatching
  - Stateless and state-aware algorithms
- **Layer 7**
  - HTTP connection
  - ***Content aware*** dispatching
  - Stateless and state-aware algorithms

Layer 7 Web switch properties

- **Main features of content-aware dispatching (or *content-based routing*)**
  - allows content/type segregation on specialized servers
  - supports persistent connections
  - facilitates caching mechanisms
  - allows HTTP/1.1 requests to be assigned to different HTTP servers

Layer 4 Web switch algorithms



Static algorithms

- **Random**
  - no information regarding the cluster state
  - no history about previous assignments
- **Round Robin (RR)**
  - no information regarding the cluster state
  - history regarding only the previous assignment

Client info aware algorithms

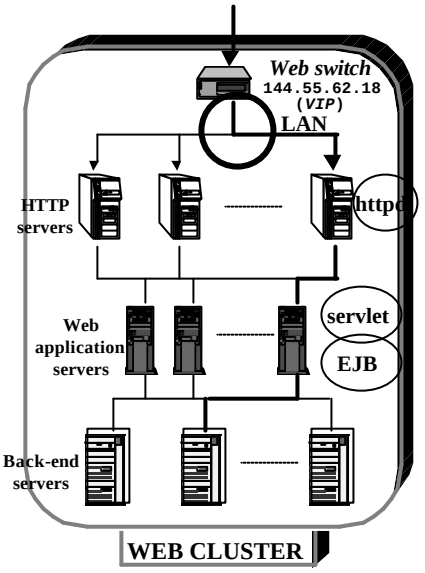
- **Client partition**
  - Request assignment based on client information in inbound packets
    - Client IP address
    - Client port
  - Rough method to implement QoWS disciplines for individuals or group of clients (not users!)

Server state aware algorithms

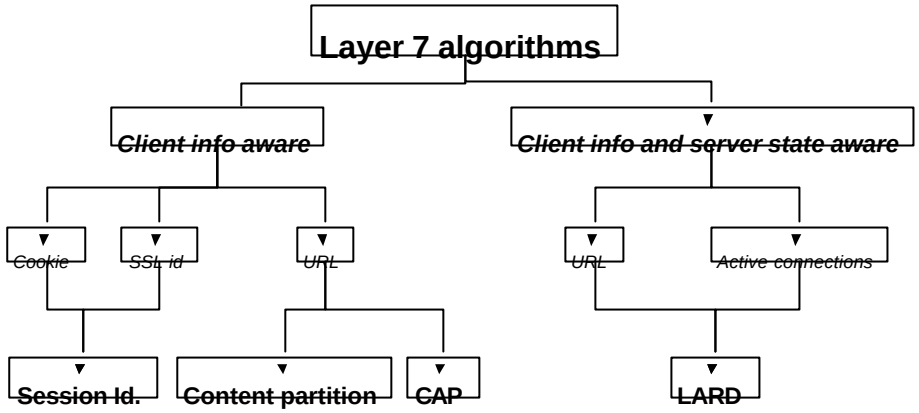
- **Request assignment based on server load info**
  - **Least loaded server** (← AVOID!)
  - **Weighted Round-Robin** (**WRR**)
    - it allows configuration of weights as a function of server load

Server state aware algorithms

- **Possible metrics to evaluate server load**
  - **Input metrics**: information get by the Web switch without server cooperation, e.g., *Active connections*
  - **Server metrics**: information get by the Web servers and transmitted to the Web switch, e.g., *CPU/Disk utilization, response time*
  - **Forward metrics**: information get directly by the Web switch, e.g., *emulation of requests to Web servers*



Layer 7 Web switch algorithms



Client info aware algorithms

- **Session identifiers**
  - HTTP requests with same ***SSL id*** or same ***cookie*** assigned to the same server
    - Goal: avoid multiple client identifications for the same session
- **Content partition**
  - Content partitioned among servers according to **file type** (HTML, image, dynamic content, audio, video, …)
    - Goal: use specialized servers for different contents
  - Content partitioned among servers according to **file size** (Thresholds may be chosen dynamically.)
    - Goal: augment load balancing
  - File space partitioned among the servers through a hash function
    - Goal: improve *cache hit rate* in Web servers

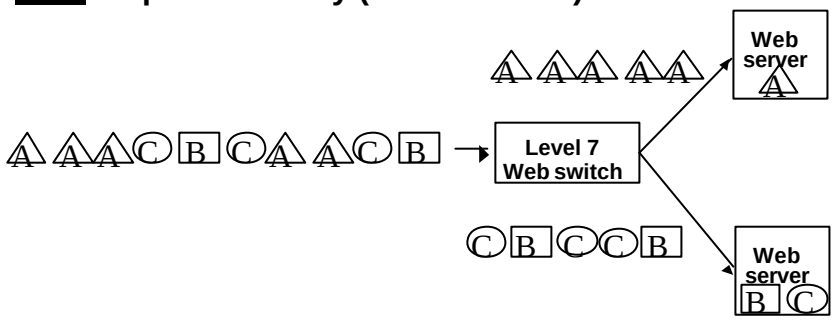
Client info aware algorithms

- **Content Aware Partition** (CAP – *Proc. WWW 2000*)
  - Resource classification according to the impact of HTTP requests on main Web server components, e.g.,
    - *Low impact* (small-medium static files)
    - *Network bound* (large file download)
    - *Disk bound* (database queries)
    - *CPU bound* (“secure” requests)
  - Cyclic assignment of each class of requests to Web servers
  - Goal: augment load sharing of *component bound* requests among Web servers

Client and server state aware algorithm

Locality-Aware Request Distribution (LARD)

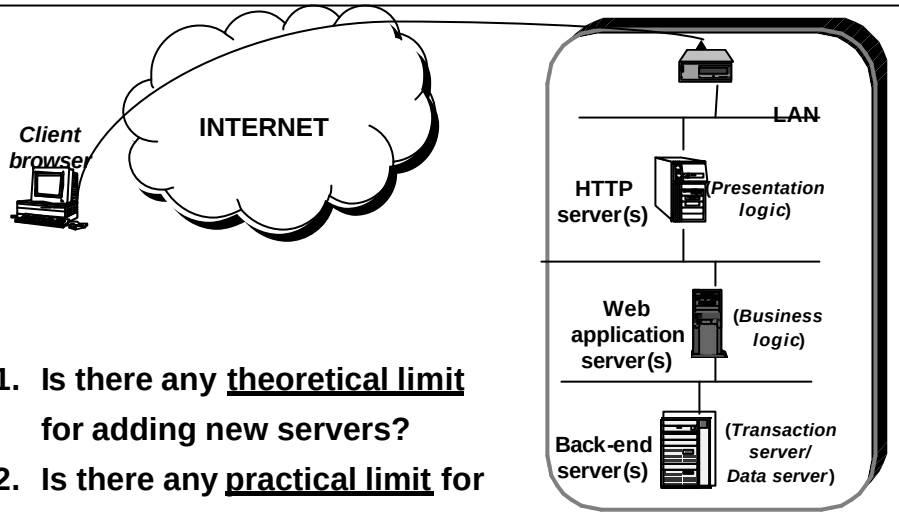
- **First request for a given target assigned to the least loaded server (metrics: *number of active connections*)**
- **Subsequent requests for the same target assigned to the previously selected server**
- **<u>Goal</u>: improve locality (*cache hit rate*) in server cache**



Final considerations (Web clusters)

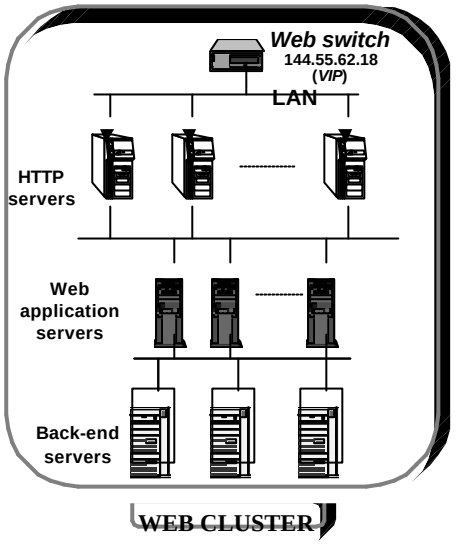
- **Alternative architettures**
  - Layer-4 Web switch (*Content information blind*)
  - Layer-7 Web switch (*Content information aware*)
- **Main pros**
  - Fine grain control on dispatching of client requests
  - High internal *availability*
  - High security
  - High performance (“infinite” performance?)

Web cluster “power”



1. **Is there any <u>theoretical limit</u> for adding new servers?**
2. **Is there any <u>practical limit</u> for adding new servers?**

Limits?



- **No limit at the Presentation level**
- **No limit at the Business level**
- **Some limits at the Data level**

➔ **NO LIMIT ??**
➔ **INFINITE POWER ??**

Web cluster cons

- **Single points of failure**
  - Internet connection
  - Web switch
- **Maximum scalability bounded by**
  - Web switch capacity
  - Internet access bandwidth

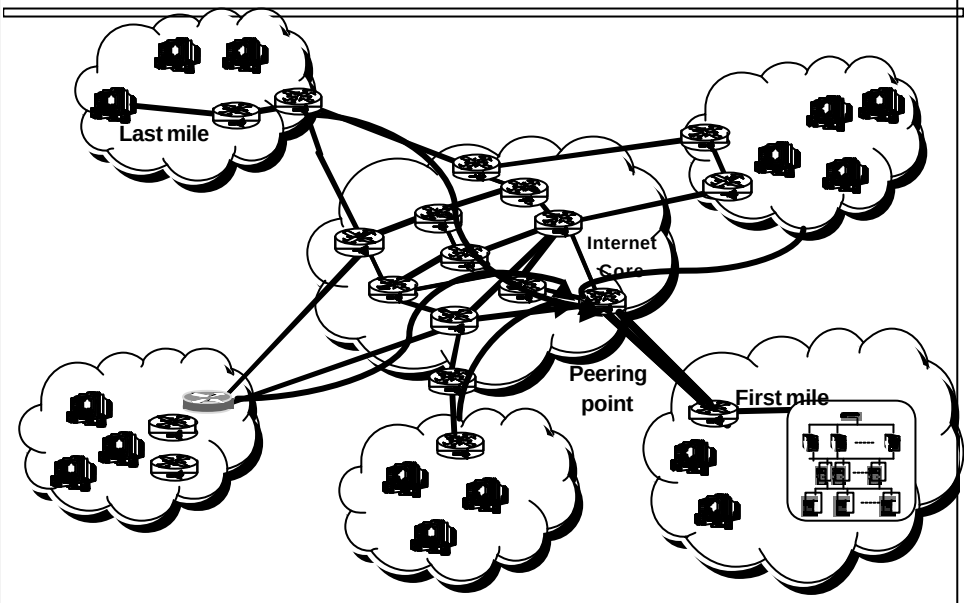
System possibilities and *network* limits

- **Web cluster throughput:**

  5 Mbps → ~~85 Mbps~~ → W

- **Network throughputs:**
  - **T1** - **T2**: Large company to ISP → 3.1/6.3 Mbps
  - **T3** - **OC1**: ISP to Internet infrastructure → 44.7-51.8 Mpbs
  - **OC3**: Large company backbone → 155.5 Mbps
  - **OC12** - **OC256**: Internet backbones → 0.62-13.2 Gbps

Network bottlenecks



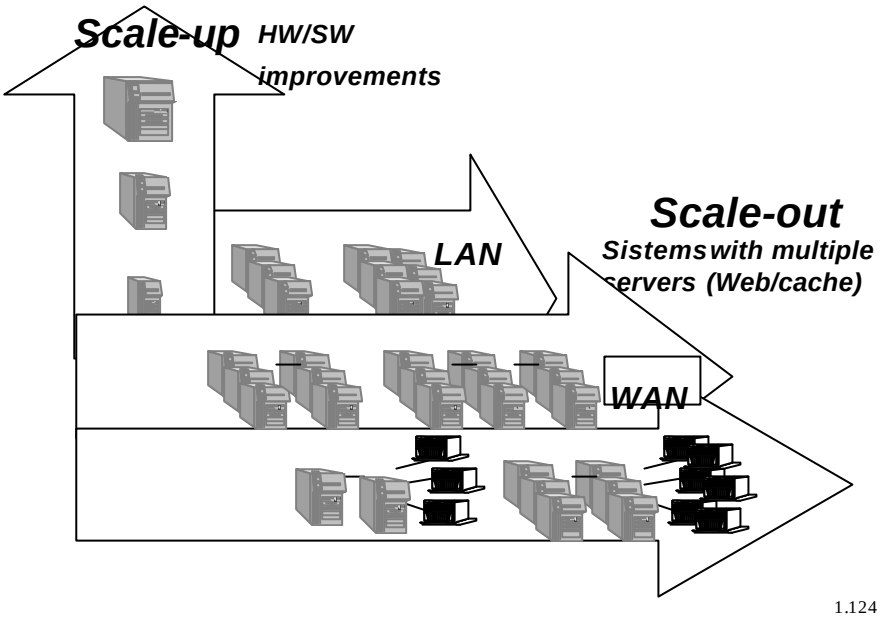
Managing a popular Web site

- **If your Web site is really popular**
  - You can afford larger investments
  - You cannot take <u>any kind of risks</u> about availability
    - Network
    - Electrical power
    - Road works
    - Bad weather
    - Unfaithful employee
    - …

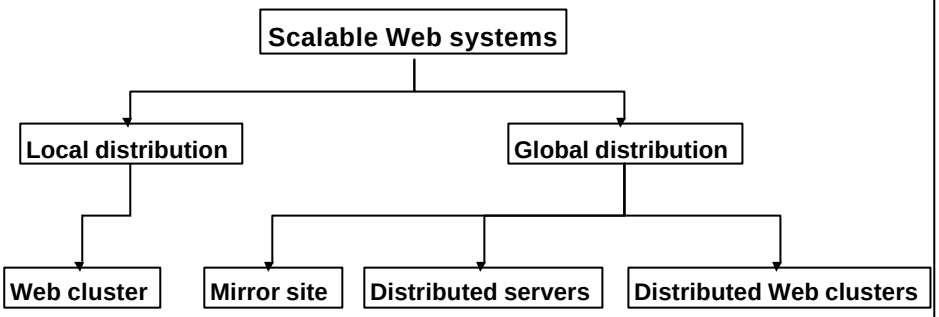
SYSTEM

(*GEOGRAPHICAL DISTRIBUTION*)

Pressure on Web-based architectures



Taxonomy



Mirror site

- ***Information that is geographically replicated on multiple Web sites***

- **Web site addresses**
  - **Multiple hostnames** (e.g., “www.site1.com”, “www.site2.com”, …, “www.site*N*.com”)
  - **One IP address for each site**

Dispatching left to users

Mirror sites

Mars Polar Lander Mission

Public Sector Mirror Sites

Mirror site	Site Address	Load Capacity
SDSC - USA	http://mars.sdsc.edu	Bandwidth
Internet2 - USA	http://mars.dsi.internet2.edu	Bandwidth
NCSA - USA	http://www.ncsa.uiuc.edu/mars	Bandwidth
Mars Society - USA	http://missions.marssociety.org/mpl	Bandwidth
KSC - USA	http://www.ksc.nasa.gov/mars	Bandwidth
HIGP - USA	http://mars.pgd.hawaii.edu	Bandwidth

Simple architecture, but

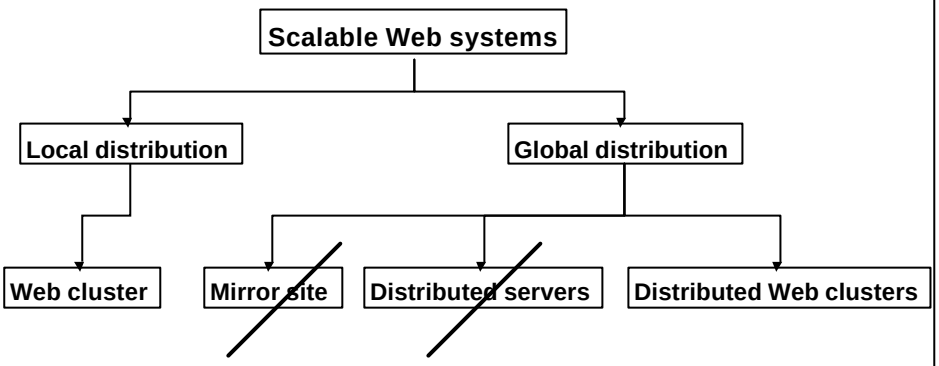
- Visibly replicated architecture
- It is very hard to maintain information consistency among mirror sites
- No way of controlling load distribution
- Hard to trace users

Geographically distributed servers

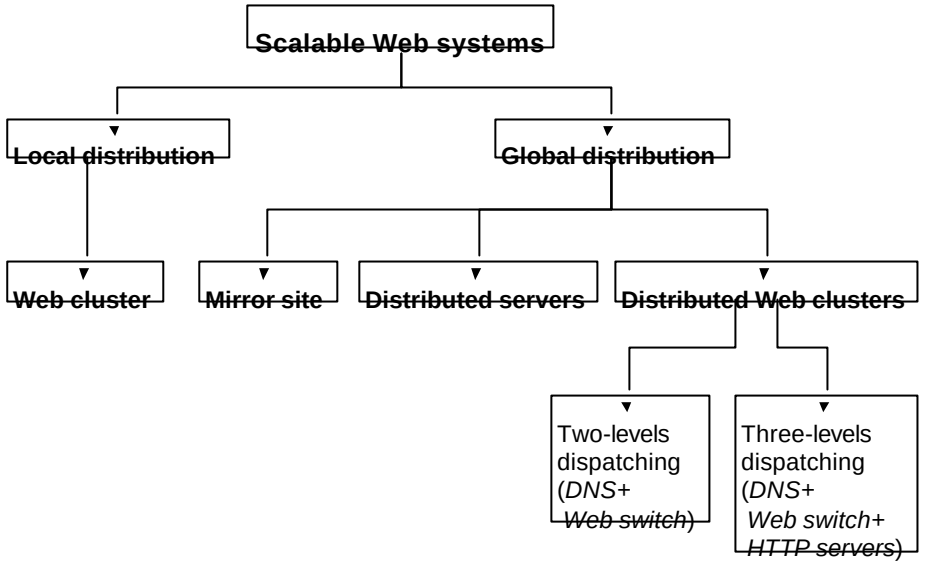
- **If your Web site is really popular**

Do you think is a good choice to spread single servers around the world?

Taxonomy



Taxonomy



Distributed Web clusters

- ***Web site realized on an architecture of geographically distributed Web clusters***
- **Web site addresses**
  - One hostname (e.g., "www.site.com")
  - One IP address for each Web cluster

First level dispatching

Authoritative DNS **or** other entity **during the lookup phase**

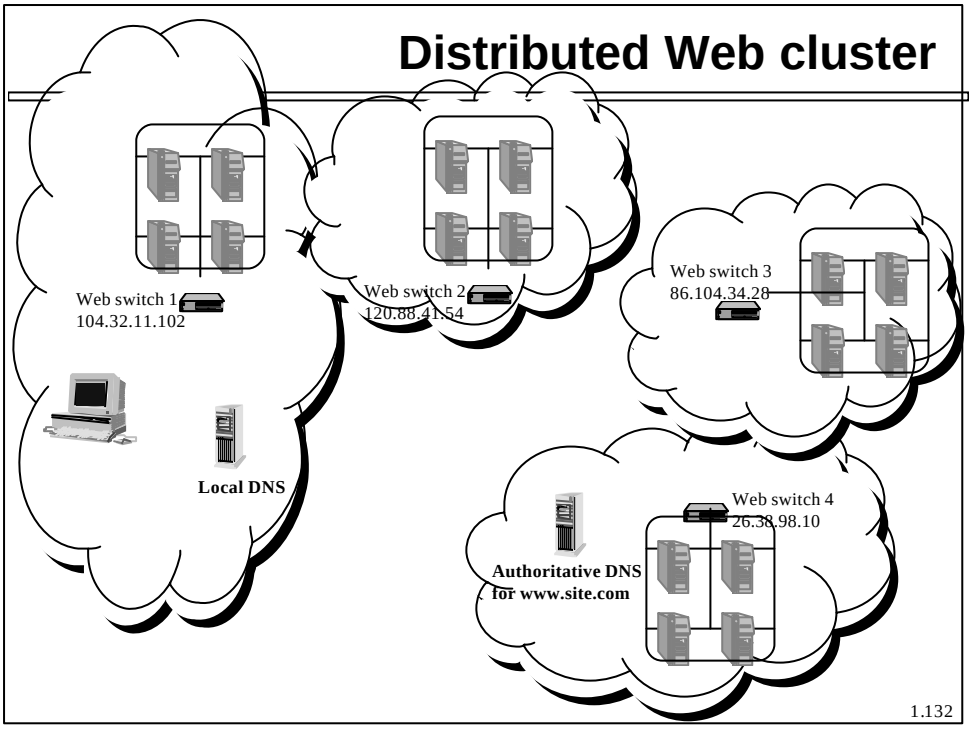
Second level dispatching

Web switch **of the Web cluster selects one server**

Third level dispatching

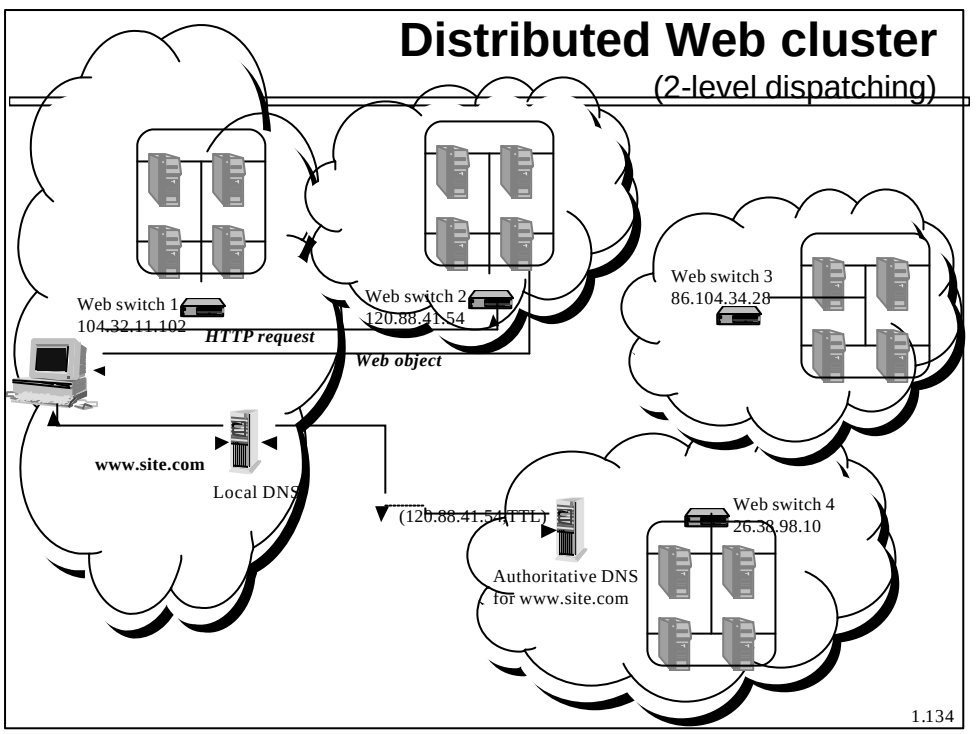
Each Web server **may redirect the received request to another server**

Distributed Web cluster



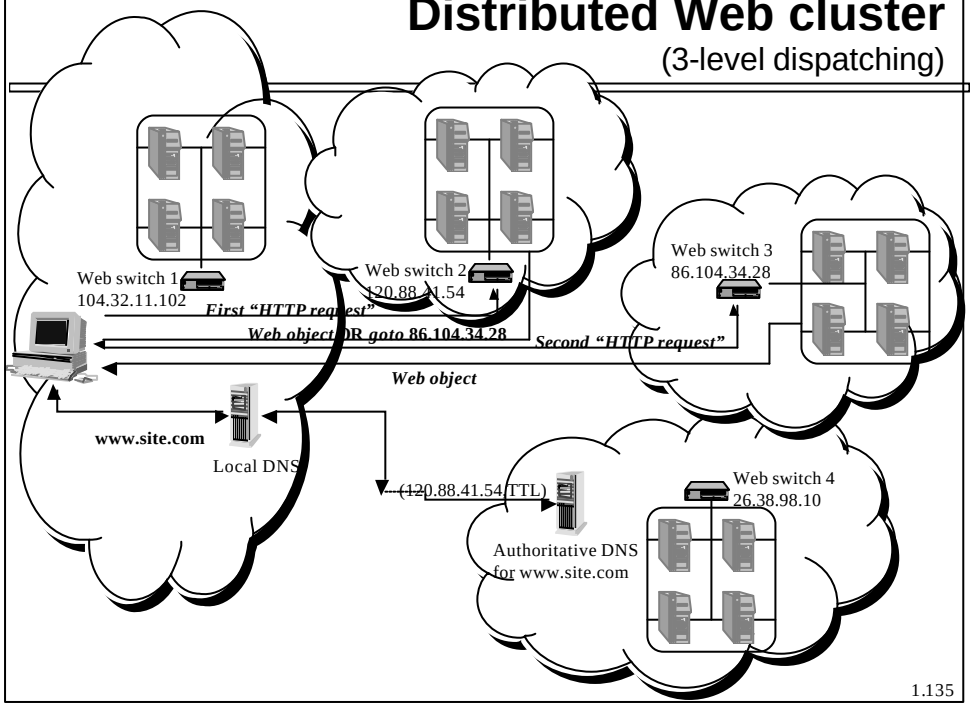
Dispatching mechanisms

1. **DNS dispatching -- global**
2. **Web switch dispatching -- local**
3. **HTTP dispatching (*HTTP redirection*) -- local**



Distributed Web cluster

(3-level dispatching)



DNS dispatching

- **The** distributed Web cluster **architectures implement global dispatching by intervening in the *lookup phase* of the address request:**
  - A client asks for the **IP address** of a Web server corresponding to the **hostname** in the URL
  - If the hostname is valid, it receives the couple

    (IP address, TimeToLive)

- **The *enhanced authoritative DNS* of the Web site (or *another entity* that replaces or integrates the authoritative DNS) can use various dispatching policies to select the "best" Web cluster**

Issues of DNS dispatching

Typical issues

- **Load spikes in some hours/days**

Additional issues

- **Traffic depending on time zones**
- **Client distribution among Internet zones**
- **Proximity between client and Web server**

- Caching of [hostname-IP] at intermediate DNS servers for TTL interval

Issues of DNS dispatching

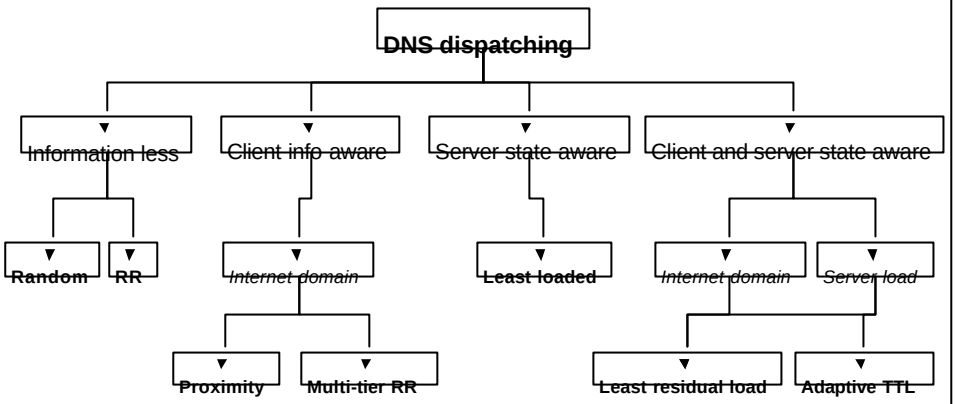
- **Because of *hostname - IP address* caching, the DNS of highly popular Web sites controls only 5-7% of traffic reaching the servers of the site [IBM source data]**
- **Unlike Web switch (controlling 100% traffic), the DNS should use sophisticated algorithms**

- **Nevertheless, all CDN architectures use some sort of DNS dispatching**

Actions on TTL values

- **Constant TTL**
  - Set TTL=0 to augment DNS control
  - Drawbacks
    - Not cooperative DNSes (name servers may ignore very small TTL values <300 seconds)
    - Browser caches
    - Risk of overloading authoritative DNS
- **Adaptive TTL**
  - Tailor TTL value adaptively for each address request by taking into account the popularity of client Internet domain and Web server loads

DNS dispatching algorithms



Internet proximity

- Internet proximity is an interesting open issue
  Client-server *geographic proximity* does not mean *Internet proximity* (*round trip latency*)
  - ***Static* information**
    - client IP address to determine Internet zone (geographical distance)
    - hop count ("stable" more than "static" information)
      - *network* hops (e.g., `traceroute`)
      - *Autonomous System* hops (routing table queries)

It does not guarantee selection of the best connected Web server, e.g., "links are not created equal"

Internet proximity

- *Dynamic* evaluation of proximity
  - round trip time (e.g., `ping`, `tcping`)
  - available link bandwidth (e.g., `cprobe`)
  - latency time of HTTP requests (request emulation)

Additional time and traffic costs for evaluation

A related open issue:

***Correlation between hop count and round trip time*?**

- "Old" measures: close to zero [Crovella 95]
- "Recent" measures: strong, reasonably strong

Addressing DNS dispatching issues

- Utilizing multiple level DNS dispatching
  - CDN choice [-->]
- **Integrating DNS dispatching with HTTP server or Web switch global dispatching:**
  - **HTTP redirection**
  - **IP tunneling**

HTTP redirection

- **The redirection mechanism is part of the HTTP protocol and is supported by current browser and server software**
- **DNS and Web switch use centralized scheduling disciplines**
- **Redirection is a distributed scheduling policy, in which all Web server nodes can participate in (re-)assigning requests**
- **Redirection is completely transparent to the user (**not to the client!**)**

HTTP redirection

message header
HTTP OK status code
302 - "Moved temporarily" to a new location

- **"New location"**
  - Redirection to an IP address (**better performance**)
  - Redirection to an hostname

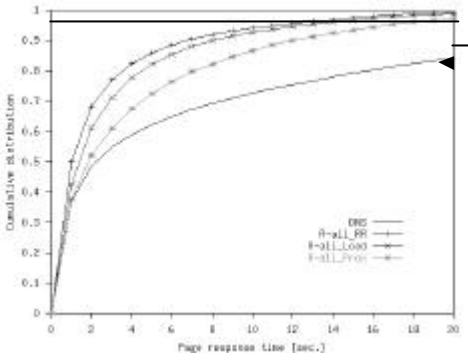
Redirection policies - *alternatives*

- **Trigger mechanism**
  - Centralized: DNS or other entity
  - Distributed: any Web server (typically when highly loaded)
- **Selection policy (page requests to be redirected)**
  - all page requests (**All**)
  - all page requests larger than a threshold (**Size**)
  - all page requests with many embedded objects (**Num**)
- **Location policy (choice of the new target Web cluster)**
  - Round Robin (**RR**)
  - Hash function
  - Least loaded server (**Load**)
  - Client-to-server proximity (**Prox**)

Performance comparison

- Two-levels ***vs.*** Three-levels dispatching
- **Which** *selection* **policy?** *Redirect-All* ***vs.*** *Redirect-Heavy*
- **Which** *location* **policy?**
- **Dispatching algorithms**
  - Level 1 (*DNS*): proximity
  - Level 2 (*Web switch*): Weighted Round Robin
  - Level 3 (*Web servers*)
    - Selection policy ("*which page requests have to be redirected?"*):
      - Redirect all requests **(All)**
      - Redirect heavy requests: **Size**, number of embedded objects (**Num**)
    - Location policy (*"towards which cluster?"*):
      - RoundRobin (**RR**)
      - Least loaded cluster (**Load**)
      - Cluster proximity (**Prox**)

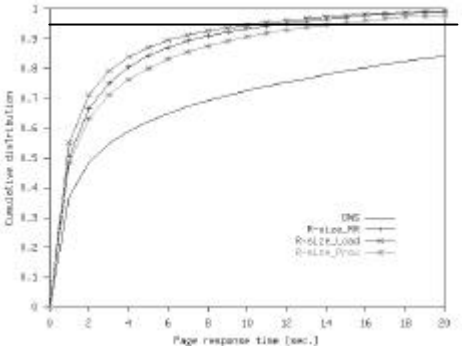
Performance results



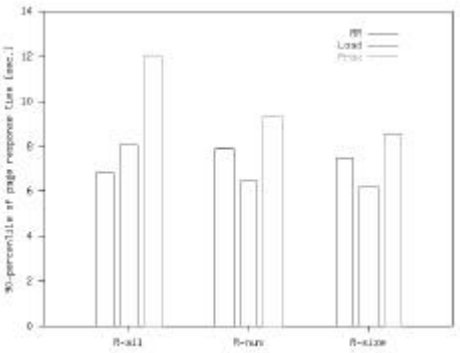
2 level dispatching

Redirect-all policy

Size-based redirection

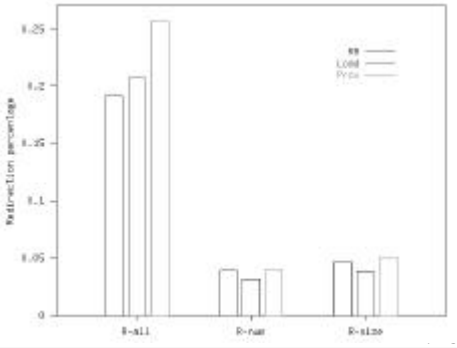


Performance results (2)



Response time (90-percentile)

Percentage of redirected requests



Summary

- **A third-level dispatching mechanism**
  - guarantees immediate actions to shift the load away from an overloaded Web cluster
  - augments the Quality of Web-based Services because the percentage of client requests with guaranteed response time is much higher
- ***HTTP redirection* is easy to implement but works for HTTP services only**
- ***IP tunneling* can be used when it is necessary to provide other services**
- **Limiting redirection to the "heaviest" requests (Redirect-Heavy) reduces the percentage of redirection to a small fraction of requests (about 5%) without deteriorating the response time achieved by Redirect-All policies**

Comparison

Distributed Web cluster (*two-level dispatching*)

- **High control on load reaching the Web cluster**
- **Slow reaction to an overloaded Web cluster**

Distributed Web cluster (*three-level dispatching*)

- **Immediate actions to shift the load away from an overloaded Web cluster**
- **Redirection valid only for HTTP services**

Summing up

Scalable Web-server systems are based on multiple server platforms

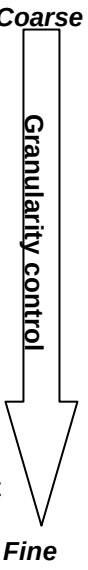
– A **dispatching mechanism** to direct the client request to the “best” server

– A **dispatching algorithm** to define the “best” server

– An **executor** to carry out the scheduling algorithm and the relative mechanism

Web dispatching mechanisms

Mechanism	*Executor*	*Item*
Hostname resolution	**DNS / Other entity**	**Session**
- *Local dispatching*		
- Global dispatching		
HTTP redirection	**Web server**	**Page request**
- *Local dispatching*		
- Global dispatching		
Packet redirection	**Web switch**	**Hit / Page request**
- Local dispatching		



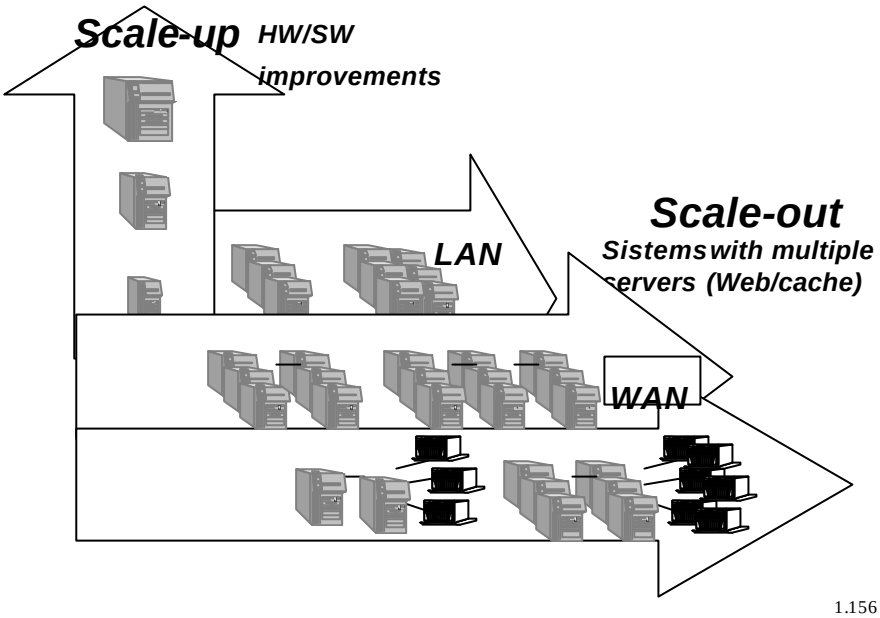
Web dispatching algorithms

- **Static** (*information-less*)
- **Dynamic**
  - client info aware
  - server state aware
  - client info and server state aware
- **Adaptive** (not yet investigated)

SYSTEM

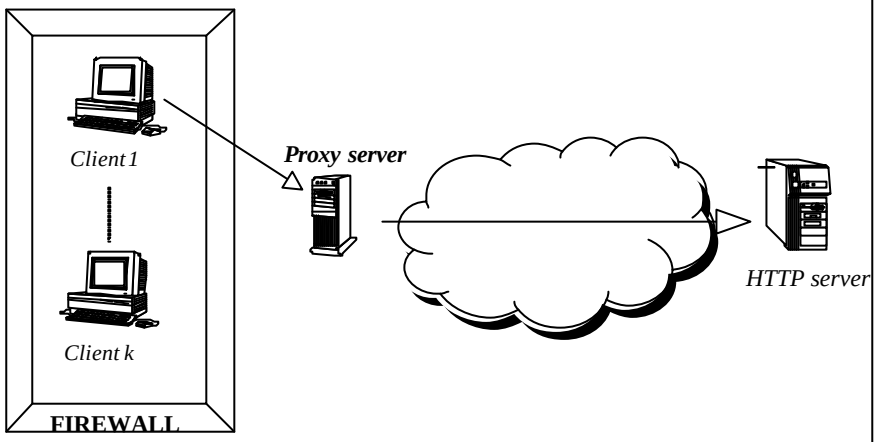
(*AFTER REPLICATION, CACHING*)

Pressure on Web-based architectures



Proxy origin

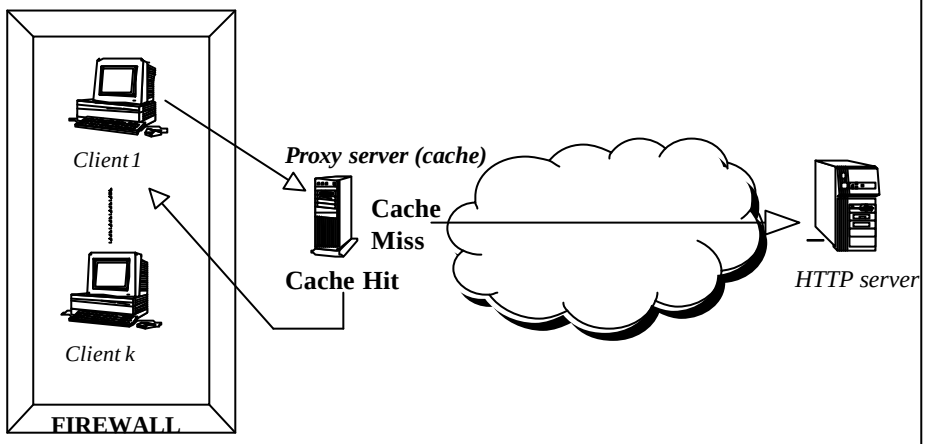
- **Intranet *gateway***
- **After, *cache repository***



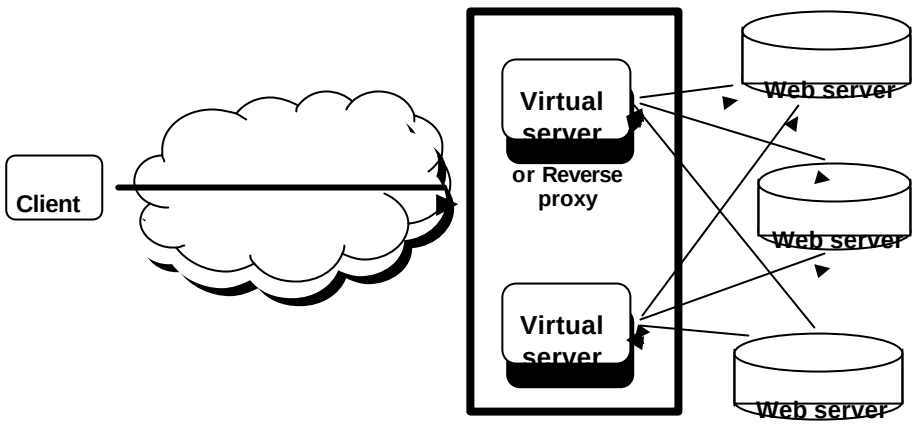
Caching mechanisms

- **CONTENT CONSUMER**
  - Proxy server
- **CONTENT PROVIDER**
  - Web cluster + Reverse proxy
  - Web cluster + Reverse proxy
- **THIRD PARTIES**
  - **ISP (*content consumer side*) → cooperative proxy servers**
  - **Companies (*content provider side*) → Content Delivery Network (CDN)**

CONTENT CONSUMER

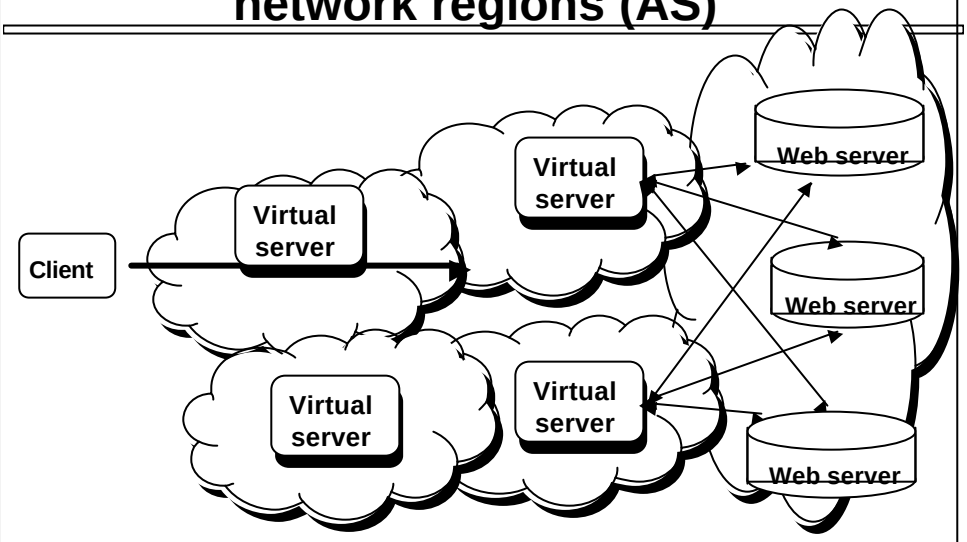


Virtual server / Reverse proxy
(*CONTENT PROVIDER* side)



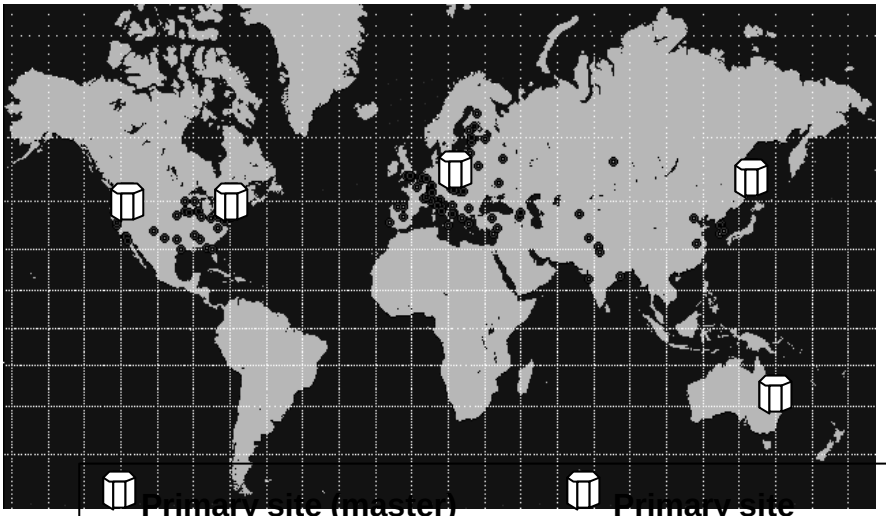
Pushing techniques

Possibility of spreading VS in different network regions (AS)



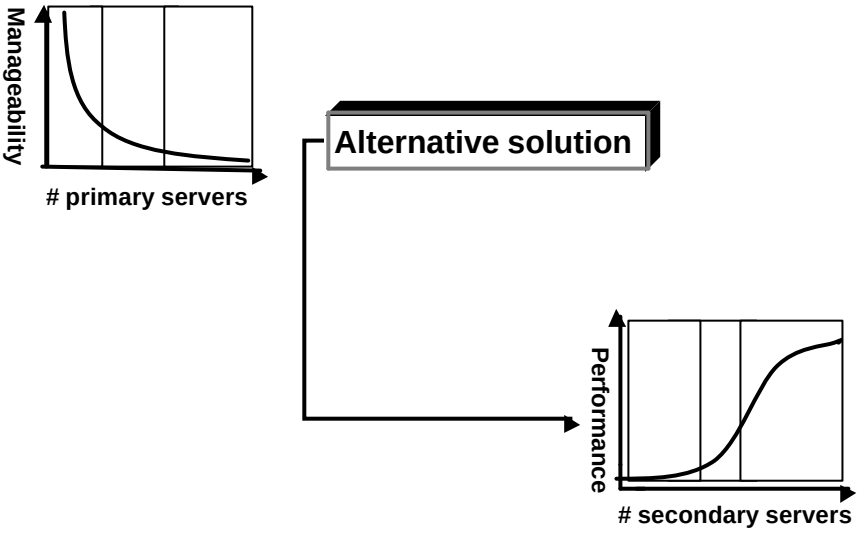
1.161

Architecture for 1 million hits per minute

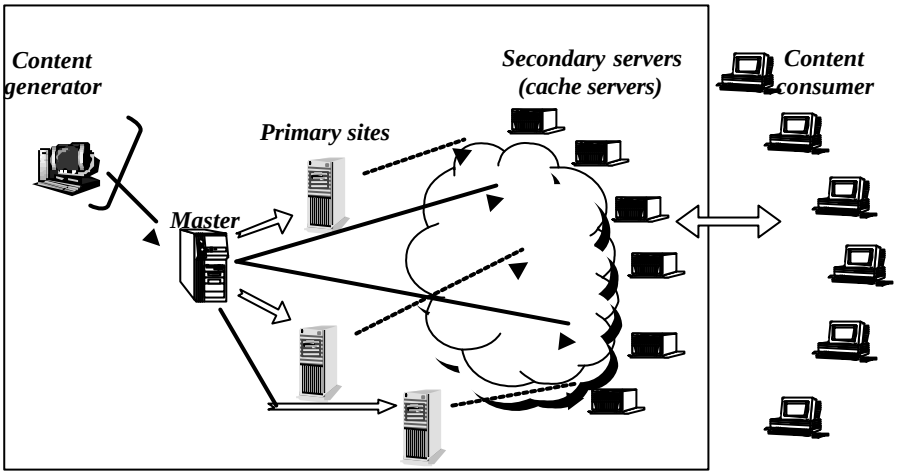


1.162

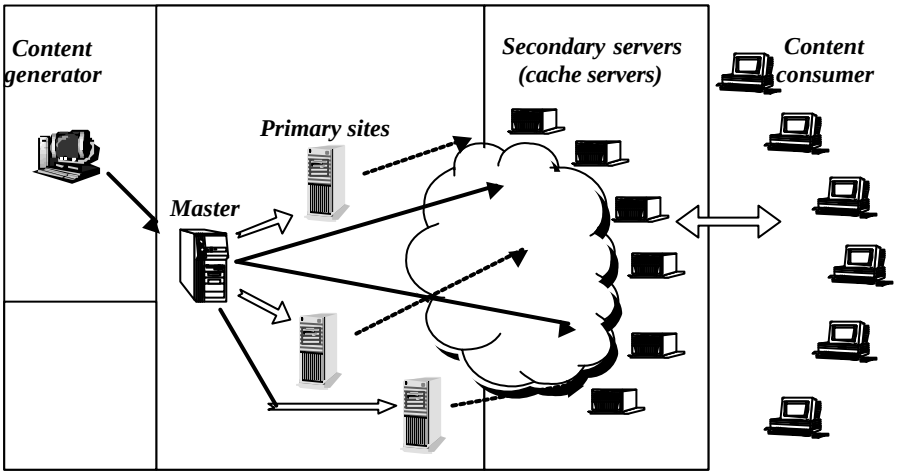
Why few primary sites and thousands of secondary servers?



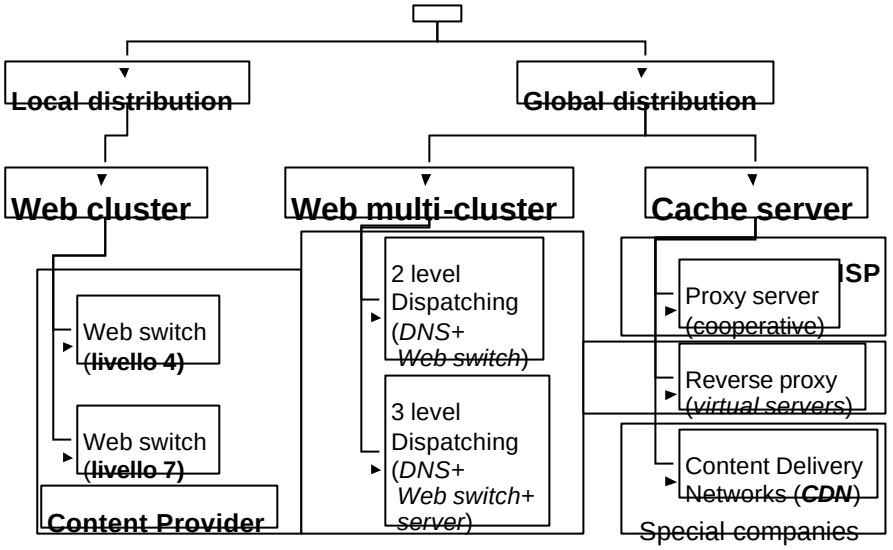
Push model
("Content provider" manages everything)



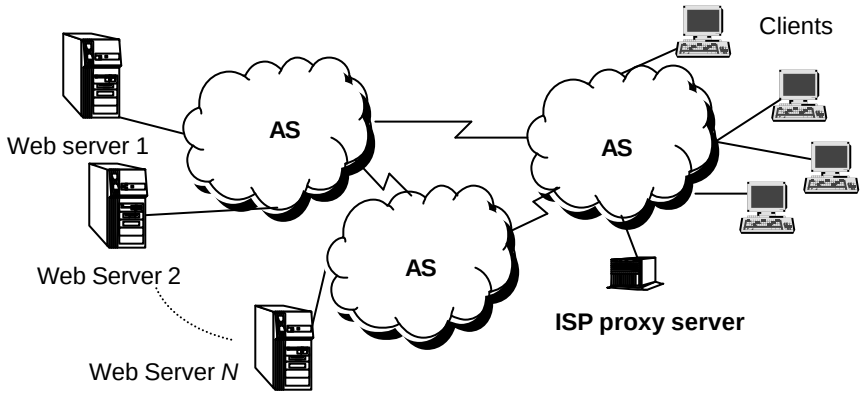
Modello push
(altre ipotesi di gestione)



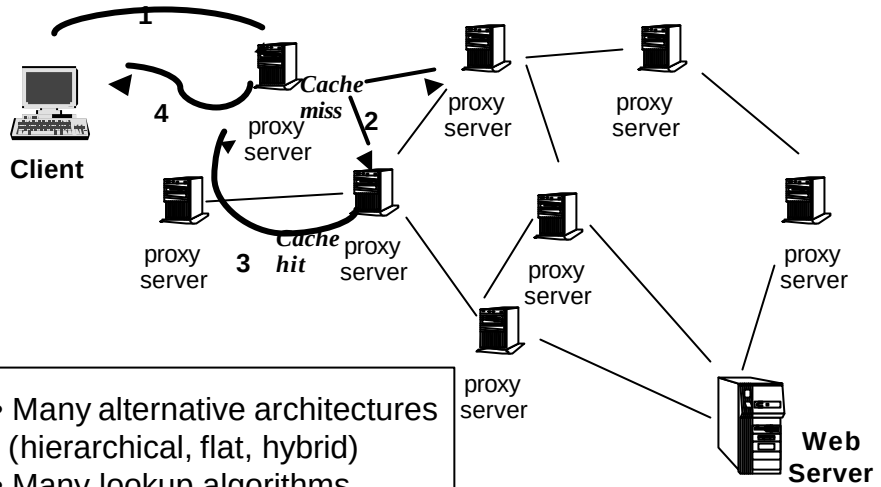
Replication and Caching



Web caching – ISP

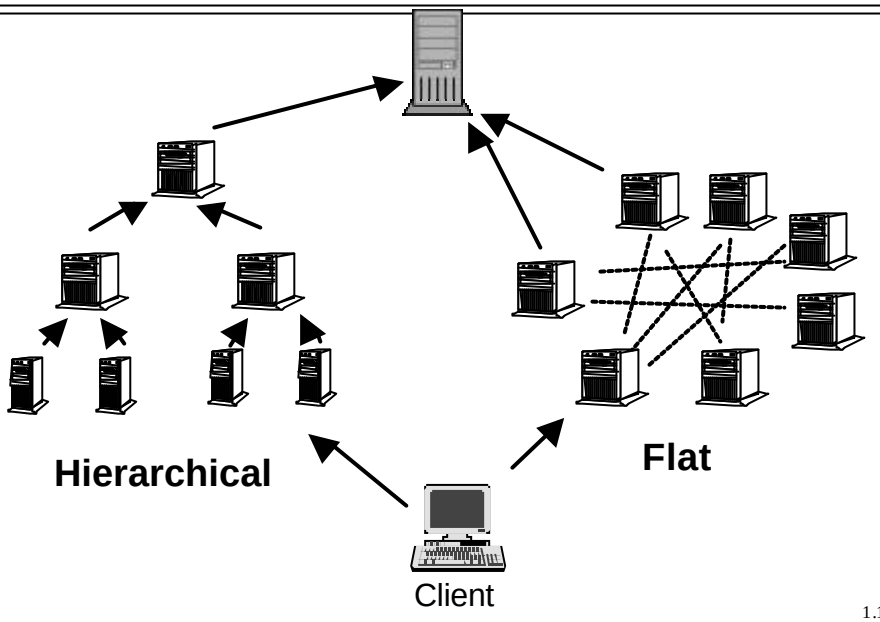


Cooperativo Web caching (ISP)



- Many alternative architectures (hierarchical, flat, hybrid)
- Many lookup algorithms
- Many cooperation algorithms

Hierarchical *vs.* Flat scheme



"Possible" advantages of Web caching

- **Latency reduction**
- **Less traffic**
- **Less overhead on Web sites**

But, many studies show that generalized Web caching has limited benefits (low *cache hit rates*)

Business alternative: *content delivery*

"Content" delivery [1] { **information delivery** / **data delivery**

***Content* = digital material for which "someone" (*user*, *ISP*, *content provider*, …) can spend money**

Exemples

- Web embedded objects
- video-on-demand
- *pay-per-download* music
- software distribution
- *pay-per-use* software

Web caching

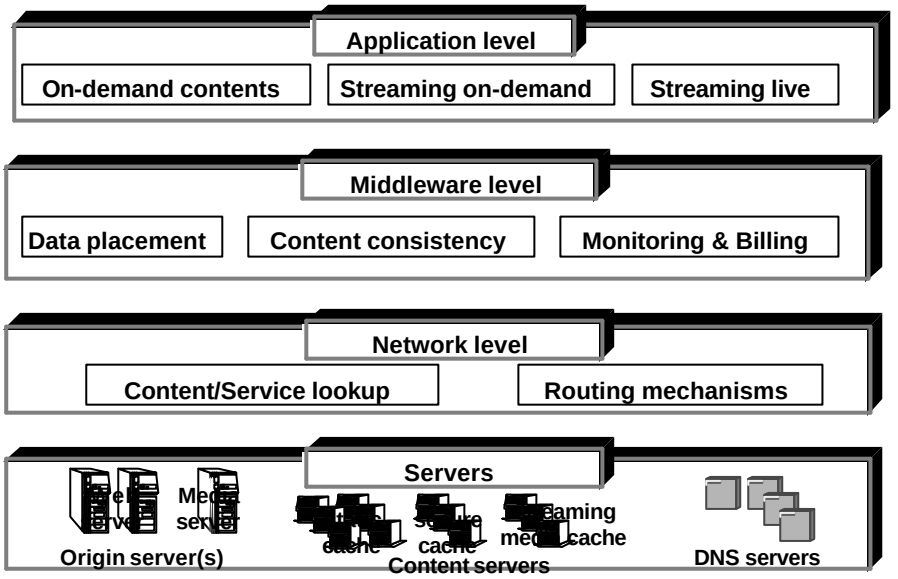
CDN

- **Cooperative cache servers**
- **Cache servers distributed over many Internet regions (AS)**

- **Web caching is independent of Web site management**
- **Generalized caching: content of any Web site**
- **ISP can pay because they save bandwidth (content provider don't)**

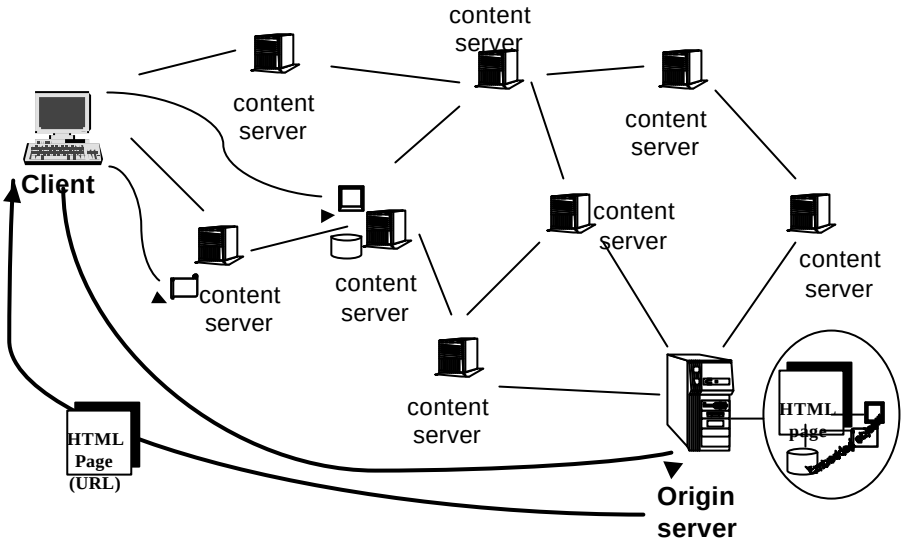
- **Content provider outsources content distribution problems to CDN company**
- **Selective caching: just content of "customer" Web sites**
- **Content providers pay (ISP don't)**

CDN Architecture



1.173

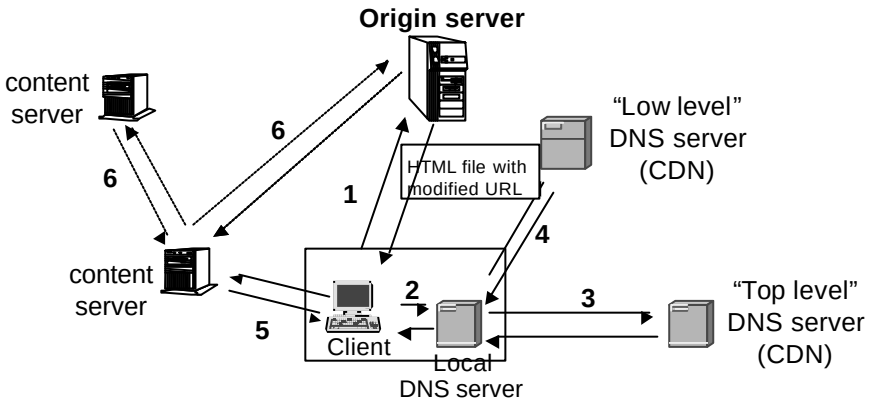
CDN routing through URL rewriting



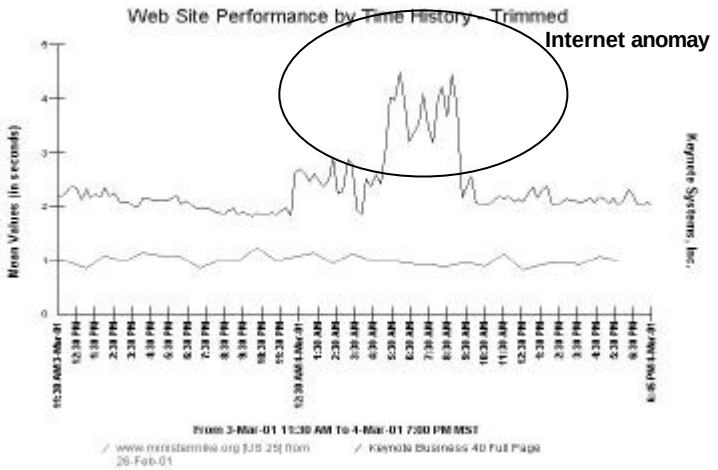
1.174

Akamai example

- **Twofold DNS resolution**

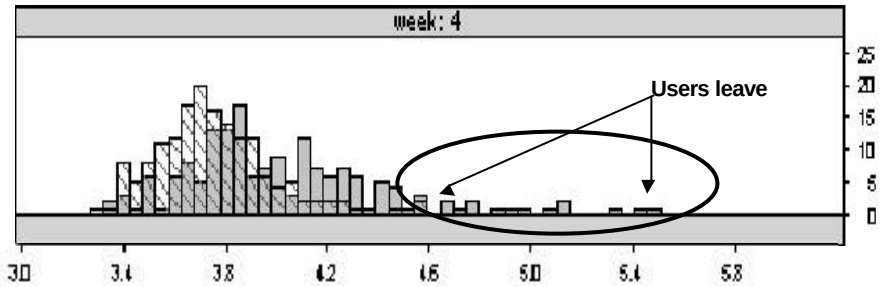


Does a CDN work? (their measures #1)



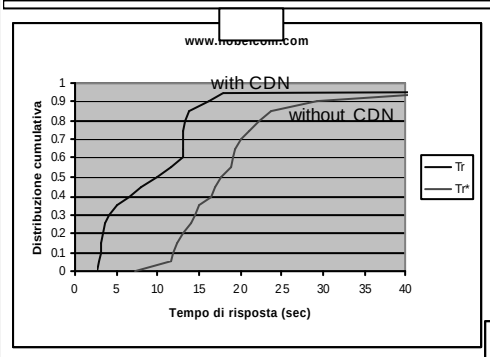
Does a CDN work? (their measures #2)

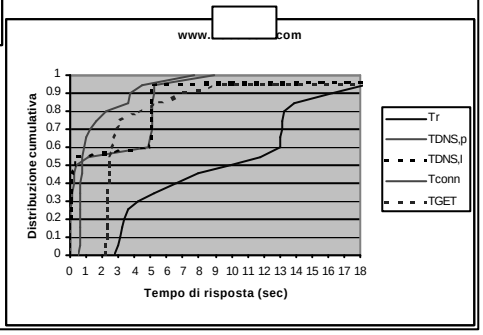
Stripe: with CDN
Cyan: without CDN



Source: Keynote Systems, Inc.

Does a CDN work? (our measures #1)

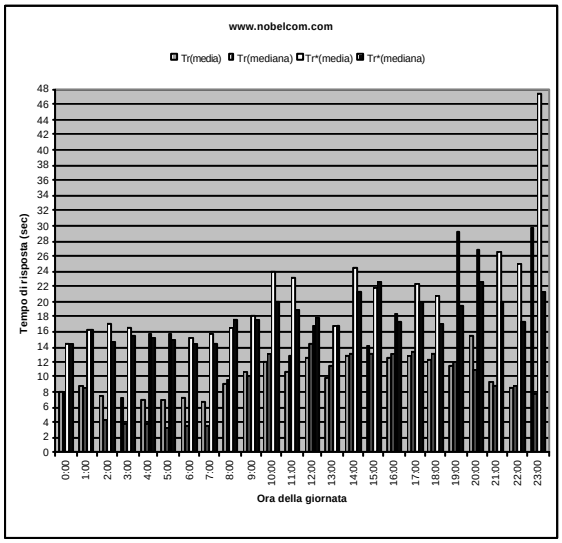




HTTP GET method is the larger contribution to the response time

Bimodal effects for DNS resolution

Does a CDN work? (our measures #2)



CDNs work!

They can have limited effects on clients on narrow bandwidth connections