Scalable Web-Server Systems: Architectures, Models and Load Balancing Algorithms

Michele Colajanni

University of Modena, Italy, colajanni@unimo.it

Philip S. Yu

T.J. Watson Research Center, NY, psyu@us.ibm.com

Valeria Cardellini

University of Roma Tor Vergata, Italy, cardellini@ing.uniroma2.it

Tutorial goals

- Overview of issues (and possible solutions) to be considered when analyzing the performance of Web transactions

- Overview of scalable Web-server systems
  - Focus on **locally** distributed solutions
  - Focus on **globally** distributed solutions

- Overview of scheduling algorithms and performance comparison

- Identification of key design alternatives

Tutorial outline

- **Part 1**
  - Motivations
  - Workload characterization
- **Part 2**
  - A taxonomy of scalable Web-server systems
  - A taxonomy of scheduling algorithms
- **Part 3**
  - Locally distributed systems
- **Part 4**
  - Globally distributed systems
- **Part 5**
  - Case study
  - (*A look at*) other solutions for scalable Web services

What this tutorial does not cover

Other solutions to improve Web performance:

- **Caching**
  - Proxy caching [*largest literature on Web*, e.g. Bar00]
  - Web server caching, e.g. [Iye00a, Son00]
- **Reverse proxy servers**, e.g. [Luo98]
- Specialized Web servers and **multimedia servers**, e.g. [Lie98, Cho00]
- **Client side solutions**, e.g. [Mos97, Yos97, Kar98, Car99a, Vin00]

Part 1

Motivations, Quality of Web Services, Web workload

Outline (Part 1)

- **Motivations**
  - Popular Web sites
  - Quality of Web Service (QoWS)
  - Web performance problems
- **Workload characterization**
  - Web drivers
  - Analysis of a Web transaction
  - Results from literature
- **Possible improvements**
  - Network
  - Web-server system
  - Web infrastructure

Motivation 1: Popular Web sites

[Load measures in hits]

Yahoo, Netscape, Lycos, Pointcast, AltaVista, CNN, … (*>40 Million hits/day*)

Event	Period	Peak day	Peak minute
NCSA server (Oct. 1995)		2 Million	
Olympic Summer Games (Aug. 1996)	180 Million	8 Million	
Presidential US Election (Nov. 1996)		9 Million	
NASA Pathfinder (July 1997)	942 Million (14 days)	40 Million	
Olympic Winter Games (Japan, 1998)	634.7 Million (16 days)	57 Million	110,000
FIFA World Cup (France, 1998)	1,350 Million (90 days)	73 Million	209,000
Wimbledon (July, 1999)	942 Million (14 days)	125 Million	430,000
Olympic Games 2000	???	???	???

Motivation 2: Web has new requirements

First generation

- An economic channel for not critical information
- 90 percent of information represented by text and some images [Arl97]
- Occasional maintenance and updating
- Highly variable performance
- No guarantee on availability
- Security not important

Second generation

- An important channel for critical information
- Always larger percentage of dynamic content
- Direct or *indirect* (say, publicity) costs
- Companies are evaluated even on the basis of their Web site

→ **Quality of Service**

Quality of Service

- **Quality of Network Service (QoNS)**

- **Quality of Web Service (QoWS)**

How to measure

- Choose a service
- Choose a metrics (e.g., *response time*, *throughput*)
- Choose a maximum value X

- **NO**
  - average among observed values for that service less than X

- **YES**
  - all observed values less than X
  - 90 or 95-percentile of observed values less than X

Quality of Network Service

- **Network quality**
  - guaranteed latency in large networks
- **Service quality**
  - network availability

Service Level Agreement: An example

- Round-trip less than 85ms for connections intra-Europe and intra-North-America
- Round-trip less than 120ms for connections between Europe and North-America
- "… If we fail to meet the SLA guarantee in two consecutive months, we will automatically credit one day of the monthly fee for the service which has not been met …"

Quality of Web Service (QoWS)

- **Availability** (*System* measure)
- **Performance** (*Service* measure - *percentile* metric)
- **Security** (*System/service* measure - *binary* metric)
- **Accessibility** (*System/service* measure - *binary* metric)

Service measures **typically apply to a subset of Web services provided by the Web system.**

Binary metrics denote a "quality" that is guaranteed or not.

Quality of Web Service (QoWS)

- **Availability**
  - *Service Level Agreement*: **Web system must be available for X% of times,** e.g.,
    - X = 99% → **7.2** hours/month downtime
    - X = 99.9% → **43** minutes/month downtime
    - X = 99.999% → **26** seconds/month downtime

- **Performance**
  - *Service Level Agreement*: **X% of (all or subset of) Web requests must have a response time less than Y seconds**. Typical measures are 90- or 95-percentile, e.g.,
    - 95% of the requests must have a response time less than 4 seconds

QoNS *vs.* QoWS

- “Less than 5 percent of organizations set and measure SLAs for distributed application availability and performance” (Gartner Group docs.) “Network carriers do”

Some motivations

- Network carriers control their backbones
- Web solutions can be applied only to some parts of the infrastructure that depend on the role of the company, e.g.,
  - Web infrastructure component (e.g., *cooperative proxy caching*)
  - Web site architecture
  - No control on clients (but for Intranet)
- The Web is changing rapidly and standards are still evolving.

Choices for QoWS

- **Differentiated Web services**
  - Define classes of users/services
  - Choose the number of priority levels
  - Guarantee different QoWSes through priority scheduling disciplines, e.g. [Pan98, Vas00]
  - Monitor for starvation

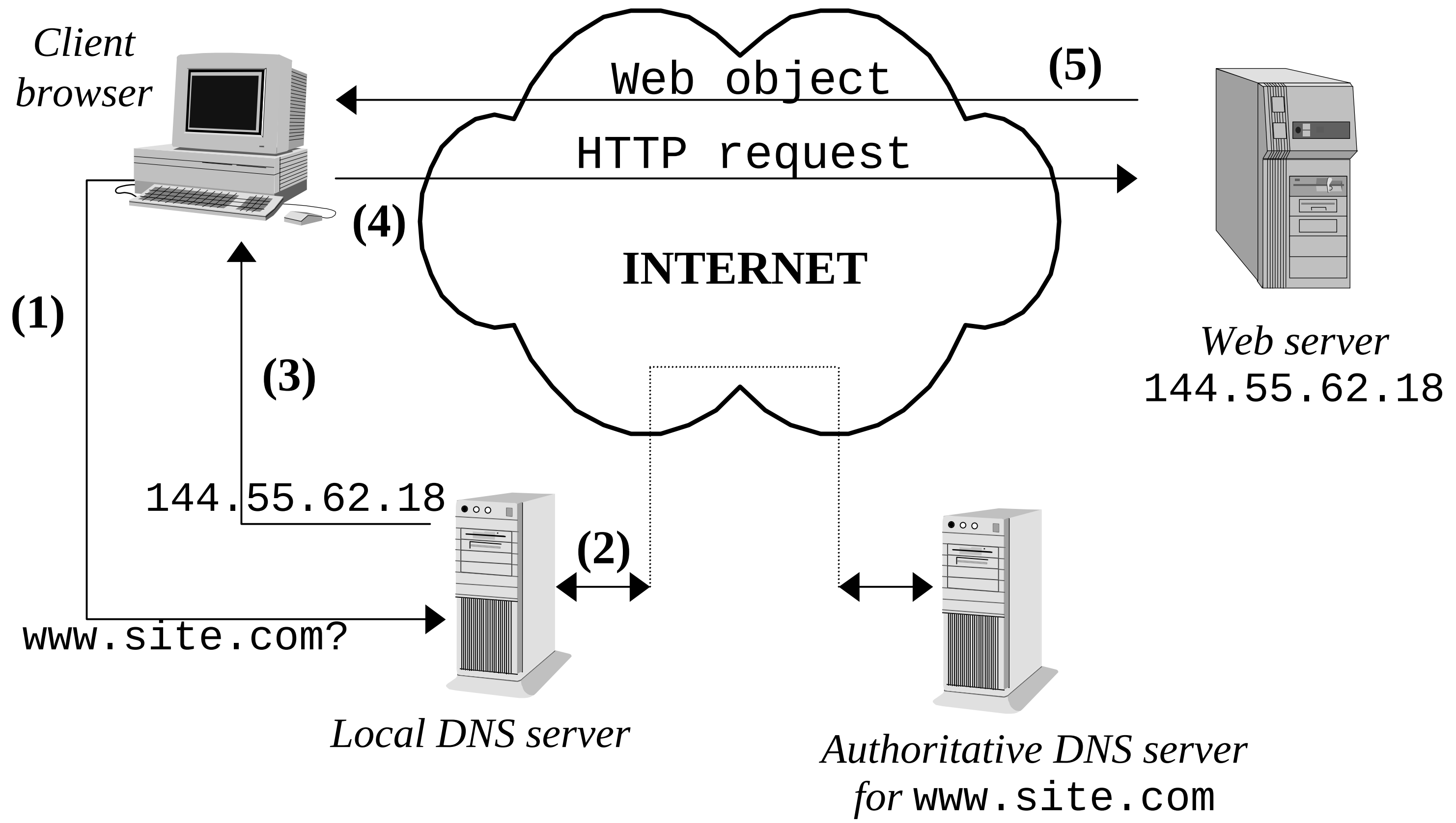
- **Architecture design**
  - Find the architecture guaranteeing the Service Level Agreement on all Web services

Tutorial focus

Definitions in this tutorial

- ***Session***: series of consecutive page requests to the Web site from the same user
- ***Page request***: a request that typically consists of multiple hits issued by the client
- ***Hit***: a request for a single object issued by the client to the Web server
- ***Types of objects***: class of file/service of a Web site
  - **static**
  - volatile
  - **dynamic**
  - **secure**

Analysis of a “simple” Web request



Lookup phase: (1) - (2) - (3)
Request phase: (4) - (5)

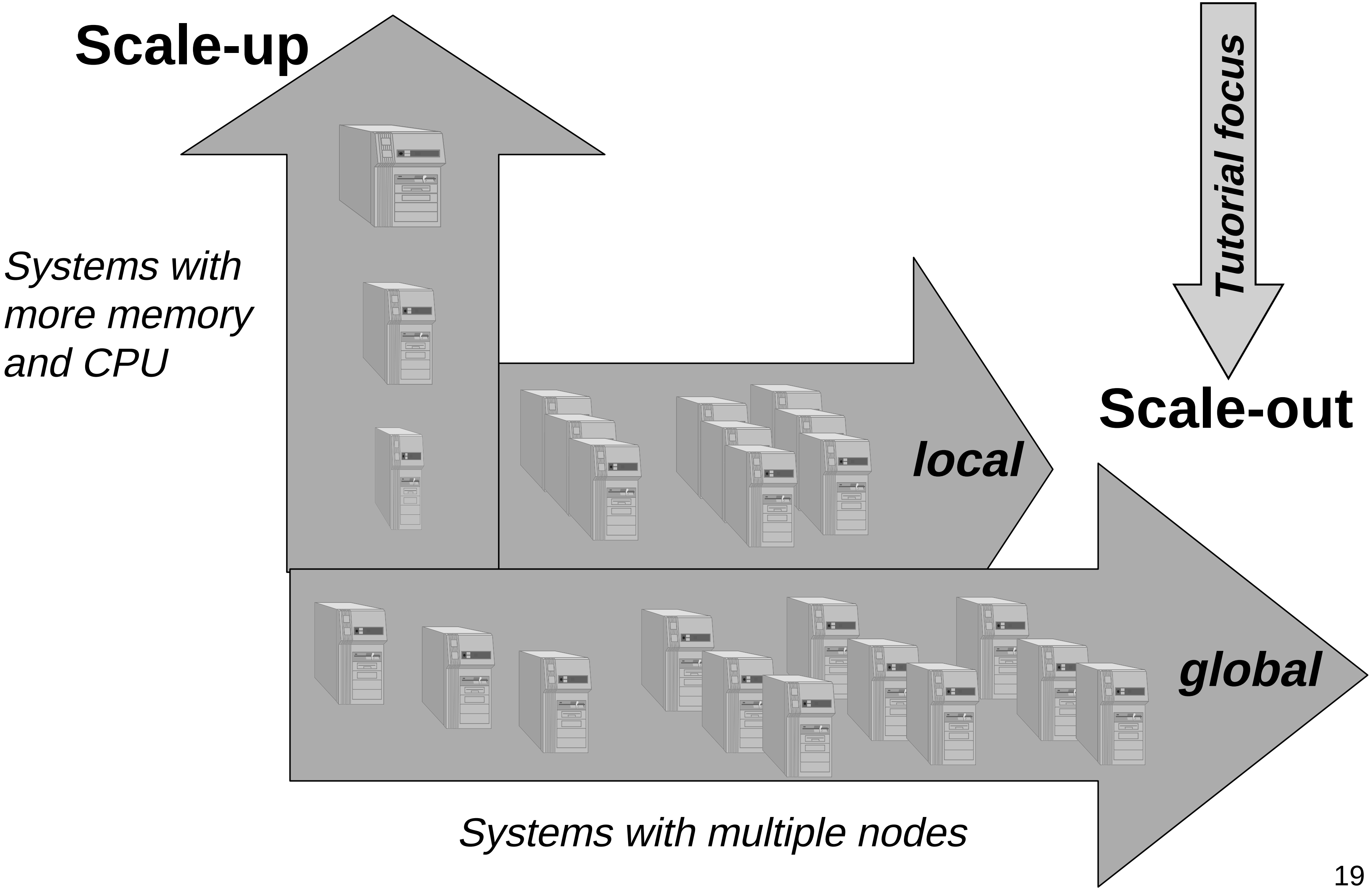
Potential sources of problems

- DNS may cache an invalid IP address
- Time-out of DNS address request (especially if *root servers* are overloaded)
- Web server may be overloaded or unreachable
- Internet links/routers may be overloaded
- Proxy server may fail or provide invalid objects

Possible Web Improvements

- NETWORK solutions
- SYSTEM solutions ← ***Tutorial focus***
- INFRASTRUCTURE solutions
  - Domain Name System
  - Caching
  - Server+Caching

System solutions have three ways



Multiple nodes Web systems
Desirable properties

- Fast access
- Architecture transparency
- Scalability

Tutorial focus

- Robustness
- Availability
- Reliability

- Accessibility (ability to deal with heterogeneous client devices and content adaptation)

Web drivers: *requirements*

- **Web publishing**

  + performance

  Tutorial focus

- **Electronic commerce**

  + security

  Tutorial focus

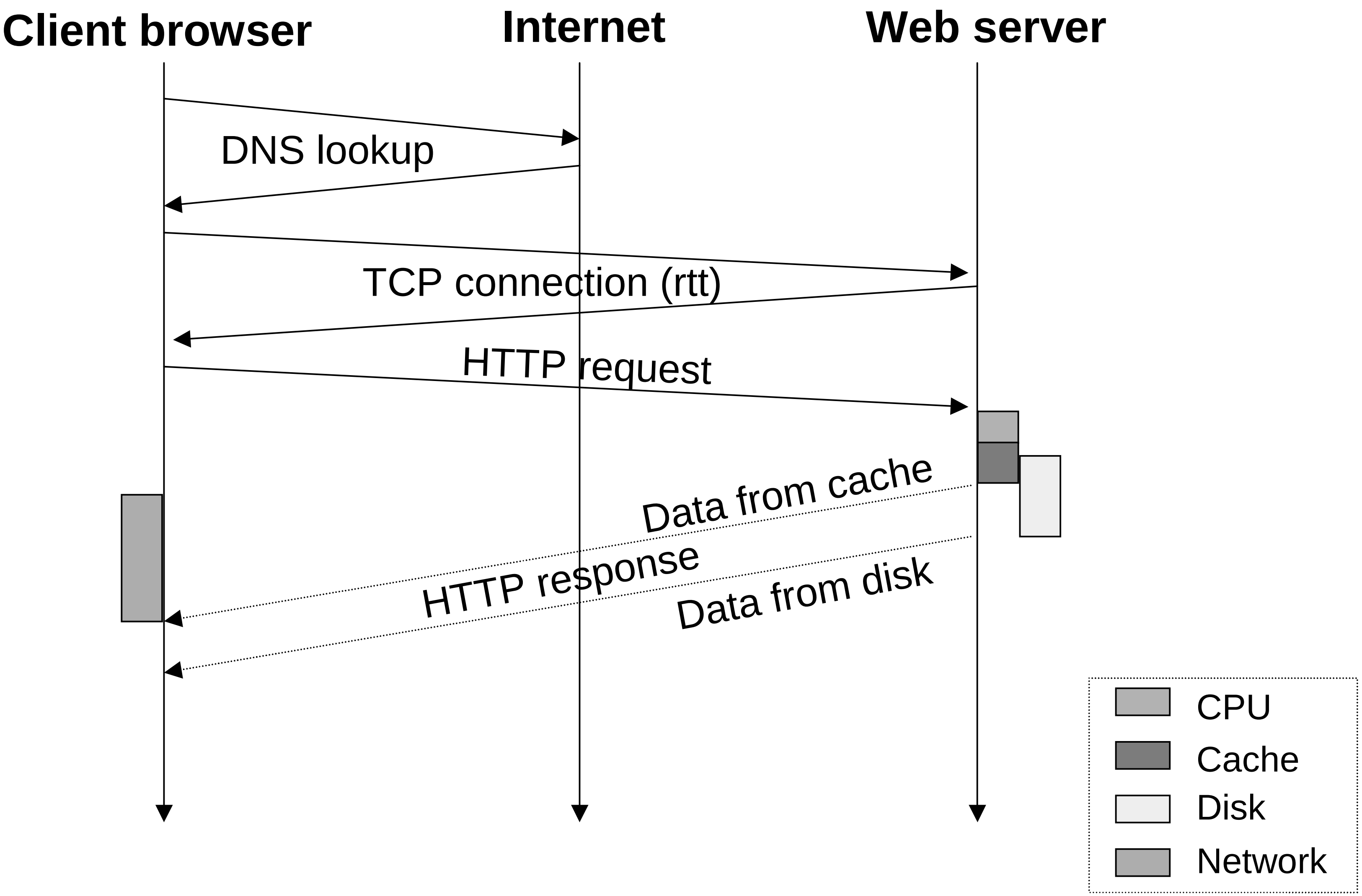
- **Education and training**

  + streaming audio and video

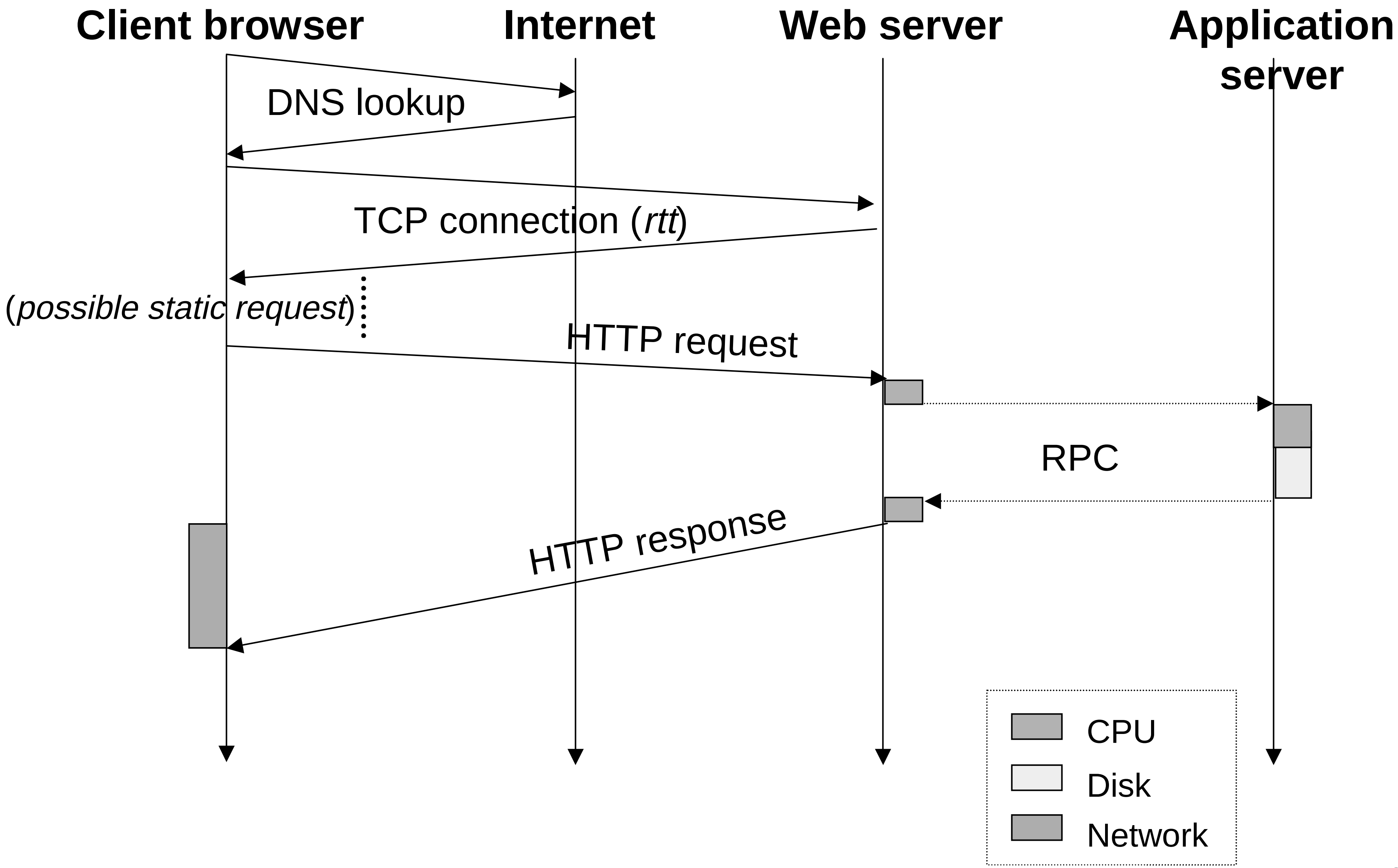
- **Ubiquitous Web**

  + accessibility

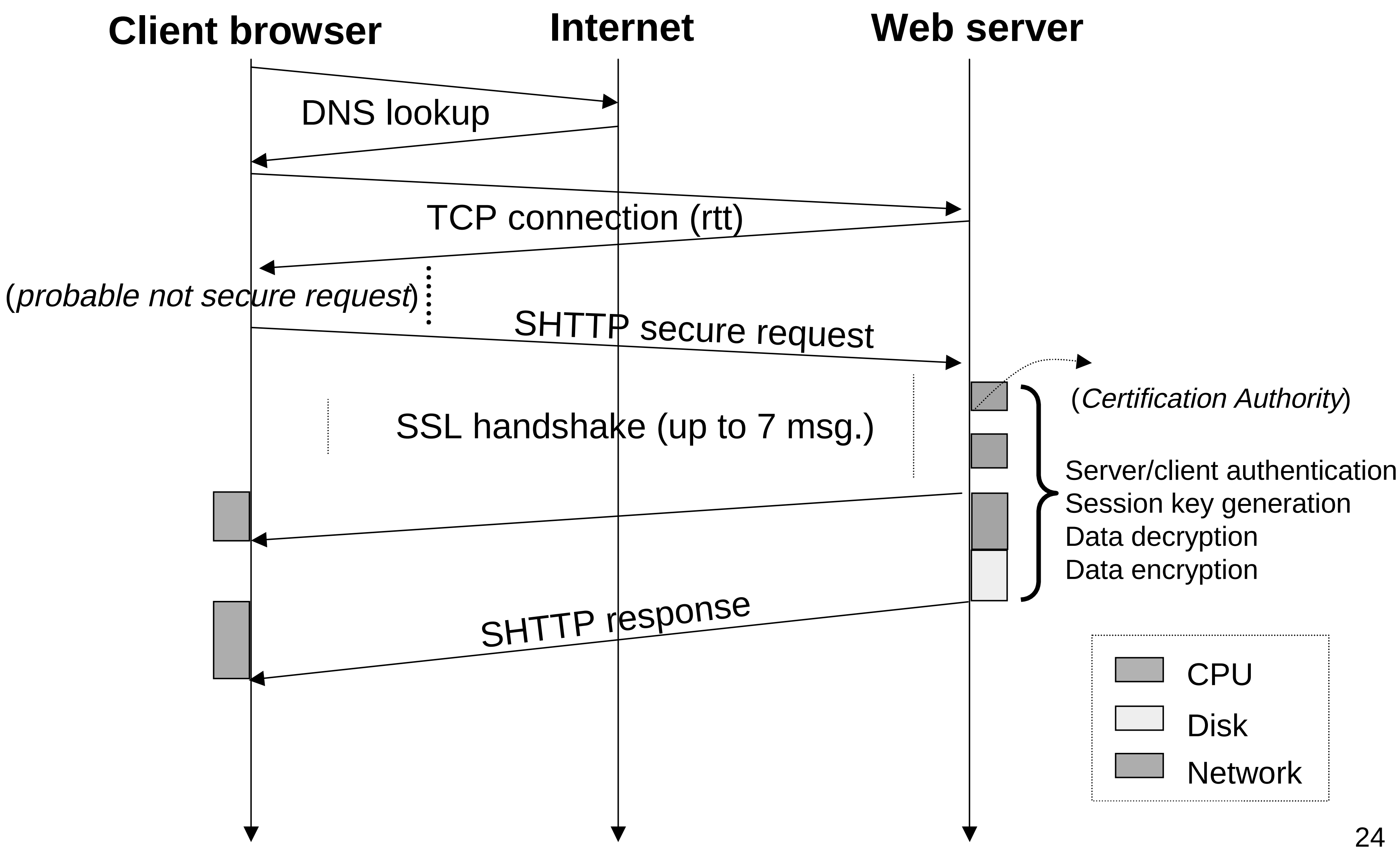
HTTP *static* request



HTTP *dynamic* request



HTTP *secure* request



Web performance is different

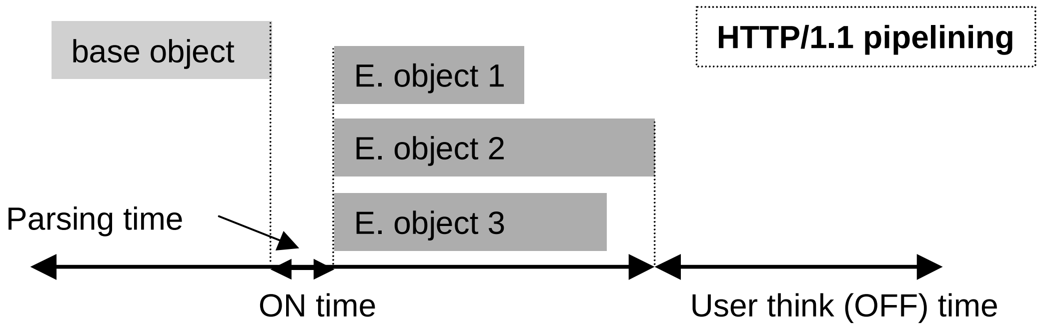
- Enormous variations
  - geographical location
  - day of the week
  - hour of the day (understanding peak periods)
- Workload is heavy-tailed distributed
  - Very large values are possible with non-negligible probability
- Dynamic nature of Web transactions
- Unpredictable nature of information retrieval and service request
  - It is difficult to size server capacity to support demand created by load spikes
- Traffic is bursty in several time scales
  - The maximum throughput decreases as the burstiness factors increase

Workload characterization

- Main components
  - Client, server, network, protocol
    - Characterization at different levels
- Focus on
  - *arrivals*
    - session, client/user times, protocol characteristics
  - *object characteristics*
    - size, popularity, type
  - *service characteristics*
    - static, volatile, dynamic, and secure

Workload: arrivals

- Session
  - Session length: heavy tailed distribution [Hub98]
  - Session arrival: Poisson process [Wil98, Liu00]
  - User request patterns [Pir99a, Pit99b]
- User/client times
  - User think time: heavy tailed distribution [Cro97a, Bar98, Arl00, Mor00]
  - Client parsing time [Bar98, Bar99b]
- HTTP protocol characteristics
  - HTTP/1.0 *vs.* HTTP/1.1 [Hei97, Bar98, Bar99b, Kri99]

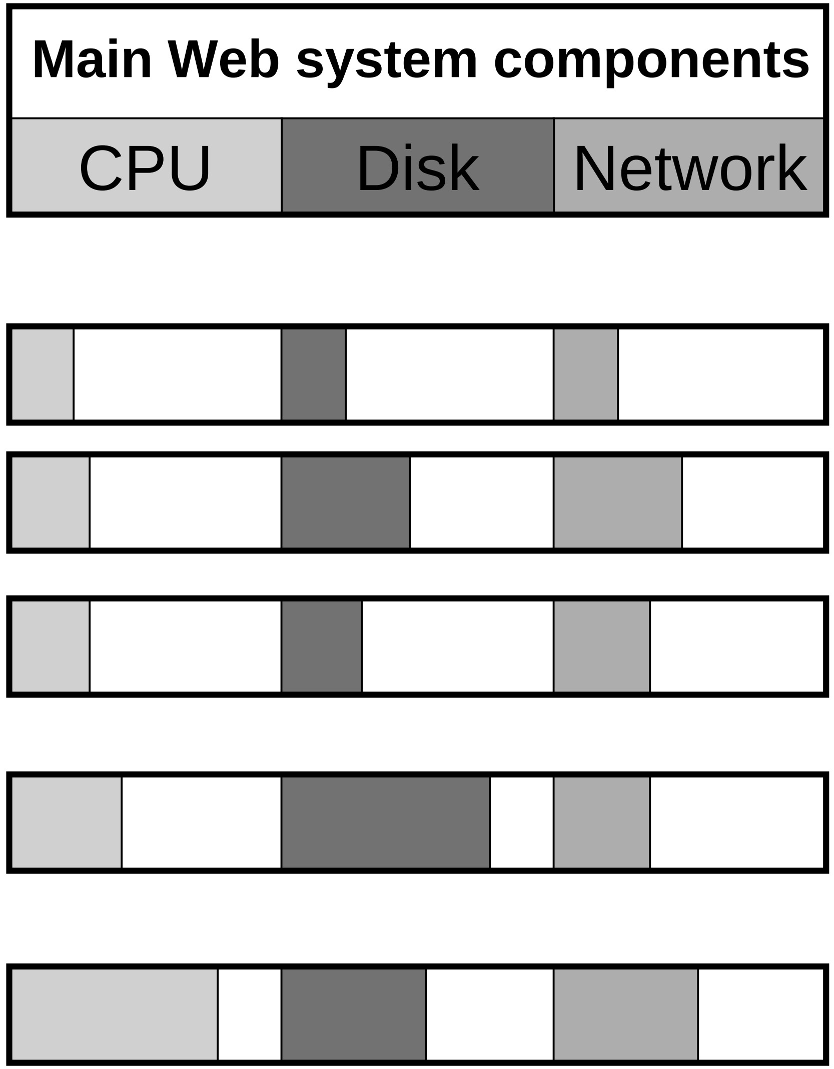


Workload: object characteristics

- Size
  - Unique objects, transferred objects [Cro97a, Arl00]
  - Heavy tailed distribution
    - Most transfers are small
- Popularity
  - Reference frequency follows a Zipf-like behavior [Cro97a, Arl00, Jin00]
- Type
  - Page composition [Arl00, Bar99a]
  - Analysis at different granularity level:
    - coarse grain level: no distinction among object type [Arl97, Bar98]
    - medium grain level: base, embedded, single object [Bar99b]
    - fine grain level: HTML, image, audio, video, application, dynamic, … objects [Arl00, Mah00]
  - Most transfers are still for HTML and image objects [Arl00]

Workload: *service characteristics*

- **Web publishing and Electronic commerce**
  - *static* objects
    - small (say, few *msec*)
    - large (disk bound)
  - *volatile* objects
  - *dynamic* objects (CPU and/or disk bound)
  - *secure* transactions (CPU bound)



Some workload references

- Significant amount of research on different Web-server environments [Arl97, Cro97a, Bar98, Arl00, Pit99a, Mah00]

- Some recent studies focused on characterization of heavily accessed and dynamic Web-server environments [Iye99, Arl00, Squ00]

Part 2

Taxonomies and classifications

Outline (Part 2)

- **A taxonomy of scalable Web-server systems**
  - Mirrored systems
  - Locally distributed systems
  - Globally distributed systems
- **A taxonomy of Web scheduling algorithms**
  - Static (*information-less*)
  - Dynamic

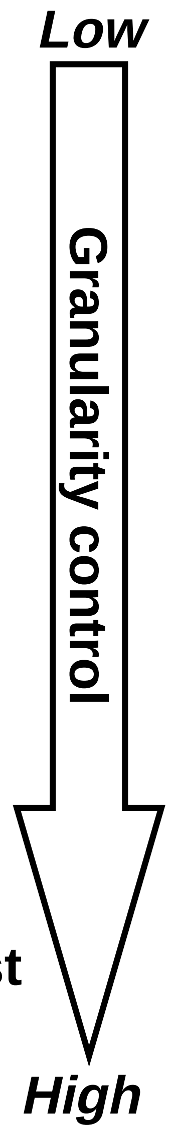
Requirements

Scalable Web-server systems are based on multiple server platforms

- A **scheduling mechanism** to direct the client request to the “best” Web-server
- A **scheduling algorithm** to define the “best” Web-server
- An **executor** to carry out the scheduling algorithm and the relative mechanism

Web scheduling mechanisms

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

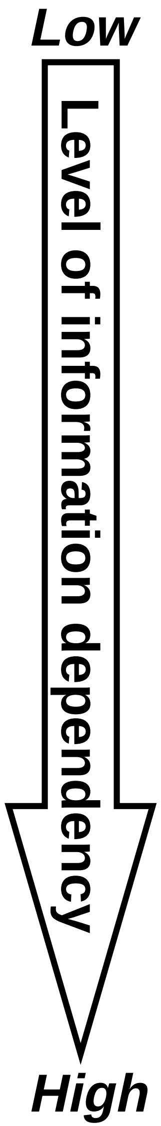


Web scheduling algorithms

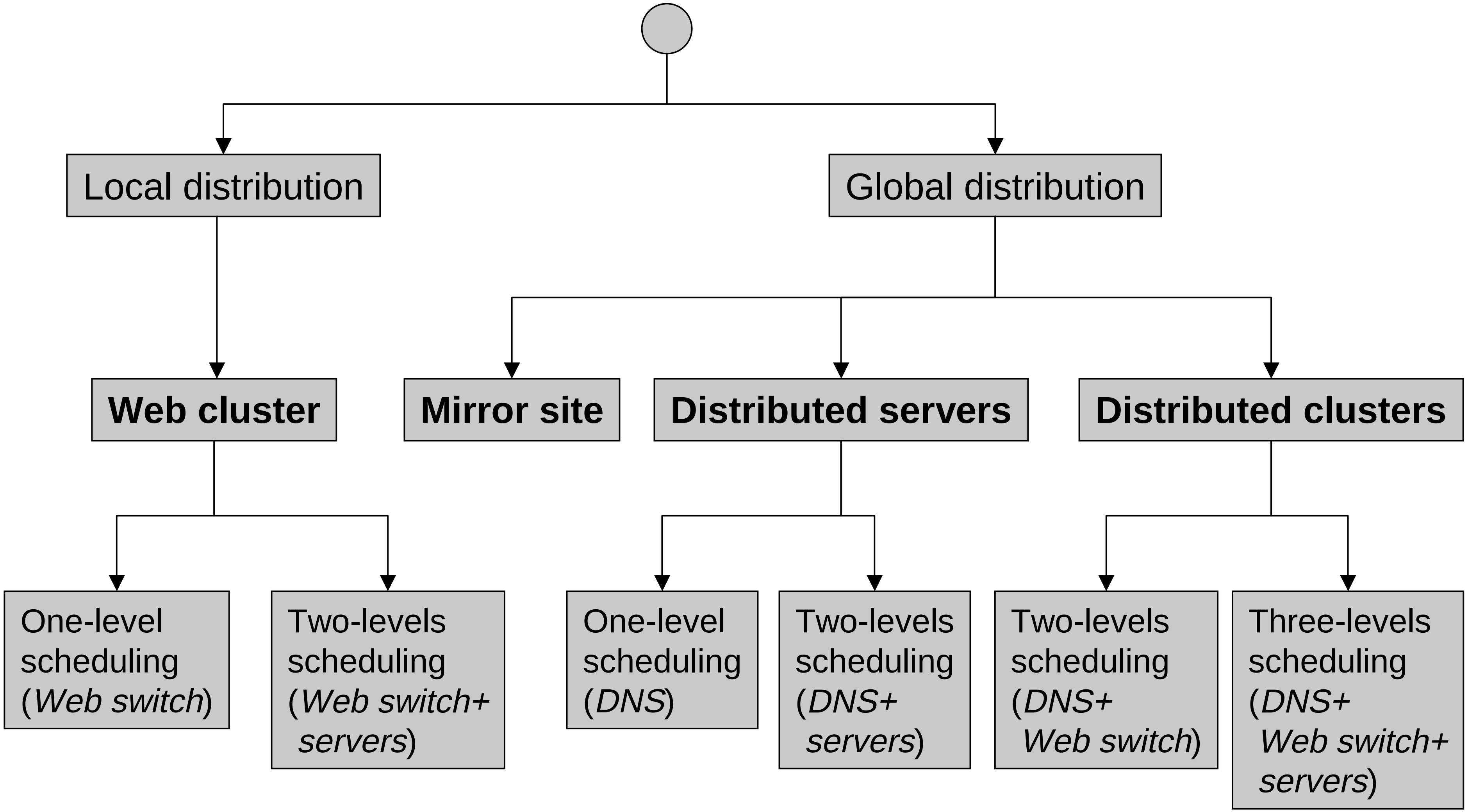
– **Static** (*information-less*)

– **Dynamic**
  - client info aware
  - server state aware
  - client info and server state aware

– **Adaptive** (not yet investigated)



A taxonomy of scalable Web-server systems



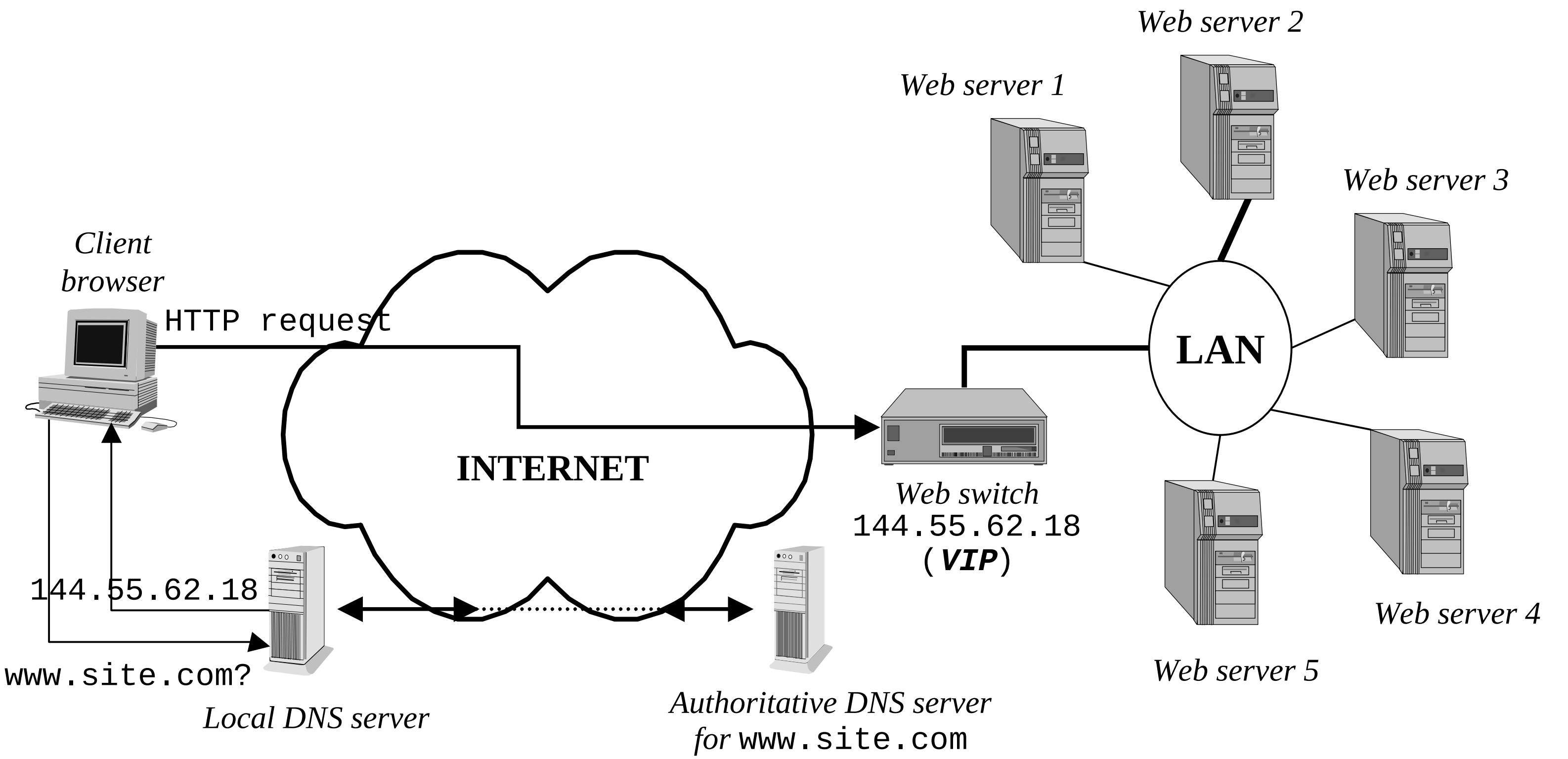
Part 3

Web clusters

Outline (Part 3)

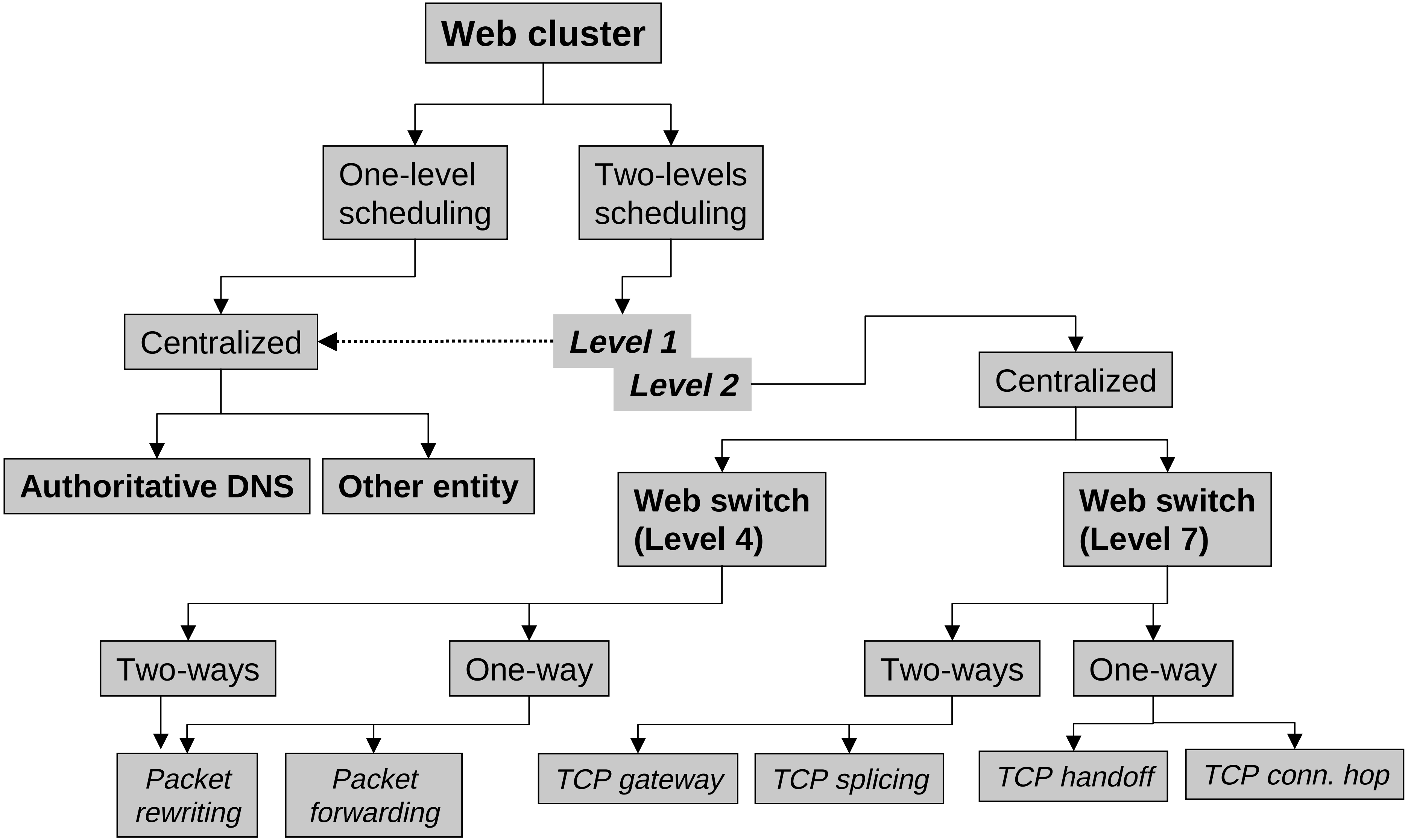
- **Locally distributed Web systems (*Web Clusters*)**
  - Systems based on level 4 Web switch
    - Architectures
    - Scheduling algorithms
  - Systems based on level 7 Web switch
    - Architectures
    - Scheduling algorithms
  - Performance metrics
  - Performance comparison of some scheduling algorithms
    - System model
    - Simulation results

Web cluster model



The response line does not appear because there are several alternatives.

Locally Distributed Web Systems



Web clusters: *Two-levels scheduling*

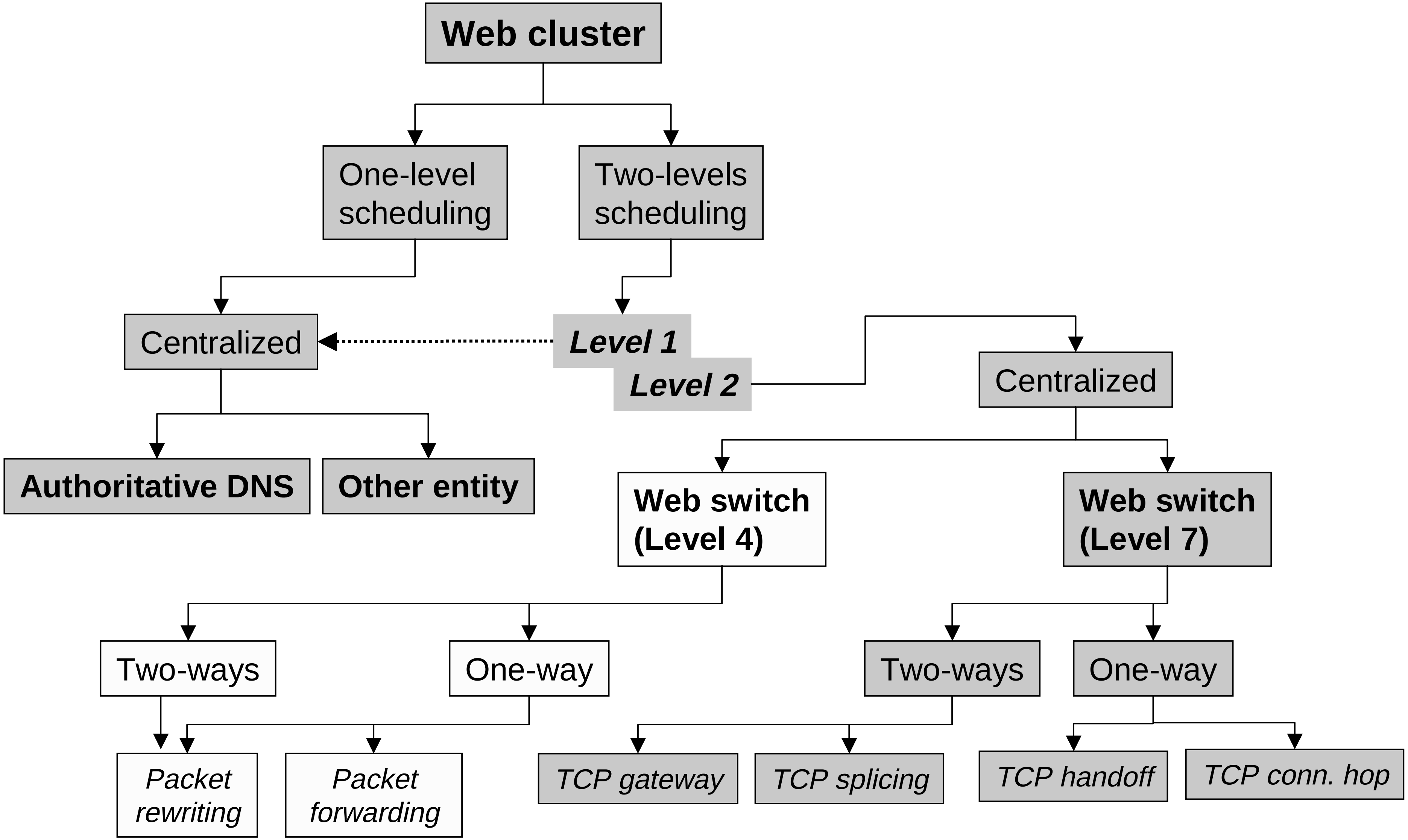
- Tightly coupled architecture at a single location
- Addressing of the Web cluster
  - One URL
  - One virtual IP address (**VIP**)
  - Private Web server addresses (at different protocol levels)
- **Web switch**: network component that acts as a dispatcher
  - Mapping from VIP to actual server address
  - Hit/Page request distribution through
    - special-purpose hardware device plugged into the network
    - software module running on a common OS
  - Fine grain control on request assignment (VIP inbound packets routed by the Web switch)

Web cluster alternatives

- Main features of Web clusters
  - Fine grain control on request assignment
  - High availability
  - Scalability limited by Internet access bandwidth

- Alternative architectures
  - **Level 4 Web switch** (*Content information blind*)
    - IP source and destination address, TCP port numbers, SYN/FIN bit in TCP header
  - **Level 7 Web switch** (*Content information aware*)
    - URL content, cookie, SSL id

Web cluster: Level 4



Level 4 Web switch

Level 4

- Level 4 Web switch works at **TCP/IP level**
- TCP session management (mapping on a per-session basis)
  - Packets pertaining to the same connection must be assigned to the same server machine
  - **Binding table** maintained by the Web switch to associate each active session 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

Web cluster architectures **Level 4**

Classification based on

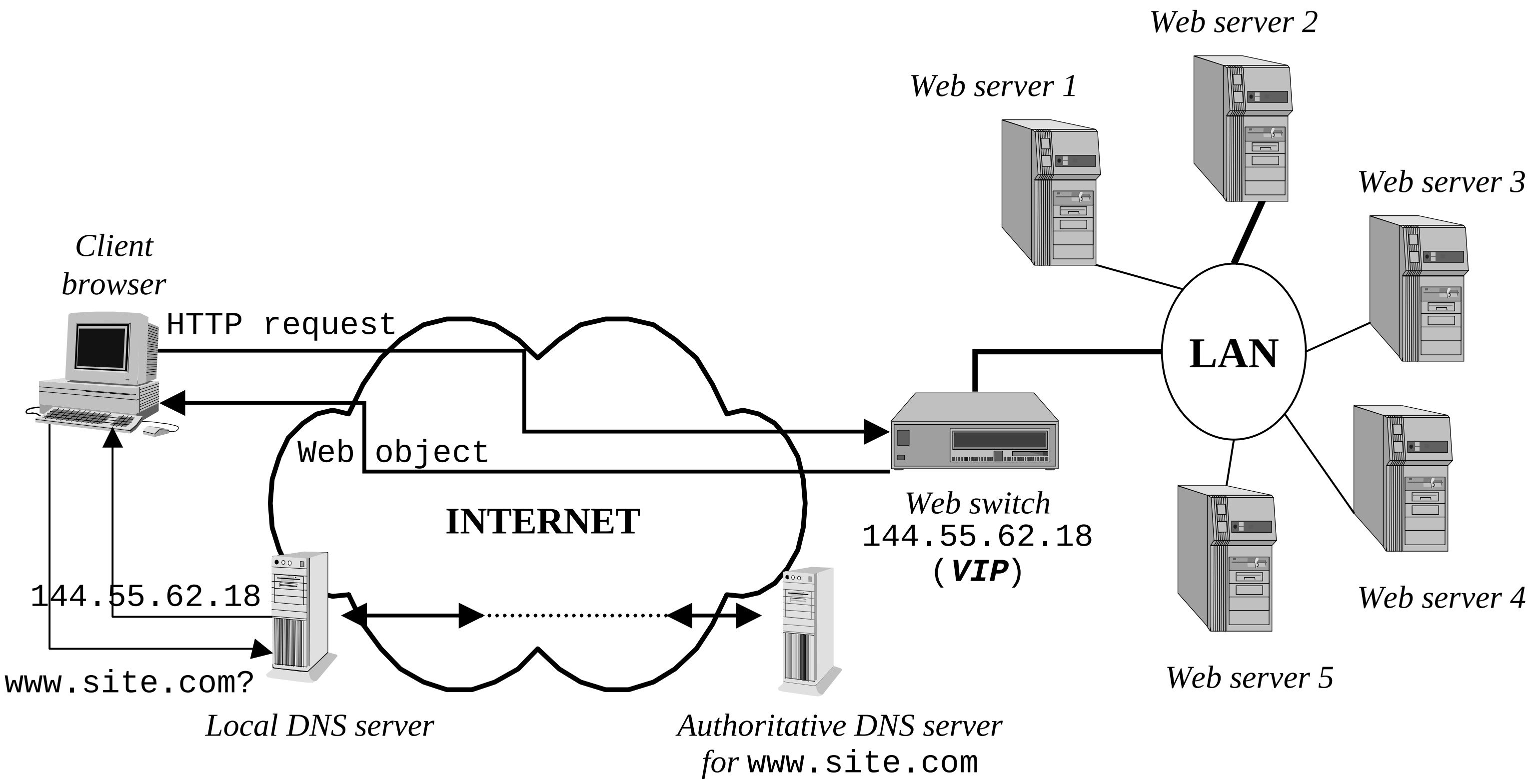
1) mechanism used by the Web switch to redirect inbound packets to the server

2) packet way between client and server (**the difference is the way back server-to-client**)

- **Two-ways architectures**
  - →inbound and outbound packets **rewritten** by the Web switch
- **One-way architectures**
  - →inbound packets **rewritten** by the Web switch
  - →inbound packet **forwarded** by the Web switch

Two-ways architecture

Level 4



Two-ways architectures

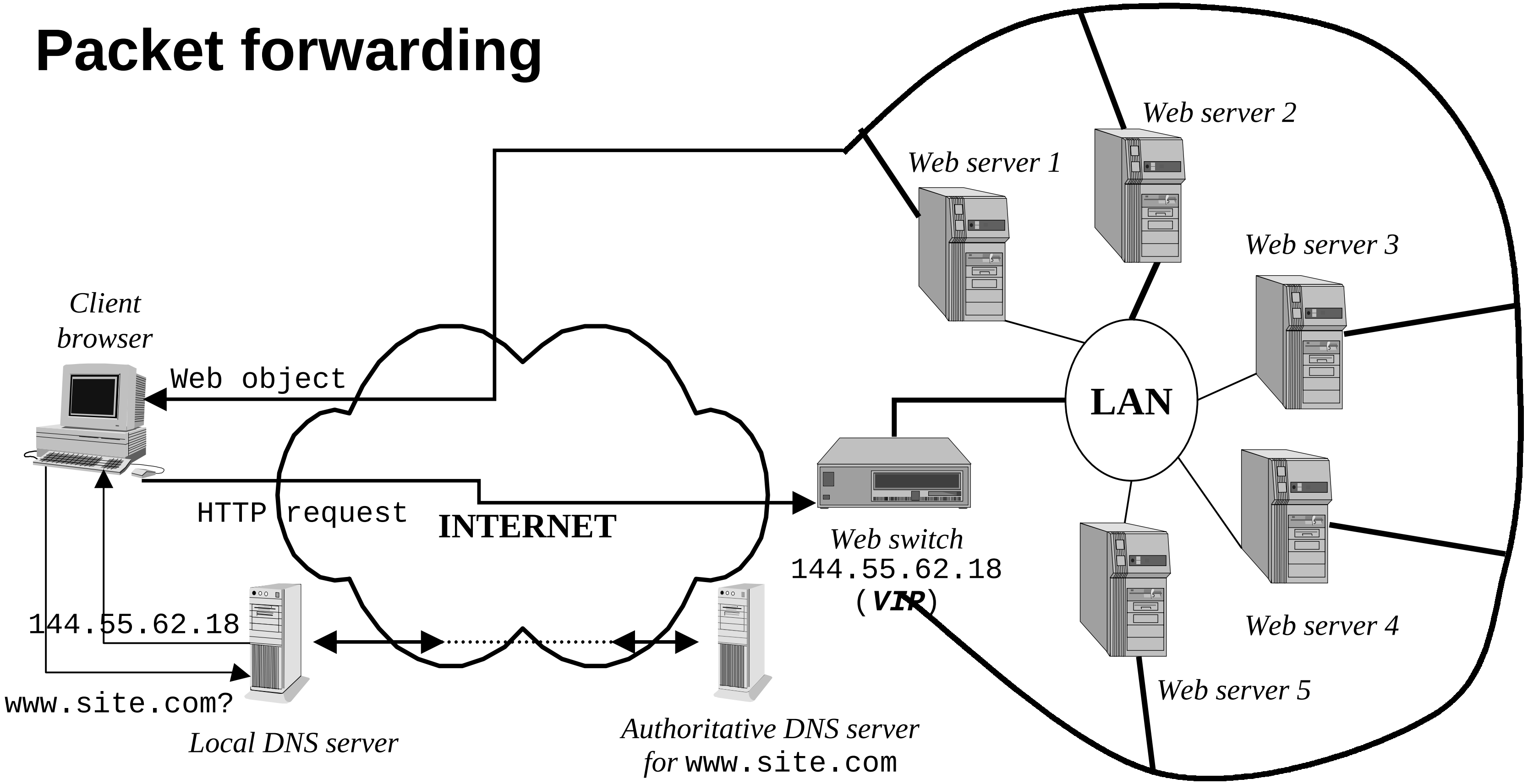
Level 4

- Packet rewriting is based on the IP **Network Address Translation** (**NAT**) approach [Ege94]
  - Each 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)
    - IP source address in outbound packet (IP server → VIP)
    - IP and TCP checksum recalculation

One-way architecture

Level 4

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



One-way packet rewriting

Level 4

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

One-way packet forwarding **Level 4**

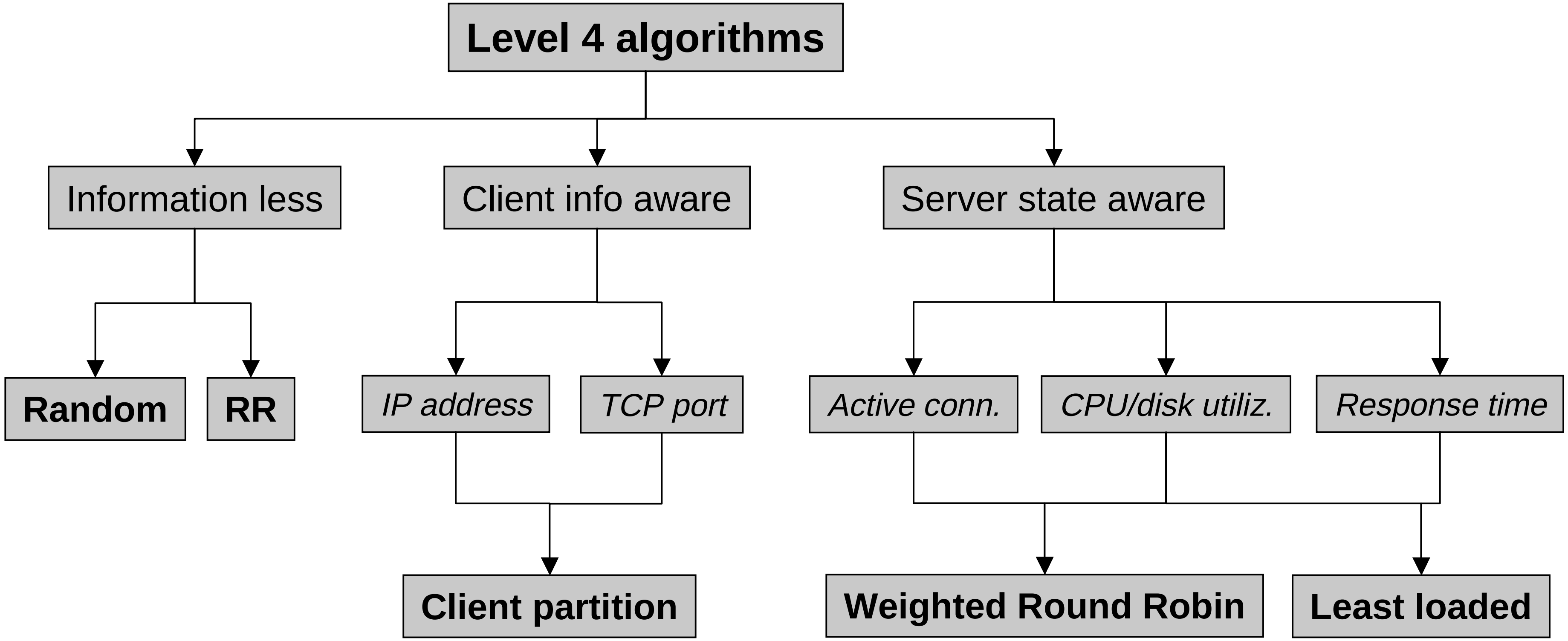
- VIP defined on the loopback interface of clustered servers (IP aliasing)
  - `ifconfig` Unix command
- No modification in inbound and outbound IP packets
  - Packet forwarding is done at **MAC level** (re-addressing of MAC frame containing the packet)
- Outbound packets do not need to pass back through the Web switch

PRO: A separate high-bandwidth connection can be used for outbound packets

CON: Web switch and servers must be on the same subnet

Web switch algorithms

Level 4



Static algorithms

Level 4

- **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

Level 4

- **Client partition**
  - Request assignment based on client information in inbound packets
    - Client IP address
    - Client port
  - Simple method to implement QoWS disciplines for individuals or group of clients

Server state aware algorithms **Level 4**

- Request assignment based on server load info
  - **Least loaded server** (**LLS**)
  - **Weighted Round-Robin** (**WRR**)
    - it allows configuration of weights as a function of server load [Hun98]

- 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

Web cluster proposals

Level 4

Two-ways

Packet rewriting

- **Cisco's** LocalDirector [CisLD]
- **Magicrouter** [And96]
- **Foundry Networks'** ServerIron [Fou]
- **Alteon WebSystems** [Alt]
- **LSNAT** [Sri98]
- **Linux Virtual Server** [Lin]
- **F5 Networks** BIG/ip [F5]
- **HydraWeb Techs** [Hyd]
- **Coyote Point Systems'** Equalizer [Coy]
- **Radware's** WSD [Rad]

One-way

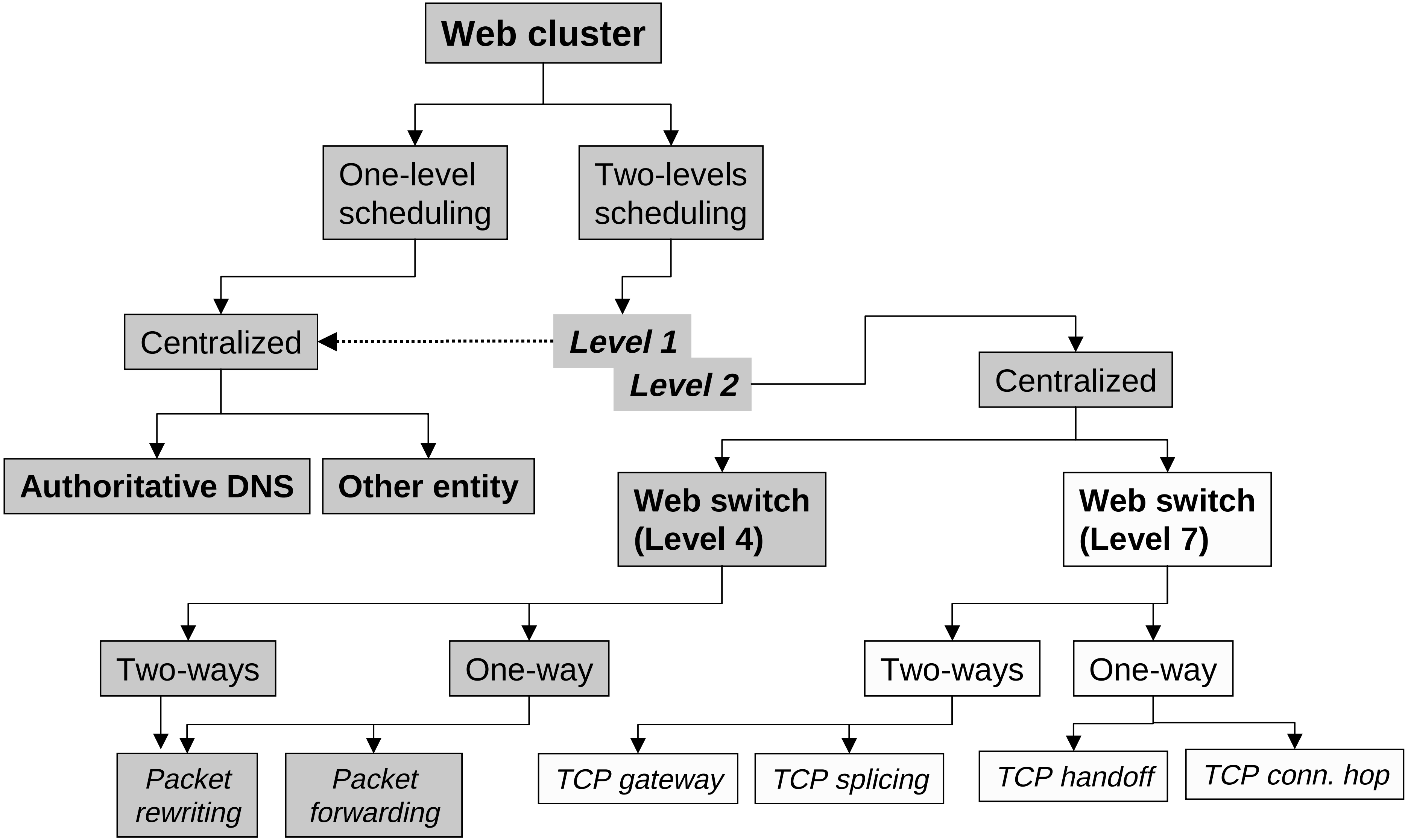
Packet rewriting

- **IBM** TCP router [Dia96]

Packet forwarding

- **IBM** Network Dispatcher [Hun98, IBMND]
- **ONE-IP** [Dam97]
- **LSMAC** [Gan00]
- **Foundry Networks'** ServerIron SwitchBack [Fou]

Web cluster: Level 7



Level 7 Web switch

Level 7

- Level 7 Web switch works at **application** level
- 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)
- Main features of content-based routing
  - allows content/type segregation on specialized servers
  - supports persistent connections
  - **allows HTTP/1.1 requests to be assigned to different Web servers** [Aro99]

Web cluster architectures **Level 7**

Classification based on

1) mechanism used by the Web switch to redirect inbound packets to the server

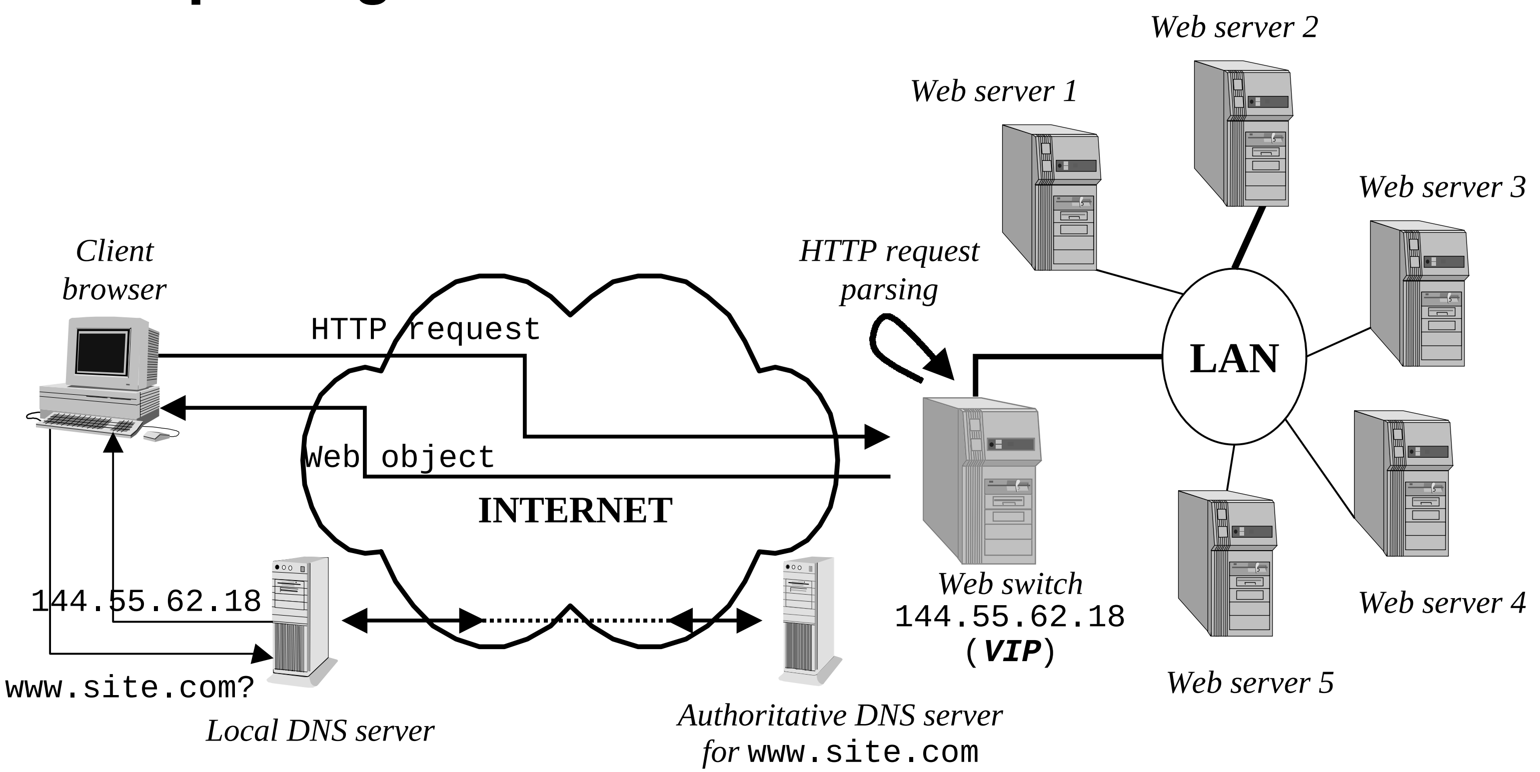
2) packet way between client and server (**the difference is the way back server-to-client**)

- **Two-ways architectures**
  - → TCP gateway
  - → TCP splicing
- **One-way architectures**
  - → TCP handoff
  - → TCP connection hop

Two-ways architecture

Level 7

- **TCP gateway**
- **TCP splicing**



Two-ways architectures **Level 7**

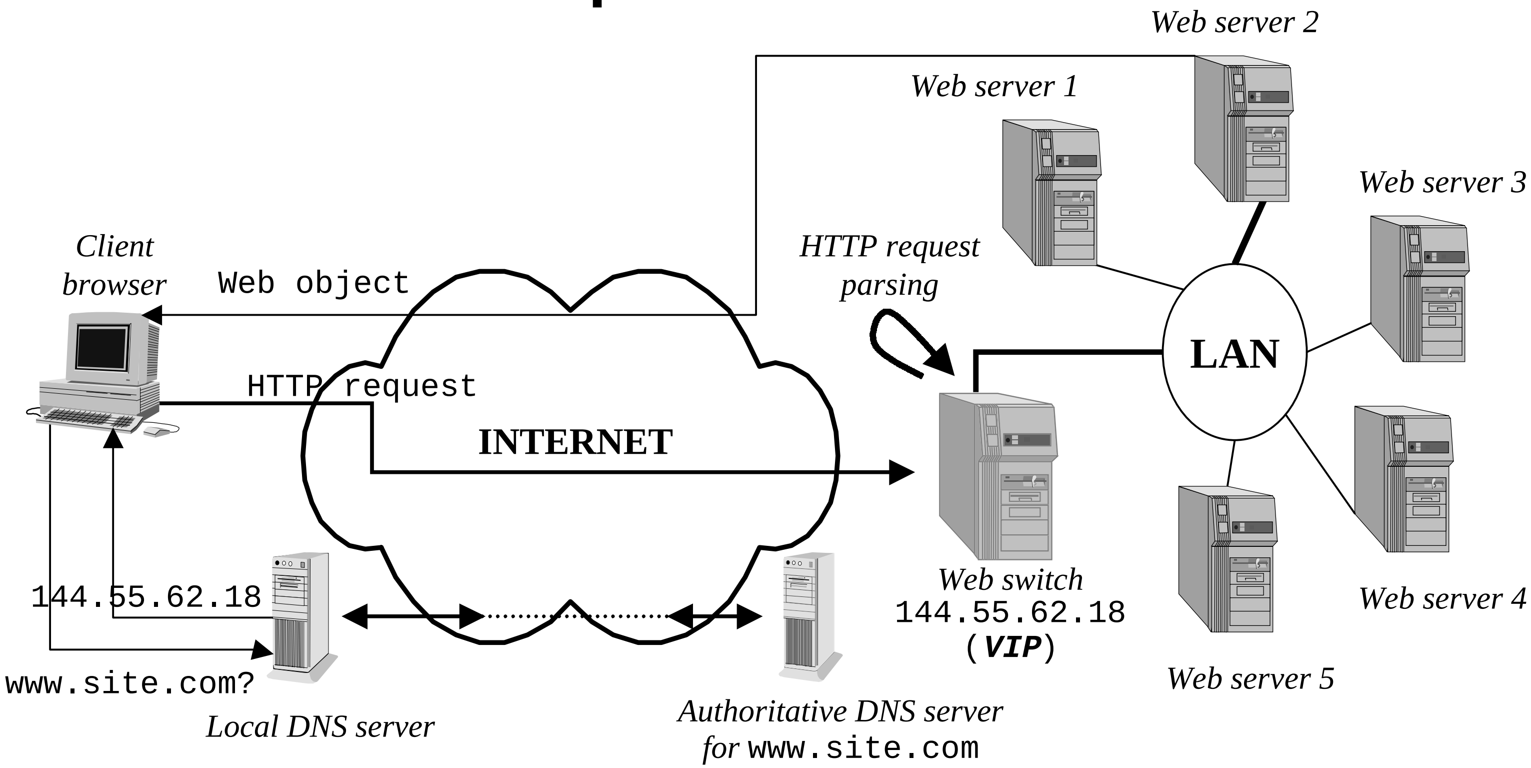
Outbound traffic must pass back through the switch

- **TCP gateway**
  - *Application level proxy* interposed between client and server to mediate their communications
    - Data forwarding at the switch at application level
  - It adds significant overhead
    - Two TCP connections per HTTP request
    - Way up and down through the protocol stack to application level
- **TCP splicing** [Coh99]
  - Optimization of TCP gateway
    - Data forwarding at the switch at network level
    - It requires modifications to the switch kernel

One-way architecture

Level 7

- **TCP handoff**
- **TCP connection hop**



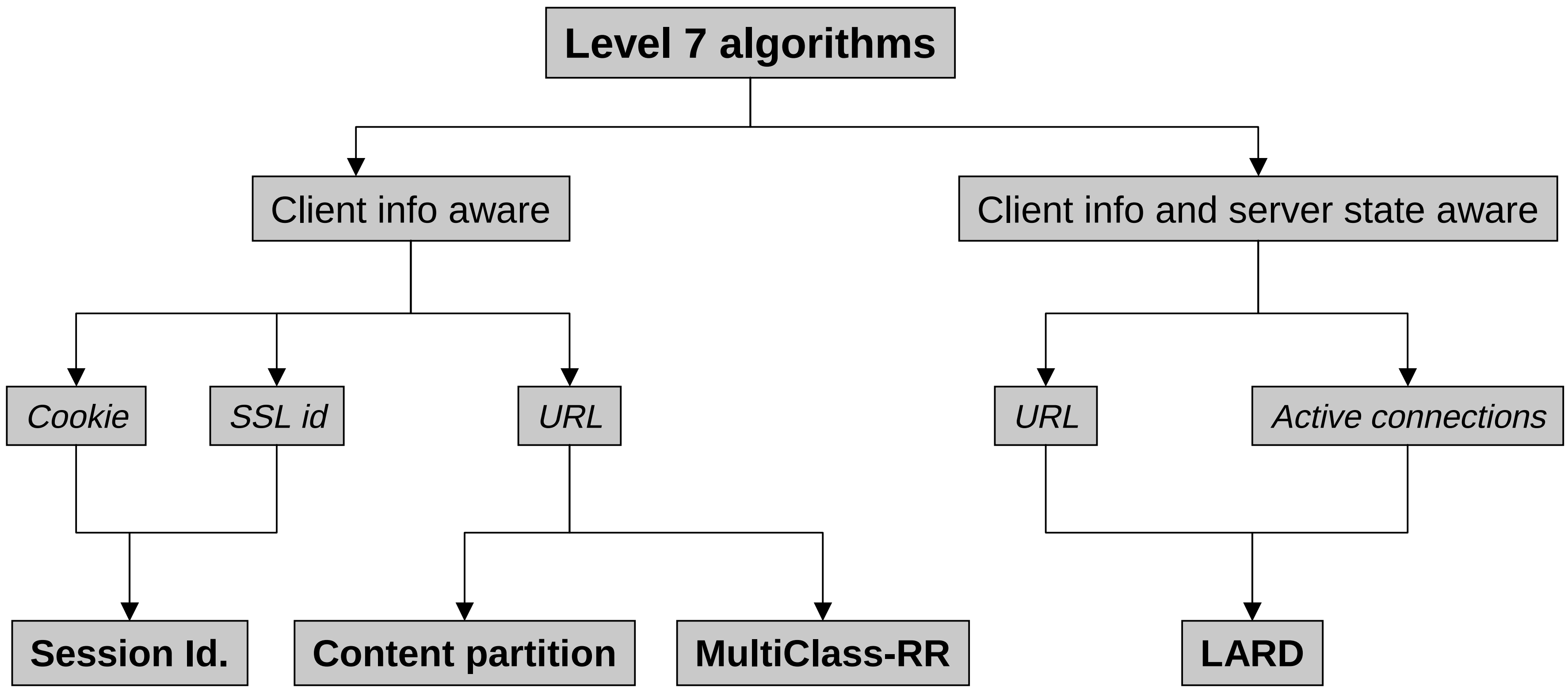
One-way architectures **Level 7**

Outbound traffic does not pass through the switch

- **TCP handoff** [Aro99, Pai98]
  - Handoff of the TCP connection established by the client with the switch to the Web server
  - It requires modifications to the switch and servers kernel
- **TCP connection hop** [ResCD]
  - Executed at the network layer between the *network interface card* (NIC) driver and the server's native TCP/IP stack

Web switch algorithms

Level 7



Client info aware algorithms **Level 7**

- **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.) [Har99]
    - Goal: augment load balancing
  - File space partitioned among the servers through a hash function
    - Goal: improve *cache hit rate* in Web servers

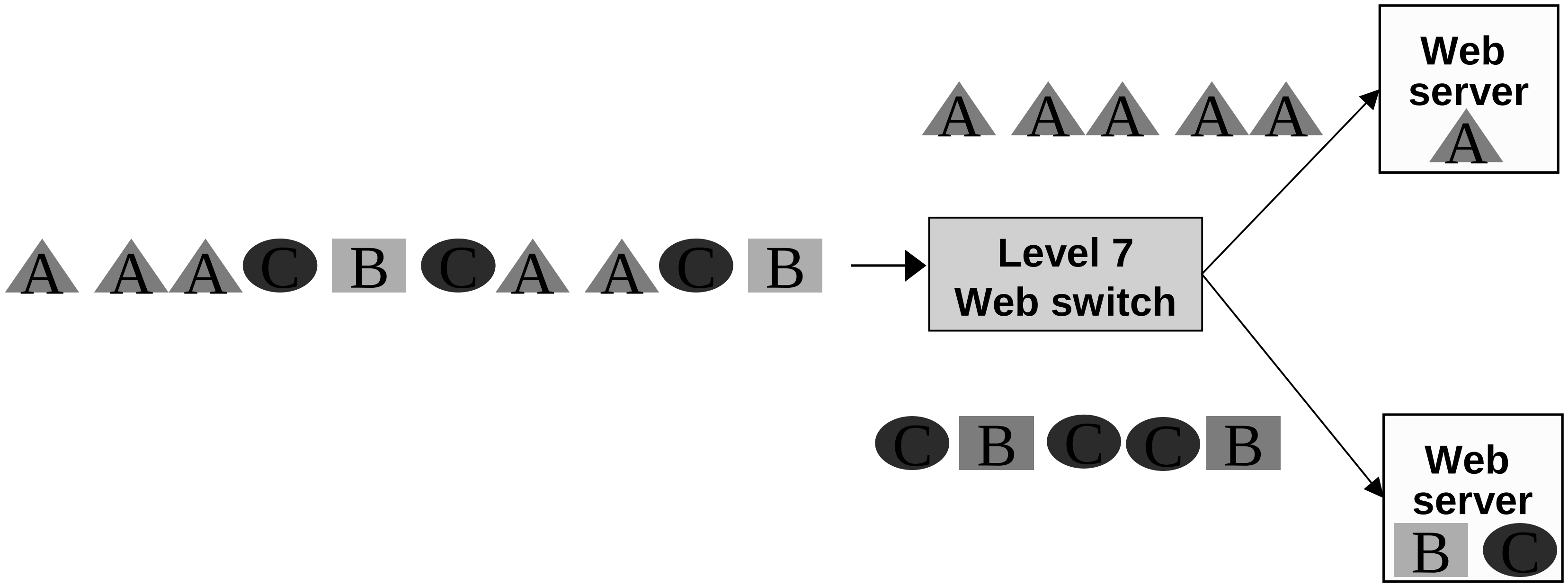
Client info aware algorithms **Level 7**

- **Multi-Class Round-Robin** (MC-RR) [Cas00]
  - 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
  - <u>Goal</u>: augment load sharing of *component bound* requests among Web servers

Client and server state aware algorithm **Level 7**

Locality-Aware Request Distribution (LARD) [Pai98]

- 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
- Goal: improve locality (cache hit rate) in server cache



Web cluster proposals

Level 7

Two-ways		**One-way**	
TCP gateway	*TCP splicing*	*TCP handoff*	*TCP conn. hop*
• **IBM** Network Dispatcher CBR [IBMND]	• [Coh99] • **Alteon Web Systems** [Alt] • **ArrowPoint** [Arr] • **Foundry Nets'** ServerIron [Fou]	• **LARD** [Pai98] • [Aro99]	• **Resonate's** Central Dispatcher [ResCD]

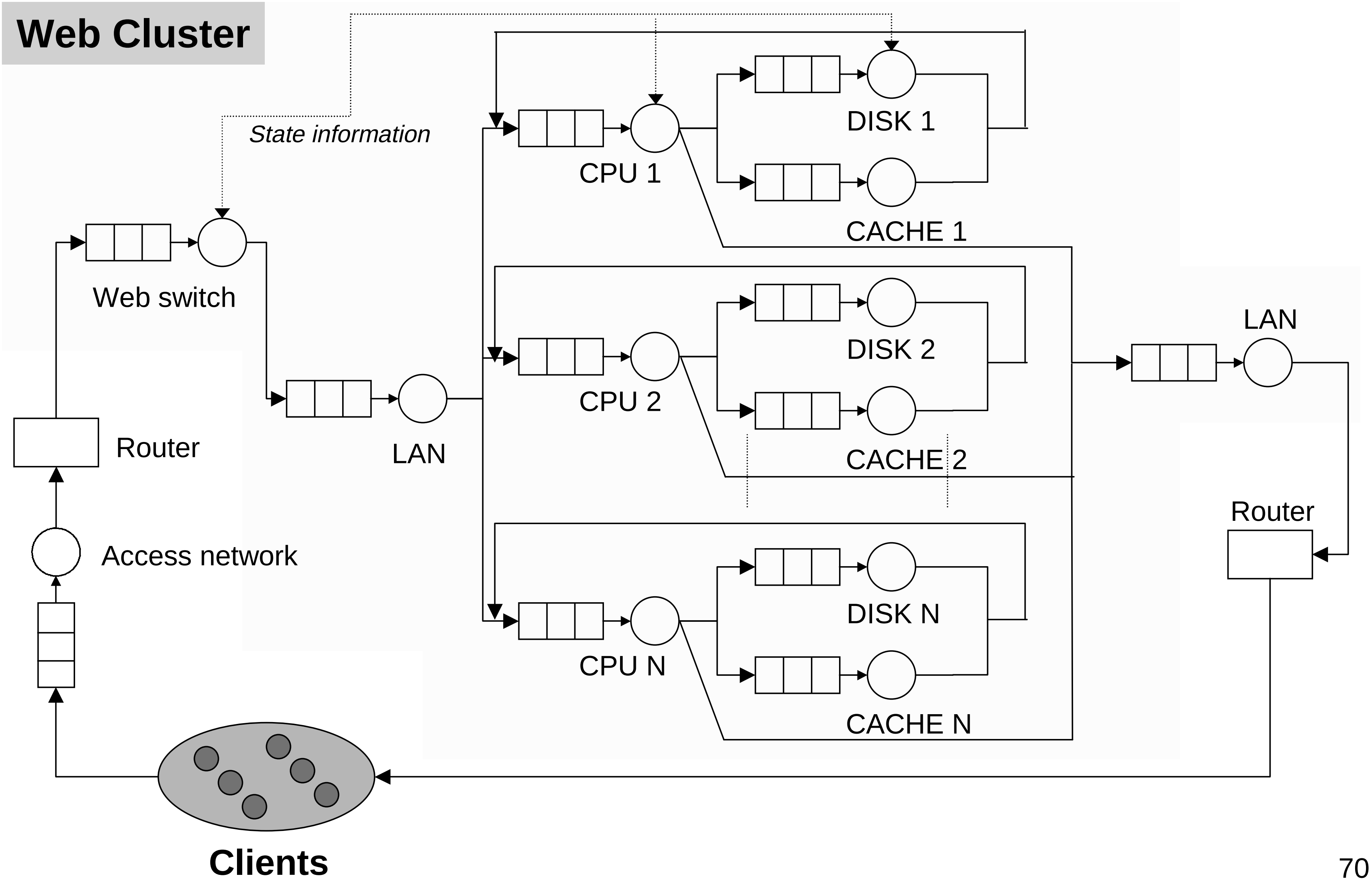
Web cluster architectures: summary

Web switch Level 4	Web switch Level 7
• Fast switching operations	• Slower switching operations
• Control on hit requests for HTTP/1.0	• Control on hit requests for HTTP/1.0
• Control on page requests for HTTP/1.1 (if request for embedded objects are in a single TCP segment)	• Control on hit/page requests for HTTP/1.1
• Client info: only at TCP/IP level	• Client info: TCP/IP information and HTTP header content

An example of performance comparison

- Web switch Level 7
- System model
- Scheduling algorithms
  - RR
  - WRR
  - MC-RR
- Metrics
  - Performance metrics
  - Load balancing metrics

System model



Performance metrics

- ***Response time***: - time to complete page or hit request
  (*latency*) - time to get the first response packet
  - client side (considering Internet delays)
  - Web system side

- ***Throughput:*** **-** quantities processed per unit time
  - number of hits completed per unit time (say, second)
  - number of files served per second
  - number of (*K*)bytes served per second

- ***Connections***: - number of connections per second
  (also number of refused connections)

- ***Utilization***
  - system
  - components (CPU, disk, memory, network)

Load balancing metrics

- **Load Balance Metric** (LBM)
  - weighted average of the instantaneous **peak-to-mean** ratios [Bun99]

$$peak-to-mean\ ratio = \frac{peak\_load_j}{\left(\sum_{i=1}^{N} load_{i,j}\right)/N}, \qquad peak\_load_j = \max_{j=1...N} load_j$$

$$LBM = \frac{\sum_{j=1}^{m}\left(\frac{peak\_load_j}{\sum_{j=1}^{m}\sum_{i=1}^{N} load_{i,j}/N} \times \frac{\sum_{i=1}^{N} load_{i,j}}{N}\right)}{\sum_{j=1}^{m}\sum_{i=1}^{N} load_{i,j}/N} = \frac{\sum_{j=1}^{m} peak\_load_j}{\left(\sum_{j=1}^{m}\sum_{i=1}^{N} load_{i,j}\right)/N}$$

$1 \leq LBM \leq N$ (number of Web servers)

Load balancing metrics (cont'd)

- **Unbalance Factor**
  - Percentage variation of the LBM value with respect to the optimal LBM value

$$UF = \frac{LBM - 1}{N - 1}$$

$$0 \le UF \le 1$$

  - Motivation: measure independent of the number of servers

Simulation experiments

- Simulation package: CSIM18
- Independent Replication Method
  - confidence level 95%
  - accuracy: within 5% of the mean

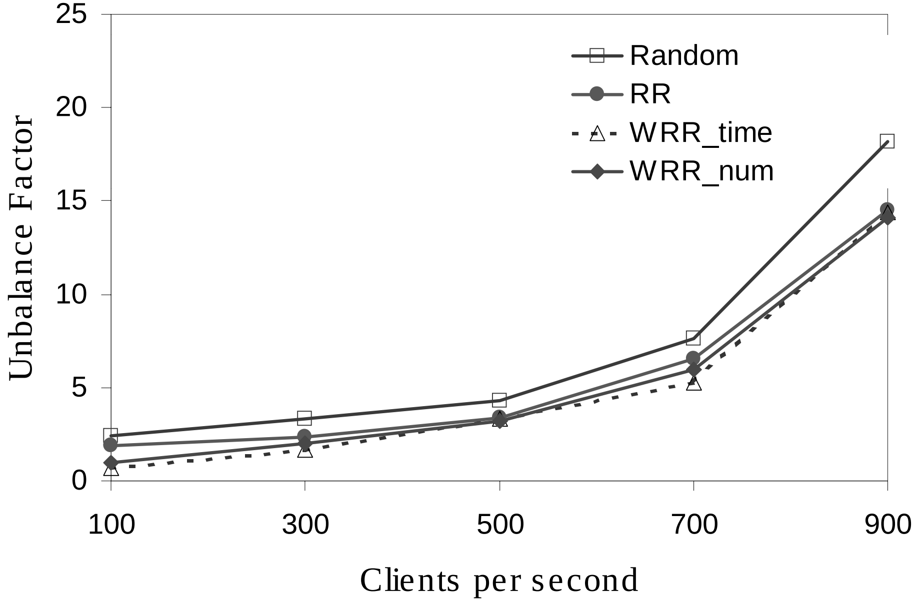
- Cases studied
  - Static *vs*. dynamic algorithms
  - Parameter setting (for dynamic algorithms)

  - Open model: arrivals in clients per second (**cps**)
  - Workload: Medium-light and heavy scenarios

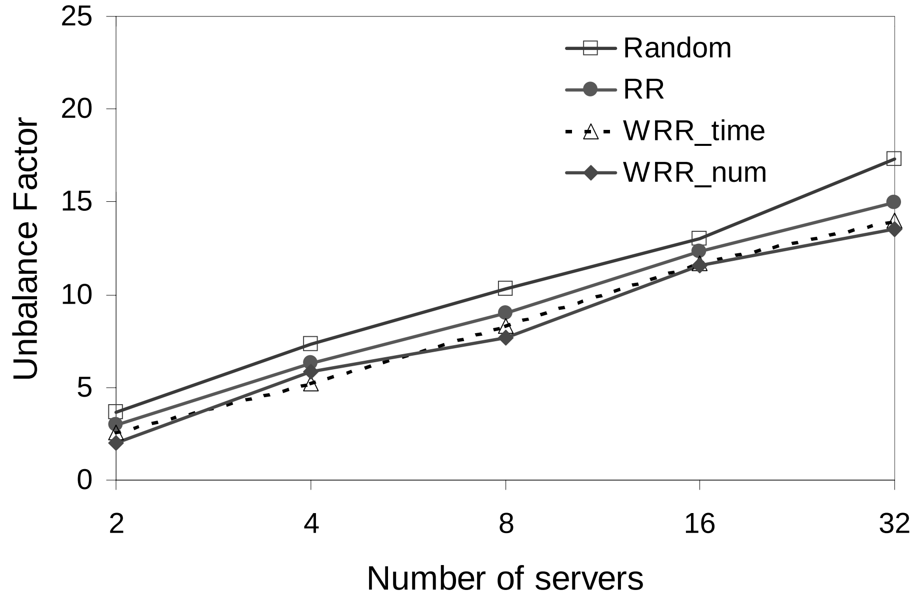
Choice of parameters

Category	Type	Parameter
Web cluster	Number of servers	2-32 (10)
	Disk transfer rate	20 MBps
	Intra-cluster bandwidth	100 Mbps
Client	Arrival rate	100-5600 (700) clients per second (**cps**)
	User think time	Pareto (α=1.4, k=2)
	Page requests per session	Inverse Gaussian (μ=3.86, λ=9.46)
	Objects per page	Pareto (α=1.1-1.5, k=1)
	Inter-arrival time of hits	Weibull (α=7.640, σ=1.705)
	Hit size request (*body*)	Lognormal (μ=7.640, σ=1.705)
	(*tail*)	Pareto (μ=7.640, σ=1.705)

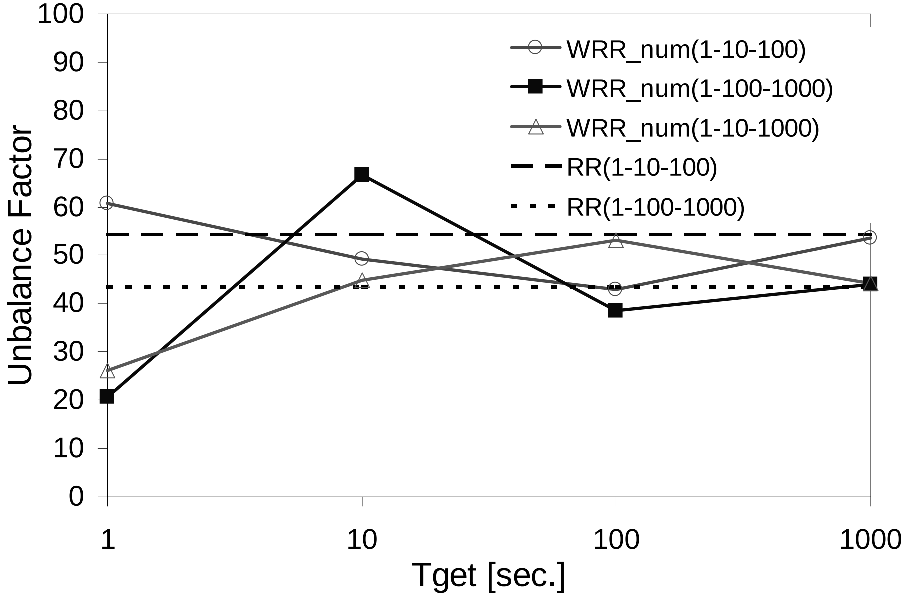
Results: *information less vs. WRR*



Scenario: 50% *static light* (**1**)
50% *static heavy* (**10**)



Results: *difficulty of parameter setting*



Scenario: 50% *static light* (**1**)

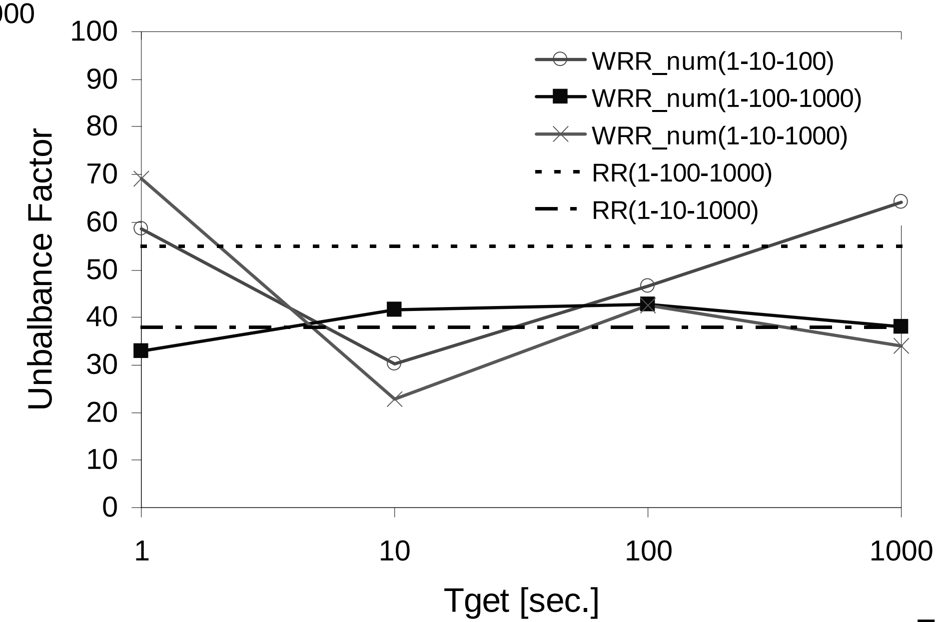
25% { *static heavy* (**10**), *dynamic* (**100**) }

25% { *static heavy* (**100**), *dynamic* (**1000**) }

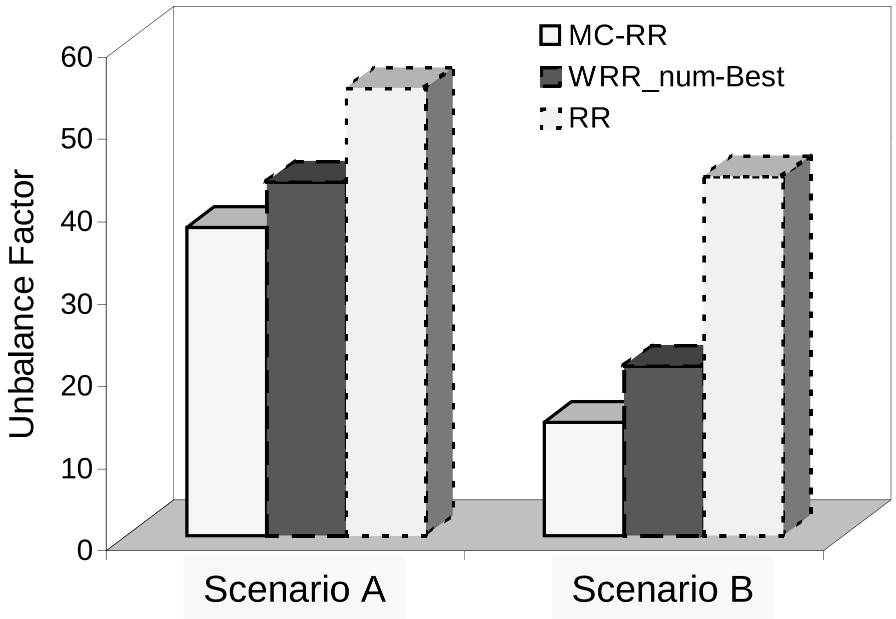
Scenario: 35% *static light* (**1**)

30% { *static heavy* (**10**), *dynamic* (**100**) }

30% { *static heavy* (**100**), *dynamic* (**1000**) }

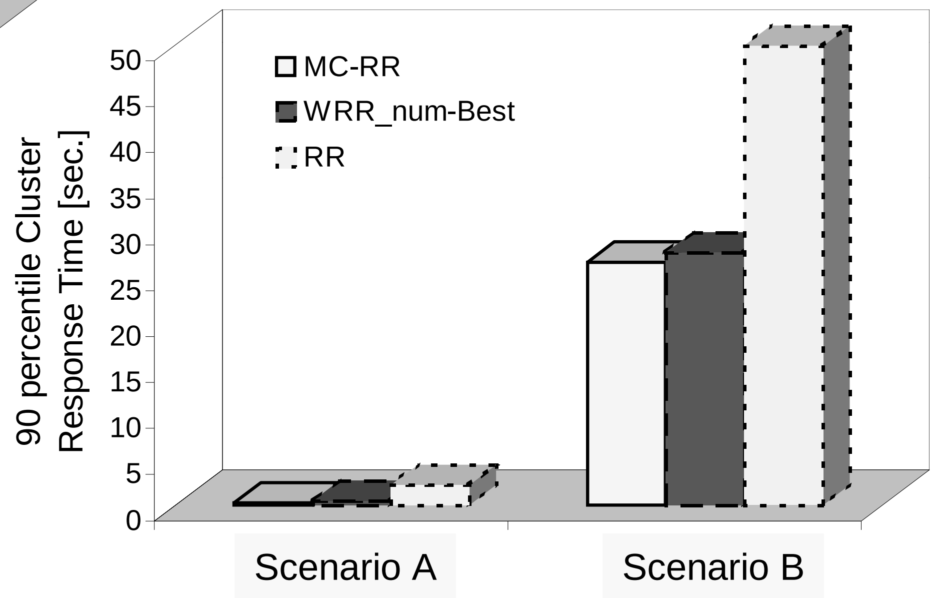


Results: *information less vs. dynamic*



Scenario A: 50% *static light*
25% *static heavy*
25% *dynamic*

Scenario B: 50% *static light*
25% *static heavy*
25% *dynamic heavy*



Note the low correlation between *Unbalance Factor* and *Response Time* for Scenario A and B

Web switch algorithms: summary

- Web switch controls 100% of traffic to the Web site
- To prevent bottlenecks, it does not requires (and it cannot use) too much complex scheduling algorithms
- Static algorithms achieve performance comparable to dynamic algorithms when all service times of Web transactions are in a range of two orders of magnitude
- Over the two order threshold, it is useful to use dynamic algorithms (*client info* or *server state aware*)
- As it is difficult to choose the best parameters for many *server state aware* disciplines, often *client info aware* algorithms are preferable
- Their drawback is the higher overhead of *Level 7 Web switch* operations

Part 4

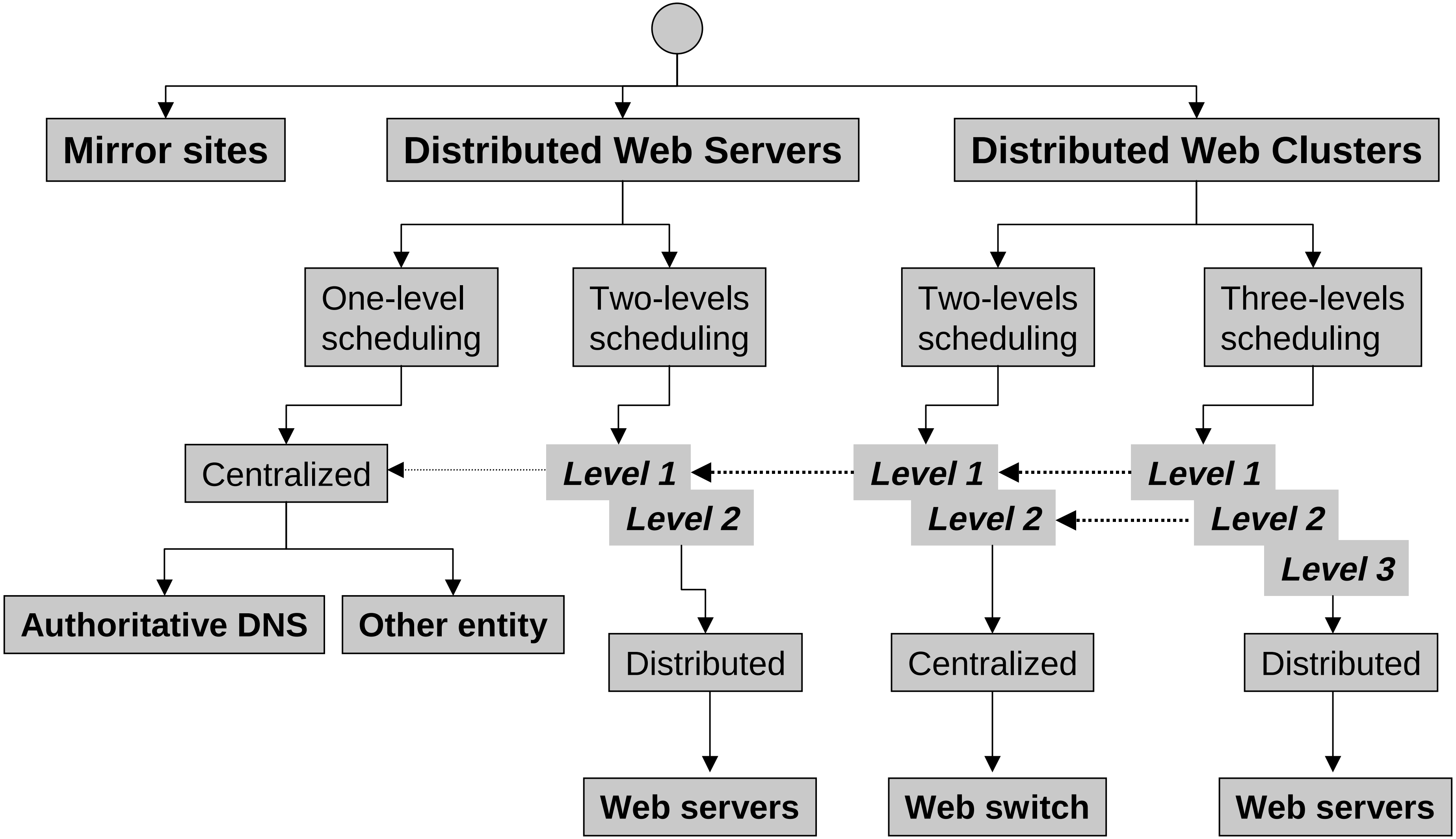
Distributed Web systems

Outline (Part 4)

- **Globally distributed Web systems**
  - Architectures
  - Scheduling algorithms (***DNS*** and ***Web server***)
  - Models
    - Network
    - System
  - Performance metrics
  - Results

- **Web infrastructures**
  - Global content distribution
  - Cooperative caching

Globally Distributed Web Systems



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**

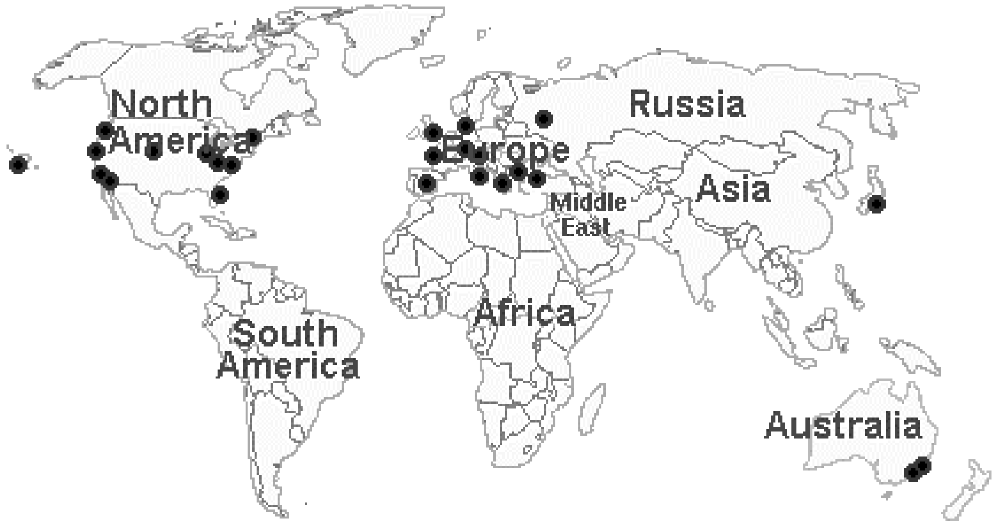
Scheduling left to users

An example of mirror site

Mars Polar Lander Mission

Location of JPL

Mirror Sites



Public Sector Mirror Sites

Location	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

Mirror site

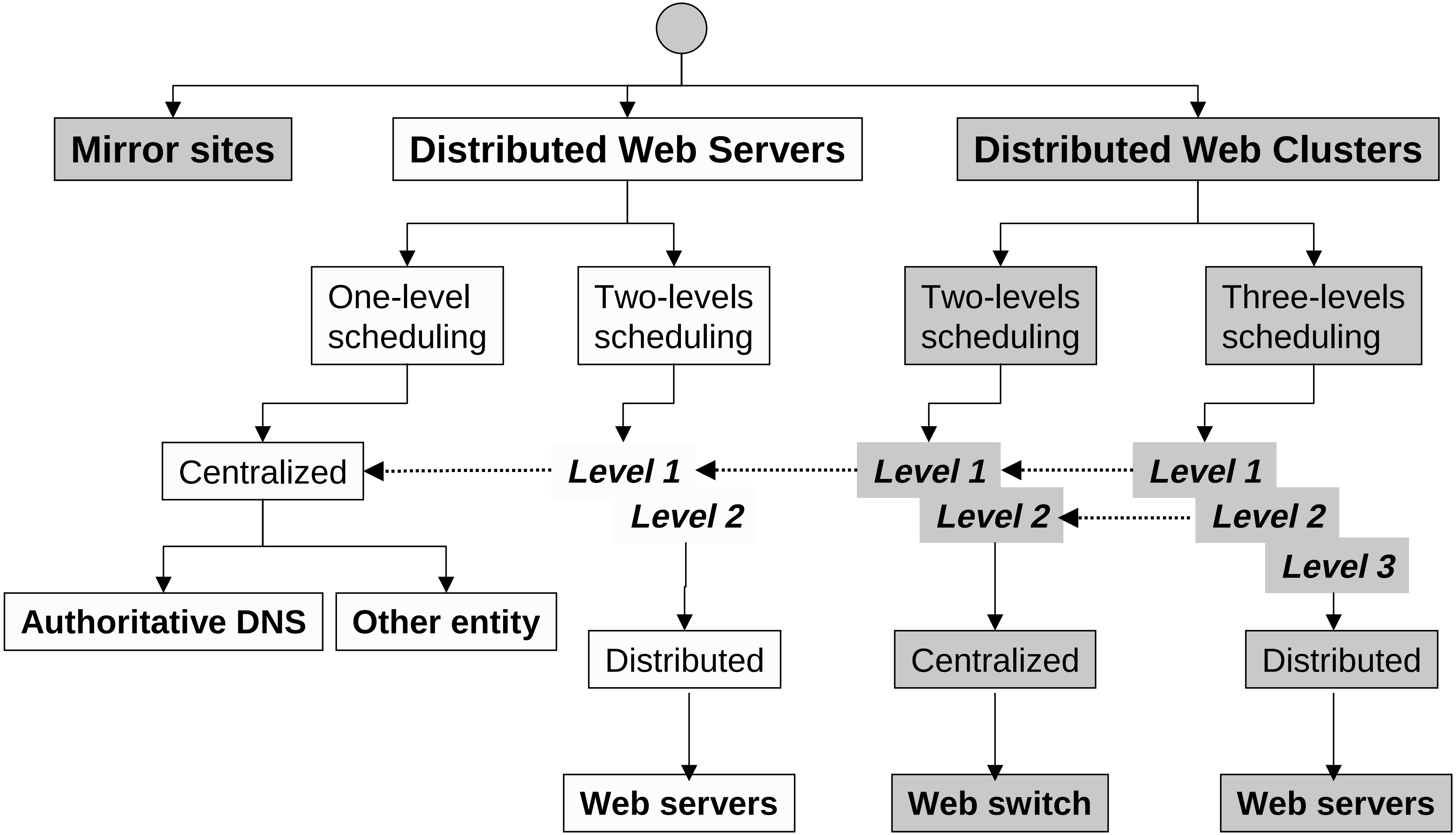
PRO

- Simple architecture

CONS

- Visibly replicated architecture
- It is very hard to maintain information consistency of Web sites
- No way of controlling load distribution

Globally Distributed Web Systems



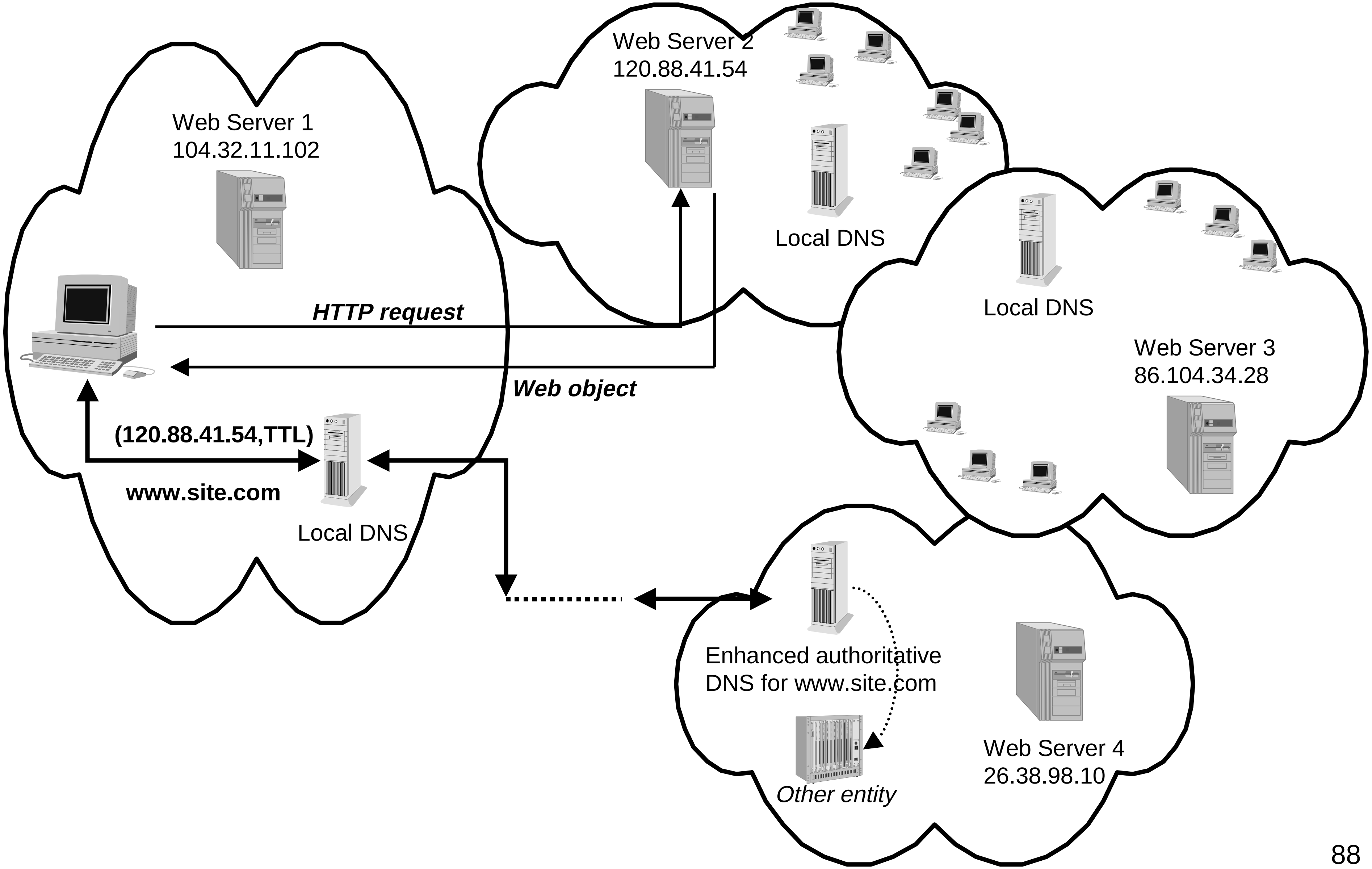
Distributed Web Servers

- *Web site realized on an architecture of geographically distributed Web servers*

- Web site addresses
  - One hostname (e.g., "www.site.com")
  - One IP address for each Web server

First level scheduling	***Second level scheduling***
The **enhanced authoritative DNS** of the Web site or **another entity** selects the "best" Web server	Each **Web server** may redirect the received page request to another server through the HTTP method

Distributed Web servers: *one-level scheduling*



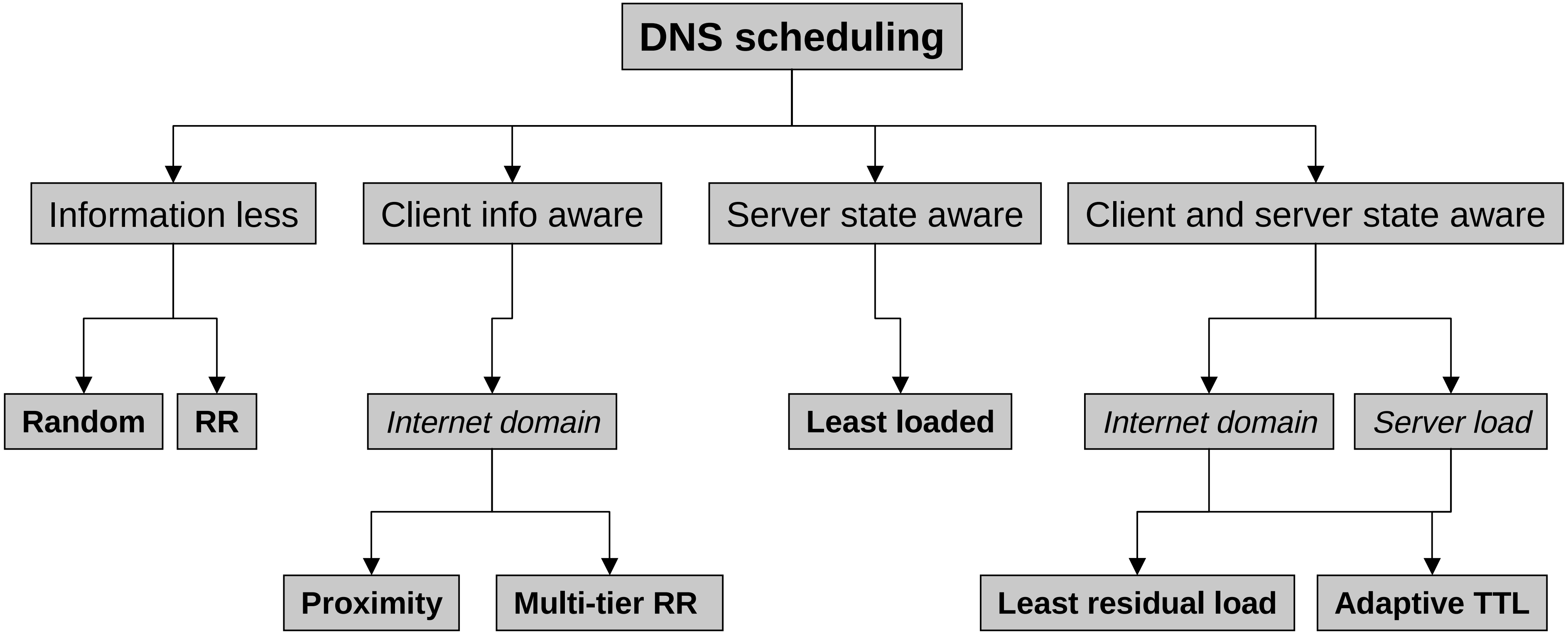
DNS scheduling

- The **distributed Web server** (**one-level**) architectures implements global scheduling 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 scheduling policies to select the “best” Web server.

DNS scheduling algorithms*



* Classification and more details in [Col98b, Car99a]

Issues of global scheduling

Typical issues

- Load spikes in some hours/days

Additional issues

- Traffic depending on time zones [Hab98, Squ00]
- Client distribution among Internet zones
- Proximity between client and Web server

- (For DNS) **Caching of [hostname-IP] at intermediate DNSes for TTL interval**

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 [Pax97a])

      → *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 (cont'd)

– ***Dynamic* evaluation of proximity**
  - round trip time (e.g., `ping`, `tcping` [Dyk00])
  - available link bandwidth (e.g., `cprobe` [Car97])
  - 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 [Cro95]

– "Recent" measures: strong [McM99], reasonably strong [Obr99]

Actions on TTL

- **Constant TTL**
  - Set TTL=0 to augment DNS control [CisDD, Sch95, Bec98]
  - Drawbacks
    - Not cooperative DNSes
    - 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 [Col98a]

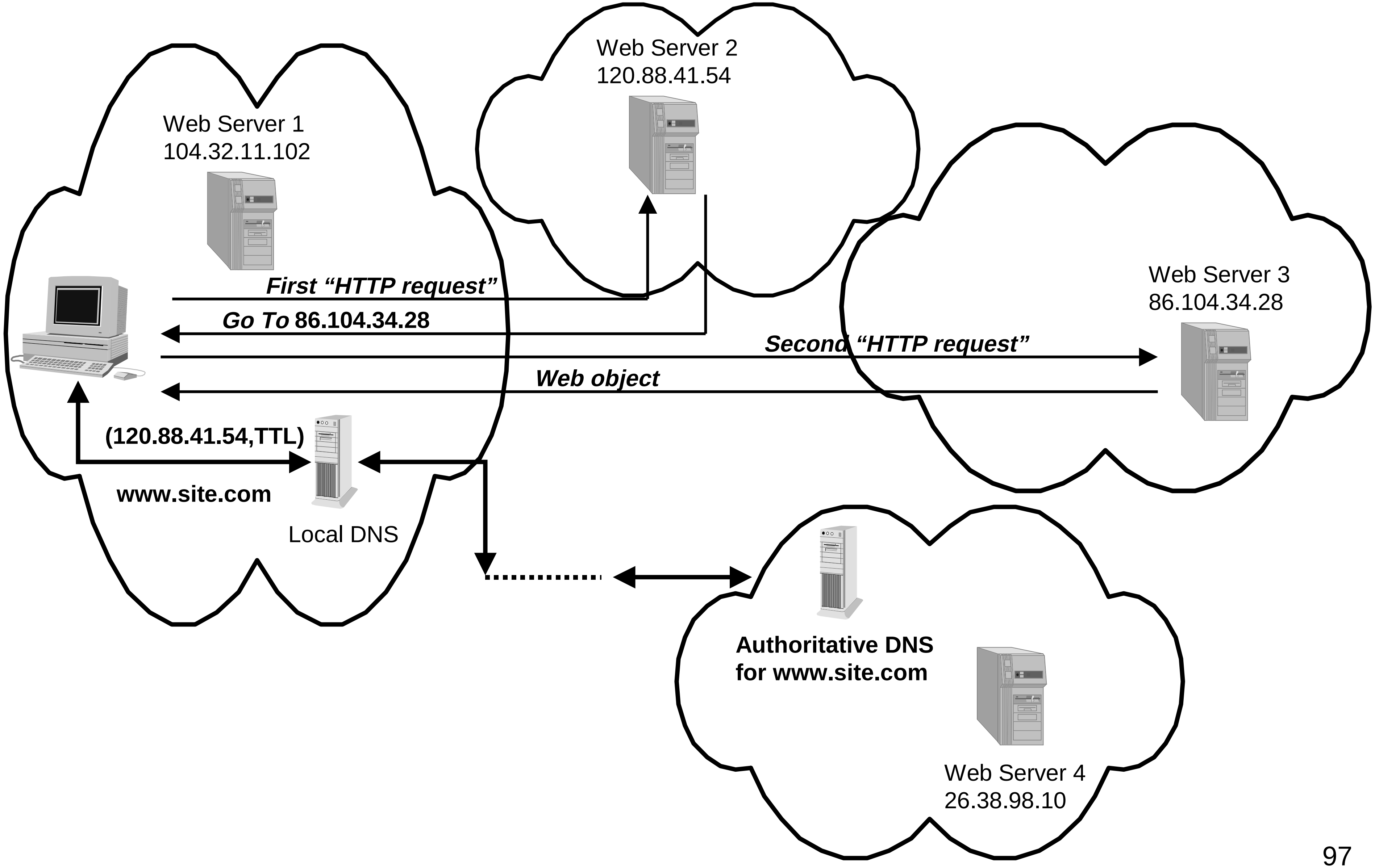
DNS scheduling: summary

- Because of *hostname-IP* caching, the DNS of highly popular Web sites controls only **5-7%** of traffic reaching the servers of the site (IBM source data)
- Reducing TTL has some limits:
  - TTL does not work on browser caches
  - non cooperative name servers ignore very small TTL values
- Unlike Web switch (controlling 100% traffic), the DNS should use sophisticated algorithms (e.g., *adaptive TTL*)
- **Nevertheless, we did not find any DNS scheduling algorithms (*does it exist?*) that is able to balance the load for <u>any workload scenario</u>**

Addressing DNS scheduling issues

- **Replacing DNS scheduling with another entity scheduling**
  - HTTP redirection [Gar95, CiscoDD]
- **Integrating DNS scheduling with Web server scheduling**
  - **HTTP redirection**
  - IP tunneling [Bes98, Lin]
- **Replacing Web servers with Web clusters**

Distributed Web servers: *two-levels scheduling*



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!**)

> *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

- **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** (Web server to which redirecting requests)
  - Round Robin (**RR**)
  - Hash function
  - Least loaded server (**Load**)
  - Client-to-server proximity (**Prox**)

HTTP redirection: pros and cons

PROS

- HTTP redirection is fully compatible with Web client and server because it is implemented at the application level
- Its distributed implementation satisfies dependability requirement because it does not introduce a single point of failure in the system

CONS

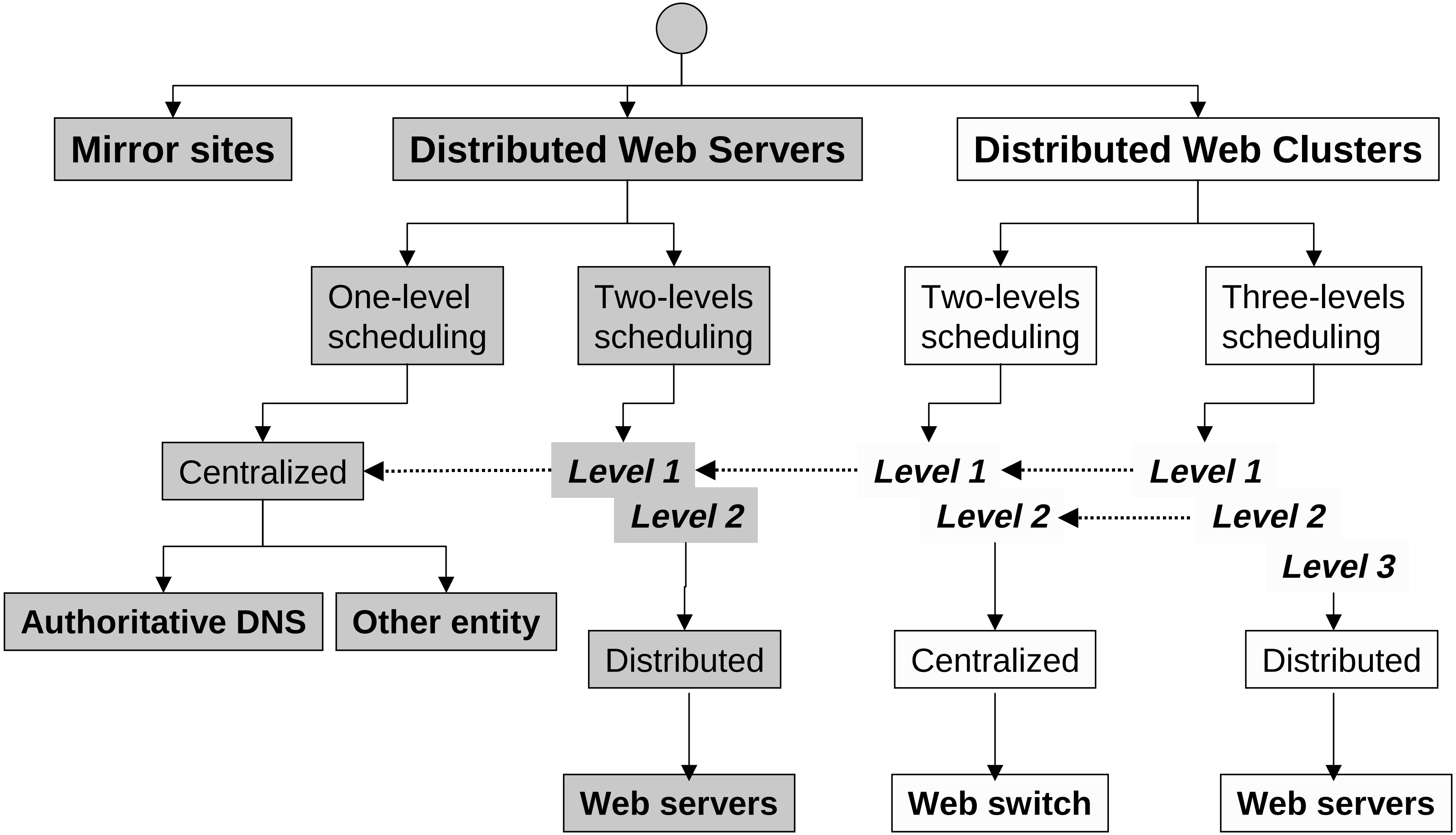
- It limits redirection to HTTP requests only
- It may increase response time and network traffic, as each redirected request requires two HTTP connections

Distributed Web servers proposals

One-level scheduling	**Two-levels scheduling**
Authoritative DNS / other entity	*DNS+server redirection*
• **NCSA server*** [Kwa95] • **CISCO** DistributedDirector [CisDD] • **lbnamed*** [Sch95] • [Col98a, Col98b, Car99c]***** • **EDDIE** [Edd] • **I2-DSI** [Bec98] • **SunSCALR*** [Sin98]	• **SWEB*** [And97] • [Car99b]***** • [Mou97]

**Originally proposed for locally distributed Web servers*

Globally Distributed Web Systems



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 scheduling

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

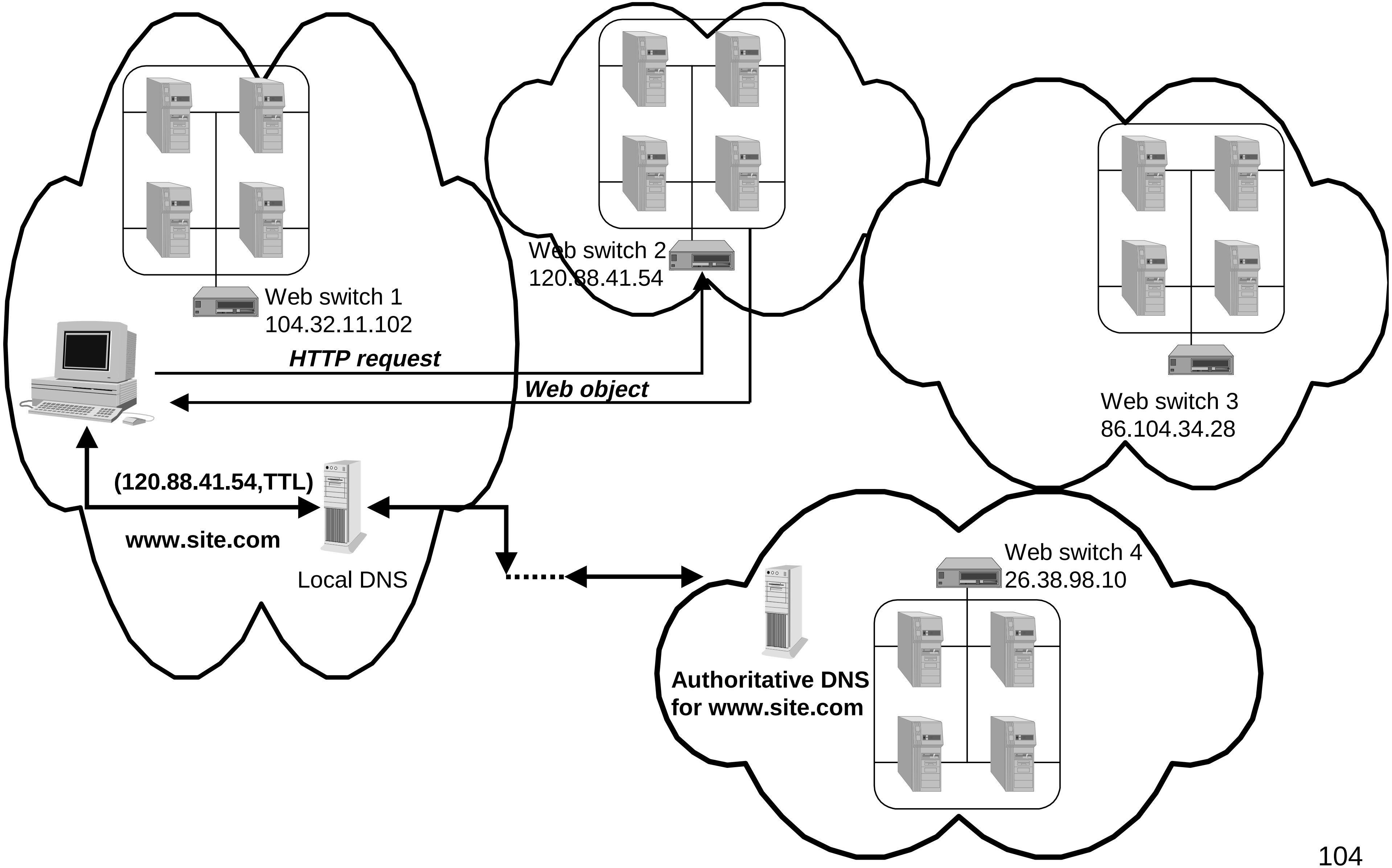
Second level scheduling

Web switch of the Web cluster selects one server

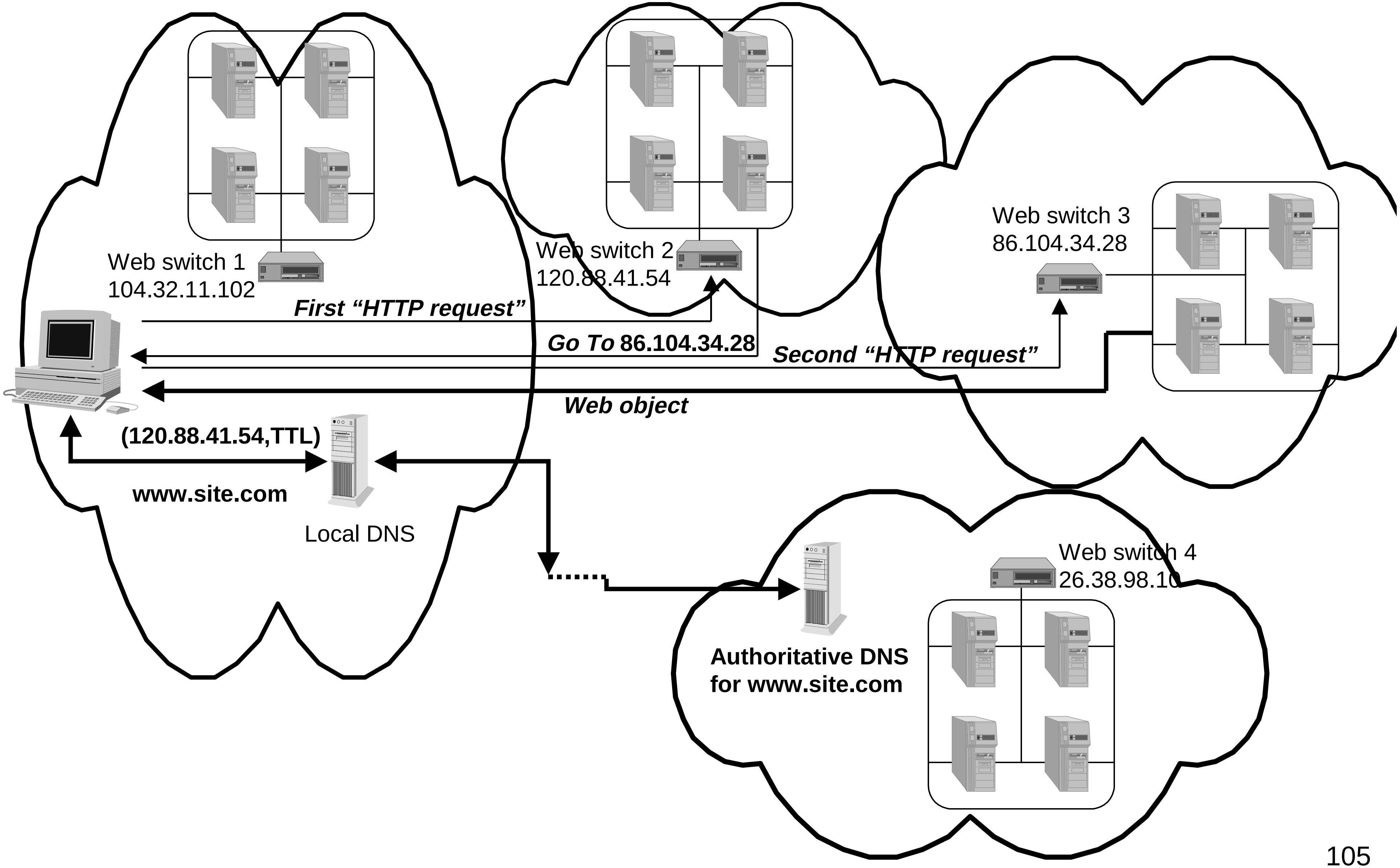
Third level scheduling

Each **Web server** may redirect the received request to another server (say, through the HTTP mechanism)

Distributed Web clusters: *two-levels scheduling*



Distributed Web clusters: *three-levels scheduling*



Three scheduling levels

- **Level 1: DNS**
  - Round robin
  - proximity
- **Level 2: Web switch**
  - static
  - client info and/or server state aware
- **Level 3: HTTP redirection**, carried out by
  - Web switches (only for Level 7)
  - Web servers

Distributed Web cluster proposals

Two-levels scheduling

DNS+Web switch

- **Alteon WebSystems'** GSLB [Alt]
- **CISCO's** DistributedDirector [CisDD]
- **Resonate's** Global Dispatcher [ResGD]
- **F5 Networks'** 3DNS [F5]
- **HydraWeb Techs.'** HydraHydra [Hyd]
- **Radware's** WSD-NP, WSD-DS
- **Coyote Point Systems'** Envoy [Coy]
- **IBM** Network Dispatcher ISS [IBMND, Iye00b]
- **Foundry Networks'** GSLB ServerIron [Fou]
- **Radware** WSD-NP [Rad]

Three-levels scheduling

DNS+Web switch+servers

- **Hermes** [Car00b]
- **RND Networks'** WSD-DS [Rnd]
- **Arrowpoint Comm.'s** Content Smart Redirect [Arr]
- **Radware** WSD-DS [Rad]

Distributed Web architectures: summary

Distributed Web servers (one-level scheduling)

- Easy to implement
- DNS scheduling
  - valid for all Web services
  - very limited control on load reaching the Web site
- HTTP redirection
  - valid only for HTTP services
  - partial control on load reaching the Web site

Distributed Web cluster (two-levels scheduling)

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

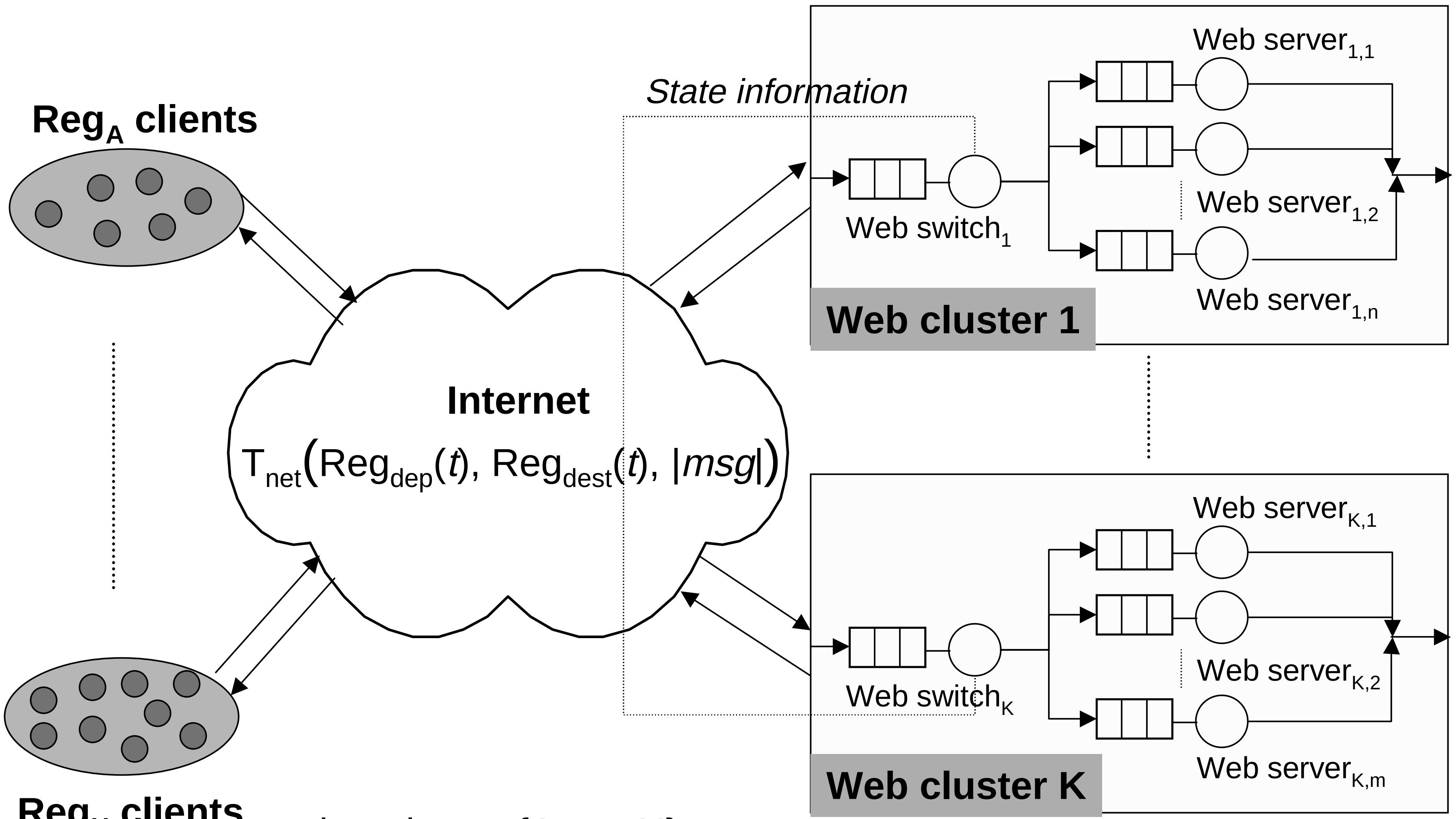
Distributed Web cluster (three-levels scheduling)

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

An example of performance comparison

- Two-levels *vs.* Three-levels scheduling
- System model
- Scheduling algorithms
  - Level 1
    - proximity
  - Level 2
    - WRR
  - Level 3
    - Selection policy (*page requests*): All, Size, Num
    - Location policy (*servers*): RR, load, proximity
- Metrics
  - Response time
  - Redirection percentage (for three-levels scheduling)

System model

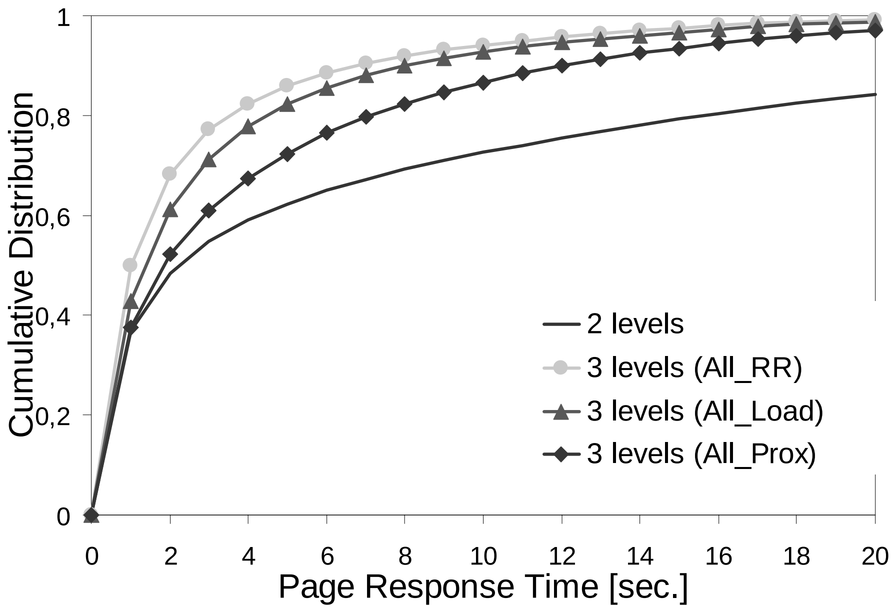


dep, dest $\in$ {A, …, H}

Choice of parameters

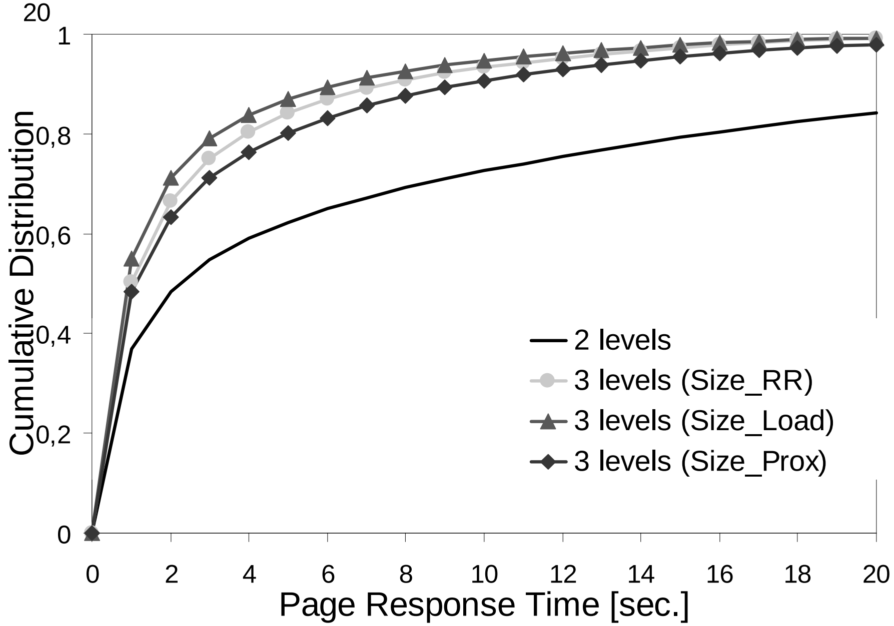
Category	Type		Parameters
Web site	Web clusters		4
	Web servers per cluster		4-8
	Disk transfer rate		20 MBps
	Intra-cluster bandwidth		100 Mbps
Client	Distribution among zones		Zipf (α=0.2)
	Arrival rate		700 clients per second (**cps**)
	User think time		Pareto (α=1.4, k=1)
	Page requests per session		Inverse Gaussian (μ=3.86, λ=9.46)
	Embedded objects per page		Pareto (α=1.245, k=2)
	Inter-arrival time of hits		Weibull (α=7.640, σ=1.705)
	Object size	(*body*)	Lognormal (μ=7.640, σ=1.001)
		(*tail*)	Pareto (α=1, k=10240)
Zones	Time zones		4
	Hours		24
	Day		Week

Results

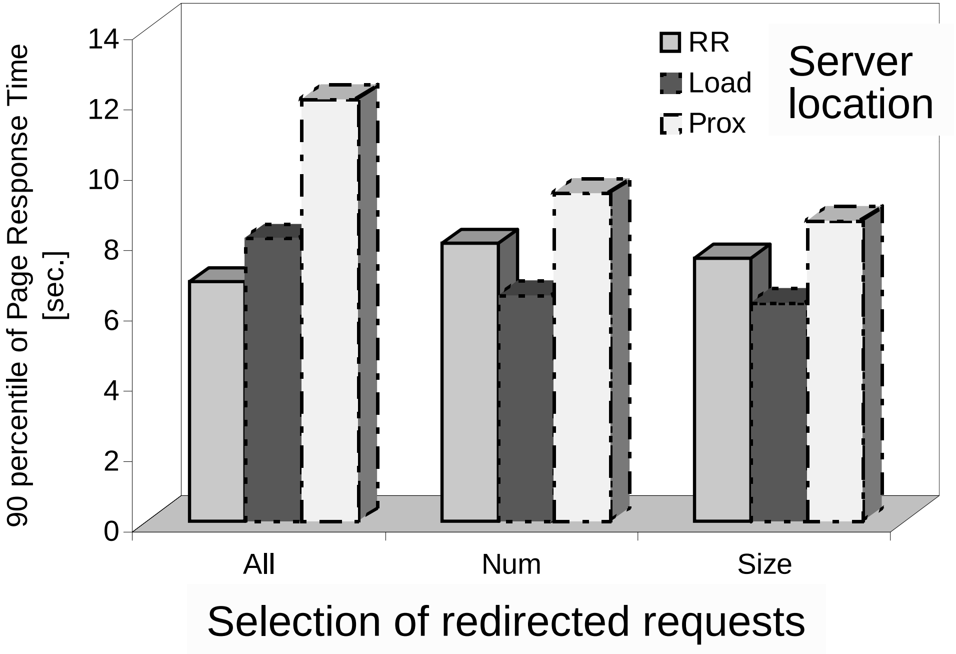


2 levels scheduling
vs.
3 levels scheduling (*Redirect **All***)

2 levels scheduling
vs.
3 levels scheduling (*Redirect **Some***)

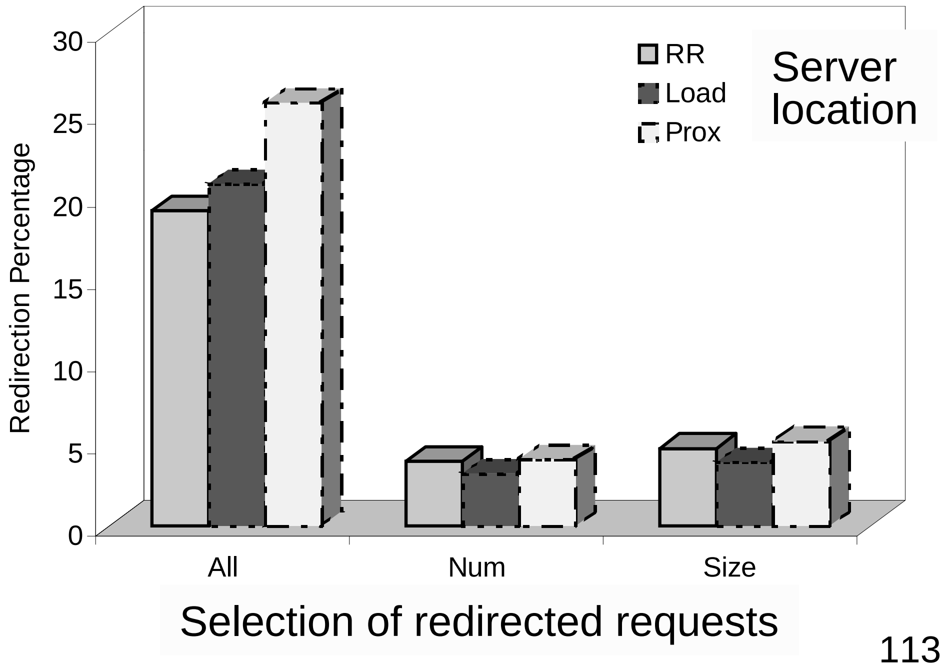


Results (cont'd)



RESPONSE TIME
3 levels scheduling (*Redirect **All***)
vs.
3 levels scheduling (*Redirect **Some***)

REDIRECTION PERCENTAGE
3 levels scheduling (*Redirect **All***)
vs.
3 levels scheduling (*Redirect **Some***)



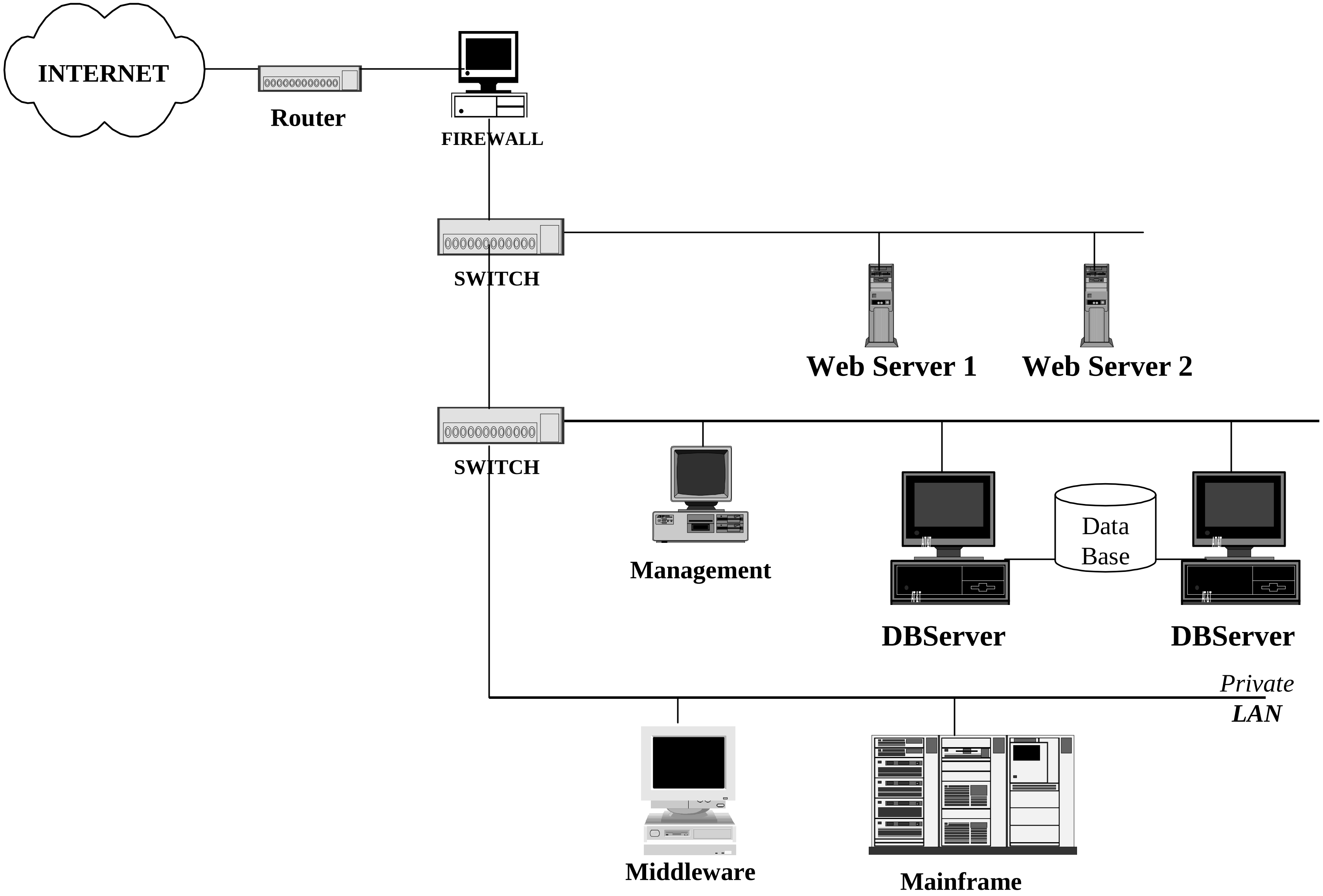
Part 5

Case study, Summary, Bibliography

Case study

- Context
  - **e-banking transactions**
    - normal requests
    - secure requests
- Goals
  - Investigate requirements for **scale-up**, **local scale-out**, **global scale-out** of Web architectures
- Assumptions
  - We consider some of the best architectures/algorithms seen before
  - The company has enough money to add more servers to the system when needed
- QoWS metrics
  - **90 percentile of response time for *normal* and *secure* requests must be less than 1 second**

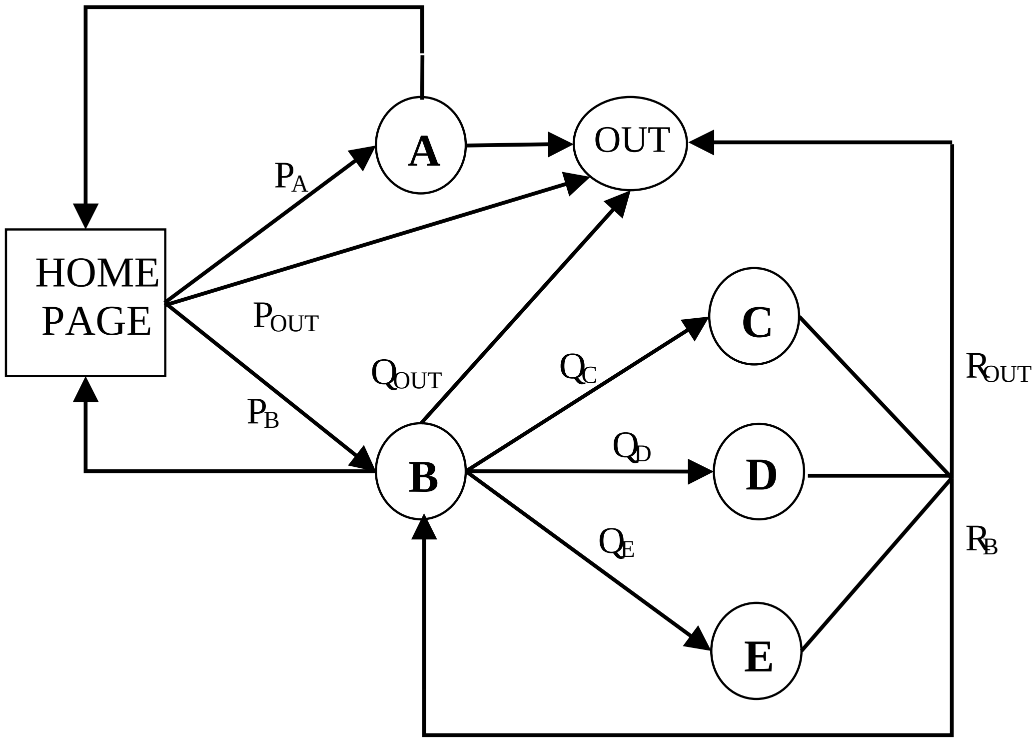
A simple e-banking architecture model



Classes of requests

- **Normal requests**
- **Secure requests**
  - CPU bound operations (by ***Front-end servers***)
    - cipher and decipher operations on information
      - → hashing algorithms (e.g., MD5) for digital signature
      - → public key cipher algorithms (e.g., RSA) during handshaking phase of SSL protocol
  - Disk bound operations (by ***Back-end servers***)
    - database operations for commercial transactions
      - → **new orders** (*light disk load*)
      - → **payments** (*mid disk load*)
      - → **stock-level** (*high disk load*)

State diagram of user session



A=**normal requests**

B=begin **secure session**

C=DB secure requests (*light*)

D=DB secure requests (*med*)

E=DB secure requests (*heavy*)

Probabilities get by real traces

e-banking workload

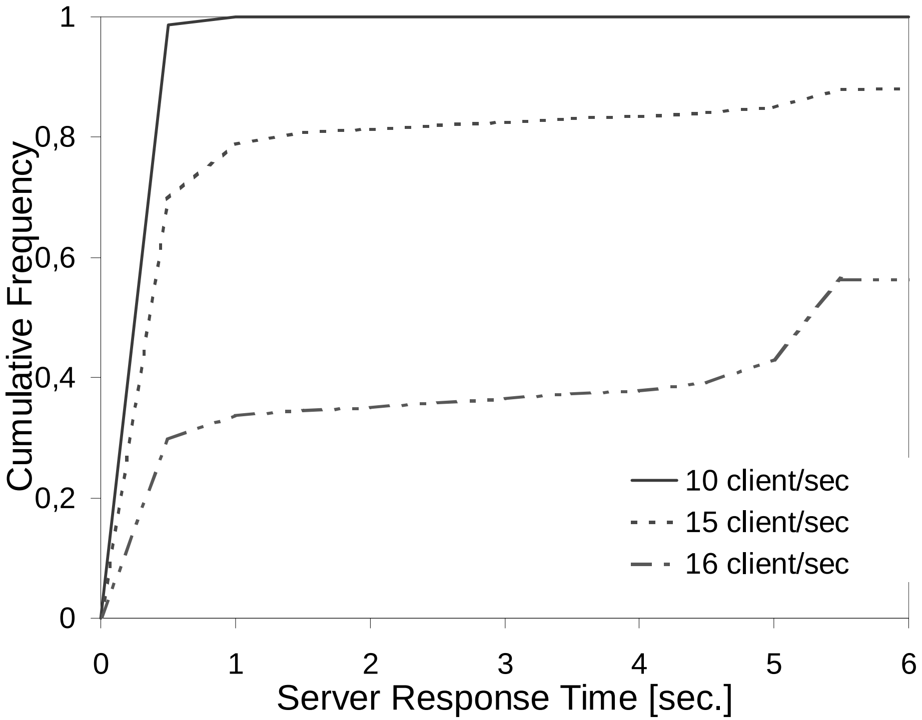
- P_A *normal requests*

- P_B *secure requests* that use SSL connections

- Service time for a *secure request* consists of
  - Service time for normal request
  - SSL overhead (cryptography + protocol)
  - DBMS request time
    - light disk load (Q_C=0.85, E[T_C]=80msec)
    - mid disk load (Q_D=0.10, E[T_D]=200msec)
    - high disk load (Q_E=0.05, E[T_E]=500msec)

Results: scale-up

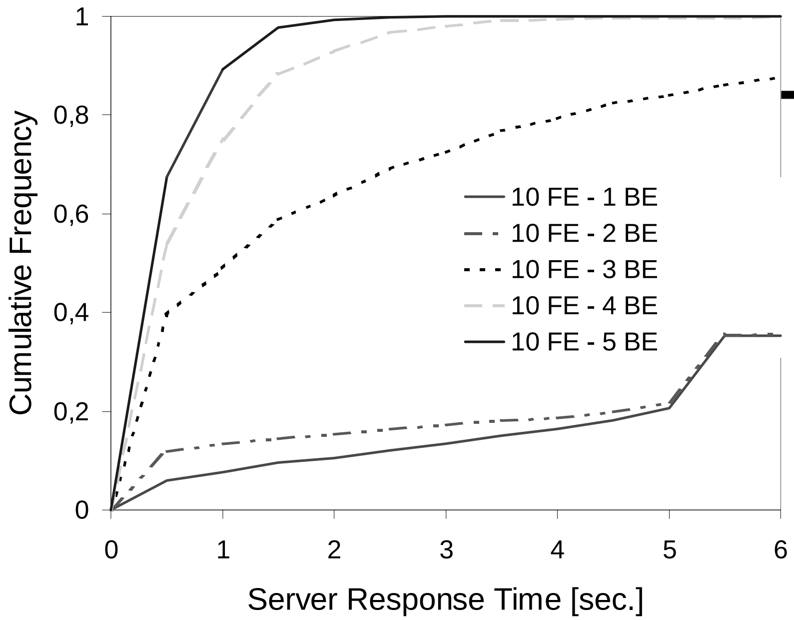
Scenario: *n* Web servers (**FE**), *m* DB servers (**BE**)



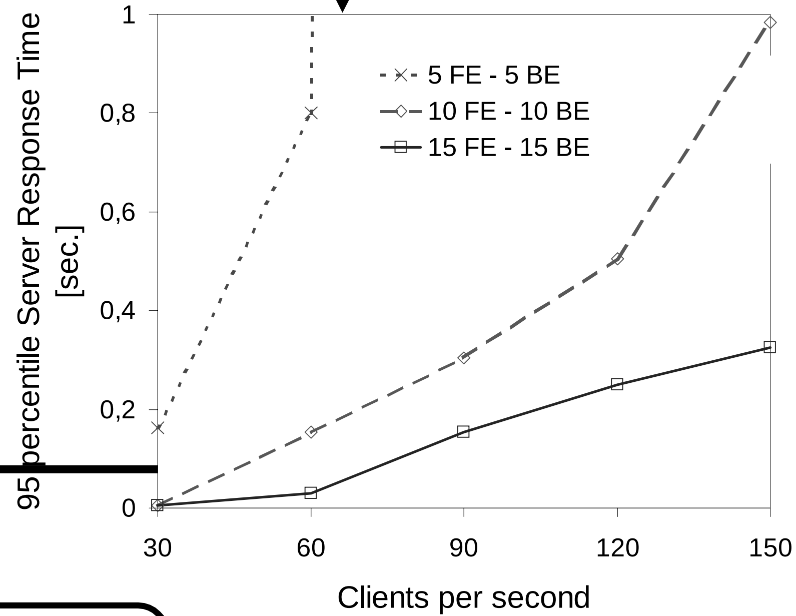
The bottleneck is the Front-End because of "secure requests" load

Local Scale-out

Results: scale-up



For **80cps**, a Web cluster with *n* FE and *n*/2 BE guarantees QoWS



For **150cps**, the Web cluster with *n=15* FE still guarantees QoWS, but its throughput is 4.9 MBps (≈ T3 connection)

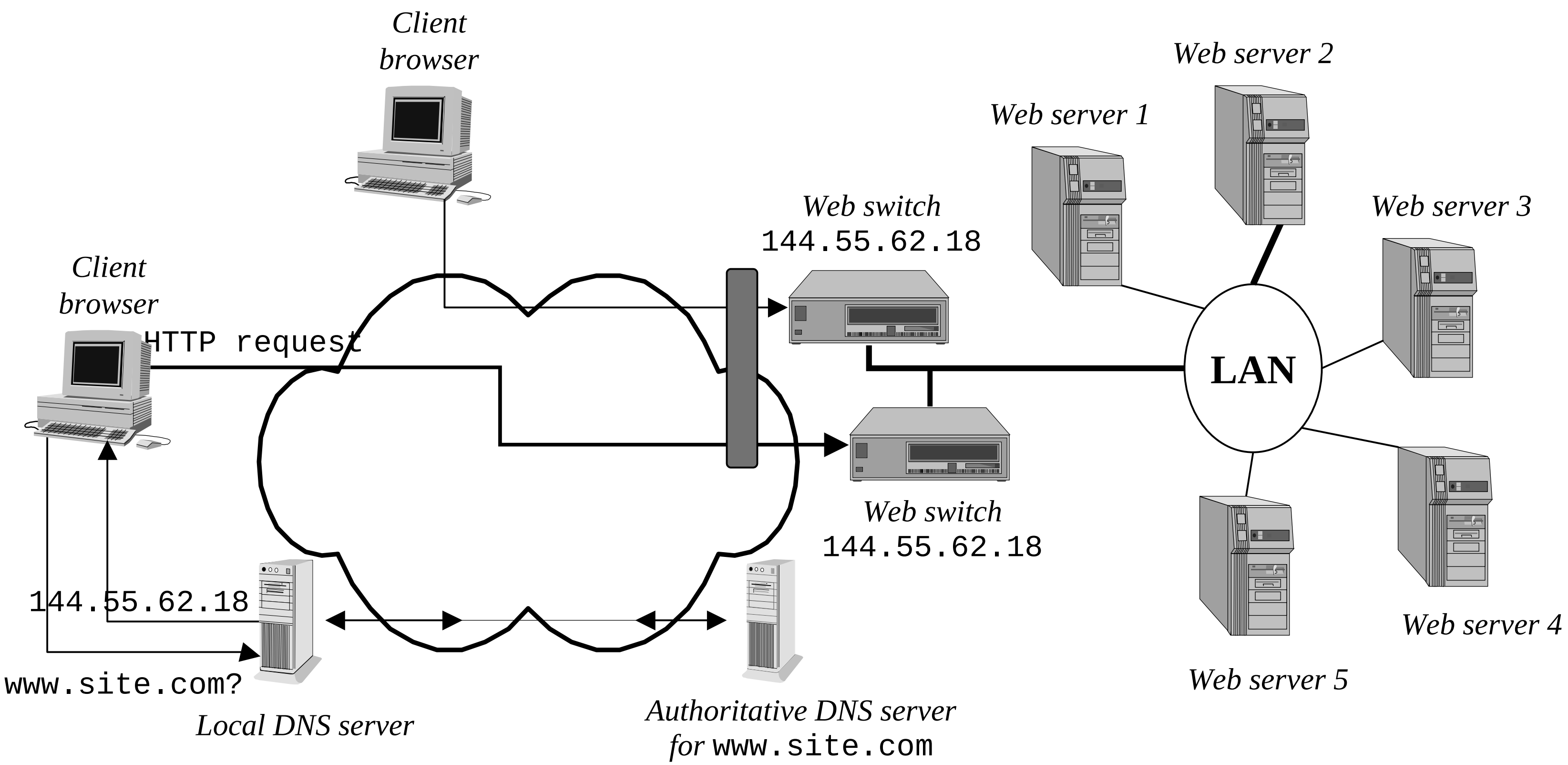
Global Scale-out

Scalable Web systems: *What's more ...*

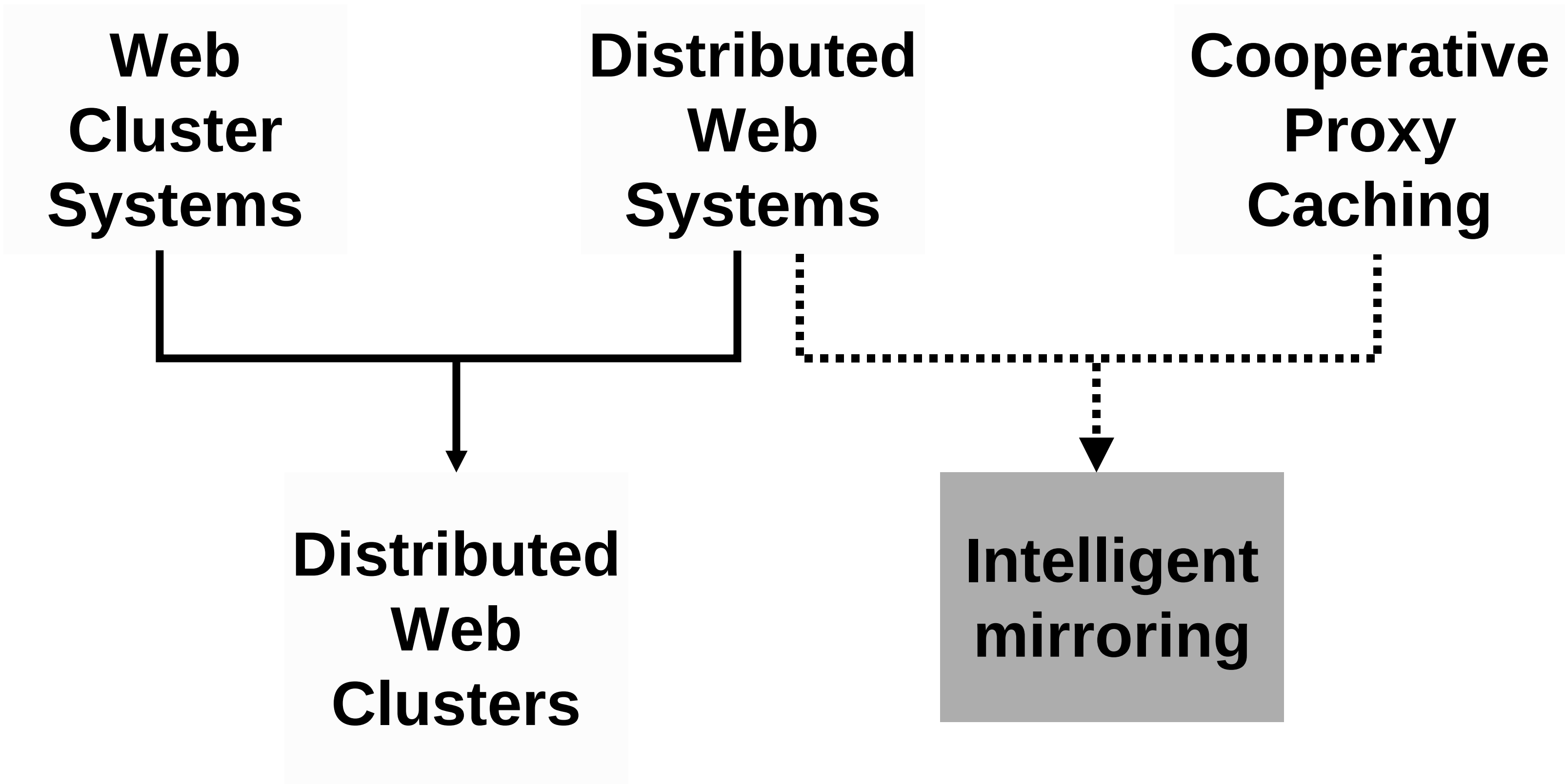
- ***For Web cluster solutions***
  - Architectures with multiple Web switches, e.g.,
    - IBM Network Dispatcher for Olympics Web site [Iye00b]
    - CISCO MultiNode Load Balancing [CisMN]
  - Multi-tier architectures

- ***For distributed Web system solutions***
  - Virtual servers (reverse proxies)
  - Intelligent mirroring
    - Akamai [Aka]
    - Mirror Image [Mir]
    - Adero [Ade]

Web cluster with multiple switches



Intelligent mirroring



QoWS ➡ QoNS (*still a long way…*)

- Workload characterization (not only for Web publishing sites)
- Real time and pro-active monitoring tools
- Solutions for secure, reliable, fast Web services
  - at Web server level
  - at Web application server level
- Solutions for *universal access* to Web information and services

Bibliography

[Ade] Adero Inc., http://www.adero.com

[Aka] Akamai Technologies, http://www.akamai.com

[Alt] Alteon Web Systems, http://www.alteonwebsystems.com

[And96] E. Anderson, D. Patterson, E. Brewer, “The Magicrouter, an application of fast packet interposing", University of California, Berkeley, May 1996. http://www.cs.berkeley.edu/~eanders/projects/magicrouter/osdi96-mr-submission.ps

[And97] D. Andresen, T. Yang, O.H. Ibarra, “Toward a scalable distributed WWW server on workstation clusters", *J. Parallel and Distributed Computing*, Vol. 42, pp. 91-100, 1997.

[Arl97] M.F. Arlitt, C.L. Williamson, “Web-server workload characterization: The search for invariants", *IEEE/ACM Trans. on Networking*, Vol. 5, No. 5, pp. 631-645, Oct. 1997.

[Arl00] M. Arlitt, T. Jin, “A workload characterization study of the 1998 World Cup Web site”, *IEEE Network*, May/June 2000.

[Aro99] M. Aron, P. Druschel, W. Zwaenepoel, “Efficient support for P-HTTP in cluster-based Web servers”, *Proc. USENIX 1999*, Monterey, CA, June 1999.

[Arr] ArrowPoint Communications, http://www.arrowpoint.com

[Bar98] P. Barford, M.E. Crovella, “Generating representative Web workloads for network and server performance evaluations”, *Proc. ACM Sigmetrics 1998*, pp. 151-160, July 1998.

[Bar99a] P. Barford, A. Bestavros, A. Bradley, M.E. Crovella, “Changes in Web client access patterns: Characteristics and caching implications”, *World Wide Web*, Baltzer Science, Vol. 2, No. 1-2, Mar. 1999.

[Bar99b] P. Barford, M.E. Crovella, “A performance evaluation of HyperText Transfer Protocols”, *Proc. ACM Sigmetrics 1999*, Atlanta, May 1999.

[Bar99c] P. Barford, M.E. Crovella, “Measuring Web performance in the wide area”, *ACM Performance Evaluation Review*, Sept. 1999.

Bibliography (cont’d)

[Bar00] G. Barish, K. Obraczka, "World Wide Web caching: Trends and techniques", *IEEE Communications*, Vol. 38, No. 5, May 2000.

[Bec98] M. Beck, T. Moore, “The Internet2 Distributed Storage Infrastructure project: An architecture for Internet content channels”, *Proc. 3th Int’l Web Caching Workshop*, Manchester, UK, June 1998.

[Bes98] A. Bestavros, M. E. Crovella, J. Liu, D. Martin, “Distributed Packet Rewriting and its application to scalable server architectures'', *Proc. 6th IEEE Int’l Conf. Network Protocols*, 1998.

[Bri95] T. Brisco, “DNS support for load balancing'', *RFC 1794*, Apr. 1995.

[Bun99] R.B. Bunt, D.L. Eager, G.M. Oster, C.L. Williamson, “Achieving load balance and effective caching in clustered Web servers”, *Proc.4th Int’l Web Caching Workshop*, San Diego, pp. 159-169, Apr. 1999.

[Car97] R.L. Carter, M.E. Crovella, “Server selection using dynamic path characterization in wide-area networks”, *Proc.IEEE Infocom 97*, Kobe, Japan, Apr. 1997.

[Car99a] V. Cardellini, M. Colajanni, P.S. Yu, “Dynamic load balancing on Web-server systems'', *IEEE Internet Computing*, Vol. 3, No. 3, pp. 28-39, May/June 1999.

[Car99b] V. Cardellini, M. Colajanni, P.S. Yu, “Redirection algorithms for load sharing in distributed Web-server systems'', *Proc. IEEE 19th Int’l Conf. on Distributed Computing Systems*, Austin, TX, pp. 528-535, June 1999.

[Car99c] V. Cardellini, M. Colajanni, P.S. Yu, “DNS dispatching algorithms with state estimators for scalable Web-server clusters'', *World Wide Web*, Baltzer Science, Vol. 2, No. 3, pp. 101-113, July 1999.

[Car00a] V. Cardellini, M. Colajanni, P.S. Yu, “Impact of workload models in evaluating the performance of distributed Web-server systems'', in *System Performance Evaluation: Methodologies and Applications*, E. Gelenbe ed., CRC Press, Mar. 2000.

[Car00b] V. Cardellini, M. Colajanni, P.S. Yu, “Geographic load balancing for scalable distributed Web systems'', *Proc. IEEE Mascots 2000*, San Francisco, August 2000.

Bibliography (cont'd)

[Cas00] E. Casalicchio, M. Colajanni, S. Tucci, “Global scheduling algorithms for high performance Web server clusters”, Tech. Rep. DISP-2000-05, University of Roma Tor Vergata, Feb. 2000.

[Cho00] C.-F. Chou, L. Golubchik, J. Lui, “Striping doesn't scale: How to achieve scalability for continuous media servers with replication”, Proc. 20th Int. Conf. on Distributed Computing Systems, pp. 64-71, Apr. 2000.

[CisDD] Cisco's DistributedDirector, http://www.cisco.com/warp/public/cc/cisco/mkt/scale/distr/index.shtml

[CisLD] Cisco's LocalDirector, http://www.cisco.com/warp/public/cc/cisco/mkt/scale/locald/

[CisMN] Cisco's MultiNode Load Balancing,http://www.cisco.com/warp/public/cc/cisco/mkt/iworks/data/mnlb/

[Coh99] A. Cohen, S. Rangarajan, H. Slye, “On the performance of TCP splicing for URL-aware redirection”, *Proc. 2nd USENIX Symp. On Internet Technologies and Systems*, Oct. 1999.

[Col98a] M. Colajanni, P.S. Yu, V. Cardellini, “Dynamic load balancing in geographically distributed heterogeneous Web-servers'', *Proc. IEEE 18th Int'l Conf. on Distributed Computing Systems*, Amsterdam, pp. 295-302, May 1998.

[Col98b] M. Colajanni, P.S. Yu, D.M. Dias, “Analysis of task assignment policies in scalable distributed Web-server system'', *IEEE Trans. on Parallel and Distributed Systems*, Vol. 9, No. 6, June 1998.

[Con99] M. Conti, E. Gregori, F. Panzieri, “Load distribution among replicated Web servers: A QoS-based approach”, *Proc. 2nd Workshop on Internet Server Performance*, Atlanta, May 1999.

[Cro95] M.E. Crovella, R.L. Carter, “Dynamic server selection in the Internet”, *Proc. 3rd IEEE Workshop on Architecture and Implementation of High Performance Comm. Subsystems*, New York, Aug. 1995.

[Cro97a] M. E. Crovella, A. Bestavros, “Self-similarity in World Wide Web traffic: Evidence and possible causes'', *IEEE/ACM Trans. on Networking*, Vol. 5, No. 6, Dec. 1997, pp. 835-846.

[Cro97b] M.E. Crovella, L. Lipsky, “Long-lasting transient conditions in simulations with heavy-tailed workloads”, *Proc. 1997 Winter Simulation Conf.*, Atlanta, GA, 1997.

Bibliography (cont'd)

[Dam97] O.P. Damani, P.E. Chung, Y.Huang, C. Kintala, Y.-M. Wang, "ONE-IP: Techniques for hosting a service on a cluster of machines'', *J. Computer Networks and ISDN Systems*, Elsevier, Vol. 30, 1997.

[Dia96] D.M. Dias, W. Kish, R. Mukherjee, R. Tewari, "A scalable and highly available Web-server'', *Proc. 41st IEEE Computer Society Int'l Conf.*, pp. 85-92, Feb. 1996.

[Dik00] S.G. Dikes, C.L. Jeffery, K.A. Robbins, "An empirical evaluation of client-side server selection algorithms", *Proc. IEEE Infocom 2000*, Mar. 2000.

[Edd] Eddieware, http://www.eddieware.org

[Ege94] K. Egevang, P.Francis, "The IP Network Address Translator (NAT)'', *RFC 1631*, May 1994.

[F5] F5 Networks, http://www.f5labs.com

[Fou] Foundry Networks, http://www.foundrynetworks.com

[Fox97] A. Fox, S.D. Gribble, Y. Chawathe, E.A. Brewer, P. Gauthier, "Extensible cluster-based scalable network services", *Proc. 16th ACM Symp. On Operating System Principles*, San Malo, France, 1997.

[Gan00] X. Gan, B. Ramamurthy, "LSMAC: An improved load sharing network service dispatcher", *World Wide Web*, Baltzer Science, Vol. 3, No. 1, 2000.

[Gar95] M. Garland, S. Grassia, R. Monroe, S. Puri, "Implementing Distributed Server Groups for the World Wide Web'', Tech. Rep. CMU-CS-95-114, Carnegie Mellon Univ., School of Computer Science, 1995.

[Hab98] M.A. Habib, G. Abdulla, E.A. Fox, "Web traffic characterization with time zones", *Proc. ISAVIIA 1998*, Germany, Aug. 1998.

[Har99] M. Harchol-Balter, M.E. Crovella, C.D. Murta, ``On choosing a task assignment policy for a distributed server system'', *J. of Parallel and Distributed Computing*, Vol. 59, pp. 204-228, 1999.

[Hei97] J. Heidemann, K. Obraczka, J. Touch, "Modeling the performance of HTTP over several transport protocols", *IEEE/ACM Trans. on Networking*, Vol. 5, No. 5, pp. 616-630, Oct. 1997.

Bibliography (cont'd)

[Hun98] G.D.H. Hunt, G.S. Goldszmidt, R.P. King, R. Mukherjee, ``Network Dispatcher: A connection router for scalable Internet services'', *J. Computer Networks and ISDN Systems*, Elsevier, Vol. 30, 1998.

[Hyd] HydraWeb Technologies, http://www.hydraweb.com

[IBMND] IBM Network Dispatcher, http://www.ibm.com/software/network/dispatcher/

[Iye99] A.K. Iyengar, M.S. Squillante, L. Zhang, "Analysis and characterization of large-scale Web server access patterns and performance", *World Wide Web*, Baltzer Science, Vol. 2, No. 1-2, Mar. 1999.

[Iye00a] A.K. Iyengar, D. Rosu, D. Dias, "Hint-based acceleration of Web proxy caches", *Proc. 19th IEEE Int. Performance, Computing, and Communications Conference*, Phoenix, Feb. 2000.

[Iye00b] A.K. Iyengar, J. Challenger, D. Dias, P. Dantzig, "High-performance Web site design techniques", *IEEE Internet Computing*, pp. 17-26, Mar./Apr. 2000.

[Jin00] S. Jin, A. Bestavros, "Temporal locality in Web request streams", Proc. ACM Sigmetrics 2000 (extended abstract), Santa Clara, CA, June 2000.

[Kar98] M. Karaul, Y.A. Korilis, A. Orda, "A market-based architecture for management of geographically dispersed, replicated Web servers", *Proc. 1st Int'l Conf. Information and Computation Economics*, pp. 158-165, Charleston, SC, 1998.

[Kri99] B. Krishnamurthy, J.C. Mogul, D.M. Kristol, "Key differences between HTTP/1.0 and HTTP/1.1", *Proc. 8th Int'l World Wide Web Conf.*, Toronto, May 1998.

[Kwa95] T.T. Kwan, R.E. McGrath, D.A. Reed, "NCSA's World Wide Web server: Design and performance'', *IEEE Computer*, No. 1, pp. 68-74, Nov. 1995.

[Lie98] P.W.K Lie, J.C.S. Lui, L. Golubchik, "Threshold-based dynamic replication in large-scale video-on-demand systems", *Proc. 8th Int. Workshop on Continuous-Media Databases and Applications*, pp. 52-59, 1998.

[Lin] Linux Virtual Server, http://www.linuxvirtualserver.org

Bibliography (cont'd)

[Liu00] Z. Liu, N. Niclausse, C. Jalpa-Villanueva, "Web traffic modeling and performance comparison between HTTP1.0 and HTTP1.1", *System performance evaluation: Methodologies and applications*, E. Gelenbe ed., CRC Press, Mar. 2000.

[Luo98] A. Luotonen, *Web Proxy Servers*, Prentice Hall, 1998.

[Mah00] A. Mahanti, C. Williamson, D. Eager, "Traffic analysis of a Web proxy caching hierarchy", *IEEE Network*, May/June 2000.

[McM99] P. McManus, "A passive system for server selection within mirrored resource environments using as path length heuristics", http://proximate.appliedtheory.com, Apr. 1999.

[Mir] Mirror Image Internet, http://www.mirror-image.com

[Mog95] J.C. Mogul, "Network behavior of a busy Web server and its clients", Research Report 95/5, DEC Western Research Laboratory, Oct. 1995.

[Mol00] M. Molina, P. Castelli, G. Faddis, "Web traffic modeling exploiting TCP connections' temporal clustering through HTML-REDUCE", *IEEE Network*, May/June 2000.

[Mor00] R. Morris, "Variance of aggregated Web traffic", *Proc. IEEE Infocom 2000*, 2000.

[Mos97] D. Mosedale, W. Foss, R. McCool, "Lesson learned administering Netscape's Internet site", IEEE Internet Computing, Vol. 1, No. 2, pp. 28-35, Mar.-Apr. 1997.

[Mou97] A. Mourad, H. Liu, "Scalable Web server architectures", *Proc. IEEE Int'l Symp. on Computers and Communications*, Alexandria, Egypt, pp. 12-16, July 1997.

[Obr99] K. Obraczka, F. Silva, "Looking at network latency for server proximity", Tech. Rep. USC-99-714, Information Science Institute, Oct. 1999.

[Pai98] V.S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, E. Nahum, "Locality-aware request distribution in cluster-based network servers", *Proc. 8th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, San Jose, CA, Oct. 1998.

Bibliography (cont'd)

[Pan98] R. Pandey, J.F. Barnes, R. Olsson, "Supporting Quality of Service in HTTP servers", *Proc. ACM PODC'98*, pp. 247-256, Puerto Vallalta, Mexixo, 1998.

[Pax97a] V. Paxson, "End-to-end routing behavior in the Internet", *IEEE/ACM Trans. on Networking.*, Vol. 5, No. 5, pp. 601-615, Oct. 1997.

[Pax97b] V. Paxson, S. Floyd, "Why we don't know how to simulate the Internet", *Proc. 1997 Winter Simulation Conf.*, Atlanta, GA, 1997.

[Pir99] P. Pirolli, J.E. Pitkow, "Distributions of surfers' paths through the World Wide Web: Empirical characterization", *World Wide Web*, Baltzer Science, Vol. 2, No. 1-2, Mar. 1999.

[Pit99a] J.E. Pitkow, "Summary of WWW characterization", *World Wide Web*, Baltzer Science, Vol. 2, No. 1-2, pp. 3-13, Mar. 1999.

[Pit99b] J.E. Pitkow, P. Pirolli, "Mining longest repeating subsequences to predict World Wide Web surfing", *Proc. USENIX 1999*, Monterey, CA, June 1999.

[Rad] Radware Networks, http://www.radware.com

[ResCD] Resonate Central Dispatcher, http://www.resonate.com/products/central_dispatch

[ResGD] Resonate Global Dispatcher, http://www.resonate.com/products/global_dispatch

[Rnd] RND Networks , http://www.rndnetworks.com

[Sch95] R.J. Schemers, "lbmnamed: A load balancing name server in Perl", *Proc. 9th Systems Administration Conference*, Monterey, Sep. 1995.

[Sch00] T. Schroeder, S. Goddard, B. Ramamurthy, "Scalable Web server clustering technologies", *IEEE Network*, May/June 2000.

[Sin98] A. Singhai, S.-B. Lim, S.R. Radia, "The SunSCALR Framework for Internet servers", *Proc. FCTS 1998*, Munich, Germany, June 1998.

Bibliography (cont'd)

[Son00] J. Song, E. Levy, A. Iyengar, D. Dias, "Design alternatives for scalable Web server accelerators", *Proc. IEEE Int. Symp. on Performance Analysis of Systems and Software*, Austin, Texas, April 2000.

[Squ00] M.S. Squillante, D.D.Yao, L. Zhang, "Internet traffic: Periodicity, tail behavior, and performance implications", *System performance evaluation: Methodologies and applications*, E. Gelenbe ed., CRC Press, pp. 23-37, Mar. 2000.

[Sri98] P. Srisuresh, D. Gan, "Load sharing using IP Network Address Translation (LSNAT)", *RFC 2931*, 1998.

[Vas00] N. Vasiliou, H.L. Lutfiyya, "Providing a differentiated Quality of Service in World Wide Web server", *Proc. Proc. Performance and Architecture of Web Servers Workshop*, Santa Clara, California , June 2000.

[Vin00] R. Vingralek, M. Sayal, Y. Breitbart, P. Scheuermann, "Web++: Architecture, design, and performance", *World Wide Web*, Baltzer Science, Vol. 3, No. 2, 2000.

[Wil98] W. Willinger, V. Paxson, "Where Mathematics meets the Internet", *Notices of the American Mathematical Society*, Vol. 45, No. 8, Aug. 1998.

[Yos97] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, T. Anderson, D. Culler, ``Using Smart Clients to build scalable services'', *Proc. USENIX 1997*, Jan. 1997.

[Zhu99] H. Zhu, B. Smith, T. Yang, "Scheduling optimization for resource-intensive Web requests on server clusters", *Proc. 11th ACM Symp. On Parallel Algorithms and Architectures*, June 1999.