Request Redirection Algorithms for Distributed Web Systems *

Valeria Cardellini
University of Roma Tor Vergata
Roma, Italy 00133
cardellini@ing.uniroma2.it

Michele Colajanni
University of Modena
Modena, Italy 41100
colajanni@unimo.it

Philip S. Yu
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
psyu@us.ibm.com

Abstract

Replication of information among multiple servers is necessary to support high request rates to popular Web sites. We consider systems that maintain one interface to the users even if they consist of multiple nodes with visible IP addresses that are distributed among different networks. In these systems, the first-level dispatching is achieved through the Domain Name System (DNS) during the address lookup phase. Distributed Web systems can use some request redirection mechanism as a second-level dispatching, because the DNS routing scheme has limited control on offered load. Redirection is always executed by the servers, but there are many alternatives that are worth of investigation. In this paper, we explore the combination of DNS dispatching with redirection schemes that use centralized or distributed control, on the basis of global or local state information. In the fully distributed schemes, DNS dispatching is carried out by simple algorithms because load sharing are taken by some redirection mechanisms, that each server activates autonomously. On the other hand, in fully centralized schemes redirection is used as a tool to enforce the decisions taken by the same centralized entity that provides the first-level dispatching. We also investigate some hybrid strategies. We conclude that the distributed algorithms are preferable over the centralized counterpart because they provide stable performance, can take content-aware dispatching decisions, can limit the percentage of redirected requests and, last but not least, their implementation is much simpler than that required by the centralized schemes.

Index Terms: World Wide Web, Load balancing, Distributed systems, Dispatching algorithms, Performance analysis.

1 Introduction

A common approach adopted by popular Web sites to keep up with ever increasing request load and provide scalable Web-based services is to deploy a distributed Web system composed by multiple server nodes. A scalable Web-server system needs to appear as a single host to the outside world, so that users need not be concerned about the names or locations of the replicated servers and they can interact with the Web system as if it were one high performance server machine. This architecture provides scalability and transparency, but requires some internal mechanism that assigns client requests to the node which in that moment can provide, possibly, the minimum response time.

We will focus on Web system architectures consisting of multiple nodes that are distributed over different network locations. Each Web node may consist of one server machine or multiple machines, but each node makes visible only one IP address. In this system the first assignment decision is typically taken at the *Domain Name System* (DNS) level, when the *authoritative name server* (A-DNS) of the Web site maps the hostname to the IP address of one node in the system [1, 8]. DNS-dispatcher based systems can easily scale from locally to geographically distributed Web-server systems and are also used in other related distributed architectures, such as Content Delivery Networks. However, dispatching requests through the DNS has three problems that prevent load balancing among the Web nodes: routing decisions are content-blind, the different amount of load coming from various Internet regions may easily overload some Web nodes, and the address caching mechanism in the DNS causes the large majority of client requests to skip the A-DNS for address resolution, to the extent that the DNS dispatcher of a popular Web site controls only a small fraction of the requests reaching the Web site. These issues have been addressed through sophisticated DNS dispatching policies working on the TTL [8], or multiple tiers of proprietary name servers combined with very low values (few seconds) for the TTL [15]. This last approach has three main drawbacks. To avoid a system bottleneck at the A-DNS, the traffic for address resolutions requires a distributed architecture of name servers that is admissible for third party companies in the content delivery business, but not for the provider of one Web site. Moreover, if any client request needs an address resolution by the A-DNS, the response time perceived by users is likely to increase [21]. Finally, the TTL period chosen by the A-DNS does not work on browser caching, while low TTL values might be overridden by non-cooperative name servers that impose their minimum TTL when the suggested value is considered too low.

In this paper we follow other two directions for controlling the load among the nodes of a distributed Web system. We investigate the combination of sophisticated DNS dispatching algorithms with a redirection scheme controlled in a *centralized* way by the A-DNS. We also examine an alternative solution that integrates the first-level request dispatching carried out by the A-DNS through a simple stateless algorithm (e.g., round-robin) with some *distributed* redirection mechanism managed directly by the Web nodes.

We also need to recognize that load balancing goals are of key importance for the system administrator, but are worth for the user only if they contribute to reduce the response time. Recent measures suggest that Web performance perceived by end users is already increasingly dominated by server delays, especially when contacting busy Web sites [5]. From this point of view, request redirection is a typical trade-off based mechanism because each redirection consumes resources of the first contacted server and increases the network time component of the user response time. Hence, these costs must be paid by the gains obtainable by a sizable reduction of the server component of the response time. The goal of reducing the percentage of the redirected requests opens a new space of alternatives for centralized and distributed schemes. To investigate when it is appropriate to activate redirection and to propose more efficient *content-aware* control

policies for redirection are other key objectives of this work.

In summary, this paper makes four main contributions and identifies some new research issues that are worth of further investigation. We provide a taxonomy of the alternative re-routing schemes including centralized vs. distributed control algorithms for the activation of the redirection mechanism, for the localization of the destination nodes, and for the selection of the requests to be redirected. These control policies can be based on local, partial or global state information. We show that a Web architecture that integrates DNS dispatching with a centralized redirection mechanism provides excellent load control and minimizes the risks of overloaded servers. Therefore, the TTL-constraint problem that affects DNS dispatching can be completely addressed. After a thorough and detailed study, of which only a small part can be reported here, we have discovered centralized and distributed schemes that not only achieve good results, but also guarantee stable performance. This result is very important, because stability is one of the most difficult attribute for any dispatching and re-routing algorithm that has to operate in the extremely variable Web environment. By comparing centralized and distributed control schemes, we have found that although in some cases a distributed redirection algorithm may also achieve slightly worse performance than some centralized alternatives, it has three major advantages that make its use preferable: it is easier to implement, imposes much lower computational and communication overheads, and allows the use of content-aware redirection policies that are gaining so much importance in the Web.

The rest of this paper is organized as follows. Section 2 describes the Web system architecture. Section 3 proposes a taxonomy for the design space of the redirection schemes based on control and state information. Sections 4 and 5 present various centralized and distributed algorithms for the activation of the redirection and the localization of the nodes, respectively. Section 6 addresses main implementation issues behind centralized and distributed redirection schemes. Section 7 focuses on the system, network, and workload models that are used to compare the performance of the redirection algorithms in Section 8. Section 9 concludes the paper with final remarks.

2 Web System Architecture

In this paper, we consider a distributed Web-server system consisting of a set of nodes that uses one Web site name to make the distributed nature of the service transparent to the outside world. Each *Web node* may consist of one or multiple server machines, as in a multiprocessor or in a Web cluster that houses servers at the same network location. The common characteristic of the components of this distributed architecture is that each Web node presents one system image to the outside. This means that, independently of the number of the servers that are part of a node, each Web node provides just one IP address visible to the client applications. This is the real IP address in the case of a Web node based on one server, or the virtual IP address of the Web switch that is in front of the Web cluster, in the case of Web nodes consisting of

multiple servers. A complete survey on mechanisms and dispatching algorithms related to Web clusters can be found in [6].

In similar Web architectures, the dispatching of the client requests among the servers typically occurs in more than one step. The first choice is for the Web node and it occurs in the lookup phase, when the client looks for an IP address corresponding to the Web site name. As we assume that each Web node has a complete replication of the Web site content and provides the same capacity, the A-DNS for that site can select any IP address of one of the Web nodes. The A-DNS can use sophisticated or simple stateless policies for distributing the client requests among the Web nodes. After the choice of a Web node by the A-DNS, if a Web node consists of more than one server, the second dispatching is left to the Web switch that can use a large set of algorithms for balancing the load among the servers of a cluster. This has been the goal of many previous researches [6], and it is not the focus of this paper. Due to the centralized control that the Web switch has on all the requests reaching a cluster, we can assume that the load inside a Web node is acceptably balanced.

The challenge that we face in this paper is how to balance the load among the Web nodes because, unlike the Web switch, the mechanism for choosing a node by the A-DNS is far from really controlling the address requests. Indeed, only if the local and intermediate name servers do not hold a valid mapping for the site name, the request reaches the A-DNS of the Web system. The address caching mechanisms and non-uniform distribution of clients among the DNS domains are the major issues that may defeat any dispatching policy carried out through the A-DNS alone [8]. In this paper, we investigate alternative solutions that integrate a DNS-based dispatching algorithm with some redirection mechanism managed in a centralized or distributed way. SWEB [1], DPR [3], and DC-Apache [19] are examples of Web system architectures based on double dispatching, where the A-DNS decision (typically, round-robin) is integrated with some request redirection mechanism. Our proposal differs because in this paper the focus is on exploring a large set of centralized and distributed control solutions that are combined with different types of state information.

3 Design Space for the Redirection Algorithms

In this section we analyze the large space of alternatives that exist when we intend to integrate some server redirection capability in the dispatching mechanisms of a distributed Web-server system. We are interested to analyze the alternatives that are compatible with existing Web standards and protocols. In particular, the A-DNS server and the Web nodes of the system are the only entities under the direct control of the Web site provider that can be modified for the purposes of request redirection. We have identified three main phases in the redirection process that can be implemented through centralized or distributed **control policies**: the *redirection activation policy* that determines the eligibility of a Web node as a redirector, the *request selection policy* that determines the offered load that is eligible for redirection, and the *node localization policy*

that determines the appropriate Web node(s) to which the first contacted node may redirect the selected load. In each of these phases, the decisions can be taken on the basis of *local*, *partial* or *global* **state information**, or *no information* at all.

Because of the large number of options for each group of factors, it is rather unfruitful and out of the scope of this paper to study the performance of all feasible combinations. Some qualitative considerations allow us to restrict the search space. We can anticipate that the two major principles that stand behind redirection are: centralized vs. distributed activation/localization. The request selection policy and the state information are used as a secondary space of choices to find the "best" policy in the centralized or distributed class. Hence, we consider briefly the request selection policy and the state information, and focus on the activation and localization policies in Sections 4 and 5.

3.1 Request Selection

The first group of factors we consider in this taxonomy is the *selection* of the offered load that is eligible for redirection. Some qualitative considerations and some research results [14] indicate that the minimum entity that is convenient to redirect is the request for an entire Web page, that is, the base file and all its embedded objects. The alternative of redirecting even individual objects consumes more server and network resources. Once the Web node has activated the redirection process, the selection policy determines which page requests actually have to be redirected.

Another choice regards the selection of the page requests that must be effectively redirected. The possibilities go from request-blind policies, that redirect all or a random subset of page requests reaching the node when the redirection mechanism is activated (namely, *redirect-all* and *redirect-partial*, respectively), to request-aware policies that consider some information about the client request. For example, the centralized selection at the A-DNS can use just some location information, such as the origin domain of the client request. A distributed selection mechanism implemented at the Web node can use more detailed information, such as the HTTP request content. This last alternative introduces the class of content-aware redirection algorithms. As the content-blind strategies tend to cause a large percentage of redirections with associated overheads, we also investigate in Section 8.4 how it is possible to limit the number of redirected requests by applying a finer redirection granularity.

3.2 State Information

Another important impact on redirection algorithm performance is the type of *state information* which is available to each decision maker. We have already discussed the information related to the request selection process. Here, we consider the information about the state conditions that are related to the redirection activation and node localization policies.

The first choice regards the load index that is representative of the state condition. We consider the entire range of possibilities, from the *stateless* policies, to the *server load*, and the *domain load rate* that is, the amount of load coming from a domain connected to the Web site during an observation interval (also called *domain popularity*) [8]. The second alternative regards the space of dissemination of the state information [16]. Specifically, each decision maker can own a complete view of the state information of the Web system (*global*), just the information about the node itself (*local*), or can acquire some information regarding a subset of nodes in the system (*partial*). The third choice is for the *information update* strategy [16], which can be demand-driven (typically related to partial information) or periodic (typically related to global information).

3.3 Space of Alternatives and Notation

We consider the activation policy as the main axis of classification for the redirection algorithms. The activation can be centralized or distributed. We also assume that in the *centralized* case the decision-maker component is based on the A-DNS. Specifically, it can be either located inside the A-DNS or even on a distinct entity that operates jointly with the A-DNS. It seems convenient to take centralized decisions on the basis of *global* state information and preliminary experiments confirm that choice. Hence, we denote the centralized activation schemes through **CA/g** as they always use global information.

On the other hand, when each Web node takes its own redirection decisions about activation (*distributed* schemes), we consider the entire spectrum of alternatives for the information. Therefore, we denote the distributed activation schemes through **DA/x**, where $x \in \{l, p, g\}$, depending on their use of local, partial or global state information, respectively. Here, we exclude the case of an algorithm based of no state information, because a random activation simply does not work.

Centralized and distributed schemes exist also for the node localization. They are denoted by **CL/g** and **DL**/x, where x stands for 0, p, g (0 means no state information). Here, we include the case of no information (for example, random or round-robin node localization), and exclude the case of local information as localization is inherently an external operation.

A *fully centralized* scheme that is, a centralized activation and localization scheme, with its global view on the system conditions aims to load balancing with a limited effort to reduce the number of redirections. Here, we basically consider an "intelligent" DNS dispatching scheme, where redirection is simply used as a tool to enforce the A-DNS decisions that without redirection would be bypassed by the address caching mechanisms. On the contrary, in the *fully distributed* activation and localization scheme, DNS dispatching can be kept as simple as possible, and redirection has the role of "intelligent" dispatching mechanism. Hybrid schemes exist as well. A centralized activation may be combined with a distributed localization (DL/x) policy, and a distributed activation may be combined with a centralized localization (CL/g) policy.

For example, in this paper a centralized activation algorithm coupled with a distributed localization policy using no state information will be denoted by **CA/g-DL/0**.

4 Redirection Activation

The *redirection activation* policy determines the eligibility of a Web node as a redirector. In both the centralized and distributed instances, the decision is taken on the basis of state load information that is periodically evaluated and, when necessary, disseminated. Once the decision on activation has been taken, the redirection executor is always the Web server when the Web node consists of one machine or it can also be the Web switch when the Web node is a cluster with multiple servers. A summary of the algorithms that will be considered for evaluation is in Section 8.1.

4.1 Centralized Activation Algorithms

In the centralized scheme, we assume that the activation decision is periodically taken by a process working in strict cooperation with the A-DNS. The centralized decision-maker broadcasts its decisions to the Web nodes that carry out the redirections. There are various choices for the centralized activation algorithms (CA/g). The first idea is to coordinate the DNS-dispatching action with the localization process executed by the Web nodes, namely a **CA/g-CL/g** scheme. In this way, the limited control of A-DNS dispatching is enforced through a redirection mechanism that puts all decisions on the A-DNS and considers the Web nodes as executors. Hence, centralized localization can be provided by a CA/g-CL/g policy that uses the so called *Domain Assignment Table* (DAT) for A-DNS dispatching and redirection. This mapping table specifies, for each domain that is a source of requests to the Web site, the Web node/s that has/ve to serve it. The basic information to build DAT is the domain popularity as defined in Section 3.2. We have also tried to use server load information to build DAT, but all experimental results were quite unsatisfactory. The reason is that server load information becomes obsolete quickly in a DNS-based distributed Web system subject to heavy-tailed workload, because of address caching effects on past assignments and high variability of client loads. This aspect makes the problem considered in this paper different from most literature on load balancing in distributed systems, where an up-to-date server load index usually provides a good indication of the future load condition [18].

Assigning each domain to one Web node does not work because the Web nodes receiving the most popular domains tend to be overloaded and this causes a large percentage of redirections. (As an example, in Section 8 we present the results for the DNS-DAT_1 policy that basically works as the classical off-line Longest Processing Time algorithm, which is used to minimize the capacity of a fixed number of bins). A much better alternative is to assign a domain to multiple nodes. This algorithm builds DAT by associating a bin to each Web node. The bin capacity is by default set to the maximum capacity that the Web node

can sustain without experiencing performance degradation. This capacity is a static *throughput* information that can be measured in terms of served requests or transferred bytes per second. At every update of DAT, the A-DNS estimates the load offered by each connected domain and calculates the total offered load. If the total node capacity is greater than the total offered load, the capacity of each bin is set to the default value, otherwise it is increased so to (virtually) handle all the offered load. As a third step, the domains are sorted in a decreasing order according to their popularity. Then, for each ordered domain (starting from the most popular), the A-DNS selects the node with the maximum residual capacity. If the chosen node has a sufficient residual capacity, the domain is entirely assigned to it, and the node capacity is decremented by a quantity equal to the domain load. Otherwise, the domain is assigned to the chosen node and its residual capacity is saturated, while the residual domain load is assigned to another node and so on until the domain load has been entirely assigned.

This algorithm also defines an additional table (called the *Weight Table*), in which for each domain the A-DNS specifies the percentage of load that is assigned to the nodes over which the domain is split up. It is worth to observe that the assignment of a domain to multiple nodes is really effective only in combination with some redirection mechanisms that can overcome address caching mechanisms. The algorithm chosen as the first example of the CA/g-CL/g schemes uses the DAT for activation, selection, and localization decisions. The A-DNS periodically broadcasts the Weight Table and DAT to the Web nodes. If a domain in DAT is split up over multiple nodes, the Web node determines randomly the destination to which it redirects the request by using the weights specified in the Weight Table.

We further consider a node load-based centralized activation scheme that can be combined with a centralized or a distributed localization algorithm. A representative example is the following. The server load is gathered periodically by the A-DNS that estimates the average load, and identifies as redirectors the nodes whose load level is in a certain range above the average. Then, the A-DNS broadcasts the result of its decision to the nodes in a list, marking the nodes that need to activate the redirection process for the next interval. The node localization decision can be centralized at the A-DNS, as in the **CA/g-CL/g(load)** algorithm, or delegated to the Web nodes, as in the **CA/g-DL/x(load)** algorithm. We will further discuss these policies in Section 5.

4.2 Distributed Activation Algorithms

The centralized algorithms that use plain DAT information for redirection decisions cause large percentage of redirections. Any new DAT is built by the A-DNS without taking into account the previous assignment. The lack of assignment persistence in DAT can cause many redirections, especially if the most popular domains are assigned to different Web nodes at each update interval. The issue of generating assignments that are consistent from one phase to the next has been addressed in [15]. We investigate here an alternative

approach that aims to activate redirections only when necessary that is, only on the nodes that are highly loaded. We can expect that avoiding the long service time at an overloaded node would compensate the redirection overheads.

In the distributed activation scheme (DA/x), we consider that each Web node decides by its own when the redirection needs to be activated through a *threshold-based* mechanism. Threshold-based load balancing policies are popular in distributed computer systems [10, 22]. They have been shown to be useful especially when jobs are independent and consist of single threads of control, which is a common feature for Web requests. In the considered schemes, any Web node periodically checks its own current load. When the load exceeds the selected threshold, the node enters in a redirection state that ends when the load returns below the threshold.

We consider two distributed activation algorithms that use different state information. In **DA/g**, the activation decision is based on load information of all nodes. This global information may be acquired by the nodes through an all-to-all exchange or by using a gather/broadcast mechanism at one node or at the A-DNS. Each Web node, more frequently than the update interval of the global information, evaluates its own load and activates redirection when its load level is in a certain range above the average load. The **DA/l** algorithm considered here is simple but effective, as demonstrated by the experimental results. Each Web node activates redirection only when the load on its server/s exceeds a given threshold, without any comparison with the load on other nodes.

5 Node Localization

The second factor of classification refers to the *localization* of appropriate Web node(s) to which the first contacted node may redirect the selected client request. The node localization decision process can be *centralized* (CL/g) or *distributed* (DL/x). A summary of the algorithms that will be considered for evaluation is in Section 8.1.

5.1 Centralized Localization Algorithms

A centralized node locator can identify one destination node for redirection or a set of nodes from which the redirecting node can select the destination. The state information is always global. The first algorithm chosen as example of the CL/g schemes uses for localization decisions the DAT and the Weight Table, which are periodically broadcasted by the A-DNS to the Web nodes. It has been described in Section 4.1.

Another example is the **CL/g(load)** algorithm, in which the A-DNS decides about node localization by sending to the Web nodes a list of available nodes to whom redirecting requests, where the list is built on the basis of the node load information already used for the activation decision. Specifically, only nodes that do not need to activate the redirection process are marked as available. Then, each redirecting node selects

the target server in a round-robin way on the list.

5.2 Distributed Localization Algorithms

In the distributed localization scheme, each Web node decides on the basis of global, partial or no state information to which node it is convenient to redirect the selected request. We consider three policies that are representative examples of the diverse types of state information available to the decision maker: the first policy is stateless (DL/0), the second uses a load information on a subset of the Web nodes (DL/p), while the third uses a global load information (DL/g).

When a request is found eligible for redirection, **DL/0** selects the destination in a round-robin way. This policy does not require any information exchange among the nodes.

DL/p exploits the use of a partial load information. The redirecting server selects randomly a set of K nodes (where $K = 1, \ldots, N-1$) and determines the node with the lightest load in this set. The redirecting server forwards the request only if the selected node has a load lower than its own. Previous works have shown that even the case of $K = 1$ permits to achieve an acceptable load sharing [10]. Our not reported experiments confirm this result. We consider a distributed periodic dissemination mechanism where each Web node measures its load and periodically broadcasts it to the other nodes. We acknowledge that there could exist other mechanisms for information exchange, but the evaluation of the alternatives is outside the scope of this paper.

DL/g requires a periodic all-to-all load information exchange. The global information is used by the redirecting node to select the Web nodes that have a load lower than its own. The target node is then picked randomly from this set. To avoid the herd effect that makes the system really unstable [9], we do not consider the strategy of sending each selected request to the node with the apparent lowest load.

6 Implementation Issues

We find useful to describe the implementation issues related to a Web-server system that applies the centralized and distributed redirection schemes. For the compatibility with the existing Web standards and protocols, any proposed modification entails only the A-DNS and the Web nodes which are under the control of the content provider.

Many redirection algorithms use some state information on Web nodes and/or client domains. The server load can be measured by using different indexes, such as the number of active connections on the server(s), or the utilization of the server(s) resources. In this paper, for each Web server comprising a Web node, we consider the maximum of the utilization between its CPU and disk. We use only the last reported node information, while interesting strategies to interpret stale load information can be found in [9].

For most centralized algorithms, the A-DNS needs to estimate the domain popularity in terms of load rate coming from each domain. This evaluation requires some cooperation of the A-DNS with the Web nodes that can track and collect the load offered to the distributed Web system through the server access logfiles. We found that the most accurate (and least expensive) way to assess the domain load rate is at the hit level. Another interesting problem related to domain popularity estimation is that the A-DNS and the Web nodes see different information related to a domain. Hence, it is necessary to group into domains the client IP addresses seen by the Web nodes, and to match the local DNS server with the corresponding domain. The issue of *client clustering* [17] has recently gained popularity due to the spread of Content Delivery Networks based on DNS dispatching (e.g., heuristics based on address masks, Autonomous System numbers, domain names [21], or more accurate and expensive techniques based on routing information [17]). Although the method based on the address mask does not provide an accurate clustering of clients into domains [17], it can be applied to A-DNS algorithms because their main concern is to identify the most popular domains and some inaccuracy is well tolerated.

We conclude this section with some brief comments about the implementation. In Figures 1 and 2 we outline the main software components needed to implement the proposed distributed Web systems when the activation and localization decisions are fully centralized at the A-DNS or fully distributed at the Web nodes, respectively. For simplicity of representation, these figures assume that a Web node consists of one server machine.

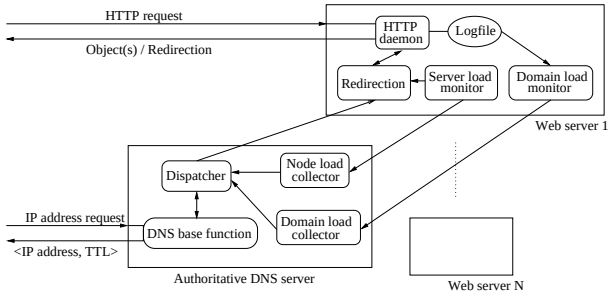

Figure 1: Architecture of the Web-server system in the case of fully centralized algorithms.

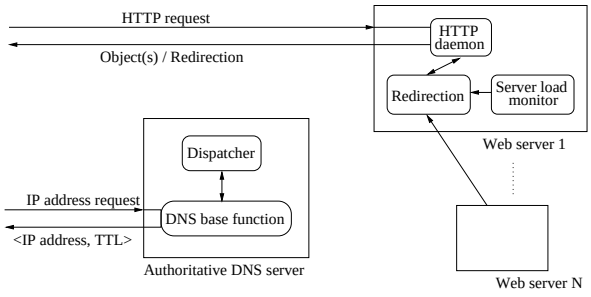

Figure 2: Architecture of the Web-server system in the case of fully distributed algorithms.

In the centralized scheme, the A-DNS software includes a dispatcher, a node load collector, and a domain load collector. The dispatcher assigns each address request to one of the Web nodes based on some algorithm. It also includes the activation and localization decision functions to perform the centralized algorithms. The node load collector tracks the load of the Web nodes, while the domain load collector gathers the domain load information from each Web node and estimates the domain popularity. Also shown is the set of components in the Web node. Besides the HTTP daemon server, the centralized algorithms may require a redirection component, a server load monitor, and a domain load monitor. The server load monitor

tracks the server load and sends this information to the node load collector located on the A-DNS. The domain load monitor estimates the load received by the server from each domain and periodically provides the information to the domain load collector in the A-DNS. Finally, the redirection component determines if a request has to be redirected and to which node, using the information received by the A-DNS dispatcher.

In the fully distributed approach shown in Figure 2, the A-DNS performs only the first-level dispatching among the Web nodes, without communicating with the software components located on the Web nodes that manage the redirection process autonomously. Similarly to the centralized scheme, the A-DNS dispatcher assigns requests to the Web nodes through the address mapping mechanism. In the Web node, the distributed algorithms require a server load monitor and a redirection component. The monitor tracks the utilization of the server resources and sends this information to the redirection component of its server and, when necessary, to that of another Web node. The redirection component implements the request selection and the node localization policies.

There are several mechanisms to re-route a request, such as the *triangulation* at the TCP/IP layer [3], *HTTP redirection* and *URL rewriting* at the application layer. The redirection mechanism provided by the HTTP protocol allows a Web server to respond to a client request with a 301 or 302 status code in the response header. These codes instruct the client to resubmit its request to another node [11]. URL rewriting integrated with a multiple-level DNS routing technique is also used by some Content Delivery Networks [15, 19]. There is not a mechanism that is clearly better than the others, as shown by the trade-off analysis in [6]. In this paper we refer to HTTP redirection, but this choice does not affect our main conclusions. There are two methods to avoid ping-pong effects that may occur when an already re-routed request is further selected for redirection. When a request is redirected for the first time, the redirecting server can set a cookie or insert a notification of the occurred redirection into the location field of the response header.

7 Simulation Model

In this section, we describe the simulation model that we use to compare the performance of the redirection algorithms. We first detail the system and workload model that include all characteristics of Web client/server interaction; then, we describe a simplified network model.

We consider 4 main geographical regions that are located in different world areas. The Web nodes and the A-DNS are located in the same region, while various *client domains* are spread among all the regions. Each Web node may consist of one or multiple servers, but each node provides the same capacity. We model all resources of a server machine, such as CPU, main memory, hard disk, and network interface. The HTTP server is modeled as the Apache 1.3.

We consider an open system model, where each new client generates an address request to the DNS that

consists of a hierarchy of local, intermediate, and authoritative name servers. The DNS model considers all mechanisms related to address caching and TTL. Once received an IP address for the Web site, the client establishes a TCP connection with the indicated Web node, which can correspond to a Web server or to the Web switch of a cluster of servers. In this latter instance, the Web switch selects a server and forwards the client request to it. The time to serve a request includes all the delays at the Web server, such as parsing time, service time for the base HTML file and all embedded objects, or redirection time if the page request must be redirected. These times have been validated with a real distributed Web system, where request dispatching is based on A-DNS and HTTP redirection.

The workload model used to drive the simulator incorporates recent results on Web characterization [2, 4]. The high variability and self-similar nature of Web access load is modeled through heavy-tailed distributions. Table 1 summarizes the probability mass function (PMF) and the parameter values we used in our workload model. More details can be found in [7].

Category	*Distribution*	*PMF*	*Parameters*
Session inter-arrival time [12]	Exponential	$\lambda e^{-\lambda x}$	$\lambda = 0.05$
Page requests per session [2]	Inverse Gaussian	$\sqrt{\frac{\lambda}{2\pi x^3}} e^{\frac{-\lambda(x-\mu)^2}{2\mu^2 x}}$	$\mu = 3.86, \lambda = 9.46$
Objects per page [2, 4]	Pareto	$\alpha k^\alpha x^{-\alpha-1}$	$\alpha = 1.245, k = 3$
HTTP request size	Lognormal	$\frac{1}{x\sqrt{2\pi\sigma^2}} e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}}$	$\mu = 5.929, \sigma = 0.321$
HTML object size [2, 4]	Lognormal Pareto	$\frac{1}{x\sqrt{2\pi\sigma^2}} e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}}$ $\alpha k^\alpha x^{-\alpha-1}$	$\mu = 7.63, \sigma = 1.001$ $\alpha = 1, k = 10240$
Embedded object size [2, 4]	Lognormal	$\frac{1}{x\sqrt{2\pi\sigma^2}} e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}}$	$\mu = 8.215, \sigma = 1.46$
User think time [4]	Pareto	$\alpha k^\alpha x^{-\alpha-1}$	$\alpha = 1.4, k = 1$

Table 1: Workload model.

In the simulation experiments, each client is assigned to one domain through a pure Zipf distribution, corresponding to a highly skewed function [8]. The client domains, ordered from the most to the least popular, are statically assigned to the Internet regions in a round-robin way. The A-DNS uses a TTL value set to 300 seconds. The distributed Web system consists of 8 Web nodes with homogeneous capacity. The local load information is checked every 8 seconds, while the update interval of global load information is set to 30 seconds. The sensitivity analysis in Section 8.5 shows that the main conclusions of this paper are not affected by the choice of system parameters such as TTL, client distribution among domains, load intensity, and number of Web nodes.

The network model aims at providing a controllable testbed where the transmission of data between a Web node and a client has some cost. This choice is motivated by the goal of the simulation model that does not aim to predict the actual response times, but to compare the impact of algorithms and redirection on the performance through a testbed that is the fairest possible for all algorithms. For this reason, we do not consider real Internet topologies, network hierarchies, and narrow network bandwidth in the last mile [12]

that could have an impact on performance stronger than that related to the redirection algorithms.

In the model of client-server interaction, we refer to the HTTP/1.1 protocol that uses persistent connections and pipelining. The client, after having retrieved the base HTML file, makes multiple requests for embedded objects on the same connection without waiting for a response between each request. From [13], we have that the time to transmit n objects belonging to the same page between region i and j is given by $T^n_{i,j} = 3rtt_{ij} + \sum_{l=1}^{n}(S_{req_l}/ab_{ij} + S_{res_l}/ab_{ji})$ where rtt_{ij} and ab_{ij} are the *round-trip time* and the *available bandwidth* between region i and j, respectively, and S_{req_l} and S_{res_l} are the size of the client request and server response for each object l, respectively. If the first contacted node in region k redirects the page request, the terms $2rtt_{ik} + S_{req_1}/ab_{ik} + S_{res_1}/ab_{ki}$ must be added to include the network overhead caused by HTTP redirection.

We briefly analyze the parameters in the above equation, while a detailed description of the network model can be found in [7]. The message sizes follow the distributions shown in Table 1, while the round-trip time value is chosen randomly in the interval corresponding to the two end-point regions as shown in Table 2. The available bandwidth ab_{ij} models the communication delays between two Internet regions. We assume that these delays are due to a static factor (*basic bandwidth*) and a dynamic factor (*traffic*). The basic bandwidth between two regions is assumed to be deterministic and the corresponding values are shown in Table 2. (We also carried out some sensitivity analysis as a function of the basic bandwidth and observed that the main conclusions of this paper are not affected.) The traffic is modeled as a random parameter that reduces the basic bandwidth. Since the traffic depends on the number of clients in the regions traversed by a connection, we assume that the fraction of bandwidth available to a connection starting from a region and ending in another one is related to the popularity of the two end-point regions [7]. Finally, we do not model the network delay due to the address resolution that occurs at the beginning of each client session, because it has the same impact on the performance of all redirection algorithms.

	Basic bandwidth	*Round-trip time*
Region 1-1	1.35 Mbps	[30, 70] msec
Region 1-2	1.2 Mbps	[60, 100] msec
Region 1-3	0.9 Mbps	[120, 180] msec
Region 1-4	0.7 Mbps	[240, 300] msec

Table 2: Parameters of the network model.

8 Experimental Results

The goal of the experiments is twofold: to measure how effectively the redirection mechanisms can solve the A-DNS problems related to limited control on request dispatching and to compare the performance of fully centralized, fully distributed, and hybrid control schemes and their relation with the different types of

system state information. The crucial performance metric is the response time, as it directly correlates with the user perception of the quality of service. The page response time corresponds to the interval between the submission of a client request and the arrival at the client of all objects related to the page request. It includes the TCP connection time, all delays at the Web server, the network transmission time, and possible redirection overheads. We use as the main measures the *cumulative distribution* and the *90-percentile* of the page response time, because average values are not meaningful in Web systems subject to large and unpredictable traffic spikes. Another important performance metric is the *percentage of redirected requests* as a further goal of this study is to propose redirection algorithms that minimize this value. The simulator, based on the method of independent replication, has been implemented using the CSIM package [20]. The experiments involved a minimum of 200,000 client arrivals and each reported value is the result of ten or more simulation runs with different seeds for each random number generator.

8.1 Motivation for Redirection Algorithms

To motivate the need for redirection, we first focus on systems based on DNS dispatching only. Figure 3 compares the cumulative page response time of two real algorithms (DNS-RR and DNS-DAT_1) with that of two ideal solutions (LNS-RR and DNS-RR(TTL=0)). **DNS-RR** is a basic implementation of the round-robin scheme where the A-DNS returns the IP address of one Web node at each address request in a cyclic way. In DNS-DAT_1, A-DNS dispatching decisions are based on a DAT where each client domain is assigned to one Web node. As described in Section 4.1, mapping to multiple nodes does not work when the A-DNS is the only dispatching entity.

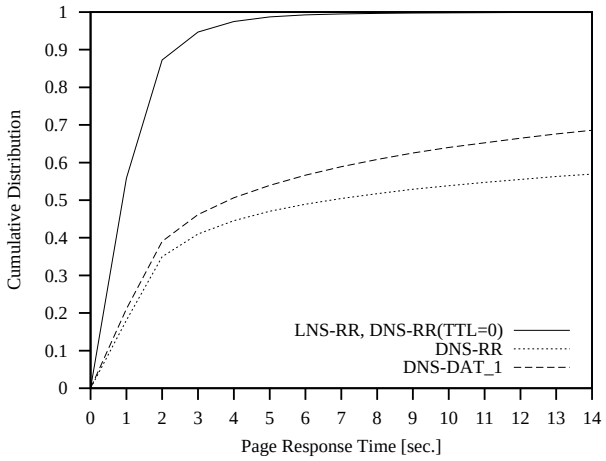

Figure 3: Cumulative page response time for DNS dispatching algorithms.

LNS-RR applies round-robin cycling at the local name servers. The difference between DNS-RR and LNS-RR should be clear. With DNS-RR, the A-DNS replies with one record that varies from request to

request, while with LNS-RR, the A-DNS returns multiple records containing the IP addresses of all Web nodes, and each name server performs round-robin on the list. LNS-RR represents an ideal policy because not all local name servers may implement it, e.g., if they have been configured to return the list in the same order or are not up-to-date. We also consider **DNS-RR(TTL=0)**, the DNS-RR algorithm in which the effects of address caching are avoided by setting the TTL value to 0 seconds. As pointed out in Section 1, this is another ideal solution. It is interesting to see that two different ideal solutions achieve an identical response time, that we consider as an upper bound for the proposed dispatching algorithms. The need for a redirection mechanism is clearly motivated by the performance gap existing between the real DNS-based dispatching algorithms and the ideal policies in Figure 3. Let us summarize the alternatives we consider for performance evaluation.

- For *centralized activation*: **CA/g-CL/g** uses the DAT and Weight Table for activation and localization. **CA/g-CL/g(load)** uses a global information about server load for both activation and localization. **CA/g-DL/x(load)** uses the same activation algorithm of the previous policy, while different types of information may be used for distributed localization.

- For *distributed activation*: **DA/g-CL/g** uses a threshold-based algorithm that considers global information about server load for activation, and the DAT and Weight Table for localization. **DA/g-DL/x** uses the same activation algorithm as the previous policy, but different types of information may be used for distributed localization. **DA/l-CL/g** uses a threshold-based algorithm that considers just the local server load for activation and the same localization algorithm as the DA/g-CL/g policy. **DA/l-DL/x** uses the same activation algorithm of the previous policy, but different types of information may be used for distributed localization.

8.2 Performance of Redirection Algorithms with Centralized Activation

Let us first consider the three algorithms with centralized activation: CA/g-CL/g, CA/g-CL/g(load), and CA/g-DL/x(load). Figure 4 shows that the performance of the fully centralized algorithm CA/g-CL/g is close to the curve of the ideal LNS-RR. This means that DNS-dispatching integrated with the DAT-based redirection mechanism is quite able to solve the TTL-constraint problem. Another important result is that the CA/g-CL/g performance is insensitive to the update frequency of the DAT table. This stability is of primary importance in reality, because larger update intervals cause lower computation and communication overheads, and reduce the number of redirected requests because of the higher persistence of the DAT assignments. In our experiments, an increment of the updating interval from 30 to 300 seconds reduces the percentage of redirections by 35% at the expense of the response time by a small increase of less than 10%.

It is important to observe that the DAT of this CA/g-CL/g scheme is built by using just domain load information, while in not reported results we found that building the DAT on the basis of server load in-

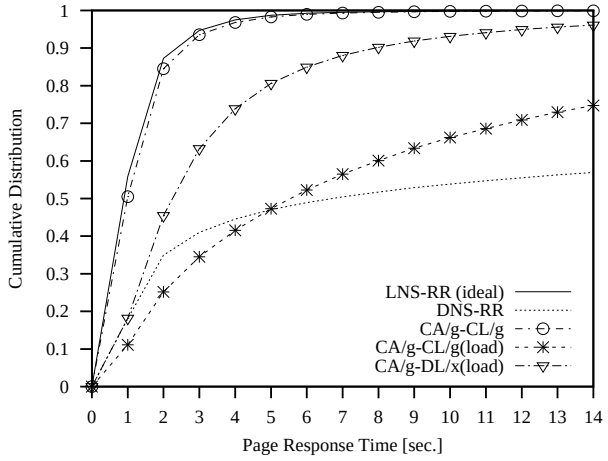

Figure 4: Response time of the algorithms with centralized activation.

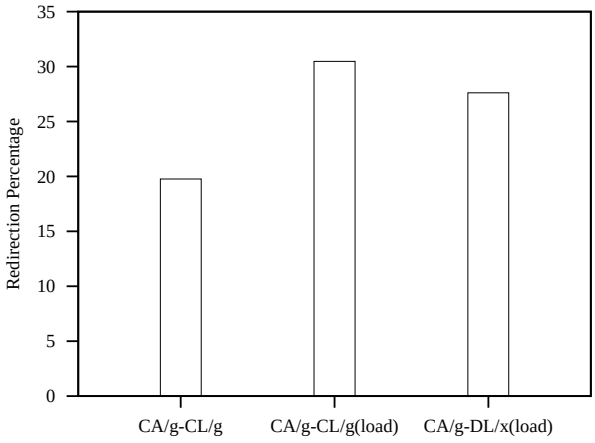

Figure 5: Redirection percentage of the algorithms with centralized activation.

formation yields to poor performance, even worse than that of DNS-RR. The unsatisfactory results of the CA/g-CL/g(load) and CA/g-DL/x(load) policies based on server load provide an indirect confirmation that it is insufficient to perform centralized activation just using server information. The reason is that server load is subject to high fluctuations, whereas A-DNS takes periodic decisions based upon information that tends to become stale soon: it may happen that a node must continue to redirect (receive) requests even if it becomes underutilized (overloaded) before the next A-DNS decision. Moreover, we found that the performance of the load-based activation algorithms degrades rapidly when the state information updating interval increases.

We cannot report all performance results about the node localization, but we outline main conclusions of our studies. The typical effect we observed is that a burst of redirected requests improves performance of redirecting nodes, but it tends to overload the receiving nodes, especially when this last set is limited. The consequence is that the redirection process is activated on the receiving nodes, and the Web system remains unstable for long periods, with noticeable impact on the user response time. Thus it is preferable to spread the load among the widest set of Web nodes rather than concentrating redirected load on a limited set of nodes. The type of state information for taking localization decisions is much less important. For example, a round-robin localization policy as in CA/g-DL/0 performs better than other DL/x schemes.

From the performance results shown in Figures 4 and 5, we can conclude that a centralized activation works better with a centralized localization, where both control algorithms use global information about domain load rather than server load. In the following, only this CA/g-CL/g scheme will be considered for comparison with distributed activation schemes.

8.3 Performance of Redirection Algorithms with Distributed Activation

When we consider schemes with distributed activation of the redirection process, we have several alternatives to explore: coupling of distributed activation with centralized and distributed localization policies, and also the use of various types of state information.

The initially unexpected result shown by Figures 6 and 7 is that hybrid and pure schemes have similar performance independently of the use of local or global information for distributed activation: the curves of DA/l-CL/g and DA/g-CL/g, and those of DA/l-DL/x and DA/g-DL/x overlap. The reason is that the value of the load threshold evaluated dynamically on the basis of global information does not differ notably from the value of the load threshold set statically on each node. All four algorithms achieve performance close to LNS-RR, although the hybrid schemes are able to limit the redirection percentage to less than 14%, while the fully distributed schemes have a double amount of redirections. We will address this issue in Section 8.4.

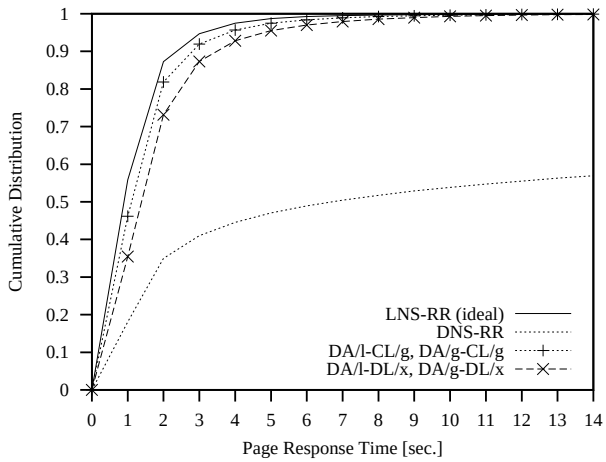

Figure 6: Response time of the algorithms with distributed activation.

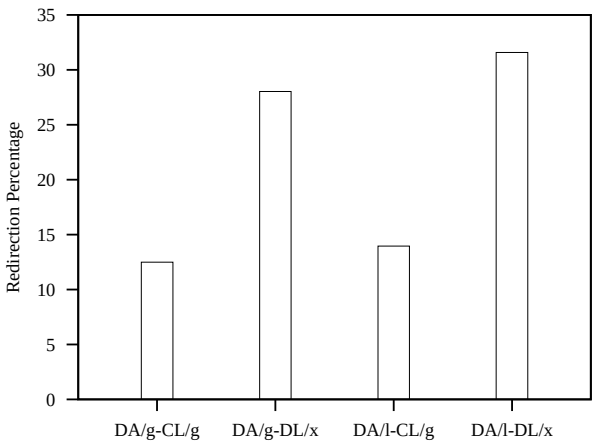

Figure 7: Redirection percentage of the algorithms with distributed activation.

As regards the node localization policy, the conclusion is similar to that observed for the centralized schemes: the state information is less important, and hence it seems convenient to use a stateless round-robin algorithm (DL/0) that does not require costly exchanges of state information.

8.4 Performance of Request Selection Policies

In the previous section we have observed that the performance of the fully distributed algorithms is penalized by a high percentage of redirected requests with respect to their hybrid counterparts. On the other hand, an algorithm with distributed selection has the potential advantage of using more detailed information than that available at a centralized entity, and using that information to limit redirection to the heaviest requests. We now exploit this potential by exploring selection policies that apply a redirection granularity finer than that

of the typical redirect-all, here denoted as **Select-all**. A content-blind alternative to reduce the percentage of redirections can select a random subset of requests that are going to be redirected. This strategy corresponds to the redirect-partial selection and is referred to as **Select-part**N, where N denotes the percentage of requests to be redirected (that is, part50 redirects 50% of the received requests). It selects the page to be redirected in a content-blind way, without taking into account the load that the requests impose on the node.

We also consider two content-aware selection policies that limit redirection to the requests imposing a larger load on the node. The first motivation is that Web workload is characterized by high variability and skewness [2, 4]. Hence, a very small fraction of the largest files determines a large fraction of the load on the Web node. Furthermore, users will be less bothered by redirects on a request that will be very long anyway. In particular, we propose **Select-size** redirecting only requests for Web pages larger than a certain size, and **Select-num** redirecting only those pages containing a large number of embedded objects. We use the average size of the base file and its embedded objects as the default size threshold to decide about redirection for Select-size, and the average number of embedded objects in a Web page as the default threshold for Select-num.

In the following experiments we focus on distributed activation algorithms based on local information (DA/l), however similar results have also been observed for DA/g policies. Figures 8 and 9 compare the cumulative distribution of the page response time and the redirection percentages for three content-blind (Select-all, Select-part25 and Select-part50) and two content-aware (Select-num and Select-size) selection policies.

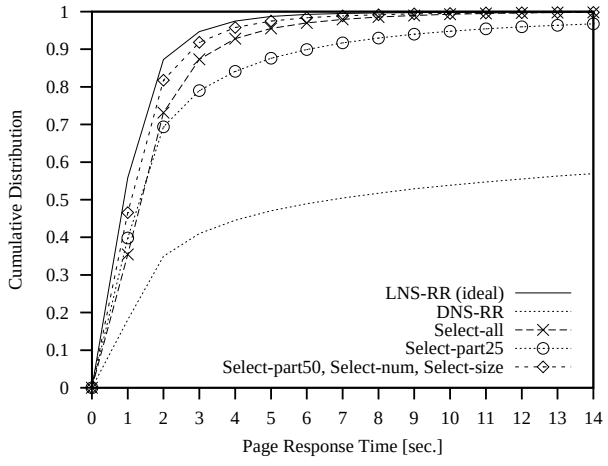

Figure 8: Response time of DA/l-DL/0 for various request selection policies.

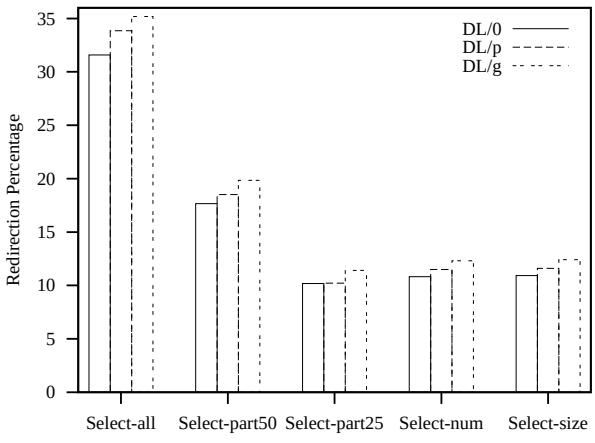

Figure 9: Redirection percentage of DA/l-DL/x for various request selection policies.

It is interesting to observe that redirecting only a subset of requests improves the performance over the Select-all strategy. This is even more appreciable when we consider the 90-percentile as performance metric: the response time decreases from 3.5 to 2.8 seconds for Select-all and Select-num, respectively. The

reason of the response time improvement is the substantial reduction of the redirection percentages achieved by the content-aware selection policies. We can see in Figure 9 that Select-all redirects more than 30% of the requests reaching the Web site, while the analogous result is close to about 10% for Select-num and Select-size. The content-aware policies are the best choice because they guarantee the best combination between response time and redirection percentage. Indeed, Select-part25 is affected by poor performance, while Select-part50 by high percentage of redirected requests. As the network model proposed here does not fully capture the complexity of the real Internet, the reduction of redirections achieved by Select-num and Select-size selection policies may reduce network latency time even more than that shown in Figure 8. This observation is also confirmed by the results reported in Figure 13.

8.5 Sensitivity Analysis

The previous sections were useful to explore the design space of the redirection algorithms and to find the fully centralized, fully distributed, and hybrid algorithms that are able to provide response times close to the ideal policy. We now test their performance for various system scenarios as a function of some critical system parameters. Specifically, we consider CA/g-CL/g for fully centralized algorithms, DA/g-DL/0, DA/l-DL/0, and DA/l-DL/0-Select (with the selection policy corresponding to Select-num) for fully distributed algorithms, and DA/g-CL/g for hybrid ones.

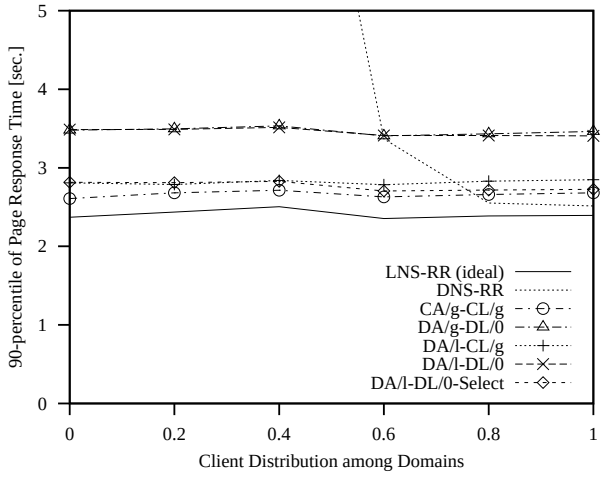

Figure 10: Sensitivity of response time to client distribution among the domains (Zipf parameter).

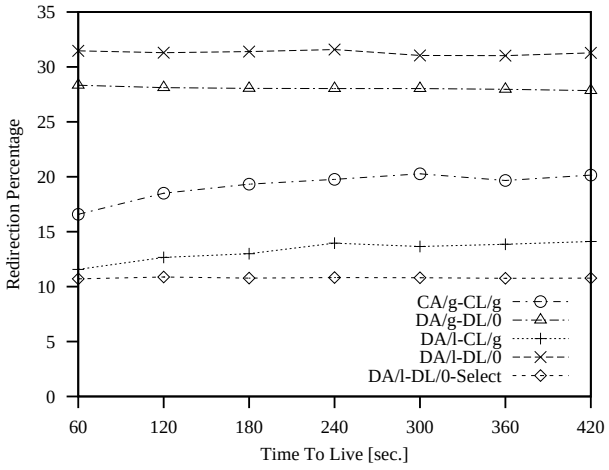

Figure 11: Sensitivity of redirection percentage to the TTL value for caching IP addresses.

Figure 10 compares the performance of the redirection algorithms in a system where the distribution of the clients among the domains varies from the pure Zipf ($\theta = 0$) to the uniform distribution ($\theta = 1$). The fully centralized algorithm always achieves better performance than the distributed and hybrid schemes, even if the results of DA/l-DL/0-Select are quite close. The robustness with respect to the client distribution is important because in the real Web environment the client scenarios tend to change frequently.

As expected, this figure also shows that the performance of DNS-RR improves as the client distribution among the domains tends to the uniform one.

Another interesting aspect is the sensitivity of the redirection algorithms to the TTL value returned by the A-DNS, because the TTL-constraint was one of the problems we aimed to solve. Although the dispatching control of the A-DNS decreases very rapidly as the TTL increases because address resolutions are cached for longer periods, Figure 11 shows that all redirection algorithms are surprisingly robust in terms of redirection percentages. We observed the same stability when we considered the page response time as the performance metric. This result demonstrates that any of these centralized, distributed, and hybrid redirection schemes is able to address the limited dispatching control of the A-DNS.

In Figure 12 we examine the sensitivity to the number of Web nodes in the system, when it changes from 4 to 56 (the default value corresponding to 8). For a fair comparison, the load offered to the Web system is kept proportional to the number of nodes that is, when the number of nodes doubles, the client inter-arrival rate doubles as well. We find that changing the number of nodes does not affect the performance ordering among the algorithms when we consider the response time as a performance metric. Hence, we focus on redirection percentages. These percentages for fully centralized and hybrid algorithms increase continuously as the number of Web nodes increases until 25-30, and then tend to stabilize. On the other hand, DA/l-DL/0-Select is almost insensitive to the number of Web nodes, always providing the lowest amount of load redirection.

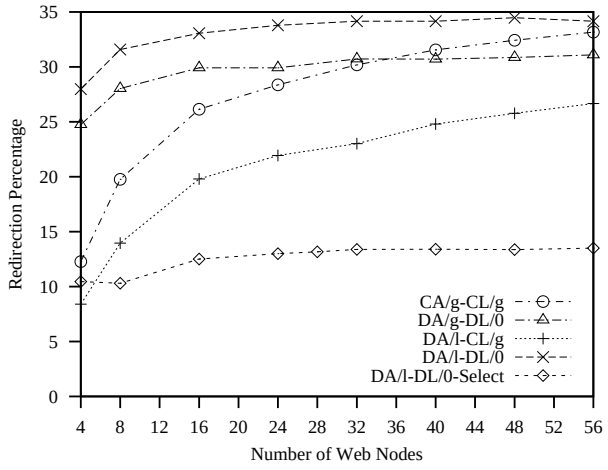

Figure 12: Sensitivity of redirection percentage to the number of Web nodes.

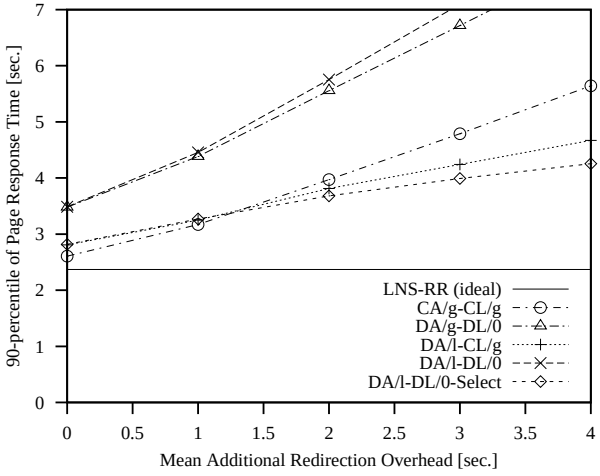

Figure 13: Sensitivity of response time to the redirection overhead.

As the network model cannot realistically represent Internet, the redirection cost may have an impact on response time higher than that shown by previous results. Hence, it is important to investigate performance of redirection schemes for higher costs of redirection, in addition to the already modeled double network round-trip time and server management cost of redirection (the default corresponding to 0 sec.) Figure 13

shows that the response time of the fully distributed algorithms DA/l-DL/0 and DA/g-DL/0 increases rapidly for higher redirection overheads due to their higher percentage of redirected requests. Although its increment is smoother, even the fully centralized scheme does not perform well when the cost of redirection increases. A more sophisticated request selection policy, such as DA/l-DL/0-Select, is able to limit the performance degradation, thus confirming the intuition that in the real Internet the reduction of redirections obtained by content-aware selection policies has a noticeable positive impact on performance. A similar observation is in order when we consider the hybrid DA/l-CL/g because it limits the number of redirections.

In the last set of experiments we evaluate the impact of different load scenarios on performance. Two different workloads are obtained by adding requests for dynamic objects, and by augmenting the probability of finding objects in the server disk cache that reduces the stress on disks. To model the generation of dynamic content, in each Web node we add a second tier of application and/or database servers. A dynamic request includes the overheads due to back-end server computation to generate the dynamic objects and is characterized by a Lognormal service time on back-end nodes with mean equal to 0.5 seconds. Figure 14 shows the performance of the redirection schemes for higher numbers of pages containing dynamic objects. We can conclude that the relative performance of the redirection algorithms does not change.

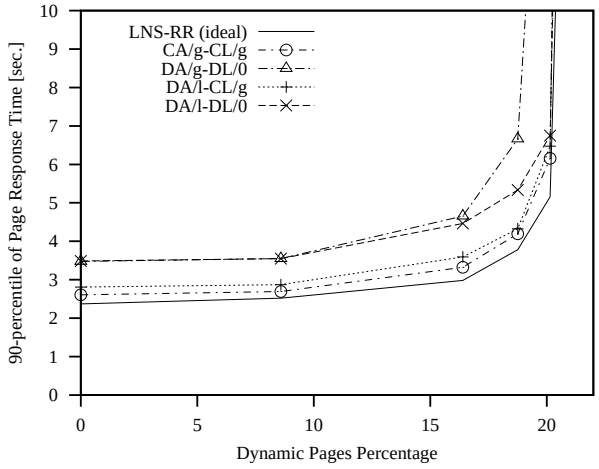

Figure 14: Sensitivity of response time to the percentage of dynamic requests.

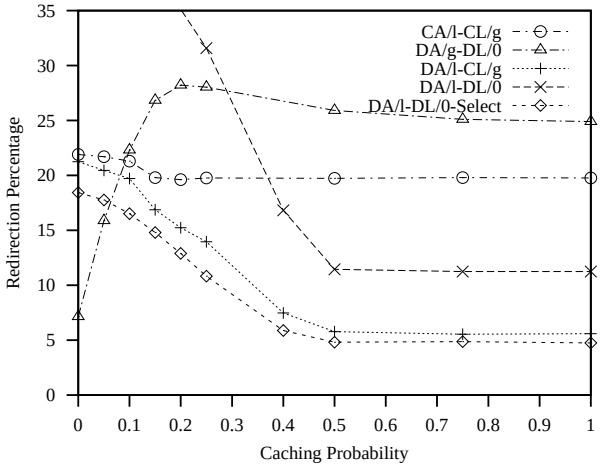

Figure 15: Sensitivity of redirection percentage to caching probability in disk cache.

We then vary the probability that the requested Web object is found in the server disk cache. A lower caching probability corresponds to a higher load on the disk resource, which represents the server bottleneck. As expected, a higher stress on the server disk degrades the performance in terms of response time. If we consider the redirection percentage shown in Figure 15, we can observe an interesting phenomenon. The fully centralized CA/g-CL/g algorithm is quite insensitive to the caching probability because the state information used by the A-DNS for the activation decision does not include the node load. On the other hand, for the algorithms with distributed activation based on local information (DA/l schemes), the redirec-

tion percentage diminishes as the caching probability increases. Indeed, when the node is less loaded, the redirection needs to be activated less frequently. On the contrary, when the distributed activation depends on a global information (curve DA/g-DL/0), the redirection percentage decreases with the caching probability. Indeed, when the caching probability is low causing the global load to be high, it happens less frequently that the local node load exceeds the global load threshold. It is worth to observe that in practice redirection does not hurt much disk cache replacement; indeed, as the object popularity is not uniformly distributed [2, 4], all servers tend to have the most popular objects in their disk caches.

Our analysis indicates some results common to centralized and distributed algorithms. The centralized activation achieves best results when coupled with a centralized localization, and the same is true for distributed activation and localization. Distributed activation and centralized localization is the only hybrid scheme that shows comparable performance to fully centralized and fully distributed schemes. For a fair comparison of centralized with distributed redirection schemes, we have to consider not only performance metrics but also implementation complexity and management overheads. Focusing on the response time only would let us conclude that the fully centralized algorithm is the most convenient choice because for any considered scenario it provides (slightly) lower response times than the fully distributed policy. The problems with centralized schemes are that they are not able to limit the percentage of redirected requests, their implementation may be hard because of the communication mechanisms and estimation of the domain popularity, and their execution is more expensive from the computation and communication point of view than any distributed scheme counterpart. The simulation model could not take into account all these complex factors, hence the results about response times are more favorable than reality to the centralized schemes. Hence, we can conclude that the fully distributed solutions are preferable, because they provide similar response times with a lower percentage of redirected requests, less overheads, and much lower system complexity.

9 Conclusions

We provide a taxonomy of the redirection schemes in distributed Web systems including centralized vs. distributed control algorithms for the activation of the mechanism, for the localization of the destination nodes, and for the selection of the requests to be redirected. These control policies can be based on local, partial or global state information. A thorough investigation has led to the proposal of centralized and distributed schemes that not only achieve good results, but also guarantee stable performance. The stability of performance under different system scenarios is crucial when considering real systems that operate in the extremely variable Web environment. Even the DNS round-robin dispatching that performs very poorly under skewed request load, when combined with a distributed redirection mechanism carried out by the Web nodes, can achieve good and stable performance with percentages of redirections lower than 15%.

By comparing fully centralized, fully distributed and hybrid control schemes, we have found that in some cases a distributed redirection algorithm may achieve slightly worse performance than the best centralized alternatives, but it has three major advantages that make its use preferable: it is easier to implement, imposes much lower computational and communication overheads, and allows the use of *content-aware* redirection policies that are gaining so much importance in the Web.

Acknowledgements

The authors would like to thank the referees for their valuable comments and suggestions which were helpful in preparing the final version of the paper. The first two authors acknowledge the support of MIUR-Cofin 2001 in the framework of the project "High-quality Web systems".

References

[1] D. Andresen, T. Yang, and O.H. Ibarra, "Toward a Scalable Distributed WWW Server on Workstation Clusters," *J. Parallel and Distributed Computing*, vol. 42, no. 2, pp. 91–100, Apr. 1997.

[2] M.F. Arlitt and T. Jin, "A Workload Characterization Study of the 1998 World Cup Web Site", *IEEE Network*, vol. 14, no. 3, pp. 30–37, May/June 2000.

[3] L. Aversa and A. Bestavros, "Load Balancing a Cluster of Web Servers using Distributed Packet Rewriting," *Proc. 19th IEEE Int'l Performance, Computing, and Communications Conf. (IPCCC)*, pp. 24–29, Feb. 2000.

[4] P. Barford and M.E. Crovella, "A Performance Evaluation of Hyper Text Transfer Protocols," *Proc. ACM Sigmetrics 1999*, pp. 188–197, May 1999.

[5] P. Barford and M.E. Crovella, "Critical Path Analysis of TCP Transactions," *IEEE/ACM Trans. Networking*, vol. 9, no. 3, pp. 238–248, June 2001.

[6] V. Cardellini, E. Casalicchio, M. Colajanni, and P.S. Yu, "The State of the Art in Locally Distributed Web-Server Systems," *ACM Computing Surveys*, vol. 34, no. 2, pp. 263–311, June 2002.

[7] V. Cardellini, M. Colajanni, and P.S. Yu, "Geographic Load Balancing for Scalable Distributed Web Systems," *Proc. 8th IEEE Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 20–27, Aug. 2000.

[8] M. Colajanni and P.S. Yu, "A Performance Study of Robust Load Sharing Strategies for Distributed Heterogeneous Web Server Systems," *IEEE Trans. Knowledge and Data Engineering*, vol. 14, no. 2, pp. 398–414, Mar./Apr. 2002.

[9] M. Dahlin, "Interpreting Stale Load Information," *IEEE Trans. Parallel and Distributed Systems*, vol. 11, no. 10, pp. 1033–1047, Oct. 2000.

[10] D.L. Eager, E.D. Lazowska, and J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed Systems," *IEEE Trans. Software Engineering*, vol. 12, no. 5, pp. 662–675, May 1986.

[11] R.T. Fielding, J. Gettys, J.C. Mogul, H.F. Frystyk, L. Masinter, P.J. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, June 1999.

[12] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," *IEEE/ACM Trans. Networking*, vol. 9, no. 4, pp. 392–403, Aug. 2001.

[13] J. Heidemann, K. Obraczka, and J. Touch, “Modeling the Performance of HTTP over Several Transport Protocols,” *IEEE/ACM Trans. Networking*, vol. 5, no. 5, pp. 616–630, Oct. 1997.

[14] J. Kangasharju, K.W. Ross, and J.W. Roberts, “Performance Evaluation of Redirection Schemes in Content Distribution Networks,” *Computer Commun.*, vol. 24, no. 1-2, pp. 207–214, Feb. 2001.

[15] D. Karger, A. Sherman, A. Berkheimer, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, “Web Caching with Consistent Hashing,” *Computer Networks*, vol. 31, no. 11-16, pp. 1203–1213, Feb. 1999.

[16] O. Kremier and J. Kramer, “Methodical Analysis of Adaptive Load Sharing Algorithms,” *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 6, pp. 747–760, Nov. 1992.

[17] B. Krishnamurthy and J. Wang, “On Network-Aware Clustering of Web Clients,” *Proc. ACM Sigcomm 2000*, pp. 97–110, Aug. 2000.

[18] T. Kunz, “The Influence of Different Workload Descriptions on a Heuristic Load Balancing Scheme,” *IEEE Trans. Software Engineering*, vol. 17, no. 7, pp. 725–730, July 1991.

[19] Q. Li and B. Moon, “Distributed Cooperative Apache Web Server,” *Proc. 10th Int’l World Wide Web Conf.*, pp. 555–564, May 2001.

[20] Mesquite Software, “CSIM18 User Guide,” http://www.mesquite.com/.

[21] A. Shaikh, R. Tewari, and M. Agrawal, “On the Effectiveness of DNS-Based Server Selection,” *Proc. IEEE Infocom 2001*, pp. 1801–1810, Apr. 2001.

[22] N. G. Shivaratri, P. Krueger, and M. Singhal, “Load Distributing for Locally Distributed Systems,” *IEEE Computer*, vol. 25, no. 12, pp. 33–44, Dec. 1992.