Distributed Web Serving Solutions: 
A Comparative Study

Katia Obraczka, Fabio Silva, Savas Guven and Sagar Mane

Abstract
This paper explores different solutions for distributed web serving (dws). Some large Internet sites are already using some of these solutions today when provisioning their servers to accommodate an increasing number of requests. We anticipate that as the Internet grows, the number of web sites requiring some kind of dws solution will increase. In this paper, we explore routing-based solutions that use recently developed hardware. We compare these schemes with DNS-based solutions and identify their strengths and weaknesses. Finally, we propose directions for future work, including ways of evaluating the performance of the different dws approaches.

1 Introduction

Today users shop on the Web, buying from books and music to stocks. People use the Web to find references on a specific discipline, saving a trip to the local library. Many web sites offer the latest news, weather conditions and stock quotes at no charge. Web sites offering auctions attract millions of users, as the numerous sites providing live audio and video streams.

The unprecedented growth of the Internet user population accompanied by the explosion of new Web applications have caused a tremendous performance impact on the communication and service providers' infrastructure. In order to keep their services operational, Internet Service Providers (ISPs) and content providers had to increase capacity.

Provisioning has been widely adopted as a solution to deal with the increasing number of requests. Internet Content Providers (ICP's) have been adding servers to their farms and using some form of load balancing to split the incoming requests among servers. A side-effect of load balancing is faster responses to users.

More recently, specialized switches have been developed to solve the server load balancing problem. They include Alteon's Layer 4 switch (L4
switch) and Arrowpoint’s Layer 5 switch (L5 switch), which operate at the transport and application layers, respectively. The L4 switches work by intercepting packets based on their source and destination IP addresses as well as source and destination port numbers. They can divert incoming requests to particular servers according to a pre-defined load balancing scheme. Besides performing L4-style load balancing, the L5 switches can also load balance based on the content of a request (providing finer-granularity filters).

In this paper we investigate distributed web serving (dws) solutions for provisioning a web site. We introduce the concept of Global IP related to routing-based provisioning schemes. We also explore name-service based dws solutions that use DNS.

The paper is organized as follows. In Section 2, we study routing-based solutions for dws. Section 3 explores DNS-based approaches and proposes a hybrid scheme using both Global IP and DNS. We compare these solutions in Section 4 and identify directions for future work in Section 5.

2 Global IP and Routing-Based Solutions

Routing-based approaches to dws rely on the network routing fabric to route the client’s request to the “closest” server. Proximity is measured in hop counts which is the metric used by routing protocols.

For the solutions presented in this section, we assume that we have several servers (or server farm), responsible for the same web content. These server may be geographically distributed. In this context, we define Global IP as follows:

Besides their own IP address, servers in a server farm share a common Global IP address. Client request to a Global IP address will reach one of these servers.

Service providers advertise their Global IP address through DNS. Clients accessing these services have their requests mapped to the corresponding Global IP address and then served by the closest server farm. Devices such as Alteon’s L4 switches are used as a front-end to a server farm deciding which local server will answer the request. L4 switches make this decision based on their knowledge of the current individual load. Besides server load information, content-aware devices such as Arrowpoint’s L5 switches select a server based on the content requested. For instance, www.cnn.com may allocate different servers for the different sections of their web site (e.g., www.cnn.com/sports, www.cnn.com/weather, etc).
We investigate routing-based dws by exploring two different scenarios. In the first one, all servers lie in a single Autonomous System (AS). Note that this does not assume that all servers are in the same geographic area. In the second scenario, servers are distributed across multiple ASes.

2.1 Example Scenarios

We start configuring a number of servers (sharing the same Global IP address) on different locations in a AS, as illustrated in figure 1. Clients C rely on an internal routing protocol (in our experiments we used RIP) to get their requests to the “closest” server S (that in this case presents the lowest hop count). Requests coming from outside (clients located in other ASes) reach a server close to the border router BR being used to route the requests.

![Figure 1: Servers in the same AS](image)

It’s already clear from this simple scenario that routing changes affecting the hop count between a client and the servers can change the server answering this client’s requests. While allowing a client to reach another server in the event of a server failure, this behavior is not always desirable.

Changing the server responsible for a session without its knowledge leads to several side-effects. The problems range from TCP resets (when a server keeps persistent connections to avoid the overhead of starting a new connection on every request) to complete losses of state (the shopping cart problem, where the web server keeps all the state for the client’s session).

We move to a general case, configuring web servers in several ASes and
Figure 2: Multiple ASes have our servers

using the same Global IP. Figure 2 pictures this case, where servers in $AS_A$ and $AS_B$ receive requests from clients located in $AS_E$ and $AS_C$ respectively.

Analogous to the previous case, this scheme relies on an external routing protocol (in our experiments we used BGP-4), which uses the AS count to decide between routes. Since multiple border routers advertise a particular route, clients reach the “closest” AS offering the specified content. It is intuitive that strategic server deployment in different geographic regions would greatly improve response time, while also performing load balancing since all regional traffic stays local.

We propose placing servers at the edges of the network, directly connected to a BGP-4 router. Besides isolating external from internal traffic, this scheme allows servers to directly contact the border router. Acting as internal BGP peers, these servers can announce the Global IP to the border routers.

This scheme also suffers from the stability problems discussed previously. For example, when a route is created from $AS_E$ to $AS_B$, client requests formerly going from $AS_E$ to $AS_A$ now go to $AS_B$.

Routing table overhead is another side-effect. Because the Global IP address would not be part of the addresses assigned to ASes hosting our servers, aggregation schemes, like CIDR, could not be used. Therefore, an extra routing entry would be required in these ASes’ border routers in order to accommodate the Global IP address. If this approach is widely used, this can generate higher demands of routing table space.
2.2 Adding Stability

We saw that stability problems may result as multiple ASes advertise the same Global IP route. Because we rely on the network fabric to route client requests, a slight route change makes these requests go to a different destination. To avoid this problem, we propose a new solution which is illustrated in Figure 3.

![Figure 3: Change Global IP to a Local IP](image)

In this new scheme, when the first TCP SYN packet arrives with the Global IP address, we send a HTTP 302 Redirect, forcing the client to redirect its request to a local server (local IP address). While adding an extra RTT to the overall response time, this approach is immune to routing changes. Clients only use the Global IP address when first contacting the server, reaching the “closest” server. After that, clients place requests to a regular IP address and are bound to a particular server. Another positive aspect of this scheme is that whenever servers have to direct a client to another server (e.g., for a dynamic object not present on the current server), contacting a peer server is simpler since each server has its own IP address.

3 DNS Based Approach

The Domain Name System is a distributed database that handles the mapping between host names and Internet addresses. Lists of host names and IP addresses are classified into domains and distributed throughout the Internet in a hierarchy of authority, making DNS extremely scalable.
Clients are configured to contact local name servers whenever a name resolution is required. Local name servers in turn interact with the DNS hierarchy getting the address of the Authoritative Name Server for the requested domain. The local name server then asks the host’s IP address, relaying the answer back to the client.

### 3.1 Pure DNS Solution

An ICP configures its Authoritative Name Server with a list of IP addresses of servers responsible for a web site. The list mapped to a web site is passed to clients querying the DNS server. Currently, DNS client-side implementations use only the first IP address in this list. Therefore, by changing the IP addresses’ position in the list we can select the server answering the request.

This simple solution works in a round-robin fashion and does not require changing any installed software. However this approach suffers from several limitations for not taking into account information about servers (e.g. availability and load). Client proximity is also not considered, making this approach not suited for heterogeneous environments.
3.2 DNS with Load Information

We improve the DNS approach by including server load information into the list of servers for a web site. Figure 4 illustrates this solution. Authoritative Name Servers run the Load Exchange Protocol (LEP) to exchange server load information. Based on the information received, name servers come up with a list of servers sorted by load. Thus clients receive the IP address of the server experiencing the lowest load.

3.2.1 Design

Web servers run a LEP Server that is responsible for:

- Collect load information from each local web server
- Calculate the site load
- Send this local load to other LEP peers
- Receive load information from other LEP peers
- Maintain a sorted list of LEP peers and their load information

Web servers run an agent process, sending the machine’s load periodically to a local LEP peer. By keeping timeouts associated with each server, the local LEP peer knows about machines’ crashes.

Similarly, local LEP peers periodically send their overall load metric to all other peers. If load information is not received from a LEP peer before a timeout that peer is considered to be down and the corresponding entry is removed from the list.

3.2.2 Implementation

Figure 5 presents the LEP peer’s functional structure. It is a layered implementation in which each layer is responsible for interfacing with a different entity (web servers, clients or LEP peers).

**Interface with web servers:** Every web server has an agent process running that periodically measures the machine’s load (e.g. by calling `getloadavg()`). The agent sends this information to a pre-configured LEP peer, which keeps a list of of each web server’s load of its site. The LEP peer then computes the overall load for its web servers that is called site load.

The Agent and the LEP-agent interface operate as follows:
Figure 5: **LEP** Peer Implementation

**Agent:**

```java
for(;;){
  Measure server load;
  Send load information to LEP peer;
  sleep(interval);
}
```

**LEP:**

```java
for(;;){
  Wait(message or timeout);
  Check for expired server;
  if (update received){
    Get domain name and server load;
    Update load information;
    Set timeout;
  }
  if (load changed){
    Calculate new overall load;
    Update list;
  }
```

8
Send new local load information to peers;
}
}

**Interface with other LEP peers:** All LEP peers periodically exchange load information. Whenever the local site load changes or after a specified timeout, LEP peers send load messages to all other peers. Since the load information received is kept in a soft-state scheme, constant updates are sent to and received from other peers, otherwise a peer is assumed do be down and has its entry removed from the list.

This LEP-LEP interface works as follows:

```c
while(1){
    if (message received from peer){
        Parse message and get peer information;
        Update load list;
        Set timeout;
    }
    if (peer’s timer expired){
        remove peer’s name from Global list;
    }
    if (change in my load or my_timeout expired){
        send site load to LEP peers;
        reset my_timeout;
    }
}
```

**Interface with clients:** LEP peers receive DNS queries from clients. The client interface layer parses these queries and if the LEP peer has the queried host configured for load balancing, it returns the least loaded server’s IP address to the client. When clients request an unconfigured host name queries are forwarded to a DNS server.

The LEP-clients interface operates as follows:

```c
for (;;){
    if (DNS query received from client){
        if (the domain queried is in our list of domains){
            Get IP address of the first server in Global list;
            Generate DNS reply packet;
            Send reply to client;
```

9
} else {
    send query to Domain Name server;
}
}

3.3 Hybrid DNS - Global IP Approach

Although the previous scheme takes into account server load, client proximity is still not considered. Since the design presented in the previous section is distributed, all LEP peers come up with the same information using the distributed protocol. Thus all LEP peers are able to answer DNS queries themselves.

Keeping the design from the previous approach, an ICP configures all LEP peers with the same Global IP address. As a consequence clients’ DNS queries reach the closest LEP. Instead of just ordering all information received from all other LEP peers, each peer places a much higher weight on its own load since the client is “closer” to it and therefore it should answer the client’s DNS query with a local server even if the local farm is a little more loaded than other peer’s servers.

4 Discussion

This section discusses the strengths and weaknesses of the routing- and DNS-based solutions to DWS.

Stability As we mentioned in Section 2, routing-based schemes make use of the network routing fabric to direct a client’s request to the closest server. Consequently, routing changes result in instability. This means that new requests from the same client may end up being served by different servers. Stateful applications not specifically designed to share state information among the various servers will not work properly in the face of route instability.

The scheme presented in Section 2.2 also uses routing to find the closest server, yet is immune to route instability. It works by binding a client’s request to the closest server the first time the request is issued. As soon as the server, say X, is contacted, the Global IP is changed into the X’s actual IP address (using a HTTP redirect). Instead of using the service’s Global
IP, subsequent requests from the same client will use X’s real IP address. Binding the service’s Global IP to a server’s IP address adds an extra RTT to the time it takes to service a request.

DNS-based solutions are not affected by routing changes: the client gets the server’s IP address even before the HTTP request starts.

**Scalability** Global IP allows a service provider to announce its service using a single address. As discussed in Section 2, this is particularly useful when service providers distribute their servers across several ASes. However, AS routing aggregation policies may not be able to aggregate “foreign” Global IP addresses, resulting in an extra route entry in their border routers. It also results in extra overhead in propagating routing updates. In the worse case, each Global IP address will result in an extra entry in the ASes’ border gateways. Note that these extra entries will make their way to other border gateways depending on the routing protocol being used.

A partial solution to this problem is for service providers to use a single Global IP to represent all its services (instead of a per-service Global IP). DNS maps the multiple service names into the same Global IP address and the provider uses a L5 Switch to inspect the requested URL and select the appropriate server.

Again, DNS based solutions do not present scalability problems since DNS maps are partitioned and distributed across authoritative DNS servers. Authoritative DNS servers will include an entry for each server farm.

**Performance** Pure DNS solutions do not take into account server load. Even with load information, pure DNS approaches still do not consider client proximity and are probably just effective when servers are located in an homogeneous network, with comparable delays (e.g., across the US). Further enhancements are possible to the DNS solutions by using a more sophisticated LEP (e.g., using a scheme that would try to measure client proximity before answering the DNS query).

Because of its ability to take a client request to its closest server, routing-based solutions can provide lower delays. The hybrid DNS solution is supposed to achieve the same performance as the stable routing solution, the only difference being that the former uses a Global IP address to reach the closest DNS server and uses this information when answering the DNS request while the later uses Global IP to locate the closest server then switching to a stable Local IP address.
5 Conclusions and Future Work

This paper describes several techniques for implementing distributed web serving solutions. We could verify the functionality of new hardware developed to solve the server load balancing problem. During our experiments we could identify several limitations in some of these techniques and for some of them we pointed solutions or work-arounds.

We have also proposed a new scheme for DNS solutions, using a distributed Load Exchanging Protocol, that increases the DNS server’s knowledge about the web servers’ load, improving its functionality.

We end the paper selecting some metrics that allow us to compare these different approaches and discuss their strengths and weaknesses.

While these techniques have been tested and compared to some extent, we expect to continue studying them to better understand their limitations and quantify the selected metrics. Since our current experiments and tests reflect only results over a limited set of conditions, we still have to test these approaches on more adverse scenarios, comparing response time and overhead over multiple ASes. Experiments conducted on different geographic locations would allow us to experience a wide range of delays and routing changes (compared to our induced routing changes and minimal delay variations).

Another possible direction for continuing our experiments is to extend some of the results from [6], studying client proximity, server response time and network stability. While the study finds a strong correlation among RTT, packet loss and AS hop count, it does not consider the case when using Global IP solutions and therefore not exploring the stability and scalability issues.

References


