Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Computer Networks 43 (2003) 695–711

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

An overview of DNS-based server selections in content distribution networks

Jianping Pan [a], Y. Thomas Hou [b,*], Bo Li [c]

[a] *Fujitsu Laboratories of America, 595 Lawrence Expressway, Sunnyvale, CA 94085, USA*
[b] *The Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA*
[c] *Hong Kong University of Science and Technology, Kowloon, Hong Kong*

Received 6 August 2002; received in revised form 1 May 2003; accepted 6 May 2003

Responsible Editor: G. Pacifici

Abstract

With the explosive growth of the Internet, many new online services and applications are emerging. Some popular applications impose new challenges to the traditional Internet architecture and protocols. To alleviate the scalability burden in delivering popular Internet services to a large number of users, *Web caching* and *content distribution* technologies have been proposed, developed and deployed. Both approaches are designed to bring Web content closer to users and to improve their perceived quality of online experience. This paper surveys content distribution networks (CDNs), and in particular, their domain name system (DNS)-based server selection schemes. To bridge the gap between underlying principles and current practices, we choose a commercial content delivery provider, Akamai, as our focal case study. We first unveil Akamai's content delivery network, as well as its site and object delivery technologies. We then examine the DNS-based server selection schemes and their variants in detail. Moreover, we offer some performance insights and discussions on their built-in strengths and weaknesses that are also applicable to other CDN providers.
© 2003 Elsevier B.V. All rights reserved.

Keywords: Internet; Scalability; Content distribution networks (CDN); Server selection; Domain name system (DNS)

1. Background and overview

Historically, the design of any practical system is guided (and sometimes limited) by its intended applications. The current Internet architecture and protocols were designed decades ago to support applications that were envisioned at that time. Although the design still has certain flexibilities to accommodate future extensions, it is far from adequate when it comes to supporting emerging services and applications nowadays. To keep the Internet in line with its growth, new add-on patches and ad hoc schemes have been proposed in recent years.

This section provides some background on the new challenges that Internet-based service deliveries have been facing. First, we review the

* Corresponding author. Tel.: +1-540-231-2950; fax: +1-540-231-8292.
E-mail address: thou@vt.edu (Y.T. Hou).

1389-1286/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S1389-1286(03)00293-7

evolution of service delivery over the Internet. We then give an overview of two possible approaches to cope with explosive Web growth: *Web caching* and *content distribution*. We also present some related efforts in these areas.

1.1. Preliminaries

Remote access and file transfer were the two major applications when the Internet was invented. Remote access, e.g., *telnet*, allows users to easily use geographically-distributed computational resources. File transfer, e.g., *ftp*, enables users to conveniently exchange data files among different computers. There are other applications combining the features of these two examples, within which the conceptual *client–server* model is followed, i.e., a client initiates the information exchange in a *request–reply* manner with a designated server (as shown in Fig. 1). In this context, reliability and consistency are the two foremost concerns, since no computer can tolerate a corrupted program. These requirements are translated into the flow, error, and congestion control mechanisms in TCP that were designed to ensure reliable data transfer within these applications.

Architecture-wise, survivability and simplicity are the two most important guidelines in such a schema. To achieve these goals, an *end-to-end* paradigm was adopted for the Internet. Network nodes such as computers and routers are connected through communication links, and are identified by globally-unique IP addresses. Packets carrying source and destination IP addresses are routed in a *hop-by-hop* manner by routers looking up the next-hop IP address for a referenced destination in their routing tables. Host *connectivity* is considered the central building block. Before any communications between client and server can take place, a client must obtain the IP address of a designated server.

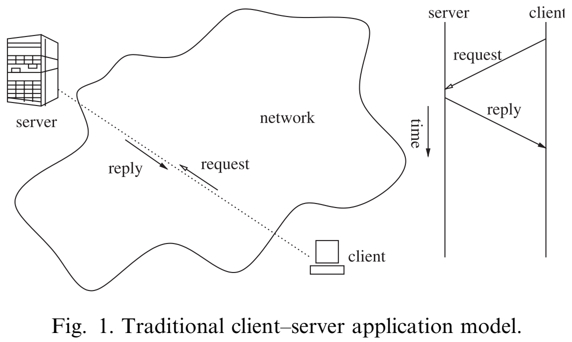

Fig. 1. Traditional client–server application model.

An IP address is a binary number with a fixed length, e.g., 32-bit under IPv4. Such a numeric identifier is very difficult for human beings to handle, even with the assistance of dotted quad notations such as 198.41.0.4. Instead, users prefer host and domain names in ASCII strings, e.g., *a.root-servers.net*. Regardless of their presentation, the coherence between addresses and their names has to be maintained somewhere. Since a centralized *hosts.txt* table did not scale well, a hierarchical domain name system (DNS) was designed and deployed [2,11] over the Internet. In this DNS hierarchy, authoritative domain servers are responsible for their local DNS databases containing *A*, *PTR* and other resource records. Their *NS* records are kept by the well-known *root*, *generic top-level domain* (gTLD), and *country code top-level domain* (ccTLD) DNS servers.

To minimize DNS-related resolution delay and traffic overhead, caching has been heavily employed in DNS. A secondary DNS server can mirror the whole database of a primary DNS server. Client resolvers and local DNS servers cache resolved records for a while for future queries. If the *A* or *PTR* record of a foreign domain is unavailable, and if its *NS* record is not cached locally, a query is first sent to the root DNS server. The query process repeats recursively through the DNS hierarchy. To avoid connection establishment and release overhead in TCP, DNS queries adopt the unreliable and connectionless UDP as their transport protocol. Normally, the *request–reply* transaction finishes in one packet for each direction. Without the flow, error, and congestion control mechanisms in their transport layer, resolvers and DNS servers have to rely on their own application-level timers and counters to detect and recover failed or unanswered queries.

The World-Wide Web follows a similar client–server model in its design, so that the IP and DNS schema worked reasonably well initially. When a user clicks or types in a hyperlink in a web browser, e.g., *http://yahoo.com*, the host and domain name, i.e., *yahoo.com*, is extracted and resolved into an IP address, e.g., 66.218.71.112. The browser then initiates a TCP connection to port 80, as specified by *http:*, on the web server identified by the resolved IP address. On connection establishment, the browser issues an HTTP *GET* command with the object identifier, and the server returns the requested object accordingly. Finally, the browser displays the received object, along with other embedded objects in the same HTML document. The performance measure for this process can be characterized by the so-called *click-to-display* delay, which is a metric for user-perceived quality of Web experience.

1.2. Problem and approaches

With the explosive Web growth in recent years, *application scalability* has become the most critical challenge to the Internet. As shown in Fig. 2, a popular web server (e.g., news portal *cnn.com* or event site *saltlake2002.com*) typically has to serve a huge number of simultaneous clients in a short period. This phenomenon is known as "flash crowd". Such a request surge can easily overload any single web server and its access links. Studies show that Web traffic and user requests follow the *Zipf*-like power law [5]: only a very few highly popular objects are responsible for most user accesses and network traffic. This extremely unbalanced client–server paradigm creates a serious scalability burden at the server side that severely degrades user-perceived Web experience.

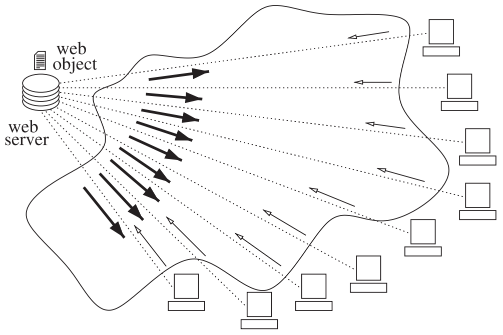

Fig. 2. The scalability issue with popular web sites.

In order to scale up service deliveries over the Internet, two new approaches, namely, *Web caching* and *content distribution*, have been developed. Here, we briefly describe and compare these two approaches.

Web caching—A group of neighbor users can set up a proxy server close to them with the assumption of time and space locality in their requests (see Fig. 3). Such a *user-oriented* approach is known as *Web caching*. All nearby user requests in the same administrative domain are first sent to this common proxy server. If the requested object was accessed recently and cached locally, and if a cached copy is still valid, the proxy returns the cached object directly. Otherwise, the proxy generates another request to the origin server for the requested object (or to an upstream proxy server in a hierarchical caching system). Web caching alleviates the scalability burden by decreasing the *amount* and *rate* of user requests sent to origin servers. This paradigm also has other advantages,

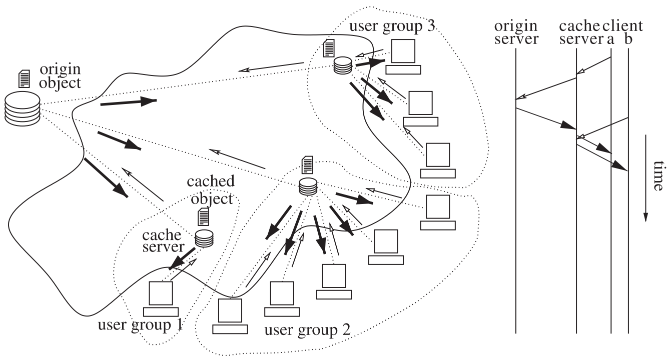

Fig. 3. Web caching.

e.g., easy management in a corporate network. Users can explicitly configure their browsers with the name or address and port number of the proxy server that they are aware of. Alternatively, a proxy server or edge router can transparently intercept certain outgoing requests. Proxy servers are also deployed near origin servers and known as *reverse proxy*. For a caching system, proxy servers can be organized according to a flat or hierarchical structure. *Sibling servers* may further exchange cache control messages and serve requests from neighbor domains. Obviously, there is certain overhead associated with Web caching, e.g., cache validation and request regeneration when cached objects are invalid. If weak consistency is adopted, users run the risk of obtaining an outdated object. Cache maintenance and validation are vital to system performance and have attracted many research efforts [3] in recent years.

Content distribution—The second approach is more *provider-oriented*: origin servers are partially or fully replicated *on-demand* or a priori; they are placed locally or remotely over the Internet (see Fig. 4). In this scenario, *server selection*, or which replica a client should contact, is vital to user-perceived performance. For local replicas, a DNS *round robin*-based scheme returns a rotated list of replica IP addresses in response to each DNS query. User requests then pick the first IP address from the list, so that they are *pseudo*-evenly distributed to all listed replicas. However, a DNS round robin scheme cannot balance requests efficiently. In addition, such a scheme requires that each replica be individually IP addressable.

A more challenging task in content distribution is to make a proper selection among geographically-distributed replicas. The selection can be made by users explicitly. For example, when being prompted by origin servers, users can choose a mirror site that is geographically close to them, or a mirror that matches their language and country preferences. It is obvious that a location-based selection does not guarantee the best mirror among all available servers, and a language or country-specific selection only has very coarse granularity.

Another way to achieve *global balancing* is to exploit the existing DNS infrastructure. Akamai [1] has adopted this approach in its content delivery network (CDN) (also known as content distribution network), which consists of thousands of Akamai servers located in more than one thousand access networks worldwide. By looking up the source address of a DNS query (typically a user's local DNS server), Akamai DNS servers choose the *best* Akamai server, according to its own criteria, to deliver the content to requesting users. Other CDN providers, e.g., Inktomi and Digital Island, adopt similar dynamic DNS mapping schemes for server selection. Some even directly replace the host identifiers in URLs by dynamic IP addresses *on-the-fly*. We focus on the underlying DNS-based server selection schemes, so that our study of Akamai is also applicable to other CDN providers.

Both Web caching and content distribution are designed to alleviate the application scalability problem in the traditional client–server model. We

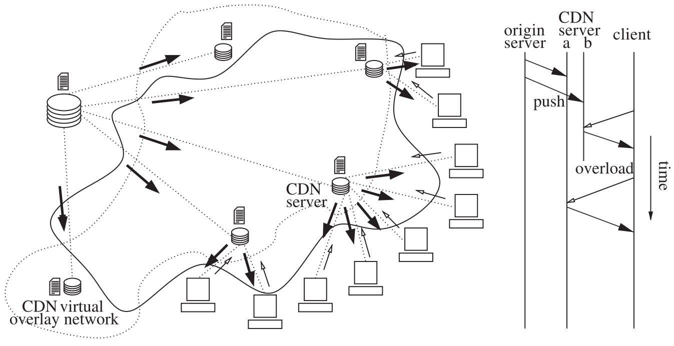

Fig. 4. Content distribution networks.

can envision a scenario in which these two approaches can work together harmonically. For example, user requests are first aggregated at a local proxy server, and then the regenerated request will be directed to a CDN server if the requested object is not locally available at the proxy. However, these are two different approaches coming from different perspectives. Web caching is mainly from the user's perspective. Thus, individual user groups or organizations control and maintain their proxy servers independently (shown in Fig. 3). CDN, on the contrary, is conceived from the service provider's perspective. Subsequently, CDN servers are coordinated, maintained and controlled by commercial CDN providers as a virtual overlay network, as shown in Fig. 4. CDN is *expected* to be more flexible and robust than regular proxy servers. For instance, depending on the current network and server condition, CDN can, if needed, seamlessly and transparently choose another replica. But it is very unlikely, or at most only available in very coarse granularity, that users can choose their proxy servers on a dynamic per-request basis. In this paper, we will focus on the approach of content distribution.

1.3. Other related efforts

Some network equipment vendors also offer their own content networking solutions. For example, Cisco has both local and distributed (global) load balancing products. Cisco's local director (LD) [7] is deployed close to cache servers. It distributes user requests to cache servers with some lower layer mechanisms, e.g., dynamic mapping of IP and MAC addresses, or IP-to-IP address translation. Round-robin or static weighted round-robin schemes are often used to select the next available server in a coarse granularity. Cisco's multi-node load balancing (MNLB) [8] looks similar to LD, but it is more sophisticated and scalable to a large number of cache servers. Server load information is collected by MNLB agents located in each server and processed at MNLB. A forwarding agent holds user requests until a decision is made by MNLB according to the current server condition. Since LD and MNLB may become a single point of failure, redundant LD and MNLB are deployed to monitor the "heartbeat" messages from the default LD and MNLB, and take over should the default LD or MNLB fail.

Cisco's distributed director (DD) [9] achieves the goal of global balancing. DD can operate in two modes. In the DNS caching mode, it behaves somewhat like Akamai DNS servers, i.e., DD dynamically maps a generic host name into the IP address for a cache server that is reported to be the *best* by a DD agent close to that server. In the HTTP redirect mode, as Fig. 5 shows, DD emulates a web server. It accepts the initial HTTP request, selects the best server according to the IP address of the actual client (not the user's local

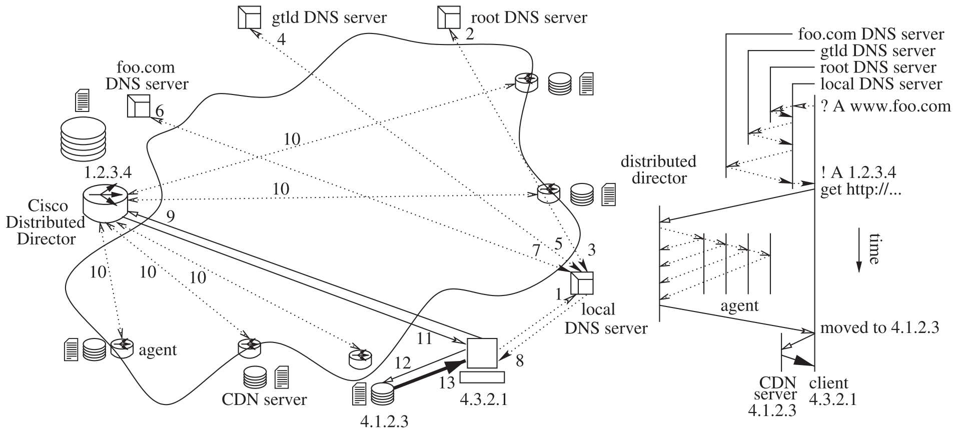

Fig. 5. HTTP redirect in Cisco's distributed director.

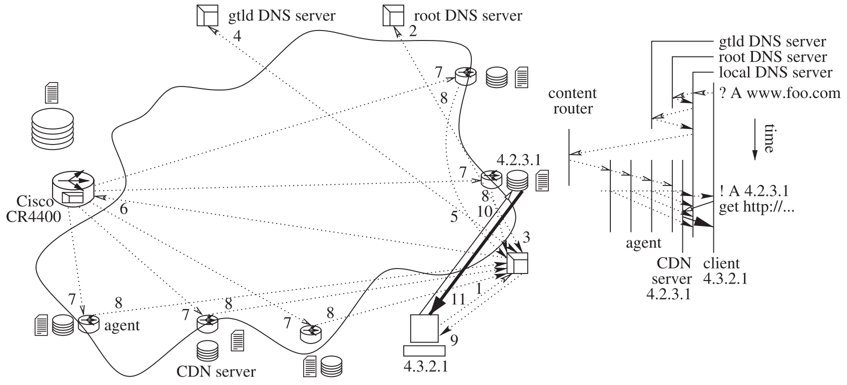

Fig. 6. DNS contention in Cisco CR-4400.

DNS server), and replies a "*302.Moved temporarily*" message with the host name or IP address of the assigned server. Note that in this case, at least two consecutive TCP connections need to be established: the first one to DD, and the second one to the chosen server.

In addition, Cisco has several content router (CR) products, most of which follow similar approaches to those used by DD. One CR deserving further mention is CR-4400 [10], which employs a different technique known as *DNS contention*. Under this scheme (see Fig. 6), after a CR receives a DNS query, it duplicates and floods the query to a group of agents located near cache servers. All of these agents reply a DNS *A* record *directly* to local DNS servers, since most short DNS queries are transported by the connectionless UDP protocol. The reply message that first arrives at the local DNS server wins, since that particular agent and the cache server are considered to be the nearest ones in terms of network delay. However, to ensure that all agents reply at the same time, DNS query messages have to be time-stamped, and all agents must be time-synchronized. This requirement introduces considerable overhead when there are a large number of agents geographically located in a wide area. Moreover, the nearest server in terms of network delay may not guarantee the user to receive the shortest click-to-display delay.

For the rest of this paper, we will focus on the DNS-based server selection schemes employed in Akamai's CDN. In Section 2, we give an overview of Akamai's CDN platform and its site and object delivery technologies. Section 3 examines in depth Akamai's DNS-based server selection schemes; Section 4 discusses their performance strengths and weaknesses. Section 5 concludes this paper.

2. Akamais content delivery network

To bridge the gap between underlying principles and current practices, we have chosen a commercial CDN provider, Akamai [1], as our focal case study. Despite different implementations, many CDN providers adopt very similar DNS-based server selection schemes, all of which are designed to be transparent to end users. Our analysis and discussions will also be applicable to these CDNs.

The materials presented here are based on publicly-available information disclosed by Akamai, as well as a set of user-oriented external experiments carried out by us. Since Akamai keeps evolving its platform and technologies, some aspects and figures listed here may change over time. For example, from two sets of measurements done by us in 2002 and 2003, we found changes in terms of IP addresses and the number of DNS servers. However, we believe that as long as the DNS infrastructure remains intact, the essence of DNS-based server selections by a commercial CDN provider will remain the same in upcoming years.

2.1. *Akamai's CDN platform*

As of May 2002, Akamai's CDN network consists of approximately 13,000 Akamai servers

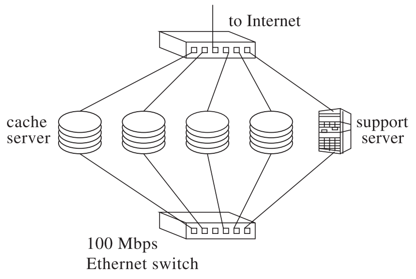

Fig. 7. Akamai server group.

deployed in more than 1000 access networks in 65 countries. An Akamai server is an Intel CPU-based off-shelf and rack-mounted computer that runs a modified Linux operating system. It is typically installed in Internet Data Centers, and replicates the designated origin server in an *a priori* or *on demand* manner with other optimizations. Akamai servers also collect network and server statistics, and report to Akamai's Network Operation and Control Center (NOCC). The collected information is processed and then used for server selections by dynamically mapping a generic host name into a particular IP address.

Akamai servers are usually configured into a group (or a *cluster*, in Akamai's term). As shown in Fig. 7, these servers are connected to two 100 Mbps Ethernet switches: one for their Internet connection, and the other dedicated for internal communications (mainly content replication and load measurement) [6]. Most servers run a highly customized *Squid* [17] caching software which handles user requests and delivers requested web objects. If a server fails, one of the remaining servers (a so-called *buddy* server) will take over and respond to the requests sent to the failed server. The support server in this group coordinates internal communications among these cache servers and reports status to Akamai's NOCC.

Akamai offers two types of delivery services: *site delivery* and *object delivery*. Under site delivery, Akamai servers wholly replicate customer [1] sites. Alternatively, customers with replicated sites just outsource Akamai's server selection technique. Under object delivery, customers use Akamai's *Akamaizer* software to rewrite the ordinary URLs in their web sites. Later, Akamai servers retrieve and cache most embedded content-rich objects such as icons, images, pictures, audio and video clips, and deliver them to users directly. If a requested object is missing in one server group, a new request is generated from a designated server to the origin server, or to other Akamai servers in the hierarchy. Once the objects are retrieved, they will be populated among all servers in the same group.

2.2. Site delivery

For site delivery, the entire origin servers are replicated a priori. This is a suitable approach for relatively static content (such as directory portals, e.g., *yahoo.com*). As Fig. 8 shows, a user's DNS query on the origin server *www.yahoo.com* is first replied by the customer's own DNS server (*nsx.yahoo.com*) with a record *CNAME www.yahoo.akadns.net*. It may involve a root DNS server (*x.root-servers.net*) for the domain *.com* and a gTLD DNS server (*x.gtld-servers.net*) for *.yahoo.com* if their *NS* records are unavailable at local DNS servers. Next, the user's DNS query on the Akamai server (*www.yahoo.akadns.net*) is replied to by an Akamai DNS server (*zx.akadns.net*) with an *A* record that specifies a list of IP addresses. This action occurs after the Akamai DNS server has determined which server farm (or service point-of-presence, S-PoP) is close to the user's local DNS server, even if that S-PoP may not be close to the actual user. The gTLD DNS server might be contacted again if the local DNS server does not have the *NS* record for the domain *akadns.net* or *yahoo.akadns.net*.

For local DNS servers on the US West Coast or Asia Pacific Rim, Akamai DNS servers return a list of addresses in the subnetwork 66.218.71 for the generic name *www.yahoo.akadns.net*. This S-PoP is located in Sunnyvale, California. User finally receive the address list in a round robin order, so that the subsequent user requests can be *pseudo-uniformly* distributed to all servers within that S-PoP. For local DNS servers located on the

[1] We term *customer* as the content provider who distributes its content to its *users* over Akamai's CDN *server* platform.

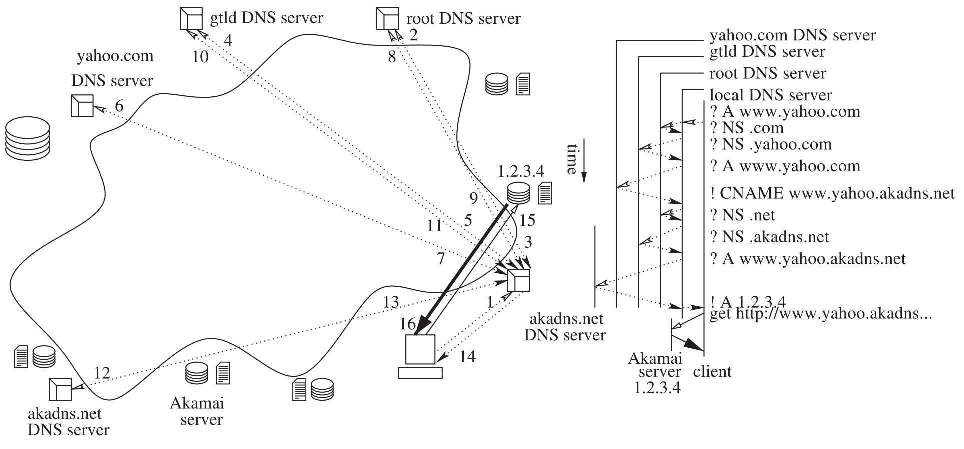

Fig. 8. Akamai site delivery.

US East Coast or in Europe, Akamai DNS servers return a list of addresses in the subnetwork 64.58.76. This S-PoP is located in Washington, DC. Other examples using Akamai's site delivery are www.about.com, www.apple.com, and www.microsoft.com.

2.3. Object delivery

Object delivery has finer granularity than site delivery, since it is designed to deliver highly dynamic content, including embedded images, audio, and video objects. Contrary to the situation with site delivery, in which the whole customer sites are replicated a priori, Akamai servers always retrieve and cache the latest objects from customer sites *on demand* in object delivery.

Fig. 9 illustrates the steps involved in object delivery. First, a user retrieves an HTML page from the origin server (e.g., www.1800flowers.com). The HTML page contains an Akamaized URL (ARL) that has the following format:

proto://aid.g.akamai.net/type/id/subid/field/original_url

In ARL, *proto* specifies the application protocol, e.g., *http* for Web or *ftp* for FTP; *id* specifies the Akamai customer; *subid* is used by the customer for a particular service. Here, *type* indicates

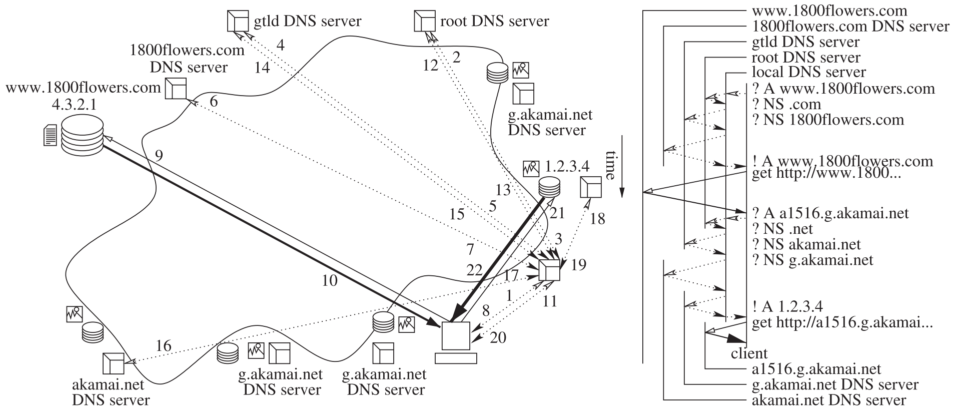

Fig. 9. Akamai object delivery.

how object freshness is expressed in the *type*-dependent *field* in ARL. Since a single Akamai server can serve more than one Akamai customer, the *id* is repeated again in the object identifier, because some old HTTP clients do not declare the host and domain name in their HTTP requests. *g.akamai.net* suggests that there are two layers of DNS-based server selections for the domain *akamai.net* and *g.akamai.net*. The reason for this extra layering will be explained in detail in the next section. A working example of ARL is

http://a1516.g.akamai.net/f/1516/1052/2h/1800flowers.com/800f_assets/images/logo.gif

where 1516 is the customer identifier for *1800flowers.com*. As the type *f* implies, this object is supposedly fresh if it was cached within 2 hours. If the object has been cached for more than 2 hours when being requested again, the cached object must be validated with and, if expired, retrieved from the origin server at

http://1800flowers.com/800f_assets/images/logo.gif.

For the host and domain name in this ARL, e.g., *a1516.g.akamai.net*, local DNS servers in the subnetwork 133.164.59 will be first referred by an *akamai.net* DNS server (e.g., *zx.akamaitech.net*) to *g.akamai.net* DNS servers (e.g., *nxg.akamai.net*) located in the subnetworks of 129.250.134, 208.187.212, and 216.32.119. According to Akamai's criteria, *zx.akamaitech.net* believes that these *g.akamai.net* DNS servers are *close* to the user's local DNS server. Finally, a list of IP addresses for Akamai servers in the subnetwork 129.250.134 are returned, since *nxg.akamai.net* believes that these Akamai servers are close to the user's local DNS server and presumably also close to the actual user. Users using other local DNS servers will get different sets of IP addresses for both *nxg.akamai.net* and *a1516.g.akamai.net*. The number of distinct *g.akamai.net* DNS servers is carefully chosen, and their location is highly correlated with the final chosen Akamai servers. Details on these procedures will be further discussed in Section 3.3.

3. DNS-based server selection schemes

In this section, we will explore the details of Akamai's DNS-based server selection schemes. Akamai has three types of DNS servers: *zx.akaDNS.net* for site delivery, and *zx.akamaitech.net* and *nxg.akamai.net* for object delivery. There are other domains, e.g., *akamaistream.net*, *akareal.net*, *edgesuite.net*, *etc.*, in Akamai's CDN, but they are the aliases of domains controlled by three basic DNS servers.

3.1. akadns.net

akadns.net has deployed multiple DNS servers (listed in Table 1) scattered around major US access networks.[2] *akadns.net*'s *NS* records are maintained by gTLD DNS servers which are not under Akamai's control. gTLD servers always return a list of *akadns.net*'s *NS* records in a simple round robin order. Thus, the first *akadns.net* DNS server in this list is not necessarily the closest to local DNS servers. The resolved addresses of *akadns.net* DNS servers are cached at local DNS servers with an initial *time-to-live* (TTL) of 2 days, or 172,800 seconds (in Table 3). Therefore, local DNS servers rarely need to contact gTLD servers for *akadns.net*'s *NS* record, which is a desired feature in DNS to minimize DNS-related traffic load and response delay. *BIND* 8 or higher, a reference DNS implementation, has a feature that rearranges received DNS *NS* records by tracking response time from these DNS servers. An *akadns.net* DNS server with a shorter response time for a local DNS server has more opportunities to serve DNS queries from that domain in the future. Akamai deployed many *akadns.net* DNS servers, hoping that, eventually, a user's local DNS server will find an *akadns.net* DNS server that is relatively close, i.e., within the same access network.

[2] As of March 2003, additional *akadns.net* DNS servers, e.g., *usex.akam.net* and *uswx.akam.net*, have been deployed on the US East and West Coasts, respectively. Also, in 2003, *zg.akadns.net* becomes unreachable from external probings.

Table 1
akadns.net DNS servers

Server	IP address	Access network	Location	Remark
ZA	216.32.65.105	exodus.net	Washington, DC	
ZB	216.52.46.145	bbnplanet.net	Denver, CO	
ZC	63.241.199.50	att.net	Dallas, TX	
ZD	206.132.160.36	glbx.net	Santa Clara, CA	
ZE	12.47.217.11	att.net	Parsippany, NJ	
ZF	63.215.198.79	level3.net	San Jose, CA	
ZG	204.248.36.131	sprintlink.net		N/A in 2003
ZH	63.208.48.42	level3.net	St. Louis, MO	

The *start of authority* (SOA) record in the 2002 version of *akadns.net* shows that the master DNS server aggressively demanded that all slave DNS servers synchronize their zone records with the master copy every 50 s. Akamai disables the zone transfer feature for external hosts due to security concerns, so we cannot identify its internal DNS replication mechanisms. From Table 2, we observe that the *serial* field in 2002 was kept constant, and that the values for zone transfer were extremely small (50 s). The *SOA* record retrieved from the same *akadns.net* DNS server in 2003 shows that the tight requirement on zone replication has been relaxed to more than 16 h. Also in 2003, only the domain *yahoo.akadns.net* has a constant *serial* and a small *refresh*. For *akadns.net*, the *serial* field now is the time in clock ticks when a *zx.akadns.net* is contacted. In other words, no matter when a slave DNS server contacts the master DNS server, it always finds a *fresher* copy of zone records to be synchronized. These settings at the master DNS server discourage record mirroring attempted by slave DNS servers.

To map a generic host name such as *www.yahoo.akadns.net* to an IP address dynamically, Akamai tries to eliminate DNS caching by assigning a very short TTL (5 min) to *A* records (Table 3). This means that after an *A* record has been cached for more than 5 min, local DNS servers have to contact *akadns.net* DNS servers again to retrieve a new copy of this record, no matter whether the record itself has changed or not. However, the *NS* record for *yahoo.akadns.net* is relatively stable, with a TTL of 25 h, which leaves enough time for local DNS servers to try all *yahoo.akadns.net* DNS servers to find the one with the shortest response time.

Table 2
akadns.net SOA record

SOA	Value in 2002 (s)	Value in 2003 (s)
Serial	50	1,048,012,862
Refresh	50	60,000
Retry	50	60,000
Expire	50	60,000
Minimum	50	300

3.2. *akamai.net*

Table 4 lists the authoritative DNS servers *zx.akamaitech.net* for the domain *akamai.net*. They are geographically spread throughout the US

Table 3
akadns.net *NS* and *A* TTL

Ask		Answer		
Name	Server	Refer	*NS*-TTL (s)	*A*-TTL (s)
net.	{a..m}.root-servers.net	{a..m}.gtld-servers.net	172,800	172,800
akaDNS.net.	{a..m}.gtld-servers.net	z{a..g}.akadns.net	172,800	172,800
yahoo...	z{a..g}.akadns.net	–	90,000	90,000
www...	z{a..g}.akadns.net	–	–	300

Table 4
akamai.net DNS servers

Server	IP address	Access network	Location	Remark
ZA	209.67.231.142	cw.net	Boston, MA	
ZB	12.47.217.18	fast.net	Bethlehem, PA	
ZC	213.161.66.159	mfn.net	London, UK	
ZD	216.32.65.14	cw.net	Sterling, VA	
ZE	210.81.97.184	alter.net	Tokyo, Japan	
ZF	63.240.15.245	attens.net	New York, NY	
ZG	213.61.5.28	colt.net	Frankfurt, Germany	
ZH	63.215.198.78	level3.net	San Jose, CA	
ZI	63.240.144.98	attens.net	Chicago, IL	
ZJ	63.210.142.26	level3.net	Dallas, TX	
ZK	64.215.170.28	gblx.net	Dallas, TX	
ZL	209.185.188.14	cw.net	Jersey City, NY	
ZM	12.129.72.181	att.net	Atlanta, GA	
ZN	193.45.1.100	telia.net	London, UK	
ZO	193.108.153.36	colt.net	UK	
ZP	209.67.231.204			N/A in 2003

and overseas access networks, and have the same coverage as Akamai's CDN server platform for object delivery. The *SOA* record in Table 5 has a reasonable *refresh* period of 25 h, for zone transfer. This table also shows a meaningful *serial* field of the time in clock ticks. If a slave DNS server finds a larger *serial* field at the master DNS server when doing refresh, it will synchronize its database by the master copy. In 2003, the default *minimum* TTL has been reduced from 9000 to 300 s to more aggressively eliminate DNS caching.

Table 5
akamai.net and *g.akamai.net* SOA

SOA	akamai.net (s)	g.akamai.net (s)
Serial	1,018,647,052	1,018,650,356
Refresh	90,000	1000
Retry	90,000	1000
Expire	90,000	1000
Minimum	90,000 (300 in 2003)	1000 (300 in 2003)

There are two dynamic mappings within *akamai.net*. As with *akadns.net*, the *NS* records of *akamai.net* returned by gTLD DNS servers are in a round robin order. Since *akamai.net* is designed to support object delivery, which is much more dynamic than site delivery, an additional DNS hierarchy (*g.akamai.net*) is introduced. The *NS* records returned by *zx.akamaitech.net* contain the IP addresses of DNS servers (*nxg.akamai.net*) for *g.akamai.net*. These servers are expected to be close to local DNS servers. *g.akamai.net* DNS servers have a TTL of 1800, 2700, or 3600 s, respectively (see Table 6). There are other similar regional intermediate domains within the *akamai.net* domain, such as *e.akamai.net* and *na.akamai.net*, which follow the same DNS structure as *g.akamai.net*.

3.3. *g.akamai.net*

The actual DNS servers for *g.akamai.net* returned by *akamai.net* DNS servers depend on the

Table 6
akamai.net and *g.akamai.net* DNS *NS* and *A* TTL

Ask		Answer		
	Server	Refer	*NS*-TTL (s)	*A*-TTL (s)
net.	{a..m}.root-servers.net	{a..m}.gtld-servers.net	172,800	172,800
akamai.	{a..m}.gtld-servers.net	z{a..p}.akamaitech.net	172,800	172,800
g...	z{a..p}.akamaitech.net	n{0..8}.g.akamai.net	1800–3600	1800–3600
a1516...	n{0..8}.g.akamai.net	–	–	20

Table 7
g.akamai.net DNS servers

Server	From local DNS server			
	133.164.59.8	129.97.34.2	192.63.105.17	202.119.24.12
n0g	129.250.134.66	130.185.5.11	194.82.174.220	210.25.241.9
n1g	129.250.134.67	130.185.5.12	194.82.174.221	210.25.241.10
n2g	129.250.134.75	130.185.5.14	194.82.174.227	210.25.241.11
n3g	129.250.134.77	130.185.5.11	62.129.135.36	210.25.241.9
n4g	129.250.134.82	130.185.5.11	64.241.221.237	210.12.127.67
n5g	129.250.134.66	130.185.5.11	194.82.174.220	210.25.241.11
n6g	208.187.212.167	63.76.54.131	213.161.66.179	210.12.127.67
n7g	216.32.119.56	130.185.5.11	62.129.135.36	212.35.120.23
n8g	129.250.134.66	130.185.5.11	64.241.221.237	210.25.241.10

location of local DNS servers (Table 7). Since host names within *g.akamai.net* supposedly have higher dynamics, they need an even smaller TTL. To avoid sending frequent queries to gTLD DNS servers (over which Akamai does not have control) or to *akamai.net* DNS servers (which might not be close to local DNS servers), a subdomain within *akamai.net*, such as *e*, *g*, and *na*, was created. The replication of a particular IP address in Table 7 represents the weight of that DNS server, and the list itself is in a round robin order. Since the chosen *g.akamai.net* DNS servers are supposedly close to local DNS servers, the TTL value for *a1516.g.akamai.net* is set extremely small at 20 s (see Table 6). This means that an *A* record for any host name within the *g.akamai.net* domain is only valid for 20 s. After that time, no matter whether the mapping changes or not, local DNS servers have to contact the *g.akamai.net* DNS server. *g.akamai.net* DNS servers also have a smaller *refresh* field in their *SOA* record, due to the higher dynamics that need to be supported (Table 5).

Usually, *akamai.net* DNS servers return a list of *g.akamai.net* DNS servers in more than 2 Akamai clusters. Therefore, local DNS servers (*BIND* 8 or higher) might have a chance to rearrange the list by tracking the response time from DNS servers in different clusters. Since the *NS* TTL in *g.akamai.net* is much smaller (1800–3600 s) than that of the site delivery DNS servers (90,000 s), the rebalancing ability of *BIND* 8 is largely compromised. Akamai assumes that if a local DNS server can receive DNS query replies from a *g.akamai.net* DNS server, users should also be able to get the requested content delivered from an Akamai server located in the same cluster. This approach avoids a situation in which a DNS server returns the IP address of an Akamai server that is actually unreachable by users. Here, users are assumed to be closely located with their local DNS servers.

4. Performance issues

In this section, we will discuss the performance issues and the built-in strengths and weaknesses of DNS-based server selection schemes. Again, we use Akamai's CDN platform as our focal case study.

4.1. User-perceived performance

By bringing web content to network edges and close to end users, CDN is expected to offer a better quality of Web experience. This expectation is based on the observation that long-haul and cross-peering traffic is likely to suffer a large delay and congestion loss. The CDN paradigm avoids the bottleneck at origin servers and their access links, and reduces the long-haul and cross-peering traffic. However, precisely how much *quantitative* performance improvement current CDN technologies can offer has not been well understood yet.

There has been debate on whether *existing* CDN technologies can improve user-perceived performance, which is measured by the *click-to-display* delay from a user's perspective. In this respect, there are several interesting but somewhat contradictory observations reported in the literature.

In [13], Johnson et al. reported that, although CDN offers a better service on average, neither of the two major CDN providers can consistently choose the best server among all available servers. This observation was based on their experimental study, which probed from three different user sites for different CDN providers. They also found that, in most cases, CDN can avoid some obvious *bad* servers. In another study, Krishnamurthy et al. [14] did a probing experiment from about two dozen academic and laboratory sites. They found that, although more and more popular sites had become CDN*ized* in 2000 than those in 1999, when taking into consideration the additional time used for dynamic DNS lookup, the overall user-perceived response time was actually *worse* than a case in which a site was not CDN*ized*. This result may appear counterintuitive. However, if we consider that: (1) in many operating systems, DNS lookup is a serialized blocking function call; and (2) many DNS servers are actually highly overloaded, then it is not hard to understand that user-perceived quality of experience may suffer in CDN.

The click-to-display delay metric also depends on many non-CDN-related factors (e.g., user location). It is understandable that different users will have different click-to-display delay experiences in the same CDN, and that experiments will have different results even with similar setups.

4.2. Strengths and weaknesses

In the rest of this section, we will examine some intrinsic strengths and weaknesses in DNS-based server selection schemes from the perspective of networking protocols, which are independent from aforementioned experimental factors such as user locations and probing setups.

4.2.1. DNS advantages

DNS-based server selection takes advantage of the existing DNS infrastructure and does not require any change for end users, which makes this technology very attractive and immediately deployable in today's Internet.

DNS-based approaches also minimize necessary changes for content providers. For site delivery, customers such as *yahoo.com* only need to give Akamai a list of addresses and locations of their replicas. They then insert a *CNAME* record in their authoritative DNS databases. Akamai's *akadns.net* DNS servers will capture the aliased DNS requests and return the IP addresses of replicas close to the inquiring local DNS server. If additional Akamai software is installed on customer premises, more performance metrics, e.g., server load, network connectivity and traffic conditions, can be added into the decision process involved with server selection.

For object delivery, customers such as *1800flowers.com* only need to run the *Akamaizer* software against their original web site in order to rewrite the URL of cacheable objects. They then make the rewritten web site available to the public. *Akamaizer* attaches a prefix to the original URL, which directs user requests for these objects to Akamai servers and indicates object attributes such as freshness. It is unnecessary to make other changes on customer premises.

4.2.2. DNS disadvantages

DNS-based server selection schemes operate on the upper layer of the networking protocol stack. However, certain applications treat resolved IP addresses differently. For example, some applications maintain a small pool of resolved IP addresses, and are not interested in the TTL of an IP address. Therefore, during the lifetime of these applications, an IP address is bound to the designated server deterministically, regardless of how Akamai dynamically maps the host name. This situation significantly reduces the effectiveness of DNS dynamic mapping. Once the transport connection is established, users have to stick with a particular server, even when traffic conditions and server load change dramatically during the session.

Besides the potential problems in applications, DNS software, including local DNS servers and client resolvers, may have trouble handling DNS records with a very small (or zero) TTL. An *A* record with a small TTL requires that local DNS servers send more queries for the same mapping to the authoritative DNS server. An *NS* record with a small TTL means that there are more queries sent to the authoritative DNS server of its parent domain. These small TTL settings sometimes unnecessarily cause more DNS traffic and introduce more DNS-related delay. Since DNS traffic is transported by UDP and has no embedded flow, error, and congestion mechanisms, DNS queries and replies are subject to congestion and packet loss. If there are multiple embedded objects served by different CDN providers in the same HTML document, users can expect surfing *pause* and sometimes browser *freeze* as a result of frequent DNS queries being implemented by blocking function calls in many systems. Moreover, since DNS was designed to map between an almost static host name and its IP address, with the goal of minimizing extra traffic and response delay, the choices of transport protocol, error detection and recovery schemes are not optimized for the highly dynamic mapping currently *hack*ed in CDN. Finally, since the mapped IP address from a generic host name normally cannot be mapped back to the same host name, it sometimes triggers certain add-on security measures (e.g., DNS anti-spoofing) in some DNS implementations.

A third issue is the resolution granularity of DNS-based server selections which are limited to local DNS servers. In other words, all requests from clients that share the same local DNS are treated equally. Since only host and domain names, not protocol and object identifiers in URLs, are handled by DNS, there is no distinction between *http://mp3.com* and *ftp://mp3.com*, even though these are two different services. [3] Therefore, DNS-based approaches have very limited granularity with regard to the type of services that they can distinguish. To preserve the visibility of its customers, Akamai uses the *CNAME* record to redirect DNS requests. But since Akamai's customers cannot give up control over the whole domain, this approach does not work as expected for a plain domain name, e.g., *http://yahoo.com*. Local DNS servers always get the same list of addresses for *yahoo.com* regardless of their actual location. This case clearly compromises the objective of DNS dynamic mapping. One way to work around this problem is to add an HTTP redirect primitive, or to use an HTML refresh option to forward *yahoo.com* to *www.yahoo.com*. However, this workaround adds the additional overhead of multiple, sequential, and short TCP connections to different web servers.

4.2.3. User-DNS proximity

There is an implicit but critical assumption within DNS-based server selections: users are always close to their local DNS server. The chosen Akamai server, which is close to local DNS servers, is also assumed to be close to end users. However, this assumption is not always true in today's Internet, especially for large corporations and ISP networks that have wide coverage, multiple connections to other domains, and various management policies on how to choose and place local DNS servers. Even if local DNS servers are close to end users, Akamai's DNS server may still mistakenly treat other DNS servers, e.g., a DNS forwarder or split DNS server, as local DNS servers, thereby leading to suboptimal decisions (see Fig. 10).

In [15], Mao et al. observed that users and their local DNS servers may not be close to each other in today's Internet. In their experiment, by using HTTP redirect, the initial web server captures the IP address of users or proxy servers, and then returns a coded host name. The authoritative DNS server for that designated domain captures the address of a local DNS server that asks for that coded name. By matching the user IP address and local DNS server address, they found that only about 64% of all users have a *local* DNS server

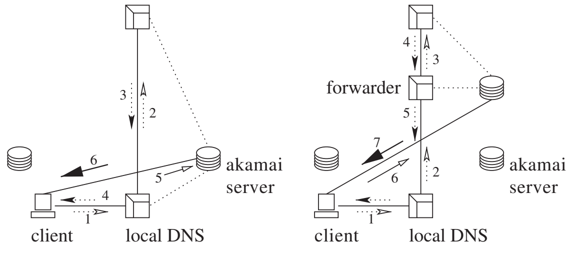

Fig. 10. User-DNS proximity in server selection.

[3] A DNS extension record *SRV* [12] has been proposed to map a generic service name to a particular host name or IP address, protocol identifier, and port number. But this extension has not been widely implemented and deployed.

with the same autonomous system (AS) number. When considering the longest address prefix in BGP routing tables, only about 16% of all users and their local DNS servers are actually in the same network cluster. This fact reaffirms the potential flaw of server selection based on local DNS servers: the chosen server might not be close to requesting users.

In another study, Bestavros and Mehrotra [4] used an approach to cluster users and their local DNS servers. They instrumented their clustering algorithms by using the data obtained from a web server in conjunction with its authoritative DNS server. Surprisingly, they found that only 32% of such clusters are correlated in the round-trip time (RTT) metric from four different probing sites. Even for the hop count, correlation is only about 60%, which is still far from satisfactory when assuming the proximity between a user and its local DNS server.

In [16], Shaikh et al. observed a similar proximity problem by collecting web and DNS access log files. After matching web clients to their DNS servers, they defined *cluster size* as the difference of hop counts for a web client and its local DNS server from a probing site. They found that only 15% of web client and its DNS server pairs are in 1-hop clusters. The median cluster size is 5 hops, and more than 30% pairs are at least 8 hops away. This fact indicates that a large number of web clients are topologically distant from their *local* DNS servers. These researchers also conducted experiments to measure the network distance from a dial-up client to the assigned local DNS servers. Again, they found that the user-DNS proximity assumption is far from realistic, even when measuring the direct hop counts between dial-up clients and their assigned local DNS servers.

4.2.4. Locationing accuracy

Given the fact that the current Internet uses mixed organization and provider-based address allocation schemes, the source address of a DNS query cannot offer complete information about its physical location and network neighborhood. A inquired DNS server can only estimate these factors by looking up host name, address allocation, *AS* number, and the longest address prefix in BGP routing tables, *etc.* [4] of the source address in a query. There are several drawbacks to these approaches. For example, some IP addresses do not have reverse DNS mapping, either because the address owner does not maintain such a database, or the owner is unwilling to release it due to security concerns. Even with a generic domain name, it is still difficult to find out the exact location of an IP address, since many large corporations and ISPs have global coverage and multiple connections to other domains. The same problem also occurs when looking up the address database maintained by ARIN, APNIC, or RIPE, because these databases keep an owner record of address blocks, not the individual address in the geographical or network space.

For a local DNS server located in the subnetwork 133.164.59, *akamai.net* DNS servers choose the DNS servers of the domain *g.akamai.net* in subnetwork 61.200.81 and 139.130.1. The subnetwork 133.164.59 is physically located on the US West Coast, but the returned *g.akamai.net* DNS servers are actually in Japan and Australia, respectively. This is because the address block 133.164 is registered in Japan. Clearly, there are other *g.akamai.net* DNS servers that are much closer to this local DNS server, since the subnetwork 133.164.59 has multiple connections to other local domains. In this case, the local DNS server has to contact the assigned transpacific *g.akamai.net* DNS server rather frequently (although unnecessarily) due to a very small TTL associated with the *A* and *NS* records.

4.2.5. Decision point

Akamai makes the server selection decision at Akamai DNS servers, which is neither close to end users nor to local DNS servers. Ideally, server selection should be done at the user side, since only end users can tell which server is the best according to their own criteria. If this ideal condition cannot

[4] There is a proposal regarding the DNS *LOC* record for domains that describe their geographical location. However, this proposal has not been widely adopted. Moreover, geographical location is not always correlated with network connectivity.

be satisfied, the closer the decision point is to end users, the better the chance that a near-optimal server will be chosen. In Akamai's approach, even when users are close to their local DNS servers, server selection cannot guarantee the best choice since Akamai cannot fully determine the location of local DNS servers and does not have complete knowledge about the network status and connectivity between end users (or their local DNS servers) and the chosen Akamai server.

In a *triangle-like* connectivity (Fig. 11) among users, their local DNS servers, and the available Akamai servers, Akamai DNS servers probably can only probe the connectivity between Akamai servers and the user's local DNS servers (assuming that Akamai DNS servers are located in the same cluster as Akamai servers). Without any knowledge about network connectivity and conditions between users and their local DNS servers, it is very difficult to assess which Akamai servers will be the best to fulfill the next request from a user.

Inktomi, another CDN provider, reportedly plans to partner with AOL to implement its server selection technique into individual user's web browsers. This approach moves the decision point toward end users. However, at the time of writing, we do not have detailed information on how this schema works, and thus cannot verify how it will cope with the scalability of control information exchanges among a large number of end users. Also, since AOL has a very special type of network connectivity, whether this approach is still applicable to many non-AOL users with other ISPs remains unknown.

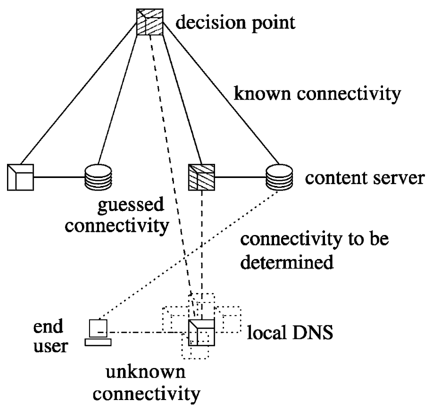

Fig. 11. Triangle relations in connectivity among users, their local DNS server, Akamai servers, and Akamai DNS servers.

5. Conclusions

In this paper, we have examined the scalability problem associated with the traditional client–server model when delivering popular services over the Internet. We focused on the DNS-based server selection technique for content distribution, and used Akamai's CDN platform for illustration. The performance of DNS-based server selection was examined through a set of user-based experimental studies. We made an assessment on the strengths and weaknesses of DNS-based selection schemes. We find that, although DNS-based server selection schemes have the advantage of causing minimum changes to existing infrastructure, they suffer certain weaknesses that lead to inaccurate decisions. Furthermore, we show that the excessive hacking of DNS in CDN may overburden the original design of DNS, leading to further performance compromises.

References

[1] Akamai Technologies, Inc., http://www.akamai.com.
[2] P. Albitz, C. Liu, DNS and Bind, fourth ed., O'Reilly, Sebastopol, CA, 2001.
[3] G. Barish, K. Obraczka, World wide web caching: trends and techniques, IEEE Communications Magazine 38 (5) (2000) 178–184.
[4] A. Bestavros, S. Mehrotra, DNS-based Internet client clustering and characterization, in: Proc. 4th IEEE Workshop on Workload Characterization (WWC'01), Austin, TX, 2001.
[5] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, Web caching and Zipf-like distributions: evidence and implications, in: Proc. IEEE INFOCOM'99, 1999, pp. 126–134.
[6] P. Christy, The network providers business case for Internet content delivery, Internet Research Group, http://www.irgintl.com/, 1999.
[7] Cisco Systems, Load balancing: a solution for improving server availability, White Paper, http://www.cisco.com/warp/public/cc/pd/cxsr/400/tech/lobal_wp.pdf.
[8] Cisco Systems, Multinode load balancing, White Paper, http://www.cisco.com/warp/public/cc/pd/ibsw/mulb/tech/mnlb_wp.pdf.

[9] Cisco Systems, Cisco distributed director, White Paper, http://www.cisco.com/warp/public/cc/pd/cxsr/dd/tech/dd_wp.pdf.

[10] Cisco Systems, Cisco content router 4400, Product Literature, http://www.cisco.com/warp/public/cc/pd/cxsr/cxrt/prodlit/cr44_ds.pdf.

[11] P. Danzig, K. Obraczka, A. Kumar, An analysis of wide-area name server traffic: a study of the Internet domain name system, in: Proc. ACM SIGCOMM'92, 1992, pp. 281–292.

[12] A. Gulbrabdsen, P. Vixie, L. Esibov, A DNS RR for specifying the location of services (DNS SRV), IETF RFC-2782, February 2002.

[13] K. Johnson, J. Carr, M. Day, M. Kaashoek, The measured performance of content distribution networks, in: Proc. IEEE WCW'01, Boston, MA, 2001.

[14] B. Krishnamurthy, C. Wills, Y. Zhang, On the use and performance of content distribution networks, in: Proc. 1st ACM SIGCOMM Internet Measurement Workshop (IMW'01), San Francisco, CA, 2001.

[15] Z. Mao, C. Cranor, F. Douglis, M. Rabinovich, A precise and efficient evaluation of the proximity between web clients and their local DNS servers, in: Proc. USENIX'02, 2002.

[16] A. Shaikh, R. Tewari, M. Agrawal, On the effectiveness of DNS-based server selection, in: Proc. IEEE INFOCOM'01, 2001, pp. 1801–1810.

[17] Squid, Squid web proxy cache, http://www.squid-cache.org.

Jianping Pan obtained his B.S. and Ph.D. degrees in Computer Science from Southeast University, Nanjing, China in 1994 and 1998, respectively. From 1999 to 2001, he was a Postdoctoral Fellow and Research Associate with the Center for Wireless Communications at University of Waterloo, Waterloo, Ont., Canada. Since September 2001, he has been a Member of Research Staff with the IP Networking Research Department at Fujitsu Laboratories of America, Sunnyvale, California, USA. His research interests include transport protocols and application services for multimedia, high-speed and mobile networks. He is a member of ACM and IEEE.

Y. Thomas Hou obtained his B.E. degree from the City College of New York in 1991, the M.S. degree from Columbia University in 1993, and the Ph.D. degree from Polytechnic University, Brooklyn, New York, in 1998, all in Electrical Engineering. From 1997 to 2002, he was a research scientist and project leader at Fujitsu Laboratories of America, IP Networking Research Department, Sunnyvale, California (Silicon Valley). He is currently an Assistant Professor at Virginia Tech, The Bradley Department of Electrical and Computer Engineering, Blacksburg, Virginia. His research interests include wireless video sensor networks, multimedia delivery over wireless networks, scalable architectures, protocols, and mechanisms for differentiated services Internet, and service overlay networking. He has published extensively in the above areas and is a co-recipient of the 2002 IEEE International Conference on Network Protocols (ICNP) Best Paper Award and the 2001 IEEE Transactions on Circuits and Systems for Video Technology Best Paper Award. He is a member of IEEE and ACM.

Bo Li received the B.S. and M.S. degrees in Computer Science from Tsinghua University, Beijing, PR China, in 1987 and 1989, respectively, and the Ph.D. degree in Computer Engineering from the University of Massachusetts at Amherst in 1993. Between 1994 and 1996, he worked on high performance routers and ATM switches in IBM Networking System Division, Research Triangle Park, North Carolina. Since January 1996, he has been with Computer Science Department, the Hong Kong University of Science and Technology, where he is now an Associated Professor. He also holds an Adjunct Researcher position at Microsoft Research Asia (MSRA), Beijing, China. His current research interests include wireless and mobile networks supporting multimedia, video multicast, and all optical networks with WDM. He has published over 140 technical papers in referred journals and conference proceedings. He has been an Editor or a Guest Editor for 14 journals, and involved in the organization of over 30 conferences. He currently serves as the Co-Chair of the Technical Program Committee of IEEE Infocom'2004. He is a member of ACM and a senior member of IEEE.