Measuring and Evaluating Large-Scale CDNs

Cheng Huang
Microsoft Research
Redmond, WA 98052

Angela Wang
Polytechnic Institute of NYU
Brooklyn, NY 11201

Jin Li
Microsoft Research
Redmond, WA 98052

Keith W. Ross
Polytechnic Institute of NYU
Brooklyn, NY 11201

ABSTRACT

CDNs play a critical and central part of today's Internet infrastructure. In this paper we conduct extensive and thorough measurements that accurately characterize the performance of two large-scale commercial CDNs: Akamai and Limelight. Our measurements include charting the CDNs (locating all their content and DNS servers), assessing their server availability, and quantifying their world-wide delay performance. Our measurement techniques can be adopted by CDN customers to independently evaluate the performance of CDN vendors. It can also be used by a new CDN entrant to choose an appropriate CDN design and to locate its servers. Based on the measurements, we shed light on two radically different design philosophies for CDNs: the Akamai design, which *enters deep into ISPs*; and the Limelight design, which *brings ISPs to home*. We compare these two CDNs with regards to the numbers of their content servers, their internal DNS designs, the geographic locations of their data centers, and their DNS and content server delays. Furthermore, we study where Limelight can locate additional servers to reap the greatest delay performance gains. As a byproduct, we also evaluate Limelight's use of IP anycast, and gain insight into a large-scale IP anycast production system.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Measurement techniques, Performance attributes.

General Terms

Measurement, Performance.

Keywords

Content Distribution Network, Delay performance, Open recursive DNS server, Data center selection, IP anycast.

IMC'08, October 20–22, 2008, Vouliagmeni, Greece.

1. INTRODUCTION

A Content Distribution Network (CDN) deploys content servers in multiple locations, often over multiple backbones and ISPs, and even in multiple POPs within different ISPs. The servers cooperate with each other, transparently moving content behind the scenes to optimize the end user experience. When a client makes a request, the CDN generally chooses a server at a location that is near the client, thereby optimizing the perceived end-user experience. There are as many as 28 commercial CDNs [2], including Akamai, AT&T, NTT Communication, Limelight, Mirror Image, Level 3, Verisign and Internap. There are also a number of non-commercial ones (e.g. [3, 4]).

The most traditional CDN services include distributing static Web pages and large file downloads, such as software patches and upgrades. CDNs also provide application acceleration, delivering dynamic content and supporting e-commerce, back-end databases, SSL, and Web 2.0 applications. CDNs are also assisting enterprise customers in providing rich Web applications with context and location-aware services. Increasingly, CDNs are marketing themselves as *the* solution for video distribution over the Internet. Leading CDN companies such as Akamai and Limelight are now offering streaming media delivery, distributing media for Major League Baseball, Paramount Pictures, BBC [1], and so on. The enormously popular user-generated video site, YouTube, is currently distributed by the Limelight CDN. The CDNs taken together, with tens of thousands of servers deployed throughout the world serving a major fraction of Internet traffic, now make up a critical and central part of the Internet infrastructure.

The current design of CDNs broadly follows two different philosophies. One philosophy is to *enter deep into ISPs*, by deploying content distribution servers inside ISP POPs. The idea is to get close to end users, so as to improve user-perceived performance in terms of both delay and throughput. Such a design results in a large number of server clusters scattered around the globe. Because of this highly distributed design, the tasks of maintaining and managing the networks

become very challenging. It also involves sophisticated algorithms to shuffle data among the servers across the public Internet. A leading representative commercial CDN of this type is the Akamai network.

The other design philosophy is to *bring ISPs to home*, by building large content distribution centers at only a few *key* locations and connecting these centers using private high speed connections. Instead of getting inside the ISP's POPs, these CDNs typically place each distribution center at a location that is simultaneously near the POPs of many large ISPs (for example, within a few miles of both AT&T and Verizon POPs in a major city). Compared to the first design philosophy, such a design typically results in lower maintenance and management overhead, *possibly* at the expense of higher delay to end users. A leading representative commercial CDN of this type is Limelight.

Given the vast number of competing CDN companies, and the radically different design approaches they are taking, we would like to quantitatively evaluate the performance of current CDN companies and their architectures. Through extensive and thorough measurements, the goal of the paper is to explore CDNs in depth, determine their delay performance, as well as compare and contrast the radically different design philosophies of CDNs. The contributions of this paper are as follows:

- **Charting CDN Networks.** A CDN's infrastructure mainly consists of a content-server network and a DNS server network. To evaluate the performance of a CDN, it is first necessary to determine all of a CDN's content and DNS servers, and additionally understand the specific internal DNS design (for example, whether DNS servers are organized in a hierarchy and/or if IP anycast is employed). However, without direct access to a large number of globally-distributed clients, it is challenging to completely chart a CDN network. In this paper, we leverage the DNS infrastructure to obtain more than a quarter million public DNS servers as query vantage points. In addition, we develop new techniques to compensate the inadequacy of DNS servers themselves. We exploit this infrastructure to find *all* of a CDN's content servers and *all* of its DNS servers, and eventually obtain *complete* coverage. We determine the IP addresses of all ∼27,000 servers in the Akamai network and all ∼4,100 servers in the Limelight network (at the time of our study). In addition, we find that Akamai operates ∼6,000 DNS servers, while Limelight operates ∼3,900. To the best of our knowledge, this is the most complete discovery of these two networks and the first report on the scale of their large DNS infrastructure (Sections 2, 3 and 4).

- **Measuring CDN Delay Performance.** In assessing a CDN, it is necessary to accurately and comprehensively measure its delay performance. A CDN has two major delay components: DNS resolution delay, that is, the time for the CDN's internal DNS system to supply the client the address of the "best" CDN content server; and the content-server delay, that is, the round-trip time between client and selected CDN server. Quantifying the delay performance of a CDN at large scale requires the ability to either capture traffic on the CDN's servers or control a large number of globally distributed clients. A *key* discovery made in this paper is that large-scale CDNs typically employ a huge number of DNS servers that are *co-located* with their content servers. This observation enables us to modify the King [8] delay measurement approach to carry out a global-scale delay analysis of Akamai and Limelight. As a result, we discovered that at the 95% percentile, the DNS resolution delay of Limelight is 23% higher than Akamai (170ms vs. 138ms); and the content server delay of Limelight is 114% higher than Akamai (222ms vs. 103ms). It is intuitive that Akamai has better delay performance than Limelight, but we quantify the difference from a large number of vantage points (Sections 6).

- **CDN Availability.** Using our knowledge of the IP addresses of all the content servers, we also explore the availability of the content servers and server clusters for both Akamai and Limelight. Additionally we provide statistics for server uptimes (Sections 5).

- **Methodology for CDN Deployment.** Employing the charting and delay measurement methodologies discussed, we show how a new entrant can choose the location of its servers to optimize delay performance. The idea is to consider each of the Akamai content servers (which have been fully charted) as potential candidate locations. A new entrant can use this methodology to evaluate the performance of a hypothetical deployment before even building its network. An existing CDN vendor can also use the methodology to refine its own deployment, by evaluating the performance improvement that can be obtained by deploying servers at additional locations. Applying the methodology to Limelight, we conclude that Limelight can reduce its delay performance to within 10% of Akamai by deploying only 5 more data centers. We also conclude that Limelight's current deployment is robust to failures, that is, with ideal global traffic management, one data center failure will minimally affect its delay performance. Even in the highly unlikely case when there are up to 4 concurrent data center failures, the worse delay performance is only 28% more than normal (Sections 7).

- **Evaluating IP Anycast.** As a byproduct of charting CDN topologies, we have discovered that Limelight employs IP anycast for its DNS system. Mea-

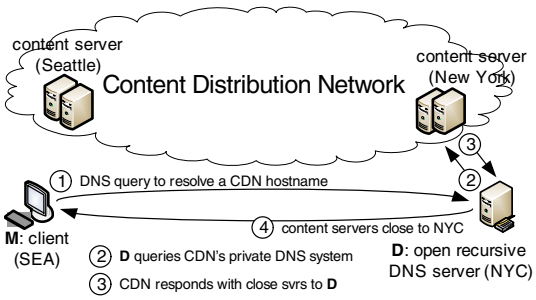


Figure 1: Charting Content Distribution Networks

suring Limelight's DNS resolution performance allows us to evaluate an IP anycast production system that is widely distributed (in 18 locations worldwide) and has extensive network connectivity (peering with hundreds of ISPs). We conclude that IP anycast is *very* effective in the real-world – fewer than 10% of the DNS queries are routed to servers more than 25ms away from the nearest server (Sections 8).

2. CHARTING CONTENT DISTRIBUTION NETWORKS

In this section we present a methodology for discovering all content servers and DNS servers of a given CDN. But before presenting the methodology, it will be instructive to first review the basics of CDN.

2.1 Brief review of CDN basics

Many modern CDNs use "DNS magic" to connect end users to *nearby* content servers. Here is a simple example – an end user visiting http://www.bestbuy.com resolves the hostname to an IP address by querying its *local* DNS (LDNS) server. The LDNS then contacts the *authoritative* DNS server of www.bestbuy.com, which returns a CNAME a1105.b.akamai.net to the query of www.bestbuy.com. The LDNS again queries the authoritative DNS server of a1105.b.akamai.net, which eventually returns the IP addresses of an Akamai content server hosting BestBuy's content. (Modern CDNs normally return more than one IP addresses – usually two – to allow client side load balancing.) Readers can refer to the detailed illustration in [23].

It's clear that a CDN maintains two different types of servers: *content servers*, which serve content to clients on the behalf of the CDN's customers; and *DNS servers*, which the CDN uses to supply the client with a nearby content server. As pointed out by Su et al. [14], the same physical machine may serve both as a content server and a DNS server for a CDN.

2.2 Charting CDN Networks

Our CDN-charting methodology is based on the following two key observations: 1) depending on where a LDNS originates the query, the CDN chooses a nearby content server (and returns the IP address thereof); and 2) due to load balancing, for the same LDNS, CDNs return different content servers to the same query over time. Therefore, to chart a CDN network, conceptually, we can take two steps:

1. First we determine all the CNAMEs that are used by the CDN's customers.

2. Second, we query a large number of LDNSs all over the world, at different times of the day, for all of the CNAMEs found in step 1. As illustrated in Figure 1, a client in Seattle (M) sends to a LDNS in New York (D) a DNS query for a CDN CNAME. When D resolves this CNAME, it will get results back from CDN's DNS servers. From the CDN's perspective, the request comes from D, so it returns a content server close to D. Hence, M will get results back from D with content servers close to New York, instead of Seattle.

We now describe these two steps in detail.

2.2.1 *Finding CDN CNAMEs*

The first step is straightforward. To find all the CNAMEs used by a particular CDN, we first gather a large set of web hostnames. This can be done in a number of different ways, including crawling a search engine. In our experiments, we obtained over 16 million web hostnames directly from Windows Live search logs. For each such hostname, a simple DNS query tells us whether it resolves to a CNAME or an IP address, and whether the CNAME belongs to the target CDN (e.g., to Akamai or to Limelight).

2.2.2 *Locating vantage points*

In order to have a full coverage of a CDN, it is desired to locate a large number of widely distributed LDNSs. In addition, these LDNSs need to be *open* to external queries for DNS resolution – due to security concerns, many LDNSs are configured to only perform DNS resolution on behalf of their internal users. Fortunately, as pointed out by Gummadi et al. [8] and repeatedly explored by others [10, 18], there are still a large number of open LDNSs, i.e., they *will* resolve DNS queries for users from anywhere. In addition, such DNS resolutions are usually recursive (involving several iterative DNS queries from the LDNS, before eventually mapping to IP addresses). Hence, we call these LDNSs *open recursive DNS servers*.

Here, we start with two sets of source data to locate open recursive DNS servers. The first set consists of a log of clients from the MSN Video service. We obtain over 7 million unique client IP addresses. Through *reverse* DNS lookup, we find the authoritative name servers for these clients. Then, trial DNS queries are sent to test the openness of these authoritative name

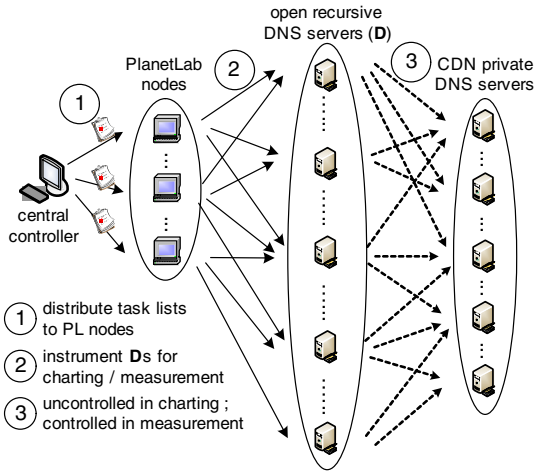


Figure 2: Measurement Platform

servers. We record all those that respond. The second set is the same ∼16M web hosts used in Step 1. Again, we find the authoritative name servers for these hostnames and record those responding to trial DNS queries. The results are summarized in list below.

DNS Servers	clients (7M)	web hosts (16M)
authoritative	83,002	1,161,439
open recursive	26,453	440,054

It turns out that, in many cases, different DNS servers map into same IP addresses, so we eventually obtain 282, 700 unique open recursive DNS servers after processing the total (26, 453+440, 054) DNS servers. These servers cover 121, 556 different /24 IP prefixes and are distributed in 210 countries, as detailed in Table 1. Note that our approach is very effective, as the two data sets (especially the search log) give us an order of magnitude more useful DNS servers than sampling IP prefixes, an approach taken by previous work [18].

2.2.3 *Measurement platform*

The complete charting of the DNS involves a large number of DNS queries, i.e., querying each of the CDN CNAMEs (found in the first step) from all the open recursive DNS servers (found in the second step). For instance, if a CDN has a few thousand of unique CNAMEs (which is indeed the case, as we will see later), even choosing a small percentage of the vantage points (say 10%) can easily result in over 100 million DNS queries. To speed up the charting process, reduce load on any single query server, as well as avoid potential complaints about DNS attacks, we have developed a distributed execution platform, which splits the complete giant task into many smaller jobs, spreads these jobs onto PlanetLab nodes and executes on the PlanetLab nodes in parallel; see Figure 2. This way, we can keep a very low load (thereby consuming very modest resources) on each PL node and still complete one round of charting quickly (e.g., a rough calculation shows that, with 300 PlanetLab nodes and 3 DNS queries per second from each node, the entire task takes slightly more than one day to complete.). Note that the same platform is also used in delay measurement experiments, which are covered in a later section.

Region	# of countries	# of ASes	# of servers	% of total
North America	33	4,875	121,491	42.97
Europe	45	5,278	82,004	29.01
Asia	52	1,863	63,481	22.45
South America	14	400	8,714	3.08
Oceania	16	381	5,042	1.78
Africa	50	207	1,545	0.55
Unknown	-	-	446	0.16
Total	210	12,219	282,723	100.00

Table 1: Coverage of Open Recursive DNS Servers

3. THE AKAMAI NETWORK

In this section and the next one, we apply our CDN-charting methodology to two leading CDNs, namely, Akamai and Limelight. For each CDN, we not only chart out its content server network but also its DNS infrastructure.

3.1 Immediate results from charting

3.1.1 *Akamai CNAMEs*

Through straightforward DNS resolutions of the 16M unique web hostnames, we obtained 3, 260 unique Akamai CNAMEs, which can be classified into 3 types: Interesting enough, these three types of CNAMEs appear to offer very different services.

	type	# of CNAMEs
(a)	*.akamai.net	1964
(b)	*.akadns.net	757
(c)	*.akamaiedge.net	539

3.1.2 *Service types*

It appears that the 3 types of Akamai CNAMEs are for 3 distinctly different services.

- type (a) – akamai.net (1964 in total). This type matches the conventional understanding of Akamai, as a content distribution network: 1) Each DNS resolution returns 2 IP addresses (occasionally more); 2) When resolved from different locations, the obtained IP addresses are different; 3) For each Akamai CNAME, there are hundreds (or even thousands) of unique IP addresses by aggregating from all the vantage points. Altogether, we've discovered over 11, 500 unique IP addresses belonging to this type.
- type (b) – akadns.net (757 in total). Apparently, Akamai customers use this type only for global load balancing [5], not for content distribution. A customer

may elect this service if it possesses its own content servers; Akamai is then solely used to direct clients to the customer's servers. We make this conjecture because each of the akadns.net CNAMEs only map to a few IP addresses. As a typical example, disasteraid.fema.gov.akadns.net maps only to 3 geographically distributed IP addresses (combined from all the vantage points).

- type (c) – akamaiedge.net (539 in total). We obtain more than 36,000 unique IP addresses from this service. The set of addresses is completely disjoint from the 11,500 IP addresses found for type (a) CNAMEs. From detailed examinations (see Appendix), we conclude that this service is for dynamic content distribution. In addition, Akamai uses virtualization technology to provide customers with isolated environment.

In summary, we have discovered 11,500 content servers for type (a) CNAMEs (handling Akamai's conventional content distribution service); 0 content servers for type (b) CNAMEs (for Akamai's load-balancing service); and another ∼36,000 unique IP addresses for type (c) CNAMEs. These numbers are particularly surreal given that Akamai claims to deploy ∼25,000 servers worldwide, a number which is much larger than 11,500 and much smaller than 36,000. This is resolved in the subsequent subsections.

3.2 Expanding the Akamai server list

We make another key observation – Akamai tends to use contiguous blocks of IP spaces. Hence, if 12.14.146.45 and 12.14.146.48 are IP addresses discovered through our charting method, then it is reasonable to conjecture that 12.14.146.46-47 are also part of Akamai IP addresses, as well as addresses beyond (e.g., 12.14.146.49). A conjectured IP address can be verified if we can download an Akamai customer's content from it. This might cause concern, because not only Akamai's server will be verified through this test, regular proxy servers will as well. As a matter of fact, this is indeed true. During the expansion process, we actually included an IP address, which through reverse DNS mapping, shows it belonging to the CoralCDN. (It is interesting that CoralCDN *does* serve clients who request content not in its delivery list. We verified that commercial CDNs (both Akamai and Limelight) do *not* allow such behavior, i.e., trying to download content served by one CDN from another will *not* succeed.)

Fortunately, we discovered a more reliable method. We send a HTTP HEAD request to potential Akamai servers – "HEAD / HTTP/1.0". This is a request to get the meta information of the root level file. To Akamai's servers, this is not a valid request (because it cannot be associated with any customer), so it will respond with "HTTP/1.0 400 Bad Request" and additionally with "Server: AkamaiGHost". It is the "Server" field that confirms this is indeed an *Akamai Global Host*. Note that Akamai servers do *not* reveal themselves if a valid HTTP HEAD request (e.g. "HEAD / HTTP/1.1" with "Host: www.bestbuy.com") is submitted, in those cases, the response will contain "Server: Apache", which isn't helpful at all. We remark that there are about 1,000 exceptions where servers reply with "Server: Microsoft-IIS/6.0". We assume these servers still belong to Akamai, as we don't expect other proxy servers to have similar characteristics – occupying only a few IP space blocks, each with a large number of consecutive addresses.

In the end, we are able to first eliminate 1,413 IP addresses from our previously discovered list of 11,500 IPs corresponding to service type (a) and further expand the Akamai server list to more than 23,000 servers.

3.3 Large-scale DNS infrastructure

It is well-known by now that Akamai employs a two-tier DSN infrastructure. As an example, a query of a1105.b.akamai.net is first sent to Akamai's top level domain servers (e.g., za.akamaitech.net), which responds with referrals of second level domain servers (e.g., n0b.akamai.net). It is the second level domain server that eventually returns IP addresses [23].

We observe that the same second level name server (e.g., n0b.akamai.net) maps to different IP addresses when being resolved from different locations. This naturally raises a question – how many DNS servers does Akamai actually operate for its DNS infrastructure? To answer this question, we need to find all the IP addresses corresponding to all the second level name servers. But this is exactly the same problem as charting the content servers of Akamai! Hence, we first obtain a list of the second level name servers (e.g., n0b.akamai.net) from the Akamai CNAME list. There are 261 of them in total. Then, we instrumented the same measurement platform and discovered that Akamai uses 5,313 name servers for the second level domain, which is a surprisingly large number. Additionally, we observe that the IP addresses of these name servers are a subset of the 23,000 IP addresses discovered when charting the Akamai type (a) content servers. Moreover, they cover all the /24 IP prefixes of those 23,000 IP addresses. At this point, it is natural to conjecture whether Akamai runs at least one name server in each of its distributed cluster. (We will confirm later that this is indeed the case!)

3.4 Summary of results

Combining with discoveries on the akamaiedge.net network, we eventually compile a list of more than 27,000 content servers, which we conclude is the *complete* Akamai global network. Among these servers, ∼6,000 are

also running as DNS servers. The discovery of Akamai's large-scale DNS infrastructure is the key to the delay performance study in later sections.

Using a commercial IP to geolocation database, the total ∼27,000 Akamai servers map to 65 different countries. More than 60% servers are in US and about 90% in 10 countries, as shown in Table 2. Note that the IP to geolocation database is reasonably accurate only at the country level. In a later section, when the need arises, we will describe how to locate the Akamai servers *precisely*. Additionally, these servers span over 656 ASes. Interestingly, the distribution here is much flat. For instance, among all the US servers, there are only 15% in 7 of the top 10 ISPs [7].

Country	# of IP	Percentage(%)
United States	16,843	61.09
United Kingdom	1,690	6.13
Japan	1,622	5.88
Germany	1,103	4.00
Netherlands	857	3.11
France	722	2.62
Australia	514	1.86
Canada	438	1.59
Sweden	396	1.44
Hong Kong SAR	370	1.34
Others	3018	10.95
Total	27,573	100.00

ISP	# of IP	Percentage(%)
Qwest	941	5.59
AT&T	869	5.16
Time Warner	365	2.17
Verizon	261	1.55
America Online	75	0.45
Cablevision	18	0.11
Charter	6	0.04
Others	14,308	84.95
Total	16,843	100.00

Table 2: Geographic and ISP Distributions of the Akamai Network.

4. THE LIMELIGHT NETWORK

The same process and measurement platform is used to chart the Limelight network (see [6] for additional details). In fact, charting Limelight network is even easier – since Limelight has its own AS, we can simply crawl all the IP addresses in the AS and verify them. In this section, we focus on unique discoveries about Limelight.

4.1 IP anycast based DNS infrastructure

Radically different from the Akamai network, the Limelight private DNS infrastructure appears to have only one level – all DNS requests to domain llnwd.net are handled by "four" name servers: dns11.llnwd.net through dns14.llnwd.net. Resolving from all the global vantage points, these name servers always map to the same IP addresses. However, tracerouting these name servers from different locations reveals what's really happening behind the scene. For example, a traceroute from a Berkeley traceroute server shows that dns11.llnwd.net is 11.47 ms away, passing the last hop Limelight router (lax.llnw.net – obviously located at Los Angeles) 11.45 ms away. Then, another traceroute from a Princeton traceroute server shows that it is 9.05 ms away, passing the last hop Limelight router (iad.llnw.net – Washington, D.C.) 8.70 ms away. Moreover, yet another traceroute from a University of Washington traceroute server shows it is 1.24 ms away, passing the last hop Limelight router (sea2.llnw.net – Seattle of course) 1.03 ms away. These results suggest that the same DNS server is simultaneously at Los Angeles, Washington, D.C. and Seattle, which apparently can *not* be true.

We therefore conclude that Limelight uses IP anycast to announce their name servers. (Recall that IP anycast is implemented by sending BGP advertisements for the anycast address from multiple, geographically-distributed nodes.) Naturally, the same question rises again – how many real servers Limelight operates for the DNS service. Although there is no direct method allowing us to uncover the mapping from an anycast IP address to actual servers, we conjectured that Limelight will reuse its content servers as DNS servers as well. To verify this conjecture, we send trial DNS queries to the entire list of more than 4,100 content servers discovered previously (simply try to reverse query "1.0.0.127.IN-ADDR.ARPA"), we discover that more than 3,900 content servers are also running as name servers (a much higher percentage than Akamai!).

4.2 Summary of results

In short, we discover ∼4,100 Limelight content servers, among which ∼3,100 are also running as DNS servers.

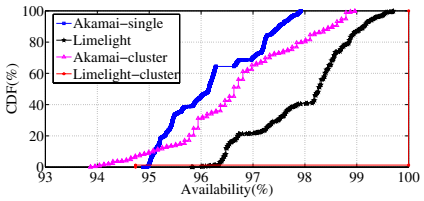
The locations of the Limelight servers are identified precisely (we will elaborate later) and shown in Table 3. In summary, there are 10 in US and 9 in other continents. Note that the single server at Sydney does *not* respond during our later study, so only the other 18 are considered.

City	# of servers	City	Country	# of servers
Washington, D.C.	552	All Cities	United States	2830
Los Angeles	523	Frankfurt	Germany	314
New York	438	London	United Kingdom	300
Chicago	374	Amsterdam	Netherlands	199
San Jose	372	Tokyo	Japan	126
Dallas	195	Toronto	Canada	121
Seattle	151	Paris	France	120
Atlanta	111	Hong Kong	Hong Kong SAR	83
Miami	111	Changi	Singapore	53
Phoenix	3	Sydney	Australis	1
Total	2830	**Total**	-	4147

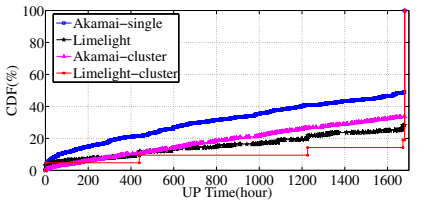
Table 3: Geographic Distributions of the Limelight Network.

5. CDN AVAILABILITY

Having charted out all of the Akamai/Limelight content servers, we can monitor the health of these servers and compare between the two CDNs. We attempt to answer the following question: is the difference in terms of management and maintenance overhead (due to the different CDN design philosophies) reflected and thus can be observed from the status of the servers? To this end, we have continuously monitored all the servers for more than two months (Feb. 15 - April 26, 2008), by connecting to port 80 of each *once every hour*. If a server cannot be connected for 2 consecutive hours, it is treated as *down* (temporary reboots are thus removed).



(a) Availability



(b) Uptime

Figure 3: CDN Availability and Uptime

We compare two characteristics – server availability and continuous uptime.

For the availability evaluation, we determine both *cluster availability* and *server availability*, where "cluster" refers to clusters of CDN servers in the *same location* (the cluster algorithm is discussed in Section 6). There are 1,158 Akamai clusters and 18 Limelight clusters in our evaluation. For each cluster, if it has at least one member up at the measurement time, we consider the cluster available. Thus the cluster availability is the percentage of available clusters among all clusters. The individual server availability is computed by removing unavailable clusters, and then calculating the percentage of available servers from the remaining servers. In this way, we remove possible network failures from the individual server statistics. Figure 3(a) provides the availability distribution. We can see that both Akamai (95%-98% for individual servers, 94%-99% for clusters) and Limelight (96%-99.8% for individual servers, 100% most of the time for clusters) have very good availability. More importantly, Limelight shows higher availability than Akamai, in terms of both individual servers and clusters.

Figure 3(b) shows the uptime results. As we would expect, clusters have much longer uptime than individual servers. In addition, many servers are *always* available throughout our measurement period – more than 50% for Akamai and more than 70% for Limelight. And again, Limelight shows much longer server uptime than Akamai. In summary, both the server availability and uptime seem to suggest that the Akamai network is indeed more difficult to maintain. This should be understandable, as Akamai has many more servers which are distributed in many more locations.

6. EVALUATING DELAY PERFORMANCE

In this section, we evaluate the delay performance of the two CDNs – Akamai and Limelight. We describe how to adapt the popular King approach [8] and then present results from our large-scale study.

6.1 Measurement methodology

6.1.1 The King approach

Because both Akamai and Limelight operate a large number of DNS servers on their regular content distribution servers, we are able to launch DNS-based delay measurement and compare the delay performance of the two CDNs. In particular, we use a variant of the popular King approach, originally described in [8], which is used to measure the delay between an arbitrary open recursive DNS server and another arbitrary DNS server (not necessarily open recursive). King has been widely used in measurement studies; and re-explored several times with new variants (the most recent examples are [10, 18]). The original King approach comes in two versions. Since we use both of them, we briefly summarize each version and point out the main differences. Suppose we desire to measure the delay between the open recursive DNS server D and a specific DNS server T.

In the first version, a measurement client M sends a DNS query to D to resolve a bogus name claimed in the authority of T. Since D is an open recursive DNS server, it will try to resolve this name on behalf of M, (and, of course, get a negative response from T). If all goes well, the RTT between D and T is simply given by the RTT from M to T (via D) minus the RTT between M and D, The simplicity of this version comes with a cost of accuracy – there is no assurance that D will forward the query to T, rather than to some other DNS server (say T') also in charge of the same domain. This might not be an issue at all if there are only a few Ts and they are all co-located. However, one needs to be extremely careful, as there could be hundreds (or even thousands) of Ts for the same domain, and to further complicate things, if these Ts are geographically widely distributed. As will become clear, this is indeed the case we are dealing with here.

Fortunately, the second version of the King approach suggests a clever way to solve this problem. As illustrated in Figure 4, one registers a domain (call it *cdn.net*) and operates a DNS server (say S) to respond to queries for this domain. During the measurement process, M sends a DNS query to D resolving a *domain name* T.cdn.net. This query will cause D to contact S, because S is in charge of cdn.net. When S receives such a query, it does *not* respond with an IP address, but instead with a referral, claiming that the subdomain T.cdn.net is delegated to another name server (say ns.T.cdn.net) and the IP address of that name server is that of T. D forwards this response back to M and this round of DNS query is completed. There is no measurement yet, but this process causes the result

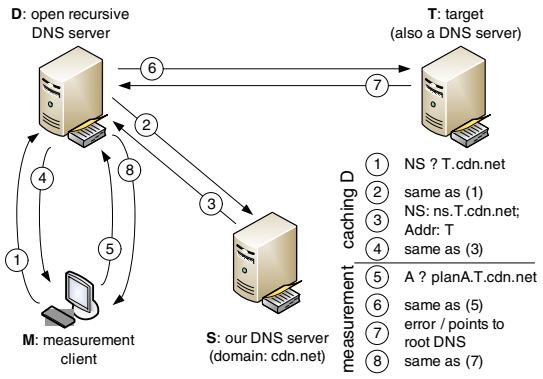


Figure 4: Measuring Delay using DNS Servers (the King approach – 2nd version)

(the authoritative name server for domain T.cdn.net is ns.T.cdn.net with address T) to be cached at D. Next, M sends another query to D to resolve a *hostname* in T.cdn.net (say planA.T.cdn.net) shortly after. At this time, D does *not* contact S again, but instead forwards the query to T directly. Obviously, T will respond with an error message, which is then sent back to M. It should now become clear this second query round includes an exactly one RTT between D and T (and not to some other T'). The sub-domains under cdn.net need to be randomized in order to combat undesirable caching at Ds (i.e., use random-T.cdn.net instead of T.cdn.net). This together with a number of other details are covered in the original King paper [8].

6.1.2 Measuring delay of CDNs

Now we are ready to describe how we can measure the delay performance of CDNs using the above approach. Our goal is to measure the delay from each of the vantage point D (that is, from each open recursive DNS server) to the CDN, that is, to the content server that the CDN redirects D to. For the "content server" we choose the CNAME resolved to the largest number of CDN servers globally, namely, a1105.b.akamai.net (used by BestBuy) for Akamai and move.llnwd.net (used by Move Networks) for Limelight. Call this CNAME N. For each vantage point D, our measurement client M first sends a DNS query to resolve N and obtains the IP address that the CDN content server used to serve D; let's call this content server C. We now need to measure the delay between D and C. If C, in addition to being a content server, is a DNS server, then we would simply use King to get the delay between the open recursive server D and the DNS server C. More generally, we have to find a content server T in the same cluster as C which *happens* to also operate as a DNS server. As long as we can ensure that T is in the same cluster as C, the delay between D and T using King will be equivalent to the delay between D and C (which is what we ultimately want). Here, the *same cluster* represents all the servers in the *same location*. Of course, to find T accurately, we need to be extremely careful to group CDN servers into the same cluster, which we elaborate on in the next two subsections.

6.2 Clustering Akamai servers

We desire high fidelity when clustering the Akamai servers together. Our main tool here is traceroute – most Akamai servers can be reached via default UDP-based traceroute and the rest can be reached via TCP-based traceroute at port 80.

Based on a key observation that many Akamai servers are deployed inside ISP POPs, which are naturally *very close* to one of the ISP routers, we use undns [9] to uncover the location of nearby ISP routers and then use that location for Akamai server. Note that we only consider routers within s (ms) from Akamai servers. In addition, we design a *divide and conquer* method to speed up the clustering process, which is based on earlier observations that Akamai tends to occupy large blocks of consecutive IP addresses. We refer interested readers to [6] for the details of the clustering algorithm and only share one lesson learned during the clustering process – it helps tremendously to launch traceroute from multiple PlanetLab nodes. When traceroute was only launched from one single location (our initial deployment), we were baffled by quite a number of IP addresses, where the last hop router before the target is often times very far (e.g., 30 ms or more) away from the target. However, when we launch traceroute from more than 180 PL nodes, we are always able to locate a last hop router within $s$ms away from any Akamai target. Using Global Crossing (gblx.net) as an example, a traceroute to an Akamai target actually enters this ISP's network at a point very close to the traceroute source, which then travels a long way during the last hop to the Akamai target located far distantly. Tracing from hundreds of locations essentially allows us to start from a point which is close enough to the target and thus avoid the long hop inside the ISP.

During our experiment, we choose s to be 1.5ms and have clustered around $26,500$ (a small number of servers are down during the clustering) Akamai IP addresses ($1,122$ /24 prefixes) into $1,158$ clusters. The average delay between the last hop router used in clustering and corresponding Akamai target is 1.07ms. It is important to point out that a crude clustering rule, such as /24 prefix clustering, is far from accurate. For example, under the same prefix 81.52.248, one block of Akamai servers is located inside Level3 (AS3356) at Chicago, while another block is located inside Open-Transit (AS5511) at Dallas, and yet another block is located at University of Connecticut at Storrs.

We acknowledge that the clustering might not be perfect. For instance, 1ms between the ISP router and the Akamai server could still mean a long physical distance.

Additionally, ISP misnaming [17] (although quite rare) could potentially cause us to wrongly cluster some Akamai servers together. Hence, on top of the above clustering, during our later measurement studies, whenever we need to choose another Akamai server close to a given one, we not only confine ourselves to the same cluster, but always choose an IP address with the smallest distance in IP space. Finally, due to the incompleteness of undns (even including two additional rule sets from [15, 16]), we are only able to identify exact locations of around 17,000 Akamai servers (about 2/3 of the complete Akamai network). The rest 9,000 can only be clustered, but not located.

6.3 Clustering Limelight servers

It turns out that clustering Limelight servers is a much easier task. Limelight is generous enough to provide reverse DNS mappings for all of its IP addresses. For example, 87.248.208.38 is reversely mapped to cds28.lon.llnw.net, which indicates that the server is located in London (LON).

6.4 CDN delay performance

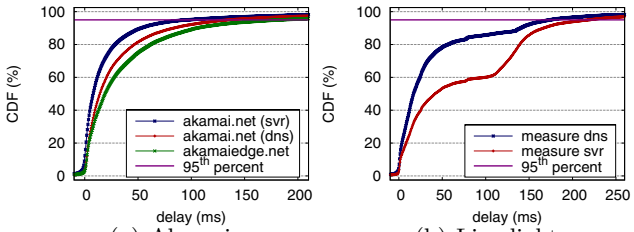
Now we are ready to evaluate the delay performance and compare between the two CDNs. For both the CDNs, we measure from all of the vantage points. We measure both the delay to its DNS system, as well as to actual content servers.

6.4.1 The Akamai network

To measure Akamai's private DNS system, we are only interested in delays at the second tier. Hence, we need to send a query to seed the vantage points (Ds) so that it does *not* need to contact the root level Akamai name servers during actual measurements. To be specific, our measurement client M first sends a query to D to resolve a popular CNAME (e.g., a1105.b.akamai.net). Next, it sends another query to D to resolve an artificial CNAME under the same sub-domain (say random.b.akamai.net). This triggers D to contact one of Akamai's second level DNS servers and eventually M gets an error response. Subtracting the delay between M and D from the total time gives the delay from D to Akamai's DNS system.

To measure delays to actual content servers, M first sends a query to D resolve a1105.b.akamai.net. When it receives the IP addresses of the content servers, which Akamai uses to serve D, it chooses another Akamai server in the same cluster (also closest in IP space, to be specific) that is also running as a name server. M then follows the approach detailed in 6.1 to measure the delay between D and the content server.

Now we report the delay performance of the Akamai network, as shown in Figure 5(a). Using common industry standard, we focus on the 95^{th} percentile of the delay distribution.



(a) Akamai (b) Limelight

Figure 5: Delay Performance

targets	delay (ms)
akamai.net svr	103.38
akamai.net dns	138.35
akamaiedge.net dns	189.29

We make the following observations: 1) Using akamai.net as an example, the results show that Akamai's second level name servers are reasonably close to clients, but *not* as close as its content servers. This is observed by the comparison between the 95^{th} percentile delays to its DNS system (138.35ms) and its content servers (103.28ms). 2) Delays to the DNS system of akamaiedge.net are significantly higher than those to akamai.net, shown by the comparison between delays to the two DNS systems – 138.35ms for akamai.net vs. 189.29ms for akamaiedge.net. This should be understandable as the akamaiedge.net network (again, using virtualization to deliver dynamic content) is likely available in far fewer locations than the akamai.net network (primarily for static content).

6.4.2 The Limelight network

Limelight has a flat DNS structure, so it's straightforward to apply the first version of King to measure delays to its DNS system. In short, M sends queries of a popular CNAME (e.g., move.vo.llnwd.net) to seed the vantage points and then sends artificial queries to measure the delay. Delays to its content servers are measured in the same way as in the Akamai case, following the approach detailed in 6.1.

targets	delay (ms)
limelight dns	170.22
limelight svr	221.74

We make the following observations: 1) Delays to Limelight content servers are actually much higher than to its DNS system. This suggests Limelight is actively redirecting clients to non-nearest locations (although *not* too far away), perhaps to optimize its peering costs with various ISPs. 2) Delays to the Limelight network are a lot higher than to the Akamai network as well – the DNS resolution delay of Limelight is 23% higher than Akamai (170ms vs. 138ms); and the content server delay of Limelight is 114% higher than Akamai (222ms vs. 103ms). The delay comparison between Akamai and Limelight at various levels are summarized in Table 4.

	akamai.net		akamaiedge.net	limelight	
delay (ms)	DNS	server	DNS	DNS	server
avg.	40.33	28.75	52.15	43.91	81.86
50%	15.01	8.92	20.58	17.13	43.15
90%	85.40	55.76	107.79	138.50	168.02
95%	138.35	103.38	189.29	170.22	221.74

Table 4: Delay Comparison between Akamai and Limelight

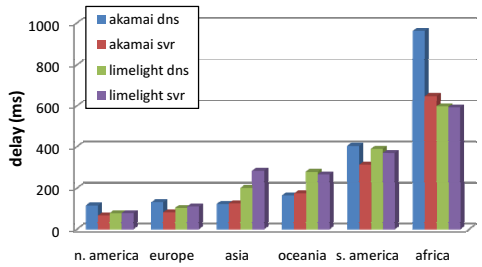


Figure 6: Delay Performance Breakdown (by continent)

6.4.3 *Performance breakdown*

So far, we've observed that there is a big gap in delay performance between Akamai and Limelight. In this subsection, we break down the performance by geographic regions and examine them individually. In particular, we separate vantage points by their continent and plot corresponding delay performance (at 95% percentile) in Figure 6 and Table 5.

We observe that the delay performance has a positive correlation with the coverage of each CDN in a specific target geographic region. Using delays to content servers as an example, Akamai and Limelight both have good and comparable performance in North America, where both CDNs have a good coverage (even Limelight has 10 data centers). To be specific, the delay to Limelight server is only 18% more than Akamai (79ms vs. 67ms). However, moving to Europe, where Limelight has fewer data centers (4 in fact), its delay is over 33% more than Akamai (110ms vs. 83ms). The gap is more prominent in Asia (284ms vs. 126ms, or 127% worse), where Limelight merely has 3 data centers to cover a much bigger geographic area. Similarly, a even bigger gap should *not* be a surprise in Oceania, South America and Africa, given that Limelight does *not* even have presence in these regions. More over, it's worth pointing out that the delay performance of Akamai also degrades significantly, entering into less covered regions (e.g., South America and Africa).

7. EVALUATING DATA CENTER DEPLOYMENT

7.1 Do additional data centers help?

In this section, we consider the marginal gain in performance when a CDN adds one or more data centers.

	Akamai		Limelight	
delay (ms)	DNS	server	DNS	server
North America	115.81	67.24	78.64	79.03
Europe	131.08	82.54	103.34	110.05
Asia	122.5	125.53	199.84	284.4
Oceania	163.68	173.17	279.12	266.66
South America	402.35	312.52	388.17	368.66
Africa	961.03	647.08	596.03	591.45

Table 5: Details of Delay Performance Breakdown

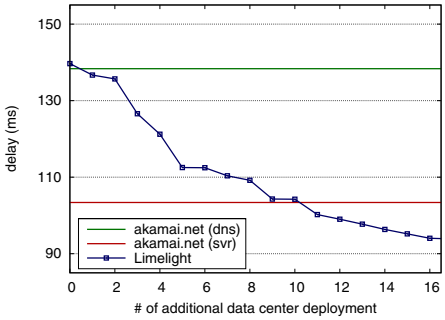


Figure 7: Performance w/ Additional Data Centers

In particular, we provide a measurement methodology that lets one CDN exploit another CDN's deployment to assess the marginal gain of additional data centers. As a case study, we will consider how Limelight can learn from Akamai's current deployment and choose good additional locations.

In this methodology, from each vantage point, we simultaneously measure the delays to Akamai's content servers and to all of the Limelight locations. Then, we design the following simple greedy algorithm to pick good next locations to improve Limelight's delay performance. The algorithm works as follows. We go through each Akamai location and evaluate the delay reduction if Limelight deploys *just* one more data center at that location. We then rank all the Akamai locations and choose the best one to add to the list of existing Limelight data centers (say L). This gives us the first additional location. Then, we repeat the same process to identify the second additional location, with first additional location treated as part of Limelight. We repeat this process to obtain the third additional location and so on. During the process of evaluating a particular vantage point D, we first determine the location from where Akamai serves D. If the location is in the list of Limelight data centers L and the delay to Akamai is less than to Limelight, we take the delay to Akamai. Otherwise, we take the delay from the best Limelight location. Figure 7 shows the delay improvement (at 95^{th} percentile) with the progressive deployment of more data centers.

We make the following observations:

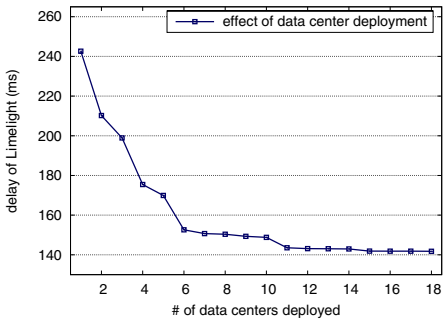


Figure 8: Effectiveness of Data Center Deployment

1. When no additional data center is deployed and we always route to the closest of the 18 Limelight data centers, the delay is 142ms. This value is far better than when redirection is used from Limelight's DNS system (222ms). Hence, if Limelight starts to focus on delay performance, it could make significant gains even without deploying additional data centers.

2. By learning from Akamai's existing deployment, a few more data centers can dramatically reduce the delay and make Limelight on par with Akamai. For instance, 5 more data centers can reduce the delay to about 10% of Akamai and 9 more is enough to match Akamai. This should be surprisingly encouraging for Limelight!

3. The choices made by our algorithm make intuitive sense. For instance, the top two suggestions are Taipei (Taiwan) and Seoul (Korean). Given the limited presence of Limelight in Asia (only Tokyo, Singapore and Hong Kong at the moment), such choices appear quite natural to Limelight. Note that, when comparing locations, we choose to rank them by the *total* delay reduction across all vantage points (instead of the 95% percentile). This choice is most likely the reason why the delay reduction does *not* follow a convex curve. Another reason might be because our algorithm is a heuristic. Finally, it should *not* be surprising that with more than 10 additional data centers, Limelight's delay can actually be better than Akamai's. As observed before [11], Akamai does *not* always redirect clients to the *closest* location, but rather a *good enough* one. Hence, it's natural that best delays to Limelight (with additional data centers) could surpass good enough delays to Akamai.

7.2 Which locations matter?

We also use measurements to evaluate the deployment of Limelight's current data centers. We ask the question: are the current locations good choice, and which of the 18 locations really matter?

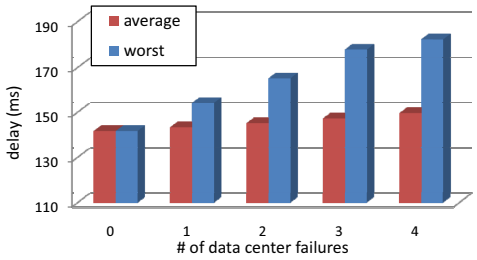


Figure 9: Performance w/ Data Center Failures

With simultaneous delay measurements to all 18 Limelight data centers, we first determine where the data center should be located if Limelight were only to deploy one single data center; and where the additional data centers should be deployed to improve the delay performance. It turns out that Chicago is on the top of the list. Moreover, with only 5 additional data centers (in ranked order: Tokyo, Frankfurt, San Jose, Washington D.C. and Singapore), Limelight can do almost as good as with its current 18 data centers. Clearly, other data centers are most likely deployed so that there are peering relationships to more ISPs, instead of solely from a delay perspective.

7.3 Robustness of data center deployment

One of the key promises of CDNs is to be able to cope with failures. High profile failures not only hurts business temporarily, but also damages customers' confidence in the long run. Compared to Akamai's scattered deployment, Limelight is likely to be more prone to cluster failures. In this subsection, we evaluate the robustness of a Limelight-like deployment.

Using Limelight's current data centers, we introduce artificial failures and study the impact on the delay performance. Specifically, we measurement delays from each vantage point to all 18 Limelight data centers simultaneously. We assume Limelight could engineer its global traffic management perfectly, which implies traffic can always be routed to the nearest (in delay terms) data center, even with the presence of failures. We artificially fail one or several data center and use the delay to the nearest remaining data center as the delay from the vantage point. Given the number of data center failures, we enumerate all the cases (e.g., there are 18 cases for 1 data center failure, $\binom{18}{2} = 153$ cases for 2 failures, and so on), and calculate the 95% percentile delay from all the vantage points. We report the delay averaged over all the cases, as well as the worst case.

From Figure 9, we conclude that the current deployment of Limelight is very robust against data center failures. One data center failure has very minimum performance impact, in both average and worst case. Even when there are up to 4 data center failures, the average delay performance only increases 5.5% from 142ms to 150ms. It is true that the delay increases more for

the worse case failures. For example, when there are up to 4 data center failures, the worst delay increases 28.2%, from 142ms to 182ms. Nevertheless, this particular failure case would happen only if all 3 Limelight data centers in Asia (Tokyo, Singapore and Hong Kong) are wiped out, plus an additional data center in San Jose down at the same time. Failures as such are very unlikely.

8. EVALUATING IP ANYCAST

We learned in Section 4 that Limelight performs its DNS resolution in the same 18 data centers that house its content servers (and in fact using the same hosts for content servers). We also saw that Limelight uses IP anycast to route a DNS query to a nearby data center. We now ask, just how good is the delay performance of Limelight's IP anycast technology? Does Limelight succeed at directing most queries to the closest center? Can its DNS-resolution delay performance be significantly improved, resulting in a lower overall delay, potentially allowing it to enter more lucrative markets of delivering dynamic content (given the increasingly fierce competition and continuing commoditization of static content delivery)?

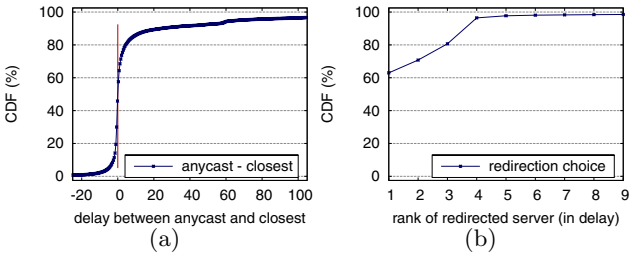


Figure 10: (a) DNS delay of Limelight's IP anycast. (b) How often Limelight redirects clients to non-nearest locations?

It is also interesting to evaluate anycast system as a study of IP anycast performance in its own right. Different from anycast systems evaluated in previous studies [18], the Limelight IP anycast system operates thousands of DNS servers, located in 18 widely distributed geographical locations. Moreover, Limelight has extensive peering relationships with a large number of ISPs.

By comparing the DNS delay with the *best* delay to the 18 data centers, we can evaluate whether IP anycast *always* routes requests to the closest DNS server. To ensure a fair comparison, for each vantage point, we measure these two delays at the same time. Figure 10(a) shows the delay gap between the DNS system and the best location is 25ms at 90^{th} percentile and 68ms at 95^{th} percentile. Compared to other commercial IP anycast deployments [18], one can conclude from these results that the Limelight IP anycast works *very* well.

Additionally, from each vantage point, we rank all the Limelight data centers based on the respective measured delay and evaluate how often Limelight redirects clients to locations not closest. From the result in Figure 10(b), it is clear that Limelight does *not* redirect clients to the closest server more than 37% of the time, and the redirection is beyond the top three locations about 20% of the time. (We treat two locations equal if delays to them differ less than 10ms.) This is most likely because that Limelight focuses more on optimizing peering costs for delivering large objects (e.g., streaming videos) rather than delay performance.

9. RELATED WORK

Although there have been many studies on Content Distribution Network [23–26] and this is generally considered a well-understood topic. Very little attention has been paid to the recent success of the alternative design philosophy – the one Limelight has championed – and its implications. To our best knowledge, this paper is the first effort to make a head-to-head comparison between difference CDN design philosophies and shed light on this matter.

The performance of CDNs has been of interest almost since their inception [11–14]. Johnson *et al.* [11] provide one of the earliest studies of the performance of Akamai and Digital Island, where a key observation is that rather than always redirect clients to the best (in terms of delay) content servers, CDNs often satisfy with a good enough one. The study is limited by having *no* relative comparisons between the CDNs, as well as using only 3 evaluation clients (vantage points). Krishnamurthy *et al.* [12] conduct a detailed study of the usage and download time for 7 CDNs that were popular circa 2001. They employ 20+ clients in the measurement. Compared to these earlier works, which made informative but rather limited scale probes into CDNs, our study aims at charting the CDNs to their completeness, as well as measuring their performance at a much larger scale (several orders of magnitude more measurement clients). Furthermore, we explore the idea that how one CDN can leverage the existing infrastructure of another CDN to refine its deployment. We relate our measurement study to data center deployment, a problem increasingly attracting more attention beyond CDNs, as many large Internet content and search companies are also rolling out their own private data centers.

Our data center placement problem shares similarities with early replica placement studies [20–22]. An CDN with operational data can easily borrow techniques from those studies. For example, it can rank all the locations based on the amount of traffic originated [21] and select top M as candidates. Most likely Akamai already has deployment in these M locations. Thus, the CDN can use our techniques to measure delay per-

formance from each vantage point to all the candidate locations. Finally, it can select the best location greedily [20], and then the next best location, and so on.

In our measurement study, we extensively employ King [8], which is widely used today in Internet measurements. Ballani *et al.* [18] conduct extensive evaluations of a number of deployed IP anycast systems. Their results show that *ad-hoc* deployment of IP anycast often does *not* route requests to the closest server. Further, they conjecture and validate that a good IP anycast deployment should be geographically distributed and preferably within a single ISP. However, they were not able to evaluate with their methodology scenarios beyond that. The Limelight IP anycast system is rather different in that Limelight is not a customer of any single ISP, but rather it peers extensively with many ISPs. Conceptually, IP anycast requests should be quickly routed to the closest Limelight data center. Hence, IP anycast should work very well, as we verified. Moreover, our measurement clients are about 4 times more than in [18], giving us much better coverage.

Mao et al. [28] conducted a large scale measurement study and quantify the closeness of clients and their LDNSs. They observe a large percent of the clients are in fact *not* close to their LDNSs (in the sense of AS, network aware clustering, network hop, etc.). Hence, a DNS-based redirection might *not* be very effective after all. We note that their problem is orthogonal to the issues we consider in this paper. Furthermore, when it is desirable and the DNS-based redirection is off by too much, CDNs can always instruct content servers to redirect clients again to much closer servers. During the recent study on dark DNS behavior, Oberheide et al. [27] made an interesting discovery and conjectured that Akamai is very likely to be using King-like DNS-based measurements as part of its global measurement infrastructure. Its own participation in leveraging public DNS infrastructure for measurement probably makes it more receptive to the type of measurements required by the methodologies discussed here.

Our method of clustering Akamai servers uses traceroute and reverse DNS names of ISP routers, a similar idea as GeoTrack [19]. However, with the assistance of a large number of PlanetLab nodes and also due to the fact that Akamai servers are usually in ISP POPs, our clustering should provide much better accuracy.

10. CONCLUSIONS

Using two CDNs, Akamai and Limelight, as examples, we have presented methods to chart CDNs, to measure their performance, and to guide CDN design decisions, e.g., whether to use IP anycast and where to deploy in next geographic location. We have discovered ∼27,000 content servers for Akamai (including ∼6,000 name servers) in 65 countries, and ∼4,100 content servers for Limelight (including ∼3,900 name servers) in 18 locations. Using an enhanced King approach launched from a large number of vantage points, we have quantified the delay performance perceived by Akamai and Limelight customers worldwide, for both DNS resolution and content serving. We have also evaluated a large-scale IP anycast deployment (used in Limelight's DNS system), and developed a methodology for locating additional content servers in a CDN.

Note that, as CDNs constantly evolve, charting and performance measurement can complement each other to maintain a complete and up-to-date view of the CDNs. A complete chart, requiring a large number of DNS queries, can be done once every few months. Between charting, the performance measurements, requiring less overhead, can be repeated more frequently. New CDN servers/locations discovered via such measurements can be verified and expanded using our method to augment the charting results.

Throughput is also an important metric in evaluating CDN performance. However, due to the nature of our approach – leveraging DNS infrastructure for delay performance – there is *no* direct way to extend the methodology to throughput measurements. Therefore, we acknowledge that this paper is limited to delay performance only.

All the vantage points are treated equally in this paper. This is *not* ideal, as the weight of a vantage point should be proportional to the amount of client traffic from its location. (Although the distribution of the vantage points themselves *does* reflect certain popularity – for instance, developed countries have higher number of vantage points – this is far from ideal.) Nevertheless, we do *not* regard this as a technical challenge – a CDN operator (or any third party) with a large amount of client information can easily incorporate traffic data into the methodologies.

APPENDIX

Uncovering service behind akamaiedge.net

We conjecture type (c) (akamaiedge.net) provides a different service from type (a) or (b). First, every query for such a CNAME from a particular vantage point only returns one IP address (no matter which vantage point handled the query). Second, for a given CNAME, the number of IP addresses it resolves to across all the vantage points is typically in the 20-100 range. For example, e128.b.akamaiedge.net resolves to 74 unique IP addresses in total. This is quite different from hundreds (or thousands) of IP addresses seen with type (a) CNAMEs, and also different from only a few seen with type (b) CNAMEs. Third, when we combine all the IP addresses discovered (from the 539 akamaiedge CNAMEs and from our vantage points), we obtain more than 36,000 unique IP addresses, and this set of ad-

dresses is completely disjoint from the 11,500 IP addresses found for type (a) CNAMEs.

Now we are ready to uncover the mysterious akamaiedge.net network. We observe that akamaiedge.net also uses a two-level DNS infrastructure. Among 900 DNS servers handling the second level domains, about 200 have already appeared in the previous expanded list of content servers from akamai.net. Note that all of these 200 servers appear among the *expanded* part, not from the original discovered list of content servers. By expanding and verifying the remaining 700 DNS servers, we discover another 3,000 new content servers. The IP addresses of these new content servers share long prefixes with those ∼23,000 content servers discovered previously. This strongly suggests that the content servers from akamaiedge.net are very likely co-located with those from akamai.net. Hence, putting them together, we compile a list of more than 27,000 content servers, which we conclude is the *complete* Akamai global network.

However, this has yet to explain the more than 36,000 IP addresses discovered from akamaiedge.net CNAMEs. As a matter of fact, we are able to not only confirm that almost all of the 36,000 IP addresses belong to Akamai, but also further expand the list to include more than 150,000 IP addresses. Apparently, Akamai is behind these IP addresses, but the numbers don't add up. Where is the missing walrus?

Finally, two additional key observations lead us to a reasonable explanation. First, very different from the IP addresses obtained from akamai.net (type (a) service), where multiple Akamai CNAMEs can map to the same IP address, the addresses obtained from akamaiedge.net (type (c) service) are all exclusive. Second, very different from the total 27,000 IP addresses, where the failure rate is always about 4-6%, IP addresses from akamaiedge.net have much lower failure rate (only about 1%). Based on this mounting evidence, we conjecture that Akamai is using virtualization technology behind all the IP addresses for akamaiedge.net. Furthermore, Akamai is providing an isolated virtualization environment to customers for dynamic content distribution. Conceivably, virtualization is available only in limited locations among all the Akamai deployments. (Please refer to [6] for more details.)

REFERENCES

[1] H. Dewing, E. Daley, R. Whiteley, and A. Lawson, "Inquiry Insights: Content Delivery Networks, Q4 2007," Forrester Research, Jan. 18, 2008.

[2] D. Rayburn, "CDN Market Getting Crowded: Now Tracking 28 Providers In The Industry," *Business of Online Video Blog*, Sep. 24^{th}, 2007.

[3] M. J. Freedman, E. Freudenthal, and D. Mazires, "Democratizing Content Publication with Coral," *USENIX NSDI*, Mar. 2004.

[4] L. Wang, K. Park, R. Pang, V. S. Pai, and L. Peterson, "Reliability and Security in the CoDeeN Content Distribution Network," *USENIX Annual Technical Conference*, Jun. 2004.

[5] "Akamai Global Traffic Management," http://www.-akamai.com/html/technology/products/gtm.html.

[6] C. Huang, A. Wang, J. Li, and K. W. Ross, "Measuring and Evaluating Large-Scale CDNs," *Microsoft Research Technical Report MSR-TR-2008-106*, Aug. 2008.

[7] "Top 21 U.S. ISPs by Subscriber: Q2 2007," http://www.isp-planet.com/research/rankings/usa.html.

[8] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating Latency between Arbitrary Internet End Hosts," *ACM IMW*, Nov. 2002.

[9] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP Topologies with Rocketfuel," *IEEE/ACM ToN*, 12(1), Feb. 2004.

[10] D. Leonard, and D. Loguinov, "Turbo King: Framework for Large-Scale Internet Delay Measurements," *IEEE INFOCOM*, Apr. 2008.

[11] K. L. Johnson, J. F. Carr, M. S. Day, and M. F. Kaashoek, "The Measured Performance of Content Distribution Networks," *Computer Communications*, 24(2), Feb. 2000.

[12] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the Use and Performance of Content Distribution Networks," *ACM IMC*, Nov. 2001.

[13] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An Analysis of Internet Content Delivery Systems," *USENIX OSDI*, 2002.

[14] A.-J. Su, D. Choffnes, A. Kuzmanovic, and F. Bustamante, "Drafting Behind Akamai (Travelocity Based Detouring)," *ACM SIGCOMM*, Sep. 2006.

[15] M. Freedman, M. Vutukuru, N. Feamster, and H. Balakrishnan, "Geographic Locality of IP Prefixes," *ACM IMC*, Oct. 2005.

[16] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: an Information Plane for Distributed Services," *USENIX OSDI*, Nov. 2006.

[17] M. Zhang, Y. Ruan, V. Pai, and J. Rexford, "How DNS Misnaming Distorts Internet Topology Mapping," *USENIX Annual Technical Conference*, Jun. 2006

[18] H. Ballani, P. Francis, and S. Ratnasamy, "A Measurementbased Deployment Proposal for IP Anycast," *ACM IMC*, Oct. 2006.

[19] V. N. Padmanabhan, and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," *ACM SIGCOMM*, Aug. 2001.

[20] P. Krishnan, D. Raz, and Y. Shavitt, "The Cache Location Problem," *IEEE/ACM ToN*, 2000.

[21] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the Placement of Web Server Replicas," *IEEE INFOCOM*, 2001.

[22] S. Jamin, C. Jin, A. R. Kure, D. Raz, and Y. Shavitt, "Constrained Mirror Placement on the Internet," *IEEE INFOCOM*, 2001.

[23] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally Distributed Content Delivery," *IEEE Internet Computing*, 2002.

[24] A. Vakali, and G. Pallis, "Content Delivery Networks: Status and Trends," *IEEE Internet Computing*, 2003.

[25] G. Peng, "CDN: Content Distribution Network," *Tech Report*, SUNY Stony Brook, 2003.

[26] A. K. Pathan, and R. Buyya, "A Taxonomy and Survey of Content Delivery Networks," *Tech Report*, Univ. of Melbourne, 2007.

[27] J. Oberheide, M. Karir, and Z. M. Mao, "Characterizing Dark DNS Behavior," *Proc. DIMVA*, 2007.

[28] Z. M. Mao, C. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang, "A Precise and Efficient Evaluation of the Proximity between Web Clients and their Local DNS Servers," *USENIX Annual Technical Conference*, 2002.