Magnus Boye

Measuring Websites from a Global Perspective

School of Electrical Engineering

Thesis submitted for examination for the degree of
Master of Science in Technology.
Espoo, 17.01.2014

Thesis supervisor:

Professor Jörg Ott

Thesis advisor:

Pasi Sarolahti, PhD

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Magnus Boye

Title: Measuring Websites from a Global Perspective

Date: 17.01.2014 Language: English Number of pages: 4+86

Department of Communications and Networking

Professorship: Networking Technology Code: S-38

Supervisor: Professor Jörg Ott

Advisor: Pasi Sarolahti, PhD

Since the invention of the World Wide Web the content and services provided on the web have changed significantly. In search of cost savings governments and businesses push online services, and the web has therefore become ever more important to many people. It is therefore important to understand the performance of web page delivery, in order to improve the user experience of the web.

The goal of this Master's thesis is to evaluate the network performance of web page delivery in relation to content distribution networks. We use the global measurement platform PlanetLab to perform active measurements of the performance of DNS and HTTP when downloading web pages. Domain names are resolved using iterative resolution, Google DNS, OpenDNS, and the default DNS server of measurement nodes. This enables us to asses how the choice of DNS resolver affects CDN server selection. The measurements reveal that network latency has the greatest impact on DNS resolution time and that nearby DNS servers will generally have the lowest resolution time. We developed an effective method of identifying CDNs and applied it to the measurement data. We analyze the accuracy of the DNS resolvers and determine that the use of a recursive DNS server close to the end-user typically results in server selections more similar to the optimal server selection of a CDN. However, we did not establish a relationship between resolver accuracy and HTTP performance. Finally, we compare the throughput of CDNs and observe that CDNs are likely optimized for different file sizes.

Keywords: Active measurements, DNS, HTTP, CDN, Websites

Acknowledgements

I would like to thank Professor Jörg Ott and Pasi Sarolahti for their insightful comments and guidance during my work with this Master's thesis.

The work was supported by the European Commission's Seventh Framework Program (FP7/2007-2013) grant number 317647 (Leone).

Espoo, 17.01.2014

Magnus Boye

Contents

1 Introduction **1**
1.1 Active and Passive Measurements 2
1.2 Domain Name System 3
1.3 Problem Statement 4

2 Background **7**
2.1 Domain Name System 7
2.1.1 Message and Record Formats 9
2.1.2 Resource Record Types 11
2.1.3 Protocol 11
2.1.4 Issues and Vulnerabilities 13
2.1.5 DNS Security Extensions 14
2.1.6 Third-Party Resolvers 15
2.2 Web Pages and HTTP 16
2.3 Content Distribution Networks 20
2.3.1 Anycast 21
2.3.2 Load Balancing 22
2.3.3 Application Layer Redirection 23
2.3.4 DNS Redirection 23
2.3.5 CDN Providers 28

3 Measurements **31**
3.1 Metrics 31
3.2 Measurement Application 34
3.3 PlanetLab 38
3.4 Measurement Procedure 39

4 Analysis **43**
4.1 Measurement Data 43
4.2 DNS and HTTP Performance 46
4.2.1 Resolution Time 46
4.2.2 Record TTL 50
4.2.3 Connect Time 53

4.3 Content Distribution Networks 55
4.3.1 Identifying Content Distribution Networks 55
4.3.2 CDN Servers 58
4.3.3 Resolver Accuracy 63
4.3.4 Connect Time 67
4.3.5 Throughput 69

5 Discussion **73**

6 Conclusions **77**

Chapter 1

Introduction

With the invention of the World Wide Web (WWW), the rapid growth of the Internet began. The "web" provided an easy platform for communication and publishing content, and two decades later it is still the primary method for accessing online content and services. Over the years the content and services offered have changed and nowadays services like Google Docs challenge the need for traditional desktop applications by moving all functionality into the browser, and the data into "the cloud". The content of web pages have changed from being mostly static text and images, to offering highly dynamic and interactive interfaces with much richer graphical elements including video. It can be argued that the changes in content have been made possible by advances in network technologies and that these advances have been driven by the increasing Internet traffic. The increasing importance of online services on the web have also created a need for better ways of evaluating the quality of such services. The concept of quality can cover multiple aspects of a service depending on the perspective, but this Master's thesis focuses on the network-related aspects of service quality, as these are quantified automatically more easily.

Three parties are involved when Internet services are delivered: the consumer, the internet service provider (ISP), and the content provider. The consumers are paying for Internet access and potentially content, and thus have an interest in making sure that the purchased service performs as expected. The ISPs and content providers have a financial incentive to keep their customers happy. Additionally, content providers may be hosting servers inside ISP networks and therefore wish to know how the network of an ISP performs.

Although video streaming traffic accounted for 64% of Internet traffic in 2012 [1], web pages remain an important part of the Internet due to the services provided on web pages. This makes the performance of web page delivery an important area of research as it affects practically all Internet users.

1.1 Active and Passive Measurements

Content providers can monitor the performance of their services with little effort, but only at point of which traffic enters the Internet. However, since the Internet is a best-effort service, there is no guarantee that good performance at the server side is equal to good performance at the client side.

The first step of delivering a good experience for a service provider, is to ensure adequate capacity on the server side, such that service does not deteriorate due to overload. Evaluating service performance at the client-side can either be done actively or passively, and each method has its advantages and disadvantages. Passive measurements can be used to monitor one or more end-users, depending on the link where traffic is captured. The main benefit of passive measurements is that the performance of *real* Internet traffic is measured, and that no additional traffic is injected into the network which would distort measurement results. Passive measurements are useful when the goal is to study general tendencies of real traffic, because traffic is generated by actual users. This of course requires users to use a specific service in order for its performance to be measured. Passive measurements can deliver large amounts of data, but has privacy issues since sensitive user data may be captured. For this reason, passive measurements require legal paperwork and adherence to data protection laws in the country of measurement.

Active measurements have fewer privacy issues because all measurement traffic is artificially generated and therefore does not contain sensitive information. The main privacy concern of active measurements is the origin of measurements, e.g. source IP address and geographical location, but this information can easily be anonymized. If the origin of measurements were not anonymized it would be possible link individuals or households with e.g. websites visited. The key benefit of active measurements is greater control in terms of which services are measured and the behavior of measurement software. Another benefit of active measurements is that they can be used for network monitoring by taking measurements with regular intervals. Additionally,

measurement applications could be used to diagnose network problems by triggering them when failures occur. The disadvantage of active measurements is that they compete with existing traffic on a network and that the measurements change the traffic profile of a network. The effects of cross-traffic can be reduced by monitoring the local network passively for traffic, but this may be problematic from a privacy point of view. If a software-based measurement agent is used, cross-traffic detection may be limited to the host running the software and therefore not be able to detect traffic generated by other devices on the local network. For hardware-based measurement agents, cross-traffic detection is complicated by the local area network configuration and the presence of both wired and wireless interfaces. An accurate way of determining cross-traffic is to pass all traffic through the measurement agent operating as a switch, and simply count the number of bytes processed by the network interfaces without ever inspecting traffic. From a performance perspective this is not the best solution and the setup requires end-users to connect the measurement agent correctly.

The goal of this Master's thesis is to evaluate the DNS and HTTP performance of web sites. In order to analyze content distribution networks the measurements will have to be carried out at many geographical locations. Performing passive measurements in many locations would have taken a great deal of time to orchestrate and would have complicated the performance comparison for specific services. For these reasons we decided to perform active measurements as this allowed specific services to be measured in the same way across all measurement locations.

1.2 Domain Name System

The Domain Name System (DNS) is an integral part of the Internet and is often referred to as the phone book of the Internet. DNS is most commonly used to resolve domain names like `www.aalto.fi` into one or more IP addresses before connecting to a service. The resolution of domain names can be done either iteratively or recursively. Iterative resolution consists of querying multiple DNS servers until the authoritative name server of the domain name is found. This server will always return the most recent version of the requested DNS record. When resolving a domain name recursively, a single query is sent to a recursive DNS server which will then resolve the domain name iteratively. Recursive DNS servers typically

cache the results of the iterative queries undertaken on behalf of its clients and is allowed to answer queries using the cached results. The validity of the cached DNS information is controlled by the authoritative name server that originally delivered the information.

Home gateways often implement a recursive DNS server so that all clients on the local area network can benefit from faster DNS replies. In a typical setup LAN clients query the home gateway which in turn queries the recursive DNS servers of the ISP when necessary. In recent years the use of third-party DNS servers outside of ISP networks have become popular for various reasons. We believe that the most significant reasons include IP addresses of resolvers being easy to remember and that DNS information is less likely to be censored by third-party DNS servers. Third-party DNS services have been criticized for not actually improving performance and for even impairing the service of users [2]. The main point of the criticism is that the latency between client and server typically defeats any performance optimizations. Even if the DNS resolver of an ISP is slow, its proximity to the end-users still makes the resolution time shorter than when using any third-party resolver in most cases. Additionally, third-party DNS resolvers are suspected of providing suboptimal server selections for content distribution networks. Because of the distance between end-users and third-party DNS servers, the assumption that the source IP address of a DNS query is equivalent to the location of the end-user is no longer valid. Understanding the performance and use of DNS is therefore an important aspect of not only web site performance, but the performance of most Internet services.

1.3 Problem Statement

Since the invention of the World Wide Web the content of the web has changed significantly and the web pages of today are sophisticated, providing desktop-like interfaces with a high degree of interactivity. The change in content has also made the web ever more important to a lot of people across the world. In search of cost savings, governments and banks push online services providing critical services to society and businesses. It is therefore ever more important to understand the performance of web page delivery, in order to monitor and improve the user experience of the web.

The aim of this Master's thesis is to evaluate the network performance of web page delivery. This will be accomplished by means of active measurements of a set of popular websites. The performance of DNS and HTTP will be measured on a page-element level in order to analyze potential performance differences depending on the origin of content. Finally, using the collected DNS information the performance of web pages will be linked to the use of content distribution networks.

This Master's thesis is structured as follows. Chapter 2 introduces the Domain Name System, web pages, HTTP, and content distribution networks. Chapter 3 explains the research goals of this thesis, performance metrics of interest, and how these will be measured. In Chapter 4 the collected measurement data is analyzed and the results of the analysis are discussed in Chapter 5. Finally, the conclusions are summarized in Chapter 6.

Chapter 2

Background

2.1 Domain Name System

In the early days of the Internet, IP addresses associated with hostnames were stored is a single hosts file (`hosts.txt`), which was distributed using FTP to all hosts in the Internet. Changes to hostnames or IP addresses were submitted to the maintainer of the hosts file, which would then update the file. The changes would propagate through the internet as each host downloaded the latest version of the hosts file. This approach offered basic name resolution and worked reasonably well due to the limited number of hosts in the Internet. However, the centralized nature of this solution also meant that it had poor scalability and with the increase in number of hosts connected to the Internet, a more efficient and scalable solution for name resolution was needed.

The Domain Name System (DNS) [3] was designed to solve the problem of resolving hostnames. Domain names are hierarchical and consists of a series of labels, each a subdomain of the previous, except for the first label which is the root domain. In written form a domain name is separated by dots and ordered by decreasing level in the hierarchy such that every domain name ends with the root domain. The domain name `www.aalto.fi.` consists of four labels: www, aalto, fi, and the root label which is empty. Because all domains names contain the root domain, the last dot is generally omitted. Figure 2.1 shows an example of the DNS hierarchy of six domain names including the root domain: "`fi.`", "`aalto.fi.`", "`elec.aalto.fi.`", "`www.aalto.fi.`", and "`mail.aalto.fi.`".

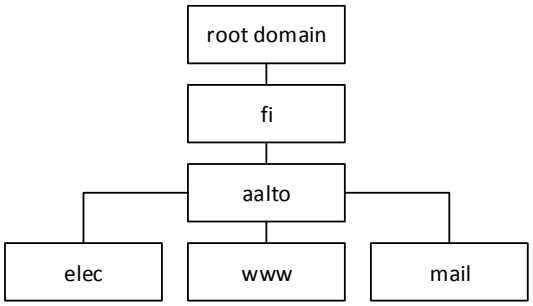

Figure 2.1: Example of DNS hierarchy.

DNS is essentially a distributed database and it was thought that it would also be used to store other information besides hostname to IP address mappings, e.g. email addresses associated with a domain. Nowadays, the primary use of DNS is resolution of domains into IP addresses and providing information about email servers. The hierarchical structure of DNS is one of the main reasons for its ability to scale well. Each domain label in Figure 2.1 can be controlled by separate authoritative name server, or by a single authoritative name server for the root domain. An authoritative name server can delegate control of its subdomains to one or more name servers and thereby reduce its load. A domain and its subdomains can also have the same authoritative name server, which is the case for the `aalto.fi` and its subdomains `elec`, `www`, and `mail`.

The second feature of DNS that improves scalability is record time-to-live (TTL) values, which specifies how long DNS records are valid. The TTL value of DNS records determine how frequently clients must refresh DNS information by resolving a domain name. When a DNS client receives a response to a DNS query, it is allowed to store the received records in its local cache until their TTL expires. Any subsequent DNS queries will be answered using the local cache without querying a DNS server. This greatly reduces the load on the DNS infrastructure, depending on the lifetime of records. Allowing DNS records to be used for a certain period of time also makes the DNS system tolerant to failures.

The fault tolerance of DNS depends on the rate at which DNS information expires at end-users. It will take longer time for a service providing DNS records with high TTL values to become unavailable to all users if the its authoritative name servers were to fail. Records with longer lifetime have a higher probability of remaining valid until a DNS problem is resolved. On the other hand, the use of high TTL

values also poses a problem if a service provider were to change DNS records. In this case the clients with cached DNS records could experience service disruption until the cached records expire and the domain name is resolved again. In order to take down the DNS infrastructure an attacker would need to make all the authoritative name servers of the root domain unavailable for a substantial amount of time. Distributed denial of service attacks have been targeted against the DNS root servers, but attackers have not been able to sustain the attack for long enough to affect the DNS infrastructure [4]. Ten out of thirteen DNS root servers are using anycast to distribute DNS queries across hundreds of servers, and it is therefore difficult to overload all root servers at the same time.

2.1.1 Message and Record Formats

The DNS protocol uses UDP as transport protocol, because the lack of a connection-setup phase reduces name resolution time and network load. Naturally DNS suffers from the lack of reliability like any protocols carried over UDP, but with the amount of DNS traffic on the internet today, it was a good decision to optimize with respect to traffic volume rather than reliability.

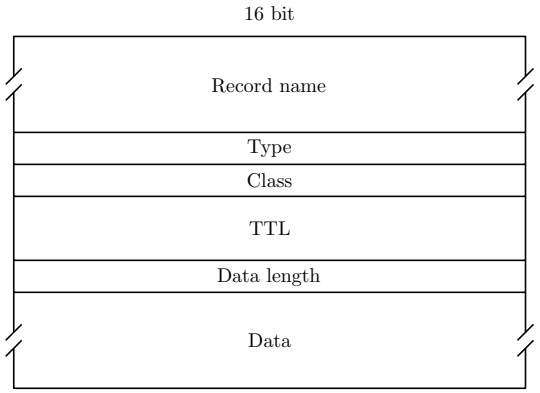

Figure 2.2: DNS record format.

The DNS protocol uses a single message format for both queries and responses. The message format is shown in Figure 2.3. A DNS message consists of a header and four record sections: questions, answers, authorities, and additional records. The header contains information about the number of records in the record sections and how a message should be processed by DNS clients and servers. Figure 2.2 shows the format of a DNS resource record. Each resource record consists of six fields: name,

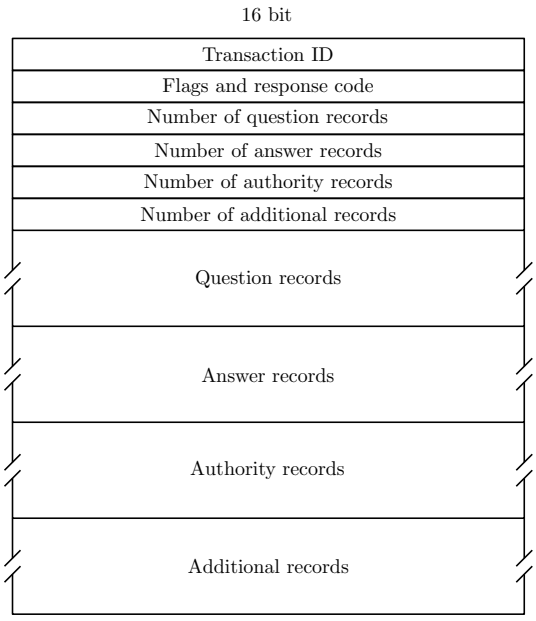

Figure 2.3: DNS message format.

type, class, TTL, data length, and data. The record name indicates which domain the record contains information about and the type tells what type of data is stored in the data field. When DNS was designed several network types existed and the class field was included in the message format as a way of separating information by network type. Nowadays, only the Internet class is used and the class field is an unnecessary legacy field that reminds us of a time when the Internet was less dominant than today. The RFCs concerning DNS define the possible record types and the format used in the data field for each type [3]. Over the years, extensions to DNS have been made in the form of extra resource record types. These extensions are compatible with the original DNS specification and the new record types specified in separate RFC documents.

Type	**Description**
A	32 bit IPv4 address record.
AAAA	128 bit IPv6 address.
CNAME	Alias pointing to another domain name.
NS	Nameserver record containing information about an authoritative name server of a domain.
MX	Mail exchange record telling what servers should handle email for a domain.

Table 2.1: Common DNS record types.

2.1.2 Resource Record Types

The record name field in a DNS record contains the fully qualified domain name about which the record contains information. The use of full domain names and not a single domain label is necessary because a name server can be the authoritative name server of multiple domains. Using full domain names prevents ambiguity. In order to reduce the size of DNS responses, domain names are compressed using a pointer scheme. Subsequent records in the same message contain references to previously used domain names rather than repeating the full domain name. Table 2.1 shows a list of the most commonly used DNS record types. The importance of these record types have not changed with time and improvements to DNS have typically been implemented as extra record types complementing the previously specified record types. IPv4 and IPv6 addresses are represented by A and AAAA records, respectively. The data field of these records has a fixed length of 32 bit or 128 bit. The CNAME record type is used to create domain aliases and contains the full alias domain name. For instance, the domain `www.aalto.fi` uses a CNAME record to point to the domain `aalto.fi`. Whereas A, AAAA, and CNAME record are the typical record types of interest to end-users, the NS record defines the hierarchy of DNS. The NS record contains information about authoritative name servers of a domain and is used to delegate control of subdomains to other name servers. The data field of an NS record contains the domain name of an authoritative name server of the domain name. When resolving a domain name iteratively NS records are used to determine the name servers relevant to the resolution of the domain name.

2.1.3 Protocol

Although the DNS message format supports up to 65535 question records in a single DNS query, the de facto standard is that only one question record is sent per query and servers will ignore queries with multiple questions. Move advanced queries involving e.g. DNSSEC will include extra query information in the additional records section of the query as specified by Extension Mechanisms for DNS [5].

A typical DNS query message contains a single question record matching the record type of interest. Thus if the client wants to know the IPv4 address of `aalto.fi`, the question record will be of type A and the name field contain "`aalto.fi`". Figure 2.4 shows an example of a DNS query message. The transaction ID of the message is selected randomly such that two pending queries never have identical IDs. Given

the size of the transaction ID space, collisions are possible if many queries are sent by the same client. In this case, the transaction ID space can be extended by also randomizing the UDP source port of queries. This effectively increases the identifier space with a factor of 65535, thereby making collisions highly improbable.

Transaction ID: 0xABCD
Flags and response code
Number of question records: 1
Number of answer records: 0
Number of authority records: 0
Number of additional records: 0
Record name: www.aalto.fi
Type: A
Class: Internet
TTL: 0
Data length: 0
Data: (empty)
Answer records: (empty)
Authority records: (empty)
Additional records: (empty)

Figure 2.4: Example of a DNS query message.

DNS records can be obtained through either recursive resolution or iterative resolution. A typical DNS resolver will send queries with the "recursion desired" flag enabled to a recursive DNS server either in the home gateway or provided by the ISP. When the flag is set, the DNS server is allowed to forward the query if it cannot be answered using cached information. Of course, the query will only be forwarded if the server configuration allows it. When using recursive name resolution, the DNS client only needs to send a single query message unless the packet is lost. The recursive name server will then handle the more complex iterative resolution of the domain name, if necessary. Iterative name resolution consists of sending the same query message repeatedly to different name servers. If the DNS resolver has no cached information, the first DNS server contacted is one of the 13 authoritative name server of the root domain, also know as root servers. If a DNS server is not the authoritative name server of the queried domain name, it will return an NS record telling which server most likely knows more about the domain in question. Consider the iterative resolution of the domain `aalto.fi`:

1. The DNS client sends the DNS message shown in Figure 2.4 to a DNS root server.

2. The root server does not have an A record for the full domain name and therefore returns an NS record saying that the authoritative name server of the `fi` domain can provide a better answer.

3. The DNS client repeats the query to the name server of the `fi` domain, and this server returns an NS record for the `aalto.fi` domain.

4. The DNS client has now received the address of the authoritative name server of the `aalto.fi` domain and when this server is queried it returns one or more A records for the `aalto.fi` domain.

Because NS records contain the *domain name* of an authoritative name server of a domain, it may be necessary for the resolver to obtain an A or AAAA record for this domain before being able to continue the resolution of the original domain name. For this reason, the use of so-called glue records in DNS responses containing NS or CNAME records are necessary in order to connect different parts of the DNS hierarchy together at the network layer. Without any IP address information it would not be possible to query name servers defined in NS records.

2.1.4 Issues and Vulnerabilities

In 2008 Dan Kaminsky described how DNS caches could easily be poisoned due to a flaw in common DNS server implementations [6]. The transaction ID of DNS messages is only 16 bits long and Kaminsky demonstrated that it was possible to predict transaction IDs. By correctly predicting the transaction ID of a DNS query, an attacker could send malicious responses that would be accepted as valid by a DNS resolver. The attack could be used to poison DNS caches with invalid DNS information, and thereby potentially redirect traffic to malicious servers. The attack was possible because most DNS software used fixed source port numbers of DNS requests and did not randomize the transaction ID. The use of sequential transaction IDs made it possible to send relatively few fake DNS responses and have one of them accepted with high probability. The problem was mitigated by randomizing both source UDP port and transaction ID, thereby increasing the transaction ID space. The DNS protocol still has a weakness but the probability of an attacker guessing

both the source port and transaction ID is extremely low. It is also infeasible for an attacker to flood a DNS client with all possible combinations, as once the correct answer is received by the DNS client all subsequent messages with the correct transaction ID are ignored.

2.1.5 DNS Security Extensions

The DNS Security Extensions (DNSSEC) [7] [8] were designed to improve the security of the original DNS standard, which did not offer any means of ensuring authenticity and integrity of DNS records. Applications protocols such as HTTP have long been protected by Transport Layer Security[1] (TLS), and this has generally mitigated the threat of DNS tampering by teaching users always to make sure a web page URL starts with `https` before providing sensitive information. However, TLS does not protect against DNS poisoning and it is possible to redirect users by manipulating DNS information. Victims could be redirected to malicious servers and be tricked into providing sensitive information. DNSSEC is important because it prevents attackers from manipulating DNS information and in combination with TLS, it will significantly improve online security.

DNSSEC ensures authenticity and integrity of DNS records using public-key cryptography in a similar fashion to TLS. The hierarchical structure of DNS can be mapped to chains of trust in a public-key infrastructure. Each authoritative name server has a public and private key, and the public key is available as a DNSKEY record. An authoritative name server that delegates control of subdomains is known as the parent of the subdomains. The parent vouches for its subdomains by signing a hash of the public key of the subdomains. The signed hash is stored as a Delegation Signature (DS) record at the parent name server. The parent name server can in turn be signed by its parent and so on, until the chain of trust reaches a trust anchor. A trust anchor is a public key distributed out-of-band and which is the root of a trust chain. The root domain has been signed and its public key distributed with DNS resolvers and can therefore be used to validate DNSSEC information. Backwards compatibility is maintained by signing hashes of existing record types and storing the signature in a Resource Record Signature (RRSIG) record. If a client has DNSSEC capabilities it will indicate so in the additional records section of its DNS queries. DNS responses will contain the requested record along with its

[1] Previously known as Secure Sockets Layer (SSL)

corresponding RRSIG record and the DNSKEY record of the domain. The client will verify that the DNSKEY of the domain is trusted by verifying the chain of trust from the domain name to a trust anchor. If the key is trusted it is used to authenticate and verify the integrity of the resource records in the DNS response.

The specification of DNSSEC started more than 10 years ago, but actual deployment in the Internet is still limited. As of November 9, 2013 36% of the top-level domains had been signed by the DNS root [9] and less than 0.5 percent of the subdomains of `com` had been signed [10]. The slow adoption can mainly be attributed to DNSSEC being more cumbersome to manage. Cryptographic keys must be managed safely and improper configuration will result in DNSSEC-capable clients rejecting DNS information and thereby be unable to access services. Many services managing domain names are providing tools to help configure DNSSEC, so hopefully the adoption rate will increase in the future.

2.1.6 Third-Party Resolvers

Google DNS and OpenDNS are the two most popular third-party DNS services. Google claims that its service "speeds up your browsing experience" by offering faster DNS resolution through higher capacity and improved caching compared to ISP DNS servers. Furthermore, the user base of Google DNS is much larger than that of any ISP and the probability of cache-misses should therefore be lower. Google DNS serves 70 billion DNS queries per day on average and is the worlds largest DNS resolution service [11]. OpenDNS served an average of 49 billion DNS queries per day between October 9 and November 7, 2013 [12]. Both services rely on anycast to direct end-users to nearby servers.

Ager et al. [13] evaluated the performance of Google DNS, OpenDNS, and the default resolver of measurement nodes using active measurements. The measurements were performed using colleagues and friends, and this approach provided 50 vantage points in different ISP networks, spread across 28 countries. The measurement locations were connected to the Internet through commercial ISPs, but the characteristics of the connections were not described. The study focused on DNS performance of web pages, and measured the name resolution of the main domain along with the domains of page elements. The 5000 most popular domain names and the 2000 least popular domain names were selected from Alexa's global rankings [14]. In

addition, the web pages of the top 1000 Alexa rankings were analyzed and 3500 domain names of page elements were selected. This selection of domain names was made such that some domains should have low cache-miss probability, while others should most likely have a high cache-miss probability. The measurement procedure was designed to measure the impact of cache misses, by issuing sequential DNS queries for the same domain name to the same DNS resolver such that differences in resolution time could be recorded. The measurements showed that the default resolver generally outperforms Google DNS and OpenDNS, but not in all cases. The better performance of the default resolver comes from the fact that the latency to the client is much lower than between the third-party resolvers and the client. The client must be very close to a Google DNS or OpenDNS server in order for the performance to be comparable to the default resolver. Additionally, the study showed that some internet service providers load balance queries to their DNS servers, and thereby impair the performance of their setup because the servers did not use a shared DNS cache. Subsequent queries for the same domain name might not be directed to a server with the requested record in its cache. Ager et al. concluded that the proximity of the client and DNS resolver is the limiting factor for DNS resolution time. Furthermore, any optimizations made by Google DNS or OpenDNS could not be confirmed, and if present generally did not yield better performance compared to using the default DNS resolver. Finally, it was observed that for the tail of the distribution of DNS resolution times, OpenDNS performed significantly worse than Google DNS.

2.2 Web Pages and HTTP

A web page is basically a file containing text and HyperText Markup Language (HTML) [15] syntax, defining the layout and files needed to render a page. HTML has become more advanced over the years and so has web pages. Modern web pages can be highly dynamic and rely heavily on JavaScript to provide interactivity and create content. The main content type categories of web pages are: HTML, JavaScript, style sheets, images, and plugins. The HTML text provides text content and the general structure of a web page along with embedded JavaScript, style sheets, and images. The JavaScript content of a web page typically consists of linked libraries and scripts utilizing these libraries. Style sheets allow web page designers to easily manipulate the look of a page without modifying HTML syntax. Images

constitute the graphical content of web pages and can be embedded in the HTML text, but are usually files linked to using HTML tags. Plugins are software that provide functionality not natively supported by web browsers. Two popular plugins for web browsers are Java and Flash. Flash became a popular choice for video playback because this was not supported by the HTML standard. Java in the form of Java applets is used on the web for instance by online banking solutions, but many companies are abandoning Java applets due to security issues and the lack of Java support on mobile devices. The specification of HTML 4.01 was completed in 1999 at a time where the web was quite different compared to the web pages of today. The standardization of HTML 5 is expected finish in the near future and is a major overhaul. Browsers which implement HTML 5 will support a range of features natively which previously required plugins. These features will be accessible as new HTML tags and JavaScript functions, and hopefully result in a more coherent and cross-platform compatible web.

```
GET /search?q=KEYWORD HTTP/1.1
host: www.google.fi
accept-encoding: gzip,deflate,sdch
accept-language: en-US,en;q=0.8,da;q=0.6
user-agent: Chrome/30.0.1599.114 Safari/537.36
accept: text/html
cache-control: no-cache
```

Figure 2.5: Example of HTTP headers in a GET request.

Web pages are delivered over the Internet using the HyperText Transfer Protocol (HTTP), which is an application-layer protocol running on top of TCP. The download of any file over HTTP can be divided into three phases: domain name resolution, TCP connection establishment, and file transfer. In the first phase the domain name is resolved by the DNS resolver of the operating system. If the resolver has the IP address of the domain in its cache, this address is used in the second step. Otherwise, the resolver continues by sending a recursive DNS query to its default DNS server. In the second phase, a TCP connection is established to the IP address of the domain on port 80 which is the default HTTP port. Once the connection has been established, the HTTP client can send a request to the HTTP server. The HTTP server either responds with the requested file, a redirection command, or an error page. Figures 2.5 and 2.6 show examples of the HTTP headers for a GET

request and a response, respectively. In a request the header contains the path of the requested file along with the domain name it belongs to. It is necessary to include the domain name in the request, because an HTTP server may serve multiple domain names on the same IP address. The header also contains information about the HTTP clients capabilities, such as HTTP version, web browser, and accepted compression schemes. The first line of a response always contains the HTTP version used by the server, a three-digit status code, and optionally a human-readable status message. The remaining lines of the response header contain information relevant to the decoding of the payload of the response, such as content type, content length, encoding, and compression scheme. HTTP responses often contain a creation and expiry time, along with information about the cacheability of the payload. This information is used by HTTP clients to refresh content when needed and for HTTP caches to determine whether content should be cached.

```
HTTP/1.1 200 OK
cache-control: private, max-age=0
content-encoding: gzip
content-type: text/html; charset=UTF-8
Date: Sun, 10 Nov 2013 09:15:41 GMT
Expires: Sun, 10 Nov 2013 09:17:41 GMT
```

Figure 2.6: Example of HTTP headers in a response.

Figure 2.7 shows the process of downloading a web page and its elements. The name resolution phase is repeated once for each unique domain name. In HTTP 1.0 a new TCP connection is used to transfer each file and the connection-establishment is thus repeated before every file transfer. Delivering a single file per HTTP connection, makes HTTP clients and servers easy to implement, but at the cost of less efficient transfers. HTTP 1.1 supports persistent connections where multiple requests can be sent over the same TCP connection, thereby reusing the connection for multiple file transfers. Figure 2.8 shows the process of downloading a web page using persistent connections. The figure shows how the files downloaded in Figure 2.7 could be downloaded using only two TCP connections. Technically, the connection-establishment phase of a file transfer is only required once per domain name as all files hosted on the same domain could ideally be transferred over a single connection. The benefit of persistent connections is that the initial TCP handshake is avoided and the congestion window is larger for subsequent file transfers, because the connections do not have to undergo slow start again. Persistent connections improve performance on

high-latency links because the number of packet round-trips is reduced. However, the HTTP client must wait for the server to respond before requesting another file and this adds a round-trip time. The extra round-trip time can be avoided by using pipelining where multiple requests are sent immediately after each other on the same HTTP connection. The server then responds to each request in the order they were sent and this results in the best utilization of an HTTP connection. The most noticeable problem with HTTP pipelining is head-of-line blocking, where processing of one request delays the processing of the subsequent requests, as requests must be processed in the order they were sent. This reduces the throughput compared to using multiple TCP connections. Modern web browsers therefore use both multiple connections and pipelining to achieve the best download performance [16] [17]. We could not find any statistics on the usage of HTTP pipelining, but the tendency appears to be that HTTP servers support persistent connections but not necessarily pipelined requests. The most popular web browsers support this pipelining [18], but this is of little use if servers do not.

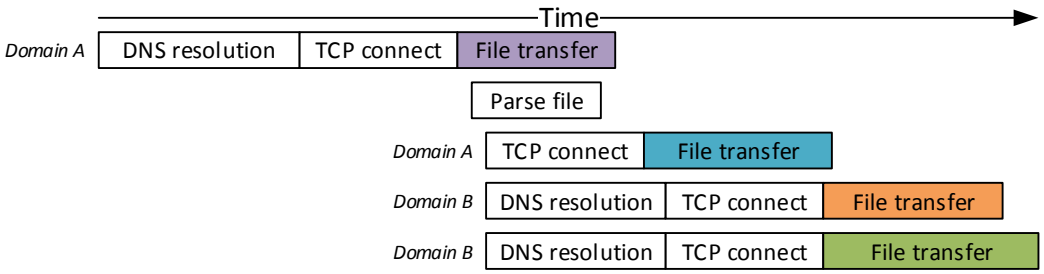

Figure 2.7: Web page download without persistent connections.

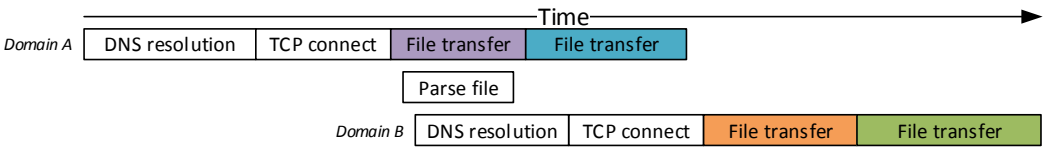

Figure 2.8: Web page download using persistent connections.

The main HTML file of a web page is the starting point of any web page download. The web browser parses the HTML syntax while it receives the file from the HTTP server and determines which additional files need to be downloaded, such that transfer of these files can begin while the first file is still being transferred. For this reason, most web browsers render web pages progressively as they receive page elements. Browsers impose limits on HTTP transfers in terms of maximum number

of overall and per-hostname simultaneous connections allowed. These limits vary from browser to browser, but typically 6 connections per hostname are allowed. The maximum number of connections independent of domain names varies a lot between browsers. Internet Explorer 9 allows up to 35 connections and Firefox 18 allows 11 connections [18].

Several studies deal with the performance of web page downloads, but unfortunately these studies are focused on the performance over mobile networks and their results are thus not directly comparable to measurements in wired networks.

Pries et al. studied the composition of the top 1 million web pages from Alexa [14] and presented an updated model of the characteristics of web page content [19]. The web page download performance was not measured, because their goal was to develop a model of web content. By considering previous models for HTTP content, it was concluded that web pages are growing in file size and number of page elements. Additionally, web pages are being downloaded from a greater number of servers than previously and this may affect download performance. Their measurements showed that the mean uncompressed file sizes was roughly 50% larger than the median file size.

2.3 Content Distribution Networks

With the growth of the Internet came the need for greater capacity to deliver content. The only way for a service provider to handle millions of simultaneous users is to distribute requests across many servers. When it comes to operating content distribution infrastructures, two approaches or a combination of both are generally used. The first and perhaps most common approach, is to build data centers which contain a large number of servers. Data centers concentrate capacity in locations with good connectivity to the Internet, and this enables easy maintenance and thereby reduced operational expenses. Furthermore, data centers are designed for high reliability with redundant links to the Internet through multiple network operators. The second approach consists of distributing the capacity of a data center across many geographical locations. These locations may be within data centers, ISP networks, or at Internet exchanges. Each location has little capacity compared to a data center and may not have as high reliability, depending on the location.

However, the use on many locations with different links to the Internet, makes it practically impossible for a service provider to end up in a situation where all servers are disconnected from the Internet. Granted, it is unlikely this would happen to a data center, but data centers can be said to be more of an "all eggs in one basket" approach. For this reason service providers use multiple independent data centers for additional redundancy.

The purpose of content distribution networks (CDNs) is to distribute the load of a service across an infrastructure such that data can be delivered with high performance and thereby give users a higher quality of service. Architectures with few or many hosting locations can be used to optimize content delivery. However, the concept of a CDN is often associated with a distributed architecture, perhaps because the largest CDN provider Akamai has such an architecture. A good CDN is characterized by not only distributing traffic across multiple servers, but also distributing the traffic geographically across multiple locations. Using multiple locations increases network capacity, improves redundancy, and reduces the average distance between client and server. In most cases shorter distance is equivalent to lower network latency and thereby higher throughput. With the use of multiple servers and locations, an important part of a CDN is to direct end-user requests to an optimal server. Determining the optimal CDN server can be done at different layers of the network stack and with varying degrees of precision. The following sections explain the most common methods of distributing the traffic of a service provider in the Internet and at the ingress point of hosting locations.

2.3.1 Anycast

The Internet Protocol carries all types of Internet traffic and the routing mechanisms for IP traffic can be used to distribute traffic on a global scale [20]. The Border Gateway Protocol (BGP) is an exterior gateway protocol used to route traffic between autonomous systems (AS) which are the largest topological units in the Internet. A service provider with its own AS number advertises its network prefixes through BGP and thereby connects its network to the rest of the Internet. The service provider can advertise its IP prefix from multiple locations in the Internet. When border routers in other autonomous systems calculate the optimal route to the prefix, the route will depend on the location of the border router in the Internet. This causes packets going to the service provider to be routed to the closest server

according to BGP. This concept is known as anycast routing and can be used for global load balancing at the network layer. Anycast routing is only useful if servers are hosted in geographical locations far apart, as the distribution of traffic might otherwise not be beneficial. Anycast routing provides stateless traffic distribution and is typically used for stateless services such as DNS. It is also possible to use anycast for statefull services, but when routes change connections will potentially break. It is only worth optimizing routing in the Internet for large service providers. For instance, Google DNS uses anycast routing to direct user to servers in 25 different IPv4 networks [21].

2.3.2 Load Balancing

Google DNS uses two IP addresses, 8.8.8.8 and 8.8.4.4 for their recursive DNS servers. These addresses are routed using anycast, but considering the amount of traffic Google DNS handles, request must be distributed across multiple servers in multiple hosting locations. Because DNS is a stateless protocol, this can be done using network-layer load balancers. Each IP address points to a load balancer which will direct incoming packets to servers based on some distribution algorithm. The distribution could be as simple as round robin, but could also be based on additional parameters such as server load. In contrast to load balancing of stateless servers, stateful protocols such as TCP require packets to always be directed to the same server. This requires a stateful load balancer to remember the server assigned to a specific traffic flow such that packets belonging to the flow are directed to the correct server. The rate at which packets can be processed depends on the number of parameters taken into consideration before forwarding a packet. If a service provider has the need for load balancing, it is probably meaningful to optimize the server configurations for load balancing. This could be done by using dedicated servers for different services, e.g. DNS, HTTP. If a single load balancer were used for multiple service types, it would be necessary to look at transport-layer information in order to direct traffic flows correctly. On the other hand, if different services had their own load balancer it would only be necessary to consider network-layer information for stateful services and stateless services could be balanced independent of any information contained in packets.

2.3.3 Application Layer Redirection

Anycast routing provides load balancing based on network topology and dose not consider the actual load of servers. An anycast address could therefore point towards a location where servers are experiencing high load. The network-layer load balancer at such a location could have more detailed information about the current load of servers. However, because server load constantly changes, the load balancer may still direct traffic to an overloaded server. In this case the request can either be dropped or be redirected at the application layer. The latter is of course preferable as a dropped request appears as an error to the end-user, whereas a redirection appears as increased processing time. Redirection at the application layer relies on mechanisms in the application protocol and is thus not transparent to the client. A good example of application-layer redirection is HTTP where certain response codes are used to indicate that the requested file has changed location. It is then up to the HTTP client to interpret this response code and initiate a new request to the address specified in the header of the redirection response.

Application-layer redirection has a number of disadvantages that makes it preferable to do proper load distribution below this layer of the network stack. First of all, an overloaded server *always* has to process a request even if the request will be redirected. Traffic is thus not directed away from an overloaded server but merely deflected by it. Although redirection responses can be sent quickly, the resources spent redirecting clients may disrupt service for existing clients. Secondly, because the redirection information is supplied at the application layer it cannot be cached by local gateways. In a network where multiple clients share a common gateway, caching could reduce the load on an overloaded server by caching redirection responses. In order for a gateway to cache redirection responses it would have to be able to understand the application-layer protocol where the address of the alternate server is supplied. If the application protocol is encrypted, the gateway has no chance of detecting redirections.

2.3.4 DNS Redirection

Despite application layer redirection generally being undesirable, the most popular method of load balancing technically relies on an application-layer protocol. DNS is the preferred method of redirecting users to an optimal server because it is transparent to all services that use domain names. Operating systems typically implement a

DNS client as a shared library used by all applications that need to resolve domain names. This essentially makes DNS act like a pseudo-layer between the application layer and network layer. DNS can be used to provide transparent redirections to all applications by having a DNS server return different responses to queries, depending on some load balancing algorithm. Furthermore, because DNS is an integral part of the Internet, most home gateways have a built-in DNS cache that performs recursive name resolution on behalf of clients on the local network. This enables redirections to be cached and thereby improve performance for other clients on the network.

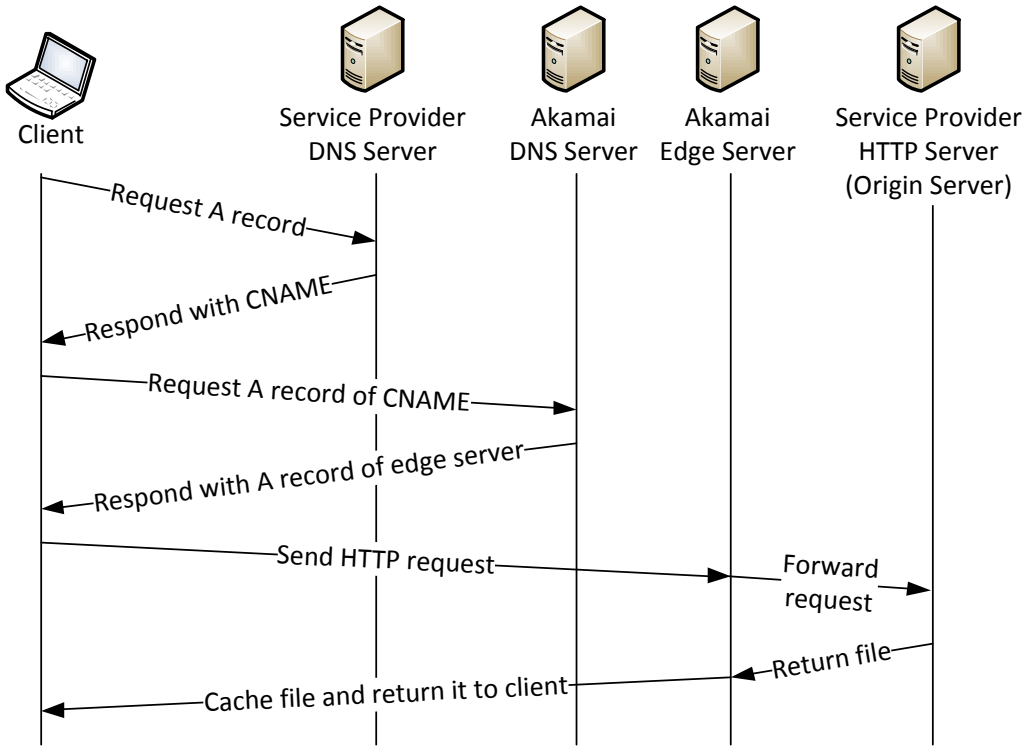

Figure 2.9: Example of HTTP download using Akamai.

Figure 2.9 shows how HTTP files are served using the Akamai CDN [22]. The following steps are completed in order to download a file from an Akamai CDN server:

1. The client or a recursive resolver begins by resolving the domain name of the service provider, e.g. www.cnn.com.

2. The DNS query is sent to the authoritative name server of the domain name

which has been configured to return a CNAME record pointing to a domain name controlled by Akamai, e.g. www.cnn.com.edgesuite.net.

3. The DNS resolver continues by resolving the domain name controlled by Akamai.

4. The Akamai DNS server returns an A record containing the IP address of an appropriate HTTP edge server. Akamai uses multiple levels of DNS redirection and may also return yet another CNAME record in this step, in which case the client repeats step 3.

5. The client has finally obtained an IP address of an edge server and sends an HTTP request to it.

6. If the edge server has the requested file in its cache it will return it to the client immediately. Otherwise, it will forward the request to the origin server of www.cnn.com.

7. The origin server will return the file to the edge server.

8. The edge server saves the file in its cache and delivers it to the client.

Content distribution networks use multiple levels of DNS redirection to direct clients to an optimal server. The optimal sever is selected based on a set of parameters, including IP address of the client, type of service, and current load of the CDN. Redirections are made by returning different NS, CNAME, or A records depending on the server selection parameters. Each redirection narrows down the area within which a CDN server will be selected. This is a good design choice, because it allows optimal redirection to be made without requiring global knowledge of the state of the CDN. Table 2.2 shows a simplified trace of how the domain name `newsimg.bbc.net.uk` is resolved iteratively. The table shows which name server was queried, the most important records in the response, and the TTL of that record. The trace shows that three CNAME records are encountered while resolving the domain name. The first CNAME points to another CNAME controlled by BBC. If the DNS records had been configured optimally, the first domain would have pointed to the `news.bbc.co.uk.edgesuite.net` domain, instead of another CNAME. The `edgesuite.net` domain is controlled by Akamai and provides the first level DNS redirection using an NS record. When the authoritative name server of the `edgesuite.net` domain is queried, another CNAME record is returned. This

CNAME record is the second level of DNS redirection. When the alias domain name is resolved the third redirection is received and this is the final server selection. When the domain `a1733.g.akamai.net` is resolved, Akamai makes the final server selection and returns two A records with IP addresses of suitable severs with respect to the client.

Name Server	**Record Name and Data**	**Record TTL**
k.root-servers.net	NS(uk)= ns1.nic.uk	2 days
ns1.nic.uk	NS(bbcimg.co.uk) = ns1.tcams.bbc.co.uk	2 days
ns1.tcams.bbc.co.uk	CNAME(news.bbcimg.co.uk)= newsimg.bbc.net.uk	1 hour
ns1.nic.uk	NS(bbc.net.uk) = ns0.tcams.bbc.co.uk	2 days
ns1.nic.uk	NS(bbc.co.uk) = ns1.rbsov.bbc.co.uk	2 days
ns1.rbsov.bbc.co.uk	A(ns0.tcams.bbc.co.uk) = 212.72.49.2	1 day
ns0.tcams.bbc.co.uk	CNAME(newsimg.bbc.net.uk)= news.bbc.co.uk.edgesuite.net	300 seconds
k.root-servers.net	NS(net) = a.gtld-servers.net	2 days
a.gtld-servers.net	NS(edgesuite.net)= ns1-2.akam.net	2 days
ns1-2.akam.net	CNAME(news.bbc.co.uk.edgesuite.net)= a1733.g.akamai.net	2 hours
a.gtld-servers.net	NS(akamai.net) = zc.akamaitech.net	2 days
zc.akamaitech.net	NS(g.akamai.net) = n1g.akamai.net	9 hours
n1g.akamai.net	A(a1733.g.akamai.net)=87.108.18.42 A(a1733.g.akamai.net)=87.108.18.40	20 seconds

Table 2.2: DNS trace of resolution of newsimg.bbcimg.co.uk.

When the TTL values of the records returned by Akamai are considered, only the records returned in the final response could have been selected based on actual server load. It was therefore suspected that the previous redirections were simply directing the DNS client to a DNS server responsible for an Akamai region. However, by resolving `a1733.g.akamai.net` from multiple locations in the Internet, it was observed that the two first levels of DNS redirection were repeated independently of the location. This suggest that Akamai server selections are based on global knowledge of the state of the CDN and that server selection is not handled by region-specific DNS servers. Akamai has a large monitoring infrastructure where performance information is gathered from every CDN location and aggregated [23]. By utilizing this information, it would be possible for end-users to get an optimal server selection from a single Akamai DNS server without multiple levels of redirection. The extra steps needed to resolve Akamai domain names are therefore likely due to redundancy, or they may play a part in the server selection that is not entirely obvious.

The TTL of the final two A records indicates that Akamai believes the performance of the selected servers will be optimal within the next 20 seconds.

DNS is a good solution for directing clients to servers based on their estimated location. However, one key feature of DNS has become a problem for CDNs with the increasing popularity of third party resolvers outside of ISP networks. DNS caches are an important part of the DNS infrastructure and most home gateways provide a recursive DNS resolver that typically forwards requests to the DNS servers provided by the ISP. The location of a client is the most important piece of information when it comes to selecting an optimal CDN server. The location is estimated using the source IP address of DNS queries. When the domain name pointing to a CDN is resolved iteratively by a client, an authoritative name server of a CDN will see the client as the source of the query. However, when a recursive DNS server resolves a request on behalf of a client, the recursive DNS server will be seen as the source of the query. This has not been a problem in the past because most users relied on ISP DNS caches for name resolution. These servers were geographically close to the end-user and the selected CDN server was the same as if the end-user had resolved the CDN domain name. The problems arise when the distance between the recursive DNS server and the client becomes large, as is the case for services like Google DNS and OpenDNS. Even with the use of anycast routing, the distance is much greater than between clients and ISP name server. When a third-party DNS resolver queries the authoritative name server of a CDN, the CDN returns the address of an optimal server for the third-party resolver, but not for the client it is sending the query on behalf of.

The only solution to this problem is for the third-party resolver to include the address of its client in the query message, or to deploy additional servers such that third-party DNS servers are as close to end-users as ISP DNS servers. The experimental IETF draft “Client Subnet in DNS Request” [24] defines a extension using Extension Mechanisms for DNS (EDNS0) [5]. The extension enables recursive DNS servers to include the IP address and network mask of the client that a query is being processed on behalf of. The most important fields of extension are source netmask, address, and scope netmask. The source netmask field defines the number of bits from the clients IP addres that are included in the address field. The scope netmask field is used by DNS servers, to specify which IP addresses the answer is intended for and can be cached for. The source netmask is useful from a privacy perspective,

because it allows a recursive resolver to only reveal a limited amount of information about the original sender of a query. Google is contributing to the specification of the extension and according to Otto et al. the extension is already in use by Google DNS [2].

The performance of content distribution networks is an active area of research. Many studies focus on video streaming because of the rise in traffic volume of this content type. Another reason for video streams being a popular content type to measure is the need for large files to evaluate the performance of CDNs in terms of bandwidth. Web pages generally consists of many small files and the average bandwidth during download of such files is therefore not representative of the CDNs throughput capacity. The duration of TCP connections is not long enough for transfers to reach their maximum transfer rate.

2.3.5 CDN Providers

Akamai is the largest CDN provider in the world and started offering commercial services in 1999, less than ten years after the invention of the World Wide Web. The traffic volume of Akamai is at least as big as the traffic volume of all other providers combined [25]. The Akamai infrastructure is optimized to reduce latency between clients and servers, by locating CDN servers as close to the end-user as possible, at the edge of the network. This meant deploying a very large number of servers across the globe and Akamai operates by far the most CDN locations of any of CDN provider.

Akamai offers to deploy servers within ISP networks at no cost and it is possible that ISPs do not charge Akamai either. This is because the presence of Akamai servers is as beneficial to ISPs as it is to Akamai [26] [27]. Both ISPs and Akamai have to pay for transit traffic and locating servers in ISP network thus reduces transit traffic and thereby the expenses of both parties. Akamai states that compared to a centralized CDN architecture, its architecture provides improved redundancy, greater throughput, and reduced latency [25]. However, operating servers in many locations comes at an increased cost compared to large data centers. According to Triukose et al. [28], Akamai could consolidate their server locations without noticeable changes in performance, but with significant cost savings. Triukose also concluded that the Akamai architecture favors clients with high-speed connections because it is easier to achieve high throughput over short distances in the Internet. As of 2013, Akamai op-

erates more than 130,000 servers and is present in 2,200 points of presence (Internet exchanges) and 1,200 networks, spread across 81 countries.

Akamai does not operate its own network, but relies on the Internet as its backbone network [25]. This is likely less expensive and less complex, but at the cost of not being able to control quality of service. Because the Internet is a best-effort service, Akamai servers behave like islands in the network with no need to communicate with other Akamai edge servers. Edge servers naturally have to communicate with the part of the Akamai infrastructure responsible for server selection and load balancing. The part of the Akamai architecture that serves files over HTTP operates in conjunction with the servers of their customers. The CDN essentially works as an enormous HTTP cache that fetches content from customer servers when needed, and end-users are directed to caches rather than the customer's servers.

Level 3 and Limelight are two competing CDN providers that take a different approach to content distribution networks compared to Akamai. Both providers have a more centralized architecture with servers located in a few data centers, as opposed to the many CDN locations of Akamai. The design philosophy of Level 3 and Limelight is that operating large data centers with lots of capacity provides good performance at a reduced cost compared to a distributed architecture. There is no doubt that locating servers closer to end-users will reduce latency, but the question is whether the difference in latency between a centralized CDN and a distributed CDN has any significant performance impact.

Otto et al. measured CDN redirections and HTTP latency for Akamai and Limelight from 10,000 locations [2]. The CDN redirections were measured using Google DNS, OpenDNS, ISP name servers, and iterative resolution. Their measurements showed that 90% of the locations observed no similarity between DNS answers by the authoritative DNS servers of Akamai, and the answers by Google DNS and OpenDNS. The same was true for 50% of the measurement locations for Limelight. The DNS answer obtained from the ISP name servers had a much higher degree of similarity than the third-party DNS resolvers for both Akamai and Limelight. The HTTP latency measurements showed that using iterative resolution or the ISP DNS server resulted in the best selection of a CDN server for both CDN providers. The difference in HTTP latency between iterative resolution and third-party resolvers, was more noticeable for Akamai. The HTTP latency of servers selected using iterative resolution and third-party resolvers were equivalent for 30% of the measurements

locations for Limelight. The same could only be said for 10% of the locatios for Akamai. This indicates that the performance of Akamai depends more on good CDN redirection than Limelight.

The performance of server selections of Akamai and Limelight were also evaluated using the "Client Subnet in DNS Request" extension. Source netmasks of 32 bit, 24 bit, and 16 bit were used. The results showed that the full client IP address always resulted in the best performance. For 60% of the measurement locations the 16 bit netmask resulted in performance similar to using the full address. The performance observed for the 24 bit prefix was comparable to that of the full IP address.

Su et al. tried to infer good paths in the Internet from the server selections made by Akamai [29]. Their research showed that Akamai typically managed to avoid congested network paths and that mapping an overlay network to Akamai's architecture could improve performance of the overlay network. Their research indicated that Akamai uses more information than just client location, latency and server load to select servers. Akamai is likely monitoring congestion in the internet from their many hosting locations and optimizes server selection based on this information.

In this chapter the protocols needed to download web pages were introduced. The structure of the Domain Name System was explained along with the process of name resolution. Finally, the key mechanisms of content distribution networks were introduced.

Chapter 3

Measurements

3.1 Metrics

The download of a web page consists of DNS resolution and HTTP download. In order to study the performance of web page delivery performance indicators of this process must be identified. From an end-user's point of view the performance can be summed up to the total download time of a web page. The DNS resolution time and HTTP download time will therefore be used to describe the performance of the protocols. By including all steps of the resolution process in the resolution time, packet loss and server errors are also captured by the metric. Although the download time is important to the end-user, it is not a good metric for estimating the performance of the network path between client and server. The congestion control mechanisms of TCP affect throughput and download time is therefore not a good metric for assessing network performance. One of the most important factors affecting TCP throughput is latency and measuring the latency between client and server will therefore give a better picture of potential throughput. The latency can be measured as the TCP connect time, as there is a fixed relationship between this value and the actual round-trip time (RTT). It is also possible to measure the RTT using ICMP, but since some routers in the Internet treat TCP and ICMP traffic differently [30], TCP connect time is a better indicator of potential TCP throughput.

CDNs often use DNS redirection and by recording the messages exchanged during the resolution process it will be possible to analyze the server selection mechanisms of CDNs along with general DNS characteristics of the domain names used by web

sites. Because CDNs direct users to servers based on their estimated location, measuring websites from a single locations will not yield any interesting results with respect to content distribution networks. To observe different CDN server selections it is necessary to perform measurements from multiple locations in the Internet. If the locations are sufficiently far apart from each other, CDNs should return different IP addresses of servers. By performing measurements from the geographically distributed measurement platform PlanetLab, it will be possible to measure CDN performance from a global perspective.

We decided to measure popular websites as these likely rely on content distribution networks to improve their service. Measuring popular websites fulfills two goals of this thesis: measuring websites that are important to many users and measuring the use of CDNs to deliver websites. Thirty-three websites were selected from the rankings of Alexa [14] from the following categories: news, shopping, social networks, video, search engines, email, and Finnish news. Websites were selected according to their popularity internationally with the assumption that such sites optimize their content delivery at a global level. The number of users is not necessarily an indication of international popularity, as the potential user base of a website largely depends on the language of the site. The Finnish news sites were measured because these were not expected to utilize CDNs. Measurements of those websites could therefore be used to evaluate the performance differences between sites using and not using CDNs.

Table 3.1 shows a list of the selected websites and their URLs. Some URLs redirected HTTP clients immediately and these URLs were replaced with the destination of the redirection, unless the redirection was based on client location. Some URLs immediately redirected HTTP clients to an HTTPS version of a page, and these URLs were also replaced with the HTTPS version of the URL. The reason for replacing URLs that lead to redirection pages is that these web pages do not contain any content. Furthermore, if a user visits a website frequently they might have a bookmark for it and thereby skip the redirection page. The measurements are therefore equivalent to a user accessing websites directly using a bookmark in a web browser.

Category	**URLs**
News	http://news.yahoo.com/ http://edition.cnn.com/ http://www.huffingtonpost.com/ http://www.weather.com/ http://www.reddit.com/ http://www.bbc.co.uk/news/ http://www.nytimes.com/ https://news.google.com/ http://www.foxnews.com/
Shopping	http://www.amazon.com/ http://www.ebay.com/ http://www.ikea.com/
Social networks	https://www.facebook.com/ https://twitter.com/ http://www.linkedin.com/ https://plus.google.com/ https://pinterest.com/ http://www.flickr.com/
Video	http://www.youtube.com/ https://vimeo.com/ http://www.dailymotion.com/ http://www.netflix.com/ http://www.hulu.com/
Search engines	http://www.google.com/ncr http://www.bing.com/ http://www.yahoo.com/
E-mail	http://www.gmail.com/ http://mail.live.com/ http://mail.yahoo.com/
Finnish news	http://www.iltalehti.fi/etusivu/ http://www.hs.fi/ http://www.iltasanomat.fi/ http://www.yle.fi/

Table 3.1: List of websites and their categories.

3.2 Measurement Application

Several DNS and HTTP tools exist for Linux, but the DNS tools are not able to provide a sufficient level of detail about the DNS resolution process. We therefore decided to implement a standalone application to collect the performance metrics of interest. An alternative solution would have been to use a collection of command-line tools and shell scripts. This would have made deployment of the measurement solution more difficult to manage across multiple platforms. Furthermore, differences in the output of command-line tools could also have complicated data processing. The measurement application consists of a DNS client working together with the popular HTTP library cURL [31]. The DNS client was implemented as a separate library and this allows other applications with the need for name resolution to also collect DNS performance information.

The DNS library is written in C and provides new API functions for DNS resolution. Compared to the name resolution functions in C, these functions are more complicated to use, as more resolution options can be controlled. Additionally, the resolution of a domain name results in more information than just IP addresses being returned for successful queries. The DNS library supports both recursive and iterative resolution, and records detailed information about the resolution process and all records received. Operating systems generally implement DNS caches to improve name resolution performance. The library therefore also supports caching of records, in order for measurements to be equivalent to the DNS performance experienced by applications using the native DNS resolution functions of an operating system. However, DNS records are only cached within a process and the cache is therefore not persistent. Because the expected run-time of the measurement application is shorter than the TTL of DNS records, records are not removed from the cache when they expire. From a research point-of-view there is no need for persistent caching as only the network performance of DNS is of interest. The DNS cache also provides the foundation for the query logic driving the name resolution process of the resolver. The cache is implemented as a tree structure because DNS information is organized hierarchically. In order for the resolver to work, the DNS cache has to be primed with information about which servers it can initially query. This is done by defining NS records for the root domain and adding the necessary glue records. The NS records can either be those of actual DNS root servers or recursive DNS

servers. The library also supports parsing of the `/etc/resolv.conf` file containing the list of default name servers on Linux systems.

Figure 3.1 shows an example of the DNS cache tree created after resolving the domain name `netlab.hut.fi`. Each label of a domain name is represented by a node in the cache tree, such that each node except the root node, has a parent and zero or more children. DNS records are added to the node with the domain label they contain information about.

The following is an explanation of how the cache tree in Figure 3.1 was created.

1. The cache tree is initialized such that queries for the root zone should be sent to `a.root-servers.net` and the A record of this domain is also defined.

2. The first query is sent to the authoritative name server of the root zone, which returns an NS record for the `fi` top-level domain along with the necessary glue records.

3. The second query is sent to the authoritative name server of the `fi` domain. The response to the query contains yet another NS record pointing to the authoritative name server of the `hut.fi` domain.

4. The third query is sent to the authoritative name server of the `hut.fi` domain and since the control of the `netlab.hut.fi` domain has not been delegated to yet another name server, the server returns the A record of the queried domain.

When subsequent domain names are resolved, the information in the DNS cache is used to make the optimal choice of which name server to contact. For instance, when resolving the domain `mail.netlab.hut.fi`, the resolver locates the leaf node "netlab" and checks if an NS record for this domain is in the cache. As Figure 3.1 shows, no NS record is know for `netlab.hut.fi` domain. The resolver continues by traversing the tree towards the root node until it finds an NS record. In this case the parent of the `netlab` domain contains an NS record. The resolver queries the authoritative name server of the `hut.fi` domain and receives the address of the `mail.netlab.hut.fi` domain. It is evident from this example that the number of queries which have to be sent is greatly reduced and that caching significantly reduces the resolution time of subsequent queries.

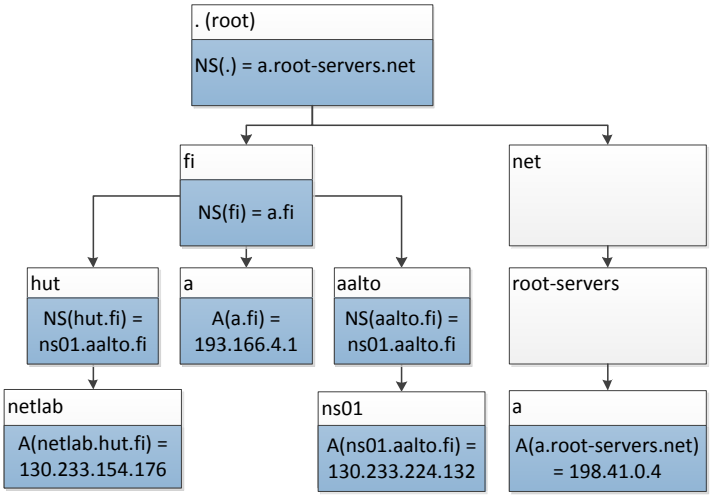

Figure 3.1: Tree structure used to represent DNS records.

The DNS library supports resolution of A, AAAA, and CNAME records. If the record type of a query is A or AAAA, the resolver will automatically follow CNAME records until an A or AAAA record is found.

We decided not to implement DNSSEC support as security aspects of DNS are not of interest in this Master's thesis. The behavior of DNS resolvers vary between different operating systems. Some resolvers are more aggressive and will send multiple requests for the same domain and thereby reduce the performance impact of packet loss or server errors. The implemented DNS resolver does not send multiple simultaneous queries as it seeks to measure the performance of the network and DNS servers without client-side optimizations. The performance impact of packet loss and server errors are therefore included in the measured metrics. The resolution algorithm is persistent and will query all relevant name servers until either a positive or negative response is received. If a DNS query is lost, the DNS resolver will simply wait until a timeout occurs before continuing. If a timeout occurs, the resolver will then send the query to the next IP address of the destination name server. If the queries to all IP addresses of a name server fail, the resolver will continue to the next authoritative name server and repeat the process. If a response contains a Start of Authority (SOA) record for the domain name being resolved and no answer records, the domain name does not exist and the resolver returns an error. If no DNS response is received after sending a query to all IP addresses of all appropriate name servers, the resolver will return to the parent level of the DNS hierarchy and send

the query to the next name server at this level. This approach will inevitably lead to the resolver querying the same name server at some point. For this reason, and because loops can also be created using CNAME records, the resolver features loop detection. The resolver keeps track of which name servers have already been queried and if a loop is detected, the resolver aborts the resolution process and returns an error. Table 3.2 shows the key metrics collected for each domain name resolved. The entire trace of queries sent and responses received is also recorded, such that the resolution process can be analyzed and new metrics be defined at a later time.

Metric	**Description**
Domain name	Primary domain name of query.
Final domain name	Domain name pointed to by last CNAME record or primary domain name if no CNAME records were followed.
Resolution time	Sum of network RTT of queries.
Answer IP address	IP Address of first A record in final DNS response.
Answer TTL	TTL value of first A record in final DNS response.

Table 3.2: Key DNS metrics recorded by measurements application.

The cURL [31] library is a popular library that implements an HTTP client among other things. The library is tried and tested, so we decided to use it to handle HTTP downloads, rather than implementing our own HTTP client. The library is not designed to be used by web browsers and therefore has a couple of shortcomings. cURL supports simultaneous connections and can enforce an overall connection limit. However, it is not possible to limit the number of connections per hostname. There is therefore a risk that the per-hostname connection limit used by web browsers is exceeded by the measurement application when downloading web pages with many page elements hosted on the same domain name.

Although cURL provides several hooks at key points of an HTTP download, it is not possible to override name resolution. The only solution is to resolve all domain names and replace the domain names in URLs with their IP address, before asking cURL to download the files. There are two issues with this solution. Firstly, name resolution and file downloads are intermixed by web browsers. The performance of DNS resolution may therefore be slightly better, because the connection is not being used for HTTP downloads at the same time. Secondly, by replacing a domain name with an IP address, multiple IP addresses provided via DNS cannot be used. Normally an HTTP client will try all IP addresses returned by DNS until a connec-

tion is established. If a HTTP connection cannot be established by cURL in the measurement application, the download will fail even if alternative IP addresses of the domain name are available.

The "user-agent" header is used to identify the HTTP client or web browser sending a request. Some websites return different files depending on the value of this header, and some sites return unexpected content or nothing at all. We chose to use the user-agent value of the Google Chrome browser as the default user-agent in the measurement application. The popularity of Google Chrome and Internet Explorer is comparable, but it should be noticed that the popularity of browsers vary greatly depending on the target audience of a web site. Additionally, older versions of Internet Explorer are known to require browser-specific HTML syntax for web pages to be rendered properly, whereas Google Chrome does not.

Tabel 3.3 shows the key metrics collected for each HTTP download. Some of the metrics are extracted from HTTP headers and some are reported by cURL.

Metric	**Description**
Connect time	Time to establish TCP connection to server excluding name resolution time.
Download size	Size of downloaded file reported by cURL.
Download time	Time to transfer file.

Table 3.3: Key HTTP metrics recorded by the measurement application.

3.3 PlanetLab

PlanetLab is a platform consisting of approximately 1000 servers that can be used for Internet research purposes. As the name suggests, PlanetLab provides measurement locations across the world, and this makes it suitable for CDN measurements. However, it should be noted that most nodes are located in Europe and North America. The multitude of locations makes it possible to measure services on the Internet from a global perspective. PlanetLab servers are hosted by organizations contributing to the platform in return for the possibility to run tests on the entire platform.

Each PlanetLab node must conform to a set of minimum hardware and connection requirements [32]. Many of the organizations contributing to PlanetLab have access

to high-speed connections and it is doubtful that any measurements location has a low connection speed. We expected PlanetLab nodes to have faster connection speeds that the average in Europe and North America. However, since web pages are rarely bandwidth intensive, users on high-speed connection should only experience marginally better performance than users with average Internet connections. High-speed connections favor download of large files because it takes time to increase the window size of TCP connections. Web pages contain many small elements and the TCP connections therefore do not last long enough for the window size to increase enough to take advantage of a high-speed connection. Measuring web page delivery performance on PlanetLab should therefore yield results comparable to locations with more typical connection speeds.

Each PlanetLab node can be shared by up to 10 users running test simultaneously. For this reason PlanetLab nodes sometimes have poor performance and become unstable. Software deployed on PlanetLab nodes is executed within virtual containers and have limited disk space and working memory. PlanetLab offers no guaranteed service level, and the reliability and uptime of nodes varies greatly. The software and scripts deployed on PlanetLab should therefore be designed with robustness in mind. Because nodes may also be rebooted at any time, PlanetLab nodes should not be relied upon for persistent storage. Measurement data should rather be transferred to a more reliable storage system regularly.

3.4 Measurement Procedure

Each PlanetLab node was configured to run a test batch every hour and the starting time was randomized by using the `hostid` function for seeding a pseudo-random number generator. Each website was measured using four different DNS resolution methods in the following order: iterative resolution, Google DNS, OpenDNS, and the default DNS resolver as configured in `/etc/resolv.conf`. Each node measured the ICMP RTT to all of the 13 DNS root servers. Based on the RTT measurements a DNS configuration was created such that each node would prefer the roots servers closest to it in terms of RTT. The root server configuration of each node was updated every morning at 9:00. All measurements were only done using IPv4 as this ensured more reliable results on PlanetLab.

Test #	DNS Servers	Cached Responses
1	Iterative resolution	Disallow
2	Google DNS	Disallow
3	OpenDNS	Disallow
4	Default	Disallow
5	Iterative resolution	Allow
6	Google DNS	Allow
7	OpenDNS	Allow
8	Default	Allow

Table 3.4: Hourly testing sequence.

In order to measure the use of HTTP caches, each measurement was carried out with caching allowed and disallowed. This was controlled using the cache-control HTTP header, which tells HTTP caches whether they are allowed to return a cached response or if the request should be sent to the origin HTTP server. Table 3.4 shows the sequence of measurements performed every hour. Web pages were first downloaded with cached responses disallowed and then with cached responses allowed. If a cache is present on the network path, the response to the first HTTP request should be cached and a difference in download time be observed for the following download of the web page with caching allowed.

To measure the performance of the download of an entire web page, a tool called PhantomJS [33] was used. PhantomJS is a headless web browser that can load and render web pages outside of a graphical environment. The measurement application supports page element detection using regular expressions, but it is difficult to write regular expressions that capture all possible tags linking to external elements in HTML. A PhantomJS script was created that downloads a web page and outputs the URLS of files that need to be downloaded to render the page. At the beginning of each measurements batch, the script is used to generate a list of URLs that should be downloaded by the measurement application for each of the target web sites. The lists remained the same for the entire measurement batch, such that the same elements were measured with different DNS resolvers and caching options.

In this chapter we explained the measurement goals of this Master's thesis and how we intend to achieve these goals. We decided to measure the DNS and HTTP performance of popular websites in order to measure the performance of content

distribution networks. In order to measure DNS performance in detail, a DNS client was implemented. A standalone measurement application was created by combining the DNS client and the cURL library. The application was then deployed on the distributed measurement platform PlanetLab.

Chapter 4

Analysis

4.1 Measurement Data

The measurement application was deployed on PlanetLab and measurement results were collected from June 19th to July 26th 2013. During this period the DNS and HTTP performance of 23 million web page downloads were measured. Deployment occurred gradually and initially less than 10 nodes were used to test the stability of the application on PlanetLab and to detect any issues with the measurement application and the data collection system. After several days of testing, the measurement application was deployed on additional nodes. It was originally planned to handpick measurement locations in order to get a good geographical distribution of nodes. However, this approach proved to be too time consuming since many nodes were offline although the PlanetLab API reported them as being online. Instead, the measurement application was deployed on as many PlanetLab nodes as possible. Based on logging information from the data collection system, the nodes that never came online were removed from the set of active measurement nodes. After removing all offline nodes, redundant nodes hosted in the same location were removed, leaving only one measurement node per location. Finally, approximately 140 nodes remained online and measurements commenced. Figure 4.1 shows a map of the location of the nodes used in this thesis. As the figure shows, the distribution of nodes is clustered in two locations: Europe and North America. The number of nodes located outside these two areas was small in comparison.

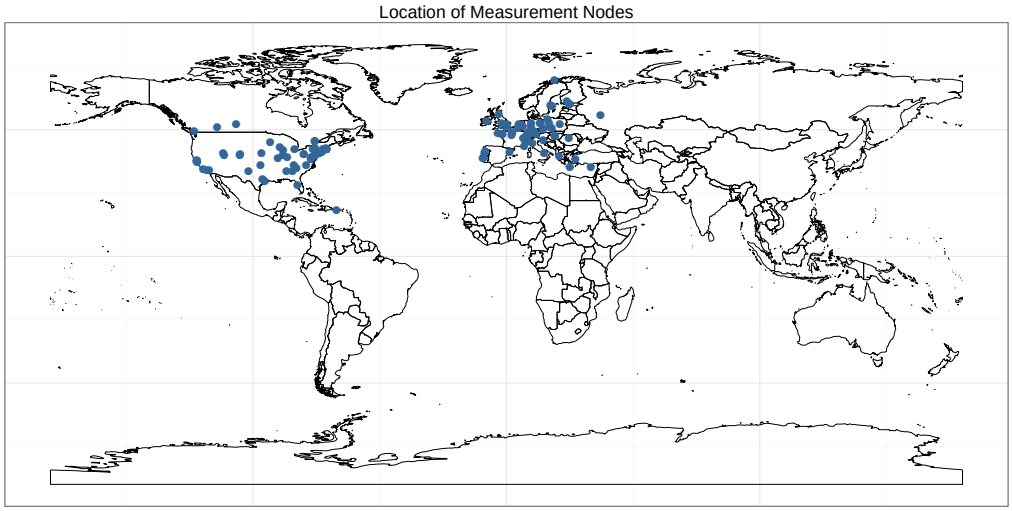

Figure 4.1: Location of measurement nodes.

During the measurement period the number of nodes submitting data to the data collection system regularly was between 110 and 134. Nodes that sometimes failed to submit data were either affected by network issues or system overload caused by tests executed by other users on the node. Nodes would continue testing even if disconnected from the data collection system and submit measurement data when connectivity was restored. Some nodes would fail to submit data for a couple of hours, some for several days.

At the end of the measurements we discovered that the measurement application had encountered problems with the cURL library for websites using TLS. The cURL package installed on PlanetLab nodes did include OpenSSL support and cURL therefore not download files over encrypted connections. The websites with URLs beginning with `https` in Table 3.1 were therefore excluded from the analysis. The data collected for the remaining websites was filtered to remove measurements from unreliable nodes. Unreliable nodes were identified by analyzing the number of measurements performed by every node. Figure 4.2 shows a histogram of the number of measurements made by a node. The plot shows that 50% of the nodes collected

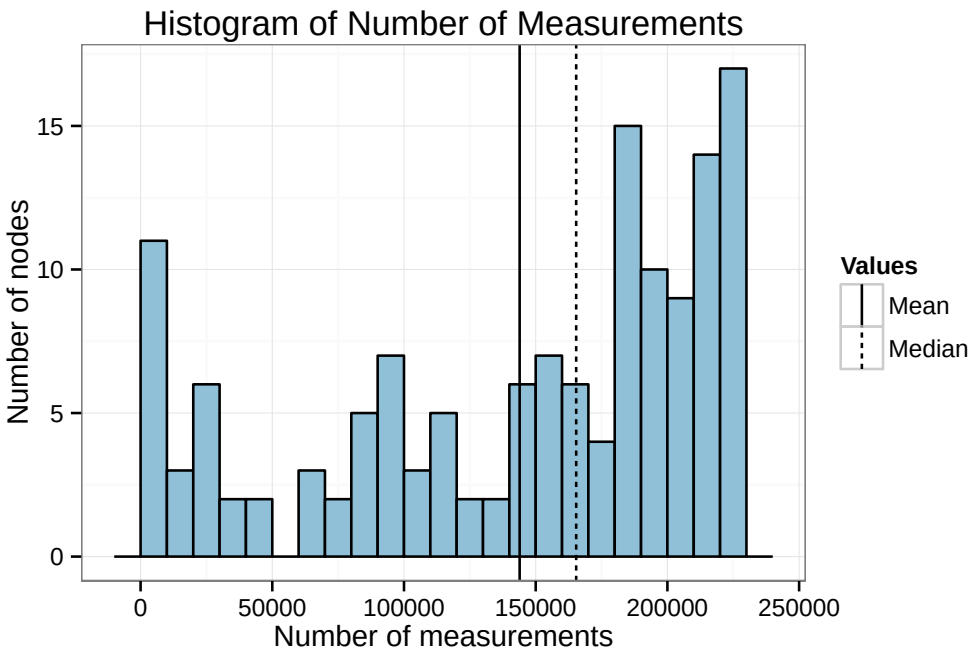

Figure 4.2: Histogram of number of measurement made by PlanetLab nodes.

more than 160,000 measurements. If a node had performed all measurements without errors, every hour throughout the measurement period, it would have submitted approximately 235,000 measurements. The plot shows that only 17 nodes consistently submitted data throughout the measurement period. We decided to ignore data from all nodes with less than 10,000 measurements because many of these nodes submitted zero measurements. Additionally, the histogram does not suggest any other logical thresholds and shows that nodes belonging to the lower half of the distribution still contributed with a significant number of measurements. Five weeks of PlanetLab measurements added up to 2.3 TB of data. In order to make data processing and analysis feasible within the time frame of this Master's thesis, we decided to analyze the week when most measurements were recorded. This was the week of July 8th 2013 and during this week 3.4 million web page downloads were measured.

The number of measurements in the selected week proved too much to analyze and the data was therefore sampled. When analyzing the data it was necessary to compensate for variations in number of measurements for each DNS resolver, number of elements per webpage, and number of measurements per website. Otherwise values

belonging to certain measurement configurations or websites would skew distributions significantly. Each analyzed metric was randomly sampled 8,000 or 10,000 times per combination of DNS resolver and website. The sampling resulted in a data set with a manageable size, while maintaining the representativeness of the entire data set. In order to improve the visualization, most of the box plots in this chapter only show first quartile, median, and third quartile of a distribution. In the few box plots that do have "whiskers", these line extend 1.5 IQR (inter-quartile range) from the first and third quartile.

In section 4.2 we analyze key performance metrics of DNS and HTTP to give a picture of the general performance of web page downloads. In section 4.3 the composition of web pages is analyzed to identify content distribution networks. The performance of the identified CDNs is then evaluated with respect to each other and non-CDN servers.

4.2 DNS and HTTP Performance

4.2.1 Resolution Time

The DNS resolution time is typically defined as the the execution time of a call to the DNS resoution function, e.g. `getaddrinfo` in C. This includes network delays, retransmissions, and processing of messages. PlanetLab nodes could experience high CPU load that affected the performance of the measurement application. In order to minimize the effects of operating system scheduling on measurement results, we decided to redefine the DNS resolution time as the sum of round-trip times including timeouts, for queries sent during the resolution process. This redefinition only changes the resolution time slightly, because the processing time of DNS messages is negligible.

Figure 4.3 shows a scatter and box plot of the DNS resolution time for each resolver. The plot shows that the resolution time of the default resolver is generally lower than for any of the other resolvers. The distribution of resolution time is asymmetric with many small values and few large values. The typical resolution time of the default resolver is 1.5 ms. When considering the typical resolution time of the other resolvers, it is clear that most queries to the default resolver were answered using cached information. The resolution time of the other resolvers is typically at

least an order of magnitude greater than that of the default resolver. The default resolver shows the largest variation in resolution time. The increased variation comes from a cache-misses where queries are forwarded to another name server or resolved iteratively. The observations above 1 ms generally fit within the range of observations for Google DNS, OpenDNS, and iterative resolution. A cluster around 11 ms suggests that the default resolver of some PlanetLab nodes might have been forwarding queries to either Google DNS or OpenDNS, or that these resolvers were queried directly.

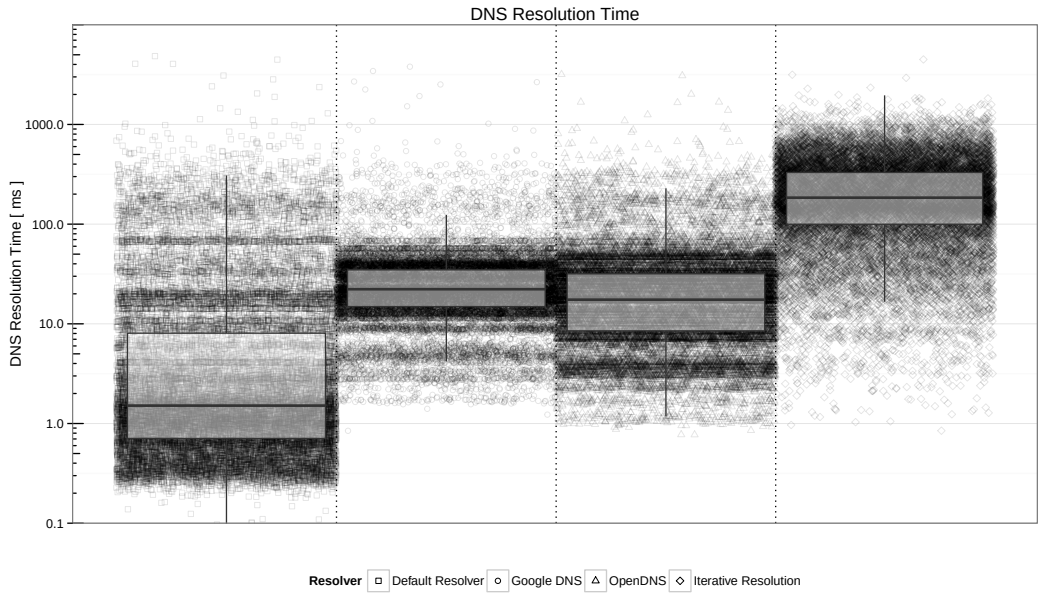

Figure 4.3: Scatter and box plot of DNS network time for each DNS resolver.

The median resolution time of Google DNS is slightly higher than OpenDNS, but Google DNS has less variation. The plot shows that OpenDNS has a cluster around 4 ms and this cluster is most likely responsible for the difference in median resolution time. When we examined this cluster of data points in detail, we discovered that the measurements were made by a few nodes located in California. The most significant component of resolution time is network latency and the nodes must therefore have had an advantage compared to the other measurement nodes. OpenDNS has a strong presence on the west coast of the United States with servers in Los Angeles, Palo Alto, and Seattle. The cluster of low resolution time was therefore likely a result of measurement nodes being connected to the Internet close to the OpenDNS servers.

Google DNS servers are hosted in 27 different IPv4 networks, whereas OpenDNS servers are hosted in 19 different networks. The extra locations used by Google DNS could explain the lower variation in resolution time, as more locations reduces the average distance between clients and servers, and thereby the average resolution time. The data supports this, as the mean resolution time of Google DNS is 3 ms smaller than that of OpenDNS.

The plot also shows that OpenDNS has more observations above 100 ms than Google DNS, and this could be a sign that OpenDNS has more cache-misses, as this group of measurements matches the typical iterative resolution time. Overall, neither of the third-party resolvers show any clear performance gain over the other with respect to resolution time.

DNS Resolver	**1st quartile**	**Median**	**Mean**	**3rd quartile**	**IQR**
Default Resolver	0.7	1.5	21.4	8.6	7.9
Google DNS	14.8	22.5	30.9	36.1	21.3
OpenDNS	8.5	17.7	33.8	32.7	24.2
Iterative Resolution	101.9	191.1	254.0	340.2	238.2

Table 4.1: Distribution summaries for each DNS resolver (milliseconds).

For both third-party resolvers the majority of the observations are smaller than the average resolution time. However, as Table 4.1 shows, the mean was 13 times greater than the median for the default resolver. In comparison the difference between median and mean was less that 100% for the remaining resolvers. The difference in skewedness between the four resolvers can be attributed to network latency. The network latency between nodes and their default resolvers is generally small. As the third-party resolvers or iterative resolution would not be able to answer a query in 15 ms, the low median resolution time is an indication of low cache-miss probability for the default resolver.

Iterative resolution was expected to have the highest resolution time, because authoritative name servers of each part of a domain name have to be queried. Table 4.1 shows that the median resolution time is two orders of magnitude greater than that of the default resolver. This clearly shows why caching is an important part of DNS and that there are significant performance benefits of using DNS caches. Although the default resolver still has the best performance in terms of resolution time, the third-party resolvers are one order of magnitude faster than iterative resolution and will therefore also result in significant reduction in resolution time.

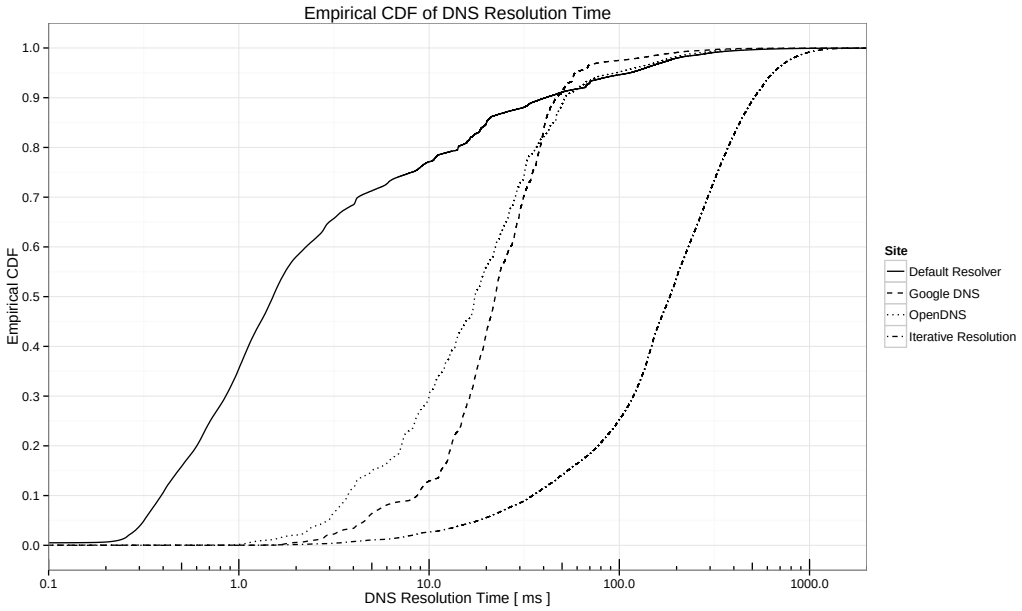

Figure 4.4: Empirical CDF of DNS resolution time for each DNS resolver.

Figure 4.4 shows the empirical CDF of resolution time for each resolver. The plot clearly shows the general performance trends for each DNS resolution method, but also reveals details about the upper 10% of the distribution. For 90% of the measurements, the default resolver always has the shortest resolution time and iterative resolution always results in the longest resolution time. The default resolver and OpenDNS have equivalent performance for the upper 10% of the observations, whereas fewer high resolution times are observed for Google DNS. The iterative resolution time can be considered the worst case resolution time as the most servers are queried. The difference between the graphs of the third-party resolvers and iterative resolution can be attributed mainly to network latency. The number of authoritative name servers for a domain typically decreases for each level of the domain name, e.g. the root domain has 386 authoritative name servers [34], `fi` has 9 servers, and `aalto.fi` has 4 servers. More name servers translates into lower average distance between client and server, if servers are hosted in multiple locations for redundancy. Iterative resolution time is higher because of the number of queries sent and the fact that the network latency of a query likely increases for each iteration of the resolution process.

The cache-miss probability is also an important factor, but given the number of users of Google DNS and OpenDNS, it is reasonable to assume a low probability for the domain names of the measured websites, except for the Finnish news websites. The 10% of the observations where Google DNS outperforms the default resolver is likely the a byproduct of the measurement process itself. Any of the PlanetLab nodes or regular visitors of the Finnish news sites using Google DNS, could potentially have primed the cache of Google DNS just before a PlanetLab node queries it. The likelihood of several PlanetLab nodes querying the same Google DNS server is higher than several nodes sharing the same default resolver. Depending on the TTL of DNS records for the Finnish news sites, the default resolver will have a higher cache-miss probability for the Finnish domains and thereby a higher average resolution time. This is supported by the fact that most of the Finnish news sites had TTL values smaller than the time between DNS requests made by the measurement application (1 hour). The default resolver would therefore never have the information in cache.

The performance of the default resolver and OpenDNS are equivalent for the top 10% of the observations, and since the default resolvers typically use iterative resolution, it can be inferred that so does OpenDNS. If this is true, Google DNS must be doing something besides iterative resolution, or more domains are simply cached by Google due to its larger number of users. A way of improving the cache-miss probability is pre-fetching of DNS records or sharing cached information between servers in different regions.

4.2.2 Record TTL

In order to evaluate the performance of DNS caches for popular websites, it is necessary to consider the lifetime of DNS records. The lifetime of records and the number of users of a DNS cache determines its cache-miss probability.

Figure 4.5 shows a scatter plot of the TTL of the first A record in the final DNS response for each of the resolvers. The plot only shows observations less than 1 hour as 95% of the observations were under this threshold. When considering the observations for iterative resolution it can be seen that most records have a TTL of 1 hour, 15 minutes, 10 minutes, or less than 5 minutes. When comparing the iterative resolver to the recursive resolvers, it is clear that the record TTL still falls into the same clusters, but that there are many data points between these clusters. This is a result of the TTL value of cached records being decremented continuously.

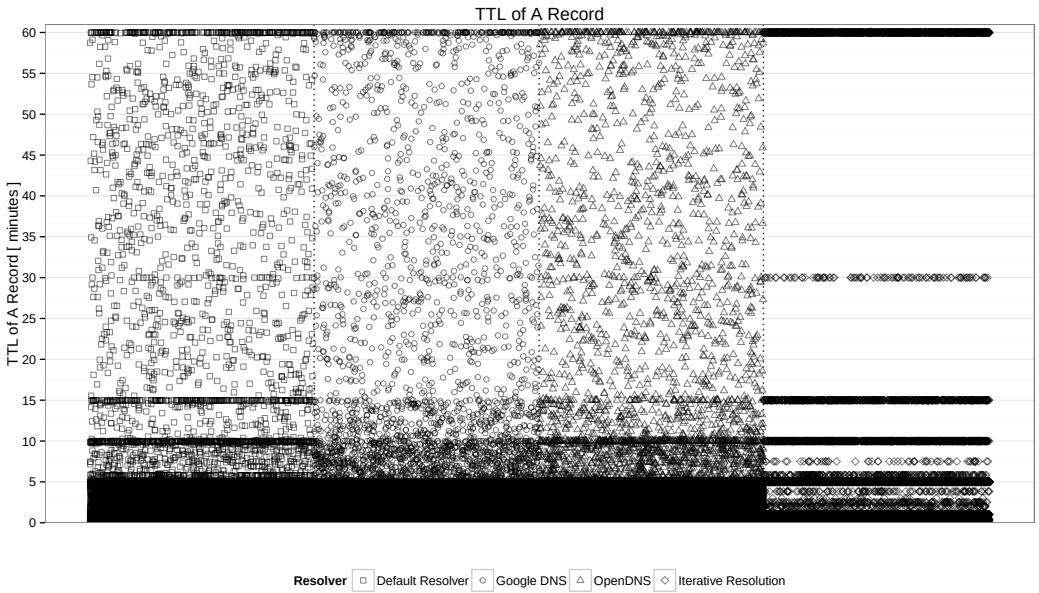

Figure 4.5: Scatter plot of A record TTL less than 1 hour for each DNS resolver.

This behavior ensures that DNS caches do not respond with information that is outdated. When the TTL reaches zero the record is flushed from the cache and a newer version of the record will be retrieved upon the next request for the record. Figure 4.6 shows the empirical CDF of the TTL values and gives a better picture of the distribution. The effects of the TTL decrementing are evident from the smooth curves between the TTL clusters. 25% of the observations are less than 20 seconds, and 90% are less than 10 minutes. None of the measured websites used TTL values that would reduce load on the DNS infrastructure significantly. Instead the DNS configurations were geared towards frequent changes in DNS information. This is likely a sign that the websites are using content distribution networks and DNS load balancing.

The CDF plot also shows performance differences between the recursive resolvers with respect to caching. In the bottom 30% of the observations it can be seen that the default resolver typically receives A records with a higher TTL than the third-party resolvers. Given that the third-party resolvers have more users than the default resolvers, the tendency of higher TTL comes from cache-misses that trigger iterative resolution. Because of the variation in record TTL for different domains it is not possible to conclude anything about the cache-miss probability of the recursive

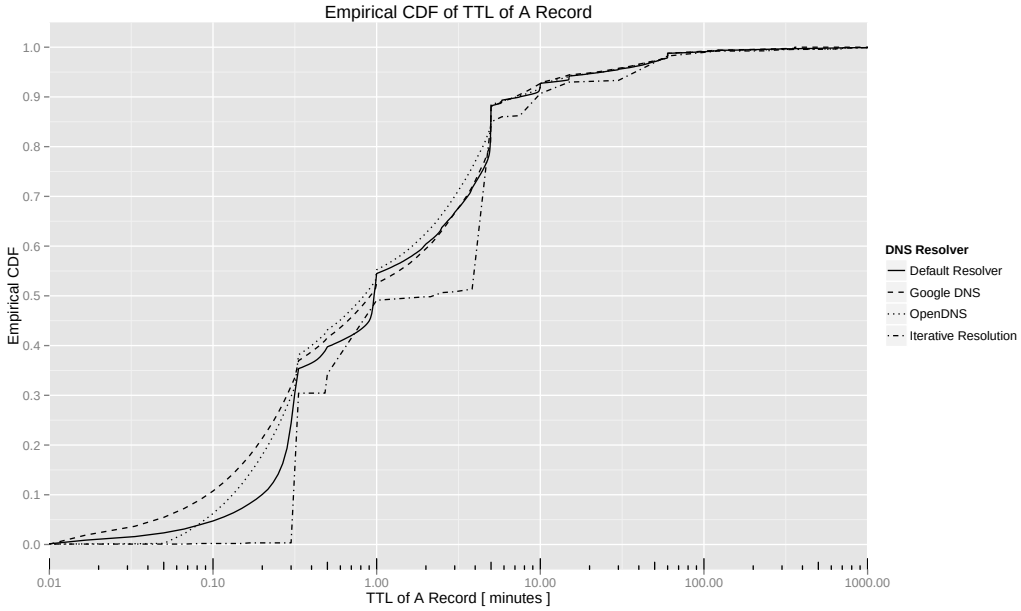

Figure 4.6: Empirical CDF of A Record TTL.

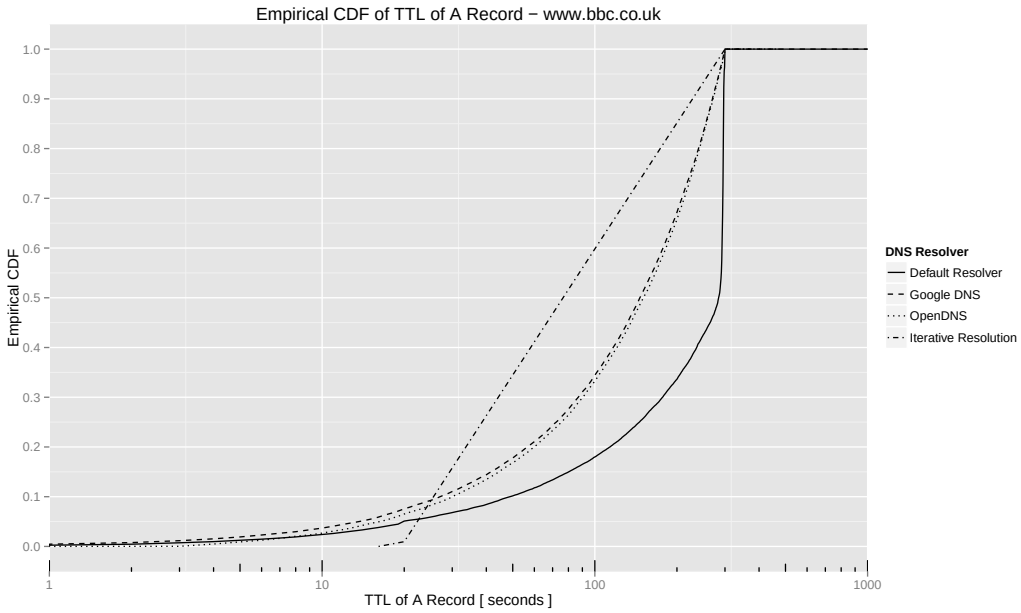

Figure 4.7: Empirical CDF of A Record TTL for the domain www.bbc.co.uk.

resolvers using Figure 4.6. By considering the TTL of a single A record for a single domain it is possible to evaluate the performance of a DNS cache. Figure 4.7

shows the empirical CDF of A record TTL only for the domain `www.bbc.co.uk`. The figure shows that almost 100% of the observations are 300 seconds for iterative resolution. Queries to the recursive resolvers that return a record with a TTL of 300 seconds must therefore indicate a cache-miss. The plot shows that Google DNS and OpenDNS have equivalent performance, and that their cache-miss probability is almost zero for the BBC domain. The default resolvers have a cache-miss probability of 50% for the domain.

DNS Resolver	**1st quartile**	**Median**	**Mean**	**3rd quartile**	**IQR**
Default Resolver	19	58	459	264	245
Google DNS	15	56	357	256	241
OpenDNS	16	51	509	223	207
Iterative Resolution	20	150	774	300	280

Table 4.2: Summary of A record TTL (seconds).

4.2.3 Connect Time

Comparing network performance by throughput is complicated by the fact that TCP performance depends on file size, bandwidth, latency, and packet loss. TCP is not able to take advantage of the entire link capacity for small file transfers, but for large file transfers the link can be filled and transfer of large files therefore give a better picture of the available capacity. Network latency is an unbiased measure of the quality of a server selection. It is not a perfect measure, but congested network paths will typically have increased latency or packet loss due to queuing. Because CDNs have servers in many parts of the Internet they will be able to direct users to servers in uncongested parts of the Internet. There is a tradeoff between congestion and latency when it comes to server selection. The use of an alternate server is only beneficial if the latency to this server is not too high.

Figure 4.8 shows the empirical CDF of the connect time reported by cURL. The connect time is measured as the time it takes to establish a TCP connection without sending any data. The connect time is therefore either two or three round-trip times depending on the implementation of TCP, as the final ACK message of the TCP handshake may contain payload data. The TCP implementation is expected to be identical across all PlanetLab nodes and the relationship between the connect time and the two-way network latency should therefore be fixed. The connect time can therefore be used to compare the performance of servers in terms of network latency.

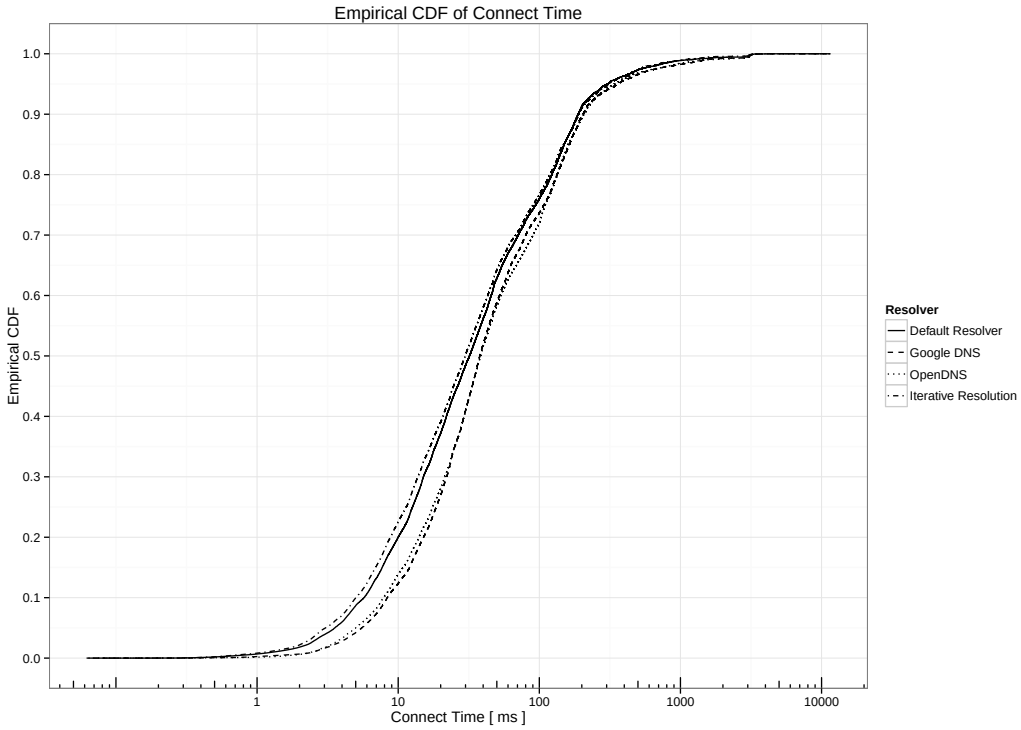

Figure 4.8: Empirical CDF of TCP connect time.

The plot shows that connect time for Google DNS and OpenDNS is equivalent, suggesting that neither service provides more optimal server selection. Iterative resolution appears to have marginally better performance than the default resolver, but overall the performance of the two are equivalent. Table 4.3 shows the distribution summary of connect time. The median and mean tell that all resolvers have many small connect times, but that Google DNS and OpenDNS observe more high connect times, than the default resolver and iterative resolution. Iterative resolution results in the lowest connect time and the default resolver provides equivalent performance.

DNS Resolver	**1st quartile**	**Median**	**Mean**	**3rd quartile**	**IQR**
Default Resolver	13	32	98	94	82
Google DNS	19	38	116	108	89
OpenDNS	17	38	116	110	93
Iterative Resolution	11	30	93	91	79

Table 4.3: Summary of connect time (milliseconds).

4.3 Content Distribution Networks

4.3.1 Identifying Content Distribution Networks

Considering the popularity of the web sites measured, we expected that most of the websites offload traffic to content distribution networks. Sites were not expected to offload *all* content, but, but at least some content like JavaScript libraries and images. As explained in the Chapter 2, CDNs typically use DNS to direct clients to an optimal server. By merely looking at the URL of page elements, it is not possible to determine whether an element is hosted on a CDN or not, because the domain name in the URL may be an alias for another domain name. By analyzing the DNS trace collected as a part of the measurements, it is possible to detect the effective domain name of an alias by analyzing CNAME records. We found that by grouping elements by their effective domain name, it was possible to determine which services were actually hosting content.

In order to compare the use of CDNs across all measurement nodes we stripped away part of domain names used for DNS redirection by the CDNs. For instance, the domain names used by Akamai servers typically have the format `a1733.g.akamai.net`, where the `a1733` part of the domain name play some unknown role in the server selection process. The effective domain names of page elements were reduced to their two top-most labels, such that e.g. `a1733.g.akamai.net` became `akamai.net`. This aggregation provided sufficient information to identify CDN providers. The same approach was also tried with the three top-most labels, but this did not provide any additional insights. Because recursive resolvers do not necessarily return CNAME records, the CDN analysis was based on the measurements for iterative resolution.

Figure 4.9 shows the first, second, and thirds most popular domains per page with respect to the number of elements hosted on the domain and its sub-domains. The plot shows where page elements were actually delivered from. For instance, most elements on the web site of Helsingin Sanomat were delivered from domain names ending in `hs.fi` which is the main domain name of the web site. The second most popular source was `adtech.de` which provides banner ads, and the third most popular source was `akamai.net` which is a well-known domain name used by the Akamai CDN. This tells that most of the page elements of Helsingin Sanomat are *not* hosted on a CDN.

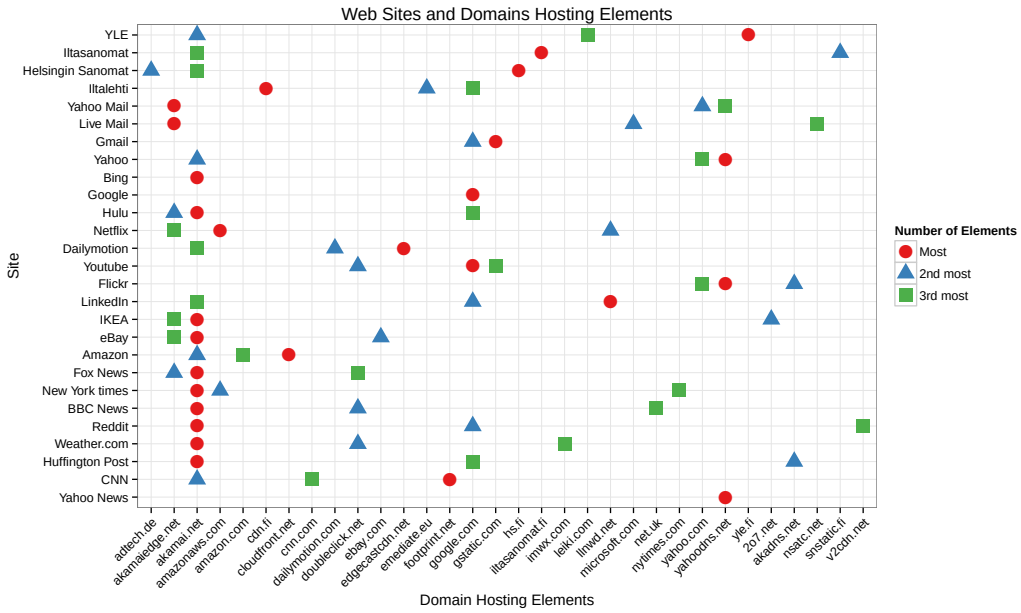

Figure 4.9: Domains hosting most page elements.

The plot shows that `akamai.net` is the most popular domain among all the web sites in terms of number of elements. All but one of the web sites have `akamai.net` or `akamaiedge.net` as either first, second, or third most popular source of content. Only Yahoo News does not contain any elements hosted by Akamai. The second domain from which many web sites download elements is `google.com`. Google provides free CDN hosting of popular JavaScript libraries and download of these, along with the use of Google Analytics is the most likely reason for websites to download files from a sub-domain of `google.com`.

Figure 4.9 shows the popularity of domains in terms of the total number of bytes downloaded from a domain. The plot shows that Akamai is still the most popular source of page elements. For the four Finnish news sites, the plot shows that all but one of the websites download most bytes from their main domains, e.g. `www.hs.fi` downloads most bytes from `hs.fi` or its subdomains. In the case of Iltalehti, most bytes came from the domain `cdn.fi` which is controlled by CyberCom Group, a company delivering website services. Considering the user base of Iltalehti, it is highly unlikely that the site is delivered from a CDN because having multiple server locations within Finland would not improve performance significantly, and using a global CDN would only move content further away from the target users of the

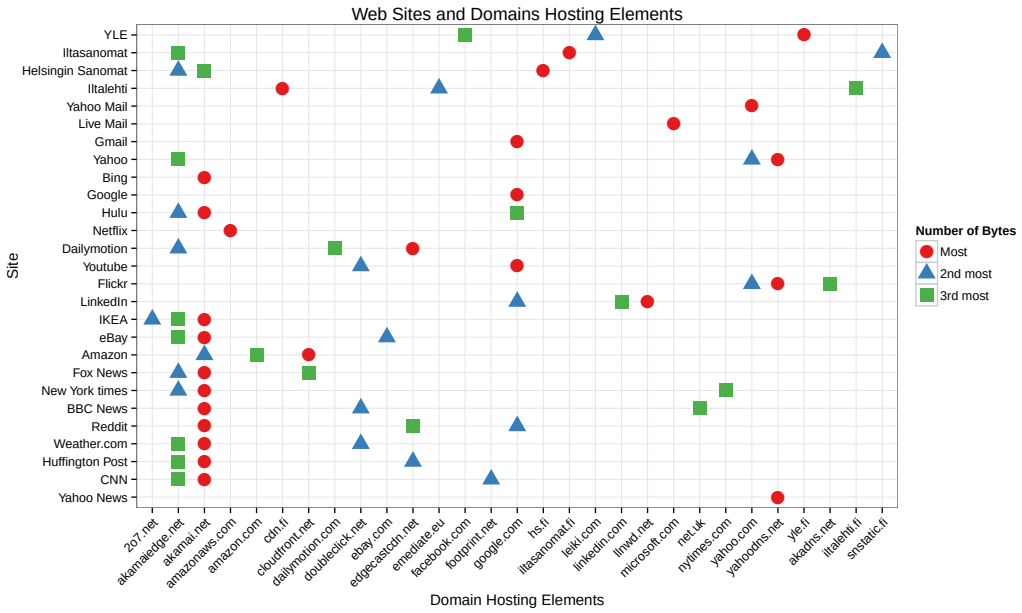

Figure 4.10: Domains hosting most most content in terms of file size.

website. By analyzing the DNS diversity of the `cdn.fi` domain it was confirmed that this is in fact not a CDN, as only a single IP address was observed globally and that this was not an anycast address.

CDN Provider	**Domain Name**
Akamai	akamai.net akamaiedge.net akamaitech.net
Amazon	amazonaws.com cloudfront.net
EdgeCast	edgecastcdn.net
Level 3	footprint.net
Google	google.com
Limelight	llnwd.net
Yahoo	yahoodns.net

Table 4.4: Identified content distribution networks.

The analysis of the effective domain names of page elements lead to ten domain names which were confirmed to be CDN providers using WHOIS information and visiting the websites of the domain owners. Table 4.4 shows the confirmed CDN domains and the name of the CDN providers controlling these domain names. Any

element hosted on these domains or their subdomains are considered to be hosted on a CDN.

4.3.2 CDN Servers

Content distribution networks rely on many servers to provide high capacity, redundancy, and geographical diversity. The IP addresses of the CDN servers identified in the measurement data were analyzed to estimate the capacity of each CDN in terms of number of servers, networks, and locations. It should be noted that the IP addresses collected may not represent all available servers of the CDNs. The PlanetLab nodes used for measurements were located in Europe and North America, and it is therefore less likely that servers outside of these areas were observed for CDNs with good server selection. Even within Europe and North America, PlanetLab nodes may not be redirected to all available CDN servers because redirections may depend on the ISP of a PlanetLab node.

Figure 4.11 shows the relationship between IP address prefix length and the number of distinct prefixes in the measurement data. The plot is based on IP addresses obtained through DNS resolution and not information about actual allocations of IP address space to network segments. For the sake of clarity the term "network" is used for distinct IP prefixes since some prefixes will inevitably be *actual* network segments. The plot visualizes the diversity in networks in which CDN servers are hosted. The increase in the number of distinct prefixes indicates how many networks the observed IP addresses belong to. A large increase shows that a CDN is served from many different networks, whereas no increase indicates that IP addresses have a sequence of bits in common. For instance, the plot shows that all EdgeCast servers are located within three "/20" IPv4 networks.

The most networks were observed for Akamai for all prefix lengths. Compared to the other providers, the networks of Akamai servers belong to many different address blocks, even for short prefix lengths. Since it is unlikely that Akamai has been allocated 70 "/8" IPv4 address blocks, Akamai servers must be hosted within many different networks in the Internet. The tail of the Akamai graph shows a big increase in networks from a prefix length of 30 bits to 31 bits. This suggests that Akamai servers are typically deployed in pairs within networks with a 30 bit network mask. It is assumed that Akamai has locations with more than two servers, because servers are deployed in locations near many potential end-users in ISP networks and

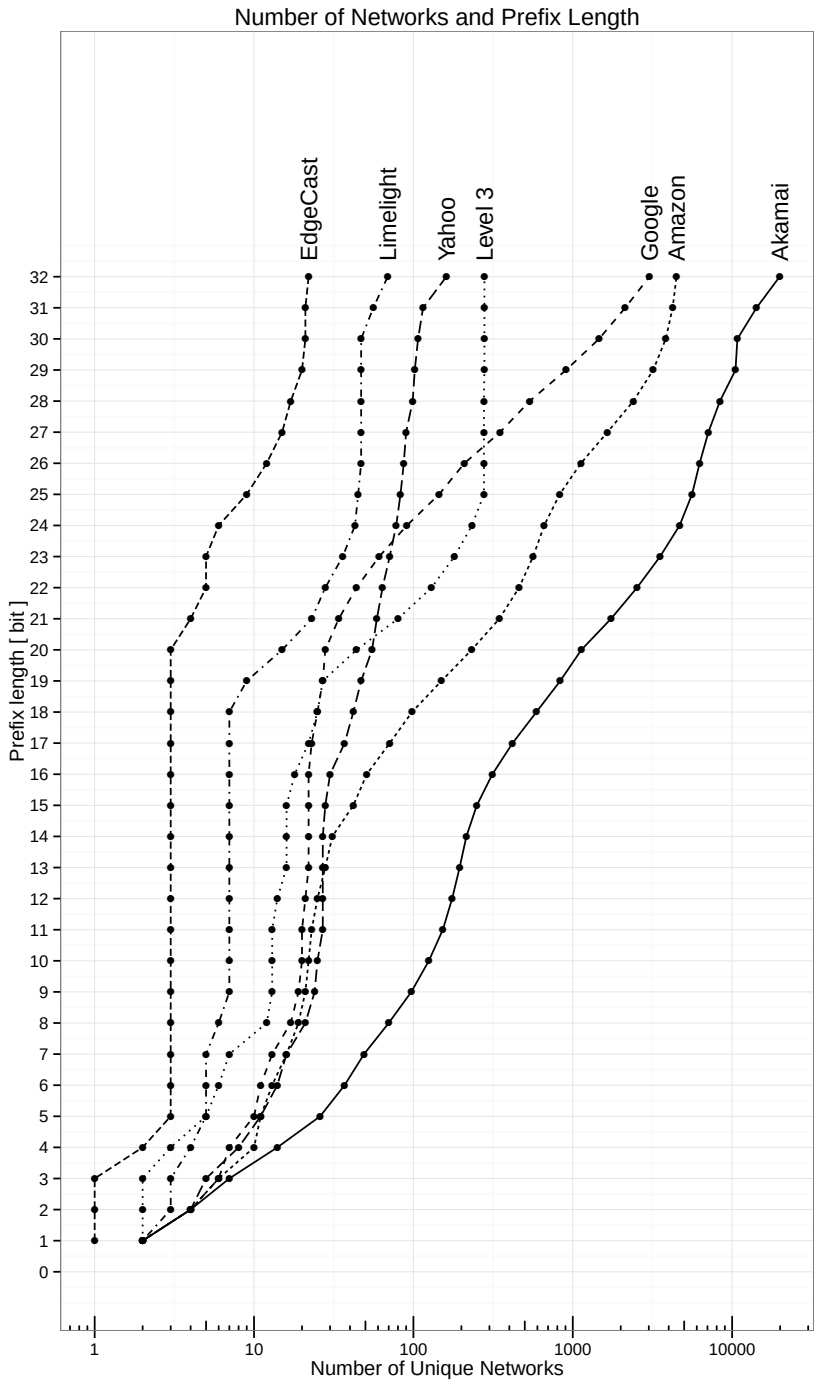

Figure 4.11: Number of unique networks depending on prefix length.

near Internet exchanges. It is possible that within large ISP networks, Akamai has multiple server locations with just two servers. The IP addresses allocated to servers might differ across the ISP network and account for the high number of IP addresses with a common 30 bit prefix. Another explanation is that due to fragmentation of the IPv4 address space, CDNs may use multiple network prefixes within a single network.

Besides revealing the number of networks CDN servers are hosted in, the plot also gives a picture of the architecture of the CDNs. The last 7 bits of all collected IP addresses were identical for Level 3. Since it is unlikely that Level 3 servers are located in separate "/25" networks at the same location or that each location would only contain a single server, the use of load balancers could explain why IP addresses have common last bits. Level 3 uses DNS to direct users to the nearest location but instead of providing the IP address of a specific physical server, the address of a load balancer is likely provided. DNS is used to distribute load globally, whereas the load balancer pointed to by DNS is used to distribute traffic across the available servers in a location. The IP addresses of Limelight and Yahoo servers also showed similarity within the last 7 bits of the addresses, so these providers may use a similar architecture.

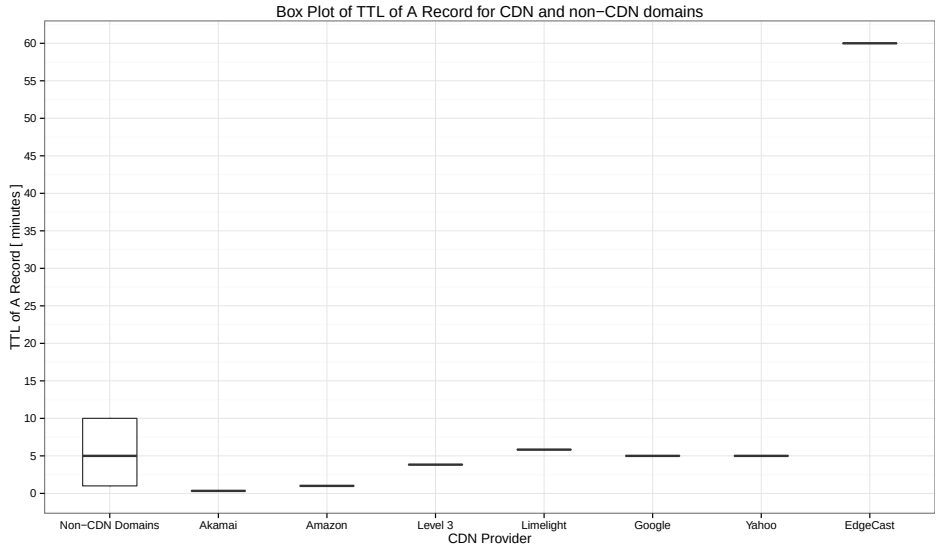

Figure 4.12: Box plot of TTL of A Records for CDN and non-CDN domains.

Thousands of IP addresses were observed for Google, Amazon, and Akamai. Since it would be unnecessary to use this many unique IP addresses to access anycast

servers, it can be concluded that these CDNs do not use anycast as their primary method of traffic distribution and load balancing. However, the remaining CDNs returned significantly fewer unique IP addresses through DNS and this could be a sign of anycast usage. We analyzed a small subset of the collected IP addresses for these CDNs using `traceroute`. By comparing the routes from a vantage point in Finland and Germany, we determined that EdgeCast uses anycast to direct clients to nearby servers, whereas the remaining CDN providers use DNS redirection.

Figure 4.12 shows a box plot of A record TTL for each CDN provider and for the domain names that were not classified as belonging to CDNs. All of the CDN providers except EdgeCast consistently returned DNS records with very low lifetime. The low TTL hints that these CDNs were using DNS redirection rather than anycast. EdgeCast on the other hand consistently used a TTL value of 1 hour and this shows that DNS information is not changed frequently and that DNS redirection is therefore not used for fine-grained server selection. Based on the typical DNS record lifetime and the traceroute analysis, it is certain that EdgeCast relies on anycast to direct clients to the nearest server. When the routes to EdgeCast servers were analyzed, it was discovered that although all the IP addresses are anycast routed there was a noticeable performance difference between IP addresses in different networks. This difference is caused by different prefixes being routed to different locations in the Internet. EdgeCast must therefore also be using DNS redirection, but instead of returning an address of a specific CDN server, and anycast address optimized for a geographical region is returned.

Overall, Figure 4.12 shows that the TTL of non-CDN domains is very low and comparable to those of the CDNs except for EdgeCast. This is due to unnecessarily low TTL values used by websites and miscategorization of CDN domains as non-CDN domains. Websites hosted on a limited set of servers with static IP addresses should not use low TTL values for DNS records as these rarely need to be changed.

In order to make the best estimate of the number of server locations, it would be necessary to analyze the routes between PlanetLab nodes and the IP addresses of the CDNs, but this information was not collected. 19,865 distinct IP addresses were collected for Akamai servers. If each address belongs to a physical server, only 15% of the total number of Akamai servers [35] were seen by measurement nodes. If Akamai servers are typically hosted in clusters of two severs, Figure 4.11 suggests that 10,000 hosting locations were observed for Akamai. Since we determined that

Akamai does not use anycast as its primary load distribution method, the number of observed IP addresses is a good measure of the number of CDN servers deployed Akamai, although the use of load balancers cannot be ruled out. The same can be said for Google and Amazon, which have CDN architectures similary with Akamai.

For Limelight, Yahoo and Level 3, the lower number of unique IP addresses and use of load balancers indicates that there is a better relationship between the number of observed IP addresses and the number of locations each CDN has servers in. Because the IP addresses returned for these three CDN providers most likely point to load balancers, it is not possible to estimate the number of available CDN servers. The use of anycast makes it impossible to estimate the number of locations for EdgeCast without analyzing traceroute or BGP data.

It is possible that the websites measured also played a role in the number of IP addresses observed for each CDN provider. Some of the measured web sites might have had more users and therefore need more CDN servers to handle traffic. However, given the popularity of the non-Finnish websites, the traffic demand of those web sites should be sufficiently high to expect content to be delivered from many CDN servers. The number of unique IP addresses is therefore a good measure to compare CDNs that do not hide servers behind load balancers by.

While Figure 4.11 reveals information about the overall structure of the identified CDNs, it does not tell anything about how specific PlanetLab nodes perceived each of the CDNs. Figure 4.13 shows the empirical CDF of the number of unique IP addresses observed by PlanetLab nodes for each CDN provider. The plot shows that although most servers were observed for Akamai globally, 85% of the nodes typically observed more distinct IP addresses for Amazon than Akamai. The remaining 15% of the nodes observed significantly more Akamai servers than Amazon servers. The highest number of servers observed by a PlanetLab node was 1,000 for Amazon, whereas the same number was an order of magnitude larger for Akamai. Five percent of the PlanetLab nodes observed more than 1,000 servers for Akamai and this indicates that these nodes were in locations far from Akamai locations. If any Akamai location had been significantly better than the others, clients would have been redirected to this locations more often and thereby have observed fewer servers.

I could be expected that because Akamai has many servers deployed in many differ-

ent networks, the average number of servers observed by a node would be similar to those of CDN providers with more consolidated architectures, such as Level 3 and Limelight. However, the plot shows that nodes observe noticeably more servers from CDN providers with many servers in many locations. A CDN location is therefore not reserved for end-users within a particular network, but the capacity can also be taken advantage of by clients outside the network. This makes server deployment inside ISP networks more profitable as an increased number of reachable users increase server utilization.

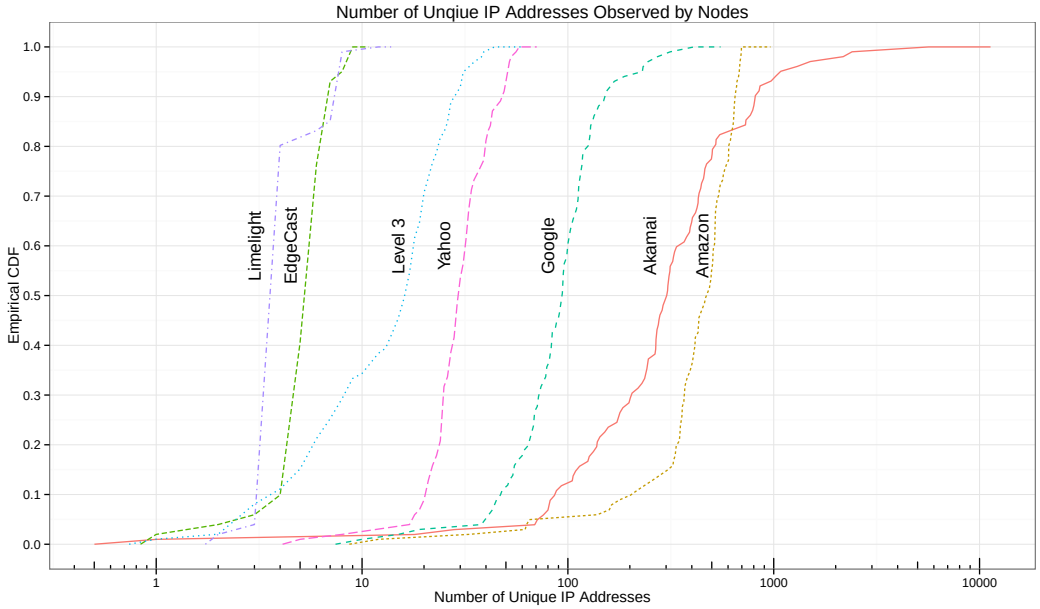

Figure 4.13: Empirical CDF of number of unique IP addresses observed by each node.

4.3.3 Resolver Accuracy

All of the seven CDNs identified in the measurement data relied on DNS to redirect users to servers optimal with respect to their estimated location. DNS is therefore an important component in the architectures of these CDNs and may affect the user experience positively or negatively depending on the accuracy of the DNS redirection. Iterative resolution always queries an authoritative name server of a domain and for CDNs this should result in the best possible server being selected as the source IP address of the DNS query is not masked by a recursive resolver. The IP address

in the A record obtained through iterative resolution is therefore used to estimate the accuracy of the answers by the default resolver, Google DNS, and OpenDNS. In order to measure the accuracy of the DNS resolvers, the concept of IP address similarity was defined as the length of the longest prefix two IP addresses have in common. For instance, the addresses `192.168.0.1` and `192.168.0.2` have an IP address similarity of 30 bit. The resolver accuracy was defined as IP address similarity between the address returned by iterative resolution and the address returned by another resolver.

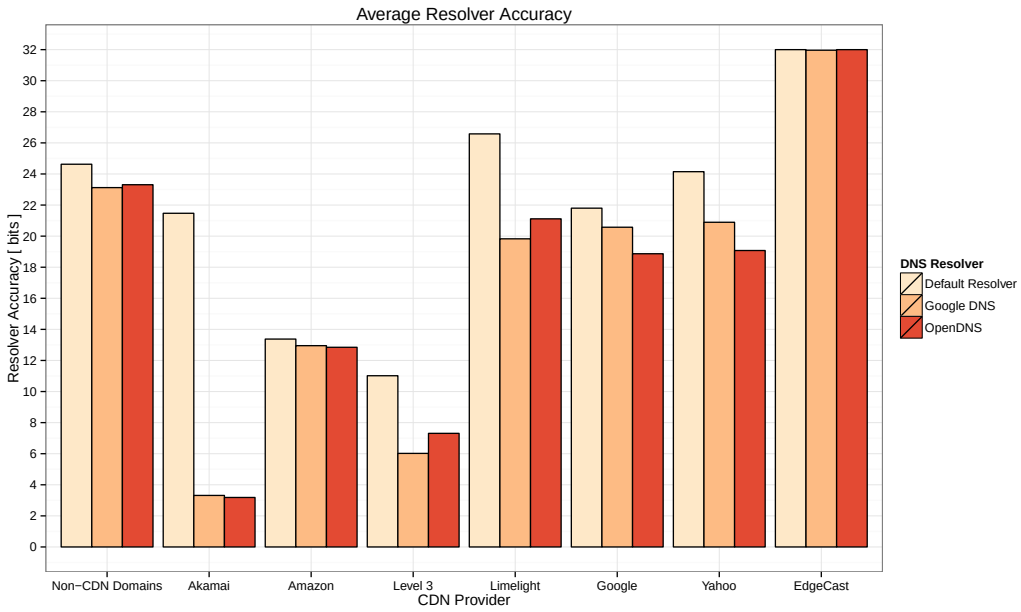

Figure 4.14: Average accuracy of DNS Resolvers.

Figure 4.14 shows the average resolver accuracy of CDN and non-CDN domain names. The accuracy for the non-CDN domains was expected to be higher, as websites hosted on a limited number of servers should have consistent DNS information. The lower accuracy is most likely due to incorrect classification of CDN and non-CDN domains, and DNS servers shuffling multiple A records in DNS responses. For all of the identified CDNs, the plot shows that name resolution using Google DNS and OpenDNS results in sub-optimal server selection, except for Amazon and EdgeCast. The IP address returned by the default resolver is generally more similar to IP address returned by an authoritative name server of a CDN provider. Surprisingly, the plot shows that the accuracy of Google DNS and OpenDNS is far worse for Akamai than for any of the other CDN providers. This may be a con-

sequence of the distributed architecture of Akamai and the high number of unique IP addresses returned by the authoritative DNS servers of Akamai. Any location-dependent server selection made by Akamai appears to be broken by Google DNS and OpenDNS. In comparison, a much smaller difference in resolver accuracy is observed for Level 3, Limelight, and Google. Although the server selections by Google DNS and OpenDNS could be better for these CDNs, the selections are still much better than those made for Akamai.

CDN Provider	**Europe**	**North America**	**Combined**
Akamai	9242	12373	19865
Amazon	2095	2765	4477
Google	1864	1204	3027
Level 3	122	190	279
Yahoo	85	123	161
Limelight Networks	28	41	69
EdgeCast Networks	12	10	22

Table 4.5: Summary of Unique IP addresses by CDN provider and continent.

Since CDN locations are not necessarily reserved for end-users in the network they are deployed in, the collected IP addresses were analyzed for overlapping server selections between Europe and North America. Is is assumed that due to the geographical distance between Europe and North America, users on either continent should preferably never be redirected to a CDN server on the opposite continent. If a node in Europe observed the same IP address as a node in North America, either node must have been redirected to a CDN server located on the opposite continent or neither continent. CDNs do not base their redirections purely on the estimated location of clients, but also consider other factors that affect the quality of service. A server with high latency is still better than an overloaded server in a congested part of the Internet.

Table 4.5 shows a summary of the number of unique IP addresses observed by nodes on the two continents and by nodes globally. The table shows that Limelight and EdgeCast were the only providers that did not have any overlap in DNS redirections between the Europe and North America. This suggest that the location-based DNS redirection worked perfectly and that none of the PlanetLab nodes were located in border zones of the CDNs. Naturally the number of CDN locations increases the risk of incorrect DNS redirection because the geographical regions mapped to IP addresses are smaller.

Figure 4.15 shows the percentage of overlapping IP addresses depending on prefix length. The plot shows that the CDN providers with many server locations have less than 10% overlap in DNS redirections. Given the combined number of servers for Akamai, Amazon, and Google, the DNS redirection of these CDN providers appears to work very well. Almost 30% of Yahoo IP addresses were observed in both Europe and North America, and this indicates that Yahoo servers lack geographical diversity compared to the other CDNs. Yahoo has three data centers in the United States and one in Singapore [36] and this could explain the higher number of overlapping IP addresses. Since IP addresses observed on the two continents did not fully overlap, Yahoo must have servers deployed in Europe in undisclosed locations. No overlapping IP addresses were observed for EdgeCast and this indicates that EdgeCast uses different anycast addresses in different regions. Finally, the small overlap in IP addresses for Google, Amazon, and Akamai can most likely be attributed to their many server locations. End-users are more likely to be redirected to a CDN server on the same continent, even if the redirection is not perfect. Furthermore, it is possible for CDNs with many server locations to direct users away from overloaded servers or congested network paths without sending them to another continent.

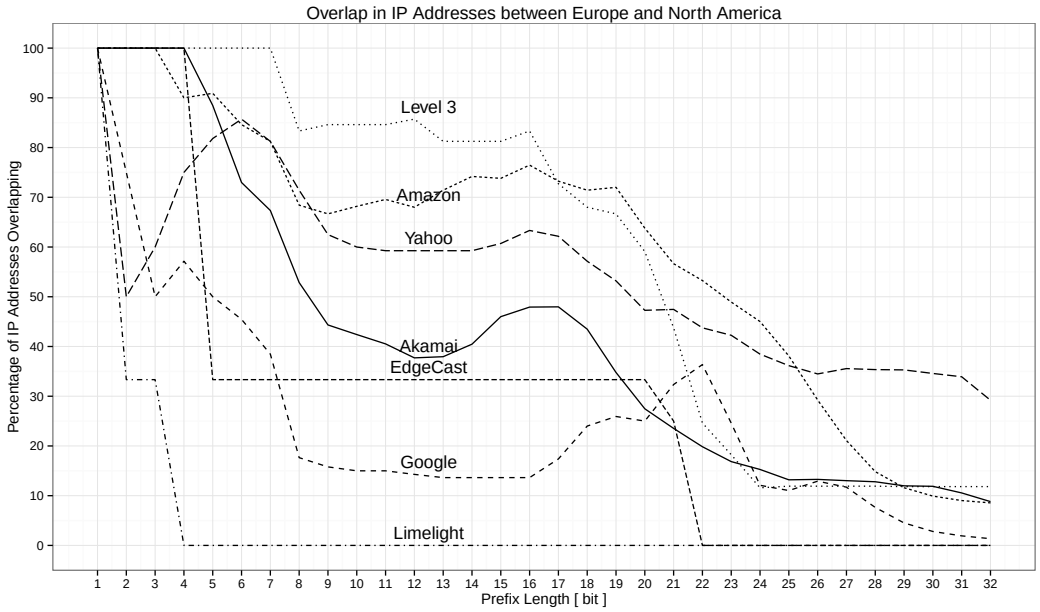

Figure 4.15: Percentage of IP addresses observed in both Europe and North America depending on prefix length.

Figure 4.15 shows several peaks between prefix lengths of 16 and 22 bits. These

peaks could indicate that overlapping servers share a common IP prefix. Akamai servers that are observed on the two continents come from "/16" networks. An IP prefix of this size could suggest that Akamai uses data centers for overflow traffic between continents, rather than hosting locations inside e.g. ISPs. The peak for Google at 22 bit prefix length suggests that the overlapping servers come from the same Google data center.

4.3.4 Connect Time

CDN providers claim that their infrastructures enable faster delivery of content to end-users in terms of both throughput and latency. To determine the validity of these claims, we analyzed the delivery performance of website elements. Figure 4.16 shows a box plot of the connect time for CDN servers and non-CDN servers for each of the DNS resolution methods. The latency between clients and non-CDN servers was typically higher than for the CDN servers. For non-CDN servers iterative resolution results in a median connect time 7 ms lower than the other resolvers. This confirms that some of the non-CDN domains must belong to CDNs using DNS redirection. The connect time of the default resolver and iterative resolution is generally comparable for the CDN servers except for Limelight which had a 12 ms difference.

Akamai servers had the lowest latency for iterative resolution. The plot also shows that despite Akamai redirections being noticeably worse for Google DNS and OpenDNS, the seemingly poor server selection only caused a 10 ms increase in connect time compared to iterative resolution. This suggests that the DNS redirections of Akamai are actually not broken by Google DNS and OpenDNS. Google DNS is experimenting with the "Client Subnet in DNS Request" extension for DNS [24], but since Google DNS did not have a resolver accuracy of CDN domains comparable with the default resolver, it would appear that this feature did not result in more accurate DNS answers. DNS queries using the client subnet extension do not contain the full IP address of the original client. This may explain why the accuracy is lower, but even if only the first 24 bits of the original client's IP address were provided, the resolver accuracy should still have been better than what was observed. The combination of DNS redirection and anycast routing could also have explained why poor CDN redirections did not result in higher connect time, but Akamai does not use anycast [37]. The most likely reason for the discrepancy between resolver accuracy

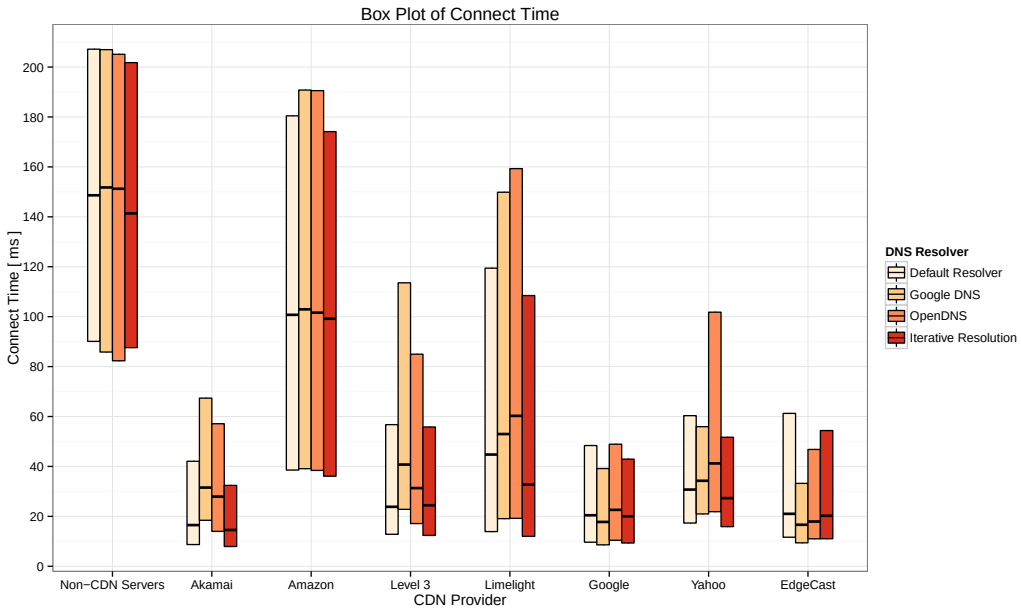

Figure 4.16: Box plot of connect time for CDN server and non-CDN servers.

and connect time for Google DNS and OpenDNS is the location of the PlanetLab nodes and their Internet connectivity, or that the CDNs do not support the client subnet extension yet. Akamai has also been experimenting with application layer redirection where poor DNS redirections are detected by measuring RTT during the TCP handshake [38]. If the RTT indicates that the client was directed to an sub-optimal CDN server, the server will automatically find a optimal server and return the address of this server through the application protocol. Because TCP connections typically do not run through proxies, the IP address of the client is not masqueraded as with recursive DNS resolution. Akamai is therefore able to make the best possible server selection using the full IP address of the client. Akamai states that the added time of application layer redirection mean that it is only beneficial to redirect clients requesting large files. Given the typical size of files hosted by Akamai for the measured webpages, it is unlikely that Akamai used HTTP redirection.

The use of Google DNS results in the lowest connect time for Google CDN servers. This indicates that Google DNS is optimized for Google services as even iterative resolution results in a higher median connect time. Normally an authoritative name server of Google would be expected to give the best possible answer. Since this is not the case, Google DNS must have additional information about the state of the

CDN and thereby be able to make better server selections.

4.3.5 Throughput

The congestion control of TCP affects the throughput of HTTP differently for different file sizes. Figure 4.17 shows a box plot of file sizes for each CDN. Different services provide different content and hence the variations in file sizes between the CDNs. The typical file size is different for each of the CDNs and this complicates throughput comparison. In order to minimize the TCP behavior affecting throughput, we decided to compare the throughput of the CDNs for file sizes in two small intervals.

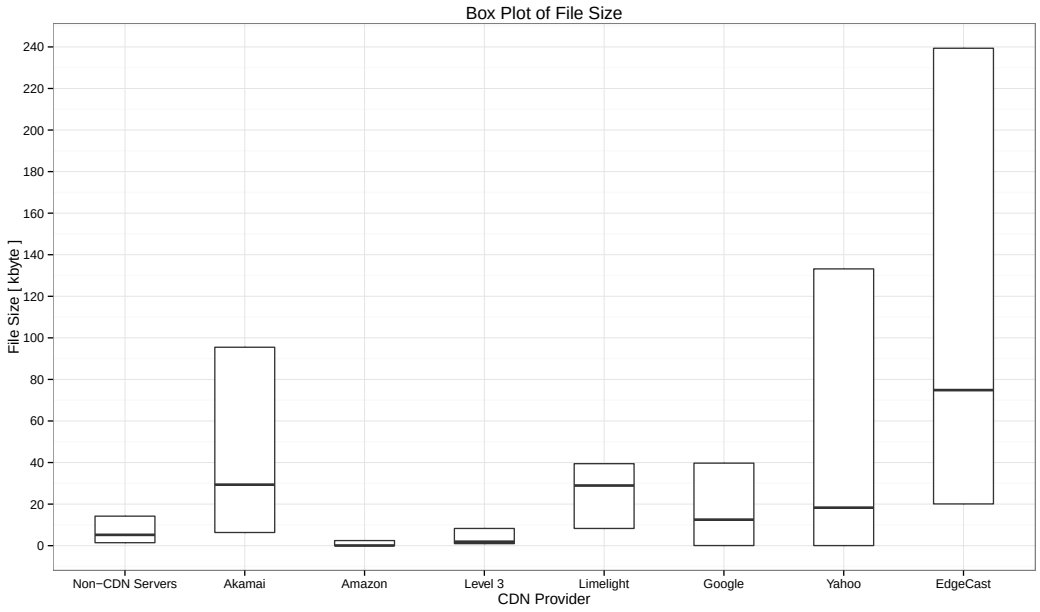

Figure 4.17: Box plot of the file size of files hosted on each CDN.

Figure 4.18 shows the empirical CDF of HTTP throughput of the CDN servers and non-CDN servers. The upper part shows throughput for small files between 5 and 15 kilobytes, and the lower part shows the throughput for larger files between 95 and 105 kilobytes. The plot for the large file sizes is missing some CDN providers as file sizes in the selected range were not observed for these providers. The overall tendency for the two plots is that larger files are transferred at a higher bitrate than smaller files. The throughput for small files is better for all the CDNs compared to files coming from non-CDN servers. The use of a CDN therefore does improve

the download time of web pages with many small elements. The performance of the CDNs fall into three groups for small files. Amazon has a median throughput of only 200 kbps and Limelight, EdgeCast, and Yahoo had a median throughput of 400 kbps. Akamai, Google, and Level 3 had a median throughput of approximately 1.3 Mbps. Despite having a lower number of estimated locations, Level 3 is still able to compete with Akamai and Google when its comes to throughput for small files. This shows that a distributed CDN architecture does not necessarily result in better throughput.

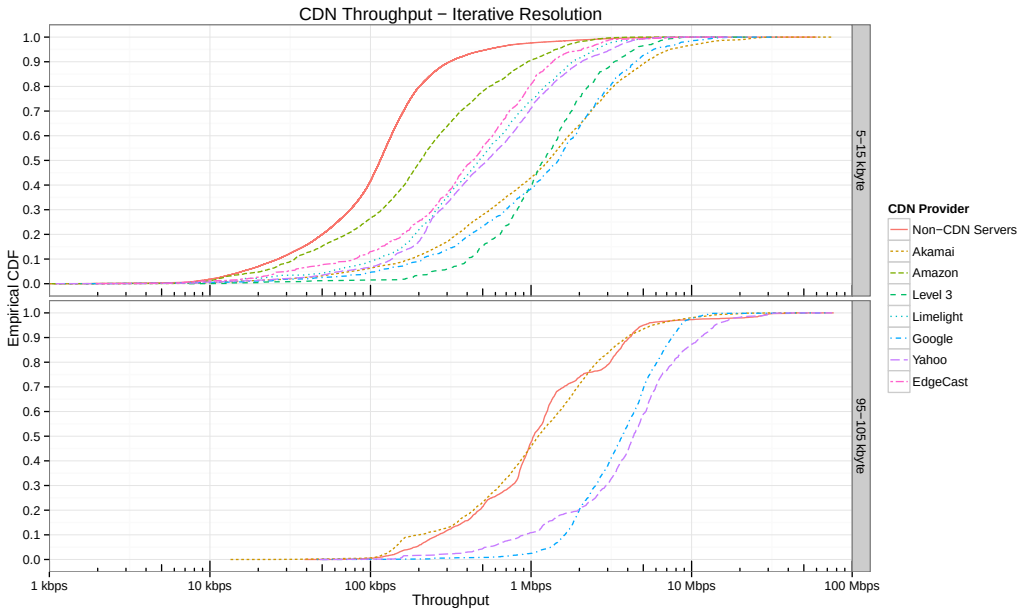

Figure 4.18: Empirical CDF of throughput for small and larger files.

The throughput of Akamai appears to be the same for the small and larger files, and the throughput of larger files is comparable to that of non-CDN servers. Compared to Google and Yahoo, Akamai does not enhance content delivery of larger files for web pages. Google and Yahoo show significantly better performance for larger files than non-CDN servers and the median throughput is 2 Mbps and 3 Mbps, respectively. The differences in throughput suggests that the Akamai, Yahoo, and Google CDNs are optimized for different file sizes. Figure 4.17 shows that Yahoo is the only CDN where the 95-105 kilobyte range lies within the inter-quartile range of the file size distribution. Since Yahoo outperforms Google and Akamai for larger files, the Yahoo CDN is likely optimized for delivering larger files, rather than small files. Google had similar performance with Akamai for smaller files, but significantly

better performance for larger files. This indicates that the Google CDN performs well in terms of throughput for both small and larger files. Given the many different file sizes observed for Akamai, it is surprising that there is a noticeable difference in throughput for small and larger files. It is possible that this is related to the measured websites and the content they provide.

In this chapter we analyzed the performance of a set of popular websites based on measurements performed by PlanetLab nodes. We determined that network latency affects DNS resolution time the most, and that using a recursive DNS server close to the client results in the best possible DNS resolution time. By comparing the the DNS resolution results of four different resolution methods we determined that the proximity of a recursive DNS affects resolver accuracy. The proximity of a recursive resolver affects the similarity of the obtained IP address with the best possible server selection obtained by querying an authoritative name server of a CDN domain. However, when the TCP connect time was analyze, we discovered that dissimilar addresses do not necessarily translate into higher connect time. Finally, we evaluated the throughput of the identified CDNs and determined that CDNs are likely optimized for different file sizes. Akamai and Google performed well for small files, but for larger files Akamai had performance similar with non-CDN servers. Google on the other hand still had significantly better throughput that non-CDN servers for large files.

Chapter 5

Discussion

The analysis of DNS resolution time showed that the latency between client and servers has the greatest impact on DNS performance. The best reason for using a third-party resolver is lower cache-miss probability compared to the recursive resolver of a home gateway. However, depending on the TTL of records the third-party resolver will only outperform the default resolver for the first query. If the default resolver performs iterative resolution it will also cache information about the authoritative name servers for a domain. These NS records may have substantially longer lifetime than the final A record. The default resolver will therefore be able to direct subsequent queries directly to authoritative name servers and thereby skip several resolution steps. Thus third-party resolvers do not necessarily have an advantage for records with short TTL values.

Currently Google DNS and OpenDNS cannot compete on resolution time with DNS caches in home gateways or ISP networks, but DNS resolution is about more than just resolution time. Google DNS and OpenDNS claim to provide better security and this is an important aspect to consider when choosing DNS resolver. For instance, Google DNS validates DNS records using DNSSEC and will not return records where the authenticity cannot be verified for domains which use DNSSEC. From a security perspective, third-party resolvers provide a better service as security enhancements are implemented sooner than by ISPs and home gateway manufacturers. It can therefore be argued that if the latency between clients and DNS servers were reduced, the use of third-party resolvers would be preferable from a security standpoint, even at the cost of slightly higher resolution time.

The content distribution networks identified in the measurement data used different architectures to improve content delivery. Six out of seven CDNs only used DNS redirection, and one CDN used both DNS redirection and anycast to direct users to optimal CDN servers. DNS is preferred over anycast by most of the identified CDNs because it is not possible to make fine-grained load balancing using BGP. Akamai claims that their CDN is able to direct traffic away from overloaded servers within seconds [37]. If a CDN only relied on anycast to distribute traffic across servers, it could happen that a surge in traffic within a region would overload servers in a hosting location. In that case, the only way of directing traffic away from the overloaded servers would be to announce new BGP information. However, the granularity of BGP information means that changes in information does not necessarily lead to changes in routing. If a BGP announcement does affect the routing in the Internet, there is a risk that the traffic will not be distributed over a larger number of servers, but instead overload another hosting location. DNS provides much more accurate control than what is possible with BGP. EdgeCast therefore uses both DNS and anycast to redirect users to optimal servers. This allows EdgeCast to benefit from the simplicity of anycast and react better if servers become overloaded for an extended period of time. In an overload situation EdgeCast would be able to update its DNS information as quickly as Akamai, but the 1 hour TTL of its DNS records means that clients already directed to the overloaded servers will be stuck with these for up to 1 hour. Because Akamai uses a much lower TTL value, clients are only stuck with overloaded servers for a matter of minutes or tens of seconds. Overload therefore has a more severe impact on the performance of the EdgeCast CDN. The only way to avoid overload situations is to over-provision servers and redirect users via DNS well before overload occurs. CDNs which only use DNS redirection are able to operate with higher server loads than EdgeCast, because traffic patterns can be changed more rapidly.

Of course there are also benefits of using anycast compared to only using DNS redirection. Because anycast works independently of DNS, third-party resolvers cannot break the server selection of the CDN. The limited number of regional anycast addresses also enabled EdgeCast to make perfect DNS redirections. Although the low resolver accuracy of the third-party resolvers for Akamai did not appear to affect the TCP connect time significantly, it still highlights the main problem of DNS redirection. To overcome the problem of recursive resolvers hiding the IP address of the end-user, Akamai could start using both unicast and anycast addresses for its

servers. Queries identified as coming from Google DNS or OpenDNS would then receive an anycast addresses rather than a unicast addresses. However, this would be complex and likely impossible to implement in all locations where Akamai hosts servers. Pushing for the adoption and deployment of the "Client subnet in DNS requests" extension is a more feasible solution for Akamai. It is possible that CDN providers with fewer server locations would be able to use both unicast and anycast, and thereby overcome the problem of poor DNS redirections by third-party resolvers.

The measurement application worked very well on PlanetLab except for the OpenSSL issue. Although package management was taken into consideration when setting up measurement nodes, we did not foresee that the default cURL package could come without OpenSSL support. In hindsight, this problem should have been detected early on, but because the data processing tools had not yet been written this did not happen. Initially there were also some problems with the output format of the measurement application and these had to be fixed during the measurement period. Automating validity checking of measurement data would have been able to detect problems earlier. We learned that when using many measurement nodes it is necessary to have a good system for managing and monitoring nodes. Logging information collected by the data collection and deployment systems was used to check if nodes were functioning properly. The data collection system also sent daily statistics and alarms in case certain statistics dropped below a threshold.

The design of the measurement application resulted in behavior that was different from that of web browsers. The cURL library lacked the ability to override DNS resolution and the integration with the DNS clients was therefore not optimal. We determined that the only solution to this problem is to implement an HTTP client with this feature. Additional hook points that will enable measurement of more precise performance metrics than cURL should also be implemented.

Although the collection of measurement data was automated, the processing of this data was only semi-automatic. When dealing with data in the order of terabytes, we realized it would have been beneficial to process data shortly after it is submitted to the data collection system. The longer data processing is postponed, the greater the need of data storage becomes and the more obvious the I/O limitations of hard drives become.

The main reason why data processing became a challenge was that many more per-

formance metrics were collected than were necessary. We chose to collect additional metrics in order to create a data set that can be used in future research of web-page delivery performance and content distribution networks.

The analysis of resolver accuracy and connect time showed that poor resolver accuracy did not translate directly into higher connect time. It would be interesting to establish a relationship between resolver accuracy and HTTP performance metrics, such that the impact of using third-party DNS resolvers can be quantified better.

We confirmed that DNS redirection is the most popular solution for CDN server selection. CDNs use multiple levels of redirection and in order to understand the operation of CDNs, it would be interesting to analyze the redirection process in detail. The full trace of messages sent and received during DNS resolution was collected and this information will be useful in future CDN research. Visualizing DNS redirections as a graph could previously reveal unseen details about the structure and operation of content distribution networks. We also observed that many websites had peculiar DNS configurations. The DNS traces can also be used to determine potential performance gains by correcting improper DNS configurations.

Chapter 6

Conclusions

In this Master's thesis we developed an application for measuring the performance of DNS and HTTP downloads. The application was successfully deployed on the distributed measurement platform PlanetLab and used to measure a set of popular websites. The measurement data was used to evaluate the performance of different DNS resolution methods and HTTP download. Finally, a method of identifying content distribution networks from a global perspective was developed and applied to the measurement data. The performance of DNS and HTTP was then analyzed with respect to CDNs.

Domain names were resolved using iterative resolution, Google DNS, OpenDNS, and the default DNS resolver of the measurements nodes. The performance of HTTP was evaluated in terms of TCP connect time and throughput. The DNS measurements showed that network latency has the biggest impact on DNS resolution time. The default resolver used by measurement nodes resulted in the shortest resolution time in 90% of the measurements. Despite having a lower cache-miss probability, the resolution time was only shorter for Google DNS in 10% of the measurements and OpenDNS showed no improvement over the default resolver. The performance of Google DNS and OpenDNS was comparable, and iterative resolution consistently resulted in the highest resolution time.

Most of the measured websites used content distribution networks and the choice of DNS resolver affected the network latency between client and server, due to the use of DNS redirection. The default resolver had the best performance for CDN servers as the latency to servers was comparable to iterative resolution, but the

resolution time was an order of magnitude smaller. While identifying CDNs, it was determined that the TTL of DNS records is not a useful indicator when it comes to identifying CDN servers, because CDNs that use anycast will likely use higher TTL values. An effective method of identifying CDN servers using CNAME records was developed and used to identify the most popular CDNs used by the measured websites. Through the global perspective of the measurements, seven CDN providers and the IP addresses of their servers were identified. Analysis of these IP addresses showed that different providers employed different architectures relying on DNS redirection, anycast, and load balancers. In addition, it was observed that the websites often relied on more than just a single CDN provider.

The A records retrieved by the default resolver were a closer match to those returned by the authoritative name servers of the CDNs. However, the analysis of connect time revealed that third-party resolvers did not result in higher connect time to CDN servers. This was especially surprising for Akamai which had the worst resolver accuracy of the seven CDNs. Further research is needed to establish a relationship between resolver accuracy and application-layer performance. The analysis of HTTP throughput showed that despite the difference in the number of estimated CDN locations, Akamai and Level 3 have comparable throughput for small files. Finally, it was observed that content distribution networks are likely geared towards delivering different file sizes, as Akamai had noticeably worse throughput for larger files, whereas Google performed well for both small and larger files.

References

[1] Cisco Systems Inc., "Visual Networking Index Forecast Highlights." http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights/index.html. Viewed on November 25, 2013.

[2] J. S. Otto, M. A. Sánchez, J. P. Rula, and F. E. Bustamante, "Content Delivery and the Natural Evolution of DNS: Remote Dns Trends, Performance Issues and Alternative Solutions," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, (New York, NY, USA), pp. 523–536, ACM, 2012.

[3] P. Mockapetris, "Domain Names - Implementation and Specification," November 1987. RFC 1035.

[4] ICANN, "Factsheet - Root server attack on 6 February 2007," March 2007. http://www.icann.org/en/about/learning/factsheets/factsheet-dns-attack-08mar07-en.pdf. Viewed on November 25, 2013.

[5] P. Vixie, "Extension Mechanisms for DNS (EDNS0)," August 1999. RFC 2671.

[6] CERT, Carnegie Mellon University, "Multiple DNS implementations vulnerable to cache poisoning." http://www.kb.cert.org/vuls/id/800113. Viewed on November 25, 2013.

[7] R. Arends and R. Austein and M. Larson and D. Massey and S. Rose, "DNS Security Introduction and Requirements," March 2005. RFC 4033.

[8] D. Atkins and R. Austein, "Threat Analysis of the Domain Name System (DNS)," August 2004. RFC 3833.

[9] ICANN Research, "TLD DNSSEC Report." http://stats.research.icann.org/dns/tld_report/. Viewed on November 9, 2013.

[10] VeriSign Inc., "DNSSEC Scoreboard." http://scoreboard.verisignlabs.com/. Viewed on November 9, 2013.

[11] J. K. Chen, "Google Public DNS: 70 billion requests a day and counting," February 2012. http://googleblog.blogspot.fi/2012/02/google-public-dns-70-billion-requests.html. Viewed on November 25, 2013.

[12] OpenDNS Inc., "OpenDNS System." http://www.opendns.com/technology/traffic-stats/. Viewed on November 25, 2013.

[13] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig, "Comparing DNS Resolvers in the Wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, (New York, NY, USA), pp. 15–21, ACM, 2010.

[14] Alexa Inc., "Alexa - The Web Information Company." http://www.alexa.com/. Viewed on November 25, 2013.

[15] World Wide Web Consortium (W3C), "HTML 4.01 Specification," December 1999. http://www.w3.org/TR/REC-html40/.

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

[17] D. Fisher, "HTTP Pipelining FAQ," March 2005. https://developer.mozilla.org/en/docs/HTTP_Pipelining_FAQ. Viewed on November 25, 2013.

[18] BrowserScope, "BrowserScope." http://www.browserscope.org. Viewed on November 25, 2013.

[19] R. Pries, Z. Magyari, and P. Tran-Gia, "An HTTP web traffic model based on the top one million visited web pages," in *Next Generation Internet (NGI), 2012 8th EURO-NGI Conference*, pp. 133–139, 2012.

[20] J. Abley and K. Lindqvist, "Operation of Anycast Services," December 1999. RFC 4786.

[21] Google Inc., "Google Public DNS Frequently Asked Questions." https://developers.google.com/speed/public-dns/faq. Viewed on November 25, 2013.

[22] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: A platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 2–19, Aug. 2010.

[23] T. Repantis, J. Cohen, S. Smith, and J. Wein, "Scaling a Monitoring Infrastructure for the Akamai Network," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 20–26, August 2010.

[24] C. Contavalli, W. van der Gaast, S. Leach, and E. Lewis, "Client Subnet in DNS Requests," July 2013. `http://tools.ietf.org/html/draft-vandergaast-edns-client-subnet-02`.

[25] P. W. Gilmore, "Akamai & ISPs," *25th UK Network Operators' Forum*, April 2013. Akamai Inc., `http://indico.uknof.org.uk/getFile.py/access?contribId=13&resId=1&materialId=slides&confId=27`.

[26] Akamai Inc., "Accelerated Network Partner Program." `http://www.akamai.com/html/partners/network_partner.html`. Viewed on Janaury 10, 2013.

[27] P. W. Gilmore, "Peering with Content Distribution Networks," February 2012. Akamai Inc., `http://ixleeds.net/files/ixleeds1/presentation/patrick_gilmore.pdf`.

[28] S. Triukose, Z. Wen, and M. Rabinovich, "Measuring a Commercial Content Delivery Network," in *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, (New York, NY, USA), pp. 467–476, ACM, 2011.

[29] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante, "Drafting Behind Akamai: Inferring Network Conditions Based on CDN Redirections," *IEEE/ACM Trans. Netw.*, vol. 17, pp. 1752–1765, December 2009.

[30] Stephen D. Strowes, "Passively Measuring TCP Round-trip Times," October 2013. `http://queue.acm.org/detail.cfm?id=2539132`.

[31] D. Stenberg, "cURL." `http://curl.haxx.se/`. Viewed on November 25, 2013.

[32] PlanetLab, "PlanetLab Network Requirements." `https://www.planet-lab.org/node/222`. Viewed on November 21, 2013.

[33] A. Hidayat, "PhantomJS." `http://phantomjs.org/`. Viewed on November 25, 2013.

[34] Root Server Technical Operations Association, "List of Deployed Root Servers." http://www.root-servers.org. Viewed on November 25, 2013.

[35] Akamai Inc., "Facts & Figures." http://www.akamai.com/html/about/facts_figures.html. Viewed on November 9, 2013.

[36] Yahoo Inc., "Yahoo Data Centers." http://www.ydatacentersblog.com/. Viewed on November 25, 2013.

[37] Akamai Inc., "Serving at the edge: Good for performance, good for mitigating DDoS - Part II," April 2013. https://blogs.akamai.com/2013/04/serving-at-the-edge-good-for-performance-good-for-mitigating-ddos-part-ii.html. Viewed on December 2, 2013.

[38] Akamai Inc., "Intelligent User Mapping in the Cloud," March 2013. https://blogs.akamai.com/2013/03/intelligent-user-mapping-in-the-cloud.html. Viewed on December 2, 2013.