Abstract

In this work, we compare average page load times for the top 500 Alexa websites across CDNs and ASNs.

Based on our algorithm, we estimated that 300 websites are using CDNs. We find that Google CDN gives the best performance with an average total response time of 0.48s. However, multiple requests to google.* domains are in fact redirected since Google detected that we are using a script to measure response timings. The worst performer was Alibaba CDN. In general, sites using CDNs outperformed those not using CDNs when comparing median time taken to load the complete page by 0.2s. Sites not using CDNs take 5x the time for TCP handshake, and 3x the time for SSL negotiation, when compared to those using CDNs. Time taken for DNS lookup, and WAIT time for server response is same across both groups.

Grouping by ASN, we find that Google and Microsoft have the best performance, while Alibaba and ChinaNet have the worst performance. This is consistent with our timing analysis when grouped by CDNs. We found that 74.67% of detected CDN names correspond directly to the AS name. Two websites, liputan6.com and trello.com were using ASNs different from the detected CDN used to host most content on their webpages. Furthermore, we found 22 websites that were not using CDNs, hosted on ASes that also offer CDNs if needed.

Introduction

CDN Analysis

Not all websites using CDNs directly mention it on their homepage. Furthermore, multiple objects on the homepage may be using different CDNs. Thus, estimating the absolute CDN and testing its performance for a website is an open problem. For our analysis, we parsed the homepage of 500 top Alexa websites to estimate the domains that were using CDNs to host a majority of the static objects. We also looked at whois records for websites that we weren't sure about, since certain whois records mention the CDN name within the whois records. Further details and improvements on this method are detailed in the section below and the README.md file.

We found that **302 of 500 sites were using CDNs**, 198 were not. 18 of 198 sites were blocked by our ISP, while 2 of 302 sites were blocked, but directly resolved to a CDN name based on their domains: googleusercontent.com and cloudfront.net.
Of 302, **163 were resolved to CDNs by parsing objects** on a page, while **139 sites had a known CDN provider mentioned in the organization name of their whois records**, but did not show any CDN related domain on parsing the page.

163 sites were using 62 confirmed and well known CDNs. Of these **Google hosted 60, Cloudfront 17, Akamai 8, Alibaba 8, accounting for 93 websites amongst 4 CDNs**. Of 139 websites detected on 24 CDNs through whois records, **34 used Cloudflare, 24 used Amazon, 22 were on Fastly, and 18 on Akamai**. Apart from this 7 more websites were using Amazon Data Services, 4 were using Alibaba/Taobao, and 8 were using Google Cloud but were not detected by parsing objects on their pages directly.
Altogether, major CDNs (Google, Akamai, Cloudflare, Amazon, Fastly, Alibaba) account for 210 of 302 sites using CDNs. 42 sites used well known CDNs (such as Open-Connect, Incapsula, Facebook, etc.). 50 sites were using unknown CDNs (i.e., the url contains the string "cdn" but we haven't resolved it to a specific site).
Of 198 sites not using CDNs, 83 sites still hosted most static content on a certain domain that didn’t seem to be a cdn (such as "assets.tumblr.com", "cfl.dropboxstatic.com", etc.).

Page Load Timing Analysis

We used CURL to perform 100 requests to each of the 500 websites and calculated the average times between various events: t_dns, t_tcp, t_ssl, t_calc, t_wait, t_fbyte, t_rx. The time_total is the sum of all the individual event times. The t_wait and t_calc refer to the time taken for the client between the completion of SSL negotiation and the time it received the first byte, and the time it takes for the server to parse the HTTP GET request and start its response, respectively.

We found that out of 50,000 requests to 500 domains, only **23898 requests returned a response code of 200 OK**, while **16090 responses were redirects** (status code 301/302). **7098 requests completely failed** or timed out (response code 000), and only a few requests were forbidden by the server. For the purpose of timing analysis, we consider these responses as successful (we also analyzed responses after allowing curl to redirect until receiving a 200 OK response in a separate notebook).
A **majority of the redirects (status code 302) were due to google.* domains** that detected we were using a script and asked us to manually confirm our IP address on visiting the homepage. In fact, **only 30% of requests to google.* domains returned a 200 status OK** response, while the rest redirected us together regardless of the google site we visited. This behavior was only shown by google.* type domains (google.com, google.co.in, etc.) and not for other services hosted on Google, such as Youtube (see figure results/google_vs_others_response_code.png).
49 of 500 websites completely failed on using curl, i.e., they only returned a 000 response. These sites and their 4900 requests were filtered out from our timing analysis. There was also a certain loop in our requests where almost all responses failed. This might be due to a problem at the ISP or a fault in the network for one round of requests (see figure results/curl_requests_filtered_status_code_pdf_hist.png).

Overall, we found that across the 42902 successful responses, the **average total_time across all responses for loading the website was 1.03 seconds** (including both 200 OK responses

as well as redirects). However, considering the median time instead, the 50 percentile time_total across requests is only around 0.8s.
The overall average downloaded data was 102 KB, and it took .25s to receive this data, making the average **overall speed across requests around 3.25 Mbps**. The overall DNS resolution and TCP handshake are quite fast (0.08s, 0.095s), but SSL negotiation took 0.24s on average. **The most major contributor to the total time was surprisingly the waiting time, averaging 0.37s across successful requests.** This means that the server genuinely waits and computes for around 0.3s (removing the transit time ~0.08s) on receiving an HTTP GET request from our curl request client. For only responses with a 200 OK status code, the average total time was 1.27s.

The above data was when we loaded websites without following any redirection. If we follow redirects, the average load time for successful responses increases to 1.53s (and to 1.65s if we consider only 200 OK status responses). 0.3s of this is average time taken for redirection.

ASN Analysis

We used python's socket library to resolve domains to IP addresses. We found that our ISP blocked 20 of the 500 top Alexa websites. These were either porn related or malware/adware/redirect viruses that infect browsers.
192 ASNs hosted 480 websites. 149 of these ASNs only hosted 1 website, while 43 hosted multiple sites. The top 10 ASes, based on number of sites, hosted a total 262 websites. 6 ASes had more than one ASN in the dataset, while the rest had only one ASN hosting sites.

Table 1: Number of sites and individual ASNs per AS for top 10 ASes.

AS Name	Akamai	Alibaba	Amazon	ChinaNet	Cloudflare	Facebook	Fastly	Google	Microsoft	Yahoo
#sites	21	18	58	9	37	3	30	69	8	9
#ASNs	3	2	2	4	1	1	1	1	2	6

Data Collection

Page load times

- We use curl to request the website homepage multiple times and return timings for various events. We calculate times for separate events in a connection based on these event times and get the average across multiple requests.
- We run the curl request to each website at least 100 times with a random sleep time in between to avoid overloading the website. We also changed the user agent string so that sites are unable to detect scripted requests. Despite this, many websites detected our curl based script and redirected us.

- Many curl requests resulted in 300 status codes, redirecting us to confirmation pages. We do a quick analysis for such websites by allowing curl requests to redirect (using the -L flag), which increases the "redirect time" for our request, but allows us to get a 200 OK status response.
- We follow curl's definition of times from https://ec.haxx.se/usingcurl-verbose.html. Individual event times from the diagram are calculated as follows:
  - t_dns [DNS] = time for DNS resolution (no redirects) = time_namelookup - time_redirect
  - t_tcp [TCP] = time for TCP connection (SYN/SYNACK) = time_connect - time_namelookup
  - t_ssl [SSL] = time for SSL handshake (only if https) = time_appconnect - time_connect
  - t_calc = time between SSL handshake acknowledgement received and issuing GET request (time for client calculation) = time_pretransfer - time_appconnect
  - t_wait [WAIT] = time between issuing GET request and first byte received (time for server calculation and transit) = time_starttransfer - time_pretransfer
  - t_fbyte [TIME TO FIRST BYTE] = time_starttransfer
  - t_rx [RX TIME] = time to receive data from first to last byte = time_total - time_starttransfer

CDN estimation

- We list 4 ways to estimate the CDN a website is using:
  a. Parse the website homepage and check if the url they were downloaded from is a CDN. We can do this by comparing urls to an offline list of well known CDNs.
  b. Parse the whois records of the website IP address. The CDN name is usually in the "Organization" field in whois records.
  c. Download all objects on the website and check if their HTTP headers contained the string "x-cache" HIT or MISS. Most CDNs generally use this header string to specify where the object was downloaded from.
  d. Get the DNS CNAME records for the website and check if the CNAME is a CDN.
- In our current analysis, we first check if the domain itself is a well known CDN. Then, we use method (a) and (b). For ensuring that a majority of objects are on a CDN, we count static objects in HTML tags on the webpage, determine the domains hosting these objects, and count for a majority of hosted objects > 50% of all objects per CDN. If there is no majority, even if multiple objects are scattered across known CDN domains, we assume that there is no single absolute CDN that the website can be attributed to, and count that no single CDN was used (see estimate_cdn_by_parse() method in estimate_cdn.py for details). This method slightly underestimates the CDNs used.
- Method (c) can be implemented by using selenium scripts to parse and download complete web pages. Method (d) is also simple to implement by performing online DNS queries. These can be added in the future when needed.
- An offline list of CDN names was populated from wikipedia and CDN ranking websites. A hash table between urls and CDN names was found on WPO Foundation's "webpagetest" page. We copied these resources to utils/CDNdomains.py

ASN

- We use python’s internal socket library to get a website’s IP address, and cymruwhois to get the AS number and AS name corresponding to the IP. There are many other alternative geoIP libraries that can do the same.

Analysis

Page Load Time Analysis

The following table shows the overall timing analysis with average and median times over all successful requests per website, as well as the maximum time taken for each event (and the website responsible). Surprisingly, **washingtonpost.com has a large server waiting time of 8.47s, while the total time and time to first byte is worst for bestbuy.com (8.6s).**
The **largest name lookup was due to rakuten.co.jp (1.56s).** On observing in detail we found that name lookup for rakuten seems to alter between a lower time and a higher time between consecutive loops. Two other sites (naver.com and kompas.com) showed similar behavior with name lookup (t_dns) and receive time (t_rx) increasing the standard deviation for these event times (see figure results/cdf_time_diff_calculated_3sites and figure results/timeseries_3sites_dns_rx_time).

Table 2: Comparison of event time (average per website) overall mean, median, max, and site (max).

	time_namelookup	time_connect	time_appconnect	time_pretransfer	time_starttransfer	time_total	t_dns	t_tcp	t_ssl	t_wait	t_rx	t_fbyte
mean	0.08	0.18	0.42	0.42	0.8	1.05	0.08	0.1	0.24	0.38	0.25	0.8
median	0.02	0.1	0.28	0.28	0.65	0.77	0.02	0.04	0.12	0.32	0	0.65
max	1.55	1.56	2.29	2.29	8.6	8.6	1.56	0.4	1.63	8.47	5.92	8.6
Site max	rakuten.co.jp	rakuten.co.jp	babytree.com	babytree.com	bestbuy.com	bestbuy.com	rakuten.co.jp	alipay.com	taobao.com	washingtonpost.com	youku.com	bestbuy.com

CDN Page Load Time Analysis

We grouped each website by their estimated CDN name. This CDN name was derived by first, checking if the site itself was a well known CDN (ex: cloudfront.net). Next, we parsed html tags on its downloaded homepage and checked if the static resources (such as images and scripts) were hosted on a url that was a well known CDN (see utils/CDNdomains.py). Finally, we parsed the whois data to find if a popular CDN name is also the organization name for the IP address we’re checking (this worked best for Fastly, Incapsula, and most other CDN providers).

Figure 1 compares event timings and total time per CDN (for CDNs serving multiple websites) as a stacked plot.

We observe that Taobao CDN had the worst overall performance due to the large average receive time and ssl negotiation time across two sites. We can also see this in Table 2 (t_ssl and t_rx for taobao.com and youku.com respectively).

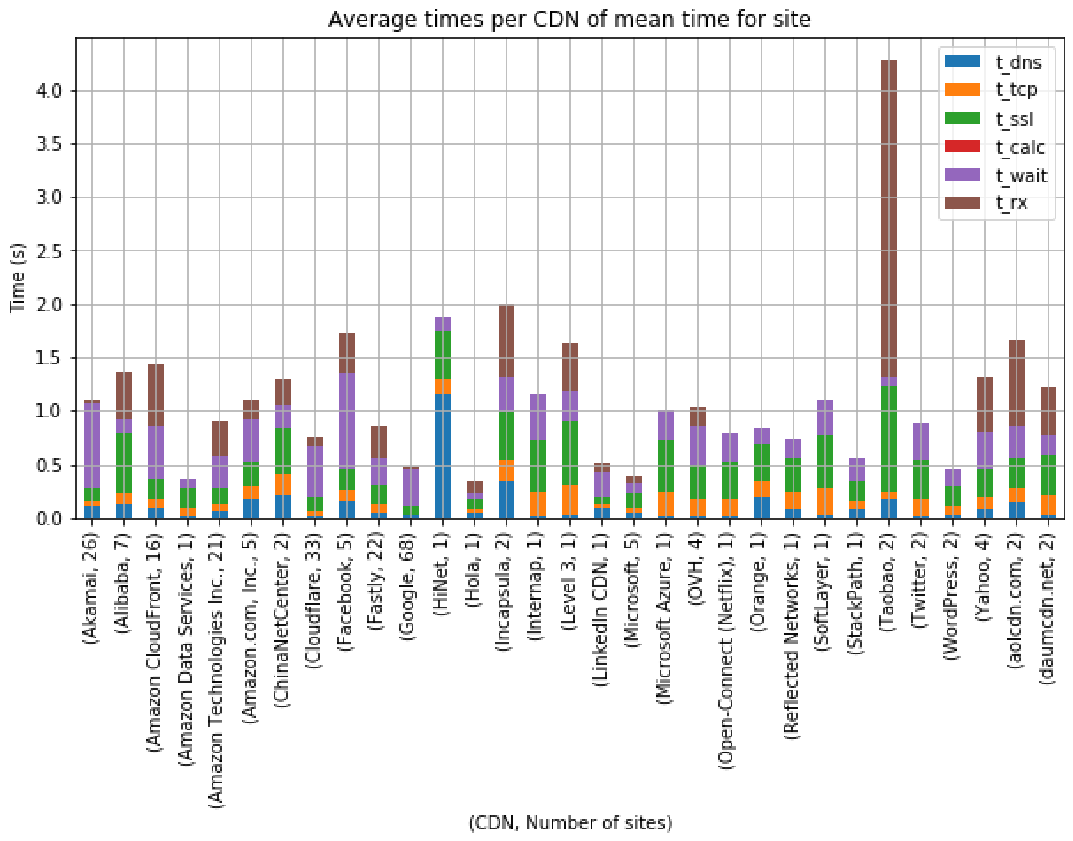


Figure 1: CDN vs Event Times for CDNs hosting multiple websites (results/stacked_calc_times_per_cdn)

For easier analysis, we grouped CDN names into major CDNs (Google, Akamai, Cloudflare, Cloudfront, Facebook, Fastly, Alibaba, Microsoft, Yahoo) that hosted many websites. Thus websites on Amazon CDN (i.e., those on Amazon data services and those using Cloudfront), and those on Alibaba CDN (i.e., using either Alibaba or Taobao) are grouped together. Other well known CDNs that host fewer websites were grouped into “KNOWN CDN”. Unknown URLs that are suspected to be CDNs are also grouped together into “UNKNOWN CDN URL”. While websites that are not using CDNs are grouped by “NO CDN”. **Figure 2 compares performance between major CDNs to KNOWN CDN, UNKNOWN CDN URL, and NO CDN categories**.

Across major providers, **Google seems to have the best performance by time**. However, note that almost 70% of requests to google.* domains redirected us. This might also be the reason that **wait time (t_wait) is the largest factor in Google’s performance**. In fact, for redirected responses, the average wait time is 0.44s as compared to 0.11s for 200 OK responses for google.* domains. Other domains such as blogger.com and blogspot.com also affect Google’s t_wait for the worse (see results/stacked_calc_times_per_site_nongoogle)

The **worst performer is the Alibaba CDN group**. However, if we remove Taobao CDN from the Alibaba group, its performance is much better (as seen in Figure 1 where the worse performing taobao.com and youku.com are part of Taobao CDN instead of Alibaba CDN).
The average performance of the "NO CDN" group is only slightly higher than that of the "UNKNOWN CDN URL" group. We also do a detailed analysis of each major CDN group in Analysis-Timing.ipynb (see results/stacked_calc_times_per_site_*CDNname*)

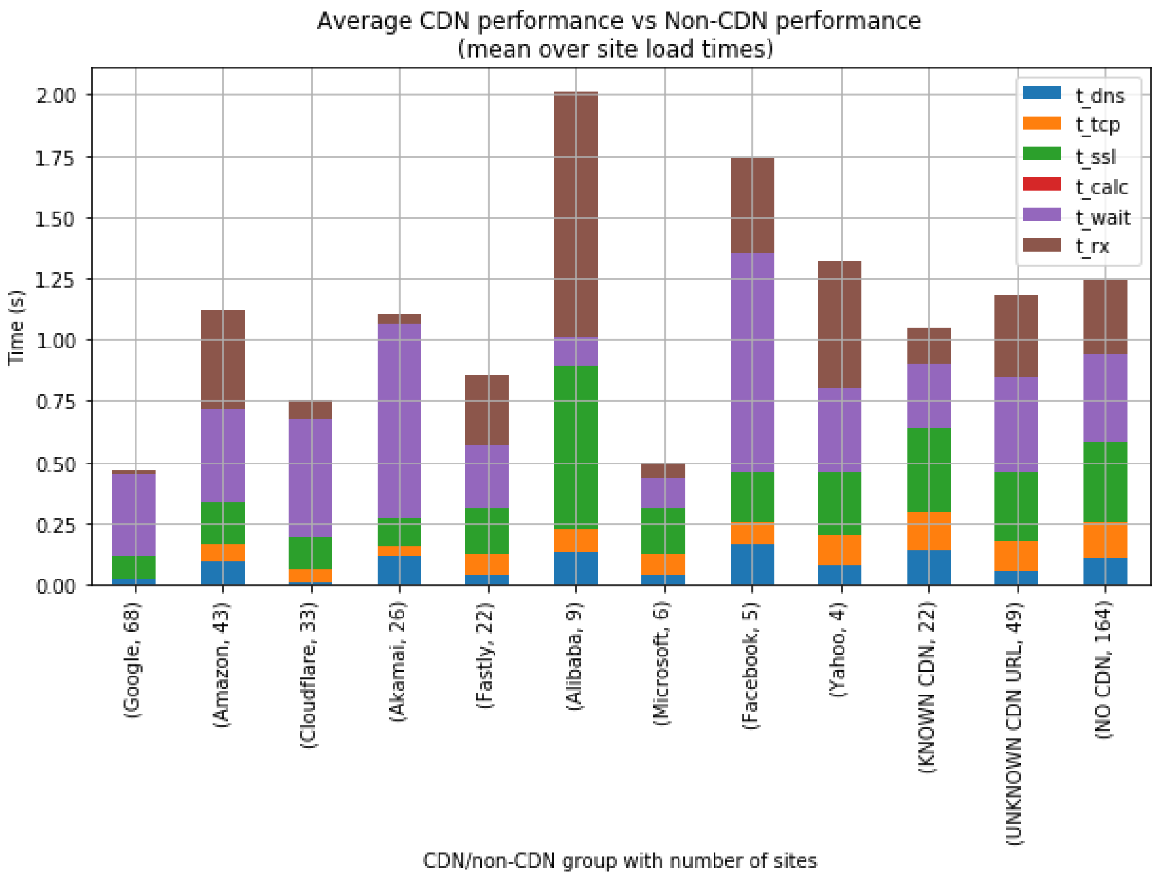
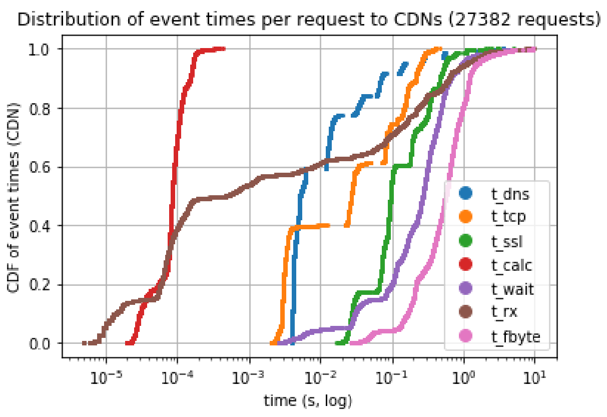


Figure 2: CDN groups vs Event Times compared to sites using No CDNs
(results/stacked_calc_times_per_cdn_vs_noncdn)

Note that this performance for CDNs is averaged across the mean performance per site. This is because almost all 451 sites had between 95-96 successful responses (including status 200 OK and redirects). As number of requests per site is almost equal, the average across sites per CDN is similar to average across all requests per CDN. We confirm this in Figure results/stacked_calc_times_per_cdn_all_requests.

Finally, figure 3 (a) and (b) compares the performance for websites using CDN to websites not using CDNs as a CDF of event times on a log scale (see results/performance_noncdn_all_requests.html & results/performance_cdn_all_requests.html).



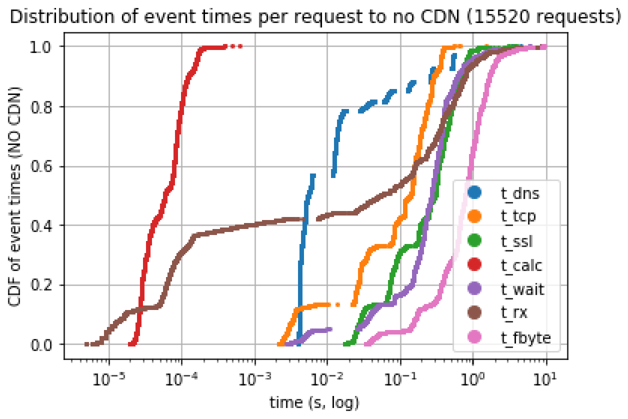


Figure 3: CDF of event times across all responses for (a) CDNs and (b) No CDNs

A comparison of median event times shows that sites on **CDNs outperform other sites in total time taken as well as time to first byte by 0.2s.** Median t_fbyte for CDNs is 0.57s with a negligible receive time. For non-CDNs, t_fbyte is 0.8s and t_rx is 0.06s. **The major difference between the two groups is between TCP handshake and SSL negotiation.** For sites using CDNs, median t_tcp is 0.027s and t_ssl is 0.096s across all requests. For sites not using CDNs, median t_tcp is 0.141s and t_ssl is 0.305s. Thus, **non-CDN websites cost 5 times and 3 times of websites using CDNs for TCP handshake and SSL negotiation, respectively**. Median time for name lookup (0.005s) and waiting time (0.27s) is almost the same across both groups, **showing that t_dns (depending on DNS servers) and t_wait (depending on time for server parsing, calculation and response) is independent of being on CDNs.**

ASN Page Load Time Analysis

Similar to the CDN Page Load Time Analysis, we group the timing data based on ASNs and take average across websites per ASN for 451 sites that responded to our curl requests. For ease of comparison, we group together all ASNs belonging to the top 10 major ASes (from table 1), while other ASes are grouped into the “Other AS” category. Major ASes host 250 websites, while Other ASes host 199 websites. Two websites, googleusercontent.com and wixsite.com could not be resolved to a valid IP (blocked by ISP).

Fig 4 compares performance of major ASes with “Other AS”. This shows that **Google and Microsoft had the best performance with lowest time taken overall** (under 1s). **Alibaba and ChinaNet had the worst performance.**
Sites using Google were all on ASN15169. Microsoft had 2 ASNs: ASN8068 hosts live.com and bing.com while all office services were hosted on ASN8085. ChinaNet had 4 ASNs hosting 9 websites, with ASN23650 hosting only one website (ci123.com). None of these sites were detected using CDNs.

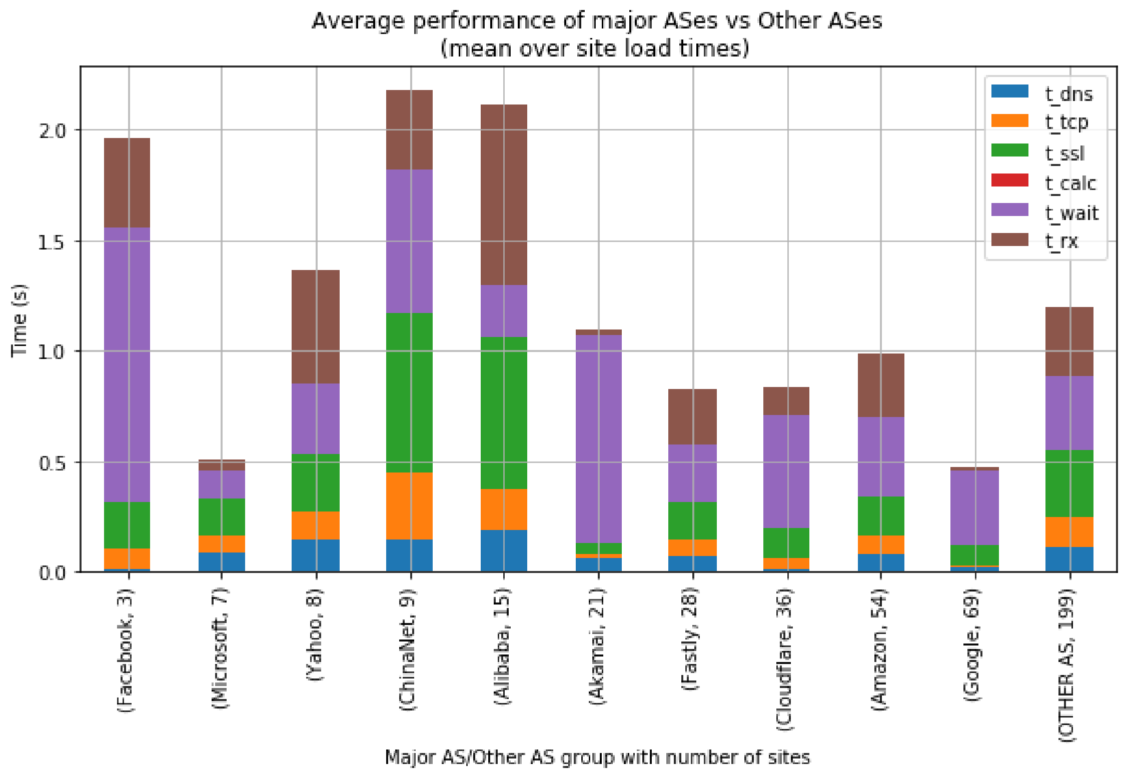


Fig 4: Event times as stacked plot for Major ASes vs Other ASes

Detailed analysis showed that **performance varies across ASNs in the same major AS group** (see results/stacked_calc_times_per_asn). Akamai ASN16625 hosts 15 sites and has a much lower wait time (t_wait) than ASN20940, which hosts 5 sites. The major contributors to worsening performance for both were bestbuy.com on ASN16625 and washingtonpost.com on ASN20940. Without these two websites (each with t_wait around 8.5s), the performance of the two ASNs is fairly similar, with average total time on ASN16625 as 0.296s and on ASN20490 as 0.24s. Similarly, performance for Alibaba's ASN45102 was better than ASN37963 by 0.75s.
In contrast, **Amazon's performance is mostly independent amongst its two ASNs** (14618 and 16509). The worst sites are mercadolibre.com.mx, mercadolibre.com.ar, mercadolivre.com.br with average total time over 2s, and receive time of around 1.5s. Three sites, nih.gov, amazon.com.de, and instructure.com had a surprisingly large lookup time (~0.8s, see Analysis-ASN.ipynb for a more detailed analysis).

CDN-ASN Mapping

Having the same ASN doesn't necessarily mean that websites are using the same CDN as other sites on that ASN. For example, certain sites may be hosted on Amazon AWS but using CDN services other than Cloudfront.
Observation shows that for a majority of sites, ASNs map to the CDN estimated by our algorithm. This implies that sites are hosting their static resources with the same provider

hosting their home page. Performance of such websites would be optimized as the Enterprise network can efficiently allocate resources to deliver content.

In our initial analysis, **we detected that 300 sites were using CDNs** (removing 2 blocked sites). **224 of these CDNs were the same as the AS name (74.67% match)**. 180 sites were detected as not using CDNs and 20 were blocked by the ISP. There were 28 other websites detected using well known CDNs that didn't match the AS name, and 48 sites for which we detected a CDN URL hosting majority content but couldn't resolve it to a known CDN name. We'll be concentrating on these 76 sites in this subsection.

Of **28 websites that used a popular known CDN that was different from the AS name**, login.tmall.com uses Alibaba CDN but is hosted on Taobao's ASN, and youku.com & taobao.com use Taobao CDN but are hosted on Alibaba's ASN. Since Taobao is part of Alibaba Inc., the performance of these sites shouldn't be affected, but our earlier analysis showed that sites using Taobao CDN had a worse performance compared to Alibaba (see Fig 1).
Similarly, msn.com hosted on Microsoft's ASN was using Akamai CDN. We detected Instagram and Whatsapp using Facebook CDN but hosted on Amazon and SoftLayer CDN. Similarly, Netflix uses Open-Connect, but for requests from India, its IP range lied on Amazon's ASN.
5 sites used Akamai as CDN but had their own private AS hosting the homepage. We found that amazon.cn was hosted on China Unicom's ASN and bitly.com had its own independent AS but used Amazon Cloudfront. **2 surprising sites were: trello.com, that detected Amazon Cloudfront as CDN but was hosted on ASN16625 belonging to Akamai; liputan6.com, that detected Akamai as CDN but was hosted on AS16509 belonging to Amazon.** Initially, we believed this was an error in our CDN estimation, but a detailed analysis proved most static objects loaded on their homepage were actually hosted on a CDN different from its own AS.

- While trello.com is hosted on Akamai, static resources from c.trello.com were hosted on Amazon Cloudfront.
- liputan6.com has a mix of objects and ad services from most major CDN providers including Cloudflare, Amazon Cloudfront, Google, Facebook, etc. but most of its static objects were hosted on Akamai while the site itself is hosted on Amazon.

Of 48 websites that hosted **most content on a CDN URL** that we couldn't resolve, **27 had the CDN name mentioned as the "Organization" in the whois record.** This obviously matched the AS name. The mapping between these URLs and CDN name (based on whois records) can be appended into our offline hash of domains and CDNs (see utils/CDNdomains.py).
21 of 48 websites did not have a known CDN in their whois records, so their CDN type URLs could not be successfully resolved. This let us explore **new CDNs that are not well known**. For example, box.com used the URL cdn03.boxcdn.net as its CDN and its AS is BOXNET (ASN33011). This list also included huffingtonpost.com that used aolcdn.com but was hosted on Yahoo's ASN. In fact, aolcdn.com is part of EdgeCast CDN and huffingtonpost.com hosts majority resources on EdgeCast even though it is detected on Yahoo's ASN. Thus, although this method isn't full-proof, it lets us explore potential CDN mappings of URLs to ASNs that are not in our offline popular CDN name list yet.

There were **180 sites that were detected not using any CDNs**. Of these, **22 websites had an AS that was also a popular CDN provider**. Not all sites hosted on the same ASN as a CDN provided need to use the same CDN provider, this list shows the best CDN that should be used by these 22 sites in case they decide to switch. This list includes tumblr.com, pixiv.net, kakaku.com on Yahoo's ASN, and duckduckgo.com, academia.edu on Amazon's ASN16509. This also includes 5 sites on ChinaNet and 6 on Alibaba, all detected as not using CDNs currently.

Conclusion

We used curl to request the top Alexa 500 sites multiple times and calculated timings for individual events. We parsed downloaded homepages for these sites and checked their whois records to estimate the CDN they are using. Analysis of page load timings averaged over CDNs showed that the median time taken for TCP handshake and SSL negotiation is lesser when using CDNs. Furthermore, time for name lookup and waiting time for server calculation and response is essentially independent of whether a site is using a CDN provider. These times depend on the DNS server and the website host server rather than the CDN. 75% of our estimated CDNs matched the AS names, and page load performance across such providers was consistent.

Both the CDN estimation algorithm and the data collection for timing analysis can be improved. Below we suggest some improvements that can be made to the algorithm.

Improvements

- Use selenium to parse chrome traces for exact time taken to access a webpage, instead of using curl to get timings. Many curl requests are actually redirected to a specific page that might be rendered much quicker than when asking the site for specific content. So we may be measuring the time response to a cached page rather than the actual homepage for a form-based website. Although we tried measuring absolute times for web page loads by performing redirected curl requests, we aren't able emulate a user's experience. A solution to this is to collect chrome traces using selenium to detail each and every event when loading a webpage, and parsing these traces to get the required event times and a detailed waterfall diagram.
- CDN estimation is not absolute. The truest test for CDNs would probably be by testing the existence and network performance of individual static objects on a webpage from multiple locations across the globe. We've estimated the CDN using information from DNS, whois, as well as comparing parsed objects to a known list of DNS urls. This hash table of urls always needs to be monitored and updated. The current estimation might miss CDNs not in the hash table, sites using multiple CDNs, or those with bad whois

traces. We might also get false positives based on urls that look like CDNs but are really not.

- Currently, our algorithm uses a majority rule (score>0.5) to estimate whether a site is using CDNs. However, if a site is using multiple different CDNs with no clear majority for one CDN, we might miss it. A way to avoid this is adding the score for all CDN URLs detected on parsing the page and comparing it to the score of objects hosted either locally or on non-CDN URLs. In this case, a site can be detected as using “Multiple CDNs” instead of “NO CDN”.
- CDN estimation can also be improved by using selenium to trace the complete downloaded webpage. We can parse HTTP headers for each static object to find if a CDN cache was HIT or MISSed. We can also use DNS CNAME records to assess if a URL used to host content is actually pointing to a CDN.