CacheLoc: Leveraging CDN Edge Servers for User Geolocation

Mingkui Wei[1(✉)], Khaled Rabieh[2], and Faisal Kaleem[2]

[1] Cyber Forensics Intelligent Center, Computer Science, Sam Houston State University, Huntsville, TX, USA
mwei@shsu.edu

[2] Computer Science and Cybersecurity, Metropolitan State University, Saint Paul, MN, USA
{khaled.rabieh,faisal.kaleem}@metrostate.edu

Abstract. In nowadays' Internet, websites rely more and more on obtaining users' geolocation to provide customized services. However, besides Internet giants such as Google, who retains a large amount of detailed user information, most websites still rely on IP addresses for user geolocation, which is proven inaccurate and misleading by existing studies. In this paper, we propose a novel approach, namely *CacheLoc*, for coarse-grained user geolocation leveraging widely-deployed content delivery networks (CDNs). This work is motivated by the fact that CDN providers deploy a number of edge servers that are geographically distributed across the world. Many of these edge servers are assigned with unique identifiers that are tied to their location, which can be easily retrieved by inspecting HTTP responses headers served by these edge servers. As a result, a website can infer coarse-grained user location by asking a user to send an HTTP request to an arbitrary domain that is known being served by a CDN, and inspecting the corresponding responses. To evaluate the usability and accuracy of the cache-based user geolocation, we conducted practical experiments based on a commercial VPN with over 160 endpoints distributed in 94 countries. Our experiments demonstrate that cache-based geolocation can achieve at least accurate country-level granularity in the regions where CDN edge servers are densely deployed. Our work sheds light on a novel light-weight and self-contained user geolocation solution.

Keywords: Content delivery networks · User geolocation

1 Introduction

Websites in today's Internet rely more and more on obtaining users' geolocation to provide customized services, such as regional campaigns or promotional activities. Currently, the primary method to obtain a user's location is based on the user's IP address, to which there are two major approaches. The first approach

N. Park et al. (Eds.): SecureComm 2020, LNICST 336, pp. 22–40, 2020.
https://doi.org/10.1007/978-3-030-63095-9_2

is to directly obtain the user's IP address, and search it against known databases such as IP2Location [1] and Whois [2]. And the second approach is to leverage web APIs provided by Internet giants such as Google, who maintains substantial user information collected via multiple means (WiFi war-driving [3], for example). Both approaches, however, have their shortages. For the former IP-based user geolocation, the major issues lie in the lack of official ground truth to validate the correctness and accuracy of existing databases. It has been found by existing studies that for the same IP address, the distance between the locations obtained from two different databases can be as large as 800 Km [4]. The API-based approach, one the other hand, can obtain very accurate results. However, modern browsers have built-in mechanisms to block such APIs from operating. For instance, Google's geolocation API [5] will trigger a pop-up window asking the user's permission to explicitly allow his/her location to be shared with the website he is visiting. As Internet users' concern regarding their privacy is daily increasing, more likely than not, the user is going to block such location requests unless there are legitimate reasons to allow them.

In this paper, we propose a novel approach for user geolocation by leveraging the popularly used content delivery networks (CDNs). The new cache-based geolocation, namely the *CacheLoc*, is motivated by the fact that CDN providers deploy a large number of edge servers geographically distributed. Many of these edge servers are assigned with unique identifiers that are tied to their geolocation, which can be easily retrieved from HTTP responses served by these edge servers. By asking a client to issue a regular HTTP request to a domain that is served by CDNs and inspect corresponding response headers, a website can infer the location of the user who is currently visiting it. Such cache-based geolocation, although coarse-grained, can be sufficient for purposes such as regional campaign or advertisement. Furthermore, it can be used as a side-channel knowledge to cross-validate the results obtained from conventional IP-based geolocations.

Compared to conventional IP-based user geolocation, the cache-based approach has the following advantages.

1. The mechanism of the cache-based geolocation is very straightforward. While IP-based geolocation requires a website to interact with databases leveraging web APIs, the cache-based approach can be implemented with just a few lines of JavaScript embedded in the web document, which asks the client to issue one regular HTTP request and retrieve one value from response headers.
2. IP-based geolocation relies on third party databases that may incur subscription fees, while the cache-based approach is self-contained and completely free since all that required is to ask the user to issue an HTTP request to a public domain.
3. The correctness and accuracy of IP-based geolocation are hard to be validated because there lacks any official ground truth. Cache-based approach, on the other hand, is based on publicly known and reliable information and therefore bears higher reliability.

In the following, we present the details of the cache-based user geolocation solution. The content of the rest of this paper is organized as follows. In Sect. 2,

we introduce necessary background knowledge that assist the reader to understand *CacheLoc*. In Sect. 3, we present the details of the novel cache-based user geolocation. In Sect. 4, we discuss the usability of *CacheLoc* with preliminary experiment results. We conduct practical experiments and present their results in Sect. 5 to evaluate the accuracy and granularity of *CacheLoc*. Finally, we conclude our work in Sect. 6.

2 Background

In this section, we briefly introduce related works in user geolocation and the necessary background knowledge for content delivery networks.

2.1 Existing Works in User Geolocation

Most browsers use IP addresses to determine a user's location due to its simplicity. For IP-based user geolocation, the webserver subscribes to the access to geolocation IP databases, which maps ranges of IP addresses with the corresponding latitudes and longitudes coordinates. The pair of coordinates provides the sever with the location of the IP address, such as time, country, and city [6]. There are abundantly available IP geolocation databases, including ip2c.org, GeoLite2Geo Targetly, IP2Location Lite, and GeoIP Nekudo. While using such databases allows a server to locate a user without the need for GPS receivers or complicated configuration switching, it suffers from plentiful of drawbacks. For instance, IP-based geolocation is far from reliable and accurate since it only provides a rough estimate of users' locations. For example, the literature in [7,8] shows that the locations obtained from different databases suffer huge accuracy errors up to 800 km in some cases. During our experiments, we also experienced many such cases. For instance, we found one IP address was located in Hong Kong by one database, but Australia by another. Further, IP databases come with many operation overheads such as paid subscriptions for support, frequent updates to guarantee better data accuracy, scalability, and management issues.

Li et al. [4] proposed city-level IP geolocation based on network topology community detection method to improve the accuracy of geolocation. They use the community detection algorithm in complex networks to find the different communities in the network topology and determine the location of the communities. The geographical position of target IP is obtained according to the communities of target IP. The experiment shows that its location accuracy ratio is above 96%. Triukose et al. examine IP address allocation in cellular data networks, with emphasis on understanding the feasibility of IP-based geolocation techniques. The authors used two commercial IP geolocation databases, MaxMind [9] and IPinfoDB [10] to test the ability of the databases to determine the ability of these databases to return host location based on IP addresses seen by the application's server.

API-based geolocation is a new approach that uses the browser's HTML5 Geo-location feature along with the Maps JavaScript API [11] to detect users'

locations, all leveraged by Google's comprehensive database about the user's profile. While this approach brings higher location accuracy, the location is only shared if the user allows location sharing in a pop-up window. With more and more Internet users begin to concern about their privacy, a user will likely deny such request unless necessary.

2.2 Content Delivery Networks

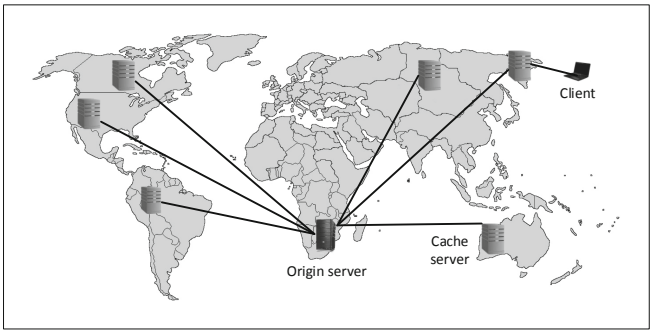

Fig. 1. Content delivery network

Content delivery network, or CDN, is a type of web cache that has undergone substantial growth in the recent decade [12]. It provides a scalable and cost-effective mechanism for accelerating web document dissemination among the Internet [13] by deploying a large number of edge servers around the globe. These edge servers sit between HTTP clients and origin servers, which cache static web documents served by origin servers, and use the cached copies to serve subsequent duplicate requests. Consequently, requests sent by a user in a certain location will always be served by the nearest CDN edge server, regardless of the origin server location. For example, as shown in Fig. 1, users located in the U.S. will be served by the servers in North America instead of the origin server in Africa. As a result, users will not only experience shorter page loading time, but the origin server will also see reduced workload in terms of the volume of HTTP requests. Because of these advantages, CDN has been adopted by a plethora of websites in recent years.

3 Cache Based Client Geolocation

The idea of CacheLoc is motivated by the fact that HTTP responses served by CDN edge servers are usually appended with CDN specific information. For some CDNs, such information reveals the location of the edge servers by whom the request was served. For example, Listing 1.1 presents typical response

headers served by a Cloudfront's [14] edge server, where the request was sent via an HTTP proxy that locates in Texas, U.S. As highlighted in line 6, the `X-Amz-Cf-Pop` response header is a customized header appended by all edge servers belong to Amazon Cloudfront, and whose value indicates the request is served by the edge server `DFW55-C1`. This header value implies that the request is served by the edge server near Dallas, TX, because it is a common practice among many CDNs to name their edge servers with the three-letter IATA airport codes that are close by [15], and `DFW` refers to the Dallas/Fort Worth International Airport. Furthermore, because CDNs always serve HTTP requests with the edge servers closest to the user, we can infer that the user who issued the request must be somewhere close to the city Dallas. In the following, we describe how such information can be leveraged to identify users' locations.

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=UTF-8
3 ...
4 X-Cache: Miss from cloudfront
5 Via: 1.1 5d52966f37c4378fd883294634452d6b.cloudfront.net (
      CloudFront)
6 X-Amz-Cf-Pop: DFW55-C1
```

Listing 1.1. Typical response headers served by a *Cloudfront* edge server.

3.1 Mechanism of CacheLoc

In Fig. 2, we present the flowchart explaining how a website can infer a user's geolocation leveraging CDN response headers. To facilitate the following illustration, we use the term *publisher* to refer the owner of the website that is visited by a user and wants to infer the user's location. We assume the publisher owns the domain `origin.com`. We also assume another domain, i.e., `pilot.com`, is a domain that is served by a CDN whose edge servers append location related headers to the responses.

As shown in Fig. 2, a user visits `origin.com`'s default main page (i.e., `index.html`) by sending a HTTP `GET` request, and the webserver will respond with the requested document once the request is received. In order to infer the user's location, the webserver inserts a JavaScript snippet as a part of *index.html*, which requests the user to issue an HTTP request to the pilot domain `pilot.com`. Because we are only interested in the response header, a `HEAD` request is sufficient.

After the document `origin.com/index.html` is received by the user's browser, the browser will execute the JavaScript and issue the request, which will be served by the closest edge server and append customized header indicating its identity. Once the response from the edge server is received at the user's browser, the JavaScript will inspect the response headers, retrieve the edge server identifier, and send it back to `origin.com`, which can be attached as the content of a `POST` request, or simply appended as a query string using a `GET` request. The

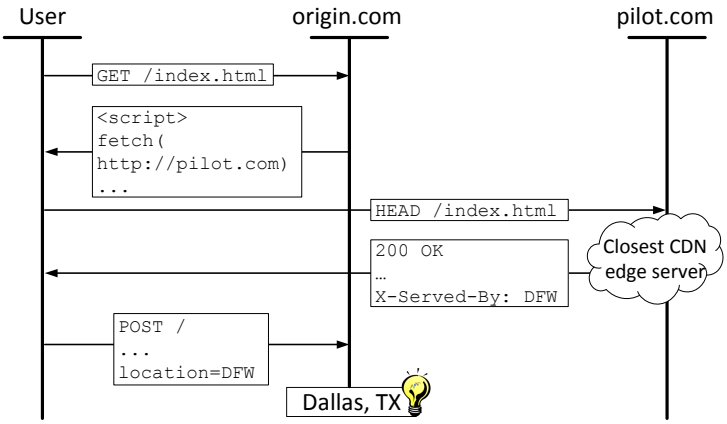

Fig. 2. Flowchart

publisher, knowing that `DFW` implies Dallas, can then infer the user is located in Texas and close to the city Dallas.

We demonstrate the necessary requirements of the pilot domain in order to implement *CacheLoc* in the following.

3.2 Pilot Domain Configuration

As depicted in Fig. 2, `origin.com` takes two steps to infer the user's geolocation: 1) it requests the user to issue an HTTP request to a pilot domain, and 2) it inspects the response to retrieve the value of a specific response header. While issuing the request and inspect response header can be easily done with just a few lines of JavaScript as presented in Listing 1.2, a barrier that may prevent the header information from being accessed lies in the same-origin policy (SOP) set forth by most modern web browsers [16].

```
1  var xhr = new XMLHttpRequest();
2  var url = 'https://pilot.com/';
3  xhr.open('HEAD', url);
4  xhr.send()
5  xhr.onreadystatechange = function() {
6   if(xhr.readyState == 4 ) {
7    var p = xhr.getResponseHeader('X-Served-By');
8   }
```

Listing 1.2. Typical response headers served by a *Cloudfront* edge server.

In specific, the same-origin policy is a critical security mechanism that is implemented on all modern web browsers, which restricts the interaction between a resource request issued from one origin and the actual resources reside on another origin, where the *origin* is composed of the three parts: *scheme*, the

host, and the *port number*. Two origins are not considered identical unless all three parts match. With strict SOP being enforced, the web browser does not allow JavaScripts in one origin to access resources, including sending requests to or reading responses from, another origin. However, because cross-origin resource referencing is prevalent in today's Internet, SOP is loosened by the cross origin resource sharing (CORS) policy, which allows scripts from one origin to access resources from another origin under certain circumstances.

Particularly, for one origin to access resources from another origin, the latter origin must allow the resource sharing by explicitly appending a set of CORS response headers [17]. For example, assume the JavaScript in Listing 1.2 is included in `origin.com/index.html` and is parsed by a user's web browser. Prior to sending the actual `HEAD` request, the browser will first send a `OPTIONS` request to `pilot.com` (known as the *pre-flight* request) as shown in Listing 1.3, and check the response headers. The subsequent `HEAD` request will be sent only if the header `Access-Control-Allow-Origin` exists in the response and either `origin.com` or the wildcard symbol `*` presents as the value. Otherwise, the browser will not sent the `HEAD` request at all because `pilot.com` does not allow `origin.com` to access its resources.

```
1 OPTIONS /index.html
2 Access-Control-Request-Method: GET
3 Origin: https://example.com
4 ...
```

Listing 1.3. Typical response headers served by a *Cloudfront* edge server.

Furthermore, even if `Access-Control-Allow-Origin` exists and `origin.com` is explicitly allowed, the browser still restricts `origin.com` that only the 7 CORS-safelisted response headers [18] can be accessed: `Cache-Control`, `Content-Language`, `Content-Length`, `Content-Type`, `Expires`, `Last-Modified`, and `Pragma`. In order to access the CDN specific header, for example, the `X-Served-By` header, another CORS header, i.e., `Access-Control-Expose-Headers`, must also exist and explicitly specify either `X-Served-By` or `*` as the value.

Therefore, in order to successfully obtain the CDN related response header by issuing HTTP request and reading the response, `origin.com` must find a pilot domain that explicitly appends the headers `Access-Control-Allow-Origin` and `Access-Control-Expose-Headers`, and specify `origin.com` or `*`, and `X-Served-By` or `*` as the values, correspondingly.

The most straightforward way to obtain such a pilot domain is for the publisher to set up a dedicated domain and subscribe to CDN services, where the pilot domain can simply be a subdomain of `origin.com`. For instance, the publisher can create the subdomain `cloudfront.origin.com` and subscribed it to Cloudfront's service. Because this domain is entirely controlled by the publisher, the two CORS headers can be directly inserted into response headers by configuring the webserver. Because `origin.com` is only interested in the response headers, the pilot domain does not need to be substantiated with any real con-

tent. For example, a completely blank HTML page will suffice the purpose. Because many CDNs offers free tier services based on limited traffic amount or cost (for example, Cloudflare offers free tier service, Fastly provide $50 worth credit for new customers, and Cloudfront set the first 50GB traffic free of charge), a HEAD request only incurs minimal traffic and negligible cost at best.

Another approach to find a suitable pilot domain is to scan the Internet and attempt to find an independent domain that subscribed to a specific CDN service, and also includes the two headers `Access-Control-Allow-Origin` and `Access-Control-Expose-Headers` and the desired value (which should be `*`, because the specific value `origin.com` and `X-Served-By` is unlikely to be set by an independent third-party domain). This task could be laborious but not impossible. For instance, by scanning the first 50K domains against the *Majestic Million* domain list [19], we found the domain `cwtv.com` is subscribed to Cloudflare's CDN service, and has the above two headers being present and value set to be `*`. Compared with the first approach, this approach only requires a one time task and is simpler since it eliminates the complexities to set up the subdomain and subscribe to CDN services.

4 CacheLoc Usability

Compared with conventional IP-based user geolocation, the cache-based geolocation has the advantages that 1) It incurs very low overhead. The publisher only needs to insert a few lines of JavaScript code, while the user only needs to issue two HTTP requests, one to the pilot domain to obtain CDN related information and one to the publisher to inform such information. 2) It is self-contained and does not rely on any third party service. And 3) It's information is obtained from CDN edge servers, which is publicly available and thus verifiable. On the other hand, it is evident that the granularity of cache-based geolocation is limited by the edge server's density and distribution, and is unlikely to achieve high accuracy. Nevertheless, we argue that such coarse-grained granularity may be sufficient in many scenarios. For example, a political campaign or commercial advertisement may target a broad region where fine-grained user location is favorable but unnecessary. Further, this cache-based geolocation can also serve as cross-validations to conventional IP-based geolocation to improve the results' reliability. For instance, during our experiment, we encounter many cases where an IP address was located in two different countries, where the cache-based geolocation can then be used to narrow down the results to the correct one. We discuss the usability and limitations of cache-based geolocation in the following.

4.1 Suitable CDN Services for CacheLoc

CDN is a relatively new business model emerged in recent decade [12], and their distribution of services shows strong regional characters. Major CDN providers in North America include both traditional Internet companies, including Google, Amazon, and Akamai, and relatively new ones founded in the last decade, such

as Cloudflare and Fastly. According to an online survey [20], currently, there are 23 CDN providers in the United States, however, not all of them are suitable for geolocation purposes. In order to be used for user geolocation, a CDN must present the following two properties: its edge servers' locations are publicly known, and their locations are identifiable from HTTP response headers.

For the first factor, i.e., publishing edge servers' information, different CDN shows different tendencies. Some providers are very transparent and actively publish detailed information regarding their CDN network. For example, Cloudflare publishes its up-to-date data centers' location (also known as the point of presence, or PoP) and the number of servers at each location [21]. On the other hand, providers such as Akamai are relatively conservative and only provide very brief information about their data centers' location.

For the second factor, different CDN providers also take different approaches. Some providers, including Cloudflare, Cloudfront, and Fastly, append a customized response header to identify the edge server that served the request. In particular, Cloudflare appends the `CF-RAY` header, for example, `CF-RAY: 572244ec8cadd266-DFW`, whose last section identifies the edge server; Cloudfront appends `X-Amz-Cf-Pop` header, for example, `X-Amz-Cf-Pop: DFW55-C2`, to indicate not only the location (i.e., `DFW`), but also specific edge server at this location (i.e., `C2`); and Fastly inserts `X-Served-By` header, for example, `X-Served-By: cache-dfw18677-DFW`, whose last section identifies the edge server. On the other hand, CDN providers such as Googles' Cloud CDN only inserts a simple `Via: 1.1 google` header to indicate the request is served by Google, Akamai does not have any header that reveals its edge server's identification either.

4.2 CDN's Data Center Locations

Based on the above discussions, in this study, we chose three CDNs to validate the proposed cache-based user geolocation, which are Cloudflare [22], Cloudfront [14], and Fastly [23]. In order to obtain a preliminary knowledge of the accuracy the cache-based geolocation can achieve, our first step is to collect and analyze information regarding each CDN, as described in the following.

Table 1. Statistics from website description. (* one cite can have multiple PoPs.)

	Cloudflare	Cloudfront	Fastly
Number of PoPs*	N/A	216	75
Number of countries	90	42	N/A
Number of cities	200	84	60

To begin with, we collected information regarding the data centers' location from each CDN's official website, and present the result in Table 1. Comparing the three, Cloudflare has the largest CDN network, which spans over 200 cities

in more than 90 countries. A CDN provider may place multiple PoPs in one city, but may not necessarily differentiate them. For instance, according to the website description, Cloudfront has 6 PoPs in Dallas, TX, and during our experiment, we found these data centers are assigned with different names including `DFW3`, `DFW50`, `DFW52`, `DFW53`, `DFW55` (we were only able to see 5 PoP names). On the other hand, Flastly states that it has 2 PoPs present at Dallas, but we were only able to see the unified identifier `DFW` and thus unable to distinguish the two servers.

Based on our preliminary evaluation, we suspect that the information published on CDN providers' website may not be up-to-date. Therefore, as the second step, we conducted a live scan to verify existing and identify new information. In specific, all three providers publish the range of IP addresses they owned on their website [24–26]. We start the experiment by scanning the whole IP range for TCP port 80. In specific, Cloudflare, Cloudfront and Flastly have 1,786,881, 1,422,793, and 222,208 unique IP addresses, respectively, among which 96,671, 140,347, and 65,969 are alive, i.e., responded to the scan. Note that these results are likely transient because CDN providers usually dynamically assign IP addresses to edge servers due to reasons such as load balancing [27], however, our results provide a snapshot of these CDN networks, based on which we can conduct the following analysis.

Then, we wrote a simple python script leveraging the *requests* library to send a `HEAD` request to each live IP address. For simplicity purposes, for each request, we set the `Host` header to be a random string (e.g., `Host: aaa`) rather than any valid host names. Because the host header is not recognizable by the edge servers, whey will respond with an error page indicating the specified host name is not accessible (`500 Domain Not Found` from Fastly, `409 Conflict` from Cloudflare, and `403 Forbidden` from Cloudfront), which nonetheless satisfied our purpose because even the error page still contains response header that includes edge servers' identifier. After we received all responses, we inspect the response headers and strip edge servers' identifier and summarize the result in Table 2. Specifically, we obtained a total of 283 unique edge server IDs from Cloudfront, which is much larger than the number of PoPs stated on its website (i.e., 216), implying the information on its website is obsolete. We also observe Fastly presents a slight difference, i.e., 78 obtained by scanning *v.s.* 75 stated on the website. We were not able to scan Cloudflare's CDN network because Cloudflare's CDN network uses Anycast [28]. As a result, even though we specifically send a request to a specific IP address, the request will always be routed to and served by the closest edge server. Therefore, we can only see the single edge server identifiers that is closest to us.

4.3 Limitation

From the edge server maps of the three CDNs [21,29,30], it is evident that their edge servers are densely deployed only in North America and Europe. Therefore, we can only expect higher accuracy and finer granularity in these regions. However, as a proof of concept, we do not aim to practically geolocate

Table 2. Statistics from experiment.

	Cloudflare	Cloudfront	Fastly
Total IP addresses	1,786,881	1,422,793	222,208
Live IP addresses	96,671	140,347	65,969
Unique IDs	N/A	283	78

users worldwide. Further, such a shortage can be easily addressed by leveraging more regional CDNs. For example, Alibaba CDN, a China-based cloud service provider, has 39 data centers deployed in major cities in China [31], which can be used to geolocate China-based users with much higher accuracy.

5 Experiment

In this section, we conduct empirical experiments to evaluate the usability and accuracy of the cache-based user geolocation.

5.1 Experiment Setup

In order to evaluate the usability and accuracy of the cache-based geolocation, the ideal approach would be issuing HTTP requests at multiple locations around the world and verify if the correct location could be obtained. Originally, we planned to leverage the Planet Lab [32], a research project incorporated more than 2000 research institutions across the world, where a user can request access to any of these nodes. However, it seemed to us the Planet Lab project had been discontinued, as we have attempted a few times to email the support staff and never get any reply. Therefore, we finally decided to take an alternative approach by using VPN services. In specific, we purchased access to Express VPN [33], a VPN provider that has 160 VPN endpoints across 94 countries, which has the largest number of endpoints among all VPN providers that we are aware of. Express VPN also has a Linux command-line interface that allows us to write scripts and conduct experiments in batch.

Because we do not own a domain by ourselves, we are unable to completely replicate the scenario described as in Fig. 2. However, since our objectives are to validate the usability and evaluate the accuracy of the cache-based user geolocation, we design the following experiment as an alternative, which achieves our objectives nonetheless.

In specific, we first scan against the *Majestic Million* domain list [19] as mentioned above, and find three arbitrary domains that use Cloudfront, Cloudflare, and Fastly's service, respectively. Then, we wrote a script, which can automatically log in to each VNP endpoint, and issue three HTTP requests to each of these three domains. Then, we collect the response from these three domains, and extract the edge server identifier and save them into a log file.

It is noteworthy that because we do not know the exact location of any of these VPN endpoints, but only the country (or city, for a few cases) each endpoint is placed, in this experiment, we do not seek to pinpoint or verify their accurate locations using cache-based geolocation. Instead, our objective is to evaluate to what extent these 160 locations can be uniquely differentiated using the proposed cache-based geolocation method. However, we argue that this restricted experiment does not diminish the effectiveness of the cache-based geolocation as a general solution for user geolocation. This is because as long as we can uniquely differentiate these endpoints, knowing the identity and accurate location is only a trivial and laborious task. For instance, a capable publisher can gradually build its own database based on users' information. Specifically, the publisher can enable the cache-based location and still ask to access the user's GPS based location. Although such requests may be rejected by most users, it is still likely to be allowed by a few users due to reasons such as carelessness or by accident. Once the publisher obtains one accurate location, it can associate this accurate location with the specific CDN edge server identifier, and be informed that users with the same CDN identifier must from a place that is close to this known accurate location. Gradually, the publisher is able to build a quite accurate geolocation map, which can be further refined each time a user allows his/her accurate location to be accessed.

5.2 Experiment Results and Analysis

Statistical and Geological Results. During our experiment, we were able to successfully connect to 148 endpoints among 160 that is claimed on Express VPN's official website [33], and collected a total of 444 HTTP responses. These 148 endpoints span in 93 countries, which covers most countries in America and Europe, many countries in Southeast Asia, and a few countries in the Middle East and Africa, which is consistent with the official website description. Most endpoints were named by the country name where they locate. Figure 3 presents the countries that were covered by Express VPN's endpoints. Among these 93 countries, 13 countries have more than one VPN endpoints present, in which case, these endpoints were named by the country name append with the city's name and a numerical index. In the following, we analyze the usability and accuracy of cache-based geolocation from both the country level and the city level.

Country Level Geolocation. As explained, the three CDNs that we have chosen, i.e., Cloudfront, Cloudflare, and Fastly, are U.S. based CDN providers and have their market focus in North America and Europe. Therefore, we expect the accuracy of cache-based geolocation bears much higher accuracy in differentiating European countries. In the following description, we separately demonstrate the results for European countries and the rest of the world.

We first present the geolocation result with a single CDN. In total, these 93 countries were served by 41 Cloudfront edge servers, 24 Fastly edge servers,

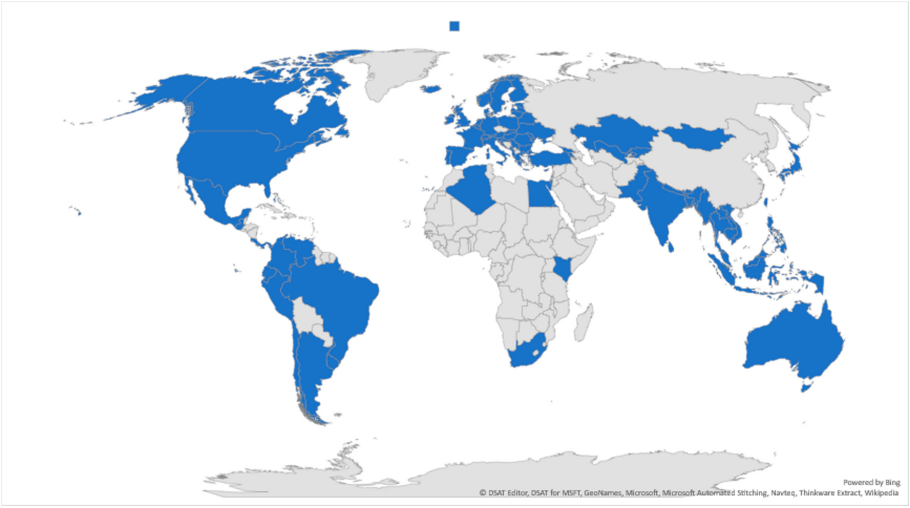

Fig. 3. Countries where Express VPN's endpoints present.

and 31 Cloudflare edge servers, respectively. Among which, 46 European countries were served by 23, 12, and 18 edge servers from Cloudfront, Fastly, and Cloudflare, and 47 non-European countries were served by 20, 14, and 15 edge serves from the same three CDN providers. This implies that, for example, using Cloudfront's edger server, we are able to at least narrow a user's location down to two countries on average, if the user is within Europe (i.e, $46/23/ = 2$). We present the visualized map of Cloudfront's result in Fig. 4, in where we use different colors to identify countries being served by different edge servers. The same result from Fastly and Cloudflare are presented in the Appendix as in Figs. 8 and 7.

Observing these figures, we are able to find edge server deployment does present strong regional characters. Take Fig. 7 as an example, we can observe that the few adjacent countries in middle Europe including Austria, Slovenia, Croatia, Serbia, and Slovakia are all served by one edge server (i.e., all colored with the same Grey color). Furthermore, by comparing the maps between different CDNs, we notice different CDN's have different edge server deployment strategies. For instance, in Fastly's edge server map, we can see while Austria and Slovakia are still served by the same edge server, Solvenia was instead served by the edge server that also serves Italy. And Croatia and Serbia were served by another different edge server. This implies that a finer granularity of user geolocation can be achieved by leveraging multiple CDNs, similar to user location using cellular towers with triangulation [34].

In specific, by holistically considering these 3 CDNs, the 93 countries now see 57 different Cloudfront-Cloudflare-Fastly edge server combinations, a higher resolution than any of the three single CDNs. For the 46 European countries, they can now be separated into 35 categories, a 25% increase in accuracy, i.e.,

a European user can now be narrowed down into an average 1.3 countries. For the 47 non-European countries, they can be separated into 23 categories. We present the new geolocation map leveraging all 3 CDNs in Fig. 5. Comparing with Figs. 4, 7, and 8, it is obvious that higher accuracy has been achieved, as less adjacent countries shares the same color.

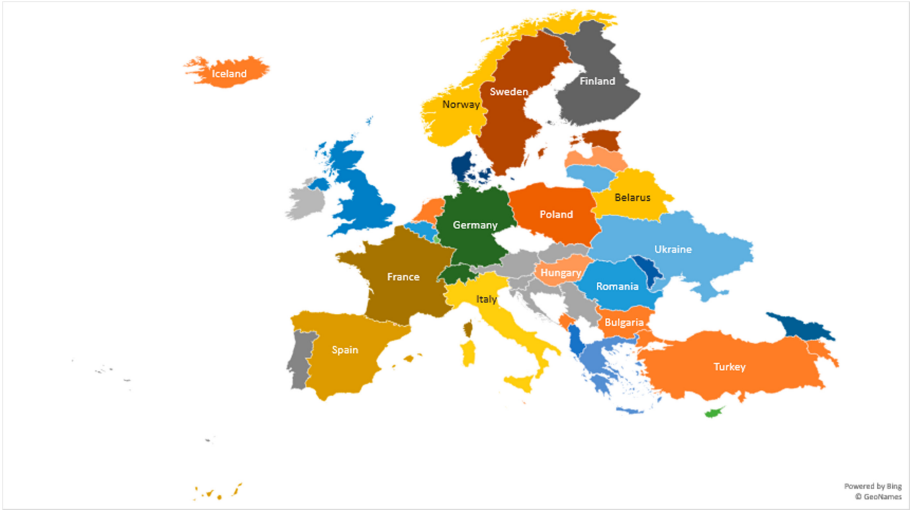

Fig. 4. European countries served by different Cloudfront's edge servers.

State and City Level Geolocation. Next, we present the result to identify states and cities in the U.S. Totally, Express VPN has 27 endpoints locates within the United States, which were distributed among 13 cities that belong to 10 states (excluding Washington, D.C.). Metropolitan cities such as Los Angeles have more than one endpoint. Shown in Fig. 6 is the result when all 3 CDNs are leveraged to geolocate the states where these endpoints are located. Because these states are geographically sparse, we found any one of the 3 CDNs alone is capable of uniquely identify these states. Among these ten states, Florida State has two endpoints located in Tampa and Miami. California State has two endpoints located in Los Angeles and San Francisco. All these cities can also be uniquely identified by either one of these 3 CDNs.

Sub-city Level Geolocation. Finally, we evaluate the accuracy of cache-based geolocation within the sub-city level. In particular, within the U.S., five metropolitan cities have more than one endpoints, which are: Los Angeles that has seven endpoints, Dallas, Miami, New York, and Washington, D.C., each has two endpoints. According to our experiment result, Cloudflare has the lowest resolution in identifying sub-city level locations, for example, all seven endpoints in

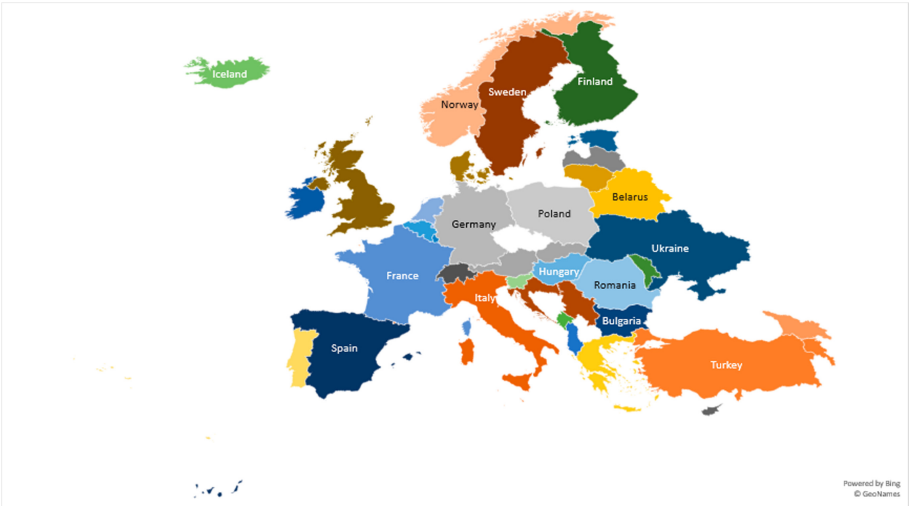

Fig. 5. European countries served by leveraging 3 CDNs' edge servers.

Los Angeles were served by the edge server `LAX`, while Cloudfront has the highest resolution on the other hand, which alone can identify five endpoints located in LA. When putting together, the seven endpoints can be differentiated into six categories, implying satisfactory resolution in high population cities. Detailed such result is demonstrated in Table 3. For the other four cities, except the two endpoints in New York, all other endpoints can be uniquely identified when three CDNs being utilized.

Table 3. 7 VPN endpoints in Los Angeles served by 3 CDN edge serves.

Endpoints' name	Cloudfront	Fastly	Cloudflare
Los Angeles	LAX3-C1	BUR	LAX
Los Angeles-1	LAX3-C3	LAX	LAX
Los Angeles-2	LAX3-C4	BUR	LAX
Los Angeles-3	LAX3-C1	BUR	LAX
Los Angeles-4	LAX50-C1	BUR	LAX
Los Angeles-5	LAX3-C3	BUR	LAX
Santa Monica	LAX3-C2	LAX	LAX

5.3 Discussion and Future Works

As demonstrated by the experiments, the cache-based user geolocation is able to achieve *at least* country-level granularity in the regions where CDN servers are

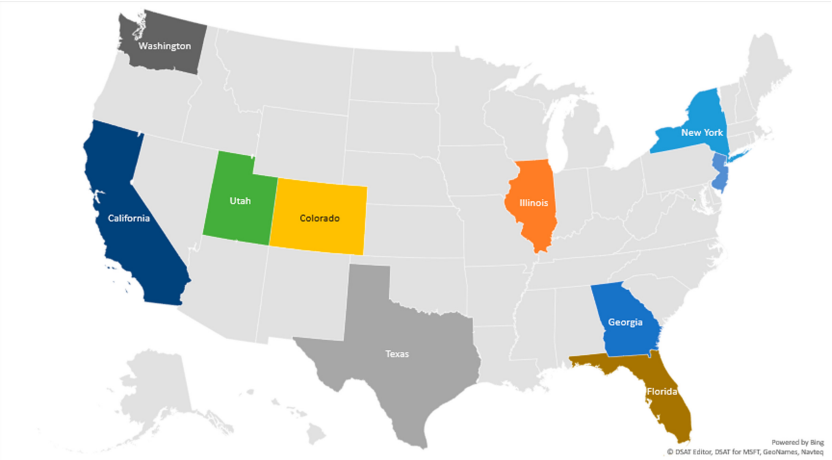

Fig. 6. U.S. states identified by leveraging 3 CDNs.

densely deployed. Due to our limited resource, we were not able to evaluate at smaller granularity, however, based on the results we have obtained, it is evident that finer granularity can be achieved. In specific, the VPN we used only has 46 endpoints present in Europe, which touched only 23, 12, and 18 edge servers belong to Cloudfront, Fastly, and Cloudflare, respectively. However, according to their official websites, these 3 CDNs posses 59, 13, and 47 total edge servers in Europe correspondingly. Therefore, if we were able to obtain more endpoints for evaluation, we can achieve much finer granularity. As such, our future work will focus on seeking more endpoints and conduct more comprehensive evaluations.

6 Conclusions

In this paper, we proposed the *CacheLoc* as a novel user geolocation solution. This cache-based user geolocation solution is easier, cost-free, and more reliable compared to conventional IP-based ones. With limited resources, we conducted multiple experiments to evaluate the usability and accuracy of *CacheLoc*, and our results demonstrate the cache-based approach is feasible and effective for coarse-grained user geolocation. We will be focusing on obtaining more resources for more comprehensive *CacheLoc* evaluation for our future works.

A Appendix

Fig. 7. European countries served by different Cloudflare's edge servers.

Fig. 8. European countries served by different Fastly's edge servers.

References

1. IP2Location: Ip2location (2020). https://www.ip2location.com/
2. Whois: Whois (2020). https://www.whois.net/

3. Schwartz, M.J.: Google wardriving: how engineering trumped privacy (2020). https://www.darkreading.com/risk-management/google-wardriving-how-engineering-trumped-privacy/d/d-id/1104126
4. Li, M., Luo, X., Shi, W., Chai, L.: City-level IP geolocation based on network topology community detection. In: 2017 International Conference on Information Networking (ICOIN), pp. 578–583. IEEE (2017)
5. Google: Geolocation API developer guide (2020). https://developers.google.com/maps/documentation/geolocation/intro
6. Taylor, J., Devlin, J., Curran, K.: Bringing location to IP addresses withIP geolocation. J. Emerg. Technol. Web Intell. **4**, 273–277 (2012)
7. Poese, I., Uhlig, S., Kaafar, M.A., Donnet, B., Gueye, B.: IP geolocation databases: unreliable? ACM SIGCOMM Comput. Commun. Rev. **41**(2), 53–56 (2011)
8. Shavitt, Y., Zilberman, N.: A geolocation databases study. IEEE J. Sel. Areas Commun. (JSAC) **29**, 2044–2056 (2011)
9. MaxMind: Maxmind (2020). https://www.maxmind.com/en/home
10. Ipinfodb: Ipinfodb (2020). https://ipinfodb.com/
11. Google maps platform. https://developers.google.com/maps/documentation/javascript/geolocation
12. Bizety: CDN market size in 2015 and 2019 (2020). https://www.bizety.com/2015/08/15/cdn-market-size-in-2015-and-2019-2/
13. Loulloudes, N., Pallis, G., Dikaiakos, M.D.: Information dissemination in mobile CDNs. In: Buyya, R., Pathan, M., Vakali, A. (eds.) Content Delivery Networks. LNEE, vol. 9, pp. 343–366. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-77887-5_14
14. Amazon: Amazon cloudfront (2020). https://www.godaddy.com/
15. How fastly builds pops
16. Same origin policy
17. MDN web docs: Cross-origin resource sharing (CORS) (2020). https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
18. MDN web docs: Access-control-expose-headers (2020). https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Expose-Headers
19. The majestic million. https://majestic.com/reports/majestic-million
20. United states CDN. https://www.cdnplanet.com/geo/united-states-cdn/
21. The cloudflare global anycast network. https://www.cloudflare.com/network/
22. Cloudflare: Cloudflare (2020). https://www.cloudflare.com/
23. Fastly: Fastly (2020). https://www.fastly.com/
24. Accessing Fastly's IP ranges. https://docs.fastly.com/en/guides/accessing-fastlys-ip-ranges
25. Locations and IP address ranges of CloudFront edge servers. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/LocationsOfEdgeServers.html
26. Cloudflare IP ranges. https://www.cloudflare.com/ips/
27. Holowczak, J., Houmansadr, A.: Cachebrowser: bypassing chinese censorship without proxies using cached content. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 70–83 (2015)
28. What is anycast? How does anycast work? https://www.cloudflare.com/learning/cdn/glossary/anycast-network/
29. A new architecture for the modern internet. https://www.fastly.com/network-map
30. Amazon CloudFront key features. https://aws.amazon.com/cloudfront/features/
31. Alibaba: Alibaba cloud's global infrastructure (2020). https://www.alibabacloud.com/global-locations

32. Planet Lab: Planet lab (2020). https://www.planet-lab.org/
33. Express VPN: Express VPN (2020). https://www.expressvpn.com/vpn-server
34. 4n6.com: Cell phone triangulation (2020). https://4n6.com/cell-phone-triangulation/