Locating CDN Edge Servers with HTTP Responses

Run Huang, Mengying Zhou, Tiancheng Guo, Yang Chen*
Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University, China
{runhuang19,myzhou19,tcguo20,chenyang}@fudan.edu.cn

ABSTRACT

Determining the physical locations of CDN Points of Presence (PoPs) is fundamental to understanding and diagnosing CDN services. Yet, the popular deployment of IP Anycast in CDNs has rendered existing geolocation tools unreliable. To fill this gap, we present an HTTP-based solution that leverages subtle geographic hints in HTTP responses to locate CDN PoPs at the city-level granularity. The evaluation shows that our technique achieves over 90% accuracy with an average error distance of less than 40 km.

CCS CONCEPTS

• **Networks** → **Network measurement**.

KEYWORDS

Geolocation, Measurement, CDN

ACM Reference Format:
Run Huang, Mengying Zhou, Tiancheng Guo, Yang Chen. 2022. Locating CDN Edge Servers with HTTP Responses. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22 Demos and Posters), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3546037.3546051

1 INTRODUCTION

Geolocation of CDN PoPs is of interest to both academia and industry. It is essential to CDN security and performance analysis [2, 17, 18]. Many researchers, web developers, and businesses rely on such information to monitor and diagnose the operation of CDN-related services. However, existing geolocation techniques are mostly IP-based, assuming a single location per IP address, and thereby fail with Anycast CDNs, whose replicas at different locations share the same IP address. This raises the need for a tool that yields more reliable and accurate geolocation results for CDNs.

Fortunately, many CDN providers embed geographic hints (*geohints*) about the physical locations of their edge servers in specific HTTP response headers. Yet, it is not easy to interpret and exploit these hints for geolocation due to their ambiguity, diverse forms, and lack of documentation. To cope with these challenges, we employ text mining techniques and conduct latency measurements to identify geohints and then infer their corresponding locations. Leveraging this approach, we monitored 22 popular CDNs and identified the presence of geohints in 20 of them. We publicly release the dataset of regular expressions for extracting these geohints and the mappings between each hint and a physical location. Based on this dataset, we implemented *LocEdge*, an open-source tool that provides rich information about CDN edge servers (such as vendor, location, and cache hit ratio), aiming to equip web developers and researchers with a handy CDN troubleshooting tool. The dataset and LocEdge can be accessed from *https://github.com/ruh010/locedge*.

Geohints at a Glance. Table 1 summarizes the identified geohints, showing that they may appear anywhere in various HTTP headers, and do not necessarily have semantics, making them difficult to identify and interpret manually, let alone infer their corresponding locations. Worse, only seven headers are confirmed by official documentation to embody geohints, while only five providers openly disclose the geohint-to-city mappings.

Table 1: Identified Geohints (pink italic text)

Type	CDNs	Examples of Geohint in San Jose, CA
IATA Code	Amazon; CacheFly; CDNetworks; Cloudflare; Azure; Fastly; Tata;	CDN: CacheFly; Header: X-CF1 *****:fF.*lax3*:co:**********:cacheN.*sjc1*-01:B
Location Abbreviation	Edgecast; KeyCDN; StackPath; CDN77; Bunny;	CDN: StackPath; Header: X-HW **********.cds212.*sj3*.hn,*********.cds218.*sj3*.c
Customize ID	Alibaba; ArvanCloud; Beluga; OVH; G-Core; Medianova; Sucuri; Section;	CDN: Beluga; Header: X-Beluga-Node ***89bfd0a5-6bc3-4f66-8f39-c245d891092a***

Related Works. Luckie et al. [12] proposed a learning-based approach to extract geohints from DNS names to locate Internet routers, which enables researchers to infer the location of an endpoint via traceroute. However, in §3 we show that this method is not widely applicable in practice. Cicalese et al. [4] used a latency-based approach to locate Anycast replicas, but it requires lots of probes to conduct active measurements, making it less convenient to use than database-driven methods like [7, 8, 12, 15] and ours.

2 SYSTEM OVERVIEW

Fig. 1 illustrates our system architecture. A *Monitor* performs HTTP measurements toward target websites. Results are parsed by an *Extractor* to retrieve geohints, whose corresponding locations are inferred by a *Localizer*. Each component is described in detail below.

Monitor. We monitor the official websites of 22 popular CDNs from 229 vantage points (VPs) deployed on Uptrends, a web monitoring platform [11]. For each visit, we collect HTTP responses and the TCP handshake time. The latter is used to describe network latency. Considering that many CDNs operate servers in only a few dozen locations, we argue that this amount of VPs is sufficient to enumerate most PoPs. However, a major drawback is that nearly 75% of Uptrends VPs are in Europe and N. America. Had there been more in S. America and the Middle East, we could achieve higher accuracy and coverage for those more widely distributed CDNs.

Extractor. We cannot apply the geohint extraction technique in [12] directly because it is based on the observation that geohints in DNS names follow specific naming conventions, which it is not the case for geohints in HTTP headers, as shown in Tab. 1. Still, analysis of intelligible geohints (i.e. those of which the corresponding

*Corresponding author

SIGCOMM '22 Demos and Posters, August 22–26, 2022, Amsterdam, Netherlands

ACM ISBN 978-1-4503-9434-5/22/08.
https://doi.org/10.1145/3546037.3546051

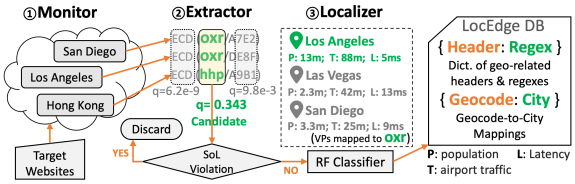

Figure 1: System Architecture

Table 2: Model accuracy and mean error distance

CDN	Acc. (%)	Mean Error (km)		CDN	Acc. (%)	Mean Error (km)	
		LocEdge	Baseline★			LocEdge	Baseline★
Amazon	92.5	25.41	1056.20	KeyCDN	95.0	26.65	571.26
CacheFly	93.2	20.74	2238.12	StackPath	90.3	30.71	2116.04
Fastly	80.8	62.34	1831.47	Tata	94.7	44.40	1086.03

★ The optimal result combining three geolocation datasets [7, 8, 12]

location is disclosed by the vendor) revealed that a given geohint appears frequently in a small set of adjacent VPs while rarely in the others. Given that, we employ TF-IDF [14] to distinguish them from gibberish, which is widely used for uncovering terms with a similar characteristic, i.e. occurring frequently in a document but rarely in the whole corpus. For simplicity, we divide the header into several segments by punctuation. Then, we apply a sliding window on each segment to enumerate the set of regular expressions for all possible terms (based on the relative position to the beginning of the segment), and calculate the likelihood of each regex as the mean TF-IDF score of all terms it extracts. Regex with the highest likelihood would be a candidate for geohint extraction.

As in [4, 12], we filter out ineligible candidates by detecting Speed-of-Light violations among the terms they extract. For example, VPs at San Jose and Tokyo both obtained "imperva" from header "x-cdn" by a candidate, and their RTTs were 3 ms and 5 ms respectively. However, even light cannot travel from Tokyo to San Jose, CA (8332km) in 8 ms, indicating that the two VPs were in contact with different servers. Thus, "imperva" is not a geohint. For a more conservative result, we apply the median inflation factor of the TCP handshake time over the speed-of-light latency [16], assuming that a packet travels 60km in 1 ms. Confirmed geohints are then passed to the *Localizer* for location inference.

Localizer. We transform the geolocation task into a classification problem. For each confirmed geohint g and the set of VPs mapped to it, we use a Random Forest Classifier to compute the probability of each VP being where g's corresponding PoP locates, and then assign g to the one with the highest probability. The classifier takes the first quartile RTT, the population, and airport traffic as features. It is trained with officially labeled geohints released by Edgecast [6] and Cloudflare [5]. The population metric is intended to bias the results toward large cities where CDNs tend to place their servers [3]. However, in regions with a dense distribution of CDN PoPs, the geohints of some major cities that have relatively small populations are often mislabeled as their nearby megacities (e.g. mislabeling Amsterdam/Osaka as London/Tokyo). We remedy such deficiency by using airport traffic as a complementary metric, which further describes the influence and importance of a city.

Finally, the system generates a dataset of (a) geo-related headers and regexes for geohint extraction (b) inferred mappings between each hint and a city. We will show that this dataset is accurate and reliable in §3. Leveraging it, we implemented a CDN debugging tool *LocEdge* for the community. For each resource recorded in the HAR file [13] generated during web navigation, LocEdge uses the above dataset to localize the edge server and retrieves hints of cache status (HIT or MISS) from cache-related headers (e.g. x-hw). Note that in order to expand the scope of applicability, for CDNs that do not provide geohints (e.g. Akamai and Imperva), LocEdge performs traceroute to the resolved IP address of the edge server and infers its location by locating routers on the path using Hoiho [12].

3 EVALUATION AND DEMONSTRATION

Ground Truth. We obtain validation data for StackPath and KeyCDN by crawling their incident pages [9, 10], and use the IATA airport code [1] dataset as the ground truth for four CDNs that are inferred to employ variants of the IATA code as geohints.

Baseline. For every endpoint, we evaluate our approach against the best among the results of multiple geolocation techniques. We run traceroutes toward target websites and identify the location of an edge server by locating routers on the last five hops on its path using Hoiho [12]. Notably, we find that only 21% of the paths we measured can be located in this way, as many operators do not give away geohints in DNS names. For the remaining paths, the location of their destinations is determined by querying the IP address of the penultimate hop in two popular commercial GeoIP databases [7, 8]. Our reasoning for geolocating the penultimate hop instead of the last hop is that the former is usually non-Anycasted, and thus provides higher accuracy. We use the minimum error distance among the three results as the baseline for each endpoint.

Evaluation. The results are shown in Tab. 2, where accuracy refers to the percentage of our approach that correctly identifies the corresponding city of a geohint. It shows that our technique generally performs well, and significantly outperforms the baseline. Erroneous results are mainly caused by the lack of VPs in S. America and the Middle East. Interestingly, CDNs' inefficient routing also accounts for inaccuracies. In our 14 days (May 4 to May 18, 2022) of monitoring, KeyCDN consistently directed clients in Hong Kong to servers in Los Angeles instead of the local one, resulting in the mislabeled geohint "cnhk". Further Ping tests to *keycdn.com* revealed that the average RTT in Hong Kong is 160 ms, compared with just 0.8 ms in other cities with a local PoP, indicating that this poor proximity is not down to a specific protocol, but rather an issue on the operator and ISP side. Such suboptimal client-server mappings exist across all six providers and have distinct geographical patterns (e.g. traffic from S. America tends to be directed to Miami). These findings again raise the need for a more comprehensive evaluation of CDNs, especially those relying on Anycast.

In conclusion, our approach effectively discovers the embedded geohints in HTTP headers and uncovers their corresponding locations. It generalizes well on unseen CDNs despite being trained with only a handful of data from two vendors. Hence, we believe that the inferred geohint-to-city mappings for the remaining CDNs that have no ground truth for validation are also accurate and reliable.

Demonstration. We built a demo website showcasing LocEdge's usage in CDN monitoring at *https://locedgex.web.app/*. Upon a request to check an URL, a backend server will visit it and report any CDN-related issues (e.g. PoP-switching, poor proximity, low cache hit ratio) and details about the edge servers that handled the requests. The website also displays a map of PoPs we observed, which helps to understand where CDN vendors deploy their infrastructures.

REFERENCES

[1] International Air Transport Association. 2022. IATA Airline and Location Codes. https://www.iata.org/en/services/codes. Accessed: 2022-05-01.

[2] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. 2015. Analyzing the Performance of an Anycast CDN. In *Proc. of IMC*.

[3] Danilo Cicalese, Jordan Augé, Diana Joumblatt, Dario Rossi, Marc-Olivier Buob, and Timur Friedman. 2015. Lightweight Anycast Enumeration and Geolocation. In *INFOCOM WKSHPS*.

[4] Danilo Cicalese, Diana Zeaiter Joumblatt, Dario Rossi, Marc-Olivier Buob, Jordan Augé, and Timur Friedman. 2016. Latency-Based Anycast Geolocation: Algorithms, Software, and Data Sets. *IEEE J. Sel. Areas Commun.* 34, 6 (2016), 1889–1903.

[5] Cloudflare Inc. 2022. Cloudflare Status. https://www.cloudflarestatus.com/. Accessed: 2022-05-01.

[6] Edgecast Networks Inc. 2022. Edgecast PoP listing. https://docs.edgecast.com/cdn/Content/Reference/POP_Listing.htm. Accessed: 2022-05-01.

[7] MaxMind Inc. 2022. GeoLite2 Free Geolocation Data. https://dev.maxmind.com/geoip/geolite2-free-geolocation-data. Accessed: 2022-05-31.

[8] IP2Location. 2022. IP2Location™ IP Address Geolocation Database. https://www.ip2location.com/database/ip2location. Accessed: 2022-06-28.

[9] Proinity LLC. 2022. KeyCDN Status. https://status.keycdn.com. Accessed: 2022-05-18.

[10] StackPath LLC. 2022. StackPath Status. https://status.stackpath.com. Accessed: 2022-05-18.

[11] Uptrends LLC. 2022. Website monitoring and web performance monitoring. https://www.uptrends.com. Accessed: 2022-05-01.

[12] Matthew Luckie, Bradley Huffaker, Alexander Marder, Zachary Bischof, Marianne Fletcher, and Kimberly C. Claffy. 2021. Learning to Extract Geographic Information from Internet Router Hostnames. In *Proc. of CoNEXT*.

[13] Jan Odvarko. 2022. HAR 1.2 Spec. http://www.softwareishard.com/blog/har-12-spec. Accessed: 2022-05-01.

[14] Claude Sammut and Geoffrey I. Webb. 2010. *Encyclopedia of Machine Learning*. 986–987.

[15] Quirin Scheitle, Oliver Gasser, Patrick Sattler, and Georg Carle. 2017. HLOC: Hints-Based Geolocation Leveraging Multiple Measurement Frameworks. In *Proc. of TMA*.

[16] Ankit Singla, Balakrishnan Chandrasekaran, Brighten Godfrey, and Bruce M. Maggs. 2014. The Internet at the Speed of Light. In *Proc. of HotNets*.

[17] Jing'an Xue, Weizhen Dang, Haibo Wang, Jilong Wang, and Hui Wang. 2019. Evaluating Performance and Inefficient Routing of an Anycast CDN. In *Proc. of IWQoS*.

[18] Mengying Zhou, Tiancheng Guo, Yang Chen, Junjie Wan, and Xin Wang. 2021. Polygon: A QUIC-Based CDN Server Selection System Supporting Multiple Resource Demands. In *Proc. of Middleware*.