Public Review for

Towards client-side active measurements without application control

Palak Goenka, Kyriakos Zarifis, Arpit Gupta, Matt Calder

Network Error Logging (NEL) is a W3C standard which defines how web servers can receive from a browser reports about performance and failures of web requests. This is achieved by setting an HTTP response header which specifies an HTTPS endpoint where the browser will upload such reports. The authors propose harnessing NEL to enable active client-side measurements (RTT and connection availability) by dynamically modifying the HTTPS endpoint where NEL reports should be uploaded. These techniques enable active client-side measurements in the browser without requiring Javascript code injection, which is the current and more invasive state of the art solution. The reviewers particularly appreciated the clever re-purposing of NEL in the context of CDNs. The paper is technically sound and the idea proposed can be very valuable to improve CDN performance. Last but not least, reviewers appreciated that code and testbed configuration used in the paper were made open source.

Public review written by
Matteo Varvello
Nokia Bell Labs

Towards client-side active measurements without application control

Palak Goenka
Microsoft

Arpit Gupta
UCSB

Kyriakos Zarifis
Edgecast

Matt Calder
Microsoft
Columbia University

ABSTRACT

Monitoring performance and availability are critical to operating successful content distribution networks. Internet measurements provide the data needed for traffic engineering, alerting, and network diagnostics. While there are significant benefits to performing end-user active measurements, these capabilities are limited to a small number of content providers with application control. In this work, we present a solution to the long-standing problem of issuing active measurements from clients without requiring application control, e.g., injecting JavaScript to the content served. Our approach uses server-side programmable features of the Network Error Logging specification that allow a CDN to induce a browser connection to an HTTPS server of the CDN's choosing without application control.

CCS CONCEPTS

• **Networks** → **Network measurement**; **Network performance analysis**;

KEYWORDS

Internet Measurement, CDN, Content Delivery Network, Latency, DNS, NEL, RUM

1 INTRODUCTION

Web applications are ubiquitous on today's Internet—offering quick access to popular services such as email, video conferencing, chat, streaming media, and cloud storage. Competition is fierce amongst online cloud services, and operators know that poor performance negatively impacts revenue [28, 37]. To ensure good user experience, content providers and content delivery networks (CDN) monitor the health of their networks and services with measurements that reflect network quality and user experience.

Client-side active measurements are important to CDNs (§2), yet often inaccessible as CDNs lack end-user application control. For the web, application control is enabled by either controlling the server generating page content or in-browser executable code such as JavaScript. The state-of-the-art and long-standing solution is Real User Monitoring (RUM), where JavaScript is dynamically injected into webpage HTML by the CDN (with permission) or embedded by the application owner [31]. With RUM JavaScript, the CDN gains a bit of application control through a small function within a larger application. The JavaScript runs in a user's browser where it performs measurements and reports the results back to the CDN.

In principle, web application owners and CDNs can use RUM for client-side measurement, however, it requires embedding a JavaScript in an HTML document, i.e., it requires some level of application control. Given that much of the content served by CDNs is static content such as video, images, CSS, and JavaScript libraries (e.g. jQuery), opportunity for such JavaScript injection is limited, which can affect RUM adoption by CDN operators. Kashaf et al. [26] showed that dependence on content from third-party CDNs (those not serving the HTML) is common – in the Alexa top-100, 76% of sites depend on third-party CDNs and 89% for the top-1000. Even for the cases where CDNs do serve the HTML for a site, our interactions with CDN operators confirms low adoption rates for their RUM-based measurements platforms.

In this paper, we present the design of a **Cl**ient-side **A**ctive **M**easurement platform, CLAM, that leverages features offered by Network Error Logging (NEL)[16], a W3C standard implemented in Chromium-based browsers. NEL provides a standard way for web servers to receive *reports* about performance and failures of web requests from a browser by setting HTTP response headers, one of which allows the server to specify an *HTTPS* endpoint where the browser will upload reports. CLAM piggybacks on NEL report upload *HTTPS connections* to enable measurements between an end-user and target servers. Thanks to the Chrome browser's dominant market share, as seen in Table 1, this technique works today on an estimated 60-69% of web users, enabling server-controlled measurements from previously inaccessible vantage points. Unlike RUM-based approaches, it requires no custom JavaScript injected into web applications. With CLAM, any request served by any web server is an opportunity to perform a client-side active measurement.

The code and testbed configuration used in this work is available at https://github.com/inspace/clam.

2 MOTIVATION

A CDN's main objective is to provide good performance to users. It achieves this goal by serving content from a server in a point-of-presence (PoP) nearby a user. The best-performing PoP for a user may not be geographically closest one, so CDNs prefer to make decisions based on path quality between users and PoPs based on measurements. CDNs may use many different metrics to decide the best performing PoP, such as a latency, throughput, and availability [21], depending on the application. CDNs that use DNS to select PoPs [23] (e.g., Akamai) also require control-plane measurements to know how to direct users to PoPs. Most of the time, these CDNs must make per-LDNS (recursive resolver) decisions, not per-end-user IP address, so knowing the set of users served by an LDNS is essential for good performance [32].

Recent work has shown that there are significant benefits for CDNs in using client-side measurements. Two measurement platforms, NEL [19] from Google and Odin [21] from Microsoft, showed

that client-side measurements are particularly useful for detecting Internet outages in real-time by reporting errors that prevent users from connecting but are invisible to servers. When collecting performance data for PoP selection, client-side active measurements allow web applications to measure different servers from the same user quickly. Measurements over HTTP(s) requests have the benefit of operating in the application layer and rarely suffer from de-prioritization or firewall drops. For the DNS control plane, the only known techniques for associating clients with LDNS IP addresses are through client-side active measurements – either through direct resolution [7] or through resolution of unique hostnames [20, 25, 30].

JavaScript is the standard for client-side measurements on the web and is often referred to as RUM (Real User Monitoring) [31]. JavaScript can passively collect application (e.g., page-load-time, above-the-fold) and network performance data or initiate and instrument synthetic HTTP requests for non-rendered resources from different servers outside of the critical rendering path. The script then uploads results to a collection server. CDNs such as Akamai [1, 2] and Fastly [4] have created RUM platforms where customers opt-in to embed a script to their application so that the CDN receives network performance data.

From speaking with several CDN operators, one of the main challenges with these platforms is adoption. Many customers prefer to deploy their own RUM code to maintain control of code and data, rather than rely on third-party JavaScript as it raises performance, security, and privacy concerns [29, 39].

Measurement platforms [3, 11] with existing vantage points in end-user networks are inherently biased as they cannot represent the CDN's actual population and traffic. Previous work has shown that ICMP probes from servers to users have limited coverage with only 60% responsiveness [25, 42] and fail to capture application layer faults such as TLS certificate errors or web server failures (e.g., HTTP 5XX response codes).

Next, we describe CLAM, which overcomes the challenges above by leveraging NEL, a standard implemented in Chromium-based browsers.

3 USING NEL FOR CLIENT-SIDE ACTIVE MEASUREMENTS

In this section, we provide an overview of Network Error Logging (NEL) and then describe how CLAM uses it to perform client-side active measurements and related challenges.

3.1 What is Network Error Logging (NEL)?

NEL is a W3C specification [16] for reporting client-side networking errors and performance in web applications. Chrome, Microsoft Edge, Opera, and other browsers based on Chromium inherit NEL functionality. It monitors and reports low-layer networking metrics that are not exposed to other JavaScript APIs. For example, NEL reports `dns.unreachable` when a DNS server cannot be reached, and `tcp.timed_out` when the TCP connection to a server times out [16]. In addition to failed attempts, NEL can be used to report high level timings of successful downloads. NEL enables a web server to specify a measurement policy for a domain under its control by setting the `NEL` and `Report-To` headers in the HTTP response as shown in Listing 1. In this example, the `NEL` header specifies a policy to sample 25% of successful requests and 100% of failed requests for up to 24 hours.

Device	Share%	Metric	Source
Desktop	69.5	Page Views	NetMarketShare
Mobile	64.5	Page Views	NetMarketShare
Desktop	68..3	Uniq Visitors	StatCounter
Mobile	63.0	Uniq Visitors	StatCounter
All	49.2	Page Views	U.S. DAP
All	80.7	Page Views	w3schools.com

Table 1: Google Chrome estimated market share in May, 2021 from NetMarketShare[9], StatCounter[12], U.S. DAP[15], and W3CSchools[17].

```
NEL:{"report_to":"default","max_age":86400,
"success_fraction":0.25,"failure_fraction":1.0}

Report-To:{"group":"default","max_age":86400,
"endpoints":[{"url":"https://test.com/report"}]}
```

Listing 1: Example NEL and Report-To response headers to configure a measurement policy.

When installed on a browser, a NEL policy applies to all resources from the same domain, and optionally sub-domain, for the `max_age` duration. A CDN can define a single NEL policy for a given domain. Updating the NEL policy for a domain overwrites the existing policy on a browser. All the subsequent reports are uploaded to the custom endpoints specified by the most recent NEL policy. These *report endpoints* are specified in the `Report-To` header, as is a list of URLs where the browser will upload reports in the background after a short delay, outside of the browser's critical rendering path. Chrome uploads the NEL reports at most every 60 seconds. The content of the report depends on the number of successful and failed requests made by the browser during this interval. NEL supports uploads using multiple redundant failover paths. If the first report endpoint is unreachable, the browser retries on subsequent endpoints [19].

NEL-based active measurement platforms will have strong coverage, which is evident from the large footprint for the Google Chrome browser (one among many Chromium-based browsers) across different device types (see Table 1). However, the success of such measurement platforms will also depend on content provider and CDN platform adoption. To answer this question, we analyze the public data from HTTP Archive [6] for February, June, and September 2020 to characterize NEL's adoption trend. This data contains page load information collected from controlled clients, which regularly load 5.5 M Alexa websites. The data exposes the HTTP response headers sent to the controlled clients from the target web sites, among other information. We look for such headers in three snapshots of the data to evaluate the increase in NEL adoption over time.

Over the period of eight months, we witness a growth trend among metrics such as number of unique web pages (120% increase), unique domains (20.1% increase), and responses with NEL headers (40.1% increase) (see Table 2). We also see a similar trend for origins that make use of multiple endpoints. We observe an explosion in the number of unique endpoint URLs in the NEL header in September. However, further investigation reveals that 99.8% of those belong to

With NEL headers	Feb '20	Jun '20	Sep '20
Unique web pages	3.4% of 4.2M	5.2% of 4.8M	7.5% of 5.4M
Unique domains	2.84% of 3.1M	3.37% of 3.8M	3.41% of 4.4M
Responses	0.17% of 357.6M	0.22% of 452M	0.24% of 509M
Unique report URLs	250	340	**187.5K**
Origins with multiple endpoints	40	44	93

Table 2: NEL's adoption trend over 2020.

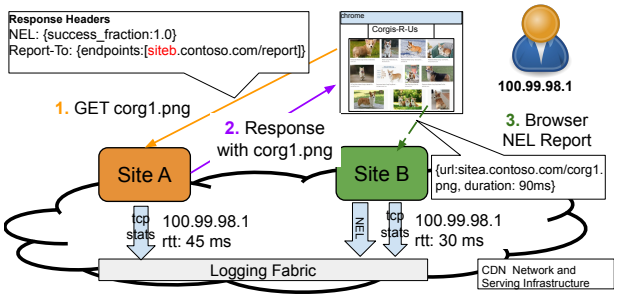

Figure 1: The CDN acquires a measurement from the user to Site B, even though Site A actively serves the user. Site A sets the Report-To endpoint to be a Site B URL. In the background, the browser uploads a NEL report to the Site B URL.

a single CDN that adopted NEL recently, and they correspond to the same endpoint but contain unique query arguments. Overall, we witness growth in the adoption of NEL—motivating the design of a NEL-based network measurement infrastructure. We now present how we use NEL to enable client-side active measurements.

3.2 How to use NEL for Client-side Active Measurements?

CLAM relies on the ability to elicit HTTPS connections from the client to arbitrary servers to perform measurements. It **does not** use the actual report data generated by NEL (but which are very useful in their own right).

3.2.1 Proposed Approach: CLAM. We now highlight the opportunities offered by NEL that enable client-side active measurements without requiring application control. More concretely, NEL: (1) covers the majority of web users due to its large browser market share (Table 1); (2) allows the server hosting the content to specify *arbitrary* HTTPS URL endpoints for uploading the report; (3) allows reports for all requests, irrespective of whether they failed or not.

We use NEL's footprint to actively send reports from clients to the end points of our choice—enabling active measurements from clients to the end points. CLAM dynamically updates the measurement targets by changing the resolved IP address for the endpoint URL or updating the endpoint URL on subsequent responses to the same client. It also controls the volume of these reports by configuring the sampling rates for the successful and failed requests. At the endpoints, CLAM uses the server-side connection profiling to capture different performance metrics (e.g., RTT) for the incoming upload requests.

3.2.2 Illustrative example. Figure 1 illustrates how CLAM works. A user with IP address `100.99.98.1` visits a popular website for hosting images. A CDN serves some of the images. For simplicity, this CDN only has two PoPs: *Site A* and *Site B*. Both sites have server-side network telemetry enabled that captures information like client IP address and RTT.

When the webpage loads, the browser resolves the hostname of the `corg1.png` image, which the CDN directs to *Site A* and the browser makes a *HTTP GET* request**(1)**. In *Site A*'s HTTP response **(2)**, it includes the NEL header specifying that successful requests should be sampled at 100% and that reports should be uploaded to *https://siteb.neltest.com/report*. Shortly after the image is received[1], the browser makes a *HTTP POST* with the NEL report data to *https://siteb.neltest.com/report*. Note here that the browser controls the NEL report upload timing and at the time of this writing there is no server or user settings to modify the upload interval or start time.

3.3 How does CLAM configure NEL Headers?

CLAM can configure different NEL headers to meet the measurement goals of a CDN while managing load. For each request, it decides whether to set the NEL header or not, and then set the endpoint URL, success and failure sampling rates; and the `max_age` field for the NEL header. By doing so, CLAM strike a balance between the quality of measurements and the workload at target servers.

Selective NEL header insertion allows to select which clients should be included in measurement campaigns. This can be used to control load or exclude all measurements from particular clients such as those participating in a DDoS attack, in sensitive geo-political regions, or in networks with very expensive mobile data plans.

Endpoint selection defines the target of the active measurement campaign. Despite that CDNs often operate hundreds of PoPs, only a small subset can serve content with low latency for a specific user. So a CDN can choose a smaller set of PoPs nearby the client to measure in order to decide the lowest-latency PoP. The sampling rates for success and failure may also control load, but conditionally, based on the request status observed from the client. For example, one might decide to monitor only failed requests (as opposed to all requests) for a domain. These can also control the size of NEL reports, affecting the quality of measurements, especially those requiring multiple RTTs. The `max_age` allows CLAM to control how often to re-evaluate a client's measurement policy to respond to changes in load, performance, or outages.

3.4 Limitations

While CLAM enables client-side active measurements at scale without requiring application control, it has some limitations, which we describe below:

Delay. There can be a delay of up to 60 second between client fetching the content from the web server and uploading the report at the end point. Such a delay is not acceptable for use-cases that require capturing the network performance dynamics at shorter time scales.

[1]Within 60 seconds in Chrome version 81.0.4044.129

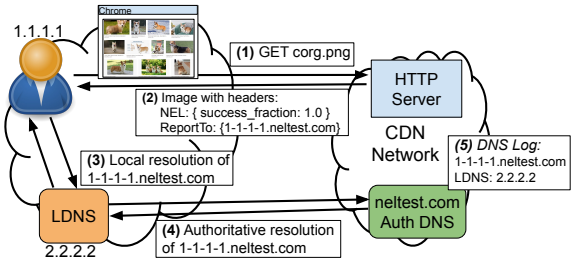

Figure 2: A CDN can associate client and LDNS IP addresses by generating custom domains server-side, and forcing the client browser to resolve it for the NEL report upload.

Targets. As CLAM extracts the active measurement performance metrics at the report endpoints, it can only probe targets under the experimenter's (CDN's) control.

Metrics. Currently CLAM is well suited for capturing latency and availability which is critical for a large number of web applications. We leave evaluation of other metrics such as jitter, packet loss, and throughput to future work. Unlike RUM, CLAM cannot capture user QoE metrics such as page load time (PLT).

Web-Only. CLAM is currently limited to browsers implementing the NEL specification. However, in principle, any desktop or mobile application using web requests or HTTP library could support NEL.

3.5 Ethical considerations.

Just as with NEL [19], CLAM complies with the following principles to preserve the end-user privacy: (1) a server cannot collect information about end-users it does not already have access to; (2) no measurements are performed outside of regular user activity; (3) end users can opt-out of NEL; and (4) only the site operators can configure collection for their site and where reports are uploaded.

4 CLAM USE CASES

In this section we describe how CDNs can use the client-side measurements enabled by CLAM.

4.0.1 Alternate PoP measurements. Directing users to a nearby PoP for good performance is fundamental for CDNs (§2). Because network conditions on the Internet change frequently, CDNs use network performance data to decide which PoP or path will provide a user with the best performance [20, 23, 36, 41]. Figure 1 illustrates a simple example using two distinct CDN PoPs, but can easily extended to more complex scenarios. Since CDNs can have 100s or 1000s of PoPs, a control system is needed to program response headers for useful, coordinated measurements based on client network and geographic characteristics. CLAM removes the need to do network experiments to alternate PoPs with production traffic, reducing harm to end-user performance and enabling measurement for cloud and enterprise traffic populations.

User to PoP measurements provide fine-grain per-client IP address performance data but the unit of redirection control for many CDNs is the user's LDNS. So, we next look at how CLAM can measure client-LDNS association.

4.0.2 Client-LDNS association. Understanding the relationship between clients and their LDNSes is also critical for good CDN performance. Figure 2 shows how CLAM captures this by adapting an existing technique [30], but without modifying the HTML[2]. A user with IP address `1.1.1.1` is viewing an image sharing website. The browser fetches one of the images **(1)** from a CDN server. The server responds with the image data, along with **(2)** a NEL policy and the Report-To endpoint set to `https://1-1-1-1.neltest.com`, with the user's IP address encoded in the CDN's hostname, which the server knows from the request's remote socket IP address. When the browser starts the NEL report upload process **(3)**, it will resolve `https://1-1-1-1.neltest.com`, which will first be forwarded to the client's LDNS (2.2.2.2), and assuming no cached record exists, will forward the request to the CDN's authoritative resolver **(4)**. The authoritative DNS logs capture the association through the client's IP address in the hostname to resolve, and the LDNS's IP address from which it received the request **(5)**.

4.0.3 Availability monitoring. A CDN must monitor PoP availability to know if client traffic is unable to reach it. Monitoring can be detected by drops in traffic volume [34], client-side active measurement [21], or commercial monitoring platforms [3, 14]. When an availability issue is detected, CDNs can mitigate the issue by moving traffic to an alternate PoP. But without additional probes to verify a fix, users cannot be safely switched back without the risk of blackholing traffic. By continuing to measure the degraded PoP with actual user web requests, the CDN can verify that end-to-end connectivity works before shifting traffic back. If continuous measurement was already in place, one could verify that the number of successful uploads to the degraded PoP is returning to historically "normal" values.

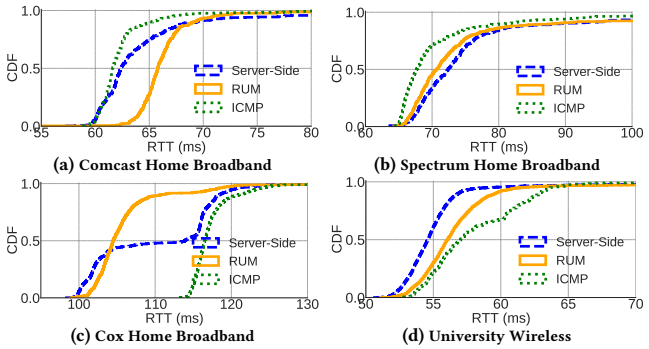

Figure 3: A comparison of Server-Side RTT captured using CLAM as compared to RUM estimated RTT and ICMP ping from 4 different end-user networks.

5 EVALUATION

In this section, we showcase CLAM addressing the following questions: (1) how effectively does CLAM capture network latency, compared to existing approaches?; (2) how effectively does CLAM track changes in user performance?; and (3) how does CLAM dynamically reconfigure network measurement policy at runtime? Finally,

[2]We use Mao's technique for illustrative simplicity, not for user tracking. In practice, randomized ids achieve the same result and offer greater privacy [21].

we demonstrate how CDNs can use CLAM to capture client-LDNS mappings.

5.1 Testbed Setup

We configured a virtual machine (VM) to run an NGINX webserver with `TCP_INFO` [13] to gather per-flow metrics. Then we created a mini testbed by deploying four such VMs in four different Microsoft Azure regions [8]: West US2 (Seattle), North Central US (Chicago), France Central (Paris), and Southeast Asia (Singapore). Two VMs (West US2 and France Central) were also configured to serve authoritative DNS for our test domain. On receiving (non-NEL) requests from the clients, the web servers use a pre-specified configuration to set the NEL response headers. On receiving the NEL reports, the web server uses `TCP_INFO` to extract the RTT value for the connection. The DNS servers log the source IP address (of LDNS) and request to a Redis database. Unless specified otherwise, we used the Private Internet Access (PIA) VPN service [10] to generate client requests from diverse geographical areas.

5.2 How effectively does CLAM capture Network Latency?

To answer this question, we compare CLAM's server-side latency measurements with that of RUM and ICMP ping. For this experiment, we send requests from four different client networks along the West Coast of the United States to a single Central US region VM. To quantify network latency for RUM, we use Akamai's Boomerang [2] tool and report the difference between `responseStart` and `requestStart` as captured from the Navigation API [40]. Page refreshes (providing RUM and CLAM) and pings were set to run every minute since CLAM has a 1 minute measurement limit with Chrome.

Figure 3 shows the distribution of latencies collected over a 2 hour period from each vantage point. The results show that while the 3 distributions show distinct shapes, the Server-Side RTT captured by CLAM tracks closely with either RUM or ICMP. The home broadband networks in Figure 3-**a** and -**b** show that for most percentiles the distributions are within 5 ms of each other, with Server-Side and RUM most similar. Interestingly, ICMP provides the lower bound in these two cases. In Figure 3-**c**, the distributions show more dramatic differences, with over 10ms between RUM and ICMP. For lower percentiles, Server-Side RTT aligns closely with RUM and then shifts abruptly to align with ICMP, hinting that RUM and ICMP use two different paths to the VM, while new connections from NEL uploads get balanced between them. The University Wireless network also shows that Server-Side and RUM distributions to be quite similar, with differences of a few milliseconds, while ICMP is consistently higher.

These differences are in line with previous work that examined the discrepancies in estimating RTT. Strowes showed how application-layer, TCP, and ICMP estimates for RTT can vary dramatically depending on which side of the connection is measuring [38]. With `tokyo-ping`, Pelsser et al. demonstrated how RTT can vary dramatically depending on flow identifiers used by ECMP and link aggregation groups [33]. For this reason, the `tokyo-ping` authors recommend that no multi-connection application make latency assumptions based on a single connection.

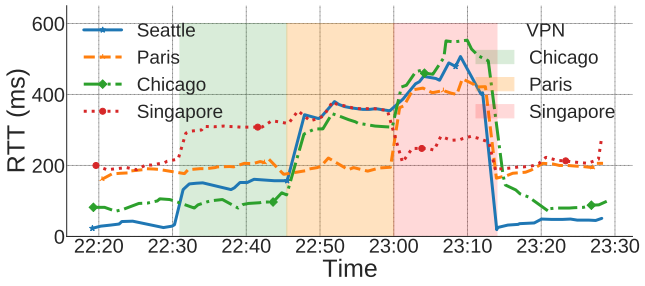

Figure 4: Performance from a single vantage point to four cloud regions with VPN gateways changed every 15 minutes.

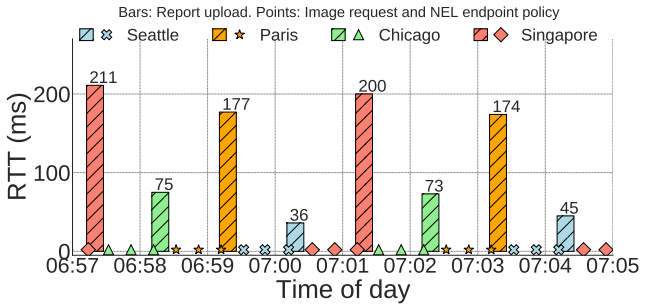

Figure 5: NEL response to policy changes over time. Report uploads are sent to the region from the most recent policy.

5.3 How effectively does CLAM track user-performance changes?

To answer this question, we designed an experiment where we change the user latency over time and show CLAM's ability to track these latency changes. We use a simple webpage with four images, all served from West US2, but each configured with a distinct NEL report endpoint in each of our four cloud regions—enabling us to collect client-side active measurements for each of these four regions every minute. We use a VPN to vary the client-server latency over time. Specifically, we switch VPN gateways for the client from Chicago, Paris, and Singapore, in order, every fifteen minutes.

Figure 4 shows the performance observed from the Pacific Northwest (PNW) home broadband network to the four cloud regions in our testbed over 70 minutes with four different VPN configurations. We extract regional RTT data directly from each region's NGINX access logs. In the first 10 minutes, we see that the lowest latency region is Seattle, followed by Chicago, Paris, and then Singapore, which is expected given our PNW vantage point. At 22:30, we enabled the VPN for the Chicago gateway and see Seattle and Singapore RTTs spike, and Chicago becomes the lowest latency region. Chicago and Paris latency remains similar since the VPN tunnel travels in the same direction as the non-VPN path. In contrast, the path to the Seattle and Singapore regions have greatly inflated RTTs since they take a circuitous route to Chicago in the East before traveling back West. We observe similar patterns with the Paris and Singapore VPN gateways. When the VPN is turned off at the end of the experiment, per-region RTTs return to similar range as the experiment start. This experiment demonstrates CLAM's ability to track changes in user performance over time.

Client	Client ISP	LDNS	Description
74.73.110.0/24 (East US)	Spectrum	25.29.108.103	Spectrum LDNS
		173.194.168.196	Google PDNS in Wash. D.C.
172.116.225.0/24 (West US)	Spectrum	66.75.177.68	TWC (Spectrum) LDNS
		172.253.0.2	Google PDNS in L.A.

Table 3: Changes in client-LDNS association tracked by CLAM.

5.4 How does CLAM enable updating measurement policy at runtime?

To answer this question, we configured the web-server to reload an image from the same domain (single NEL policy) every 20 seconds and the server-side controller responds with a different report endpoint every minute but with a fixed `max_age` of 300 seconds. Figure 5 shows the report endpoint measurement results over an 8-minute window, showing 2 full cycles for each upload region. The points at 0 along the x-axis show the report endpoint region sent from the server with each image request. We can see that within the same minute, all 3 images contain the same report endpoint. At the end of the browser's 1 minute report interval, it sends reports to the region from the *latest* NEL policy as shown by the bars and their corresponding RTTs. This demonstrates that when a policy is updated, the new one overwrites existing policies for the same domain, even if the previous policy's `max_age` has not expired.

5.5 Use Case: Capture Client-LDNS Mapping

In this section, we demonstrate how CLAM can perform client-LDNS mapping over time, which is needed by CDNs on a continous basis. We use a test page implementing the technique described in Figure 2 and two clients, one on each coast of the US. Each client loads the test page multiple times with their ISP's default DNS settings and then another round after configuring their machine to use Google's Public DNS. Table 3 shows the client-LDNS mapping extracted using CLAM. Here, note that even though both clients' ISP is Spectrum, we see that their requests came from two different LDNS servers. Since the clients are on opposite ends of the country, it make sense that they would use 2 different Spectrum LDNSes (even in different /8s). When the clients configured Google Public DNS, we see that the East US client was directed to the Washington D.C. instance while the West US client was sent to Los Angeles [5].

6 RELATED WORK

The system design of NEL was presented at NSDI 2019 [19] by the authors of the specification [16]. Their work focuses on evaluating NEL for what it was explicitly designed for: reporting passively observed performance and failures in the browser, even in the presence of network failures. Given the utility of active measurements in CDN operations [21], our work shows how NEL can be used as a protocol for active measurements with widespread user support. The technique and applications we propose are not covered by previous work.

Passive server-side measurements through web logs or packet captures are widely used by CDNs to measure performance [23, 27, 35, 36, 44] or availability [34]. Our work also uses server-side logging to capture end-user performance but instead of organic traffic, we log traffic from server programmed NEL report uploads that are designed to monitor targets in a controlled fashion.

Active layer 3 measurements from serving-infrastructure to users or LDNSes [24, 42] can suffer from low response rates [25] and fail to test end-to-end functionality that reflects user experience, including errors on other layers, such as invalid HTTPS certificates.

Web measurements from end-users can be run with JavaScript [18, 21, 22] or by simply embedding a transparent 1x1 pixel image in HTML. Akamai [1, 2] and Fastly [4] operate JavaScript measurement platforms where customers opt-in to embed a script to their application. From speaking with several CDN operators, customer adoption is challenging. Injection of third-party code raises performance, security, and privacy concerns [29, 39]. Unlike JavaScript, CLAM minimizes interference with web application execution because it runs in a background browser task. CLAM is complementary to systems like AdTag [22], which uses targeted ads to acquire end-user vantage points, by replacing JavaScript with less invasive measurements.

Applications enabling low-layer measurements such as traceroute are used by Akamai [23, 43] and Microsoft [21]. Coverage is subject to customer willingness to install desktop applications whereas CLAM works with any CDN customer end-user using a compatible browser.

7 CONCLUSION

In this work we describe CLAM, the first approach that enables CDN, cloud, and content providers to initiate client-side measurements to their serving infrastructure without client-side application control. We describe how to use CLAM to perform several measurements critical to CDN operations such as alternate PoP measurements, client-LDNS association, and availability monitoring. Our cloud-based testbed and evaluation demonstrates that CLAM is an effective measurement approach that accurately captures network latency between end-users and target servers, quickly tracks changes in network performance, and enables rapid change of client measurement policy across a large user base.

ACKNOWLEDGMENTS

We would like to thank John Rula, Tobias Bajwa, and Brandon Schlinker for feedback on early drafts. We thank Kyle Schomp and the anonymous reviewers for their feedback that improved this work.

REFERENCES

[1] Akamai mPulse. https://www.akamai.com/us/en/products/performance/mpulse-real-user-monitoring.jsp.
[2] Boomerang. https://akamai.github.io/boomerang/.
[3] Citrix Intelligent Traffic Management (Previously Cedexis. https://www.citrix.com/products/citrix-intelligent-traffic-management/.
[4] Fastly Insights. https://insights.fastlylabs.com/.
[5] Google Public DNS FAQ. https://developers.google.com/speed/public-dns/faq.
[6] HTTP Archive. https://httparchive.org/.
[7] Introducing a New whoami Tool for DNS Resolver Information. https://developer.akamai.com/blog/2018/05/10/introducing-new-whoami-tool-dns-resolver-information.
[8] Microsoft Azure Regions. https://azure.microsoft.com/en-us/global-infrastructure/regions/.
[9] NetMarketShare: Browser Market Share. https://netmarketshare.com/browser-market-share.aspx.

[10] Private Internet Access. https://www.privateinternetaccess.com/.
[11] RIPE Atlas. https://atlas.ripe.net/.
[12] statscounter: Browser Market Share Worldwide. https://gs.statcounter.com/browser-market-share.
[13] TCP INFO. https://www.measurementlab.net/tests/tcp-info/.
[14] Thousandeyes. https://www.thousandeyes.com/.
[15] U.S. Federal Goverment Digital Analytics Program. https://analytics.usa.gov/.
[16] W3C Network Error Logging. https://w3c.github.io/network-error-logging/.
[17] W3C Schools Browser Statistics. https://www.w3schools.com/browsers/default.asp.
[18] A. Ahmed, Z. Shafiq, H. Bedi, and A. Khakpour. Peering vs. transit: Performance comparison of peering and transit interconnections. In *ICNP*, 2017.
[19] S. Burnett, L. Chen, D. A. Creager, M. Efimov, I. Grigorik, B. Jones, H. V. Madhyastha, P. Papageorge, B. Rogan, C. Stahl, et al. Network Error Logging: Client-side measurement of end-to-end web service reliability. In *NSDI*, 2020.
[20] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, and J. Padhye. Analyzing the Performance of an Anycast CDN. In *IMC*, 2015.
[21] M. Calder, R. Gao, M. Schröder, R. Stewart, J. Padhye, R. Mahajan, G. Ananthanarayanan, and E. Katz-Bassett. Odin: Microsoft's Scalable Fault-Tolerant CDN Measurement System. In *NSDI*, 2018.
[22] P. Callejo, C. Kelton, N. Vallina-Rodriguez, R. Cuevas, O. Gasser, C. Kreibich, F. Wohlfart, and Á. Cuevas. Opportunities and Challenges of Ad-based Measurements from the Edge of the Network. In *HotNets*, 2017.
[23] F. Chen, R. K. Sitaraman, and M. Torres. End-user mapping: Next generation request routing for content delivery. *SIGCOMM*, 2015.
[24] W. B. de Vries, R. d. O. Schmidt, W. Hardaker, J. Heidemann, P.-T. de Boer, and A. Pras. Verfploeter: Broad and load-aware anycast mapping. In *IMC*, 2017.
[25] C. Huang, D. A. Maltz, J. Li, and A. Greenberg. Public dns system and global traffic management. In *INFOCOM*, 2011.
[26] A. Kashaf, V. Sekar, and Y. Agarwal. Analyzing third party service dependencies in modern web services: Have we learned from the mirai-dyn incident? In *Proceedings of the ACM Internet Measurement Conference*, pages 634–647, 2020.
[27] R. Krishnan, H. V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving Beyond End-to-End Path Information to Optimize CDN Performance. In *IMC*, 2009.
[28] G. Linden. Make Data Useful. http://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-28.ppt, 2006.
[29] S. Ludin. Measuring what is not ours: A tale of 3rd party performance. In *PAM*, 2017.
[30] Z. M. Mao, C. D. Cranor, F. Douglis, M. Rabinovich, O. Spatscheck, and J. Wang. A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers. In *USENIX ATC*, 2002.
[31] P. Mastin. *Real user measurements*. O'Reilly Media, Incorporated, 2016.
[32] E. Nygren, R. K. Sitaraman, and J. Sun. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44, 2010.
[33] C. Pelsser, L. Cittadini, S. Vissicchio, and R. Bush. From paris to tokyo: On the suitability of ping to measure latency. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 427–432, 2013.
[34] P. Richter, R. Padmanabhan, N. Spring, A. Berger, and D. Clark. Advancing the Art of Internet Edge Outage Detection. In *IMC*, 2018.
[35] B. Schlinker, I. Cunha, Y.-C. Chiu, S. Sundaresan, and E. Katz-Bassett. Internet performance from facebook's edge. In *IMC*, 2019.
[36] B. Schlinker, H. Kim, T. Cui, E. Katz-Bassett, H. V. Madhyastha, I. Cunha, J. Quinn, S. Hasan, P. Lapukhov, and H. Zeng. Engineering egress with edge fabric: Steering oceans of content to the world. In *SIGCOMM*, 2017.
[37] S. Stefanov. Yslow 2.0. In *CSDN SD2C*, 2008.
[38] S. D. Strowes. Passively measuring tcp round-trip times. *Communications of the ACM*, 56(10):57–64, 2013.
[39] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall. Demystifying page load performance with WProf. In *NSDI*, 2013.
[40] Z. Wang. Navigation Timing. https://www.w3.org/TR/navigation-timing/.
[41] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, et al. Taking the edge off with espresso: Scale, reliability and programmability for global internet peering. In *SIGCOMM*, 2017.
[42] Z. Zhang, M. Zhang, A. G. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian. Optimizing cost and performance in online service provider networks. In *NSDI*, 2010.
[43] M. Zhao, P. Aditya, A. Chen, Y. Lin, A. Haeberlen, P. Druschel, B. Maggs, B. Wishon, and M. Ponec. Peer-assisted content distribution in akamai netsession. In *IMC*, 2013.
[44] Y. Zhu, B. Helsley, J. Rexford, A. Siganporia, and S. Srinivasan. LatLong: Diagnosing Wide-area Latency Changes for CDNs. In *Transactions on Network and Service Management*, 2012.