Improving Fidelity in Video Streaming Experimentation on Testbeds with a CDN

Calvin Ardi
USC/ISI
calvin@isi.edu

Alefiya Hussain
USC/ISI
hussain@isi.edu

Michael Collins
USC/ISI
mcollins@isi.edu

Stephen Schwab
USC/ISI
schwab@isi.edu

ABSTRACT

Video streaming is the leading network traffic on the Internet, yet there are few tools to run high fidelity experiments with video streaming traffic on network emulation-based testbeds. In this paper, we present a framework to enable higher fidelity and principled experimentation with 36 different video streaming traffic scenario combinations that can be configured and deployed on a notional CDN and data metrics infrastructure. This framework can be used to further study and experiment with adaptive bitrate algorithms and other AI/ML solutions for video delivery.

ACM Reference Format:
Calvin Ardi, Alefiya Hussain, Michael Collins, and Stephen Schwab. 2022. Improving Fidelity in Video Streaming Experimentation on Testbeds with a CDN. In *Workshop on Design, Deployment, and Evaluation of Network-assisted Video Streaming (ViSNext '22), December 9, 2022, Roma, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3565476.3569097

1 INTRODUCTION

Video streaming is one of the top network traffic applications on the Internet [19], however, there are limited tools to model representative video streaming traffic for systematic and disciplined experimentation. Prior work in traffic generation and simulation typically focus real-world deployments [1, 2, 25] and algorithm development and stress testing [13] or on simulating the behavior of the underlying protocols [3, 5, 21, 26].

Networking and cyber security researchers need video streaming traffic generation tools in representative environments in order to develop and evaluate the next generation of systems, including those for AI/ML network traffic classification and quality of service management. These tools allow experimenters to test and evaluate their solutions during development, and include reconfigurable client and server components.

Today, experimenters iterate on and test their video streaming software and algorithm components on the Internet, simulators/emulators, or "one-off" on their development workstations. These environments presents a challenge for fidelity in experimentation. There is a trade-off between fidelity and reproducibility, and at another level, ease of reproducibility. Running or deploying services on the Internet gives you the highest fidelity, but the resource and monetary costs are prohibitively high for almost everyone except large commercial entities. Similarly, network simulators are fast and easy ways to simulate network traffic conditions, but are often focused on simulating the underlying protocols (TCP, UDP) rather than application traffic.

In order to create representative video traffic on-the-wire for experimentation on testbeds, we combine and use various components of a video streaming architecture to emulate real users watching actual videos on a modern Internet. In § 2, we present a notional DNS-based content distribution network that can be configured for the experiments. While preliminary at this stage, this model enables reproducible video streaming experiments at a higher fidelity with edge caches at multiple points-of-presence (PoPs) in the emulated network.

The benefit of experimentation on testbeds is reproducibility and repeatability of *actual* network traffic and behavior in a controlled environment. While the clients, servers, middleboxes, and network topologies are emulated on VMs or bare-metal, the traffic on the wire is *real*.

Our contributions are to: (1) build and support a Content Delivery Network (CDN) on an emulation-based networking testbed for video streaming, (2) extend existing reproducible video traffic generators to use a CDN, and (3) instrument clients and servers to support collecting Quality of Experience (QoE) metrics and build a data collection architecture to support live and offline analysis of QoE data. We describe the architecture and methodology of our video streaming experimentation framework in § 2. In § 3, we demonstrate our CDN and video streaming traffic generators. We cover related work in § 4, discuss future work and conclude in § 5. To further enable research in this domain, we make our network traffic generators, experimentation framework, and tools freely available at https://mergetb.org/projects/searchlight/.

2 A FRAMEWORK FOR VIDEO STREAMING EXPERIMENTATION

Our framework of video streaming experimentation on network emulation-based testbeds at a higher fidelity consists of three components: a simplified CDN (§ 2.1), video streaming traffic generators (§ 2.2), and instrumentation and architecture for analyzing QoE metrics (§ 2.3). We discuss implementation details in § 2.4.

ViSNext '22, December 9, 2022, Roma, Italy

ACM ISBN 978-1-4503-9936-4/22/12…$15.00
https://doi.org/10.1145/3565476.3569097

Table 1: Comparing Our Testbed vs. an Internet CDN Features

System	Our Testbed CDN	An Internet CDN
Traffic steering	Single-level DNS	Multi-level DNS, anycast routing
Content management	"Static" (already synchronized and replicated), pre-computed audio/video chunks	Dynamic (cache priming and miss strategies), complex replication strategies (resource, geographical constraints) on-the-fly audio/video encoding
Infrastructure monitoring	Throughput, QoE, QoS metrics (control plane)	Accounting, availability, network and application performance (including QoE, QoS)

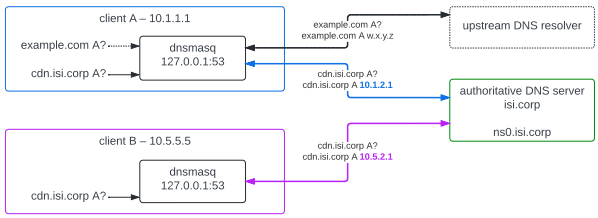

Figure 1: Client-side Split DNS Configuration

2.1 Improving Network Experimentation Fidelity with a CDN

We take a step forward in improving the experimentation fidelity of video streaming experiments by implementing and providing a usable CDN within a testbed.

Today's Internet is both much more complex and consolidated: most traffic today is being delivered to Internet users from Hypergiants [10, 14], a small set of content providers and CDNs. In the past, a client connecting to a website would be a simple point-to-point client-server connection: today, the top websites, on average, loaded roughly 96 different URLs from 16.5 different domains [18], many of which resolve to the same CDN.

Our goal is to build a working and easily extensible CDN for experimentation on testbeds. A CDN system consists of many complex subsystems [17], and because the often proprietary technology is constantly evolving, we focus our efforts on what the end-user client would "see" on the network when interacting with a CDN rather than an actual Internet CDN. Table 1 summarizes, at a high-level, some of the distinctive features between our CDN implementation and an Internet CDN.

From the perspective of the end-user, a CDN roughly consists of two major systems: (1) traffic steering using DNS or anycast routing and (2) a set of edge caches located as close as possible to the end-user client.

To steer traffic to a particular edge cache, we deploy authoritative DNS with a number of answering strategies that are available to experimenters. Figure 1 shows a typical client configuration: a client has a local forwarding resolver that forwards all requests in a particular domain to an authoritative Domain Name System (DNS) server (all other domains are sent to an upstream resolver). In response to a client's query, this authoritative can be configured to return a record that matches the closest edge cache server using IP geolocation, or "random" records. While returning the closest server is typically the optimal behavior, our ability to return arbitrary records is useful for validating testbed experiments and experimenting with network capabilities. For example, the experimenter or the testbed platform can validate or spot-check that an experiment is working and performing as specified (correctness in DNS, network reachability, etc.) before running experiments and taking measurements. Similarly, we configure how records are returned to experiment with fault tolerance and network service failures (Internet outages, cyberattacks, overloading) or capabilities (load balancing, measuring availability and performance).

In addition to arbitrary records, we can configure wildcard DNS to always return random and valid answers. Using wildcard DNS enables us to control both the effects of DNS caching and number of lookups required.

We focus on content delivery from an edge to the end-user client, resulting in a "static"-like CDN. A typical CDN on the Internet generally pulls or gets pushed content from one or more origin servers. We instead deploy a set of already replicated edge caches: from the end-user's perspective, this represents the typical browsing or viewing experience as the initial added latency only affects the first user. Prior work [12] and discussions with researchers at CDN providers has found that the latency cost in the initial cache miss is highly optimized (via priming or network pathing) or greatly amortized over time in production CDNs. To simplify our initial implementation, we consider replication strategies and evaluating the costs of cache priming and misses for future work.

Figure 2 shows our edge server implementation. On each edge server, assets are served by a Caddy webserver from an already-synchronized and consistent backing store (i.e., the origin server (dotted boxes) has already pushed content to all caches (dotted lines)). We also provide a live watermarking image proxy to help with caching (or cache busting) and visually validating that the CDN is working as intended.

2.2 Extending and Using Reproducible Video Streaming Traffic Generators

We next extend a video streaming traffic generator to use our CDN and instrument the generators with QoE metrics collection.

Video streaming traffic is defined as video that is streamed on-demand from a server to client (Netflix, YouTube, or Vimeo) as opposed to video streamed live (Facebook Live, Twitch). In a classic video streaming deployment, one or more clients connect to and stream a video from a remote server, which hosts both the audio/video assets and supporting website (HTML, JavaScript). To improve realism, we add support for hosting both audio/video and

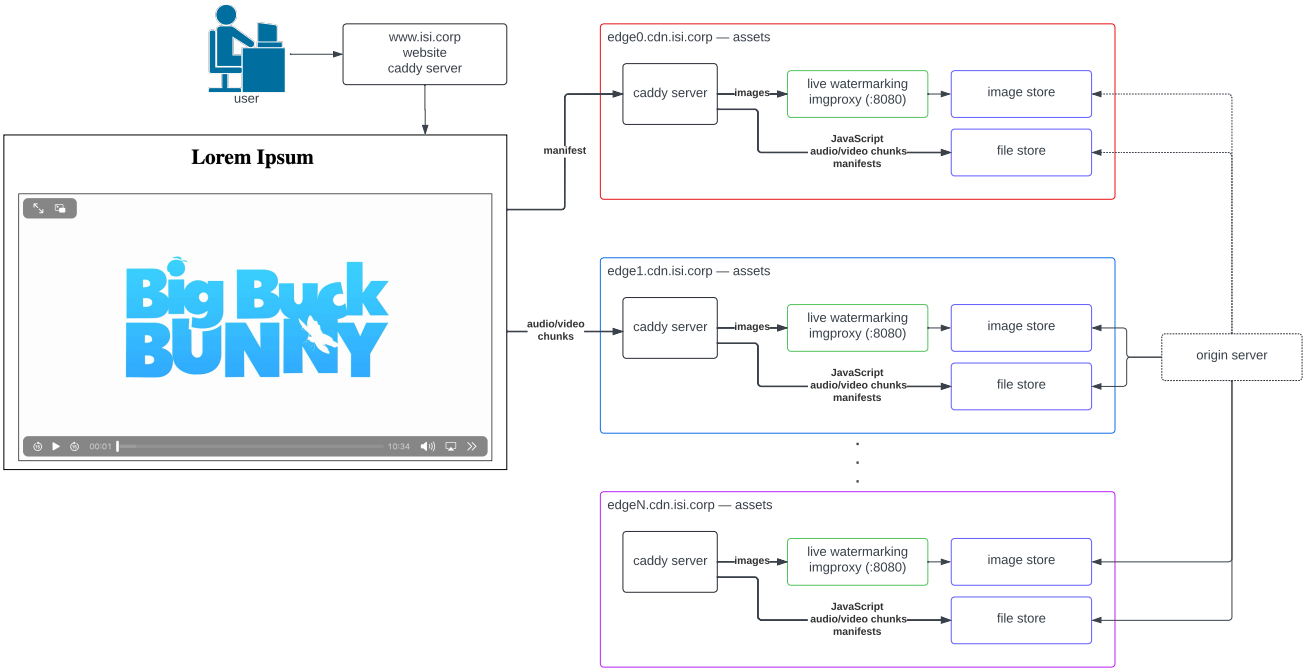

Figure 2: Edge Server Implementation and Client Interaction Example

Table 2: Video Streaming Configurable Settings [4]

Name	Configuration Values
Resolution	576p, 720p, 1080p
Protocol:	
Streaming	DASH, HLS, HTML5
Transport	HTTP/1.1, HTTP/1.1+TLS, HTTP/2, HTTP/3
Watching:	
Behavior	Simple, Random (*future*), Popular (*future*)
Time	*n* seconds

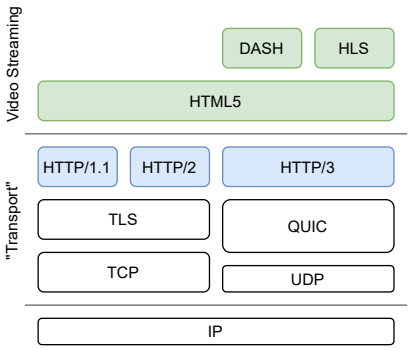

Figure 3: Video Streaming and "Transport" Protocol Stack

website assets on our CDN. To improve experimentation and evaluation, we add instrumentation for collecting QoE ground-truth metrics for both live and offline analysis.

We extend and use an open-source video streaming traffic generator [4], which supports 36 combinations of multiple video streaming (DASH, HLS, HTML5) and transport protocols (HTTP/1.1, HTTP/2, HTTP/3) over a variety of resolutions, as shown in Table 2 (Figure 3 shows the protocol stack and how protocols build on top of another). The client consists of the Google Chrome web browser instrumented using JavaScript and the Playwright [16] browser automation framework to emulate a user watching video. The client watches videos using a simple human behavior model on a website for a pre-determined length of time. To easily enforce a configuration for ground truth, the server can strictly deliver websites at specific URL endpoints that indicate the streaming and transport protocols used.

We extend this video streaming traffic generator to incorporate our CDN and QoE components. To use our CDN, we dynamically generate the video website by rewriting URLs of page assets (video manifests of audio/video chunks, JavaScript, images) to use domains that resolve to an edge cache. We discussed earlier that we can configure how the domain names of our CDN edges resolve to its IP addresses via IP geolocation or round-robin. Depending on the experimenter's needs, clients can connect to URLs that resolve to the closest edge cache (`cdn.example.corp`), or to an arbitrary edge that is likely to be distant "geographically" (e.g.,

`qkf7kjj.cdn.example.corp` → `nyc.cdn.example.corp`). In an extreme case used to demonstrate our CDN's functionality, for example, each asset's URL points to a unique domain which results in a client connecting to every edge cache in our CDN.

We also instrument the client and server to collect QoE metrics for analyzing the overall performance and health of video streaming delivery and playback. A defined set of QoE metrics are standardized in the industry as Common Media Client Data (CMCD) [6]. CMCD includes video streaming performance data like measured bandwidth, buffer starvation events, and requested/actual bitrates of audio/visual segments. On the client-side, we use existing instrumentation on the ABR video players, which send CMCD data as URL query parameters to the web server. We further configure the web server to send HTTP access log entries to a lightweight logger we developed that sits alongside Caddy. This logger parses and sends video metrics to multiple "sinks" over the control plane, including a remote time-series database, for further analysis.

2.3 Experiment Instrumentation and Architecture for Live and Offline Analysis of QoE Metrics

The final major component in our framework for video streaming experimentation is the architecture for collecting and processing QoE metrics.

Analyzing QoE metrics is important in experimentation as the metrics reveal to us the actual behavior and ground truth of network and video playback performance as the client end-user perceives it. We do not always receive the full picture of client performance by looking at data on the wire or packet captures—this is especially true as network traffic and corresponding metadata moves to being encrypted by default. Similarly, spot checking all video streaming clients using VNC or remote desktop is both labor-intensive and infeasible on large-scale experimentation. (While prior work has shown that one can use AI/ML techniques [8] to build models on network traffic data to infer playback performance, training these models requires labor-intensive and often manual labeling of ground truth.)

We support the instrumentation of ground truth or QoE data in our video streaming traffic generators (discussed earlier in § 2.2) by deploying monitoring backend infrastructure in our experiments. We provide live visualization or situational awareness using dashboards and enable offline analysis by archiving data after each experiment run.

QoE metrics are sent from our video traffic generators to a remote monitoring server running the InfluxDB Time-Series Database (TSDB). To avoid adding unnecessary noise to an experiment's network traffic, we typically send logging and metrics data over the control plane (akin to a management VLAN). To emulate a production environment, we could also run all monitoring traffic over the data plane as well.

Live and offline analysis of QoE metrics helps with iterating and validating an experiment run, and enables comparisons between many experiments. Support for live and offline analysis is done by connecting various frontends to the TSDB backend. For offline analysis, we export and archive the TSDB and Hypertext Transfer Protocol (HTTP) access log data for further processing.

To visually validate a live experiment or demonstration, we can monitor both network throughput and QoE metrics in real-time using web dashboards (Grafana), shown in Figure 4. For example, if we reduce the bandwidth on a link between a client and server during an experiment, we can visually verify that the measured network throughput has decreased and that the video player on the client reduces the playback bitrate and possibly experiences buffer starvation.

2.4 Implementation and Usage

We implement each component in our framework with the goal of maximizing experimentation flexibility and configurability. While each of our components are designed to interoperate together, they can also be used independently, depending on experimentation needs (e.g., our server-side components could be used with an independently developed mobile client).

We use and support our implementations on Merge [7], a network emulation-based Linux testbed. The Merge testbed platform enables users to deploy experiments with arbitrary network topologies and dynamically configure routing and links with arbitrary bandwidth capacity (up to 10 Gbps), delay, and loss. Merge also provides efficient hardware resource allocation and sharing and isolation between experiments.

All experiment nodes on the testbed are VMs that run an unmodified, baseline Ubuntu 20.04 LTS operating system. The hardware requirements will vary depending on experimental needs: we have found that 2 to 4 CPU cores on a modern CPU with 8 GB of memory on each node provides smooth video streaming performance while supporting data instrumentation and collection. We have also run video streaming experiments successfully on a Merge testbed of resource-constrained, embedded devices [11].

We create and deploy Ansible playbooks for all stages of an experiment (topology configuration, software provisioning, traffic generation, data capture and archiving, topology teardown), including repeated runs for statistical validity. These playbooks make the experimental setup and execution portable to other environments without requiring custom operating system images.

Our network traffic generators, experimentation framework, and tools are freely and publicly available at https://mergetb.org/projects/searchlight/.

3 USE CASE DEMONSTRATION

We next look at a video streaming experiment running on a testbed to demonstrate our video streaming traffic generators using our CDN on a testbed.

We deploy a small-scale tiered network to provide a controlled environment that roughly approximates an enterprise network while balancing the topology size with resource constraints. The tiered network consists of a tree-like hierarchical topology with traffic generating clients at the leaves (enclaves) and multiple levels of routers above. Figure 5 shows our 19 node topology, with clients (`h*`, orange) at the furthest leaves. Our CDN has edge caches (`cdn-*`, blue) deployed at the edge of each enclave, connected to a border router (magenta). Finally, all border routers (and corresponding enclaves) are connected in a tree-like hierarchical structure rooted at a common router `c0` (blue).

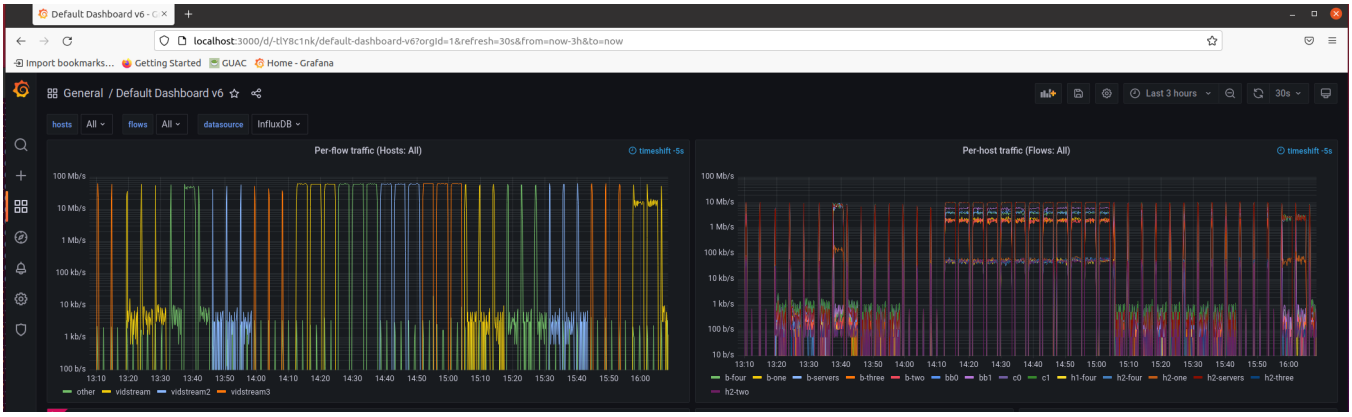

Figure 4: Live Metrics Visualization Dashboard

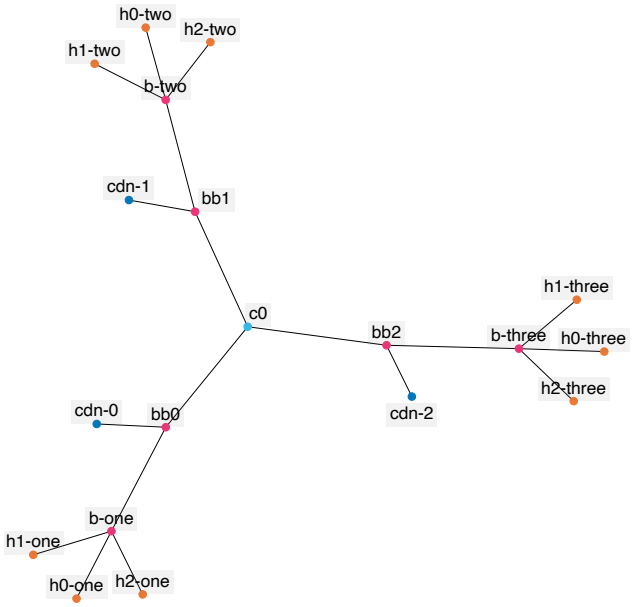

Figure 5: Tiered Topology with a CDN

Table 3: Link Capacities in Tiered Topology

Link(s)	Capacity (Mbps)	RTT (ms)
All end-hosts to their border router	500	0
{b-one, cdn-0} ↔ bb0	500	0
{b-two, cdn-1} ↔ bb1	500	0
{b-three, cdn-2} ↔ bb2	500	0
bb0 ↔ c0	500	20
bb1 ↔ c0	100	200
bb2 ↔ c0	50	600

We deploy the network with "geographically distant" enclaves, representing satellite or remote offices, by adjusting the bandwidth capacity and latency of each link. We deploy the network such that the enclaves are "geographically distant" by adjusting the bandwidth capacity and latency of each link. Table 3 summarizes the bandwidth of each link. Clients and the edge cache in each enclave are connected to one another with high bandwidth and very low latency. In this demonstration, each enclave is connected to the core at varying bandwidths and latencies: while enclave `one` is connected at a high speed (500 Mbps, 20 ms), enclaves `two` and `three` are much slower: 100 Mbps/200 ms and 50 Mbps/600 ms, respectively. While Merge allows experimenters to dynamically and programmatically adjust link constraints (bandwidth, delay, loss) while an experiment is running, we initially keep them constant as controlled variables.

To demonstrate our CDN, we run a video streaming traffic client that accesses video on all three edge caches. While all the edge caches are replicated in the same way, we modify the video manifest file to retrieve assets at specific resolutions from specific caches. The client, running on `h0-one` connects to the website and plays the video at each resolution (576p, 720p, 1080p) for approximately 30 s each (total runtime 120 s) over DASH and HTTP/2.

Figure 6 shows the connections and throughput over time as captured on border router `bb0`. Each connection to a server is color-coded and labeled for reference. We see that the initial spike (purple) in the loading of the webpage, JavaScript, and video manifest from `cdn-0`. The client then starts playback at 576p from `cdn-2` (orange), 720p from `cdn-1` (blue), and finally 1080p from `cdn-0` (purple). Overall, we see the expected "sawtooth"-like pattern of ABR video streaming. We see a noticeable spike around 80 s when switching to 1080p, as the video player takes advantage of the massive increase in bandwidth (from 100 Mbps to 500 Mbps) to refill its "initial" buffer.

We have highlighted some of the capabilities of running video streaming traffic experiments with a CDN on a testbed. We believe that experimenters will be able to emulate a variety of network conditions and scenarios with CDNs to design and test new and existing ABR algorithms.

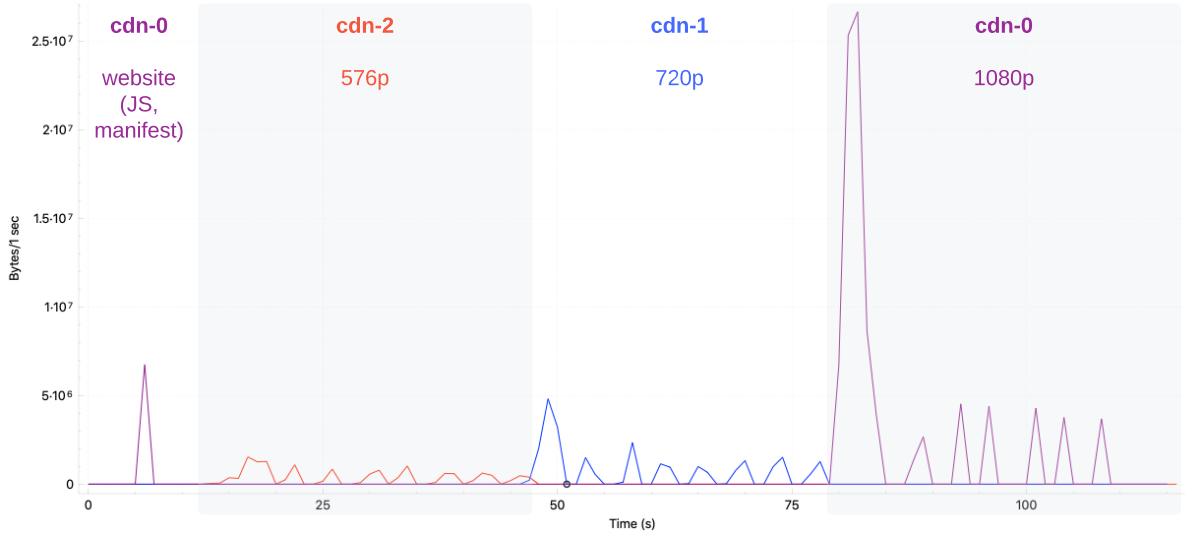

Figure 6: Streaming video at multiple resolutions from three edge cache servers to one client via HTTP/2.

4 RELATED WORK

Prior work in traffic generation has focused on simulating the behavior of the underlying protocols [3, 21, 26]. Stohr et al. [22] conduct a systematic approach using the Mininet network emulator on DASH video players to find optimum configurations and to directly compare players and algorithms. Our approach expands the scope and fidelity of video streaming traffic in experimentation by deploying the traffic in a variety of network conditions and topologies with other competing network flows.

Prior work in CDNs typically focus on the architecture or measurements of CDNs deployment on the Internet [1, 9, 23]. Schomp et al. [20] describe the architecture of Akamai's DNS system, one of the largest DNS systems that supports their CDN. While our notional CDN uses a simplified DNS setup, we can learn from their design decisions to improve upon our own in the context of improving fidelity for experimentation.

CDNs have also been deployed at varying levels of fidelity on testbeds [15, 24]. CoDeeN [25] was a CDN deployed on the PlanetLab testbed, and acted as web proxies for PlanetLab clients to cache and distribute content from origin sites. While CoDeeN and PlanetLab provided high levels of fidelity by virtue of being deployed on the Internet, it would have been difficult to reliably reproduce and repeat experiments using CoDeeN. Our CDN and traffic generators provides reproducible, high fidelity experimentation in a controlled environment for ease in repeatability.

5 FUTURE WORK AND CONCLUSION

In future work we will focus on improving the fidelity of our CDN, and developing data collection techniques to further improve analysis. These modifications will also allow us to implement more extensive experiments and different categories of experiments.

The DNS-based redirection implemented in our work is the most fundamental CDN technique, and also the easiest to control. CDN providers have developed many other techniques to optimize delivery, including anycast routing and Edge Side Includes (ESI). In order to accurately emulate more recent CDN architectures, we must implement such technologies. These approaches require more complex coordination—anycast routing requires manipulating Border Gateway Protocol (BGP), and ESI involves configuring specific web pages. Implementing these techniques not only improves our ability to emulate CDNs, but also to consider how different combinations of the CDN toolbox can impact systems.

Our current instrumentation is built on existing network-based data collection tools; however, this approach is inadequate for the complex multi-host and multi-network phenomena which make up modern web traffic. In order to properly reconstruct and analyze events, we need a capability which identifies all of the distinct observables of a single event at the endpoint and then correlate that information with data from all the diverse servers and networks comprising the experiment. Accurate modern network experimentation requires examining large networks and the concomitant large volume of data, some form of summarization similar to NetFlow is necessary to reduce the data footprint, reduce reliance on raw data and increase analysis speed.

In this paper, we have taken the first steps towards increasing fidelity in video streaming experimentation on testbeds. We implemented the core components of a CDN, extended a video streaming traffic generator to use the CDN, and instrumented both the traffic generator and testbed to collect and analyze QoE metrics during and after experiment runs. Our network traffic generators, experimentation framework, and tools will be freely available at https://mergetb.org/projects/searchlight/.

REFERENCES

[1] Giancarlo Fortino and Carlos E. Palau (Eds.). 2012. *Next Generation Content Delivery Infrastructures: Emerging Paradigms and Technologies*. IGI Global. https://doi.org/10.4018/978-1-4666-1794-0

[2] Pablo Ameigeiras, Juan J. Ramos-Munoz, Jorge Navarro-Ortiz, and J.M. Lopez-Soler. 2012. Analysis and modelling of YouTube traffic. *Transactions on Emerging Telecommunications Technologies* 23, 4 (2012), 360–377. https://doi.org/10.1002/ett.2546 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.2546

[3] Doreid Ammar, Thomas Begin, and Isabelle Guerin-Lassous. 2011. A New Tool for Generating Realistic Internet Traffic in NS-3. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques* (Barcelona, Spain) *(SIMUTools '11)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, 81–83.

[4] Calvin Ardi, Alefiya Hussain, and Stephen Schwab. 2021. Building Reproducible Video Streaming Traffic Generators. In *Cyber Security Experimentation and Test Workshop* (Virtual, CA, USA) *(CSET '21)*. Association for Computing Machinery, New York, NY, USA, 91–95. https://doi.org/10.1145/3474718.3474721

[5] Sachin Ashok, Sai Surya Duvvuri, Nagarajan Natarajan, Venkata N. Padmanabhan, Sundararajan Sellamanickam, and Johannes Gehrke. 2020. iBox: Internet in a Box. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks* (Virtual Event, USA) *(HotNets '20)*. Association for Computing Machinery, New York, NY, USA, 23–29. https://doi.org/10.1145/3422604.3425935

[6] Consumer Technology Association. 2020. Web Application Video Ecosystem - Common Media Client Data. CTA-5004. https://cdn.cta.tech/cta/media/media/resources/standards/pdfs/cta-5004-final.pdf

[7] MergeTB Authors. 2022. *The Merge Testbed Platform*. https://mergetb.org

[8] Francesco Bronzino, Paul Schmitt, Sara Ayoubi, Guilherme Martins, Renata Teixeira, and Nick Feamster. 2019. Inferring Streaming Video Quality from Encrypted Traffic: Practical Models and Deployment Experience. *Proc. ACM Meas. Anal. Comput. Syst.* 3, 3, Article 56 (Dec. 2019), 25 pages. https://doi.org/10.1145/3366704

[9] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. 2015. Analyzing the Performance of an Anycast CDN. In *Proceedings of the 2015 Internet Measurement Conference* (Tokyo, Japan) *(IMC '15)*. Association for Computing Machinery, New York, NY, USA, 531–537. https://doi.org/10.1145/2815675.2815717

[10] Petros Gigis, Matt Calder, Lefteris Manassakis, George Nomikos, Vasileios Kotronis, Xenofontas Dimitropoulos, Ethan Katz-Bassett, and Georgios Smaragdakis. 2021. Seven Years in the Life of Hypergiants' off-Nets. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 516–533. https://doi.org/10.1145/3452296.3472928

[11] Ryan Goodfellow, Stephen Schwab, Erik Kline, Lincoln Thurlow, and Geoff Lawler. 2019. The DComp Testbed. In *12th USENIX Workshop on Cyber Security Experimentation and Test (CSET 19)*. USENIX Association, Santa Clara, CA. https://www.usenix.org/conference/cset19/presentation/goodfellow

[12] Thomas Koch, Ke Li, Calvin Ardi, Ethan Katz-Bassett, Matt Calder, and John Heidemann. 2021. Anycast In Context: A Tale of Two Systems. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Virtual Event, USA) *(SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 398–417. https://doi.org/10.1145/3452296.3472891

[13] ESnet / Lawrence Berkeley National Laboratory. 2020. iperf3 v3.9. https://software.es.net/iperf/

[14] Craig Labovitz, Scott Iekel-Johnson, Danny McPherson, Jon Oberheide, and Farnam Jahanian. 2010. Internet Inter-Domain Traffic. *SIGCOMM Comput. Commun. Rev.* 40, 4 (Aug. 2010), 75–86. https://doi.org/10.1145/1851275.1851194

[15] Ge Ma and Zhen Chen. 2014. Comparative study on CCN and CDN. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 169–170. https://doi.org/10.1109/INFCOMW.2014.6849209

[16] Microsoft. 2021. Playwright v1.10.0. https://playwright.dev/

[17] Al-Mukaddim Khan Pathan and Rajkumar Buyya. 2007. A Taxonomy and Survey of Content Delivery Networks. Technical Report, GRIDS-TR-2007-4, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia.

[18] Simran Patil and Nikita Borisov. 2019. What Can You Learn from an IP?. In *Proceedings of the Applied Networking Research Workshop* (Montreal, Quebec, Canada) *(ANRW '19)*. Association for Computing Machinery, New York, NY, USA, 45–51. https://doi.org/10.1145/3340301.3341133

[19] Sandvine. 2022. 2022 Global Internet Phenomena Report. (20 Jan. 2022). https://www.sandvine.com/phenomena

[20] Kyle Schomp, Onkar Bhardwaj, Eymen Kurdoglu, Mashooq Muhaimen, and Ramesh K. Sitaraman. 2020. Akamai DNS: Providing Authoritative Answers to the World's Queries. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 465–478. https://doi.org/10.1145/3387514.3405881

[21] Joel Sommers, Hyungsuk Kim, and Paul Barford. 2004. Harpoon: A Flow-Level Traffic Generator for Router and Network Tests. *SIGMETRICS Perform. Eval. Rev.* 32, 1 (June 2004), 392. https://doi.org/10.1145/1012888.1005733

[22] Denny Stohr, Alexander Frömmgen, Amr Rizk, Michael Zink, Ralf Steinmetz, and Wolfgang Effelsberg. 2017. Where Are the Sweet Spots? A Systematic Approach to Reproducible DASH Player Comparisons. In *Proceedings of the 25th ACM International Conference on Multimedia* (Mountain View, California, USA) *(MM '17)*. Association for Computing Machinery, New York, NY, USA, 1113–1121. https://doi.org/10.1145/3123266.3123426

[23] Ao-Jan Su, David R. Choffnes, Aleksandar Kuzmanovic, and FabiÁn E. Bustamante. 2009. Drafting Behind Akamai: Inferring Network Conditions Based on CDN Redirections. *IEEE/ACM Transactions on Networking* 17, 6 (2009), 1752–1765. https://doi.org/10.1109/TNET.2009.2022157

[24] Miran Taha. 2016. A Novel CDN Testbed for Fast Deploying HTTP Adaptive Video Streaming. In *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications* (Xi'an, China) *(MobiMedia '16)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, 65–71. https://doi.org/10.4108/eai.18-6-2016.2264163

[25] Limin Wang, KyoungSoo Park, and Ruoming Pang. 2004. Reliability and Security in the CoDeeN Content Distribution Network. In *2004 USENIX Annual Technical Conference (USENIX ATC 04)* (Boston, MA). USENIX Association. https://www.usenix.org/conference/2004-usenix-annual-technical-conference/reliability-and-security-codeen-content

[26] Michele C. Weigle, Prashanth Adurthi, Félix Hernández-Campos, Kevin Jeffay, and F. Donelson Smith. 2006. Tmix: A Tool for Generating Realistic TCP Application Workloads in Ns-2. *SIGCOMM Comput. Commun. Rev.* 36, 3 (July 2006), 65–76. https://doi.org/10.1145/1140086.1140094