# Far Cry: Will CDNs Hear NDN's Call?

Chavoosh Ghasemi
University of Arizona
chghasemi@cs.arizona.edu

Hamed Yousefi
Aryaka Networks
hamed.yousefi@aryaka.com

Beichuan Zhang
University of Arizona
bzhang@cs.arizona.edu

## ABSTRACT

Content Delivery Networks (CDNs) have become indispensable to Internet content distribution. As they evolve to meet the ever-increasing demands, they are also facing challenges such as system complexity, resource footprint, and content security. In this paper, we look at CDNs once again, but this time from the eyes of a young networking technology called *named-data networking* (NDN). NDN supports content distribution without requiring an overlay service to bridge the gap between network services and application needs. Therefore, it can realize content distribution at large scale with an arguably simpler system design.

We conducted real-world experiments to compare the standard deployment of NDN (i.e., the global NDN testbed) and two leading CDNs (Akamai and Fastly) in terms of caching and retrieving static contents through streaming videos from four different continents over these networks for two weeks. We found that although NDN can provide a satisfactory quality of service in most cases, it falls behind CDNs mainly due to its lack of hardware infrastructure and software/protocol immaturity. Nevertheless, NDN outperforms CDNs in terms of server workload and failure resiliency due to its ubiquitous in-network caching and adaptive forwarding plane. Besides, NDN comes with built-in content security, but it needs an efficient solution for content privacy. NDN's architectural advantages make it a natural fit for Internet content distribution in the long run. That said, in terms of forthcoming goals, this paper reveals several limitations of the current NDN deployment and discusses why the future of NDN hinges on addressing those limitations.

## CCS CONCEPTS

• **Networks** → **Network performance evaluation**; **Network architectures**; **Network protocols**; **Network structure**.

## KEYWORDS

Content Delivery Networks, Named Data Networking, Performance Evaluation, Information-Centric Networks

## 1 INTRODUCTION

Radical changes in Internet usage during the past few decades have created a big semantic gap between what the network provides (i.e., transport service) and what applications want (i.e., contents). The current Internet is an enormous *transport network* moving bits from one endpoint to another and requires applications to provide the addresses of the endpoints. Such a network has no sense of what content has been requested; thus, a machine might relay the traffic for the same files over and over without knowing how much burden it could remove from the network by simply caching the files. The urgent need for a large-scale content distribution solution has resulted in the emergence of *content networks*. A content network understands what content has been solicited and tries to retrieve it from anywhere in the network, whether a cache or the origin (a.k.a. content producer). In a content network, some/all of the machines can cache contents to serve future requests and each request is forwarded based on the content name rather than the destination's IP address. Such a network is a natural fit for today's Internet as modern applications/services mostly ask for certain contents instead of accessing particular machines. To realize content networks within TCP/IP architecture with minimal changes to the current Internet, *content delivery networks (CDNs)* — massive distributed caching overlay networks — have been deployed. CDNs have been a huge commercial success and indispensable to Internet content distribution. However, as the scale increases, CDNs are also confronting challenges such as system complexity, resource footprint, and content security.

The emerging Information-Centric Networking (ICN) [10] technology, NDN (Named Data Networking) [48] in particular, offers a network architecture that directly supports content distribution without requiring a CDN overlay to bridge the gap between network services and application needs (Sec. 3.1). Thus, ICN/NDN might be able to provide large-scale content distribution with less system complexity and resource footprint [21]. However, to the best of our knowledge, NDN-based content distribution has not been evaluated in real-world scenarios at large scale, yet (Sec. 2).

In this paper, for the first time, we put the current NDN deployment [7] under a real-world measurement microscope and compare it to Akamai and Fastly, two well-known content distribution solutions on the market. Our intention, however, is not to compare CDNs with NDN in all aspects or draw conclusions across the board as modern CDNs are profoundly complex systems with varieties of features while NDN is still research in nature. Instead, we focus on the most basic feature of CDNs: *caching and retrieval of static contents*. More specifically, using an adaptive video streaming application[1] [4, 6, 22], we streamed a large number of videos over each network from four different continents for two weeks. We captured Terabytes of traffic on the end-users and the origin (i.e., our video server) from/to each network to learn how these content networks operate internally (Sec. 3.2). We then compared the performance of these networks in terms of quality of experience (QoE), origin workload, failure resiliency, and content security.

*ICN '20, September 29-October 1, 2020, Virtual Event, Canada*
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8040-9/20/09…$15.00
https://doi.org/10.1145/3405656.3418708

[1]According to the Cisco VNI report [5], video traffic will account for 82% of total Internet traffic by 2022.

This paper provides an assessment of where the standard deployment of NDN is standing today and points to future directions. **(1)** From end-user's point of view, NDN provides a *satisfactory* QoE in terms of video resolution, startup delay, and re-buffering; however, compared to CDNs, it clearly falls behind. Two major contributing factors are inadequate resource provision (including deployment footprint) and immaturity of employed software/protocol in NDN (Sec. 3.3). **(2)** NDN outperforms CDNs in terms of origin load by 69% and failure resiliency by 86%, mainly due to its stateful and adaptive forwarding plane that can utilize in-network caching in a more scalable and effective way (Sec. 3.4 & 3.5). **(3)** NDN has content security built in; nevertheless, it needs a cache-friendly and efficient content privacy solution (Sec. 3.6). **(4)** NDN's architectural advantages make it a natural fit for content distribution in the long run with notable potential to reduce financial costs of CDNs (Sec. 4). **(5)** Last but not least, in terms of near-term goals, we discuss three important limitations of the current NDN deployment and share our ideas/suggestions from the real-world experiments for future improvements and research in the field (Sec. 5).

## 2 BACKGROUND AND RELATED WORK

### 2.1 Content Delivery Networks

Although the details of CDN design and implementation can be vastly different, they all have three common, essential building blocks. (1) *Mapping system*: To direct a consumer's requests to CDNs, a mechanism is required to map each consumer to an edge server (i.e., surrogate) as the entry point to the overlay network [12, 38]. Here, the most straightforward, yet widely deployed approach is to utilize DNS-based solutions, (2) *Caching contents*: To minimize the workload on origins, increase network resiliency, and provide the consumers with a better QoE, CDN servers can temporally cache the contents to serve future requests [41, 43]. The decision on which servers to enable caching capability is a network planning issue that includes network topology, traffic characteristics, policies, etc. For instance, in Akamai, the majority of contents are cached/stored in parent servers [37], and (3) *Transport system*: Upon a cache miss on a surrogate, a forwarding mechanism is required to retrieve the solicited content from somewhere in the network. In many CDNs, when a surrogate receives a request for which it does not have the desired content in its cache, it forwards the request to a nearby Point of Presence (PoP). Within a PoP, a centralized approach is employed to determine if any of the PoP servers contains the content. If so, the request will be forwarded to that server; otherwise, the content will be fetched directly from the origin through the CDN's transport network, which typically includes different transport protocols, policies, technologies, resources, etc.

### 2.2 Related Work

This section presents the literature that reveals issues of existing CDNs and then looks at the studies that attempted to address CDN challenges by employing the ICN technology.

***CDN***: Several papers in the literature studied CDNs from different angles, including, but not limited to QoE [35, 42], scalability [17], resiliency [27, 45], and security [14, 23, 31]. Plagemann et al. [40] focused on the significant gap between the existing CDNs and a full-fledged content network and tried to motivate the necessity of removing this gap. Rossi et al. [42] discussed why the centralization in CDNs negatively impacts the volume of controlling messages/reports in the network and proposed a solution to decrease this traffic. Motivated by incidents such as the flash crowd event at Barack Obama's 2009 inauguration, Srinivasan et al. [45] studied CDNs scalability and resiliency issues. Liang et al. [31] showed how the conflicts between the end-to-end nature of HTTPS and the man-in-the-middle nature of CDNs causes serious widespread issues in CDNs, such as private key sharing. In another study, Gilad et al. [23] discussed the high financial and resource cost of adopted solutions by existing CDNs to mitigate denial of service attacks.

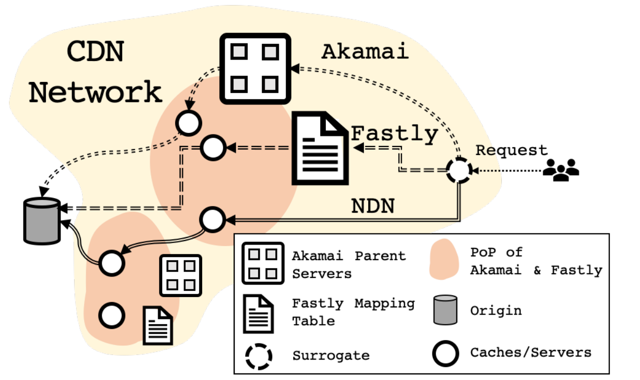

**Figure 1: Overview of the network architecture of Akamai, Fastly, and the NDN testbed.**

***ICN***: Very few papers in the ICN literature focused on improving CDNs using ICN advantages. nCDN [28] highlighted the benefits of employing NDN as CDNs' underlay network and the architectural changes that come with it. R-iCDN [33] observed the problem of sub-optimal paths in ISP-operated CDNs and addressed it by employing a centralized ICN's name-based routing solution. Inaba et al. [25] employed ICN technology in cooperation with CDNs to enable consumers to cache contents and serve the future requests for the same contents. In another study, Ma et al. [34] tried to draw a realistic picture of how NDN improves CDN in terms of quality of service and network security.

Although these works show the pros and cons of employing ICN in/as a CDN, they are either evaluated in simulations or small-scale demonstrations. To the best of our knowledge, this paper is the first that compares ICN/NDN with existing CDNs at the Internet-scale using real-world settings.

## 3 NDN VS CDN

Before conducting a comparative study of CDN and NDN, we need to clarify the scope and the focus of such a study. Today's CDNs represent a wide-spectrum of network types offering varieties of services, e.g., enterprise CDNs, video streaming CDNs, general-purpose CDNs, etc. They also have many different features, including caching static contents, generating dynamic contents, defending against distributed denial of service (DDoS) attacks, providing user statistics and analytics, etc. NDN, on the other hand, comes with a few prototype software and research-level systems with no optimization or specialization for particular environments. It also lacks many features that CDNs offer. Therefore, we cannot compare them in all aspects as if we are comparing two end products. Instead, we limit our comparison to the most basic feature of content networks: *caching and retrieval of static contents.*

## 3.1 Network Architecture and Design

Content networks employ different solutions/mechanisms to enable their servers to find contents in the network upon receiving users' requests. CDNs and NDN, however, take different approaches to enable this functionality due to their different network architectures.

The forwarding plane (of servers) in CDNs cannot find contents in the network on its own; thus, CDNs employ a centralized module to keep track of each content and where it is stored in the network. The servers then need to query this module upon receiving a request to find the solicited content in the network and forward the request, accordingly. Fig. 1 shows our understanding of Akamai and Fastly network architectures. This centralized module in Akamai is called *parent servers layer* that includes a group of servers for caching/storing contents. Upon receiving a request in this layer, it returns the address of the parent server that has the desired content. In Fastly, this module is the *mapping table*, which is indexed by the hash of contents' names and indicates which server(s) might have the solicited content. In both networks, if no potential cache is found, the content is fetched directly from the origin.

This centralized design enables a fine-grained control and optimization; however, it raises major scalability challenges. To support increasing contents and users, CDNs like Akamai and Fastly partition their networks into multiple *islands.* An island is a small regional content network that includes a Point of Presence (PoP) and the surrogates connected to it. The islands have almost the same design, infrastructure, and technology, and they work in an isolated fashion. Therefore, if a given content is not available in one island, it will be fetched directly from the origin even though a nearby island might have the content.

NDN, on the other hand, supports in-network caching and request forwarding natively at the core of its novel forwarding plane. This forwarding plane enables a couple of unique features. (1) ***Adaptive packet forwarding***: Every NDN node observes content retrieval performance and learns about network changes (both topology and cache updates) over time. For example, an NDN node can send out copies of a request to different neighbors simultaneously and learn from which neighbor the content returns fastest; it can repeat this process multiple times during content retrieval. This way, it captures network changes and updates its knowledge about the content availability in the network to tune its forwarding decisions for future requests seeking the same content [20]. Such a forwarding plane enables the nodes to find contents in the network on their own, without relying on a centralized management. (2) ***Universal hop-by-hop request forwarding***: The requests and responses between consumers and content holders (including caches and origins) are forwarded entirely through NDN nodes. As evident from Fig. 1, NDN sees the network as a whole, without partitioning it. In NDN, contents can be cached by any server while they travel in the network, and the network can answer future requests directly from the caches. In contrast, upon a cache miss in CDNs, the requests take the path from the island to the origin over a transport

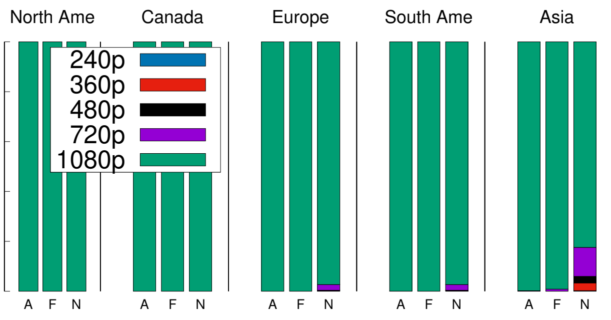

**Figure 2: Percentage of each video resolution that the end-users in five different locations experienced during the two-week experiment via Akamaia (A), Fastly (F), and the NDN testbed (N).**

network. Since there is no content network between the island and the origin, the requests have no chance to hit an in-network cache, and the resolved content will not be cached anywhere on the path from the origin to the island either.

In theory, using NDN to build a CDN can potentially reduce the overall system complexity because it does not need heavy machinery that CDN employs to keep track of contents, monitor network status, and forward requests. Less complexity will lead to less resource provisioning and eventually less operational and maintenance cost of the system. In practice, however, it is unclear whether this simpler system architecture will be able to provide acceptable performance with real applications at scale, and how it may stack up against the current CDNs. The goal of this paper is to answer these questions through experiments and analysis.

In the following sections, we present how CDN's centrally managed caching and request forwarding compares to NDN's distributed and opportunistic caching and request forwarding. That said, there are certainly other types of ICN/NDN-based solutions for content distribution; for example, one may introduce mechanisms that explicitly coordinate cache placement and access in NDN networks [30]. While the eventual design of an ICN/NDN-based content distribution network may still be under research, understanding the advantages and disadvantages of the basic NDN solution will provide a baseline assessment and help the development of a complete solution.

## 3.2 Experiment Setup

To study and compare CDNs and NDN at the Internet-scale, we need to access a real-world NDN network. At the time of this work, the NDN community has been running a global testbed for over a decade and we use this testbed for experiments. Thus, we evaluate three different content networks, i.e., Akamai, Fastly, and the NDN testbed [7]. Due to the lack of access to the network core (like an individual servers) of Akamai and Fastly, we treat each of them as a black box by studying the network's outputs (i.e., the traffic received by the video server) according to different types of network's inputs (i.e., requests from end-users). This way, we have collected an analyzed a large body of log files on the end-users and the origin video server. We also need a real-world application to generate traffic so that we can compare its performance over Akamai, Fastly, and NDN. To this end, we deployed an *adaptive*

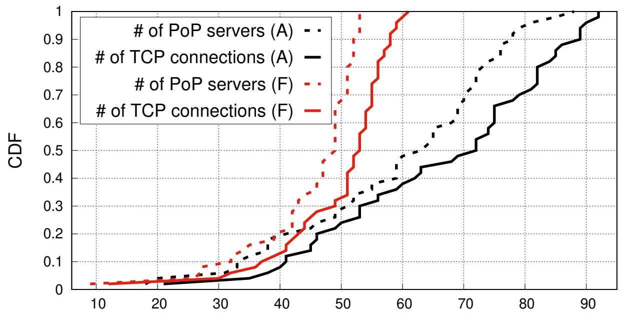

**Figure 3: Number of PoP servers of Akamai (A) and Fastly (F) that contacted the origin and the number of TCP connections they established for video streaming during the two-week experiment.**

*video streaming* service [22], a popular application that contributes to the majority of modern Internet traffic [5]. One challenge, though, is that existing IP services cannot run over NDN directly. This is because all network communications in IP are abstracted by *sockets* while communications in NDN are based on *names*. Thus, we need to modify the streaming application that we use on Akamai and Fastly and make it run over NDN. We must, though, ensure that the performance differences over the CDNs and NDN come from the differences between the networks and not the implementation details of the IP and NDN versions of the application. We met this goal by developing a JavaScript library that loads on the client-side and adds NDN functionalities to the browser without modifying anything in the streaming application. This way, on Akamai or Fastly, the HTTP requests issued by the video player will be resolved by sending IP packets to the network; whereas on NDN, the same HTTP requests will be resolved by expressing NDN packets to the NDN testbed. The details of the design, implementation, and deployment of this video streaming application are available in [22]. The source code of this application is released on GitHub [4, 6].

End-user virtual machines are placed in five different locations, including United States (referred to as North America), Canada, Europe, South America, and Asia, and we have up to two end-users in each location. The video server (i.e., the origin) is located in Arizona; it serves several hundreds of videos of variable lengths (all less than 15 minutes). We run a headless video player on the end-users for two weeks, randomly picking and streaming videos while collecting data from both the player and the origin.

Each video of our experiments is encoded in five different resolutions from `240p` (the poorest) to `1080p` (full HD). The adaptation logic in the application dynamically estimates the network bandwidth and asks the video player to switch to a higher or a lower resolution without involving the end-user.

## 3.3 Quality of Experience

We measure QoE on the end-user side in terms of (1) *video quality*, the video resolution that the end-user experiences, (2) *startup delay*, the amount of time that the end-user waits until the video starts playing, and (3) *number of re-bufferings*, the number of stops/stalls during video playback.

*3.3.1 Video Resolution.* Fig. 2 shows the experienced video resolution by the end-users via different networks. All three networks

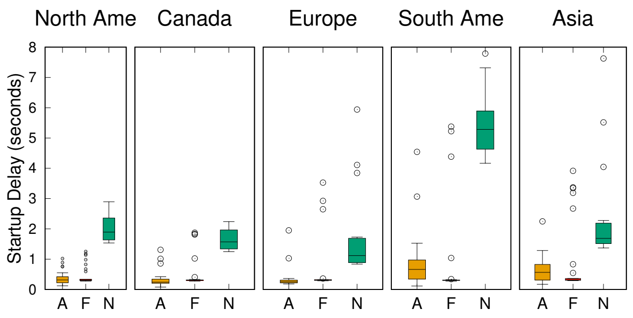

**Figure 4: Video startup delay for end-users in five different locations during the two-week experiment via Akamai (A), Fastly (F), and the NDN testbed (N).**

provide satisfactory QoE on average, with no re-buffering experienced by end-users during video playback. For almost all video streams, Akamai and Fastly provided the end-users with a full HD resolution. The video resolution over the NDN testbed is high for North American and Canadian end-users, but sometimes poor for South American and Asian users. Looking at this figure, we can see that increasing geographical distance between the end-users and the origin does not affect the video resolution in CDNs, but it directly impacts the QoE in NDN — the question is *why this happens?* To answer this, Fig. 3 shows the distribution of number of Akamai and Fastly PoP servers that contacted the origin as well as the number of TCP connections they established for *a single video streaming* during the two-week experiment. For example, we can see that during 90% of all video playbacks, Fastly downloaded the media files (e.g., audio and video segments) by establishing more than 38 TCP connections to the origin through over 30 different PoP servers.

To get a better sense of these numbers, we need to briefly explain how the adaptive video streaming works. In this service, each video is encoded into five different resolutions each of which is broken into 4-second audio and video segments as separate files. A manifest file will be created to put the information of all these files together. The video player asks for a batch of files, including video and audio segments as well as the manifest file, all together to buffer a few seconds of the video. In our service, the buffering time is 12s, meaning that the video player asks for 7 files, including 3 audio and 3 video files as well as the manifest file, at once. This process repeats till the end of the presentation. Fastly and Akamai download all these files together through different servers to maximize the overall throughput on the end-user side (as seen in Fig. 3). In NDN, however, lack of sufficient resources causes serialization such that the solicited files are downloaded one at a time. Putting Fig. 3 and Fig. 2 together, we can see that a major reason of the apparent superiority of CDNs over NDN is compensating the geographical distance with massive parallelism of the resources.

*3.3.2 Startup Delay.* As another QoE metric, Fig. 4 shows the experienced startup delay by the end-users via different networks during the two-week experiment. The startup delay via CDNs is mostly below 0.5s, while through the NDN testbed it is about 2s (and worse for end-users in South America). Understanding why CDNs outperform NDN leads us to Fig. 5. The changes in startup delay mostly come from whether the video is cached in the network

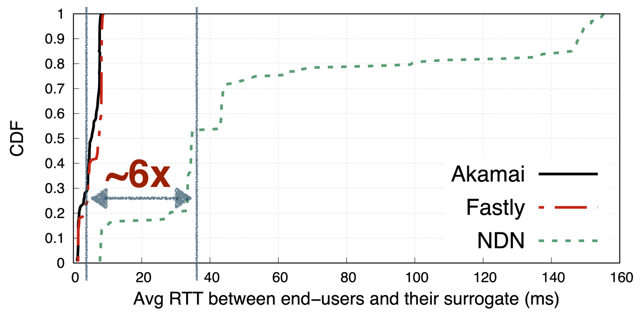

**Figure 5: Average RTT between the end-users and their surrogates during the two-week experiment.**

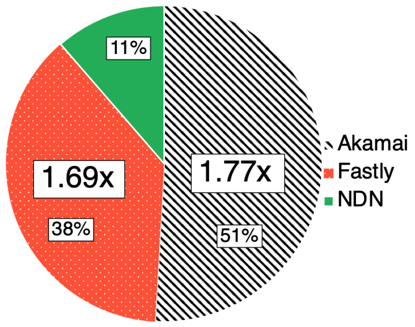

**Figure 6: Compared to Akamai and Fastly, the NDN testbed reduces the origin workload by 77% and 69%, respectively during the two-week experiment.**

and if so, how far the cache is from the end-users. Fig. 5 shows the average round-trip time between the end-users and the surrogate they connected to during video streaming. For 100% of the streamed videos over Akamai and Fastly, the surrogate was located less than 9ms away from the end-user. However, in NDN, only during 14% of the streams did the end-users connect to such a close surrogate. As we can see, if the whole video was cached at the surrogate, the median startup delay in Akamai and Fastly would be 6 times better than NDN on average. A clear example of this observation is a high startup delay in South America due to the lack of testbed infrastructure in that area — the closest NDN surrogate to the end-users in South America was 148ms away. These observations once again remind us the substantial role of caching at the edge of the network in improving QoE [16, 44] that is the product of the network's large infrastructure and massive deployment.

Alongside their large deployment coverage, another reason for CDNs' high QoE is their optimized software/protocol implementations. Compared to CDNs, NDN has been mostly an academic project developed and maintained by the research community. NDN's software, as of today, is merely a prototype of the technology, delivering the basics of NDN design with no production-level fitness. To explore the impact of software implementation on the QoE, we ran a separate set of experiments in which we filtered out the contribution of hardware maturity, including parallelism and in-network caches. In these experiments, we *roughly* quantified the maturity/performance of software implementation (including tools, applications, algorithms, transport protocols, etc.) in each content network by measuring the throughput of the end-to-end connection between an end-user and the origin over each network. For one week, we downloaded a single large file multiple times via each network from different locations. For this set of experiments, (1) neither Akamai nor Fastly employ parallelism as end-users download a single file, and (2) none of the networks utilizes in-network caches as we purge the file from the networks before each run.

The results show that the NDN testbed achieved throughput of only 7.54 Mbps on average (and 23.48 Mbps at best). In comparison, the average throughput over Akamai and Fastly was 96.7 Mbps and 83.21 Mbps, respectively. It is worth mentioning that during each experiment, we monitored the testbed servers to ensure that they did not hit a resource (e.g., CPU and memory) threshold. This confirms that the apparent superiority of the evaluated CDNs over the testbed does not come from the testbed servers' insufficient hardware resources, but instead, it originates from their software implementations (e.g., their forwarder daemon [8]).

What discussed in this section has been all from the end-users' point of view, and neither CDNs' centralized nor NDN's decentralized design seem to play the main factor in determining the QoE. In fact, both Akamai and Fastly strive for (1) caching the contents as close as possible — less than a few milliseconds — to the end-users, and (2) downloading contents from the origin using their highly optimized software, through a massive parallelism to maximize the overall throughput. Compared to CDNs, the current version of the NDN testbed can barely support these two goals due to its software immaturity and lack of large deployment. To summarize, Akamai and Fastly clearly outperform the NDN testbed in terms of QoE, but it seems that this advantage mainly comes from resource over-provisioning and software maturity.

As mentioned earlier, we had no access to Akamai and Fastly core networks; thus, our methodology for measuring these systems is limited to observing their inputs and outputs. This clearly filters out the role of a wide variety of factors, such as protocols, policies, type of hardware and software, etc., from our measurements. While our observations are not comprehensive, they still provide some high-level insights. For example, we cannot quantify the contribution of the CDNs' cache lookup algorithm and compare it with NDN's, but this does not change the fact that a high parallelism, mature software, and edge caching in CDNs contribute significantly to their high QoE.

## 3.4 Origin Workload

Another main goal of CDNs is to reduce the workload of origin servers. In other words, CDNs try to minimize their dependency on origins by serving the contents directly from CDN network as much as possible. In this section, we try to understand how Akamai and Fastly meet this goal and how well the NDN testbed performs. The island-like network architecture of Akamai and Fastly significantly increases their dependency on origins. This dependency comes with two downsides, *increased origin workload* and *reduced failure resiliency* (Sec. 3.5).

*3.4.1 Network Architecture.* Fig. 6 compares the traffic received by the origin during the two-week experiment through each network. Small numbers in this figure present each network's traffic load contribution against the total amount of traffic received by the origin during the two-week experiment. For example, Fastly is accountable for 38% of the total traffic received by the origin from all three networks. The big numbers show the ratio of traffic load contributions of Akamai and Fastly to that of NDN. For example, Akamai incurs 1.77 times more traffic load to the origin than NDN does. In other words, NDN reduces the origin workload by 77% and 69% compared to Akamai and Fastly, respectively. One of the main reasons for this advantage is that the testbed handles the redundancy between end-users' requests more efficiently. The end-user demands on the Internet, specifically for multimedia contents, are highly redundant over time and space [26]. For the requests that ask for the same content, the chance of hitting an in-network cache over the NDN testbed is much higher than CDNs. This is because of universal hop-by-hop request forwarding and non-partitioned network of the NDN testbed. However, increasing the number of islands in CDNs directly increases the chance of receiving duplicated requests by the origin. This partially explains the higher origin workload in Akamai compared to Fastly in Fig. 6 as there is a great gap between the number of PoPs in Akamai and Fastly. At the moment, Akamai owns around 300 PoPs [3], while Fastly has only 60 PoPs [1].

On the NDN testbed, the very first end-user accessing a content will retrieve it from the origin, but future requests for the same content will be mostly handled by the network, regardless of where the end-users are. On the testbed, a request can find the solicited content in three possible locations: (1) *on-path caches*, (2) *off-path caches*, and (3) origin. On-path caches refer to all servers on the path to the origin, and due to hop-by-hop request forwarding, any request may be satisfied by the servers on the path to the origin. Among on-path caches, the server to which the origin is directly connected has the highest chance of having the content. This is because that server is on the path between any end-user and the origin, regardless of the end-user's location.

Off-path caches refer to any server that is not on the path to the origin but contains the solicited content. Because of their stateful forwarding plane, NDN servers can actively send out a given request to different interfaces to learn whether any other server in the network has the desired content. During this process, it is very likely that a nearby server that has recently cached the solicited content replies, even though it might be in the completely opposite direction (i.e., off-path) from the origin. Eventually, if a request does not hit a cache, it will be satisfied by the origin. It is worth mentioning that on-path and off-path caches drastically reduce network dependency on the origins as the numbers show.

*3.4.2 Request Aggregation.* Another important factor that impacts origin workload is *request aggregation*. Through this feature, the network aggregates simultaneous requests for the same content and sends only one copy of the request upwards to the content holder. When the server that aggregated the requests receives the content, it will respond to the pending requests. Because of their stateful forwarding plane, NDN servers natively support request aggregation. Our experiments show that Fastly also supports request aggregation at surrogate and PoP level. Akamai, however, does not support this feature anywhere in the network. To confirm this observation, we started streaming *a single video* by a group of end-users in one location all *at the same time* and repeated this experiment via each network. The end-users need to be geographically close to ensure that they all connect to the same surrogate, and consequently, the same PoP in Akamai and Fastly. Fig. 7 shows the number of requests received by the origin against the number of concurrent video streams. As we can see, when end-users use Fastly and the NDN testbed, increasing the number of concurrent streams does not increase the number of received requests on the origin. This means that these two content networks suppress duplicated requests on the path to the origin. Akamai, however, expresses all the received requests from the end-users to the origin because increasing the number of concurrent streams for the same video increases the number of received HTTP requests on the origin. This is another major reason why Akamai incurs the highest workload on the origin in Fig. 6.

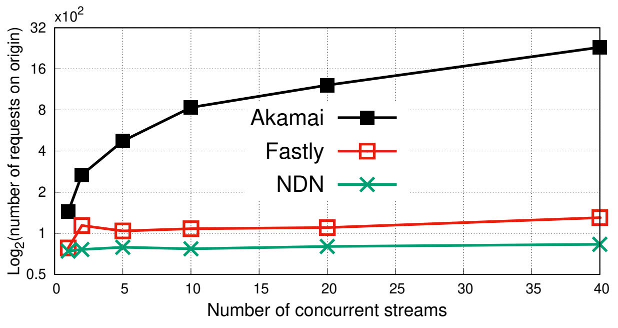

**Figure 7: Changes on the number of received requests on the origin by increasing the number of concurrent streams for a single video from one location via different networks.**

Here, we can see that CDNs' large deployment cannot reduce their dependency on the origin, a result of their island-like network design. In contrast, despite having substantially limited resources, the NDN testbed performs better due to its decentralized design and stateful forwarding plane. To summarize, this section revealed that compared to CDNs, NDN has higher potential to keep the workload of the origins lower by utilizing off-path and on-path caches as well as native request aggregation.

**Table 1: The success ratio of each network in serving the end-users' requests from their caches when the origin stops serving content.**

| Network | Akamai | Fastly | NDN |
|---|---|---|---|
| Success ratio | 9.42% | 13.98% | 100% |

## 3.5 Failure Resiliency

CDNs have shown their vital role in maintaining *connectivity* during outages on the Internet [9]. Decades ago, the connectivity referred to having a physical connection to a given machine in the network to access its resources (e.g., printer, tape drive, processor, etc.). This definition of connectivity works in a transport network; however, in content networks, the meaning of connectivity evolved to *having access to a given content*. Thus, to measure the resiliency of a content network during any kind of failure (e.g., in software, hardware, and network), we should not ask whether the network can still connect the end-users to a specific machine, but if it can keep serving solicited contents. To better understand this distinction, consider the following example. In January and December 2008, parts of Southeast Asia, the Middle East, and Egypt experienced a large network outage when a series of undersea cables were accidentally cut [2]. During that time, Akamai continued serving its customers' contents to many end-users in those locations [37]. Here, from end-users' point of view, no outage to their application experience as they were able to access the desired contents. However, from the network's point of view, many machines were inaccessible. Therefore, *the higher the chance of resolving a solicited content, the more resilient the network is*. This shift in the meaning of connectivity shapes our approach for evaluating the failure resiliency of the studied networks.

To measure the resiliency of Akamai, Fastly, and NDN, we study how they perform when the origin (or its content) temporally goes offline. Accordingly, we run the following experiment for each network. First, we let a North American end-user on the U.S. East Coast download a file. In Akamai and Fastly, this file will be cached in an east coast island, and in NDN, the file will be cached by the servers on the path from the end-user to the origin. Immediately afterwards, we remove the content on the origin and then begin downloading the same file from different locations around the world to see what percentage of end-users' requests can successfully access the file. We repeated the entire experiment multiple times by changing the location of the first end-user and the results were consistent. Also, it is prudent to mention that for both Akamai and Fastly, we enabled the *serving-stale-content* feature, so that if a server has the content then it will reply no matter if the cached content is stale or not.

Table 1 shows the percentage of the requests that could successfully access the file via each network. As we can see, fewer than 10% and 14% of requests via Akamai and Fastly, respectively, could successfully download the file; the rest encountered an error. In Akamai and Fastly, all successful requests were satisfied by the same East Coast island (which had the content in its cache). This is a good example of when a content *is* available in a CDN, but it cannot be served due to the network's design. However, when the end-users switched to the NDN testbed, all of their requests were successfully served by the network. This is because NDN removes the limitations that come with network partitioning by looking at the network as a whole and employing on-path and off-path caches that significantly increases the chance of hitting a cache in the network. For example, all requests over the testbed have a very good chance of being served by the server that is directly connected to the origin even if the requests could not find the content anywhere else in the network.

To summarize, we quantified two technical issues that come with CDNs' island-like network architecture that cannot be addressed by resource provisioning. The results also show that these two issues are properly answered by the NDN testbed since its stateful forwarding plane prevents network partitioning and enables unique in-network caching features.

## 3.6 Security

One of the major requirements of Internet-scale content distribution is security and privacy. This, however, is quite a large topic to cover. Therefore, here, we only scratch the surface and briefly discuss the main idea of security and privacy in CDNs and NDN.

CDNs, like other TCP/IP-based systems, secure the communication channel between two end points, such as the user and the surrogate, by encrypting the traffic at the transport layer – a common approach is to employ Transport Layer Security (TLS) [15]. This mechanism protects data confidentiality and integrity because third-parties would not be able to either see what content is being retrieved or make any changes to the content. However, this protection only applies to the channel between the user and the surrogate.

NDN, on the other hand, adopts a data-centric security model in which every data packet is signed by the producer at the time of content production [47]. The data signature binds the data name, payload, and signing key of a content to make content's data packets immutable. This way, each data packet, no matter where in the network it comes from, explicitly provides its own integrity and provenance without depending on an external communication channel or entity. As a result, contents can safely move around and be cached anywhere. To better understand the comparison of these two approaches, let us look at some common scenarios/examples.

***Keep the private key private.*** Assume company $X$ just started delivering its contents to its end-users through a CDN service. This means the content will be provided by CDN servers instead of company $X$'s servers. This cannot be done *transparently* since a CDN server cannot authenticate itself as a company $X$'s server; unless, it holds the private key of company $X$. As a common practice to mitigate this issue, company $X$ shares its private key with the CDN company. This, however, constitutes a major security concern as private keys are never meant to be shared. This issue is rooted in the fact that end-users place their trust in the machine they are talking to instead of the content they receive. In contrast, NDN does not have such an issue because the end-user can verify the content integrity and its producer (i.e., company $X$) no matter from where (e.g., a surrogate, an origin, a network cache, etc.) and through what channel (e.g., an encrypted channel) it is received.

***In data we trust.*** In CDN's security model, once a surrogate is authenticated, the end-user trusts everything coming from the surrogate. However, a wide variety of attacks are possible to exploit this assumption, including DNS redirection (to redirect the end-user to a malicious server instead of a legitimate surrogate), man-in-the-middle (to relay the traffic between the end-user and the surrogate), and a compromised/malicious surrogate (when a legitimate CDN surrogate serves forged contents). NDN, on the other hand, will always verify data using the signature that comes with the data, therefore not susceptible to compromised channels and surrogates.

***Schematized trust for dynamic contents.*** Accelerating dynamically generated contents has become a major feature of CDN services. Supporting dynamic contents seems to pose a challenge to NDN, though. This is because generating contents on-the-fly requires signing the data by the machine that generates it, usually a server other than the original producer. To get a better sense of this issue, consider the following example. It is very common that CDN companies modify their customers' web pages to embed some controlling information and metadata in them. Here, the content of the web pages belong to a customer, but they are generated by the CDN servers upon receiving a user's request, on-the-fly . To support such behavior, CDNs need to access the customer's private key; NDN, on the other hand, does not. According to NDN's principle, whoever produces/generates the data must sign it. Therefore, content receivers (e.g., end-users, servers, etc.) will trust the data they receive only if the entity/machine that signed it is known as an *authorized* entity by the original producer. NDN employs a *trust schema* [47] to let an origin explicitly specify which keys are allowed to sign its contents in the network. So, considering the above example once again, using NDN's security model, the content producer could simply specify a CDN company as an authorize entity to sign its contents, without sharing its private key with the CDN company. This mechanism provides a versatile, general-purpose way to configure trust in the network and can be readily applied to support dynamically generated contents.

***Moving targets for attacks***. One critical threat of today's online services is distributed denial of service (DDoS) attacks which consume great volumes of network bandwidth and computing resources of targeted machines. The common strategy that CDNs use against DDoS attacks is to throw in massive amount of resources and hide origin servers behind them. However, the unprecedented growth of the number of devices on the Internet (e.g., smart home and IoT devices) magnifies the power of DDoS threats [11], causing an ever-increasing resource demand, financial cost, and system administration complexity in CDNs. NDN, however, handles DDoS attacks in an architecturally different way. It removes the concept of destination in the network, meaning traffic is not delivered to a particular node, but *towards* desired contents and can be satisfied by any cache in the middle. Without a well-defined target, DDoS attacks will be much harder to achieve [13, 18].

***Privacy versus efficiency***. NDN's data-centric security model has data integrity built in and natively guarantees the integrity of every piece of content in the network. However, it needs an efficient solution for *privacy* to support content name and payload confidentiality. An end-to-end encryption solution, although protects the content from eavesdropping, makes the contents' names opaque to the network (except the end points of the encrypted channel), so the network will not be able to cache the contents. One possible alternative is to employ hop-by-hop encryption, meaning that NDN servers establish a secure tunnel between each other and end-users. This way, content privacy is protected and contents can be cached by any server while they travel in the network. The unknown part is the overhead, i.e., how to balance the need of privacy versus the efficiency of data transfer. Nevertheless, to the best of our knowledge, this issue is an open problem under active research.

To summarize, large-scale content distribution can highly benefit from NDN's data-centric security model. This model lets the contents be safely cached and retrieved by any network machine while enabling content owners to explicitly enforce/configure the trust in the network. However, privacy protection can reduce the efficiency of content distribution in NDN and further research is needed to reach an appropriate solution.

## 4 NDN AS CDN

In this section, we put the content network built by CDNs and NDN side-by-side to understand NDN's major benefits over CDNs' regarding network architecture and features. Here, instead of repeating some of CDNs' many benefits [3, 37, 44], we consciously focus our discussion on the benefits that NDN introduces to content networks to show more about the advantages of this emerging technology. In the next section, we will articulate several first-hand lessons and challenges learned about NDN deployment during this study.

### 4.1 Centralization to Decentralization

There are two main centralized modules in CDNs, both outcomes of CDNs' forwarding plane, that NDN can remove. The forwarding plane in a content network needs two types of information to work efficiently, (1) network topology, and (2) content availability.

To provide the forwarding plane with the changes in network topology, CDNs employ a centralized entity, called the *application-layer routing module*. This module exploits the path diversity of the underlay network (i.e., the Internet) and steers the packets through high throughput, low latency overlay paths. It frequently computes overlay paths using a massive amount of both historic and real-time statistics collected from both underlay and overlay networks. It then populates the forwarding table of servers to promptly react to changes in the network. Clearly, the larger and more responsive the CDN, the more network and computation resources this module demands. The stateful forwarding plane of NDN, on the other hand, enables each server to adaptively react to network changes locally without having a strong dependency on application-layer routing information. As a result, the apparent benefit of NDN's adaptive forwarding plane is to reduce the frequency of collecting and processing information by the application-layer routing module and asking for less network and computational resources, compared to what CDNs demand.

The second module provides CDNs' forwarding plane with content availability information. This module tracks the cached contents in each island and guides servers where to find a solicited content in the network. NDN's forwarding plane is deliberately designed to find in-network (off-path and on-path) caches at each hop, with no centralized orchestration between the servers. In fact, NDN technology reunifies the islands in a CDN, and makes one universal content network, where contents can be potentially served from anywhere. This way, the allocated resources to the modules like parent servers in Akamai and the mapping table in Fastly can be restored in each island.

Both of these changes can sharply reduce the system complexity of CDNs; thus, sufficient number of resources can be freed to be used for other purposes or be completely cut from the system. This is directly translated to reducing financial system costs in terms of hardware, maintenance, and administration.

### 4.2 Network-level Features

Besides request aggregation, discussed in Sec. 3.4.2, this section introduces two more unique features that NDN's forwarding plane natively supports at the network layer, *multipath* and *multisource*. These two features enable the forwarding plane to fetch different parts of a single file from different network paths (multipath) and/or different content holders (multisource). In none of our experiments did we observe any sign of Akamai and Fastly supporting these features. If multipath was a standard network feature in the studied CDNs, the origin would receive HTTP requests for the same file concerning different byte-ranges from different machines. To check the multisource feature, we ran two video servers and configured our Akamai and Fastly account to use both of them as origins.

However, neither of the networks interleaved the HTTP requests for the same content between the servers.

NDN technology, in contrast, supports these features intrinsically because contents in NDN are chunked into small pieces, and the consumer sends requests for each of those chunks separately. Those requests might take completely different paths or be satisfied by different caches in the network. Thus, the consumer fetches different parts of the solicited content from different network paths and different content holders. However, fully utilizing these features needs a sophisticated solution, and our results confirm that the NDN testbed lacks such a solution. More specifically, due to the lack of a stable and efficient algorithm, the testbed servers cannot use more than one path at a time to download a content.

## 5 LESSONS & DISCUSSION ON NDN

We want to wrap up the paper by sharing several important lessons we learned about the limitations/challenges of the current NDN deployment during this work to help future research.

***Hardware & Software Maturity***: In Sec. 3.3, we speculated that one of the main causes of a poor QoE on the NDN testbed is the lack of sufficient hardware and optimized software. At the time of writing this paper, there are only a dozen servers on the testbed that can relay the traffic of our streaming service between the end-users and the origin; several of them are old machines without sufficient computation and bandwidth capacity. Moreover, in these experiments, we clearly observed that the lack of optimization in NDN's forwarder (NFD [8]), impedes NDN demonstrating the strengths of its novel design in the real world. For example, we identified and resolved several major software bugs on the testbed that were throttling the system performance. To highlight how software maturity is important, before fixing the bugs, the QoE numbers on the NDN testbed were at least two times worse than what Sec. 3.3 shows currently. Although one may argue that hardware and software of the testbed do not need to be at the production-level, we believe a lack of such a mature system has been holding back NDN technology from showcasing its abilities and being promoted in academia and industry for a long time. This paper, as the first work that measured NDN technology in the real-world at large scale, makes it clear that what gives NDN a real shot of being part of the future networking is a production-level demonstration, not solely a good design.

***The Need for NDN Applications***: At the moment, IP applications and services cannot directly run over NDN. This, by itself, might not be an issue; however, developing NDN applications is a quite complex and time-consuming process. For example, in this work, we spent months in developing the NDN version of streaming application, debugging and tuning for performance. This well-known issue is confirmed by several other works in the literature [19, 24, 32, 46]. Although the NDN community has developed a rich collection of libraries and tools over the last decade, it has produced only a handful of applications, most of which are quite difficult to use. Lack of a bundle of popular applications/services for Internet users in the community is turning into NDN's Achilles heel, demanding researchers to spend significant time and energy on engineering/development instead of solving scientific questions. We believe that at this stage of architectural development what is keenly needed is to run applications/services for daily Internet users on the testbed. This not only encourages end-users to use this technology, but it also allows researchers to evaluate and measure NDN technology based on real traffic from real users.

***Management and Debugging***: Alongside the unique features of NDN's forwarding plane, there are some concerns, in particular management difficulties. In NDN, it is fairly challenging to accurately determine the path that end-users' requests take towards the solicited content in the network. This is because servers treat routing information as a hint and mostly make their forwarding decisions based on their historical local measurements. In this work, we spent many hours learning how the NDN testbed forwards requests of a given end-user. For instance, during one video streaming from London, we confirmed that the requests took eight different paths to three different caches in the network. Although such diversity has several benefits (like failure resiliency), its combination with lack of a sophisticated monitoring platform on the testbed hinders learning how NDN works in the real-world. Some papers [29, 36, 39] in the literature have appropriately emphasized the importance of this issue and attempted to address it. However, compared to the current CDNs, NDN's decentralized nature makes network management and debugging quite complex. We believe deploying a monitoring/debugging platform on the NDN testbed significantly facilitates not only the network management for administration purposes, but also the network measurements and evaluations for researchers and developers.

## 6 CONCLUSION

A decade of research and development on NDN technology has produced today's global NDN testbed, a world-wide content network that employs NDN technology. In this paper, for the first time, we evaluated this network in the wild and compared it with two well-known CDNs, Akamai and Fastly. By running an adaptive video streaming service and streaming hundreds of videos from four different continents, we attempted to understand how the content network built by CDNs compares to that of NDN. We argued that QoE in these networks is mainly determined by their hardware and software maturity; this is why the QoE in CDNs is better than the NDN testbed at present. We also demonstrated that features such as origin workload and failure resiliency are mainly the products of the design of these networks; that is where NDN outperforms CDNs. We discussed the security model in these networks and then articulated the main benefits and challenges we learned about NDN. In the end, we believe that having a resilient and scalable content network is not a matter of the distant future; NDN technology has a great potential to realize such a network if accompanied by mature software and sufficient hardware resources.

## ACKNOWLEDGEMENT

We are grateful for the invaluable suggestions made by the Shepherd, Dirk Kutscher, and the comments from anonymous reviewers.

This material is based upon work supported by the National Science Foundation under Grant No. CNS-1629009 and a Huawei grant. Any findings, discussions, and recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsor.

## REFERENCES

[1] Fastly release notes. https://www.fastly.com/release-notes/q1-2019. [Online].
[2] TeleGeography. Cable cuts disrupt Internet in Middle East and India. https://www.commsupdate.com/articles/2008/01/31/cable-cuts-disrupt-internet-in-middle-east-and-india/, January 2008. [Online].
[3] Akamai facts & figures. https://www.akamai.com/us/en/about/facts-figures.jsp, 2020. [Online].
[4] The back-end implementation of an adaptive video streaming service over NDN. https://github.com/chavoosh/ndn-mongo-fileserver, 2020. [Online].
[5] Cisco Visual Networking Index: Forecast and Methodology, 2017–2022. https://bit.ly/3ftToFW, February 2020.
[6] The front-end implementation of an adaptive video streaming service over NDN. https://github.com/chavoosh/ndn-video-frontend, 2020. [Online].
[7] NDN Global Testbed. https://named-data.net/ndn-testbed/, 2020. [Online].
[8] NFD developer's guide. http://named-data.net/doc/NFD/current/, 2020. [Online].
[9] South East Asia–Middle East–Western Europe 4 (SEA-ME-WE 4). http://goo.gl/kW3bE, 2020. [Online].
[10] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.
[11] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al. Understanding the mirai botnet. In *26th USENIX Security Symposium*, pages 1093–1110, 2017.
[12] F. Chen, R. K. Sitaraman, and M. Torres. End-user mapping: Next generation request routing for content delivery. In *ACM Conference on Special Interest Group on Data Communication*, SIGCOMM'15, pages 167–181, 2015.
[13] A. Compagno, M. Conti, P. Gasti, and G. Tsudik. Poseidon: Mitigating interest flooding DDoS attacks in named data networking. In *38th annual IEEE conference on local computer networks*, pages 630–638, 2013.
[14] S. Cui, M. R. Asghar, and G. Russello. Multi-CDN: Towards privacy in content delivery networks. *IEEE Transactions on Dependable and Secure Computing*, 2018.
[15] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. 2008.
[16] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: Incrementally deployable ICN. In *SIGCOMM*, pages 147–158, 2013.
[17] A. Flavel, P. Mani, D. Maltz, N. Holt, J. Liu, Y. Chen, and O. Surmachev. Fastroute: A scalable load-aware anycast routing architecture for modern cdns. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 381–394, 2015.
[18] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. DoS and DDoS in named data networking. In *22nd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, 2013.
[19] A. Gawande, J. Clark, D. Coomes, and L. Wang. Decentralized and secure multimedia sharing application over named data networking. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, pages 19–29, 2019.
[20] C. Ghasemi, H. Yousefi, K. G. Shin, and B. Zhang. A fast and memory-efficient trie structure for name-based packet forwarding. In *26th International Conference on Network Protocols (ICNP)*, pages 302–312, 2018.
[21] C. Ghasemi, H. Yousefi, and B. Zhang. iCDN: An ndn-based cdn. In *7th ACM Conference on Information-Centric Networking*, 2020.
[22] C. Ghasemi, H. Yousefi, and B. Zhang. Internet-scale video streaming over NDN. *IEEE Network Magazine*, 2020.
[23] Y. Gilad, A. Herzberg, M. Sudkovitch, and M. Goberman. CDN-on-demand: An affordable DDoS defense via untrusted clouds. In *NDSS*, 2016.
[24] P. Gusev and J. Burke. NDN-RTC: Real-time videoconferencing over named data networking. In *2nd ACM Conference on Information-Centric Networking*, pages 117–126, 2015.
[25] Y. Inaba, Y. Tanigawa, and H. Tode. Content retrieval method in cooperation with CDN and ICN-based in-network guidance over IP network. In *IEEE 40th Conference on Local Computer Networks (LCN)*, pages 454–457, 2015.
[26] S.-W. Jeon, S.-N. Hong, M. Ji, G. Caire, and A. F. Molisch. Wireless multihop device-to-device caching networks. *IEEE Transactions on Information Theory*, 63(3):1662–1676, 2017.
[27] Y. Jia and A. Kuzmanovic. Perceiving internet anomalies via CDN replica shifts. In *IEEE Conference on Computer Communications (INFOCOM'19)*, pages 2197–2205, 2019.
[28] X. Jiang and J. Bi. ncdn: CDN Enhanced with NDN. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 440–445, 2014.
[29] S. Khoussi, D. Pesavento, L. Benmohamed, and A. Battou. NDN-trace: a path tracing utility for named data networking. In *4th ACM Conference on Information-Centric Networking*, pages 116–122, 2017.
[30] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong. Popularity-driven coordinated caching in named data networking. In *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 15–26, 2012.
[31] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu. When HTTPS meets CDN: A case of authentication in delegated service. In *IEEE Symposium on Security and Privacy*, pages 67–82, 2014.
[32] T. Liang, J. Pan, and B. Zhang. NDNizing existing applications: research issues and experiences. In *5th ACM Conference on Information-Centric Networking*, pages 172–183, 2018.
[33] T. Lin, Y. Xu, G. Zhang, Y. Xin, Y. Li, and S. Ci. R-iCDN: An approach supporting flexible content routing for ISP-operated CDN. In *9th ACM Workshop on Mobility in the Evolving Internet Architecture*, MobiArch'14, pages 61–66, 2014.
[34] G. Ma, Z. Chen, J. Cao, Z. Guo, Y. Jiang, and X. Guo. A tentative comparison on CDN and NDN. In *2014 IEEE international conference on systems, man, and cybernetics (SMC)*, pages 2893–2898, 2014.
[35] M. Mangili, F. Martignon, and A. Capone. Performance analysis of content-centric and content-delivery networks with evolving object popularity. *Computer Networks*, 94:80–98, 2016.
[36] K. Nichols. Lessons learned building a secure network measurement framework using basic NDN. In *6th ACM Conference on Information-Centric Networking*, pages 112–122, 2019.
[37] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, 2010.
[38] M. K. Pathan and R. Buyya. A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, 2007.
[39] D. Pesavento, O. I. E. Mimouni, E. Newberry, L. Benmohamed, and A. Battou. A network measurement framework for named data networks. In *4th ACM Conference on Information-Centric Networking*, pages 200–201, 2017.
[40] T. Plagemann, V. Goebel, A. Mauthe, L. Mathy, T. Turletti, and G. Urvoy-Keller. From content distribution networks to content networks—issues and challenges. *Computer Communications*, 29(5):551–562, 2006.
[41] K. Poularakis and L. Tassiulas. On the complexity of optimal content placement in hierarchical caching networks. *IEEE Transactions on Communications*, 64(5):2092–2103, 2016.
[42] D. Rossi and E. Turrini. Analyzing performance data exchange in content delivery networks. In *International Conference on Networking*, pages 737–745. Springer, 2005.
[43] J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib. A survey on replica server placement algorithms for content delivery networks. *IEEE Communications Surveys Tutorials*, 19(2):1002–1026, 2017.
[44] R. K. Sitaraman, M. Kasbekar, W. Lichtenstein, and M. Jain. Overlay networks: An Akamai perspective. *Advanced Content Delivery, Streaming, and Cloud Services*, pages 305–328, 2014.
[45] S. R. Srinivasan, J. W. Lee, D. L. Batni, and H. G. Schulzrinne. ActiveCDN: Cloud computing meets content delivery networks. *Columbia University, Technical Report*, 2011.
[46] J. Thompson, P. Gusev, and J. Burke. NDN-CNL: A hierarchical namespace API for named data networking. In *6th ACM Conference on Information-Centric Networking*, page 30–36, 2019.
[47] Y. Yu, A. Afanasyev, D. Clark, K. Claffy, V. Jacobson, and L. Zhang. Schematizing trust in named data networking. In *2nd ACM Conference on Information-Centric Networking*, pages 177–186, 2015.
[48] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. Named data networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73, 2014.