Hindawi
Mobile Information Systems
Volume 2021, Article ID 6663243, 12 pages
https://doi.org/10.1155/2021/6663243

Research Article

NCDN: A Node-Failure Resilient CDN Solution with Reinforcement Learning Optimization

Zhihao Wang,[1] **Shengyong Du**,[1] **and Min Ren**[2]

[1]*School of Management Science and Engineering, Shandong University of Finance and Economics, Jinan 250014, China*
[2]*School of Mathematics and Quantitative Economics, Shandong University of Finance and Economics, Jinan 250014, China*

Correspondence should be addressed to Zhihao Wang; hy861125@126.com

Received 5 October 2020; Revised 24 December 2020; Accepted 30 December 2020; Published 11 January 2021

Academic Editor: Yugen Yi

Content Delivery Networks (CDNs) have enabled large-scale, reliable, and efficient content distribution over the Internet. Although CDNs have been very successful in serving a large portion of Internet traffic, they have several drawbacks. Despite their distributed nature, they rely on largely centralized management and replication. This can affect availability in case of node failure. Further, CDNs are complex infrastructures that span multiple layers of the networking stack. To address these issues, in this paper, we introduce NCDN, a novel highly distributed system for large-scale delivery of content and services. NCDN is designed to provide resilience against node failure through location-independent storage and replication of content. This is achieved through a two-layer architecture: the first layer (exposure layer) exposes services implemented by NCDN (e.g., Web, SFTP) to clients; the second layer (hidden layer) provides reliable distributed storage of content and application state. Content in NCDN's hidden layer is stored and exchanged as Named Data Network (NDN) content packets. We employ the reinforcement learning (RL) to dynamically learn the optimal numbers of duplicates for different type of contents, because the RL agent has the advantage of not requiring expert labels or knowledge and instead the ability to learn directly from its own interaction with the world. The combination of NDN and RL brings NCDN fine-grained, fully decentralized content replication mechanisms. We compare the performance and resilience of NCDN to those of an idealized CDN via extensive simulations. Our results show that NCDN is able to provide higher availability than CDNs (between 8% and 100% higher under the same conditions), without substantially increasing content retrieval delay.

1. Introduction

In the last two decades, the Internet has established itself as a significant component of our critical infrastructure. Availability of resource and content over the Internet has therefore become an issue of primary importance. Because the Internet is a large, complex, open, geographically distributed artifact, availability of data and resources is limited by common events such as hardware and software malfunctions and misconfigurations [1], natural disasters affecting compute and networking resources [2], and intentional attacks [3]. To address this issue, researchers and practitioners have proposed several distributed architectures that are able to achieve high availability. Probably the most popular of these architectures is Content Delivery Network (CDN) [4]. A CDN is a collection of hosts spread across the Internet and managed by a single entity in a centralized fashion. All new content is uploaded to a host (or a collection of hosts) called *origin server*, which takes care of replicating the content to *CDN servers* across the network. Replication strategies are usually designed to maintain content as close as possible to the consumers that will request it. This minimizes content retrieval delay, maximizes the available bandwidth to consumers, and limits the amount of traffic in the core of the Internet [5]. CDNs are able to provide high availability because of their highly replicated and distributed nature. Although commercial CDNs are reliable, it happens that CDN provider occasionally encounters failure of servers in the past years. For example, in 2017, AWS that hosts most websites and web apps went offline for 4 hours and it surely did cause a huge panic for all netizens. In July 2019, a CDN provider, Cloudflare, encountered network outage, and the

contents stored on Cloudflare CDNs were not accessible for around two hours[1]. Designing a more reliable CDN (resilient to nodes failure) is also an active research topic. For example, [6] introduces VDN, a practical approach to a video delivery network that uses a centralized algorithm that provides CDN operators. It does this in spite of challenges resulting from the wide area (e.g., state inconsistency, partitions, and failures). In [7], the authors proposed a framework for CDN infrastructure upgrade that performs sparse link and replica addition with the objective of maximizing the content accessibility under targeted link cut attacks. Besides the node failure issue, another drawback of CDN is that CDNs are complex entities that rely on specific modifications of several layers (including DNS and application layer). Content distribution and replication are centrally managed by the origin server, and content replication is usually done at a file-level (or coarser) granularity [8]. To address these issues, in this paper, we introduce NCDN, a novel architecture for large-scale distribution of content and services. NCDN is designed from the ground-up to provide resilience to common node failure scenarios [9, 10] through scalable, efficient, and transparent replication of content and services across geographically distributed endpoints. This is achieved through a two-layer architecture. The first layer (exposure layer, see Figure 1) provides lightweight implementation of common services (e.g., Web, databases, and SFTP). Each host in this layer stores no content and is intended to keep only transient (per-transaction) state. The second layer (hidden layer) implements highly available, scalable, low-latency highly distributed storage. This layer leverages Named Data Network (NDN) [11] for data representation, for communication between hidden layer nodes, and between exposure and hidden layer nodes. The combination of NDN and RL brings NCDN fine-grained, fully decentralized content replication mechanisms. Note that hosts in each layer of NCDN are functionally equivalent and geographically distributed. This allows high resilience against node failure [12], because even if a node becomes permanently unavailable, a number of other nodes can immediately and transparently replace it without loss of functionality. Because every functionality implemented by NCDN is fully distributed, our architecture has no single point of failure.

The design of NCDN allows the use of NDN as either an overlay on top of IP or natively (without loss of generality, in this paper, we assume that NDN is run as an overlay). Consumers do not need to be aware of NDN, as they have no direct access to hidden layer hosts. Rather, they interact exclusively with exposure layer nodes, which act as "interfaces" to the content and state stored in the hidden layer. NCDN leverages location independent fixed-size named content packets, which can be efficiently stored in and retrieved from any of its nodes. This simplifies replication and dispersal of data and services, which is the core goal of NCDN. The network as a whole keeps track of the number of copies of each data object in a decentralized fashion. In contrast to IP, NCDN addresses content packets. This implies that a data object can potentially be served by any node in the hidden layer. By removing the need to specify which node must serve a piece of content or implement a service, NCDN substantially reduces architectural single points of failures.

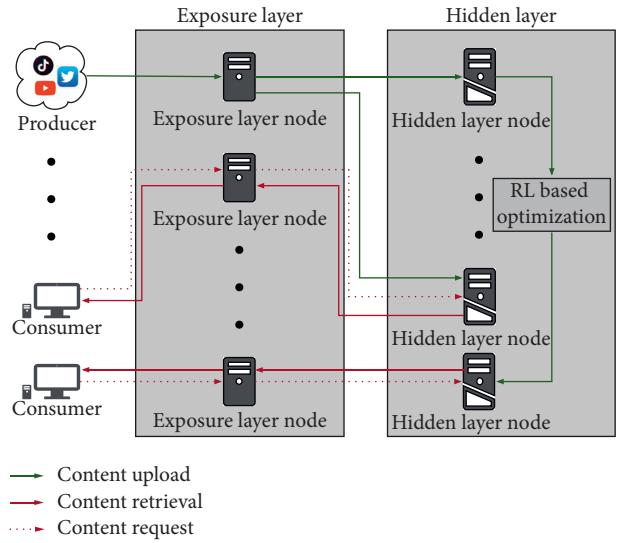

Figure 1: Overview of NCDN.

Reinforcement Learning/Deep Learning-based methods have been used in the networking-related applications for years. For example, [13] applied such technique in the IoT with edge computing. In this paper, the RL-based content replication method is proposed. Each hidden layer node employs Reinforcement Learning (RL) [14] to dynamically determine the optimal numbers of duplicates for different type of contents, based on network conditions and constraints, when and how to perform replication. Thanks to the use of NDN for data representation, hidden layer nodes can store and replicate individual packets that constitute each piece of content. Further, because of the location-independent nature of NDN addressing (content is addressed by name rather than by location), each packet can theoretically be stored in a different host. Discovery and reassembly of the content are handled transparently by the NDN layer.

1.1. Contributions. In this paper, we introduce the design of NCDN and perform an extensive evaluation of its performance in various node failure scenarios. NCDN is an NDN-based two-layer architecture with geographically and fully distributed hosts. By introducing an RL algorithm, NCDN maintains the optimal number of copies of contents to strike a balance between availability and cost. We model NCDN and a generic CDN and compare the two architectures. Our results show that NCDN is able to provide substantially higher availability compared to CDN: under realistic node failure scenarios, NCDN was able to serve between 65% and 100% of the requests, while the CDN was often limited to satisfy 50% of overall requests under the same conditions. We implement NCDN on ndnSIM [15], an NS-3-based network simulator. Our simulation results support the conclusions from our model and demonstrate that NCDN is able to recover quickly in case of node failure.

1.2. Organization. The rest of the paper is organized as follows. Section 2 and Section 3 introduce the designs of CND and NCDN, respectively. Section 4 introduces preliminary considerations for our analysis. In Section 5, we compare the content availability and content retrieval delay in CDN and NCDN. We establish the optimal number of content copies in NCDN in Section 6. In Section 7, we report the results of our evaluation. In Section 8, we summarize the state of the art. We conclude the paper in Section 9.

2. CDN Model

A CDN is a large distributed system that consists of hundreds of globally distributed hosts. Content owners rely on CDNs for large-scale content distribution [16]. Because of their distributed nature, CDNs enable high content availability and low content retrieval delay. The details of CDNs are usually not disclosed by the commercial entities that operate them (e.g., Akamai2). To capture most CDN use cases, in this paper, we model CDN as follows.

Figure 2 shows the overview of the considered CDN architecture in our paper. CDNs are composed of two types of nodes: one or more origin servers and a large number of CDN servers. Producers generate content and upload it to the *origin server* as the green line in Figure 2. The origin server stores all content locally and distributed copies of the content to a subset of the CDN servers as the dotted green lines shown in Figure 2. Consumers retrieve content from a closest CDN server in order to minimize delay (without access to the origin server as the red lines shown in Figure 2.). If a request cannot be satisfied by the selected CDN server, the consumer makes a request to the origin server after a short timeout. Figure 2 summarizes our CDN model.

In practice, the origin server and the CDN servers might be implemented as a large data center rather than a single host. In this paper, without loss of generality, we model the origin server as a single unit. As long as some of the hosts and the bandwidth of the origin server are available to serve consumers, the origin server can satisfy requests for any piece of content.

3. NCDN Design

In this section, we introduce the functionalities of the exposure and hidden layers. We then present a scenario that illustrates how content submission and retrieval are implemented in NCDN.

3.1. Exposure Layer. NCDN is a two-layer architecture that allows transparent, scalable, efficient, and robust services and data replication. The exposure layer is composed of a large number of hosts, which are charge of receiving, translating, dispatching, and responding to requests from clients (i.e., producers and consumers). Exposure layer nodes implement interfaces for arbitrary application-layer protocols. For instance, an exposure layer node can implement an HTTP interface, which listens on port 80 and converts each GET and POST request to the corresponding hidden layer node query. Although in principle different exposure layer nodes can implement different protocols, without loss of generality, in the rest of the paper, we assume that all nodes implement the same protocol. To perform their duties, by introducing the advantages of named data networking, all exposure layer nodes implement the two following data structures: *the Forwarding Information Base* (FIB), and *the Pending Requests Table* (PRT). The FIB stores entries for each piece of content as tuples $\langle \text{namespace}, (\text{node}_1, \text{cost}_1), \ldots, (\text{node}_n, \text{cost}_n)\rangle$, where node_i indicates the IP address or DNS name of a node in the hidden layer that stores content with name starting with namespace and cost_i is a cost metric for requesting data to that node (e.g., round-trip time).

To find a requested content, an exposure layer host searches FIB by using the name of the content. The lookup returns a list of available hidden layer nodes that contain the requested content. If an exposure layer node forwards a request to a hidden layer node and does not receive any response, the exposure layer node considers that the hidden layer node is unavailable and updates the FIB accordingly. Periodically, the exposure layer node sends probe messages to a subset of the hidden layer nodes that store the content to refresh cost_i in the FIB. This mechanism ensures that NCDN is able to deliver content with low delay even when a large number of hidden layer nodes have failed and also the consistency of copies for content. The PRT keeps track of requests that have been forwarded to hidden node layers and for which the corresponding content has not been returned yet. Client can select any exposure layer node to access NCDN. In practice, by querying DNS server, a client is directed to the closest exposure layer node.

3.2. Hidden Layer. The hidden layer stores and replicates data objects and serves request objects received from the exposure layer. It is composed of a large number of nodes; each node is addressable using one identifier (e.g., a DNS name or IP address). A node can be either a single host or a collection of hosts (e.g., a data center). Nodes advertise their capabilities (e.g., bandwidth and available storage space), as well as their topology information and geographical location, to exposure layer nodes. When a hidden layer node receives new content, it sends a *namespace advertisement message* to exposure layer nodes in order to update their FIB. The message contains one or more namespaces, common to the data objects received.

The unit of replication in NCDN is an individual content packet: large pieces of data are split into several small packets, which can independently be replicated and requested. This allows NCDN RL-based replication algorithms to work at a fine granularity level, thus maximizing resource utilization. State explosion is prevented by grouping large sets of data objects into common namespaces. Because there is no central authority in charge of managing replication, there is no small subset of nodes that, if failed, prevents NCDN from making new copies of existing content. We will detail the RL algorithm in Section 6. In summary, a pair of an exposure layer node and a hidden layer node are functional equivalent to a CDN server plus an

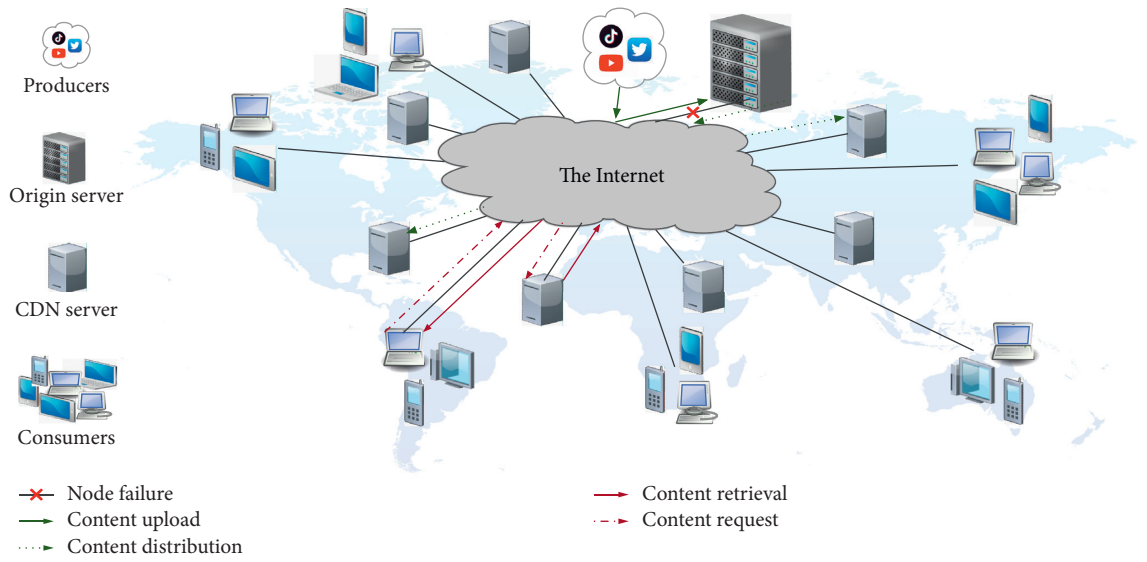

Figure 2: CDN architecture.

ability of receiving uploaded content. That is to say, a pair of an exposure layer node and a hidden layer node are able to store content and distribute content to the consumers as a CDN server and receive uploaded content as an origin server.

3.3. Example Scenario. We now provide an example of how NCDN works. We consider a scenario in which a producer (Alice) uploads a file to a NCDN, and consumer (Bob) retrieves it. Alice connects to an exposure layer node and uploads a file named video.mp4 as http://sdufe.edu/users/Alice/video.mp4. Then, the exposure layer node breaks the file into signed content packets following a protocol-specific naming convention, for example, /edu/sdufe/users/Alice/video.mp4/1 for the first packet, /2 for the second, and so on. Each content packet is forwarded to one or more hidden layer nodes. The hidden layer nodes that receive the data objects advertise them to all nodes in the exposure layer and possibly start the replication process.

Bob retrieves the file by connecting any exposure layer node of NCDN and requesting http://sdufe.edu/users/Alice/video.mp4. The exposure layer node translates the request and sends it to hidden layer node(s) that are able to satisfy it. Once the hidden layer nodes return the corresponding content packets, the exposure layer node reassembles them and forwards them to Bob. Figure 3 shows the procedure where Bob retrieves a content from NCDN and how end-users affect the content of the FIB and the PRT.

3.4. Resources Allocation in CDN and NCDN. Akamai has a total of 3,000 CDN server locations [17]. We assume that exposure layer nodes and hidden layer nodes follow the same geographical distribution of CDN servers. Therefore, in this paper, we consider NCDN consisting of 3,000 exposure layer nodes and 3,000 hidden layer nodes, and exposure layer nodes and hidden layer nodes are colocated. Unlike NCDN where all the nodes in each layer are functionally equivalent, CDN architecture has potential node failure issue, as shown in Figure 2. This is because the origin server has different functionalities compared with CDN servers. In addition, thanks to the named data networking, large pieces of data can be split into several small packets in NCDN, which can independently be replicated and requested. This allows NCDN replication algorithms to work at a fine granularity level, thus maximizing resource utilization.

Note that NCDN "nodes" are not equivalent with CDN "servers," and a pair of an exposure layer node and a hidden layer node are functionally equivalent to a CDN server, so it is not fair to compare NCDN and CDN with same number of servers (or nodes). Same number of "machines" in CDN and NCDN does not indicate same hardware overheads. In the following sections, we compare performance and resilience of NCDN and CDN by allocating the same amount of storage and bandwidth.

4. Preliminaries

In this section, we discuss the preliminaries of our analysis. We begin by discussing how we model content popularity. Then, we discuss how resources are allocated in NCDN and CDN.

4.1. Modeling Content Popularity. Not all content distributed over the Internet is requested the same number of times. For instance, it is widely believed [18–20] that requests of Web content follow the Zipf distribution, where a small number of Web pages are requested a disproportionated number of times. Further, content may follow different distributions in different applications (e.g., Web and YouTube videos). In our analysis, in order to abstract from specific popularity distributions, we assume that the distribution of consumers requests among content is captured by a function $f: \mathbb{C} \longrightarrow [0, 1]$, where $\mathbb{C}$ is the set of all pieces of content

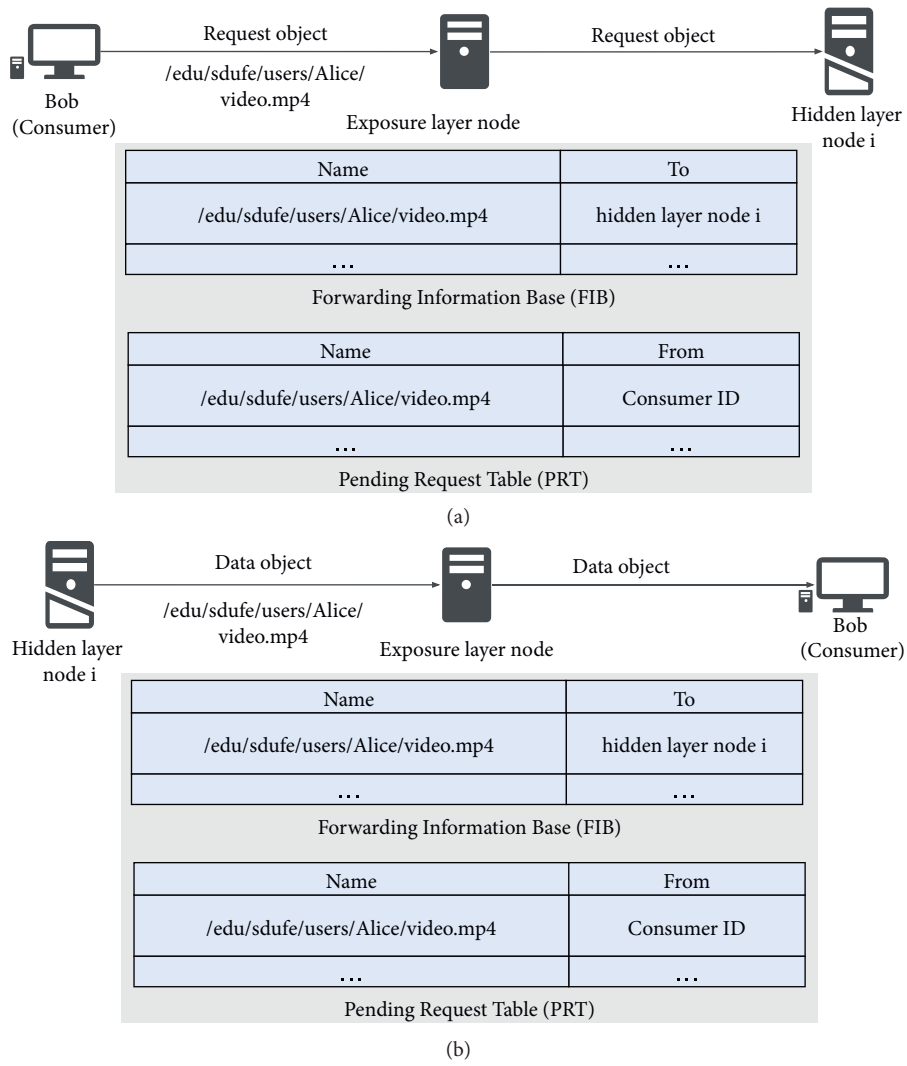

Figure 3: Procedure of content retrieval. (a) An exposure layer host receives a request object from a client. The exposure layer node checks the FIB and forwards the request object to the appropriate node(s) after updating the PRT. (b) Once a data object that can satisfy the request is returned by a hidden layer node, the exposure layer host checks its PRT and sends the data object to the client.

(e.g., all Web pages served by NCDN or by a CDN). $f(c_i)$ represents the popularity of content c_i and is expressed as the total number of requests for c_i, divided by total number of requests to all content. To spread requests evenly across CDN servers, and among NCDN hidden layer nodes, we assume that $f(\cdot)$ can be efficiently sampled and that the sum of the popularities of all pieces of content stored on each CDN server is $1/|S|$, where S is the set of all CDN servers. Similarly, the popularity of all pieces of content stored on each NCDN hidden layer node is $1/|B|$, where B is the set of all hidden layer nodes.

4.2. Initial Setup. In order to provide a fair comparison of NCDN and CDN, we assign the same amount of resources (storage and bandwidth) to the two architectures. NCDN "nodes" are not equivalent with CDN "servers," and a pair of an exposure layer node and a hidden layer node are functionally equivalent to a CDN server, so it is not fair to compare NCDN and CDN with same number of servers (or nodes). In this section, we discuss how we allocated and equalize resources (i.e., storage capacity and and bandwidth) for NCDN and CDN.

We set the number of hidden and exposure layer nodes in NCDN to the number of CDN servers. Therefore, m is equal to the number of CDN servers, to the number of exposure layer nodes, and to the number of hidden layer nodes (the notations used in this paper and the corresponding descriptions are summarized in Table 1). Producers upload content at rate $\mathcal{U}$ to system, while consumers retrieve content at rate $\mathcal{D}$ from the system. Because in many applications (e.g., the Web) the amount of content retrieved by consumers is much larger than the amount of content uploaded by providers [21], we assume $\mathcal{D} \gg \mathcal{U}$. Let c_i be the number of copies of each piece of content i, where $c_i < m$. NCDN requires state to be stored on exposure layer nodes, and not all of this state has counterpart in CDN. To account for this extra space requirement, we assume that a CDN can store $c_i + 1$ copies for the same cost of storing c_i copies in NCDN.

Table 1: Notations.

Notation	Description
m	Number of nodes in CDN
$\mathcal{U}$	Producers upload content rate
$\mathcal{D}$	Consumers retrieve content rate
c_i	Number of copies of content i
k	Origin server capacity
λ	Overprovisioning factor
q	Portion of CAU
t	Round trip delay from users to CDN servers

Let c be the number of copies of each piece of content, where NCDN requires state to be stored on exposure layer nodes, and not all of this state has counterpart in CDN. To account for this extra space requirement, we assume that a CDN can store $c + 1$ copies for the same cost of storing c copies in NCDN.

In order to equalize the bandwidth of NCDN and CDN, we first calculate the minimum bandwidth that the two systems require. Then, for the system requiring smaller bandwidth, we enlarge its bandwidth to the larger one. The way we enlarge bandwidth is to increase the bandwidth of each link of the system with a same ratio. In NCDN, in order to have enough bandwidth to send the requested content to consumers, the worst-case bandwidth required for exposure layer nodes and hidden layer nodes is $\mathcal{D}$. Therefore, the minimum worst-case bandwidth required by NCDN system is $2\mathcal{D}$. This assumes that none of the content packets are retrieved by an exposure layer node from the colocated hidden layer node. In practice, we expect the required bandwidth to be between $\mathcal{D}$ and $2\mathcal{D}$. Because consumers send requests to CDN servers first, the combined bandwidth of all CDN servers must be sufficient to satisfy all simultaneous requests. In order to send content to consumers successfully, the minimum combined bandwidth allocated to all the CDN servers is $\mathcal{D}$. The origin server cannot usually serve all the requests from consumers. We set a factor k to represent the portion of requested content that the origin server is able to deliver to consumers, where $k \leq 1$. In practice, the value of k varies among different applications, so the range of value k is $[0, 1]$. Note that, in the extreme case, where k equals the maximum value 1, $\mathcal{A}$ is always not larger the bandwidth of origin server. However, in other cases where the value of k is less than 1 (or even smaller such as close to 0), $\mathcal{A}$ is larger than the bandwidth of origin server. The minimum bandwidth required by the origin server to serve partial requests is $k\mathcal{D}$. Therefore, the minimum bandwidth required by CDN is $(1 + k)\mathcal{D}$. Because $k \leq 1$, CDN needs to enlarge its bandwidth of each link with a ratio r, where $r \geq 1$:

$$r = \frac{2\mathcal{D}}{(1+k)\mathcal{D}} = \frac{2}{1+k}. \tag{1}$$

In this paper, we also consider bandwidth overprovisioning. In practice, because of flash crowds [22], service providers tend to allocate more bandwidth to their servers (e.g., origin server and the CDN servers in CDN) than what is expected under average traffic conditions. Therefore, in our analysis, the bandwidth of all links in both CDN and NCDN is increased by an overprovisioning factor λ, where $\lambda > 1$. In our analysis, the combined bandwidth allocated to CDN and NCDN is $2\lambda\mathcal{D}$. In NCDN, the combined bandwidths allocated to exposure layer nodes and hidden layer nodes are both $\lambda\mathcal{D}$. In CDN, the bandwidth of each link is multiplied by factors r and λ and, hence, the combined bandwidth allocated to CDN servers is

$$B_s = \frac{2\lambda\mathcal{D}}{1+k}. \tag{2}$$

The bandwidth allocated to origin server is

$$B_{\text{origin}} = \frac{2k\lambda\mathcal{D}}{1+k}. \tag{3}$$

5. Comparison between NCDN and CDN

In this section, we compare the performances of NCDN and CDN. Our comparison is based on *content availability* and *content retrieval delay*. *Content availability* represents how much of the content can be retrieved by consumers and is defined as the number of successful content requests divided by the total number of requests from consumers. For instance, if consumers send 100 requests in aggregate and 86 of these requests are satisfied, then content availability is 86%.

Content retrieval delay is defined as the average time between when a consumer initiates a request and when the consumer receives the first packet of requested content. We calculate content retrieval delay only for satisfied requests. Unsatisfied requests are factored in the *content availability* metric.

In our comparison, we simulate node failures of various intensities. The intensity is defined as the amount of resources that node failure affects. In this paper, we use network bandwidth as the resource that will be unavailable during node failures. When a node failure scenario starts, we set the bandwidth of an arbitrary subset of nodes to zero (i.e., CDN servers, exposure layer nodes, or hidden layer nodes) or disable part of bandwidth of a node (i.e., an origin server).

5.1. Content Availability. In our analysis, we consider the worst-case scenario of node failure; that is, the subset of failed nodes is chosen to cause the largest possible damage to the network for a given failure size. Let $\mathcal{A}$ be the combined bandwidth affected by a node failure event. If $\mathcal{A}$ is equal to (or is larger than) $\lambda\mathcal{D}$, NCDN cannot deliver any content to consumers (i.e., content availability is 0%): the combined bandwidth capacity of exposure layer nodes—or hidden layer nodes—is $\lambda\mathcal{D}$, and therefore a node failure of this size can take an entire layer offline. However, to make all the exposure layer nodes (or hidden layer nodes) offline, $\mathcal{A}$ is expected to have an extremely large value (half of the NCDN nodes are attacked), which is not realistic in our real world. In addition, NCDN is proposed to solve the single-point failure of CDN. How to mitigate the attack targeting half number of the nodes is

out of scope of this paper. Therefore, in the rest of the paper, we consider $\mathscr{A} \leq \lambda\mathscr{D}$.

In NCDN, failure of hidden layer nodes has larger impact on content availability than failure of the same number of exposure layer nodes. This is because failure of a number of exposure layer nodes has no impact on content availability, while if all hidden layer nodes with copies of a particular content fail, that content becomes unavailable. In CDNs, $\mathscr{A}$ is allocated to the origin server. This is because once the origin server is unavailable, it cannot receive new content from producers and, hence, it cannot replicate such content to CDN servers. Therefore, none of the consumers can retrieve content from the system which was going to be uploaded after the origin server fails. If $\mathscr{A}$ is large than the bandwidth of origin server, the extra bandwidth is allocated to CDN servers.

In NCDN, during nodes failure, the combined bandwidth of hidden layer nodes is

$$B_b' = \lambda\mathscr{D} - \mathscr{A}. \tag{4}$$

The hidden layer needs to deliver total $\mathscr{D}$ content to consumers. The portion of such content the hidden layer node has enough bandwidth to respond to is

$$P_{\text{bandwidth}} = \left\{ \begin{array}{ll} \lambda - \dfrac{\mathscr{A}}{\mathscr{D}}, & \text{if } \mathscr{A} > \mathscr{D}(\lambda - 1), \\ 1, & \text{otherwise} \end{array} \right\}. \tag{5}$$

Let the number of failed hidden layer nodes be m_f. We have

$$m_f = \left\lceil \frac{\mathscr{A}}{\lambda\mathscr{D}} \cdot m \right\rceil, \tag{6}$$

and c hidden layer nodes have a copy of a particular content, because each content is replicated c times. The probability that the content is existent among available hidden layer nodes is P_{content}:

$$P_{\text{content}} = \left\{ \begin{array}{ll} 1 - \dfrac{\binom{m_f}{c}}{\binom{m}{c}}, & \text{if } c \leq m_f, \\ 1, & \text{otherwise} \end{array} \right\}. \tag{7}$$

Therefore, content availability in NCDN is

$$P_{\text{NCDN}} = P_{\text{bandwidth}} \cdot P_{\text{content}}. \tag{8}$$

In CDN, the threshold of $\mathscr{A}$ to make the origin server unavailable is B_{origin}. Therefore, if $\mathscr{A} > B_{\text{origin}}$, we consider that the origin server is unavailable ($\mathscr{A} - B_{\text{origin}}$ is allocated to CDN servers). Under this circumstance, CDN servers cannot receive copies of new content, and therefore they can only respond to requests for content that is already stored in CDN servers.

In CDN, we divide the content requested by consumers into two categories: (1) content that is uploaded before a node failure event begins (we refer to this type of content as CAU, standing for *content already updated*) and (2) content that is uploaded during the event of nodes failure—for which the upload might fail (we refer to this type of content as CNU, standing for *content that needs to be updated*). This taxonomy is meaningless in NCDN, because NCDN does not have a single origin server, and therefore content can always be uploaded as long as enough bandwidth is available. Once the origin server fails, consumers might not be able to retrieve content because (1) content is CNU or (2) the CDN servers serving CAU are unreachable. The portion of each type of content is different depending on different applications. In our analysis, we consider the portion of CAU as q. So the portion of CNU is $1 - q$. If the available bandwidth of CDN servers is larger than $q\mathscr{D}$, even with origin server failure, all CAU content could be delivered to consumers. Therefore, the content availability in CDN is

$$P_{\text{cdn}} = \left\{ \begin{array}{ll} 1, & \text{if } \mathscr{A} < B_{\text{origin}}, \\ q, & \text{if } B_{\text{cdn}} - \left(\mathscr{A} - B_{\text{origin}}\right) > q\mathscr{D}, \\ \dfrac{B_{\text{cdn}} - \left(\mathscr{A} - B_{\text{origin}}\right)}{q\mathscr{D}}, & \text{otherwise} \end{array} \right\}. \tag{9}$$

Content availability of NCDN depends on the number of copies in the hidden layer nodes (cf. equation (2)), while the content availability of CDN is independent on the number of copies (as per equation (3)).

5.2. Content Retrieval Delay. In this section, we model the *content retrievable delay* in NCDN and CDN. According to [23], on average, the content retrieval delay to the origin server is three times larger than the delay to CDN servers. Therefore, in our analysis, we set that the round-trip delay from users to the origin server is $3t$ and that to CDN servers is t.

In NCDN, since the nodes are distributed as CDN servers and an exposure layer node could be colocated with a hidden layer node (the delay between two colocated nodes can be neglected), the round-trip delay between users and hidden layer nodes is the same as the delay between users and CDN servers, which is t. We set the timeout as $10t$; if a destination server is unavailable, a client will wait for $10t$ and then establish a connection to another server.

In NCDN, $\mathscr{A}$ is allocated to hidden layer nodes. Therefore, all the exposure layer nodes are available to clients. Once an exposure layer node receives a request from a consumer, the exposure layer node directs the request to a hidden layer node that stores the content and has minimum delay. If the exposure layer node does not receive the response from the hidden layer node after a timeout, the exposure layer node will send probe messages to all the hidden layer nodes that have the replica of the content. The exposure layer node will send the request to the hidden layer

node that has the minimum latency according to the responses of the probe messages. The probability that a hidden layer node is offline is

$$P_{\text{boffline}} = \frac{\mathscr{A}}{\lambda \mathscr{D}}. \tag{10}$$

Therefore, the content retrieval delay in NCDN is

$$D_{\text{NCDN}} = (10t + t) \cdot P_{\text{boffline}} + t \cdot \left(1 - P_{\text{boffline}}\right). \tag{11}$$

In CDN, $\mathscr{A}$ is allocated to the origin server with higher priority. Once all the bandwidth of origin server is depleted, the remaining $\mathscr{A}$ (if available) is allocated to the CDN servers. Therefore, the unavailable bandwidth of CDN servers is

$$\mathscr{A}_{\text{cdn}} = \left\{ \begin{array}{ll} 0, & \text{if} \mathscr{A} \leq B_{\text{origin}}, \\ \mathscr{A} - B_{\text{origin}}, & \text{otherwise} \end{array} \right\}. \tag{12}$$

The probability of a CDN server being offline is

$$P_{\text{cdnoffline}} = \frac{\mathscr{A}_{\text{cdn}}}{B_s}. \tag{13}$$

Therefore, the average content retrieval delay in CDN is

$$D_{\text{cdn}} = (10t + 3t) \cdot P_{\text{cdnoffline}} + t \cdot \left(1 - P_{\text{cdnoffline}}\right). \tag{14}$$

Note that, in this section, we do not consider the delay caused by FIB and PRT. This is because FIB and PRT delay basically are the content lookup (i.e., routing) delay. The current CDN architecture also has the same type of delay. According to [11], the routing delay of NDN is comparable with the current Internet architecture. In addition, optimizing such delay is out of the scope of this paper. However, in the experiment, we use NS3-based ndnSIM simulator that captures FIB and PRT delay as discussed in Section 7. Note that the main goal of this section is to model the delay of CDN and NCDN. Therefore, we consider that the numbers of copies are the same across all the contents. However, in the following, we propose a Reinforcement Learning-based optimization method to optimize the number of copies for different content.

6. Determining the Optimal Number of Copies

In NCDN, the number of content copies affects the content availability (cf. equation (2)). NCDN achieves higher content availability if the number of content copies increases. However, the increasing number of content copies also brings higher costs (e.g., additional bandwidth to transfer the copy and additional storage to store the copy). In this section, we introduce an RL algorithm to optimize the number of copies for different type of contents.

In RL, the agent observes the state of the environment and, based on this state/observation, takes an action as illustrated in Figure 4. The ultimate goal is to compute a policy—a mapping between the environment states and actions—that maximizes expected reward. RL can be viewed as a stochastic optimization solution for solving Markov Decision Processes (MDPs) [24], when the MDP is not known.

We consider that the entire NCDN system is an RL environment. For content duplication optimization problem, improving the long-term sum of rewards is more important than a single immediate reward. Therefore, in the following, we define the "state," "action," and "reward," respectively. The state is defined as a combination of the current resource utilization (e.g., bandwidth), available resources, and the content popularity. As we discussed in Section 4.1, we consider that the popularity follows the Zipf distribution, where a small number of Web pages are requested a disproportionated number of times. The action is to determine increased or decreased number of contents. Because the objective is to optimize the number of duplications, our solution interacts with the NDN system and monitors the overall "hitting rate" (found the content). This behavior is inherent in RL. The RL agent has the advantage of not requiring expert labels or knowledge and instead the ability to learn directly from its own interaction with the world.

We use Q-learning [25] as our model-free approach to build our optimization agent. It updates Q value that denotes value of performing action a in state s.

$$Q(s_t, a_t) \longleftarrow (1 - \alpha) \cdot Q(s_{t-1}, a_{t-1}) + \alpha \cdot \left(r_t + \gamma \cdot \max_a Q(s_{t+1}, a)\right), \tag{15}$$

where $Q(s_{t-1}, a_{t-1})$ is the old value, α is the learning rate, r_t is the reward, γ is the discount factor, and $\max_a Q(s_{t+1}, a)$ is the estimated optimal future value. In particular, action a indicates the following actions: adding one more copy, removing one copy, and keeping the current number of copies. State s represents the available resources, attacking bandwidth, the content availability to end-users, content retrieval delay, and content popularity modeled by Zipf distribution. Reward r indicates adding 1 reward if all the requested contents are available and adding minus the percent of missing content multiply 1000 if part of the requested contents cannot be satisfied (as the penalty).

7. Evaluation

In order to validate our analysis, in this section, we assess content availability and content retrieval delay of NCDN via simulations. We simulate NCDN using a modified version of ndnSIM [15]—an NS-3-based network simulator. In the following, we first present the setup of the simulation. Then we show the comparison of *content availability* and *content retrieval delay* in NCDN and CDN, respectively.

7.1. Evaluation Setup. In our simulation, we measure content availability and per-packet content retrieval delay. Therefore, without loss of generality, in our simulations, we use 6000 consumers and 50 producers. The data rate of each producer is 20 Kb/s. Each consumer downloads content at the same rate. Therefore, $\mathscr{U}$ = 1 Mb/s, and $\mathscr{D}$ = 120 Mb/s. We set the overprovisioning factor λ to 150%, which is a reasonable value in practice [26]. According to [17], Akamai controls CDN servers in 3000 locations around the world. In our paper, we consider a CDN server (or a NCDN node) as a

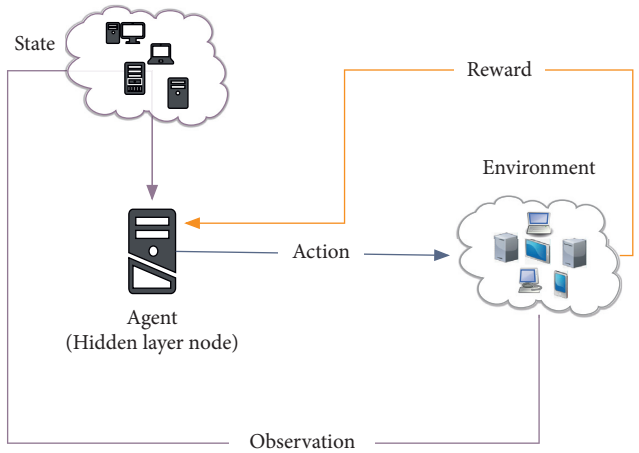

Figure 4: RL environment example. By observing the state of the environment, the RL agent makes actions (i.e., increase or decrease duplications) for which he receives rewards as revenues. The agent's goal is to find the optimal number of duplicates.

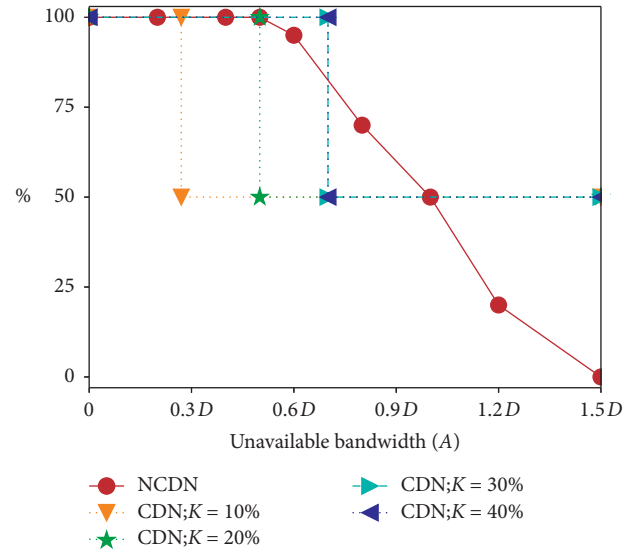

Figure 5: Content availability.

collection of hosts (e.g., a data center). Therefore, we have $m = 3000$. In our analysis, we vary k from 10% to 20%, 30%, and 40%. With these parameters, the optimal number of copies in NCDN is 15 ($c = 15$). For CDN, we set $q = 50\%$. Based on the above values, in the following, we show our comparison of NCDN and CDN.

7.2. Content Availability. Figure 5 shows the comparison of content availability in NCDN and CDN based on the model presented in Section 5. The content availability of CDN (with different k) has tumbling decreasing from 100% to 50%. This is because $\mathscr{A}$ is preferentially allocated to the origin server. With different k, the bandwidth capacity of the origin server is different. Once all bandwidth of origin server is depleted, it cannot replicate copies of CNU content. Although CDN servers are online and have enough bandwidth to respond to the requests from consumers, such CDN servers only have the CAU content. In our analysis, $q = 0.5$, and it means that the minimum content availability CDN provides (if it has enough bandwidth to do so) is 50%. However, the value of q changes in different applications. In the applications, such as video streaming or microblogging services, q could be even smaller, which leads to smaller content availability of CDN in such applications. In NCDN, $\mathscr{A}$ is preferentially allocated to the hidden layer nodes. When $\mathscr{A}$ is less than $0.5\mathscr{D}$, although some of the hidden layer nodes fail, the other hidden layer nodes still have enough bandwidth to serve all the requests. In addition, NCDN always keeps the optimal number of copies, which guarantees that content existed in the network when a smaller number of hidden layer nodes fail. As $\mathscr{A}$ increases, the content availability of NCDN reduces. As shown in Figure 5, when $\mathscr{A}$ is less than $0.5\mathscr{D}$, NCDN achieves better content availability than the CDN with $k = 0.1$ and $k = 0.2$. We would argue that, considering that the CDN consists of 3,000 CDN servers, it is impractical that k is larger than 0.2.

In order to show the validity of our model, we also run simulations to assess the amount of responding content in NCDN when the value of $\mathscr{A}$ is equal to $0.2\mathscr{D}$, $0.6\mathscr{D}$, and $\mathscr{D}$, respectively. We start the event of node failure (to deplete the bandwidth of hidden layer nodes) 5 seconds from the beginning of the simulation and bring all nodes back online 15 seconds from the beginning of the simulation. The length of the entire simulation is 30 seconds. This is sufficient to allow the network to fully recover. As shown in Figure 6, when $\mathscr{A}$ equals $0.2\mathscr{D}$, all the requests can be satisfied. This is because, under this node failure scenario, the remaining bandwidth of hidden layer nodes is larger than $\mathscr{D}$, which means that all the requested content is able to be delivered to consumers. In the scenario where $\mathscr{A}$ equals $0.6\mathscr{D}$, during the node failure event, the remaining bandwidth of hidden layer cannot deliver all $\mathscr{D}$ content to consumers. Therefore, around 90% of contents are responded to. The unsatisfied requests are kept in the hidden layer. Once the node failure event stops (at 15 seconds), the hidden layer nodes have enough bandwidth to respond to such unsatisfied requests. This is the reason why, between 15 and 19 seconds, the amount of content responded to is larger than the consumer requested. After $19'$, all the unsatisfied requests (because of the node failure) are responded to consumers, and the behavior of network becomes similar to the behavior before the node failure event. When $\mathscr{A}$ equals $\mathscr{D}$, as shown in the green curve in Figure 6, the network needs more time between 15 seconds and 27 seconds to send the in-responded content. This figure also shows that the content availability of NCDN during the node failure event (between 5 seconds and 15 seconds) is 100%, 90%, and 50%, when $\mathscr{A}$ equals $0.2\mathscr{D}$, $0.6\mathscr{D}$, and $\mathscr{D}$, respectively. These values follow the results shown in Figure 6.

7.3. Content Retrieval Delay. Figure 7 shows the comparison of content retrieval delay in NCDN and CDN ($k = 0.1$, $k = 0.2$, $k = 0.3$, and $k = 0.4$). As shown in this figure, the content retrieval delay in NCDN is larger than the delay in CDN under different node failure scenarios. This is because,

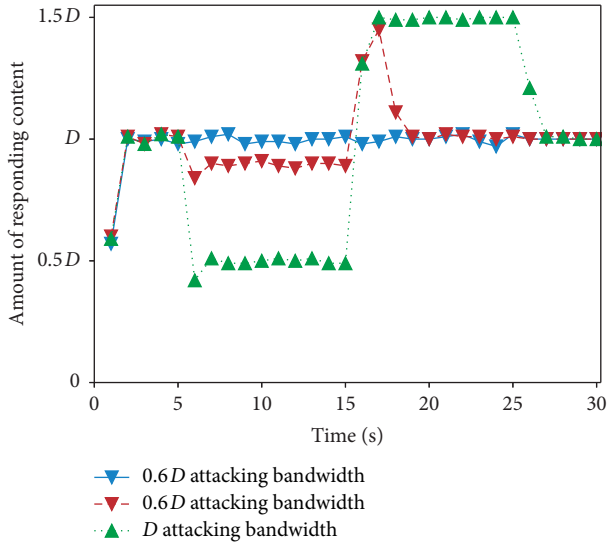

Figure 6: *Amount of responding content*. Because, during nodes failure, some requests are not satisfied, after nodes go online, the amount of responding content is larger than $\mathscr{D}$.

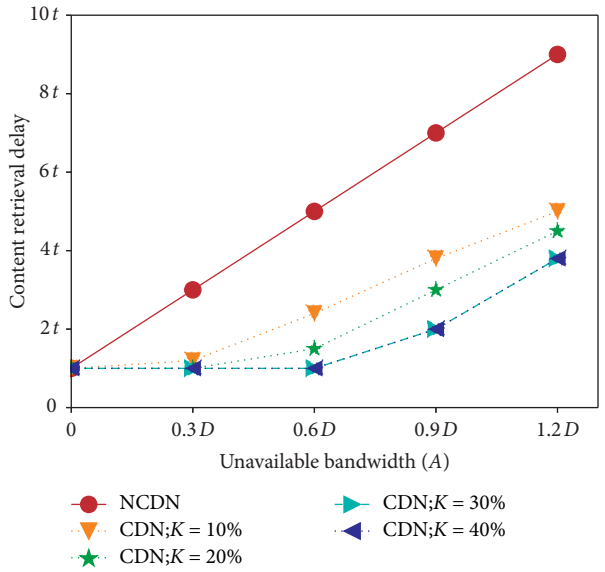

Figure 7: Content retrieval delay based on our model.

with the same amount of $\mathscr{A}$, more hidden layer nodes fail compared with CDN servers. Note that we only plot the delay of requests that have been satisfied during a node failure event.

Figure 8 shows the simulation result of content retrieval delay in NCDN. The behavior follows our model. According to these two figures, we conclude that, (1) under the situation without nodes failure, the content retrieval delays of NCDN and CDN are similar, and, (2) under the situation with nodes failures, the content retrieval delay of NCDN is larger than the content retrieval delay of CDN. The delay of NCDN increases linearly as $\mathscr{A}$ increases.

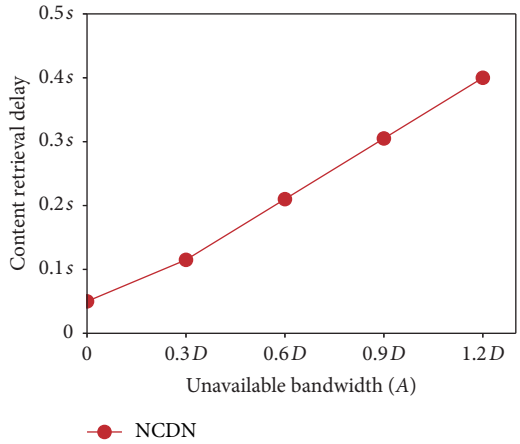

Figure 8: Content retrieval delay based on simulations.

8. Related Work

Peer-to-peer (P2P) systems, such as Chord [27] and Kademlia [28], are designed around the notion of equal functionally equivalent peer nodes. Nodes in the system are organized using distributed hash table (DHT), which are resilient to node failure. However, Ding et al. [29] demonstrated that the routability of DHT-based P2P system is severely hampered when a large number of nodes fail. In addition, with DHT-based P2P system (e.g., Chord), end-users need to perform, on average, $\log N$ lookups to route messages to a targeted node, where N is the total number of nodes in such system. In contrast, retrieving content in NCDN requires only two lookups (one to identify an exposure layer node, and another—performed by the exposure layer node—to identify a suitable hidden layer node). Therefore, users are expected to experience a larger delay on conventional P2P systems compared to NCDN.

CDNs [8, 30, 31] are designed to improve network performance (i.e., increase bandwidth, reduce delay to end-users, and increase content availability) by moving content as close as possible to the intended consumers. If a consumer requests content from a failed CDN server, the consumer will resend the same request to the origin server. To reduce network delays, Biliris et al. [23] argue that the consumer should instead resend the request to the next closest CDN server. This mechanism allows implementation of CDNs with a low delay to end-users even if a significant subset of CDNs servers fail at the cost of increased complexity.

Hybrid CDN-P2P [32–34] is a combined architecture that combines the scalability P2P networks with the reliability of CDNs. However, similar to traditional CDNs, CDN-P2P architectures rely on the origin server to distribute and replicate content to their various nodes. Further, with CDN-P2P, the origin server must undertake additional tasks and is a single point of failure of these architectures.

NDN [11] is a network architecture based on the concept of Content-Centric Networking [4]. In NDN, data packets are identified by name rather than by the host that distributes them. This substantially simplifies content distribution, caching, and replication. NCDN leverages this property of NDN to achieve the same benefits. However,

there are several important differences between NCDN and NDN. First, NCDN puts a strong emphasis on resilience to node failures. All components of NCDN are designed to maximize the probability that content is available even when a substantial subset of the network becomes unreachable. Further, routing and forwarding in NCDN are substantially simpler than those in NDN because NCDN is a two-layer architecture, while in NDN consumers and the data they request can be separated by an arbitrary number of nodes. Finally, the number of nodes in NCDN is significantly smaller than the expected number of hosts and routers in NDN, in part because NCDN abstracts collections of hosts as a single node.

9. Conclusions and Future Work

In this paper, we present NCDN, a novel highly distributed system for large-scale delivery of content and services. Our evaluation shows that NCDN is able to provide higher content availability than traditional CDNs under nodes failure events. Further, our simulations show that NCDN is able to quickly recover from large node failures.

Our work represents just the first steps towards a comprehensive evaluation of NCDN. For instance, this paper leaves optimal replication strategies to future work. Further, as with NDN routers, NCDN exposure layer nodes can opportunistically cache arbitrary content. We will address optimal caching strategies and their benefits on performance and availability in future work.

Data Availability

The data can be found at https://www.akamai.com.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was jointly supported by the Social Science Project of Shandong Province (Project no. 18CHLJ09), the National Social Science Foundation of China (Project no. 20CDCJ01), the National Natural Science Foundation of China (Project no. 71801142), and the Key R&D Project of Shandong Province (Project no. 2019GSF108222).

References

[1] L. Cavedon, C. Kruegel, and G. Vigna, "Are bgp routers open to attack? an experiment," in *Open Research Problems in Network Security*, pp. 88–103, Springer, Berlin, Germany, 2011.

[2] J. Cowie, A. Popescu, and T. Underwood, "Impact of Hurricane Katrina on Internet Infrastructure,"Report, Gujarat, India.

[3] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[4] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7.

[5] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 59–62, 2006.

[6] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang, "Practical, real-time centralized control for cdn-based live video delivery," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 311–324, London, UK, July 2015.

[7] C. Natalino, A. Yayimli, L. Wosinska, and M. Furdek, "Infrastructure upgrade framework for content delivery networks robust to targeted attacks," *Optical Switching and Networking*, vol. 31, pp. 202–210, 2019.

[8] A.-M. K. Pathan and R. Buyya, "A Taxonomy and Survey of Content Delivery Networks," Technical Report, University of Melbourne, Melbourne, Australia, 2007.

[9] D. Ding, M. Conti, and R. Figueiredo, "Wide-scale internet disconnection: impact and recovery on social-based p2p overlays," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 4, pp. 734–747, 2018.

[10] D. Ding, M. Conti, and R. Figueiredo, "Sand: social-aware, network-failure resilient, and decentralized microblogging system," *Future Generation Computer Systems*, vol. 93, pp. 637–650, 2019.

[11] L. Zhang, D. Estrin, J. Burke et al., "Named Data Networking (Ndn) Project,"Relatório Técnico NDN-0001, Palo Alto, CA, USA.

[12] D. Ding, K. Jeong, S. Xing, M. Conti, R. Figueiredo, and F. Liu, "Send: a social network friendship enhanced decentralized system to circumvent censorships," *IEEE Transactions on Services Computing*.

[13] H. Li, K. Ota, and M. Dong, "Learning iot in edge: deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[14] A. Haj-Ali, N. K. Ahmed, T. Willke, J. Gonzalez, K. Asanovic, and I. Stoica, "A View on Deep Reinforcement Learning in System Optimization," 2019, http://arxiv.org/abs/1908.01275.

[15] A. Afanasyev, I. Moiseenko, L. Zhang et al., "Ndnsim: Ndn Simulator for Ns-3,"Technical Report, Los Angeles, CA, USA.

[16] D. Ding, M. Conti, and A. Solanas, "A smart health application and its related privacy issues," in *Proceedings of the 2016 Smart City Security and Privacy Workshop (SCSP-W)*, pp. 1–5, IEEE, Vienna, Austria, April 2016.

[17] Akamai for Responsive Web Design, 2017.

[18] L. A. Adamic and B. A. Huberman, "Zipf's law and the internet," *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.

[19] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in *Proceedings of the IEEE INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 126–134, IEEE, San Diego, CA, USA, April 1999.

[20] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, pp. 41–54, ACM, Berlin, Germany, August 2004.

[21] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pp. 15–28, ACM, San Diego,MA, USA, October 2007.

[22] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, "Dynamic provisioning of multi-tier internet applications," in *Proceedings Autonomic Computing, 2005. ICAC 2005. . Second International Conference*, pp. 217–228, IEEE, Seattle, WA, USA, June 2005.

[23] A. Biliris, C. Cranor, F. Douglis et al., "CDN brokering," *Computer Communications*, vol. 25, no. 4, pp. 393–402, 2002.

[24] R. Bellman, "A markovian decision process," *Indiana University Mathematics Journal*, vol. 6, no. 4, pp. 679–684, 1957.

[25] C. J. Watkins and P. Dayan, *Q-learning, Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[26] Z. Duan, Z.-L. Zhang, and Y. T. Hou, "Service overlay networks: slas, qos, and bandwidth provisioning," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 6, pp. 870–883, 2003.

[27] I. Stoica, R. Morris, D. Liben-Nowell et al., "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[28] P. Maymounkov and D. Mazieres, "Kademlia: a peer-to-peer information system based on the xor metric," in *Proceedings of the International Workshop on Peer-To-Peer Systems*, pp. 53–65, Springer, San Jose, CA, USA, April 2002.

[29] D. Ding, M. Conti, and R. Figueiredo, "Impact of country-scale internet disconnection on structured and social p2p overlays," in *Proceedings of the 2015 IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–9, IEEE, Boston, MA, USA, June 2015.

[30] G. Pallis and A. Vakali, "Insight and perspectives for content delivery networks," *Communications of the ACM*, vol. 49, no. 1, pp. 101–106, 2006.

[31] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Computing*, vol. 7, no. 6, pp. 68–74, 2003.

[32] H. Yin, X. Liu, T. Zhan et al., "Design and deployment of a hybrid cdn-p2p system for live video streaming: experiences with livesky," in *Proceedings of the 17th ACM International Conference on Multimedia*, pp. 25–34, ACM, Nice, France, October 2009.

[33] C. Huang, A. Wang, J. Li, and K. W. Ross, "Understanding hybrid cdn-p2p: why limelight needs its own red swoosh," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 75–80, ACM, Braunschweig, Germany, May 2008.

[34] S. Seyyedi and B. Akbari, "Hybrid cdn-p2p architectures for live video streaming: comparative study of connected and unconnected meshes," in *Proceedings of the Computer Networks and Distributed Systems (CNDS), 2011 International Symposium*, pp. 175–180, IEEE, Tehran, Iran, December 2011.