future internet

MDPI

Article

ICN-Based Enhanced Content Delivery for CDN

Lei Gao [1,2] and Xiaoyong Zhu [1,2,*]

[1] National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, No. 21, North Fourth Ring Road, Haidian District, Beijing 100190, China; gaol@dsp.ac.cn
[2] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, No. 19(A), Yuquan Road, Shijingshan District, Beijing 100049, China
* Correspondence: zhuxy@dsp.ac.cn; Tel.: +86-131-2116-832

Abstract: With the rapid growth of internet traffic, the traditional host-to-host TCP/IP architecture is subject to many service limitations faced with content-oriented applications. Various novel network architectures have been proposed to solve these limitations, among which Information-Centric Networking (ICN) is one of the most prominent. ICN features the decoupling of content (service) from the physical devices storing (providing) it through location-independent naming, and offers inherent enhancement to network performance, such as multicast and in-network caching. ICN in-network caching has been extensively studied, and we believe that it may also be the main incentive for ISPs to deploy ICN. A CDN (content delivery network) is a typical content-oriented network paradigm that aims to provide the fast delivery of content. In this paper, we leverage the advantages of the in-network caching of ICN to enhance the content delivery efficiency of CDN by integrating ICN as a service. First, we present our design of a content delivery network enhanced with ICN, called IECDN. Additionally, we formulate a mathematical model to optimize the performance of our proposed design and conduct a series of evaluations. The results indicate that our proposed design provides significant performance gains while reducing bandwidth consumption and shows better resilience to traffic surge.

Keywords: CDN; ICN; in-network caching; multi-commodity flow problem

Citation: Gao, L.; Zhu, X. ICN-Based Enhanced Content Delivery for CDN. *Future Internet* **2023**, *15*, 390. https://doi.org/10.3390/fi15120390

Academic Editor: Franco Davoli

Received: 22 September 2023
Revised: 28 November 2023
Accepted: 28 November 2023
Published: 30 November 2023

1. Introduction

For the past few decades, the development of the internet has always adhered to the tenet of connecting a computer to another by establishing host-to-host communication. However, with the evolving user interests and requirements comes the boom of novel applications, such as Instagram and TikTok, which puts the nature of the internet under question. According to Cisco's 2020 annual report [1], video and other applications are in enormous demand in today's home, and there will be significant bandwidth demands with the video application requirements in the future. This simply means the internet is shifting towards a content-oriented delivery network of video files. Many content providers use content delivery networks (CDNs) as a patch to the current IP network infrastructure in order to enhance the availability of content. For example, Netflix has used services from both Akamai and Limelight to build an ISP-independent content delivery network [2]. A CDN caches content on edge servers in point-of-presence (POP) locations that are close to end users, thereby reducing access latency. When the user requests a file via a special domain name, the DNS routes the request to the optimal POP location, which is typically determined by geographic proximity. If no edge servers in the POP have cached the file, the file is retrieved from the origin server which is comparatively time consuming.

Meanwhile, in the context of future internet study, many studies focus on a new networking paradigm called Information-Centric Networking (ICN). Contrary to the current IP network, ICN emphasizes the content itself rather than its location. The shift from a host-centric to content-centric communication paradigm facilitates the communication

between networking parties by the identifier of the content instead of the address of the host holding a copy of the content, which provides solutions to many issues faced by the IP infrastructure today. The design specifics of ICN paradigm have been proposed by many studies. Some of these research works advocate a clean-slate architecture, such as Content-Centric Networking (CCN) [3], Named Data Networking (NDN) [4] and Publish Subscribe Internet Technology (PURSUIT) [5]. However, many others acknowledge the unbearable cost of kicking IP networks out of the game and settle for the coexistence with existing IP networks to achieve incremental deployments, such as Data-Oriented Network Architecture (DONA) [6], the Network of Information (NetInf) [7], MobilityFirst [8] and SEANet [9]. Despite the differences in design, all of these candidate ICN architectures share the fundamental principle of ubiquitous in-network caching.

Since the very beginning of the proposition of ICN paradigm, numerous research fields have endeavored to leverage its striking feature of in-network caching to enhance their performance, such as Internet of Things (IoT) [10], vehicular networks [11], 5G networks [12] and IPFS [13]. The benefits that the ICN in-network caching brings are particularly similar to those of CDN [14]. Meanwhile, the effectiveness of in-network caching has already been proven by the commercial success of CDN. The ICN content-centric communication paradigm meets the CDN content-oriented service requirements. CDN has already been incorporating technologies to emancipate itself from the bounds of host-centric communication paradigm of IP networks, such as GeoDNS [15] and Anycast [16]. In that sense, we envision a future CDN architecture with the support of ICN infrastructure. We incorporate the ICN in-network caching as a service for CDN to further improve the content availability. The main contributions of this paper can be summarized as follows:

- We design an enhanced content delivery network architecture by integrating the ICN in-network caching as a service. Furthermore, we provide details on the mechanisms for enabling collaboration between different communication paradigms within these two networks.
- We quantify the optimization goals of content delivery network architecture by formulating a mathematical model derived from the Multi-Commodity Flow (MCF) problem, which jointly considers the QoS demand and bandwidth capacity constraint.
- Based on the optimization model, we use a real-world network topology to gauge the performance improvements of our proposed design by conducting a series of experiments. The results demonstrate our design's benefits in terms of reduced delivery latency, decreased network bandwidth consumption and better resilience to traffic surge.

The rest of this paper is organized as follows. In Section 2, we first introduce the background of the ICN cache and content discovery mechanism, and then review the related research, which jointly studies ICN and CDN. In Section 3, we present our design of the ICN-enhanced CDN architecture, called IECDN. In Section 4, we describe the formulation of the optimization model in detail. Then we conduct a series of experiments and analyze the results in Section 5. Finally, the conclusion is drawn in Section 6.

2. Background and Related Work

2.1. ICN Cache in a Nutshell

There have been early attempts [17–19] to develop a ubiquitous caching system within the IP networks to avoid repeated transmission. Nevertheless, the emergence of ICN has raised both the intensity and breadth of research on in-network caching to a new level. There is a vast literature addressing the challenges of content caching in ICN, most of which focus on the cache placement strategy [20–22], cache replacement policy [23] and cache capacity allocation [24]. Three major features differentiate the ICN cache from the existing cache system. First, ICN caching is transparent to upper-level applications and exists as an underlying service, making it more efficient and available to different types of traffic. Second, in ICN, the cache can be equipped on any network node. This ubiquity allows for content to be cached closer to the end users, reducing latency and improving content

delivery. Third, the objects cached in most traditional cache systems are generally based on files. However, to meet the line-speed constraints of ICN routers [25], ICN employs the smaller unit of chunk as its basic caching unit. As a side effect of this, many analysis reference models of traditional cache become outdated to ICN.

2.2. Name Resolution Mechanism

A primary concept of ICN is naming data independently from their physical location. Each Named Data Object (NDO) in ICN is identified with a unique name, which is not directly usable for forwarding or routing within the current IP network. Consequently, the issue of locating and discovering NDOs using their location-independent names has arisen. To address this challenge, a Name Resolution Service (NRS) is required, which provides the mapping between Identifiers (IDs) and Network Addresses (NAs). A NRS in ICN is defined as the service that provides the name resolution function for translating an object name into some other information, such as a locator, another name, metadata, next-hop info, etc., that is used for forwarding the object request [26]. There are generally three kinds of name resolution service approaches:

1. Explicit name resolution approach: This approach is also referred to as a standalone name resolution approach. In this approach, the name resolution process and content request routing process are decoupled. The NRS first returns to the client the locators of the content and then it is used by the underlying network as the identifier to route the content request. This approach is employed by many ICN architectures, such as PURSUIT [5], DONA [6], NetInf [7] and MobilityFirst [8]. For instance, in the MobilityFirst ICN architecture, the infrastructure of NRS is called the Global Name Resolution Service (GNRS). It is designed as a massively scalable distributed system which provides the dynamic binding between a content's Global Unique ID (GUID) and its NA.
2. Name-based routing approach: Some ICN architectures employ a name-based routing approach which integrates the name resolution process with content request routing. In this approach, the name is used as the basis for routing and forwarding the content request. When a request is initiated, the name is used to search for the NDO in the NRS cache. The NDO is forwarded back to the requester along the same path used for request routing but in the opposite direction. A typical example is NDN [4].
3. Hybrid approach: The hybrid approach entails both an explicit name resolution approach and name-based routing approach to complement each other. For example, if the name-based approach fails at a certain router, the explicit name resolution approach can be utilized as an alternative, and vice versa.

2.3. Related Work

CDN and ICN are two network paradigms proposed to enhance the content distribution efficiency in the presence of service limitations in current IP networks. Previous studies regarding the discussion of these two paradigms can be classified into two broad classes:

1. Comparative studies: These studies have attempted to conduct comparative analyses of these two paradigms to evaluate their respective performance bounds. In their research, Ma Ge et al. [27] discussed the differences between CDN and NDN in terms of distribution efficiency, protocol overhead, security, robustness, and cost, and testbed research was conducted to compare the data transmission efficiency of the two paradigms, with the conclusion being that NDN outperforms CDN under the same network topology and cache storage budget. In [28], Ma Ge et al. constructed comparable testbeds on cloud computing platforms for CDN and CCN separately to systematically compare their performance metrics in content delivery capability. This research found that CCN is comparable to CDN, even though it introduces overhead in content chunk decoding and restructuring. M. Mangili et al. [29] studied the performance bounds of these paradigms using optimization models and proposed a comparative model to gauge the performance difference of a CDN with respect

to CCN. The results showed that, under the same total caching storage, CDN can have slightly better performance than CCN. In [30], real-world experiments were conducted to compare the performance of NDN and two leading CDNs (Akamai and Fastly). The results showed that although NDN can provide satisfactory service, it lags behind CDN due to the hardware infrastructure and software/protocol immaturity. NDN outperforms in terms of server load balancing and failure resiliency enabled by ubiquitous caching. The comparative studies have shown that ICN does not necessarily outperform CDN in terms of content distribution efficiency; however, it is favorable in terms of lower management cost and better resiliency. Rather than being two separate models, these two can be explored to complement each other.

2. Hybrid studies: Some prior research proposed to integrate CDN and ICN using hybrid approaches to achieve better performance. Ref. [31] proposed a named content delivery network called nCDN, which embeds NDN into the existing CDN framework by setting up NDN running over UDP/TCP to simplify the implementation and improve efficiency. In nCDN, ICN is only in charge of request routing and content delivery. Similarly, in [32], the authors borrow the ICN name-based routing idea and integrate it into the CDN architecture with a centralized routing engine to fully exploit the ISP underlay network infrastructure. In [33], Benkacem et al. integrate NDN with CDN as two network slices with the technology of Network Function Virtualization (NFV). In this architecture, NDN serves as a service slice designated for request routing and in-network caching over a specific region. The CDN slice is considered the content publisher of the NDN slice. Upon the first request for a particular content, the content is retrieved from the CDN service slice and simultaneously published to the NDN slice. Instead of building a CDN using NDN by the direct borrowing of the NDN network, the authors in [34] proposed another CDN-NDN architecture called iCDN, with two major mechanisms: first, to fully utilize in-networking caching, iCDN builds a cache hierarchy with the CDN full-mesh topology to exploit on-path and off-path caches and cuts the forwarding plane's dependency on routing information; second, iCDN introduces a new forwarding strategy for enabling the forwarding plane to make efficient decisions. J. Chen et al. [35] designed a CDN architecture over MobilityFirst in the application layer so that minor modification is needed for existing protocol stack. Meanwhile, it uses a self-certifying naming schema to enable low-cost, efficient content validation and uses proactive caching to further improve the performance of CDN. The main features and discussions regarding the above hybrid approaches are compared and summarized in Table 1. To conclude, although the above hybrid studies differ in their approaches, they all clearly indicate that by integrating ICN into the CDN architecture, the content distribution efficiency of CDN can be significantly improved.

When incorporating ICN into CDN, addressing the challenges of communication paradigm translation between the two is an important research area. The translation operations are typically executed by gateways or proxies that support both paradigms. The first is to convert HTTP semantics into a ICN naming convention. In a hierarchical ICN naming convention like NDN/CCN, the most common approach [36–38] is to construct a hierarchical name structure by extracting information from the HTTP URL and HTTP header. In [36], the sub-level of the HTTP domain name is split and reversely appended in the NDN name, followed by the version, segment, and other information. This approach improves the NDN routing capability with route aggregation. The strategy in [38] puts a default prefix at the beginning of the CCN name, HTTP protocol, request method, and request domain, and the request parameters are hierarchically appended.In a flat naming convention like MobilityFirst, the key to translation is generating a globally unique ID that identifies the content/host [35,39]. In [35], the authors use a public key generated by the Elliptic Curve Cryptography (ECC) algorithm as the content ID. Secondly, regarding the different HTTP request methods, primarily GET and POST, the GET method can naturally correspond to the ICN pull-based interest packet, but the POST method is more complex.

In [37,38], the proposed method is for the gateway/proxy to save data upon receiving a POST, then sends a special interest to the ICN terminal. This prompts the ICN terminal to send an echo interest to the gateway and pull the data, thus achieving a push-like effect.

In the integration methods discussed above, using overlay or network slice techniques can facilitate the fast deployment of ICN. However, they also introduce complexities in management and incur overhead associated with network abstraction. Inspired by the above research, we propose our ICN-enhanced CDN architecture, called IECDN. The ubiquitous caching capabilities of ICN can be a compelling incentive for ISP to embrace its deployment. In light of this, this paper envisions a scenario where ICN is employed as an infrastructure service, providing network services through dual-stack gateways. To ensure a cost-effective deployment of ICN, we propose the introduction of an IP-compatible ICN paradigm. This strategic combination of infrastructure deployment and an IP-compatible protocol stack aims to address the challenges posed by ICN deployment, presenting a balanced approach that capitalizes on the strengths of ICN while mitigating the associated complexities.

Table 1. Comparison of hybrid approaches in related work.

Approach	ICN Paradigm	Integration Method	Discussions
nCDN [31]	NDN	An overlay method by setting up NDN running over UDP/TCP	Provide better QoS, reliability and scalability especially in dynamic network
R-iCDN [32]	Not mentioned but name-based routing	Integrate a centralized content routing engine in CDN, using ICN name-based routing	Improve CDN sub-optimal routing in ISP networks
CDN with NDN slice [33]	NDN	Integrate NDN as a network slice	Shorten delivery time and reduce traffic load by leveraging ICN for regional content distribution
iCDN [34]	NDN	Build CDN in NDN's logic using stateful forwarding plane and in-network caches	Using NDN technology on the CDN full-mesh topology, shorter delay and less traffic load is achieved, FIB explosion is avoided
CDN over MobilityFirst [35]	MobilityFirst	Build CDN on top of MobilityFirst	Combine the CDN application awareness and ICN efficient network forwarding
IECDN proposed in this paper	IP-compatible and ID-based protocol	ICN infrastructure service layer with translation gateways	Show reduced latency, less overall bandwidth consumption and better resilience to traffic surge

3. Proposed Design

In this section, we first clarify our motivation to utilize the ICN functionalities to enhance CDN and present the general description of the architecture design and its key components. Next, we explain the working mechanism of the architecture by presenting the workflow of a content request and delivery process. Finally, we introduce the details of our solution to several issues in converting from the HTTP paradigm to the ICN paradigm.

3.1. Motivation

Over the past few decades, CDN has evolved to demonstrate remarkable stability and high service performance. However, the inherent design of CDN often relies on a centralized set of content servers. On the contrary, ICN emphasizes more lightweight and ubiquitous caching. This motivation section explores the potential advantages of adopting ICN to complement the CDN architecture.

1. Emphasis on lightweight and ubiquitous caching in ICN: ICN introduces a paradigm shift by emphasizing the lightweight nature of caching and embracing heterogeneous and ubiquitous caching capabilities. ICN nodes can seamlessly join and leave the network, offering great scalability and lower management costs. This departure from centralized caching servers brings better flexibility and failure resiliency.
2. Leveraging ICN for cost-efficient content delivery: One of the primary costs of CDN is from bandwidth consumption and maintenance. The ubiquitous caching capability of ICN allows for the storage of content replicas within the network at any place. Utilizing idle resources of ICN nodes, such as bandwidth and storage, to provide content service can reduce CDN deployment costs and enhance system scalability.
3. Benefits of the ICN cache relative to the CDN HTTP-based cache: In contrast to the CDN HTTP-based caching at the application level, the ICN cache operates at the network layer. The control overhead is minimized, involving only a few bits in the data packet for cache management. The ICN router can achieve line-speed caching operation. Furthermore, the ability of ICN to split content files into data chunks facilitates load balancing by distributing requests for locally popular content across multiple nodes.

3.2. Design Overview

An overview of our proposed design is illustrated in Figure 1. The major modification to CDN in our architecture is that we integrate an ICN service layer with routers supporting ICN paradigms, which mainly involves in-network caching and naming service. These routers constitute a large distributed caching entity of the ICN service layer. Moreover, Ingress Gateways (IGs) and Egress Gateways (EGs) are respectively deployed at the entry and exit points of the ICN service layer. These gateways function as both HTTP proxy and protocol conversion components. To be specific, IG can be viewed as an HTTP proxy, to which HTTP requests for content are redirected, and as a protocol conversion component which translates an HTTP request into an ICN chunk request so as to discover and retrieve content from the ICN service layer. EG receives HTTP content responses from the internet content provider (origin server) and publishes this content to the ICN service layer, which subsequently performs in-network caching in compliance with certain caching policies. A detailed process of the content request and delivery will be elaborated on in Section 3.3.

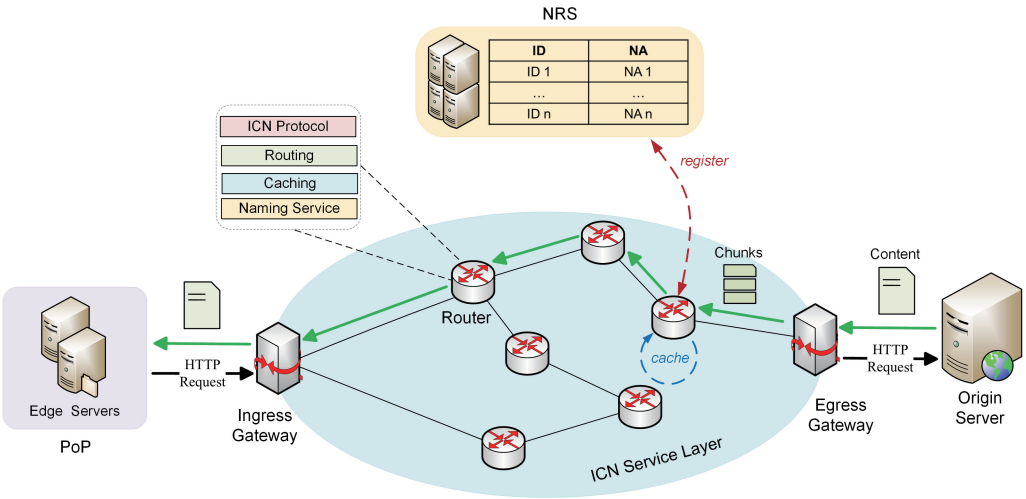

Figure 1. An overview of content delivery network enhanced with ICN (IECDN).

A standalone NRS system is deployed to resolve IDs to NAs. When a router caches a chunk, it simultaneously registers to NRS the value of its physical device NA with the key of chunk ID. Afterwards, this chunk can be discovered and retrieved with this ID: first, an ID resolution request is submitted to the NRS, and NRS returns a resolution response that comprises a list of chunk replica NAs, then, a replica selection mechanism is adopted to

select a suitable chunk replica and triggers the download. In our context, the nearest replica selection mechanism [40] is adopted since it is commonly used in related work. Finally, the subsequent transmission is conducted through NA routing.

Content is cached at the granularity of a file in a CDN, whereas ICN performs caching at the granularity of a chunk. For the purpose of this discussion, we use the terms "content" and "file" interchangeably to refer to data within a CDN network, and the term "chunk" within an ICN network. Further information regarding this issue is discussed in Section 3.4.

3.3. Workflow

Hereafter, we elaborate on our proposed mechanisms for enabling seamless collaboration between CDN and ICN to achieve efficient content delivery as illustrated in Figure 2.

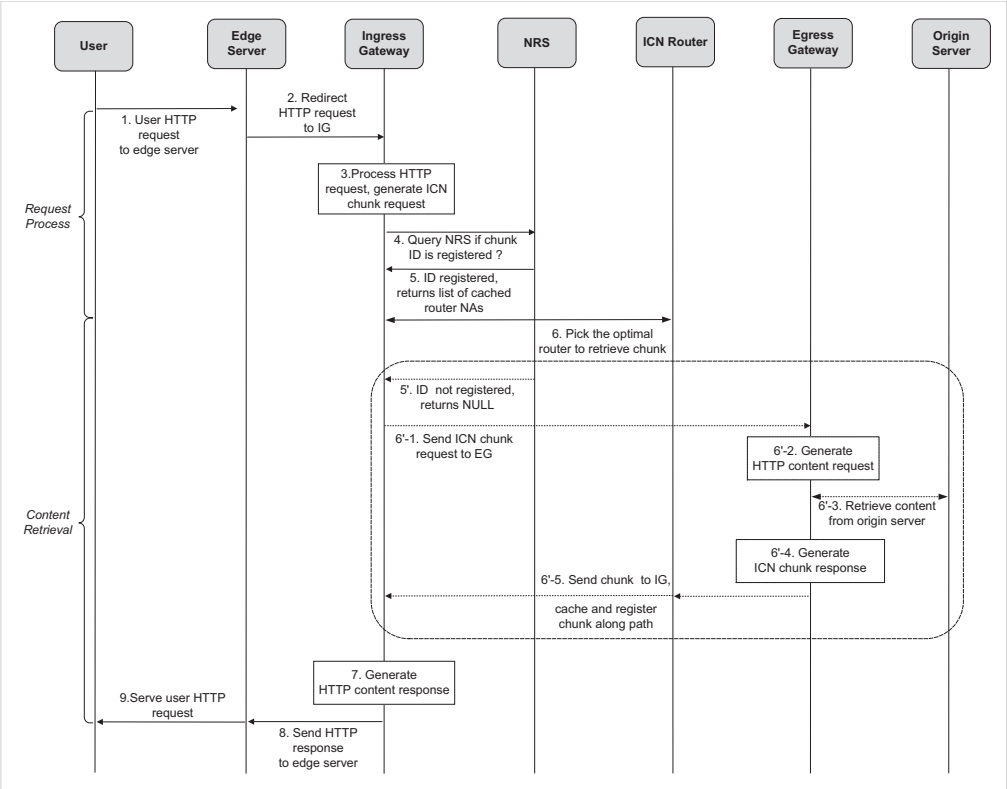

Figure 2. Flow diagram of content request and retrieval process.

Upon a cache miss on the CDN edge server, the content request is redirected to IG. This can be achieved by configuring IG as a proxy (steps 1–2 in Figure 2). IG parses the HTTP content request and initiates a name resolution request to NRS, querying if the corresponding chunk ID is registered (steps 3–4 in Figure 2). If registered, NRS returns the list of NAs of chunk replicas, and then IG selects the optimal replica using the nearest replica mechanism (steps 5–6 in Figure 2). If not registered (as in the dotted box), NRS returns NULL and the content needs to be retrieved from the origin server: first, IG initiates a chunk request to EG; second, EG handles the chunk request and generates an HTTP request of the content; third, EG requests the content from the origin server and sends the content back to IG in the form of a chunk. While forwarding, the chunk is cached along path (steps 5′–6′ in Figure 2). The details of the conversion between the HTTP and ICN paradigm will be discussed in the following section. Once IG retrieves the content, it serves the CDN edge server (steps 7–8 in Figure 2).

3.4. Details

3.4.1. ID-Based ICN Protocol Stack

In our design, an ID-based ICN communication paradigm is utilized to decouple the name and location. In that sense, the content is named by a unique ID but retrieved with a late-binding network address as presented in the above section. In order to process IDs while avoiding an overhaul in the current IP network, it is favorable to deploy the ICN paradigm over the existing IP infrastructure. Figure 3 shows the whole picture of this ICN protocol stack. The identifier protocol (IDP) [41] is extended over the IP protocol, and it regulates ID-based NA operations, such as ID-to-NA register and resolution. Above IDP, an ID-to-ID ICN transmission protocol [42] is running, and it provides a pull-based, chunk-oriented transmission service with efficiency and reliability mechanisms, such as retransmission and congestion control.

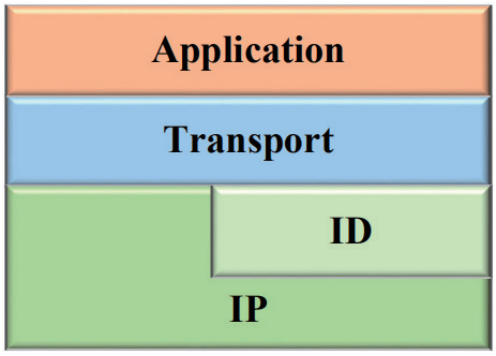

Figure 3. ID-based ICN protocol stack.

3.4.2. Design of IG and EG

IG and EG are the most important components of the ICN service layer. As the interchange points between ICN and HTTP, the HTTP-ICN gateways must be compatible with both the HTTP and ICN paradigms. Figure 4 shows the design diagram of the HTTP-ICN gateway.

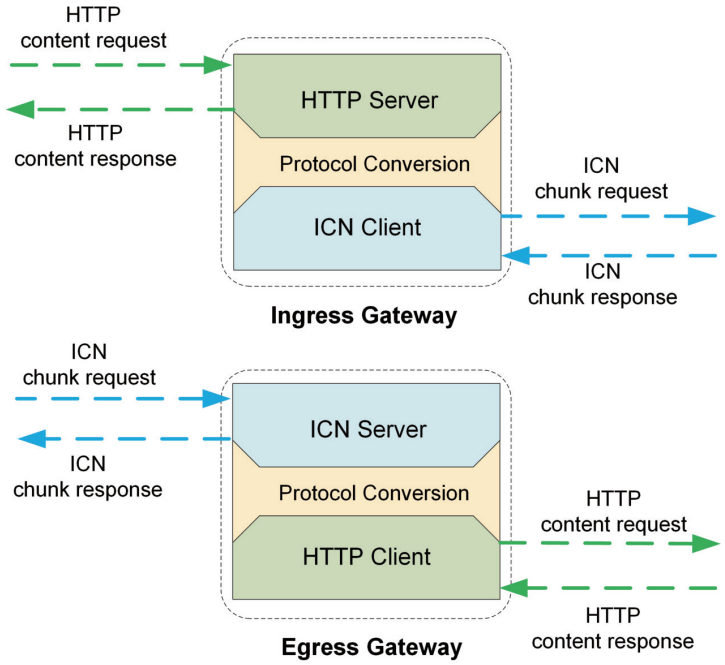

Figure 4. The design diagram of the HTTP-ICN gateway.

The HTTP server on IG parses and translates an HTTP request into an ICN chunk request with a protocol conversion mechanism. The chunk request is subsequently handled by the ICN client. Vice versa, upon receiving the chunk response on the ICN client, IG wraps the data in the HTTP payload and sends the HTTP response back to the CDN server. Similarly, EG receives an ICN chunk request, translates it into an HTTP content request and extracts data from the HTTP response payload to generate the ICN chunk response. The ICN client and server run on the ICN protocol stack described in the above section.

3.4.3. Protocol Conversion

The protocol conversion in our context is twofold:

1. HTTP to ICN: An HTTP request generally consists of a request line and several request headers. The request line commonly consists of three items: a method that instructs the server what it should do with the resource; a URI that locates and identifies the resource; and an HTTP version number. In ICN, each chunk is identified with a unique ID. Since the URL of each content is unique, we use SHA1 to generate the 20-byte hash digest of the content URL as its chunk ID. This process is illustrated in Figure 5. Afterwards, this content will exist in the form of chunks in the ICN network with a unique ID used for registration/publishing and resolution/retrieval. Upon receiving a GET request for content on the HTTP server, the ID of its chunk counterpart is calculated to locate its physical address by querying NRS, then the follow-up download is triggered.

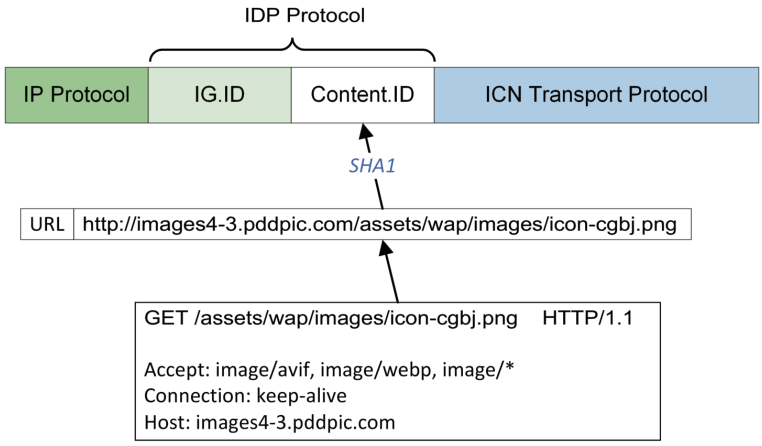

Figure 5. Protocol conversion from HTTP to ID-based ICN protocol.

2. ICN to HTTP: HTTP services on the application layer and is rich in semantics, whereas most of the ICN architectures reach up to only the network layer or transport layer. Apart from that, in our context, we use the hash digest of a content URL as its naming ID in ICN, which is a non-reversible operation. This means that a literal translation from the ICN protocol to HTTP is infeasible. To address this issue, we adopt a method commonly used in related work [43,44] that the ICN packets piggyback with extra control or feedback information. Specifically, since an HTTP request needs to be reconstructed by EG, IG sends to EG the ICN chunk request piggyback with HTTP request headers related to this content. Likewise, EG sends to IG the chunk response piggyback with the HTTP response headers for IG to reconstruct the HTTP response, and the response headers are cached on IG for future use. This can achieve zero loss of HTTP semantics at the cost of relatively low overhead.

3.4.4. Large Content Chunking

When a content is too large to fit into one chunk, it needs to be split into smaller pieces. Therefore, a proper chunking size should be determined. In [45], it is concluded that chunk size has a large impact on system performance.

However, determining the optimal chunk size is indeed a challenging task since both large and small chunk sizes exhibit trade-offs across different performance metrics. We summarize the related work in [45–47] and provide analysis in Table 2. In CCN, 4 KB is adopted as the default chunk size [3]. However, this value is not applicable in the context of HTTP-based web content. Inspired by the latest report on [48] that the average size of web content is about 2.3 MB, the rounded-down value of 2 MB may be suggested as an appropriate chunking size, as it is storage-friendly, which helps mitigate cache fragmentation and is large enough to avoid chunking in many cases. The optimal chunk size is to be further adjusted based on extensive empirical research.

Table 2. Performance impact of chunk size.

Impact	Large Chunk Size	Small Chunk Size
Cache Performance	Higher cache hit ratio longer cache entry lifetime; cache space not fully utilized	Lower cache hit ratio and shorter cache entry lifetime; better utilization of cache space
Transmission Performance	Longer retrieval delay; less transmission control overhead	Shorter retrieval delay but bad throughput; more control overhead introduced in packet header
Replica Router Load	One-hot cache more likely occurs and entails an imbalance of the router load	Distortion of content popularity but hot cache is distributed across more replica routers for load balancing

After multiple chunks are generated, each one of them requires a unique chunk ID. We use a metadata file, called manifest, to store the mapping relation between one content to many chunks so that chunks can be retrieved from the ICN distributed cache system and assembled as the content. The operation of the manifest can be described as follows: first, when EG pulls content from the origin server, the manifest for this content is generated and stored on EG; second, we assign the hash digest of the content name to manifest. As for the ID of each chunk within the content, it is generated by hashing the content name tagged along with the numerical order of each chunk. The list of the chunk NAs are encoded into the manifest with a TLV format. The steps of content chunking, ICN chunk response and manifest generating are described in Algorithm 1.

Algorithm 1 Large content chunking.

Input: HTTP content response ($HCRP$)
Output: ICN chunk responses ($ICRPs$), manifest chunk (MC)
Initialization: $ICRPs = \{\oslash\}$, $chunk\_size = 2MB$
$MC.ID \leftarrow hash(HCRP.url)$
if $HCRP.content\_length \leq chunk\_size$ **then**
    $ICRP.ID \leftarrow SHA1(HCRP.url + string(0))$
    $ICRP.payload \leftarrow HCRP.payload$
    $ICRPs.add(ICRP)$
    $MC.payload \leftarrow encode(ICRP.ID)$
else
    $chunk\_num \leftarrow \lceil HCRP.content\_length/chunk\_size \rceil$
    for $i = 0; i \leq chunk\_num; i++$ **do**
        $ICRPi.ID \leftarrow SHA1(HCRP.url + string(i))$
        $ICRPi.payload \leftarrow [i \times chunk\_size, (i+1) \times chunk\_size)$ in $HCRP.payload$
        $ICRPs.add(ICRP)$
    end for
    $MC.payload \leftarrow encode([ICRP0.ID, ICRP1.ID, \ldots, ICRPn.ID])$
end if
return $ICRPs, MC$

In such a case, the process of handling the HTTP request discussed in Section 3.3 should also be updated. The first step is to retrieve the manifest from EG, and after the retrieval, it is decoded to generate a list of chunk IDs belonging to the content. Afterwards, IG sends a resolution request of each chunk ID to NRS, and the follow-up process is the same. Multiple chunks can be retrieved concurrently from different cache replica routers so that the download latency can be reduced.

4. Problem Formulation

4.1. Model Description

The core philosophy of CDN is to move content from the places of origin servers to the places of edge servers, which are much closer to users so that a better QoS (lower access latency and higher transfer rate) is ensured. To this end, the optimization model is QoS-oriented. Meanwhile, a significant traffic engineering goal is to be responsive to sudden traffic surges and have better control over the link load since network links packed with traffic may experience significant degradation in performance. This means our model is subject to bandwidth constraints on each link, which translates our model into a Multi-Commodity Flow (MCF) problem.

We formulate a QoS-oriented model subject to bandwidth constraints to mathematically represent our proposed design. The model is considered a directed graph $G = (V, E)$ with $|V|$ vertices and $|E|$ edges. The set $V = \{v_1, v_2, \ldots, v_n\}$ and set $E = \{e_1, e_2, \ldots, e_n\}$ represent nodes and links in the network, respectively. A node can act as one of the following roles: user, router, CDN edge server, origin server, IG and EG. We assume that there is only one origin server providing all the contents to the whole network, which we denote as V_s. IG and EG are chosen from the set of routers $R = \{r_1, r_2, \ldots, r_n\}$ with a specific mechanism which will be introduced in Section 5. Let $U = \{u_1, u_2, \ldots, u_n\}$ be the set of users and $N = \{n_1, n_2, \ldots, n_m\}$ be the set of CDN edge servers. Thus, we have that $V = U \cup N \cup R \cup V_s$. Each router and CDN edge server can potentially cache content and is restricted with finite cache capacity $C(v)$. For each link (i, j) in the network, it is characterized by its bandwidth capacity $b_{(i,j)}$ and delay $d_{(i,j)}$.

A large proportion of content cached on CDN edge servers is web files whose sizes range from several bytes to megabytes. In most cases, a content is equivalent to a chunk. And we discuss content/chunks from an object level referring to [49]. To this end, we define the set of content as $F = \{f_1, f_2, \ldots, f_n\}$ and $Size(f_p) = Size(f_q), \forall f_p, f_q \in F$. Derived from MCF model, we define $K = \{k_1, k_2, \ldots, k_n\}$, $k = (s^k, t^k, \omega^k)$ as the set of demands in our model and refer to k as a content delivery demand, which needs to route a content from source s^k to destination t^k with a demand for transmission rate at ω^k. In our context, s^k is the node where the content is stored, and t^k is the user requesting the content. The content f demanded by each k complies with a certain probability distribution, which we denote as $f = \xi(k)$. In the context of CDN, Zipf distribution is one of the possibilities. For each demand k, we define variable $z_{(i,j)}$ such that if $z_{(i,j)} = 1$, the route goes through link (i, j) and $z_{(i,j)} = 0$ otherwise. The traffic generated by demand k on link (i, j) is denoted as $\gamma^k_{(i,j)}$. Moreover, the conversion from an HTTP paradigm to an ICN paradigm in our proposed design can introduce overhead, which we denote as α^k.

In our proposed design, both routers and CDN edge servers have to make the decision of caching a content according to its caching policy. Let $X(f, r) \longrightarrow \{0, 1\}$ be the Boolean function of caching the content f at router r, and $Y(f, n) \longrightarrow \{0, 1\}$ be the Boolean function of caching it at edge server n. After a content is cached at a router or edge server, it becomes an available source for content delivery demand k. We denote the content availability at a certain node with binary value $x \in \{0, 1\}$ and $y \in \{0, 1\}$ such that:

$$x^k_r = \begin{cases} 1, & \text{content demanded by } k \text{ is available at router } r \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$y_n^k = \begin{cases} 1, & \text{content demanded by } k \text{ is available at server } n \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

When a content is available at the edge server, it will be routed directly to user. However, on encountering a cache miss on the edge server, the routing occurs in two stages. The first is to route the content from a source that is either a router or the origin server in our proposed design, to the CDN edge server. The second is to route content from the edge server to user. Meanwhile, the content is cached along the path. We also assume that IG is placed near edge servers; thus, the delay from IG to an edge server is ignored. The routing process from the origin server to EG is simply regarded as a one-hop transmission with unlimited bandwidth and a large delay.

We evaluate the efficiency of completing a content delivery demand based on QoS, which we measure by its most important factor: the total latency from the source to the destination. In the meantime, each demand must fulfill constraints on link and node capacity. To proceed with further discussion, Table 3 summarizes the notations used in this paper.

Table 3. Summary of the notation used in this paper.

Notation	Comment
V	Set of vertices (network nodes)
E	Set of edges (network links)
U	Set of user nodes
N	Set of edge server nodes
R	Set of router nodes
V_s	The single origin server node
F	Set of contents
K	Set of demands
$C(v)$	Finite cache capacity of node v
$Size(f)$	Size of content f
$f = \xi(k)$	Mapping of demand k to content f
$b_{(i,j)}$	Link capacity of (i,j)
$d_{(i,j)}$	Link delay of (i,j)
$\gamma_{(i,j)}^k$	Traffic flow related to demand k at link (i,j)
α^k	Latency overhead related to demand k
$z_{(i,j)}^k$	0–1 variable, indicating whether (i,j) is used for routing demand k
$X(f,r)$	Boolean function of whether to cache content f at router r
$Y(f,n)$	Boolean function of whether to cache content f at edge server n
x_r^k	0–1 constraint, indicating availability of demand k at router r
y_n^k	0–1 constraint, indicating availability of demand k at edge server n

Given the above definitions and assumptions, we formulate our QoS-based optimal design model as follows:

$$min \sum_{k \in K} (\sum_{n \in N} \sum_{r \in R} (1 - y_n^k)[x_r^k l_1^k + (1 - x_r^k) l_1'^k]) + l_2^k \quad (3)$$

$$l_1^k = \sum_{(i,j) \in E} z_{(i,j)}^k d_{(i,j)} \ + \ \alpha^k, \forall \ k \in K, s^k \in R, t^k \in N, |s^k| \geqslant 1 \quad (4)$$

$$l_1'^k = \sum_{(i,j) \in E} z_{(i,j)}^k d_{(i,j)} \ + \ \alpha^k, \forall \ k \in K, s^k \in V_s, t^k \in N \quad (5)$$

$$l_2^k = \sum_{(i,j) \in E} z_{(i,j)}^k d_{(i,j)} \ , \forall k \in K \ , s^k \in N, t^k \in U \quad (6)$$

subject to constraints (7)–(15):

$$\sum_{j:(i,j)\in E} z^k_{(i,j)} - \sum_{j:(j,i)\in E} z^k_{(j,i)} = \lambda^k_i, \forall\, k \in K, (i,j) \in E, \tag{7}$$

$$\lambda^k_i = \begin{cases} 1, & i = s \\ -1, & i = t^k \\ 0, & i \neq s^k and\, i \neq t^k \end{cases}$$

$$\gamma^k_{(i,j)}\, z^k_{(i,j)} = \left\{0, \omega^k\right\}, \forall k \in K, \forall (i,j) \in E \tag{8}$$

$$\sum_{k\in K} z^k_{(i,j)}\, \gamma^k_{(i,j)} \leqslant b_{(i,j)}, \forall (i,j) \in E \tag{9}$$

$$\sum_{k\in K} [Y(n, \xi(k)) + y^k_n] \leqslant C(n), \forall n \in N \tag{10}$$

$$Y(n, \xi(k)) + y^k_n \leqslant 1, \forall n \in N, k \in K \tag{11}$$

$$\sum_{k\in K} [X(r, \xi(k)) + x^k_r] \leqslant C(r), \forall r \in R \tag{12}$$

$$X(r, f\xi(k)) + x^k_r \leqslant 1, \forall r \in R, k \in K \tag{13}$$

$$y^k_n = \{0,1\}, \forall k \in K, n \in N \tag{14}$$

$$x^k_r = \{0,1\}, \forall k \in K, n \in N \tag{15}$$

The objective function (3) minimizes the total content delivery latency over all content. For each demand k, l^k_1 in Equation (4) denotes the latency to deliver content from an ICN in-network caching router to the edge server. Note that $|s| \geq 1$, which means there is generally more than one available replica for content. Equation (5) means that if there is no available content in the ICN in-network cache, the source will be assigned to origin server, and the latency is denoted as l'^k_1 in such a case. l^k_2 in Equation (6) denotes the latency to deliver content from the edge server to the user. In (7), we use the standard flow conservation constraint of the MCF problem, i.e., for nodes that are neither the source nor destination, the inflow is equal to the outflow. This means our model does not support multicast routing, even though multicast is another attractive enhancement technique of ICN; we intend to explore this further in our future work. Constraint (8) ensures that each link chosen by demand k fulfills its demanded transmission rate of ω^k. Constraint (9) ensures that the traffic on link (i,j) does not surpass its capacity. Content cached on each router and the CDN edge server should not surpass its capacity, and only one replica of the content is cached per server/router as enforced by constraint (10)–(13). Finally, we add to the model the binary constraint (14) on y^k_n and (15) on x^k_r.

4.2. IECDN Model and Conventional CDN Model

The optimal design model we described above is based upon our proposed content delivery architecture. However, both our proposed design and a conventional CDN can be adapted to it. In our proposed design, α^k in the objective function (3) is a value larger than zero, whereas in a conventional CDN model, α^k is always equal to zero. Since routers in a conventional CDN do not perform caching, constraint (12)–(15) can be removed, and constraint (15) is re-adapted to (16). Therefore, a conventional CDN model has that objective function (3), wherein $\alpha^k \equiv 0$, subject to constraints (7)–(11), (14) and (16):

$$x^k_r = 0, \forall k \in K, n \in N \tag{16}$$

From a practical point of view, when a certain cache policy for ICN (e.g., LCE, LCD and CL4M) is given, the Boolean function $X(f,r)$ is given as an input of the problem. Meanwhile $Y(f,n)$ is deterministic when the HTTP response header indicates a preference for caching content (e.g., Cache-Control: max-age = 300 instructs server to cache the content for 300 s). For simplicity, we assume that content will always be cached on the CDN edge server, i.e., $Y(f,n) = 1$. Additionally, when a certain cache eviction policy is given for both ICN and CDN (e.g., LRU, LFU), constraints (10)–(13) are always satisfied. As for

constraints (14)–(15), the CDN edge server is aware of the existence of cached content, so y_n^k is also given as an input. Likewise, a content discovery mechanism of ICN provides the value of x_r^k . As we introduced in Section 3, a name resolution service is integrated in our proposed design, which facilitates content discovery. The value of variable $z_{(i,j)}^k$ is dependent upon both content delivery architecture design and routing. When a candidate design is given from $\{IECDN, Conventional\ CDN\}$, the problem is reduced to a routing problem, which prompts us to solve the problem using Linear Programming.

5. Simulation Study

5.1. Methodology

We implemented the simulation program in Python and the optimization problem was solved using CPLEX (v 22.1.0).

5.1.1. Topology

In order to establish a network topology for our proposed design that contains the key components, we developed a mechanism to generate the topology:

- Routers: We use a real network topology for core routers as follows:
  1. Abilene topology (12 nodes, 14 bidirectional links);
  2. Geant2 topology (32 nodes, 54 bidirectional links).
- IG and EG: Both IG and EG are selected from routers. Based on our previous assumption, IG is placed close to CDN edge servers. In light of the topology-informed placement strategy for CDN servers proposed in [50], we select IG as the router with the highest out-degrees so that it reaches out to more users with smaller latency. In contrast, EG is selected as the router with the maximum sum of distances to all other routers, reflecting that EG is placed towards the origin server, which is distant from users.
- Edge servers and origin server: Edge servers are arranged in a ring topology, with each one of them linked to the IG. We set the number of edge servers in proportion to the number of users, with one edge server per ten users. Lastly, the origin server is directly connected to the EG as previously assumed. A topology of our proposed design generated with the Abilene topology is shown in Figure 6.

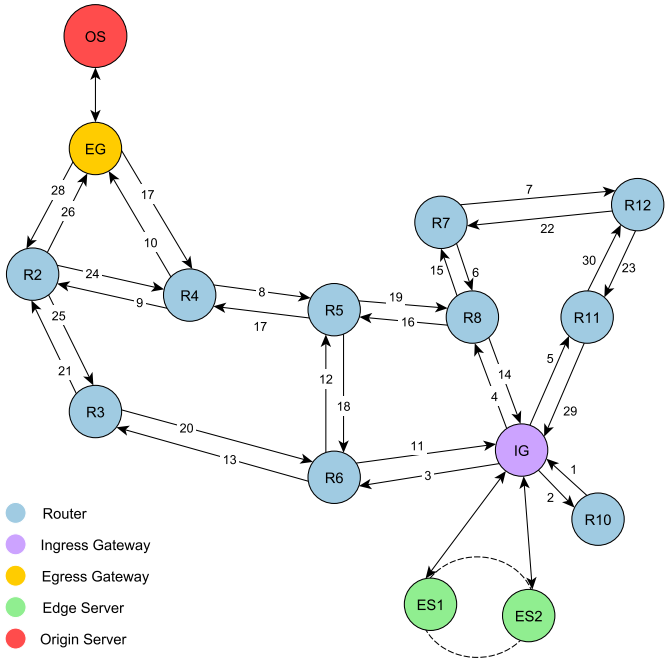

Figure 6. The Abilene topology used for IECDN.

5.1.2. Parameter Configuration

As stated in Section 4, our problem can be solved using an ILP (Integer Linear Programming) model. Hereafter, we list the parameters we used in our numerical analysis, which can be categorized as follows:

1. Content and request parameters: Since working with objects of a large cardinality is practically unfeasible in an ILP model, we generated a workload extracted from a catalog of 1×10^4 content objects following the Zipf distribution, and to study the impact of different content popularity on our model, we varied the parameter α of the Zipf distribution to within the range of [0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2], referring to [51]. The requests of this workload were designed up to 2×10^4, with each request mapped to a content delivery demand in our model. We generated different demand profiles at the numbers in the range of [10, 12, 14, 16, 18, 20] to model different user demand frequency. Within an analysis, the destination of each request/demand was randomly assigned to an unrepeated user; the CDN edge servers servicing these requests were scheduled in a Round Robin manner. To stay faithful to the real CDN routing scenario, routing for each requested content was carried out in two stages if necessary as mentioned in Section 4.1. Furthermore, we analyzed two possible user demand rates of 0.5 and 1.0 Gbps. Finally, we computed the mean total bandwidth consumption and delay as the outcome.
2. Network parameters: We classified links into two main categories, the core links and the edge links.
    - Core links refer to links connecting routers. Bandwidth limits on core links in both topologies are set to 10 Gbps. Delays on core links in the Abilene topology are set to the measurement results of [52].
    - Edge links connect routers to user. For both topologies, bandwidth limits on edge links are set to the highest user demand rate of 1.0 Gbps, and delays are set to random values between 1 and 5 ms, following a uniform distribution.

    In particular, according to our previous assumption, for the link between the origin server and EG, the bandwidth is set to infinite, and the delay is set to a large number, for which we deem 100 ms to be reasonable, considering the dataset we employed from [52]. Meanwhile, we relaxed the bandwidth constraints on links connected to IG since it is the meeting point of all the influx when delivering content from the source to the PoP. Their limits are elevated to the value of max (content demand frequency) $\times$ max (contend demand rate) to allow the concurrency of all the flows.
3. Cache parameters: In the context of our study, caching is involved in both CDN and ICN. As we assumed in Section 4.2, content is always preferably cached on CDN servers in our model. Regarding ICN, we implemented the LCE policy, which we deemed as the counterpart. We set the cache capacity for CDN nodes to be within the range of [10, 50, 100, 200, 300, 500] (objects). From a practical point of view, the cache capacity for CDN servers is assuredly larger than that of ICN routers. Therefore, we set the cache capacity of all ICN routers equally as one-tenth of the CDN server capacity, conforming to the homogeneous cache allocation strategy. When reaching the maximum cache capacity, the eviction policy of LRU is adopted for both ICN and CDN. The fundamental parameters we tested in our numerical analysis are shown in Table 4.

Table 4. Parameters for numerical analysis.

Parameter	Value	Default
Topology	Abilene, Geant2	Abilene
Catalog size	1×10^4	-
Request number	2×10^4	-
α of Zipf	[0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2]	0.8
Content demand frequency	[10, 12, 14, 16, 18, 20]	10
Content demand rate [Gbps]	0.5, 1.0	1.0
CDN node cache size [Objects]	[10, 50, 100, 200, 500]	100
ICN node cache size [Objects]	[1, 5, 10, 20, 50]	10

5.2. Result Analysis

5.2.1. Enhancement to Performance

In this experiment, we attempt to evaluate the performance of our proposed design compared with conventional CDN. We use the Abilene topology for analysis and vary the cache size in the network, noting that by changing the maximum cache size of CDN servers, the maximum cache size of the ICN routers is correspondingly changed. Other parameters are set to their default values. The results are illustrated in Figure 7a,b with the direct comparison of total bandwidth consumption and average content delivery delay under different cache sizes between IECDN and conventional CDN.

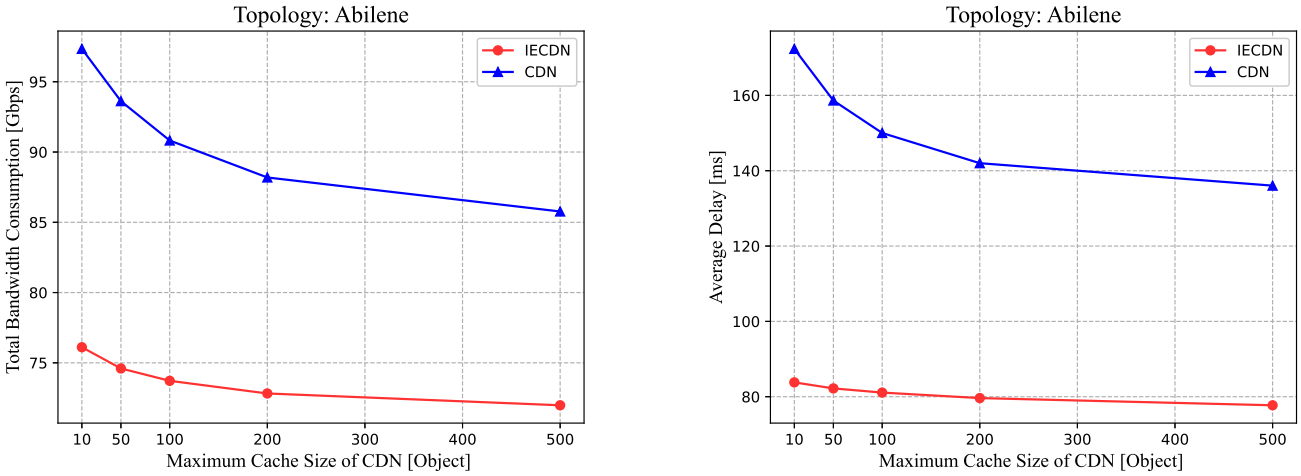

Figure 7. Comparison of total bandwidth consumption (**a**) and average delay (**b**) under different cache sizes between IECDN and conventional CDN.

Overall, it is obvious that compared with conventional CDN, IECDN achieves considerable performance gains while reducing bandwidth consumption. Therefore, it is verified that it is favorable to integrate ICN into the CDN architecture. Specifically, with a default cache size of 100 objects on CDN servers and correspondingly 10 objects on ICN routers, the average delay is reduced by approximately 46%, and bandwidth consumption is decreased by around 19%. Meanwhile, it is noteworthy that in the context of CDN, the reduction of bandwidth consumption and average delay exhibits a significant decline when the cache size per server increases from 10 to 200 objects, whereas there are only marginal benefits observed in the wide range of 200 to 500 cache size per server. This phenomenon can be attributed to the inherent characteristics of the Zipf distribution, wherein numerous objects are rarely accessed. The analysis of the impact of Zipf distribution on our proposed design is presented in the following section. A similar but more moderate trend can be also observed in the case of IECDN: when enlarging the cache size, despite a noticeable decline at the beginning, a saturation of both bandwidth consumption and average delay ensues

with minor benefits continuing to be achieved. This enlightened us that less storage cost is required for CDN providers to achieve an adequate performance goal in the presence of in-network caching.

5.2.2. Impact of Content Popularity

As discussed in the above section, the Zipf distribution exerts inherent influence on our model. In this experiment, we attempt to explore this further by studying the most import factor of Zipf distribution: for the α parameter, the higher the value of α, the more focused the users' preferences. We still use the Abilene topology for analysis and change the α parameters of the Zipf distribution. Other parameters are set as the default. Additionally, we present the results with both the comparison between the two architectures and their performance gap as the α parameter varies. As demonstrated in Figure 8a,b, as α increases, both the total bandwidth consumption and average delay decrease as expected because with a higher α value, there are more requests for popular content, which makes caching play a more important role in improving performance. However, as α increases, the performance gaps between IECDN and conventional CDN also tend to narrow (as indicated by the green columns). Specifically, with the α value of 0.6, the total bandwidth consumption is reduced by 19.24 Gbps, and the average delay is reduced by 77.16 ms, whereas these two gaps narrow down to 6.66 Gbps and 27.43 ms, respectively, with the α value of 1.2. This is also because a higher α value leads to a more focused preference for popular content. Most content with high popularity is requested and cached on CDN servers, which in turn makes more content readily available on CDN servers. Therefore, the ICN in-network caching tends to be of service for fewer content requests, resulting in smaller performance improvement.

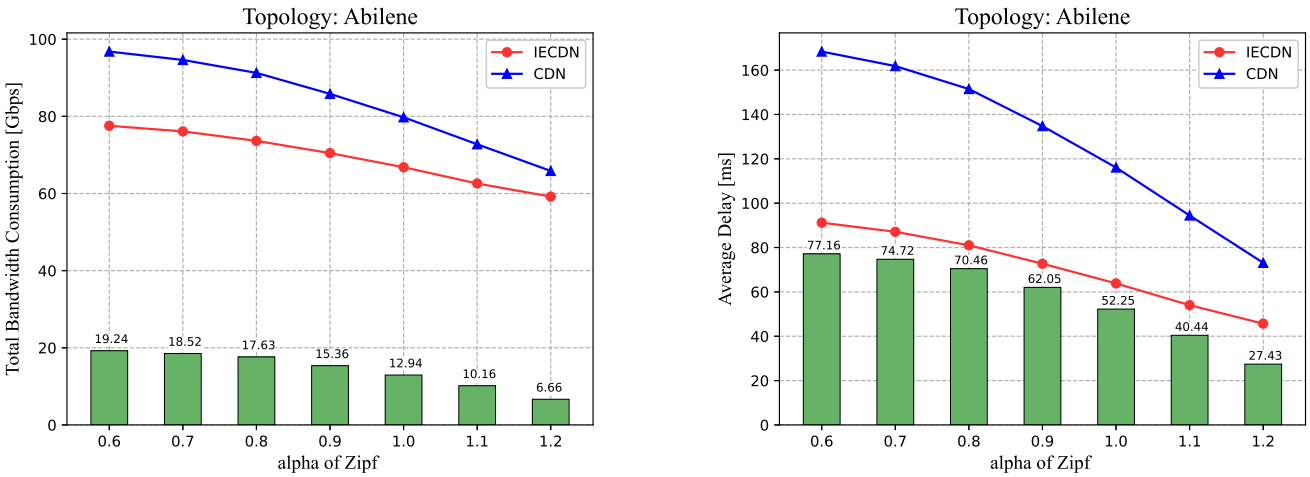

Figure 8. Comparison of total bandwidth consumption (**a**) and average delay (**b**) under different alpha parameter between IECDN and conventional CDN.

5.2.3. Resilience to Traffic Surge

At last, we attempt to dig deeper into our design's resilience to traffic surge compared with conventional CDN. By increasing the content demand frequency, more content is demanded to be routed in the network simultaneously, thus increasing the traffic. We also set the demand rate at two different values, 0.5 and 1.0 Gbps. Both Abilene and Geant2 topologies are used for analysis. Figure 9 depicts the trend of total bandwidth consumption in the Abilene and Geant2 network topologies as the content demand frequency increases.

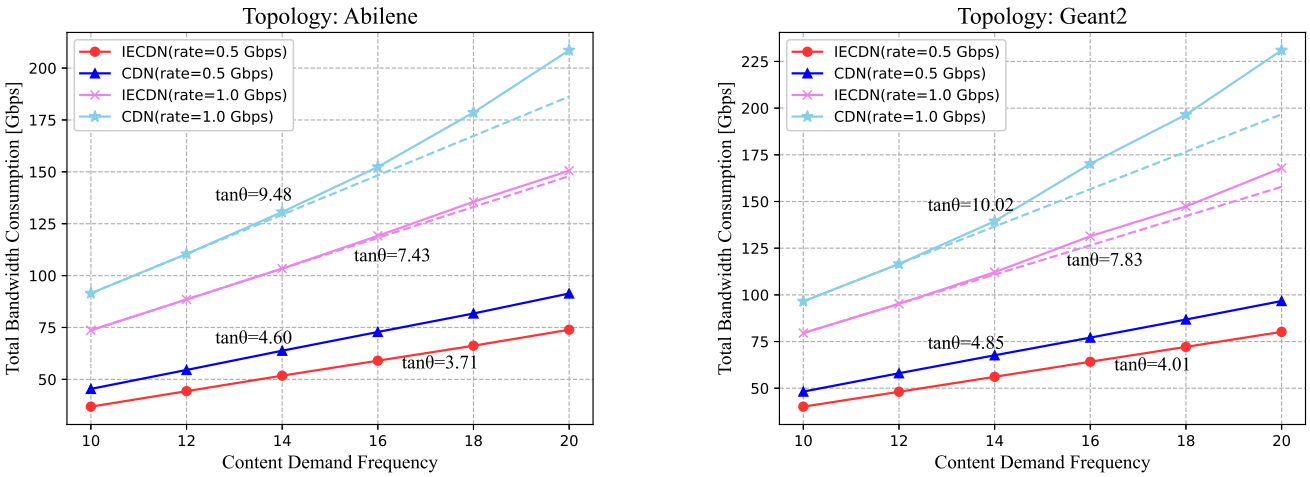

Figure 9. The trend of total bandwidth consumption in Abilene (**a**) and Geant2, (**b**) topology as content demand frequency increases.

It is observed that at the demand rate of 0.5 Gbps, the total bandwidth consumption is trended as near linear in all scenarios. This meets the expectation because even with the largest demand frequency of 20, the network links are not subject to congestion. Apart from that, there is a subtle feature that the line of conventional CDN rises more steeply than that of our proposed design. Specifically, we calculate the slope of each fitted line (as indicated by the dotted lines) to provide a clearer indication of its rate of growth, and it shows that the growth rate of the total bandwidth consumption in conventional CDN outweighs that of our proposed design. At the demand rate of 1.0 Gbps, congestion is more possibly to occur in the network links as demand frequency increases. More and more content has to make a longer detour to avoid congestion instead of a Dijkstra shortest path as imposed by link constraints. Under these circumstances, a linear trend of bandwidth consumption is unsustainable in conventional CDN. However, in our proposed design, the bandwidth consumption still maintains closely linear, which benefits from the presence of in-network caching that alleviates network congestion. This demonstrates a better resilience to traffic surge by integrating ICN and is of great importance to traffic engineering.

5.3. Overhead Analysis

In general, for small content, the latency overhead in our proposed design is mainly caused by the process of protocol conversion and name resolution. For each HTTP content request, a single hash operation is required for discovering content in the ICN network. We adopt a piggyback method to deliver information required for protocol conversion; thus, an additional session of data transmission is avoided. It was suggested by [53] that the cost of updating the control layer is considerably lower than that of data downloading. Meanwhile, by adopting the deterministic-latency name resolution method [54], α^k can be viewed as a constant, making resolution-based content discovery a promising solution. From a holistic point, the fraction of latency overhead decreases as the cardinality of the transmitted content in the network increases, which can be formulated in our model as $\lim_{k\to\infty} \sum_{k\in K} \alpha^k / \sum_{k\in K} l^k = 0$. On the other hand, as shown in Figure 7b, the content delivery latency nearly halves in the presence of ubiquitous cache, indicating that the latency overhead is worthwhile for the purpose of utilizing the ICN in-network caching.

However, for large content, another overhead is introduced since manifests are required to store the mapping between content and chunks. Here, we discuss this overhead in detail.

We conducted a simple simulation study focusing on content chunk and manifest generation under different chunking size. Web content sizes generally follow a Pareto

distribution, a typical heavy-tailed distribution. Assuming a random variable X follows a Pareto distribution, the probability distribution of X is given by Equation (17) as follows:

$$P(X > x) = (\frac{x}{x_{min}})^{-\alpha} \tag{17}$$

Suppose there are 100,000 pieces of content whose sizes follow a Pareto distribution with model parameter x_{min} = 133 KB, referring to [55]. As in Algorithm 1, the manifest primarily stores 20 byte chunk IDs. Therefore, we can calculate the corresponding manifest sizes based on the content sizes and chunking size. We utilize the average size (bytes) of manifests as the evaluation parameter because it is a crucial determinant of both the transmission and storage overhead. We use different α parameters of Pareto distribution to simulate varied content size distributions. When α takes values of 1.1, 1.2, and 1.3, the corresponding average content sizes are 1141, 761 and 517 KB. Figure 10 depicts the variation of average manifest size with changing chunk size. The trend implies that when the chunk size grows exponentially, the average manifest size exhibits a corresponding exponential decrease when the chunk size is relatively small. However, as the chunk size increases significantly, particularly surpassing the average size of all contents (as indicated by the dashed line position), the decreasing trend reduces and tends to stabilize. This is because when the chunk size is large, content smaller than that chunking granularity cannot be further divided, so increasing the chunk size further does not significantly decrease the average manifest size. Combining this observation with [48], which reports an average web content size of 2.3 MB, selecting the chunk size of 2 MB in our framework is acceptable as a value for reducing manifest management overhead.

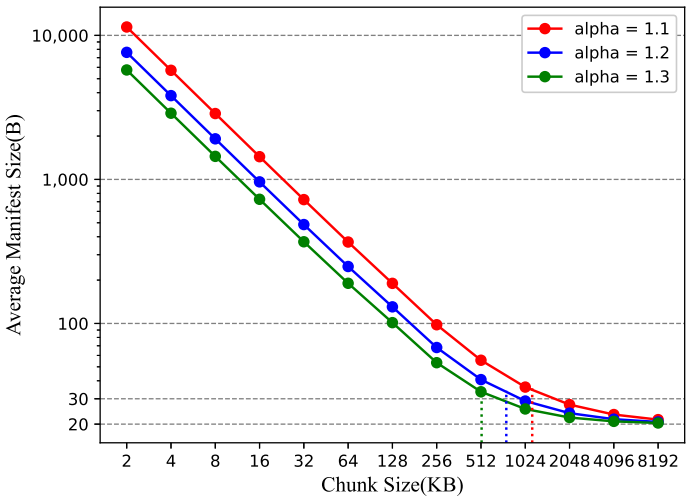

Figure 10. Average manifest size under different chunk sizes.

6. Discussion and Conclusions

ICN in-networking caching has been widely adopted to address issues in other fields because of its advantages in terms of less content retrieval latency, low network traffic and so on. This study focuses on the design and evaluation of utilizing the ubiquitous caching of ICN as a service to enhance the performance of CDN. To this end, first, we introduce the overview of our proposed design of a CDN enhanced with ICN, called IECDN, and elaborate on the mechanisms for enabling the collaboration between CDN and ICN to achieve efficient content delivery. Second, we formulate an optimization model derived from the MCF problem, which quantifies the optimization goal for content delivery. Through a series of numerical analyses, the results demonstrate that our design not only improves content delivery efficiency but also reduces the network load and is more resilient to network traffic surge. Based on these findings, we venture the conclusion that the

ICN-based enhanced CDN distribution can economically satisfy the exponential growth of internet content without placing an unacceptable burden on link loads.

However, the ubiquitous caching and dynamic replica selection of ICN also pose a great challenge to the security design within ICN. In our framework, an associated manifest is generated when the content is published in ICN. Therefore, the manifest can be leveraged as a handle to ensure content-level security. In our forthcoming work, we intend to further explore the solutions to security issues by considering the distinctive features of our framework and using the existing mature security mechanism support. Moreover, we will explore more advantages of ICN in the domains where it can possibly provide better solutions.

Author Contributions: Conceptualization, L.G. and X.Z.; methodology, L.G. and X.Z.; software, L.G.; writing—original draft preparation, L.G.; writing—review and editing, L.G. and X.Z.; supervision, X.Z.; project administration, X.Z.; funding acquisition, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Goal-Oriented Project Independently Deployed by Institute of Acoustics, Chinese Academy of Sciences: Distributed Supercomputing Based on SEANET Network (project no. MBDX202114).

Data Availability Statement: Data are contained within the article.

Acknowledgments: We would like to express our deepest gratitude to Linlin Hu and Chunmei Liu for their meaningful support for this work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco Annual Internet Report (2018–2023) White Paper. Available online: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed on 29 November 2023).
2. Norton, W. The emerging 21st century access power peering. *Commun. Strateg.* **2011**, *84* , 55–73.
3. Jacobson, V.; Smetters, D.K.; Thornton, J.D.; Plass, M.F.; Briggs, N.H.; Braynard, R.L. Networking named content. In Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, Rome, Italy, 1–4 December 2009; pp. 1–12.
4. Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; Claffy, K.; Crowley, P.; Papadopoulos, C.; Wang, L.; Zhang, B. Named data networking. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 66–73. [CrossRef]
5. Fotiou, N.; Nikander, P.; Trossen, D.; Polyzos, G.C. Developing information networking further: From PSIRP to PURSUIT. In Proceedings of the Broadband Communications, Networks, and Systems: 7th International ICST Conference, BROADNETS 2010, Athens, Greece, 25–27 October 2010; Revised Selected Papers 7; Springer: Berlin/Heidelberg, Germany, 2012; pp. 1–13.
6. Koponen, T.; Chawla, M.; Chun, B.G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A data-oriented (and beyond) network architecture. In Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Kyoto, Japan, 27–31 August 2007; pp. 181–192.
7. Dannewitz, C.; Kutscher, D.; Ohlman, B.; Farrell, S.; Ahlgren, B.; Karl, H. Network of information (netinf)—An information-centric networking architecture. *Comput. Commun.* **2013**, *36*, 721–735. [CrossRef]
8. Venkataramani, A.; Kurose, J.F.; Raychaudhuri, D.; Nagaraja, K.; Mao, M.; Banerjee, S. Mobilityfirst: A mobility-centric and trustworthy internet architecture. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 74–80. [CrossRef]
9. Wang, J.; Chen, G.; You, J.; Sun, P. Seanet: Architecture and technologies of an on-site, elastic, autonomous network. *J. Netw. New Media* **2020**, *6*, 1–8.
10. Zhang, Z.; Lung, C.H.; Wei, X.; Chen, M.; Chatterjee, S.; Zhang, Z. In-network Caching for ICN-based IoT (ICN-IoT): A Comprehensive Survey. *IEEE Internet Things J.* **2023**, *10*, 14595–14620. [CrossRef]
11. Khelifi, H.; Luo, S.; Nour, B.; Moungla, H. In-network caching in ICN-based vehicular networks: Effectiveness & performance evaluation. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
12. Serhane, O.; Yahyaoui, K.; Nour, B.; Moungla, H. A survey of ICN content naming and in-network caching in 5G and beyond networks. *IEEE Internet Things J.* **2020**, *8*, 4081–4104. [CrossRef]
13. Zeng, R.; You, J.; Li, Y.; Han, R. An ICN-based IPFS high-availability architecture. *Future Internet* **2022**, *14*, 122. [CrossRef]
14. Passarella, A. A survey on content-centric technologies for the current Internet: CDN and P2P solutions. *Comput. Commun.* **2012**, *35*, 1–32. [CrossRef]
15. Hawley, J. GeoDNS-Geographically-aware, protocol-agnostic load balancing at the DNS level. In Proceedings of the Linux Symposium, Montreal, QC, Canada, 13–17 July 2009; pp. 123–130.

16. Calder, M.; Flavel, A.; Katz-Bassett, E.; Mahajan, R.; Padhye, J. Analyzing the Performance of an Anycast CDN. In Proceedings of the 2015 Internet Measurement Conference, Tokyo, Japan, 28–30 October 2015; pp. 531–537.
17. Ari, I. *Design and Management of Globally Distributed Network Caches*; University of California: Santa Cruz, CA, USA, 2004.
18. Bhattacharjee, S.; Calvert, K.L.; Zegura, E.W. Self-organizing wide-area network caches. In Proceedings of the IEEE INFOCOM'98, the Conference on Computer Communications, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies, Gateway to the 21st Century (Cat. No. 98), San Francisco, CA, USA, 29 March–2 April 1998; IEEE: Piscataway, NJ, USA, 1998; Volume 2, pp. 600–608.
19. Rosensweig, E.J.; Kurose, J. Breadcrumbs: Efficient, best-effort content location in cache networks. In Proceedings of the IEEE INFOCOM 2009, Rio de Janeiro, Brazil, 19–25 April 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 2631–2635.
20. Laoutaris, N.; Che, H.; Stavrakakis, I. The LCD interconnection of LRU caches and its analysis. *Perform. Eval.* **2006**, *63*, 609–634. [CrossRef]
21. Laoutaris, N.; Syntila, S.; Stavrakakis, I. Meta algorithms for hierarchical web caches. In Proceedings of the IEEE International Conference on Performance, Computing, and Communications, 2004, Phoenix, AZ, USA, 15–17 April 2004; IEEE: Piscataway, NJ, USA, 2004; pp. 445–452.
22. Chai, W.K.; He, D.; Psaras, I.; Pavlou, G. Cache "less for more" in information-centric networks. In Proceedings of the 11th International Networking Conference (NETWORKING), Prague, Czech Republic, 21–25 May 2012; Springer: Berlin/Heidelberg, Germany, 2012; number Part I, pp. 27–40.
23. Shailendra, S.; Sengottuvelan, S.; Rath, H.K.; Panigrahi, B.; Simha, A. Performance evaluation of caching policies in ndn-an icn architecture. In Proceedings of the 2016 IEEE Region 10 Conference (TENCON), Singapore, 22–25 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1117–1121.
24. Rossi, D.; Rossini, G. On sizing CCN content stores by exploiting topological information. In Proceedings of the 2012 Proceedings IEEE INFOCOM Workshops, Orlando, FL, USA, 25–30 March 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 280–285.
25. Mansilha, R.B.; Saino, L.; Barcellos, M.P.; Gallo, M.; Leonardi, E.; Perino, D.; Rossi, D. Hierarchical content stores in high-speed ICN routers: Emulation and prototype implementation. In Proceedings of the 2nd ACM Conference on Information-Centric Networking, San Francisco, CA, USA, 30 September–2 October 2015; pp. 59–68.
26. Hong, J.; You, T.; Dong, L.; Westphal, C.; Ohlman, B. RFC 9138 Design Considerations for Name Resolution Service in Information-Centric Networking (ICN). Available online: https://www.rfc-editor.org/rfc/rfc9138.html (accessed on 29 November 2023).
27. Ma, G.; Chen, Z.; Cao, J.; Guo, Z.; Jiang, Y.; Guo, X. A tentative comparison on CDN and NDN. In Proceedings of the 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, USA, 5–8 October 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 2893–2898.
28. Ma, G.; Chen, Z. Comparative Study on CCN and CDN. In Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 27 April–2 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 169–170.
29. Mangili, M.; Martignon, F.; Capone, A. A comparative study of content-centric and content-distribution networks: Performance and bounds. In Proceedings of the 2013 IEEE Global Communications Conference (GLOBECOM), Atlanta, GA, USA, 9–13 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1403–1409.
30. Ghasemi, C.; Yousefi, H.; Zhang, B. Far cry: Will cdns hear ndn's call? In Proceedings of the 7th ACM Conference on Information-Centric Networking, Virtual, 29 September–1 October 2020; pp. 89–98.
31. Jiang, X.; Bi, J. ncdn: Cdn enhanced with ndn. In Proceedings of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 27 April–2 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 440–445.
32. Lin, T.; Xu, Y.; Zhang, G.; Xin, Y.; Li, Y.; Ci, S. R-iCDN: An approach supporting flexible content routing for ISP-operated CDN. In Proceedings of the 9th ACM Workshop on Mobility in the Evolving Internet Architecture, Maui, HI, USA, 11 September 2014; pp. 61–66.
33. Benkacem, I.; Bagaa, M.; Taleb, T.; Nguyen, Q.; Toshitaka, T.; Sato, T. Integrated ICN and CDN Slice as a Service. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–7.
34. Ghasemi, C.; Yousefi, H.; Zhang, B. icdn: An ndn-based cdn. In Proceedings of the 7th ACM Conference on Information-Centric Networking, Virtual, 29 September–1 October 2020; pp. 99–105.
35. Chen, J.; Xu, H.; Penugonde, S.; Zhang, Y.; Raychaudhuri, D. Exploiting ICN for efficient content dissemination in CDNs. In Proceedings of the 2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), Washington, DC, USA, 24–25 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 14–19.
36. Marchal, X.; El Aoun, M.; Mathieu, B.; Cholez, T.; Doyen, G.; Mallouli, W.; Festor, O. Leveraging NFV for the deployment of NDN: Application to HTTP traffic transport. In Proceedings of the NOMS 2018—2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, 23–27 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–5.
37. Fahrianto, F.; Kamiyama, N. Comparison of migration approaches of ICN/NDN on IP networks. In Proceedings of the 2020 Fifth International Conference on Informatics and Computing (ICIC), Gorontalo, Indonesia, 3–4 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–7.
38. Wang, S.; Bi, J.; Wu, J.; Yang, X.; Fan, L. On adapting http protocol to content centric networking. In Proceedings of the 7th International Conference on Future Internet Technologies, Seoul, Republic of Korea, 11–12 September 2012; pp. 1–6.

39. Jahanian, M.; Chen, J.; Ramakrishnan, K. Managing the evolution to future internet architectures and seamless interoperation. In Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, USA, 3–6 August 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–11.
40. Zhang, F.; Zhang, Y.; Raychaudhuri, D. Edge caching and nearest replica routing in information-centric networking. In Proceedings of the 2016 IEEE 37th Sarnoff Symposium, Newark, NJ, USA, 19–21 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 181–186.
41. Dang, S.; Han, R. An In-Network Cooperative Storage Schema Based on Neighbor Offloading in a Programmable Data Plane. *Future Internet* **2021**, *14*, 18. [CrossRef]
42. Xu, Y.; Ni, H.; Zhu, X. An effective transmission scheme based on early congestion detection for information-centric network. *Electronics* **2021**, *10*, 2205. [CrossRef]
43. Yang, Y.; Song, T.; Zhang, B. OpenCache: A lightweight regional cache collaboration approach in hierarchical-named ICN. *Comput. Commun.* **2019**, *144*, 89–99. [CrossRef]
44. Yang, W.; Qin, Y.; Yang, Y. An interest shaping mechanism in NDN: Joint congestion control and traffic management. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
45. Nasis, C.; Sarros, C.A.; Tsaoussidis, V. The Impact of Chunk Size on Named Data Networking Performance. In Proceedings of the 2020 3rd International Conference on Hot Information-Centric Networking (HotICN), Hefei, China, 12–14 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 108–113.
46. Song, Y.; Ni, H.; Zhu, X. Analytical modeling of optimal chunk size for efficient transmission in information-centric networking. *Int. J. Innov. Comput. Inf. Control* **2020**, *16*, 1511–1525.
47. Wang, L.; Bayhan, S.; Kangasharju, J. Optimal chunking and partial caching in information-centric networks. *Comput. Commun.* **2015**, *61*, 48–57. [CrossRef]
48. Report: State of the Web. Available online: https://httparchive.org/reports/state-of-the-web89-99 (accessed on 22 November 2023).
49. Rossini, G.; Rossi, D. Evaluating CCN multi-path interest forwarding strategies. *Comput. Commun.* **2013**, *36*, 771–778. [CrossRef]
50. Jamin, S.; Jin, C.; Kurc, A.R.; Raz, D.; Shavitt, Y. Constrained mirror placement on the Internet. In Proceedings of the IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213), Anchorage, AK, USA, 22–26 April 2001; IEEE: Piscataway, NJ, USA, 2001; Volume 1, pp. 31–40.
51. Guillemin, F.; Kauffmann, B.; Moteau, S.; Simonian, A. Experimental analysis of caching efficiency for YouTube traffic in an ISP network. In Proceedings of the 2013 25th International Teletraffic Congress (ITC), Shanghai, China, 10–12 September 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–9.
52. Mekaouil, S.; Benhamed, C.; Ghoumid, K. Traffic matrix estimation using the Levenberg-Marquardt neural network of a large IP system. *Data Manag. Secur. Appl. Med. Sci. Eng.* **2013**, *45*, 85.
53. Azimdoost, B.; Westphal, C.; Sadjadpour, H.R. Resolution-based content discovery in network of caches: Is the control traffic an issue? *IEEE Trans. Commun.* **2017**, *65*, 2943–2955. [CrossRef]
54. Liao, Y.; Sheng, Y.; Wang, J. A deterministic latency name resolution framework using network partitioning for 5G-ICN integration. *Int. J. Innov. Comput. Inf. Control* **2019**, *15*, 1865–1880.
55. Melazzi, N.B.; Detti, A.; Pomposini, M.; Salsano, S. Route discovery and caching: A way to improve the scalability of Information-Centric Networking. In Proceedings of the 2012 IEEE Global Communications Conference (GLOBECOM), Anaheim, CA, USA, 3–7 December 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 2701–2707.