Implementation of Proxy to Make Content Shared on BitTorrent Retrievable from CCN

Kohei Okazaki
Graduate School of Information Sciences
Hiroshima City University
Hiroshima 731-3194 JAPAN
okazaki@net.info.hiroshima-cu.ac.jp

Junichi Funasaka
Graduate School of Information Sciences
Hiroshima City University
Hiroshima 731-3194 JAPAN
funa@hiroshima-cu.ac.jp

***Abstract*—In the modern era, it's widely acknowledged that the Internet Protocol (IP), originally designed for location-based communication, is increasingly misaligned with today's content-centric network usage. This divergence has led to the emergence of partial solutions like Content Delivery Networks (CDNs) and Peer-to-Peer (P2P) protocols such as BitTorrent. While these technologies address the immediate need for efficient content delivery, they do not offer a fundamental resolution to the issues arising from the growing disconnect between IP's original design philosophy and contemporary network demands. Recent years have seen the advent of new networking architectures like Information-Centric Networking (ICN) and Content-Centric Networking (CCN). These groundbreaking designs aim to align network operations more closely with content-centric usage patterns. However, they struggle with compatibility issues when interfacing with the existing IP-based systems, a challenge that hinders their widespread adoption. This study introduces a proxy implementation that allows content from BitTorrent to be retrieved by CCN clients, filling a significant gap in the literature by confirming the feasibility of such an approach. Our experimental results indicate that the proxy enables content retrieval at about 85% of BitTorrent's standalone performance, accounting for inevitable overheads. Moreover, our data show that when CCN content caches are available, retrieval speeds can indeed surpass those of standalone BitTorrent.**

***Index Terms*—Proxy Design, Information-Centric Networking (ICN), Content-Centric Networking (CCN), Peer-to-Peer (P2P), BitTorrent, Network Architecture**

I. Introduction

Since its inception, the Internet has undergone significant developments, becoming an indispensable means of communication over the last few decades. Initially dominated by text-based information retrieval and email communication, modern-day Internet usage now encompasses a myriad of applications, such as high-definition video streaming, real-time online gaming, and even Internet of Things (IoT) devices. Particularly noteworthy is the exponential surge in demand for large-volume content like videos, a trend corroborated by Cisco's recent reports [1].

This evolution has exposed a misalignment between the original location-centric design of the IP protocol and the content-centric needs of contemporary Internet usage. To bridge this gap, technologies like Content Delivery Networks (CDNs) and Peer-to-Peer (P2P) protocols have been developed. However, these technologies introduce their own challenges. CDNs often differ in architecture depending on the service provider, leading to operational complexity and elevated costs. P2P protocols, notably BitTorrent, suffer from efficiency constraints unless participated in by a large user base.

Against this backdrop, Information-Centric Networking (ICN) or Content-Centric Networking (CCN) is proposed as a new network architecture [2] [3]. This architecture assigns names to content and performs routing and caching based on these names, making it highly suitable for today's content-driven communication needs. However, this novel architecture lacks fundamental compatibility with existing IP networks, a factor that has historically impeded the adoption of new technologies, as witnessed in the transition from IPv4 to IPv6.

The contributions of this paper are proposing a proxy which enables CCN clients to retrieve the contents shared by BitTorrent on the IP network, designing and implementing a proxy program on a real operating system, and evaluating the performance of the implementation to confirm the effectiveness of our proposal. Our proxy provides a measure bridging the existing gap between ICN/CCN and conventional IP-based P2P protocols like BitTorrent.

The remainder of this paper is organized as follows: Section Ⅱ presents an overview of related work in the field. In Section Ⅲ, we elaborate on the design considerations for the proxy. Section Ⅳ describes the experimental evaluation conducted to assess the proxy's performance, followed by a discussion of the results. Finally, Section Ⅴ concludes the paper and outlines future directions for research.

II. Related Work

In the realm of network protocols, Content-Centric Networking (CCN) stands out as a groundbreaking shift. Unlike traditional IP networking, which concerns itself with *where* to send the data, focusing on the destination IP address, CCN changes the narrative to *what* is being sent—the *content*. This mechanism is especially fitting in our data-hungry world where large volumes of content—videos, texts, images—are disseminated across vast geographical locations. Notably, the naming scheme in CCN is content-focused, assigning a unique identifier to each content piece, which forms the basis for request and transfer.

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

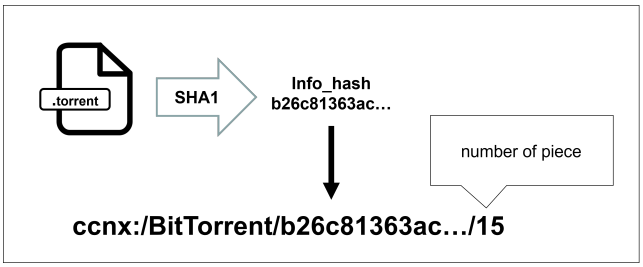

Fig. 1. Naming conventions for identifying BitTorrent content within CCN

Contrastingly, BitTorrent is another giant but in the realm of P2P file-sharing protocols. Its design optimizes the distribution of large files across distributed networks. It employs a hybrid architecture that incorporates a centralized server called a tracker, which aids peer coordination. One of the distinguishing features is the fragmentation of large files into smaller parts called 'pieces,' typically ranging from 256KB to 2MB. Each of these pieces is verified through unique hash values, maintaining the integrity of the downloaded content.

While both CCN and BitTorrent offer unique attributes that cater to the specific needs of content distribution, they inherently function based on different sets of protocols, making integration a challenge. Bridging this gap is an area of ongoing research. Fahrianto and Kamiyama [4] proposed a dual-channel translation gateway between IP and NDN, with NDN being one of the instantiations of Information Centric Networking (ICN). Their innovation lies in enabling seamless bi-directional accessibility between IP and NDN. However, their design is not straightforward; mutual conversion between IP packets and NDN ones presents challenges since a computer can store multiple contents, meaning one content name does not necessarily correspond to one computer or IP name. In contrast, BitTorrent has torrent information uniquely identifying each content, demonstrating a high degree of compatibility with ICN/CCN concepts.

Notable studies like [5] and [6] have made strides in integrating HTTP-based systems with CCN. While these works offer efficient means of data transfer by converting HTTP requests to CCN Interest packets and vice versa, they face challenges in handling dynamic content. Moreover, HTTP-based systems frequently struggle with scalability issues, particularly when dealing with large-capacity contents. These limitations make CCN an attractive alternative for handling such content efficiently, which is the focus of our approach.

Our work takes a step further to fill this integration void. We aim to design and implement a proxy mechanism that can successfully integrate the high efficiency of CCN's content-centric design with the robustness and widespread adoption of BitTorrent in P2P file sharing. To the best of our knowledge, this is the first attempt to make shared contents among BitTorrent peers accessible from CCN.

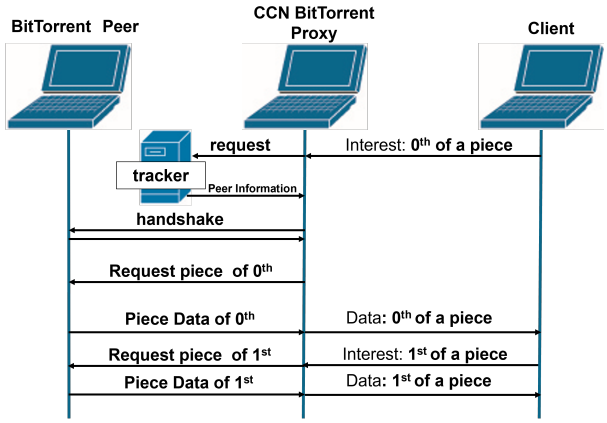

Fig. 2. Sequence Diagram

III. Proxy Design

BitTorrent and Content-Centric Networking (CCN) serve as effective paradigms for content distribution, albeit with divergent methodologies. While BitTorrent excels with its efficient content-sharing mechanism, CCN offers the advantage of name-based addressing. The challenge lies in integrating these fundamentally different protocol semantics to harness the strengths of both. This section presents a proxy design aimed at this integration, assuming that routing is pre-configured and the proxy retains relevant torrent files for conversions.The implementation code is publicly available on GitHub[1][2].

A. Role of the Proxy

The proxy serves as a mediator between CCN and BitTorrent networks. Upon receiving an *Interest* packet from CCN, the proxy joins the corresponding BitTorrent swarm to begin fetching content "pieces". Each fetched piece undergoes integrity verification before being relayed back to the CCN network as a *Data* packet.

The process of retrieving content from a BitTorrent peer by a CCN client is depicted in the sequence diagram presented in Figure 2. When joining a BitTorrent swarm, if the proxy is not already a participant, it initiates the download process by querying the tracker for peer information. Subsequently, after performing handshakes with the proxy and the introduced peer, the proxy requests the first piece from the BitTorrent peer. It is important to emphasize that the time required to access the tracker and complete the handshake with the BitTorrent peer is unavoidable during the initial content access phase. Once the first piece is retrieved, subsequent pieces can be directly requested from the BitTorrent peer.

B. Protocol Conversion Mechanisms

1) Name Mapping: BitTorrent uniquely identifies content using an *info_hash*. By retaining this info_hash as a part of the CCN content name, we establish a direct mapping between

[1]https://github.com/Marie673/Torrent_Proxy.git
[2]https://github.com/Marie673/ccn_torrent_client.git

BitTorrent and CCN semantics. Additionally, the piece number is also incorporated into the CCN name to facilitate efficient content retrieval. The CCN naming follows the pattern: `ccnx:/BitTorrent/{info_hash}/{piece_index}`. Figure 1 shows a concrete example of name mapping.

2) Content Verification: Stored within each torrent file are the hash values corresponding to individual pieces of content. Upon receipt of a piece, the proxy uses these hash values to verify the integrity of the data. Should the verification succeed, the data is then forwarded into the CCN network as a *Data* packet.

C. Client Design and Role

Beyond serving as a proxy, the system also involves the development of a specialized client. The client is responsible for parsing torrent files and subsequently issuing *Interest* packets to retrieve content. This essentially makes the client an active requester in the CCN network, targeting the proxy to fulfill its content requirements. Additionally, the client implements CCN's congestion control using the CUBIC algorithm, ensuring efficient and stable data flow within the network.

D. Data Chunking and Efficiency

BitTorrent typically deals with content pieces that are further divided into smaller blocks, usually of size 32KB. However, aligning this directly with CCN's chunk size would be inefficient. Following previous research [6], our approach delivers piece data in 4096B units, which better suits the CCN architecture for optimal data flow.

E. Implementation

The implementation of this research is based on cefore (v0.9.0b), a communication software for ICN/CCN developed by NICT. Cefore is advantageous in enhancing interoperability between protocols as it adheres to the specifications of CCN.

1) Software and Packages Utilized: For the actual development of the proxy and client, Python 3.10 is employed along with the cefore application development package, cefpyco (v0.6.3). Cefpyco provides functionalities to easily send and receive CCN Interest and Data packets, allowing seamless integration with cefore.

2) Detailed Implementation of Proxy: Upon receiving an Interest from CCN, the proxy joins the corresponding BitTorrent network to fetch the required content pieces. This process also involves protocol translation between CCN and BitTorrent, facilitated by cefpyco. The APIs provided by cefpyco make packet parsing, and data generation and transmission straightforward.

3) Detailed Implementation of Client: The client also utilizes cefpyco for its operations. Through cefpyco, the client parses the torrent files and issues Interest packets for the required content. Again, API functionalities simplify the efficient retrieval of the needed content pieces.

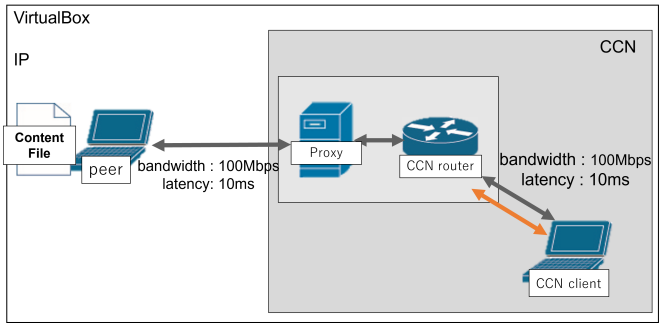

Fig. 3. Experimental Environment (Node Configuration 1)

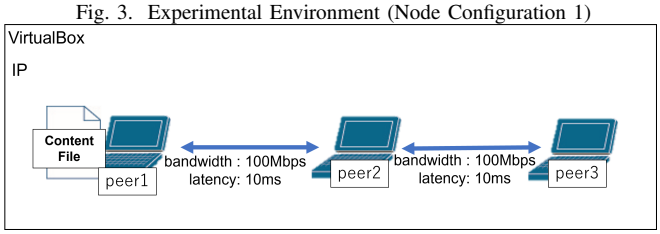

Fig. 4. Experimental Environment (Node Configuration 2)

4) Data Integrity and Chunk Size Optimization: After verifying the integrity of the retrieved content, the proxy transmits it as CCN Data using cefpyco. Additionally, based on existing research, the chunk size is optimized to 4096 B, thereby enhancing the efficiency of data transfer.

Through the utilization of cefore and cefpyco, the efficient integration of CCN and BitTorrent is made possible, realizing the implementation of the proxy and client proposed in this research.

IV. Experiment and Discussion

To utilize BitTorrent content from CCN, we prepare a test-bed network as shown in Figure 3. The test-bed includes three nodes: CCN client, proxy node involving CCN router and proxy functions, and BitTorrent peer. The proxy node has CCN router function which inherently holds caching ability and proxy function which is mainly focused in this paper and convert BitTorrent pieces into CCN packets, and the BitTorrent peer has the whole content and provides its pieces for the proxy node. In addition, the node also functions as a tracker.

The experimental environment and implementation details are as follows. We set up the test-bed using Oracle VM VirtualBox VM Selector v6.1.38_Ubuntu. Each node is a Ubuntu 20.04 virtual machine connected through an internal network. Both the peer-proxy link and the CCN router-CCN client link have a bandwidth limit of 100 Mbps and a latency of 10 ms. Our BitTorrent implementation is based on qBitTorrent (v4.5.5) [3]. For the CCN infrastructure, we use cefore.

The hardware specifications of the experimental machine are as follows: CPU is an Intel Xeon W-2295 with an operating frequency of 3.0GHz, consisting of 18 cores and 36 threads, and the system memory is 192GB DDR4-2933.

[3]https://www.qbittorrent.org/

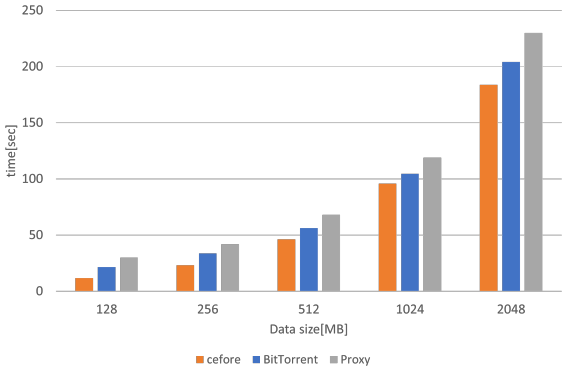

Fig. 5. Comparison of data acquisition times

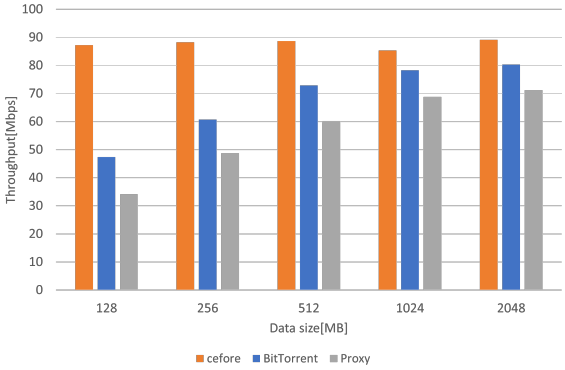

Fig. 6. Comparison of throughput

In an experiment, the CCN client parses the torrent file for the target file and requests each piece in order by an Interest packet. When the Interest packet reaches the proxy, the proxy retrieves the corresponding data from the BitTorrent peer, converts them into CCN packets, and sends them into CCN as Data packets. It should be noted that the torrent file is pre-obtained and Interest packet routing is directed towards the proxy for this experiment.

Regarding the measurement of retrieval time, the measurement starts when the CCN client issues the first Interest packet after completing the analysis of the torrent file. The measurement concludes when all the content has been retrieved and its integrity has been verified by the client. This approach provides a comprehensive measurement of the time required for complete data retrieval and verification.

For measurements, we share content sizes of 128MB, 256MB, 512MB, 1024MB, and 2048MB and take the average retrieval time over five trials.

We also measure the content retrieval time on a traditional IP network using BitTorrent, as illustrated in Figure 4, to compare it with the case using the proxy. In Figure 4, each link is set to have a bandwidth limit of 100Mbps and a latency of 10ms.

Through the experiments, we have confirmed that our proposed proxy can convert BitTorrent pieces into CCN packets and the CCN client can retrieve the content shared within the BitTorrent peers without any failure. We compare the retrieval

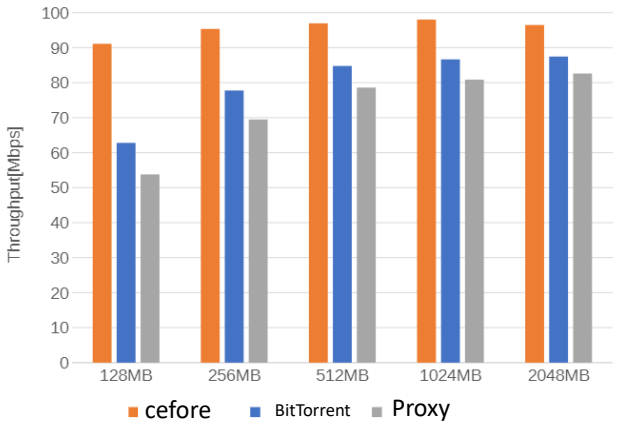

Fig. 7. Comparison of enhanced throughput

time for cefore, BitTorrent, and proxy in Figure 5. Moreover, by dividing the content size by each retrieval time, we compare the calculated throughput in Figure 6. In these figures, the legend ”proxy” corresponds to the scenario where content is retrieved from CCN through the proxy that we designed and implemented for this study. The ”BitTorrent” label represents the case where content is retrieved solely using BitTorrent over IP. Lastly, the ”cefore” label indicates the scenario where all the content is retrieved from CCN (cefore) and all necessary caches are present in the network.

From Figure 5, it can be observed that the data retrieval time is shortest when utilizing CCN’s caching, followed by traditional BitTorrent, and lastly content retrieval via the proxy. Based on retrieval times, the throughput comparison in Figure 6 shows that the throughput is consistently around 90Mbps when cache exists in CCN. In both BitTorrent and proxy-mediated data retrieval, the throughput tends to increase in proportion to the data size, especially for smaller (128MB) and larger data sizes (1024MB, 2048MB). This is because the communication overheads associated with trackers in BitTorrent and peer handshaking significantly affect the total communication time for sharing content. Additionally, the throughput performance when using a proxy is 72% of that of BitTorrent for 128MB content and 89% for 2048MB content.

Following the release of the new version of cefore and cefpyco in 2023, we have continued to advance our proxy program to augment system performance. Figure 7 illustrates the improved performance for Cefore alone, BitTorrent alone, and our proxy. As depicted in Fig. 7, the performance of cefore experiences an increase due to the version update. In addition, the performance in the topology employing our proxy is as close as that of BitTorrent alone. This can be because enhancements in our asynchronous processing in Python3 were effective to apply the new functionalities introduced in the updated cefpyco. In this case, the throughput performance achieved when utilizing our proxy is 94% of that observed

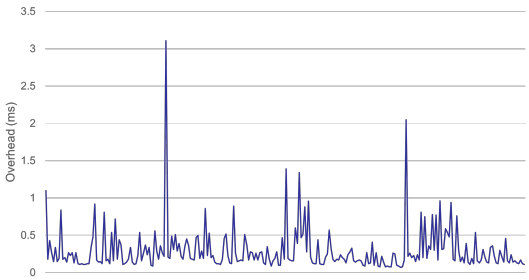

Fig. 8. Time Required for Proxy to Receive Interest and Send Data When Content Exists in Cache

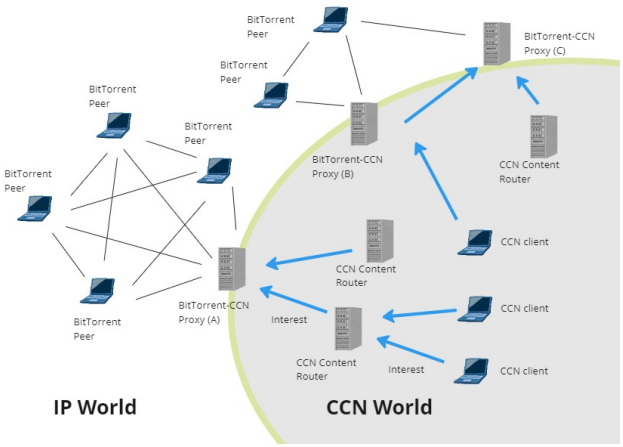

Fig. 9. Multiple instances of proposed proxies to manage a substantial influx of accesses to contents shared by BitTorrent peers

with BitTorrent for a 2048MB content.

Figure 8 shows the time elapsed from the reception of the Interest packet to the transmission of the corresponding Data packet when the proxy holds the content. This was tracked over 1000 chunks. The average time for this operation is 0.294 ms, with a standard deviation of 0.301 ms. The large standard deviation can be attributed to the presence of significant outliers. These results indicate that the time required to decode the Interest packet and return the corresponding Data is around 0.3 ms. This latency is considered negligible, especially when compared to other operational latency in networking tasks.

The experimental results demonstrate that when retrieving content shared via BitTorrent through a proxy, it is possible to achieve approximately 94% of the performance compared to traditional BitTorrent sharing. Furthermore, it was also shown that if caching exists in CCN, data retrieval can be carried out more efficiently than with BitTorrent. Based on these findings, we conclude that the BitTorrent content, which cannot be retrieved without our proposed proxy, can be successively provided for CCN world at a practically acceptable performance under the conditions our experiments investigated. Our proxy can enrich the shared contents within CCN worlds by retrieving extensive contents shared via BitTorrent in the IP world. Additionally, our proxy has the potential to mitigate redundant content delivery between Autonomous Systems (ASes) generated by BitTorrent. By introducing our proxy at the boundary of ASes to provide contents for CCN nodes within an AS and replacing BitTorrent nodes in an AS with CCN nodes, we can realize efficient content delivery by CCN cache trees.

In this paper, we have proposed a novel proxy solution designed to make shared contents on BitTorrent swarms available from CCN. We discuss access load to contents in this context. Within a P2P network like BitTorrent, access to content holders is inherently distributed across a multitude of peer nodes. Conversely, in a CCN, the burden on content holders can be alleviated, particularly as downstream caches accumulate a substantial content repository. However, when a substantial demand arises for a lot of content sourced from different BitTorrent swarms via CCN, a single instance of our proposed proxy may prove insufficient for practical application. In such scenarios, we present the option to deploy multiple instances of our proposed proxy, as illustrated in Figure 9. As depicted in Figure 9, we can establish two or more proxies, such as (A) or (C), to retrieve diverse content from BitTorrent swarms staying over the IP world. Furthermore, when the target content within a BitTorrent swarm is currently being shared and each proxy possesses only partial content, we present alternative strategies, involving simultaneous content retrieval through the upper layer of the routing tree (C) and the intermediary layer (B). Note that proxy (B) accepts interests from CCN clients and forwards them to the upper layer. Note that our current implementation does not support the functionality of an intermediate content router within CCN. It should be worth noting that multiple proxies may host duplicated content if they are independently deployed by various organizations. We do not view this as a critical issue; that must be solved by CCN researchers as an ”interest routing” issue for multiple content sources if it is challenging. However, cooperative deployment of multiple proxies stands as a prospect within our future research plan.

We plan to extend our proxy functions to adapt to Web applications. Despite the existence of previously proposed HTTP-CCN gateways [5] [6], enhancing our proxy to fetch shared content from web servers for CCN users is a valuable extension. Furthermore, considering that popular content shared by Content Distribution Networks (CDN) can be treated as identifiable copies, it is better to retrieve the contents through our proxy for CCN users in the future. In such scenarios, the conversion function currently handled by DNS (Domain Name System) should be partially addressed by our proxy.

V. Conclusion

In this study, through the design and implementation of a proxy, we successfully harnessed BitTorrent content within CCN infrastructure. We state that this technology holds the potential to enhance network traffic efficiency, lower devel-

opment and operational expenses on each CCN client, and facilitate the dissemination of larger content within the CCN world. This, in turn, addresses the scarcity of content resulting from the current limited prevalence of CCN.

The experimental evaluation also demonstrates the utility of this proxy. As shown in the experiment section, the performance of retrieving BitTorrent content via CCN reaches about 80% of the retrieval performance when using BitTorrent alone. These results suggest that the overhead introduced by the protocol conversion is within an acceptable range. Additionally, we confirmed that when caching exists within CCN, content retrieval can be more efficient than using BitTorrent alone.

The efficiency and flexibility of the proxy demonstrated in this study represent an important first step towards the coexistence and collaboration between IP and CCN. Further improvements in proxy design and algorithmic refinement are expected in the future. We plan to further extend this proxy to accommodate a wide range of network environments and content formats.

REFERENCES

[1] Cisco, "VNI Complete Forecast Highlights," https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf, 2021.

[2] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, B. Ohlman, "A Survey of Information-Centric Networking," IEEE Communication Magazine, Vol. 50, No. 7, pp. 26-36, 2012.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard, "Networking Named Content," Proc. of CoNEXT '09, pp. 1 – 12, 2009.

[4] F. Fahrianto and N. Kamiyama, "Migrating From IP to NDN Using Dual-Channel Translation Gateway," IEEE Access, vol. 10, pp. 70252-70268, Jun. 2022.

[5] S. Wang, J. Bi, J. Wu, Z. Li, "On adapting HTTP Protocol to content centric networking," Proc. of CFI '12, pp. 1 – 6, 2012.

[6] Z. Li, J. Bi, S. Wang, "HTTP-CCN gateway: Adapting HTTP protocol to Content Centric Network," Proc. of ICNP 2013, pp. 1-2, 2013.

[7] H. Asaeda, A. Ooka, K. Matsuzono, R. Li, "Cefore: Software platform enabling content-centric networking and beyond," IEICE Trans. Commun., Vol. E102-B, No. 9, pp. 1792-1803, 2019.

[8] Y. Liu, X. Piao, C. Hou, K. Lei, "A CUBIC-Based Explicit Congestion Control Mechanism in Named Data Networking," Proc. of CyberC 2016, pp. 360-363, 2016.