ICN Performance Enhancing Proxies Intended to Mitigate Performance Degradation in Global Content Delivery*

Kazuaki UEDA† and Atsushi TAGAMI‡, Members

SUMMARY  A global content delivery plays an important role in the current Internet. Information-Centric Networking (ICN) is a future internet architecture which attempts to redesign the Internet with a focus on the content delivery. However, it has the potential performance degradation in the global content delivery. In this paper, we propose an ICN performance enhancing proxy (ICN-PEP) to mitigate this performance degradation. The key idea is to prefetch Data packets and to serve them to the consumer with the shorter round trip time. By utilizing ICN features, it can be developed as an offline and state-less proxy which has an advantage of scalability. We evaluate the performance of ICN-PEP in both simulation and experiment on global testbed and show that ICN-PEP improves the performance of global content delivery.

key words: Information-Centric Networking, performance enhancing proxy, middlebox

1. Introduction

Internet traffic is forecasted to increase continuously, especially for the video traffic[2]. As the video streaming services are available in many countries, traffic demand increases rapidly. This drives the capacity update of the international backbone network. Recently many content holders, such as Google, Facebook and Microsoft, have invested in deployment of submarine cables to improve the network capacity of their platform[3], [4]. This trend means that the Internet content and services are consumed on a global scale, and efficient global content delivery will play an important role in the future Internet.

A long fat network (LFN) is defined as a network which has a large bandwidth-delay product. A global long distance network is a typical example of LFN. From the past, TCP is reported to have a performance degradation in the LFN[5]. One of the most remarkable characteristics of LFN is a long round-trip time (RTT) between end nodes. In such an environment, it takes long time to get a large congestion window size so as to utilize the available bandwidth, and this leads to low throughput. In order to mitigate such performance degradation, several solutions have been proposed, e.g. customized TCP variants[6], [7], parameter tuning of congestion control[8], and performance enhancing proxy (PEP)[9], [10]. PEP has an advantage that it does not require end nodes’ modification and can be resolved only by the intermediate node in the network. It can promote the increase in congestion window size and mitigate the negative effects of long RTTs. Proxy servers are installed on both sides or one side of LFN and terminate TCP connection by sending ACKs on behalf of the remote server and/or clients. In this case, the node can receive ACKs at a lower RTT and increase its window size quickly.

Information-Centric Networking (ICN) is a promising future Internet architecture which addresses the content delivery at the center of the communication[11], [12]. Due to its rich features such as in-network caching and name-based forwarding, ICN has advantages in terms of scalability, efficiency and availability over IP-based networks. Although ICN provides efficient content delivery, we should state that the current ICN transport layer technique has the potential performance degradation in the content delivery on a global scale. A number of ICN congestion control schemes have been proposed[13], [14]. Most of them follow the design of existing TCP congestion control schemes which control the number of request packets in flight, and thus they have a similar performance degradation in LFN.

In order to overcome such performance degradation, it is natural to convert existing solutions of TCP to ICN. However, we cannot apply an above PEP solution directly to ICN because of the difference in the communication principle between TCP and ICN. TCP follows the receiver-driven congestion control, and handling ACKs can control the sender’s congestion window size. On the contrary, ICN follows the receiver-driven congestion control, and only the Data packets can control the consumer’s congestion window size. In other words, a TCP sender can increase its window size after receiving an ACK from a proxy server. However, an ICN consumer increases its window size only after receiving a Data packet that can only be created by the origin producer. Therefore, proxy server cannot be realized in ICN unless it stores all the Data packets requested from the consumer.

The main aim of this paper is to propose and to evaluate an ICN-PEP[1] in details, which shortens the content download time for the global content delivery. A key idea of ICN-PEP is prefetching and storing only a small portion of Data packets which will be requested by consumers shortly. Even if the producer is located remotely, this can increase the congestion window size quickly without having a large amount of Data packets at ICN-PEP. To this end, we introduce the short-term prefetch technique under the assump-
This paper, we use NDN which is actively investigated from both industry and academia. In a content receiver, compared to the TCP between a producer and a consumer, i.e., a content sender and the consumer. In ICN, a simple communication is established between a producer and a consumer, i.e., a content sender and a content receiver, compared to the TCP/IP. The producer does not have any state for the communication and only replies a Data packet corresponding to a received Interest packet issued by a consumer. This pull-based communication scheme has benefits in scalability of producer and utilization of in-network cache. On the other hand, the amount of traffic is controlled by the amount of Interest packets issued by consumers. This consumer-driven congestion control is opposite to the TCP’s sender-driven congestion control.

There exist many prior studies about congestion control in ICN. The NDN team proposed a congestion control scheme named PCON [14] used in a download tool ndncatchunks [17] by default. PCON follows the TCP-like congestion control scheme which controls the congestion window in AIMD manner upon reception of Data packet. In order to detect an early stage of congestion, PCON reacts not only a packet loss event but also a congestion marker stored in the Data packet header. When a link becomes congested, the router’s queue starts to be filled with packets and queuing delay becomes longer. The congestion marker is set when the queuing delay at a router exceeds threshold, and upon reception of the marked packets, a consumer decreases the congestion window to avoid the heavy congestion.

Similar to PCON, most of the congestion control schemes in ICN changes the consumer’s congestion win-

![Fig. 1 Overall of the TCP-PEP](image)

dow upon reception of the Data packets in order to reflect the network statistics from received packets. In this design, increase speed of congestion window is directly influenced by the RTT of received Data packet. And it yields delay in the longer RTT environment, i.e., in LFN. Although consumer can retrieve the Data packets with closer intermediate router’s cache in ICN, many communication patterns cannot be satisfied with router’s cache such as downloading unpopular content, initial download for the content, and live video streaming.

### 2.2 Performance Enhancing Proxy

TCP is reported to have a performance degradation in an environment that has longer propagation delay and broader bandwidth, called as LFN [5]. International communication and satellite communication are typical examples on LFN. PEP improves performance of such communication without end nodes’ modifications. Figure 1 shows an example of PEP processing. Network operator who knows the link connected with LFN, i.e., overseas link or satellite link, installs the PEP servers at both sides or one side of the link. PEP server terminates TCP connection between a sender and a receiver and sends a pseudo ACK packets to the sender on behalf of the receiver. This allows the sender to receive ACK packets earlier than the actual RTT and to increase the congestion window size rapidly, since the window size grows up each time an ACK packet is received. In the example shown in Fig. 1, the congestion window has increased three times during one original RTT, $t_1 + t_2$.

Since PEP is a stateful solution, it has scalability issues. PEP must keep track of all sessions and perform packet relaying, packet retransmission and flow control for them. Inline proxy, which is placed in the middle of the communication path, is frequently used in the commercial networks since it can work without any modifications of the end nodes. However, such proxy has to monitor all incoming packets and pick up appropriate packets to be accelerated by the proxy. And it has a severe problem that a proxy failure leads to stop the entire communication. As mentioned above, although PEP is a powerful solution for LFN, it has challenges in scalability. We can say that ICN has a potential to improve communication performance at LFN with PEP while overcoming these performance issues of PEP with its rich network features such as in-network caching and name-based forwarding.
3. ICN Performance Enhancing Proxy

3.1 Design Overview

In this section, we design the ICN-PEP that improves the ICN communication performance in the LFN. The essential point of the current TCP-PEP is to increase sender’s congestion window size rapidly by handling ACKs on behalf of the receiver. We divert this point to the consumer-driven ICN communication. In order to increase consumer’s congestion window size rapidly, ICN-PEP prefetches Data packets which will be requested in the next round and stores them in its cache. Since these packets are cached at closer location to the consumer, consumer can retrieve these packets in a shorter RTT, and its congestion window size can be increased rapidly.

Figure 2 shows a segmentation and a naming process at a producer. ICN name consists of routable name, content name, and protocol-dependent name like segment number and version number [18]. Due to the limitation in payload size of Data packet, a content is divided into multiple segments. Each segment is identified by a segment number in its name starting from 0 and incrementing by 1. When downloading a content, a consumer first fetches a Data packet whose segment number is 0 and continues to fetch Data packets sequentially. After fetching a Data packet of the final segment, the consumer reassembles the whole content from them.

We use such sequential communication pattern and naming of ICN for prefetching Data packets which will be requested in the next round. ICN-PEP monitors Interest packets forwarded to a LFN link, and estimates Data packet’s name requested in the next time from the continuity of segment number. Based on the above estimation, ICN-PEP creates and issues next Interest packets instead of the consumer. Although consumer retrieves a first Data packet from an origin producer, the next Interest packets are satisfied from in-network cache, and thus RTT of these Interest packets can be dramatically reduced.

Figure 3 illustrates an example of the proposed ICN-PEP’s sequence diagram. ICN-PEP is installed on the consumer side of LFN. \( t_1 \) is an RTT from the consumer to ICN-PEP, and \( t_2 \) is an RTT from ICN-PEP to the producer. Upon receipt of an Interest packet whose segment number is \( i \), ICN-PEP creates and forwards Interest packets with segment number \( i + 1 \) to \( i + N \). The consumer receives a first Data packet with segment number \( i \) in \( t_1 + t_2 \), but next Data packets (\( i + 1 \) to \( i + N \)) are already stored in the cache. Thus, next Interest packets are satisfied with the cache and RTT for these packets is reduced to \( t_1 \).

The most remarkable point of this design is that ICN-PEP does not manage any state. It only prefetches Data packets according to the received Interest packet and stores the response Data packets in its cache.

3.2 Offline Proxy with Name-Based Forwarding

As described in Sect. 2.2, TCP-PEP is installed as an inline proxy and has scalability issues in forwarding and monitoring all packets. Moreover, inline proxy has a performance degradation by its processing delay and becomes a single point of failure in the system. By making use of the ICN features, we can set up the ICN-PEP as an offline proxy which can avoid these issues.

Figure 4 shows the network configuration of the offline ICN-PEP. We assume that the content which requires the PEP processing can be identified by its name pre-
In this paper, we assume the scenario where LFN link is a bottle neck. In order to shorten the duration of slow start phase, ICN-PEP only prefetches first $N_s$ Data packets to each consumer. And it does not prefetch the Data packets whose segment number exceeds $N_s$ to ensure fairness among large content download and small content download. The main point of acceleration strategy is how to determine $N_s$. $N_s$ needs to be sufficient to increase consumer’s congestion window, but setting too large value results in congestion.

The aim of our strategy is to prefetch enough Data packets to complete the consumer’s slow start phase. We assume that ICN-PEP knows the link capacity $w$ and the propagation delay $d$ of the LFN link. These parameters are configured by the network operator or retrieved by the measurement, and ICN-PEP can calculate the bandwidth-delay product of the LFN link. ICN-PEP also retrieves the information of available bandwidth from the network nodes so as not to put too much load with its prefetch traffic $N_s$. When LFN is the bottleneck, the theoretical congestion window size $c_t$ is as follows:

$$c_t = \alpha (w/8s) 2d$$

where $s$ is the maximum Data packet size in Bytes and $\alpha$ means a filling factor, i.e., how many flows are sharing the LFN link. In the slow start phase, the congestion window size doubles when receiving the same number of Data packets as the current window size. Assuming that the initial congestion window size is $c_1$, the maximum step number $m$ for the congestion window size which does not exceed $c_t$ is derived as follows:

$$m = \lceil \log_2 (c_t/c_1) \rceil.$$  \hspace{1cm} (1)

Then, the number of Data packets $N_s$ required to complete the slow start phase is:

$$N_s = c_1 \sum_{i=1}^{m} 2^i.$$  \hspace{1cm} (2)

When ICN-PEP receives an Interest packet with segment number 0, i.e., the consumer starts the slow start phase, it requests the next $N_s$ Data packets instead of the consumer. When $s \times N_s$ is larger than the current available bandwidth, ICN-PEP should decrease $N_s$. Detailed protocols for retrieving available bandwidth and adjusting $N_s$ to its value are our future works.

### 4. Evaluation

#### 4.1 Evaluation Environment

We evaluated an ICN-PEP performance with both simulation and experiments over the ICN global testbed. Figure 5 shows the evaluation environment which we used in the evaluation. This network topology consists of three ICN routers at Japan and three ICN routers at Europe, and this is constructed on the global ICN testbed service named CUTEi [20]. A link between routers in Osaka and Rome is
UEDA and TAGAMI: ICN PERFORMANCE ENHANCING PROXIES INTENDED TO MITIGATE PERFORMANCE DEGRADATION IN GLOBAL CONTENT DELIVERY

Fig. 5 Evaluation environment constructed on the CUTEi testbed

### Table 1

| Parameter                  | ndncatchunks | ConsumerPcon |
|----------------------------|--------------|--------------|
| Initial congestion window size ci | 10           | 10           |
| Data packet payload size s   | 4000 byte    | 1024 byte    |
| # of consumers 1/α          | 1            | 4            |

an overseas link and we assume this as an LFN link. We measured the average bandwidth and average RTT of each link in the testbed which are illustrated in Fig. 5. We use half of measured average RTT as the approximated value of propagation delay \( d \) in this evaluation. We constructed and used the same environment in the simulation.

Consumers and producers were placed at Japan and Europe respectively, and ICN-PEP is installed on an ICN router in Osaka. The consumer is standard implementation, \( ndncatchunk \) for experiments and \( ConsumerPcon \) for simulation evaluation on ndnSIM [21], without any modifications. The number of prefetched Data packets \( N_s \) is calculated by Eqs. (1) and (2) with measured parameters, i.e., propagation delay \( d = 0.1 \) sec the link capacity \( w = 50 \) Mbps, and consumer’s parameters shown in Table 1. The filling factor \( \alpha \) is the reciprocal of the flow number in evaluation scenario. Under these parameters, \( m \) and \( N_s \) have the same value in both cases as follows:

\[
m = 4, N_s = 300.
\]

In order to investigate the influence of \( N_s \), various \( N_s \) values, e.g. \( N_s = 60 \) \( (m = 2) \), 620 \( (m = 5) \), are also used in the following evaluation.

4.2 Performance Analysis with Simulation

First, we evaluated the impact of \( N_s \) for the performance of downloading content with simulation. The evaluated topology and link parameters are the same as those shown in Fig. 5. Four consumers and four producers are connected to the edge routers in Japan and Europe, respectively. Two consumers start to download at \( t = 0 \) and the others start at \( t = 2 \) sec. Each consumer downloads different content from the different producers.

Figure 6 shows the average goodput of consumers with various content size. ICN-PEP always improves the goodput of communication. Especially when the content size is small, the gain becomes large. This is because the ratio of slow-start phase duration is higher when downloading small content than when downloading large content. When the content size is 20MB or larger, the slope of the graph is almost equal. The gain is stable in this area, and it achieves about 35% gain with \( N_s = 300 \).

Figure 7 shows the congestion window size of a consumer which starts downloading at \( t = 2 \) sec. Without ICN-PEP, it has to wait an elapse of 1 RTT for producer, i.e., 250 msec, to get each Data packet. On the contrary, as designed in Sect. 3.1, ICN-PEP provides the small RTT to the consumer with the prefetched Data packets, and the consumer increases its congestion windows size rapidly. Although \( N_s = 60 \) is not enough to complete the slow start phase, when we use \( N_s = 300 \) or more, slow start phase is completed quickly. This is also reflected in the performance difference shown in Fig. 6. With \( N_s = 620 \), the average goodput is lower than that of \( N_s = 300 \). When \( N_s = 620 \), the congestion window size exceeds the theoretical congestion window size \( c_t \), and the congestion is occurred by the prefetched Data packets.

Figure 8 shows the fairness among the four consumers. We used Jain’s fairness index [22] obtained from

\[
\text{Fairness Index} = \left( \frac{\sum_{i} x_i}{k \sum_{i} x_i^2} \right)^2
\]

where \( k \) and \( x_i \) are the number of consumers and the goodput of the consumer \( i \), respectively. The fairness index is clearly reduced when \( N_s = 620 \). This is because that the congestion occurred from too large congestion window, and
Table 2  Average download time until the start-up buffer is full

| Buffer size $n$ | 2 sec | 5 sec | 10 sec |
|-----------------|-------|-------|--------|
| w/o ICN-PEP     | 1944.2 ms | 2998.5 ms | 5139.4 ms |
| ICN-PEP ($N_s=60$) | 1338.2 ms | 2204.1 ms | 4359.0 ms |
| ICN-PEP ($N_s=300$) | 1158.6 ms | 2092.2 ms | 4181.0 ms |

The results of our evaluation clearly show that ICN-PEP improves the communication performance on LFN, and that the acceleration strategy described in Sect. 3.3 can determine the appropriate number of the prefetched Data packets $N_s$.

4.3 Experiments of Realistic Use Case with CUTEi Testbed

As a realistic use case, we tried the PEP-enabled video streaming experiments over the CUTEi testbed. In this experiment, we focus on reducing start-up delay of video streaming in order to improve user’s QoE with ICN-PEP. We constructed a simple linear topology that connects one consumer at Tokyo, one ICN-PEP at Osaka, and one producer at Rome shown in Fig. 5. We used the video content that is open to the public at [23], and we fixed video resolution to 480p in order to ignore the influence of the rate adaptation. The video content is divided into multiple fragmented files and stored on the producer. The consumer downloads these fragmented files sequentially, and the video client starts to play video after $n$ files are downloaded in order to prevent the video playback from stalling.

We used ndnputchunks and ndncatchunks [17] as the producer and the consumer, respectively. The start-up buffer size $n$ is configured to increase robustness of video streaming in case of the throughput fluctuation. In this evaluation, we used $n = 2$ sec (4 MB), 5 sec (10 MB) and 10 sec (20 MB).

Table 2 shows the average download time until the start-up buffer is full. ICN-PEP can reduce the start-up delay of video streaming around 15-40%. Although this result is slightly inferior to the simulation result, it shows almost the same tendency on the relationship with the content size and with the number of prefetched Data packets $N_s$. This result shows the benefit of ICN-PEP on the real LFN environment. Figure 9 shows the congestion window size at the consumer, and it also shows the similar behavior to that in the simulation.

5. Discussion

5.1 Avoiding Self-Induced Congestion at ICN-PEP

ICN-PEP introduces additional traffic into the network, and we should carefully design not to cause severe congestion with prefetch. As designed in the Sect. 3.3, ICN-PEP obtains the information of available bandwidth of LFN link from the network node and performs prefetch when the prefetch data rate is within the available bandwidth. However, even if ICN-PEP performs prefetch in consideration of the available bandwidth, burst traffic may create temporary congestion at the bottleneck link. To avoid temporary congestion from the prefetch traffic itself, we add pacing feature to the ICN-PEP that takes advantage of the characteristics of prefetch. ICN-PEP does not need to forward $N$ Interest packets at once. As shown in the Fig. 10, the prefetch of $N_s$ Data packets should be finished within $t_2 + m \times t_1$. Therefore, a last Interest packet for the prefetch should be forwarded within $m \times t_1$. By scheduling the forwarding Interest packets evenly, the prefetch traffic can be smoothed to

$$\frac{N_s}{m \times t_1} \text{(pps)}.$$ 

In our evaluation setting, where $t_1 = 10$ ms, $m = 4$, $N_s = 300$, and payload size is 1 KB, prefetch data rate is 7.5 Mbps. This rate is also used to determine whether prefetch can be performed by comparing it to the available bandwidth.

5.2 Analyzing Scalability and Requirement of ICN-PEP

5.2.1 Scalability Analysis of ICN-PEP

As stated in the Sect. 3.2, scalability is one of the benefit of ICN-PEP. To validate this benefit, we analyze the scalability and requirement of ICN-PEP. First, we describe the
ICN-PEP satisfies the users’ Interest packets from the prefetched Data packets stored in its cache. Therefore, if its cache size is too small, the prefetched Data packets may be replaced, and ICN-PEP cannot satisfy Interest packets. However, the requirement can be relaxed since ICN-PEP stores prefetched Data packets only for the limited duration defined by the \( m \times t_1 \) (Fig. 10). Where \( m \) is the step number defined in 1 and \( t_1 \) means a RTT from consumer to ICN-PEP. The requirement of ICN-PEP’s cache size \( S_c \) in byte can be roughly estimated as follows:

\[
S_c = \frac{m \times t_1 \times L}{8}
\]

where \( L \) means the link speed in bps which represents the maximum incoming traffic of ICN-PEP, and we use FIFO for cache replacement logic. In the case where \( L = 40 \text{Gbps}, \ t_1 = 20 \text{ms}, \) and \( m = 4, \ S_c \) should be larger than 400 MB. Therefore, ICN-PEP can be realized with reasonable memory size.

5.3 Feasibility of Consumer-Based Approach

5.3.1 Applying LFN-Specific Parameters

In this paper, we focused on the middlebox-based approach, ICN-PEP, for improving performance at LFN. On the other hand, modification of congestion control parameters such as increasing initial congestion window size \( c_i \) will be an alternative approach. And this may work similar to execute prefetch at the consumer. However, such approach has a fundamental challenge due to the communication principle of ICN. In ICN, content can be delivered anywhere of the network, and the location of content is essentially unknown to consumers. When the content is stored at the nearby cache or server, setting large initial congestion window yields too much traffic to network and results in unfair bandwidth allocation.

To verify the feasibility of this approach, we evaluated the performance impact of initial congestion window size for downloading content over the both local and global network with simulation. We used the same network environment shown in Fig. 5 as the global network and used the...
same one except the RTT between Osaka and Rome is 20 ms as the local network. The simulation scenario is almost same as one described in Sect. 4.2, i.e., two consumers start downloading at $t = 0$ and others start downloading at $t = 1$ sec. Figure 11 illustrates the impact of the initial congestion window size for the average goodput of downloading content. Although setting large initial congestion window can increase the goodput in global network, it has negative effect on the goodput in local network. In the global network, increase speed of the congestion window size is very slow, and setting large initial congestion window size can shorten the time to get enough congestion window size for the available link capacity. However, in the local network, the congestion window size increases rapidly, and the large initial congestion window size creates aggressive traffic which results in congestion and decreasing goodput. Figure 12 shows the fairness index of this evaluation. When we set $c_i \geq 50$, although fairness index is stable in the global communication, it starts to decrease in the local communication. And we see relatively high average goodput in global communication with $c_i = 30$, but its low fairness index indicates that this is caused by the unfair bandwidth allocation.

As shown in the above results, setting large $c_i$ has negative performance impact in case of misuse for the local communication. Following the ICN principle, we can say that it is hard for consumer to use parameters specific to the location of the content, which is unknown to the consumer. On the contrary, our middlebox-based approach, ICN-PEP, can perfectly identify the traffic for LFN, and it is more suitable for ICN.

5.3.2 Impact of Congestion Control Algorithm

In the Sect. 4, we used a simple AIMD-based congestion control algorithm, but using congestion control algorithm which is more resilient to the LFN is also an alternative approach. We evaluated the performance of CUBIC in our simulation to see the impact of congestion control algorithm. In TCP/IP, CUBIC is reported to achieve higher throughput in LFN [25]. We changed the consumer’s congestion control algorithm from AIMD to CUBIC and evaluated the same scenario of Sect. 4.2 with simulation. Figure 13 shows the average goodput of this simulation, and the results of AIMD are also plotted as a baseline. We can see that CUBIC with NDN can achieve higher throughput in the LFN compared to the AIMD. When we disabled ICN-PEP, CUBIC achieves almost 38% goodput improvement compared to the AIMD. Moreover, we can see that ICN-PEP can improve goodput of the CUBIC. CUBIC with $N_s = 300$ can improve goodput around 9.9-72.3% compared to the CUBIC without ICN-PEP. ICN-PEP can provide more performance gain when downloading small content, and this is the same trend in the Sect. 4.2.

This result shows the possibility to improve the performance in the LFN by using more sophisticated congestion control algorithms in NDN, such as CUBIC. However, even if we use CUBIC, it takes several round-trips to get sufficient congestion window in the longer RTT environment, and this results in lower goodput in the small content download. ICN-PEP, which executes prefetch as an in-network function of LFN, can work complementarily with the sophisticated algorithm. It can reduce download time even for the small content download. By using ICN-PEP together with CUBIC, we can achieve higher goodput in downloading both small content and large content.

5.4 Security Considerations of ICN-PEP

As shown in Sect. 3.1, ICN-PEP creates N Interest packets for prefetch. When the segment numbers of generated Interest packets are larger than the end of file, which is defined as finalSegmentNumber in NDN [18], these Interest packets request non-existent Data packets. This causes redundant.
packet transmission which is known as DDoS attack in ICN, interest flooding attack [26]. Since ICN producer responds such Interest packets with NACK packets, this does not create any wrong states at ICN routers. However, this yields additional processing at producer and routers, and thus we should avoid such occasion as much as possible. There are three countermeasures for this issue. First solution currently supported is the restriction of $N_s$. As described, current acceleration strategy only prefetches first $N_s$ Data packets. In our evaluation, we used $N_s = 300$ which means the total amount of prefetched data size is about 300KB when the payload size is 1024 byte. For the general content such as video or image file, $N_s$ is smaller than the finalSegmentNumber, and thus it does not create redundant packets. Second solution is the name-based filtering and estimation of $N_s$. There are two name-based filters in the ICN-PEP system; the first is the ICN router and the second is ICN-PEP. In our offline proxy configuration, only the traffic associated with a particular name prefix is routed to the ICN-PEP. And we can set the detailed filename-based filtering to estimation module of ICN-PEP, e.g., only the video file "*.mp4" is accepted as a prefetch target. Moreover, we can estimate the appropriate $N_s$ from filename. In NDN, finalSegmentNumber is stored in the packet header of Data packet. Therefore, by using the received finalSegmentNumber, ICN-PEP can learn the optimal segment number for the filename and update the estimation module. A third solution is to use the manifest [27], [28] to input the correct name information into the ICN-PEP. Manifest is a type of the Data packet which contains the list of content name and is created by the producer. In the manifest-based content retrieval, a consumer first requests the manifest for producer in order to get the whole list of names. Upon reception of a manifest, consumer issues Interest packets for the names listed in it. By using the manifest, ICN-PEP can create the prefetch Interest packets whose names are correctly handled by producer.

6. Conclusion

In this paper, we revealed the potential performance degradation in the global content delivery over ICN due to its congestion control design. To mitigate such performance degradation, a PEP is a well-known solution which does not require the end nodes’ modification. However, ICN follows a pull-based communication model and its congestion control is mainly performed by a consumer, unlike the current TCP/IP. Therefore, PEP cannot be directly applied to ICN. We proposed an ICN-PEP which is a first PEP designed for ICN. The key idea of ICN-PEP is to prefetch Data packets that will be requested from consumers in the next rounds. ICN-PEP can accelerate the growth of the consumer’s congestion window size by serving their requests at closer location. Additionally, by using ICN features, we can put ICN-PEP as an offline proxy which is beneficial for scalability and fault tolerance. We evaluated the performance of ICN-PEP with both simulation and real testbed. The evaluation showed about 35% goodput improvement on the simulation and 15 - 40% reduction in video buffering time on the real testbed. This study shows the feasibility and usefulness of ICN-PEP.

References

[1] K. Ueda and A. Tagami, “ICN-PEP: ICN Performance Enhancing Proxy for the Efficient Global Content Distribution,” 2019 IEEE Global Communications Conference (GLOBECOM), pp.1–6, Dec. 2019.
[2] Cisco, “Cisco Visual Networking Index: Forecast and Trends, 2018 – 2023.” https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/ white-paper-c11-741490.html. Accessed: 11-Dec-2020.
[3] Z.S. Bischof, R. Fontugne, and F.E. Bustamante, “Untangling the world-wide mesh of undersea cables,” Proc. ACM Workshop on Hot Topics in Networks, pp.78–84, Nov. 2018.
[4] “Global internet map 2018.” https://www2.telegeography.com/global-internet-map. Accessed: 11-Dec-2020.
[5] D. Borman, B. Braden, V. Jacobson, and R. Scheffenegger, “Tcp extensions for high performance,” RFC 7323, IETF, Sept. 2014.
[6] S. Floyd, “Highspeed tcp for large congestion windows,” RFC 3649, IETF, Dec. 2003.
[7] C. Caini and R. Firrincieli, “TCP Hybla: a TCP enhancement for heterogeneous networks,” International journal of satellite communications and networking, vol.22, no.5, pp.547–566, Sept. 2004.
[8] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin, “An argument for increasing TCP’s initial congestion window,” ACM SIGCOMM Computer Communication Review, vol.40, no.3, pp.26–33, June 2010.
[9] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, “Performance enhancing proxies intended to mitigate link-related degradations,” RFC 3135, IETF, June 2001.
[10] C. Caini, R. Firrincieli, and D. Lacamera, “PEPsala: a Performance Enhancing Proxy designed for TCP satellite connections,” 2006 IEEE 63rd Vehicular Technology Conference, pp.2607–2611, May 2006.
[11] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard, “Networking named content,” Proc. ACM International Conference on emerging Networking EXperiments and Technologies, pp.1–12, Dec. 2009.
[12] G. Xylomenos, C.N. Ververidis, V.A. Siris, N. Fotiou, C. Tsiropoulos, X. Vasilakos, K.V. Katsaros, and G.C. Polyzos, “A survey of information-centric networking research.,” IEEE Communications Surveys and Tutorials, vol.16, no.2, pp.1024–1049, July 2014.
[13] G. Carofiglio, M. Gallo, and L. Muscariello, “Optimal multipath congestion control and request forwarding in information-centric networks: Protocol design and experimentation.” Computer Networks, vol.110, pp.104–117, Oct. 2016.
[14] K. Schneider, C. Yi, B. Zhang, and L. Zhang, “A practical congestion control scheme for named data networking,” Proc. ACM Conference on Information-Centric Networking, pp.21–30, Sept. 2016.
[15] “Named-data networking.” http://named-data.net/. Accessed: 11-Dec-2020.
[16] “CICN.” https://wiki.ifi.uio.no/view/Cicn. Accessed: 11-Dec-2020.
[17] “ndn-tools.” https://github.com/named-data/ndn-tools/. Accessed: 11-Dec-2020.
[18] Y. Yu, A. Afanasyev, Z. Zhu, and L. Zhang, “NDN technical memo: Naming conventions,” Tech. Rep. NDN-0022, Named Data Networking, Oct. 2019.
[19] A. Afanasyev, J. Shi, B. Zhang, and L. Zhang, “NFD developer’s guide,” Tech. Rep. NDN-0021, Named Data Networking, Oct. 2016.
[20] H. Asaeda, R. Li, and N. Choi, “Container-based unified testbed for information-centric networking,” IEEE Network, vol.28, no.6,
[21] S. Mastorakis, A. Afanasyev, and L. Zhang, “On the evolution of ndnSIM: An open-source simulator for NDN experimentation,” ACM SIGCOMM Computer Communication Review, vol.47, no.3, pp.19–33, Sept. 2017.

[22] R.K. Jain, D.M.W. Chiu, and W.R. Hawe, “A quantitative measure of fairness and discrimination,” Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA, Jan. 1984.

[23] S. Lederer, C. Müller, and C. Timmerer, “Dynamic Adaptive Streaming over HTTP Dataset,” Proc. ACM Multimedia Systems Conference, pp.89–94, Feb. 2012.

[24] C. Ghasemi, H. Yousefi, K.G. Shin, and B. Zhang, “A fast and memory-efficient trie structure for name-based packet forwarding,” 2018 IEEE 26th International Conference on Network Protocols (ICNP), pp.302–312, IEEE, Sept. 2018.

[25] S. Trivedi, S. Jaiswal, R. Kumar, and S. Rao, “Comparative performance evaluation of tcp hybla and tcp cubic for satellite communication under low error conditions,” 2010 IEEE 4th International Conference on Internet Multimedia Services Architecture and Application, pp.1–5, IEEE, Dec. 2010.

[26] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, “Interest flooding attack and countermeasures in Named Data Networking,” 2013 IFIP Networking Conference, pp.1–9, IEEE, May 2013.

[27] C. Tschudin, C.A. Wood, M. Mosko, and D.R. Oran, “File-Like ICN Collections (FLIC),” Internet-Draft draft-irtf-icnrg-flic-02, IETF, Nov. 2019.

[28] J. Kurihara, E. Uzun, and C.A. Wood, “An Encryption-Based Access Control Framework for Content-Centric Networking,” Proc. IFIP Networking 2015, Toulouse, France, pp.1–9, Oct. 2015.

Kazuaki Ueda received the B.E. degree in Electrical and Electronic Engineering and the M.E. degree in Informatics from Kyoto University, Japan in 2011 and 2013, respectively. He joined KDDI Corp. in 2013. He is currently a research engineer at Future Communication System Laboratory, KDDI Research, Inc. His research interests include Information-Centric Networking and emerging network protocols.

Atsushi Tagami received the M.E. and Ph.D degrees in Computer Science from Kyushu University, Japan in 1997 and 2010, respectively. He joined KDDI R&D Laboratories Inc. in 1997, where he has been engaged in research and development on Performance Measurement of Communication Networks and Overlay Networking. He is currently a senior manager at Future Communication System Laboratory of KDDI Research, Inc. He received the Excellent Paper Award from IEICE in 2015.