future internet

MDPI

Article

Request Expectation Index Based Cache Replacement Algorithm for Streaming Content Delivery over ICN

Haipeng Li [1,*], Hidenori Nakazato [2] and Syed Hassan Ahmed [3]

[1] Graduate School of Global Information and Telecommunication Studies, Waseda University, 66 bldg., 3-14-9 Ookubo, Shinjuku-ku, Tokyo 169-0072, Japan
[2] Department of Communications and Computer Engineering, Waseda University, 66 bldg., 3-14-9 Ookubo, Shinjuku-ku, Tokyo 169-0072, Japan; nakazato@waseda.jp
[3] Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816, USA; s.h.ahmed@ieee.org
* Correspondence: freebirdlee@ruri.waseda.jp; Tel.: +81-3-5286-2667

Received: 2 October 2017; Accepted: 8 November 2017; Published: 14 November 2017

Abstract: Since the content delivery unit over Information-Centric Networking (ICN) has shifted from files to the segments of a file named chunks, solely either file-level or chunk-level request probability is insufficient for ICN cache management. In this paper, a Request Expectation Index (RXI) based cache replacement algorithm for streaming content delivery is proposed. In this algorithm, RXI is introduced to serve as a fine-grained and unified estimation criteria of possible future request probability for cached chunks. RXI is customized for streaming content delivery by adopting both file-level and chunk-level request probability and considering the dynamically varied request status at each route as well. Compared to prior work, the proposed algorithm evicts the chunk with the minimum expectation of future request to maintain a high cache utilization. Additionally, simulation results demonstrate that the RXI-based algorithm can remarkably enhance the streaming content delivery performance and can be deployed in complex network scenarios. The proposed results validate that, by taking fine-grained request probability and request status into consideration, the customized in-network caching algorithm can improve the ICN streaming content delivery performance by high cache utilization, fast content delivery, and lower network traffic.

Keywords: information centric networking; in-network caching; cache replacement; popularity

1. Introduction

Currently, the main usage of the Internet is gradually shifting from connection-driven communications, such as end-to-end conversations, to information-driven communications, e.g., content broadcasting and retrieval [1]. To cater to this trend, information oriented future internet architectures have been proposed, such as Data-Oriented Network Architecture (DONA), Publish–Subscribe Internet Technologies (PURSUIT), Network of Information (NetInf), Content-Centric Networking (CCN) and Named Data Networking (NDN) [2–8]. The approach of these architectures is commonly called Information-Centric Networking (ICN) [9,10].

ICN enables transferred information (content) to be named uniquely at the network layer so that the named content can be forwarded directly without any location information, namely the IP address. This mechanism favors the deployment of in-network caching, content multicasting, content-level security, etc. Accordingly, ICN is capable of facilitating the efficiency of data transmission from the source to the end users.

To further enhance the efficiency of content delivery, in-network caching is adopted, which has become a distinct research field of ICN [11,12]. The named content in ICN can not only be transferred but also be temporarily stored in each network device for directly responding to the potential

subsequent requests. Precisely, in-network caching studies focus on the management of cached content and aim to shorten the delivery distance between the content source and requesters to enhance the performance of content delivery [13]. Therefore, an appropriate cache management scheme can significantly improve the network performance.

Cache management schemes can be classified into cache decision schemes and cache replacement algorithms. The cache decision scheme focuses on choosing appropriate caching locations in the network for particular contents to reduce the cache redundance and to improve the cache utilization. Recent cache decision studies mainly depend on the content popularity [14–19], node features, such as node capability and importance [20–23], and cache coordination [24–26]. In addition, a per-fetching based cache decision scheme specifically for adaptive video streaming is proposed in [27], which aims to maintain high quality video playback to the clients.

A cache replacement algorithm is necessary for each ICN router when the cache decision scheme is processed. For instance, Leave Copy Everywhere (LCE) and Least Recently Used (LRU) are adopted to work cooperatively as the default cache management schemes of NDN, as shown in Figure 1. The cache replacement algorithm for ICN refers to the process that evicts a selected piece of content in the storage space to make room for incoming content. Cache replacement is not a fresh issue raised by ICN. For memory management in computers, cache replacement algorithms are designed to improve memory page utilization. LRU, Least Frequently Used (LFU), First-In and First-Out (FIFO) and the Optimal replacement algorithm (OPT) [28] (also named Belady's algorithm) have been proposed to overcome the access speed difference between the slow auxiliary storage devices and the fast cache memory. Among these cache replacement algorithms, the OPT algorithm is the most inspiring one and has been proven to be theoretically optimal [29,30]. The OPT algorithm evicts the item that will not be accessed for the longest period of time. However, this theoretically optimal algorithm is rarely implementable because it is difficult to predict when the cached items will be accessed in the future. In addition, the cache replacement issues for web caching scenarios are well studied [31,32]. These studies can be classified as recency-based replacement (LRU, HLRU, Pyramidal Selection Scheme (PSS), etc.), frequency-based replacement (LFU, LFU-Aging, etc.), function-based replacement (Greedy Dual (GD)-Size, Taylor Series Prediction (TSP), etc.) and randomized-based replacement (RAND, etc.) [33]. Although these cache replacement algorithms are not aiming for ICN, they are excellent references for designing the ICN cache algorithm.

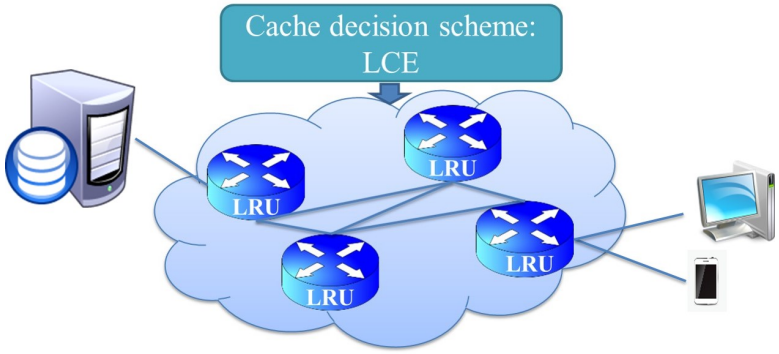

Figure 1. Default cache management scheme in Named Data Networking (NDN). LCE: Leave Copy Everywhere; LRU: Least Recently Used.

For the ICN scenario, the performance of the well-known cache replacement algorithms, e.g., LRU, LFU and FIFO, have been analyzed in [34,35], and new cache replacement algorithms for ICN have been proposed. A dynamic classification based fast convergence caching replacement algorithm is proposed [36], which develops a dynamic classification method to reduce the time complexity of cache inquiry. The CLOCK mechanism based compact Clock with Adaptive Replacement (CAR) algorithm is introduced to achieve low-complexity approximation and reduce the consumption of

cache resources [37]. However, these studies mentioned above concentrate on distributing general independent contents. For a set of organized contents such as video streams, for which requests have particular behavior, the cache scheme has to be reconsidered.

1.1. Motivation and Contributions

In-network caching specifically for streaming delivery (e.g., video delivery, audio broadcasting and file distribution) over ICN architectures raises new challenges. Firstly, the cached object has a different granule size. This variation leads to inadequacy in simply extending the existing file-oriented caching research results to ICN caching studies, such as the popularity of cached objects and the request feature [38]. Additionally, for streaming content delivery over ICN, the chunk request sequence for a file is well ordered rather than generated independently. Thus, the access probability among chunks in the same file, which we call chunk-level internal probability, is traceable and calculable once the file is requested. In the case of streaming content delivery over ICN, the correlation in the occurrences of requests for chunks in the same file needs to be considered. A cache replacement algorithm for layered video content is proposed [39]. However, the experiment is conducted on a straightforward and linear network with five nodes, thus the scalability of this proposal is unclear. Two-level popularity oriented time-to-hold policy (TLP-TTH) [40] also adopts the two levels of popularity and future requests prediction. Nevertheless, the chunk-level internal probability is not included, which may further improve the streaming content delivery performance.

In this study, to further improve the ICN streaming content delivery performance of previous studies, we propose the RXI-based cache replacement algorithm, which aims to enhance cache utilization and to reduce the content retrieval time and unnecessary network traffic. RXI considers both the file-level and chunk-level internal request probability and estimates the possible future request probability according to the dynamically varied request status at each ICN router. Subsequently, the RXI-based algorithm evicts the cached chunk that has the smallest request expectation in the future in order to keep a high cache utilization. Moreover, the RXI based algorithm avoids the additional management overhead, such as messages or control packet transmission, which increases the network burden to some extent. We have evaluated the proposed algorithm compared with LRU, LFU, and TLP-TTH in both a hierarchical topology and a hybrid topology. The results indicate that, in both scenarios, the RXI-based algorithm improves several aspects of network performance and can be deployed in complexity network scenarios.

1.2. Organization

This paper is organized as follows. We introduce the scenario of in-network caching for streaming content delivery and our assumptions in Section 2. Then, we present the RXI and RXI-based cache replacement algorithm in Section 3. In Section 4, we evaluate the RXI-based algorithm in two scenarios in comparison with the LRU, LFU and TLP-TTH. Finally, a summary and the conclusions of this study are presented in Section 5.

2. Scenario Description and Assumptions

2.1. In-Network Caching for Streaming Content Delivery Scenarios

Content transmission in ICN is performed through two types of packets: request and data. The request packet is designed for requesting a target chunk by the chunk name, and the data packet is for delivering the chunk back to the requester. Unique chunk name is used for chunk-by-chunk communications in ICN. Generally, the name structure used in ICN is hierarchical, e.g., /Prefix/FileName/ChunkName, and streaming content such as a video file is typically structured into chunks, which are transferred in sequence.

The in-networking caching for streaming content delivery is illustrated in Figure 2. Once the ICN router receives a request, the cache space is traversed according to the chunk name carried by

the request to locate the target chunk. If the target chunk exists in the cache space, it is forwarded to the client directly, which results in a cache hit. Otherwise, according to the forwarding strategies, the request is forwarded to other routers, which leads to a cache miss in this router. Thus, every ICN router in the network is able to act as a content producer for returning the data chunk directly. For the ICN router receiving a data chunk, whether to cache is determined according to a cache decision algorithm, and if the cache space is exhausted, the cache replacement process will be triggered. The chunk selected by a cache replacement algorithm is evicted to make room for incoming chunks.

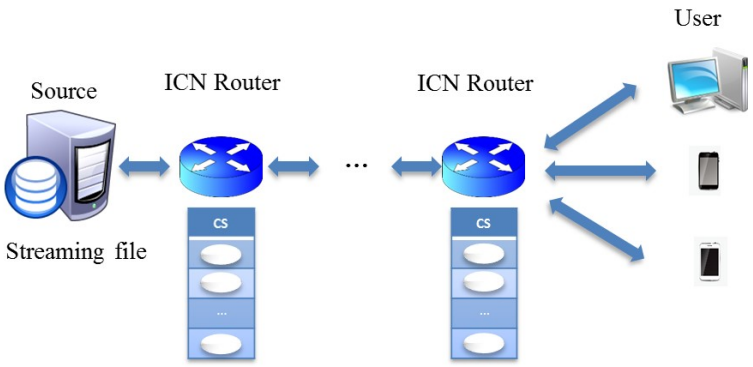

Figure 2. In-network caching for a streaming content delivery scenario. ICN: Information-Centric Networking; CS: Cache Space.

An in-network cache replacement algorithm is designed not only to identify an evictee to make room for new chunks but also to rationally manage the cache space globally to enhance the network performance in terms of the cache hit ratio, chunk delivery distance, retrieval time, etc.

2.2. Definitions and Assumptions

In our setting, we make the following assumptions, and our notation is given in Table 1.

- The name of the streaming chunk in ICN is abstracted as two components: the streaming file name i and the chunk sequence number j, which can be used to distinguish different chunks of the file (e.g., */3/9*).
- The chunk sequence number j is ordered based on the streaming sequence (e.g., the sequence number of the first chunk is 1, the second is 2, etc.).
- Accordingly, a streaming file f_i consists of a set of chunks $\{C_{i,1}, C_{i,2}, ..., C_{i,j}...\}$, which are ordered by the sequence number j, and each chunk $C_{i,j}$ is assumed to have the same chunk size.
- We assume that the streaming chunks are requested in sequence starting with the first chunk.
- The request duration of f_i (the period between the start and end of one continuous user request for file f_i) follows the distribution $Q(x)$. For example, in the case of video, viewers may quit the session before the end of the video.

Table 1. Notation. REQ: Request ratio.

$P_f(i)$	File-level request probability of file f_i among different files at an Information-Centric Networking (ICN) router.
$REQ_f(i)$	Local request ratio of file f_i measured at each ICN router.
δt	The time unit of the $REQ_f(i)$ measurement, e.g., $\delta t = 1s$ denotes that $REQ_f(i)$ is recalculated each second.
$P_{f_i}(j)$	Chunk-level internal request probability for chunk $C_{i,j}$, which refers to the request probability for chunk $C_{i,j}$ among all other chunks within streaming file f_i.
$IRXI_{f_i}(j)$	Internal request expectation index (IRXI) of chunk $C_{i,j}$.
$RXI(i,j)$	Request expectation index (RXI) of chunk $C_{i,j}$.
$Q(x)$	Request duration distribution.
$q(x)$	Probability density function of request duration distribution $Q(x)$.
ΔT_i	Average request interval of consecutive chunks in file f_i.
M_i	Total chunk number of file f_i, e.g., for chunk $C_{i,j}$, $j \in \{1, ..., M_i\}$.
n	Total number of chunks cached in a router.

3. Request Expectation Index Based Cache Replacement Algorithm

3.1. Request Expectation Index $RXI(i,j)$

$RXI(i,j)$ is the request expectation index of chunk $C_{i,j}$, which estimates the expectation of chunk $C_{i,j}$ being requested in the future. $RXI(i,j)$ is calculated according to two levels of the request probability and the recent request status, as shown in Equation (1):

$$RXI(i,j) = P_f(i) \times IRXI_{f_i}(j). \tag{1}$$

$P_f(i)$ is the real-time file-level request probability of file f_i among different cached files in the ICN router, and $\sum_{i=1}^{N} P_f(i) = 1$. $IRXI_{f_i}(j)$ indicates the internal request expectation index of chunk $C_{i,j}$, which estimates the expectation of chunk $C_{i,j}$ being requested in the future among the chunks within the file f_i. The difference between $RXI(i,j)$ and $IRXI_{f_i}(j)$ is that $RXI(i,j)$ denotes the request expectation of chunk $C_{i,j}$ among all cached chunks and $IRXI_{f_i}(j)$ only reflects the request expectation among the chunks within file f_i. For instance, assume there are two streaming files f_1 and f_2 and each file has two chunks, i.e., $f_1 = \{C_{1,1}, C_{1,2}\}$, $f_2 = \{C_{2,1}, C_{2,2}\}$. In addition, the file-level request probability of each file is given by $P_f(1) = 0.3$ and $P_f(2) = 0.7$, while, within file f_1, the request expectation of each chunk is given by $IRXI_{f_1}(1) = 0.8$ and $IRXI_{f_1}(2) = 0.2$. Then, in this ICN router, the request expectation index of chunk $C_{1,1}$ is $RXI(1,1) = P_f(1) \times IRXI_{f_1}(1) = 0.3 \times 0.8 = 0.24$.

3.1.1. File-Level Request Probability $P_f(i)$

File-oriented request probability has been well established in extensive studies. For example, the request probability for web objects follows the Zipf distribution and Mandelbrot–Zipf distribution for objects delivered by P2P networks [38]. However, the file-level probability $P_f(i)$ at an ICN router varies by time and location. Therefore, at each ICN router, the cache replacement should be adjusted based on the dynamic variation of the local file-level probability.

To achieve this goal, each router monitors the local real-time file-level access frequency to obtain request ratio $REQ_f(i)$, as shown in Figure 3, and the calculation of $REQ_f(i)$ is given bellow, as shown in Equation (2):

$$REQ_f(i) = \frac{Rec_i}{Rec}. \tag{2}$$

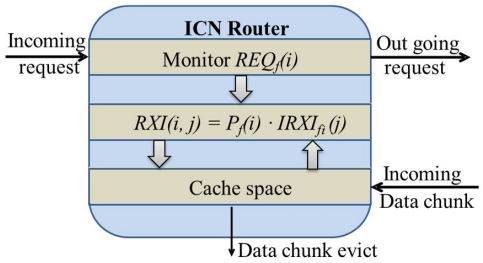

Figure 3. Monitor the request ratio $REQ_f(i)$. REQ: Request ratio; RXI: Request expectation index; IRXI: Internal request expectation index.

Rec_i refers to the number of requests for file f_i in time period $(t, t+\delta t)$, and Rec indicates the total number of request received by this node in the same duration. Additionally, the calculation time unit δt affects the ability of $REQ_f(i)$ to accurately reflect the variation of the local file-level request probability. $REQ_f(i)$ may be changed by the arrival of new requesters; therefore, we stipulate that the time unit for these calculations is equal to the average time interval for the arrival of a new requester. In summary, the file-level request probability $P_f(i)$ can be reasonably approximated by the measured request ratio $REQ_f(i)$, as shown in Equation (3):

$$P_f(i) \approx REQ_f(i). \tag{3}$$

3.1.2. Chunk-Level Internal Request Probability $P_{f_i}(j)$

For the streaming request, the user does not always request all the chunks of the file. The request probability of different chunks within a streaming file may not be equal. For instance, it is unlikely that all the viewers watch the whole video from the first chunk until the end. Thus, within a video, there is a significant request probability gap between the first chunk and the last chunk. Therefore, to accurately express this request probability difference among chunks within the same file, the chunk-level internal request probability $P_{f_i}(j)$ is introduced.

$P_{f_i}(j)$ denotes the request probability of chunk $C_{i,j}$ among the other chunks of file f_i, and $\sum_{j=1}^{M_i} P_{f_i}(j) = 1$. For a particular type of streaming file, the request duration follows a certain distribution [41], e.g., a Pareto distribution (with mean between 54% and 61% of the total video length) fits well for short videos which last less than 5 min, and the Wellbull distribution (with mean between 18% and 40% of the total video length) fits well for long videos. Therefore, we assume that the request duration of f_i follows distribution $Q(x)$.

$P_{f_i}(j)$ denotes the inherent access frequency among chunks within a particular type of streaming file. According to the statistical characteristics of the request duration distribution $Q(x)$, $P_{f_i}(j)$ is given by Equation (4).

In Equation (4), ΔT_i represents the average request interval of a user requesting two consecutive chunks for file f_i. For instance, assume ΔT_i is 1 s; accordingly, the chunk $C_{i,2}$ will be requested at time 1 s, $C_{i,3}$ will be requested at time 2 s and $C_{i,j}$ at time $(j-1)$ s, etc. Additionally, the request duration $((j-1)\Delta T_i, j\Delta T_i)$ refers that the user start to request the file from time 0 and quit between time $(j-1)\Delta T_i$ and $j\Delta T_i$. In addition, $\int_{(j-1)\Delta T_i}^{j\Delta T_i} q(x)dx$ denotes the probability that the user request duration falls within $((j-1)\Delta T_i, j\Delta T_i)$. According to the assumptions, $\int_{(j-1)\Delta T_i}^{j\Delta T_i} q(x)dx$ indicates the probability that the chunk $C_{i,j}$ is requested just before the user drops out of the session. Furthermore, a user request duration longer than time $j\Delta T_i$ indicates that chunk $C_{i,j}$ is definitely requested. Therefore, according to the $Q(x)$ distribution, the numerator of Equation (4) represents the number of possible requests for chunk $C_{i,j}$. Meanwhile, for the denominator, the request duration $(0, \Delta T_i)$ refers that the user quit requesting before time ΔT_i. Thus, if the request duration is in $(0, \Delta T_i)$, there is only one request for chunk $C_{i,1}$, and if it is in $(\Delta T_i, 2\Delta T_i)$, there are two requests for chunk $C_{i,1}$ and $C_{i,2}$ and so

on. Thus, the denominator indicates the possible total number of requests for file f_i based on the $Q(x)$ distribution. Since the denominator does not vary by sequence number j, we stipulate that it equals the constant $\frac{1}{G_i}$. Please note that $P_{f_i}(j)$ can be precalculated once the type of file f_i is known and $Q(x)$ for the file is determined.

$$P_{f_i}(j) = \frac{Number\ of\ request\ for\ chunk\ C_{i,j}}{Number\ of\ total\ request\ for\ f_i}$$

$$= \frac{1 \cdot \int_{(j-1)\Delta T_i}^{j\Delta T_i} q(x)dx + 1 \cdot \int_{j\Delta T_i}^{(j+1)\Delta T_i} q(x)dx + ... + 1 \cdot \int_{(M_i-1)\Delta T_i}^{M_i\Delta T_i} q(x)dx}{1 \cdot \int_0^{\Delta T_i} q(x)dx + 2 \cdot \int_{\Delta T_i}^{2\Delta T_i} q(x)dx + ... + M_i \cdot \int_{(M_i-1)\Delta T_i}^{M_i\Delta T_i} q(x)dx}$$

$$= G_i \cdot \int_{(j-1)\Delta T_i}^{M_i\Delta T_i} q(x)dx, \tag{4}$$

where

$$G_i = \frac{1}{\sum_{k=1}^{k=M_i} k \cdot \int_{(k-1)\Delta T_i}^{k\Delta T_i} q(x)dx}.$$

3.1.3. Chunk-Level Internal Request Expectation Index $IRXI_{f_i}(j)$

Since we assume that users will request the streaming chunks according to the sequence number, the subsequent chunks will have a relatively high probability of being requested in the following time interval. For instance, if an ICN router receives a request for chunk $C_{2,4}$, the subsequent chunks, such as $C_{2,5}$ and $C_{2,6}$, are likely to be requested later. Therefore, if any of these subsequent chunks have already been cached, they will have higher access probability.

$IRXI_{f_i}(j)$ is estimated according to the chunk-level internal probability $P_{f_i}(j)$ and the current request status of file f_i at this router. We record $IRXI_{f_i}(j)$ for every cached chunk in the ICN router and *Update/Recover* it promptly. When a chunk is initially cached, the $IRXI_{f_i}(j)$ of the chunk is equal to the $P_{f_i}(j)$, as shown in Equation (5):

$$IRXI_{f_i}(j) = P_{f_i}(j). \tag{5}$$

$IRXI_{f_i}(j)$ is updated when the ICN router receives additional requests for the same streaming file f_i. According to the linear request feature of streaming content delivery, a received request implies that some cached subsequent chunks will likely be requested in the future; thus, the $IRXI_{f_i}(j)$ of the cached chunks is updated under the following constraints:

1. $C_{i,j}$ has been cached in the cache space of the ICN router.
2. $IRXI_{f_i}(j)$ updating only occurs when this ICN router receives a request for chunk $C_{i,g}$ of the same file f_i.
3. The sequence numbers j of the cached chunks are larger than the sequence number of the received request, i.e., $j > g$.

The $IRXI_{f_i}(j)$ value is updated on the basis of the $P_{f_i}(j)$ gap between the just-received request $C_{i,g}$ and the first chunk of file f_i, as shown in Equation (6):

$$IRXI_{f_i}(j) = P_{f_i}(j) + \Delta\mu, \quad j > g, \tag{6}$$
$$where \quad \Delta\mu = P_{f_i}(1) - P_{f_i}(g).$$

$\Delta\mu$ is calculated according to the following considerations. First of all, if there are no new users requesting file f_i, the chunk $C_{i,g}$ having just been requested refers to the next subsequent chunk

$C_{i,g+1}$ temporarily having the highest internal request probability. Because those chunks whose sequence number are less than or equal to g will never be requested again by the same user. Therefore, we increase $IRXI_{f_i}(g+1)$ by $P_{f_i}(1) - P_{f_i}(g)$ so that $IRXI_{f_i}(g+1)$ is the highest one currently. Secondly, the $IRXI_{f_i}(j)$ of the cached subsequent chunks will be added the same value $\Delta\mu$ so that the same request probability tendency remains compared with the chunk-level internal request probability $P_{f_i}(j)$. Finally, if there is a new user request for file f_i, which refers to the first chunk $C_{i,1}$ being requested, according to Equation (6), the $IRXI_{f_i}(j)$ of chunks in file f_i will return back to the initial value and equal to $P_{f_i}(j)$.

When a request for $C_{i,g}$ is received, the subsequent chunks of file f_i will likely be requested later. Therefore, the $IRXI_{f_i}(j)$ of the subsequent chunks is temporarily increased by $\Delta\mu$, that is, if any of these chunks are cached in the router, they will have high probability of being requested soon and should not be easily replaced by the cache replacement algorithm, as shown in Figure 4.

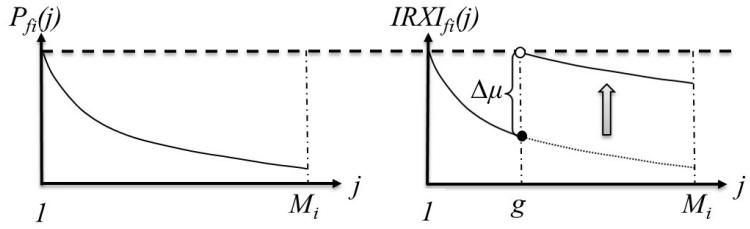

Figure 4. $IRXI_{f_i}(j)$ calculation.

The temporarily increased $IRXI_{f_i}(j)$ of the subsequent chunks should be recovered to $P_{f_i}(j)$ after being requested, and, based on Equations (5) and (6), the calculation of $IRXI_{f_i}(j)$ is summarized in Equation (7).

$$IRXI_{f_i}(j) = \begin{cases} P_{f_i}(j), & C_{i,j} \text{ is initially cached;} \\ P_{f_i}(j) + \Delta\mu, & C_{i,g} \text{ is requested, } (j > g); \\ P_{f_i}(j), & C_{i,j} \text{ is requested.} \end{cases} \tag{7}$$

The Update/Recover procedure is presented in Algorithm 1. When an ICN router receives a request for chunk $C_{I,g}$, the cache space is traversed and the value of $IRXI_{f_i}(j)$ within file f_i is checked for an updating or recovery. According to Algorithm 1, the time complexity of $IRXI_{f_i}(j)$ update/recover process is $O(n)$, where n denotes the total number of chunks cached in the router.

Algorithm 1 $IRXI_{f_i}(j)$ Update/Recover

Require: $P_{f_i}(j)$, $\Delta\mu$, $C_{I,g}$
1: **while** All cached chunk $C_{i,j}$ in the cache space **do**
2: **if** $i = I$ **then**
3: **if** $j > g$ **then**
4: $IRXI_{f_i}(j) \leftarrow P_{f_i}(j) + \Delta\mu$
5: **else if** $j = g$ **then**
6: $IRXI_{f_i}(j) \leftarrow P_{f_i}(j)$
7: **end if**
8: **end if**
9: **end while**

In summary, according to Equations (1)–(4) and (7), the $RXI(i,j)$ of chunk $C_{i,j}$ is given, which is the gist of the RXI algorithm.

3.2. RXI-Based Cache Replacement Algorithm

In an ICN router, the chunk with the minimum $RXI(i,j)$ denotes that the chunk $C_{i,j}$ has the lowest expectation of being requested in the future. Therefore, when cache replacement occurs, the $RXI(i,j)$ of each cached chunk is calculated according to the current value of $P_f(i)$ and $IRXI_{f_i}(j)$. Subsequently, the cached chunk with the minimum $RXI(i,j)$ is evicted, as shown in Algorithm 2. Since the $RXI(i,j)$ of each cached chunk is calculated when cache replacement occurs, a traversal of cache space is needed for the calculations; thus, the time complexity of RXI-based cache replacement algorithm is $O(n)$.

Algorithm 2 RXI-based Cache Replacement

Require: $P_f(i)$, $IRXI_{f_i}(j)$, $MinRXI$, $MinChunk$
1: $MinRXI \leftarrow 1$
2: **while** All cached chunk $C_{i,j}$ in the cache space **do**
3: $RXI(i,j) \leftarrow P_f(i) \times IRXI_{f_i}(j)$
4: **if** $MinRXI > RXI(i,j)$ **then**
5: $MinRXI \leftarrow RXI(i,j)$
6: $MinChunk \leftarrow C_{i,j}$
7: **end if**
8: **end while**
9: Evict the chunk $MinChunk$

4. Simulations and Results

We implement the RXI cache replacement algorithm in the ndnSIM [42] simulator. We compare the proposed scheme with other cache replacement algorithms: LRU, LFU and TLP-TTH. To explicitly address the performance of the RXI algorithm, we adopt the default cache decision policy named Leave Copy Everywhere (LCE) to work cooperatively with the cache replacement algorithm.

The simulation is implemented by using two topologies: a 5-level complete binary tree topology, as shown in Figure 5a, and a hybrid topology consisting of a core network and access networks, as shown in Figure 5b. The China Education and Research Network (CERNET) [43], which is the first nationwide education and research computer network in China, is selected as the core network, and we use a 3-level complete binary tree for the access networks.

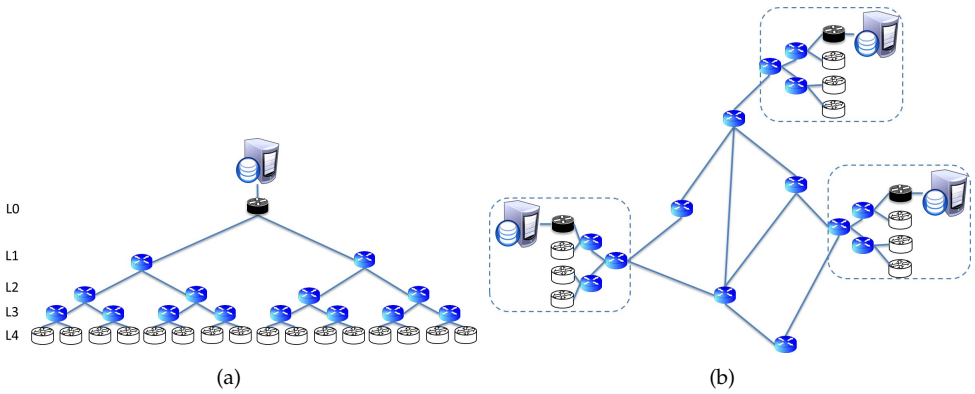

Figure 5. Simulation topologies. (**a**) hierarchical topology; (**b**) hybrid topology.

According to the Cisco Virtual Networking Index (CVNI) project [1], the video streaming delivery has become a major consumer of network traffic and is representative of streaming transmission. We therefore customize the chunk-by-chunk video streaming request model for our simulations in

both scenarios. Four metrics are adopted to evaluate the performance of the replacement policies in each scenario:

1. *Average cache hit ratio* is used as the primary performance metric to measure the cache utilization efficiency. In our simulation, the cache hit ratio is calculated individually by each router and represents the number of cache hits divided by the number of requests received by each router.
2. *Average hop count* indicates the average transmission round-trip distance (the sum of the hops of one pair of request and data packets transferred) for each video chunk delivery.
3. *Network traffic* represents the sum of the data forwarded by every ICN router during the entire simulation.
4. *Total retrieval time* denotes the sum of the retrieval time between a request being sent and the data bing received by the users.

4.1. Scenario 1: Hierarchical Topology with One Producer

In this scenario, we evaluate the performance of the cache replacement algorithms in the 5-level complete binary tree topology. We implement one video streaming server connected to the root router as the black router shows in Figure 5a. Twenty-five video streaming files are supplied on this server, and each contains 1000 chunks that can be requested independently. Accordingly, the total chunk number is 25,000 items. With respect to the file-level popularity, studies have shown that it follows the Zipf distribution. Therefore, we implement 100 video viewers connected to the leaf routers, as represented by the white routers in Figure 5a, and the popularity of requests for different video files follows a Zipf distribution (α = 0.8) [44]. In addition, these viewers request videos with given intervals that are assumed to follow the Exponential distribution with a mean of 1 s, that is, the time unit δt of the $REQ_f(i)$ measurement is set to 1 s. Additionally, the viewer requests for a video following the order of the sequence number j from the beginning of the video, but may stop before the end of the video. According to the observations in study [41], a Pareto distribution (with the mean between 54% and 61% of the total video length) fits the measured viewer request duration. Therefore, we assume that the request duration of viewers follows the Pareto distribution with a mean of 57% of the video duration. The simulation is performed across a range of cache capacity from 200 chunks to 2000 chunks for each ICN node.

The simulation results presented in Figure 6a,b show significant differences among the LRU, LFU, TLP-TTH and RXI-based algorithm. The average cache hit ratio and the average hop count are calculated among each ICN router with the router cache capacity ranging from 200 chunks to 2000 chunks. Compared with TLP-TTH, the average cache hit ratio is improved from 7.44% to 8.05% by the RXI-based algorithm, thus the average cache hit ratio increment is over 8%, while TLP-TTH shows an 11% increment over LFU. Therefore, the overall performance increment is approximately 20% for the RXI-based algorithm compared with LFU and 32% compared with LRU. The RXI-based algorithm shows a 1.1% decrement in average transfer distance compared with TLP-TTH and a nearly 3.3% decrement compared with LFU. The results above show that, compared with the reference algorithms, the RXI-based algorithm enhances the utilization of the cache space; thus, viewers can acquire video chunks from closer routers. In other words, the RXI-based algorithm lightens the pressure on the original video sources. The RXI-based algorithm reduces the total network traffic and the total retrieval time compared with LRU, LFU and TLP-TTH, as shown in Figure 6c,d. Because of the reduction in the average delivery distance between viewers and the target content chunks, the unnecessary network traffic among intermediate routers and the average retrieval time are reduced. These reductions are further intensified by the high cache hit ratio of the RXI-based algorithm. In conclusion, to deliver the same number of video chunks, the RXI-based algorithm consumes less network resources and requires less time than LRU, LFU and TLP-TTH, and when the cache size is limited (e.g., cache size $<$ 1400 chunks), the improvement is more evident.

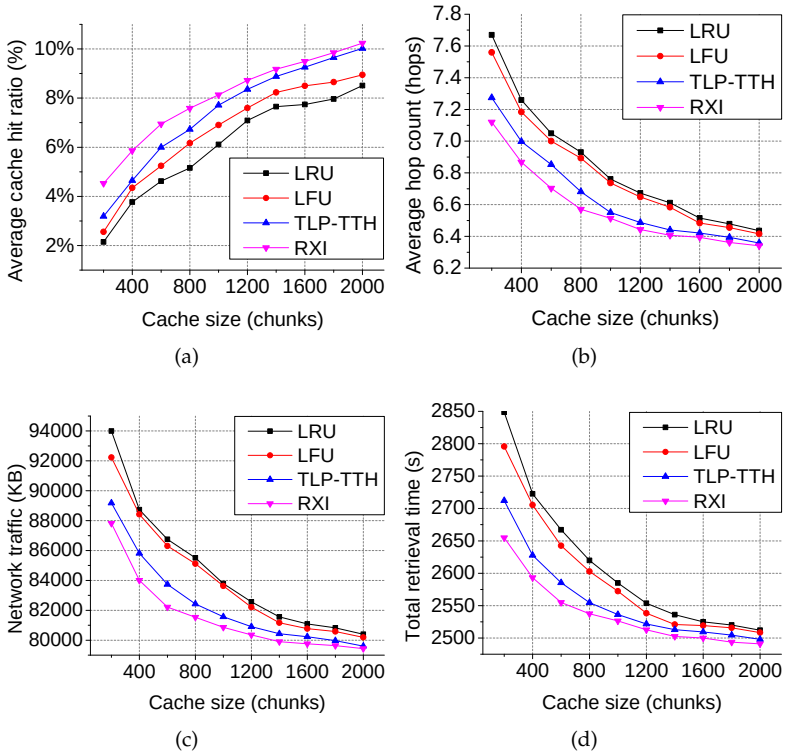

Figure 6. Simulation results of hierarchical topology. (**a**) average cache hit ratio; (**b**) average hop count; (**c**) network traffic; (**d**) total retrieval time. LFU: Least Frequently Used; TLP-TTH: Two-level popularity oriented time-to-hold policy.

The performance improvements are mainly due to the following facts. Fine-grained estimation of future request probability helps with the accurate prediction of future request behaviors, so that the underutilized cache capacity can be efficiently used. For instance, both the TLP-TTH and RXI-based algorithm adopt two-levels of the request popularity for cache replacement, and both algorithms significantly improve the network performance compared with LRU and LFU, which only take the simple chunk-by-chunk request probability into account, as confirmed in all four figures of Figure 5. In addition, unlike TLP-TTH, the RXI-based algorithm includes the chunk-level internal request probability and introduces a unified estimation criteria of possible future request probability for all cached chunks, which may belong to different files and be located in different positions in the file. Therefore, by further considering the internal relationships among chunks within a file, the estimation of future request probability is more accurate, so that the network performance is further improved by the RXI-based algorithm, as shown in Figure 6. Nevertheless, accurate estimation of request probability requires more calculation resources. Due to the simple principle of LRU and LFU, the eviction time complexity can achieve $O(1)$. By contrast, the eviction time complexity of TLP-TTH and RXI-based algorithm is higher, but still able to attain linear time complexity $O(n)$.

In addition, we are curious about how the RXI-based algorithm performs in different domains of a network. Therefore, we recalculate the cache hit ratio of scenario 1 (cache size = 800 chunks) in different network regions. One interesting observation is that, compared with the other algorithms, the RXI-based algorithm increases the cache hit ratio at the edge of the network and at the intermediate routers allocated at the core network, as shown in Figure 7. The improvement in the intermediate network is mostly due to the accurate and independent estimation of the future request probability of the cached chunks at each router, and the eviction of the chunk that has the lowest expectation of been requested later.

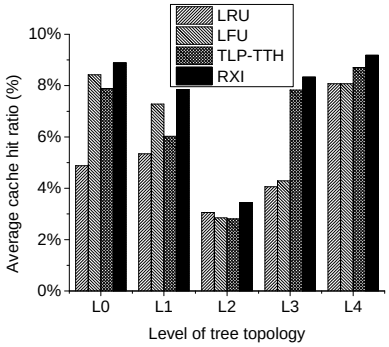

Figure 7. Average cache hit ratio in different levels of hierarchical topology.

4.2. Scenario 2: Hybrid Topology with Multiple Producers

In this scenario, we aim to present the network performance of cache schemes on a complex hybrid network topology. To evaluate the performance in a realistic scenario, video servers are connected to the edges of the access networks. We place three streaming sources that are connected to the black routers, and 100 video viewers are connected to other leaf nodes (white routers) in each access network, as shown in Figure 5b. We randomly place 25 video streaming files in these streaming sources so that the viewers may request these videos from other access networks. With the exception of the topology difference, all of the configurations are the same as in scenario 1.

In the complex network topology and request model, the performance improvement of the RXI-based algorithm is still good, as shown in Figure 8. Since the average cache hit ratio and the average hop count are significantly improved by the RXI-based algorithm compared with the other three policies, unnecessary network traffic and the content retrieval time are reduced.

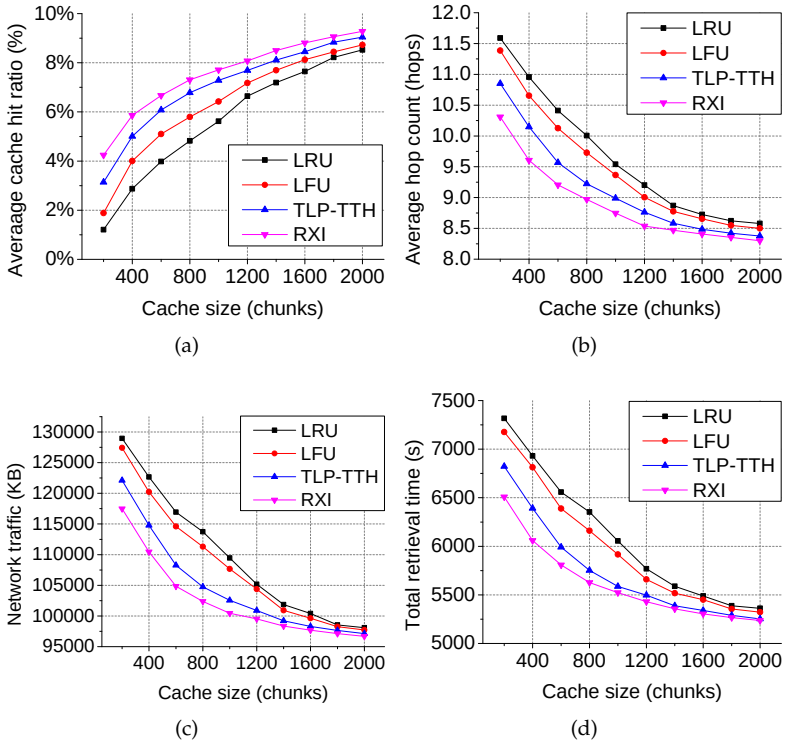

Figure 8. Simulation results of hybrid topology. (**a**) average cache hit ratio; (**b**) average hop count; (**c**) network traffic; (**d**) total retrieval time.

Because the $RXI(i,j)$ is calculated independently at each router based on real-time observations and current streaming request status, the complexity of the network topology and the request model does not affect the performance of the RXI-based algorithm. In addition, the RXI-based algorithm can also avoid additional management overhead, such as messages or control packets transmission, which may obviously increase the network burden in the complex network topology. On the basis of these results, we can conclude that the RXI-based algorithm can be deployed in complex network scenarios.

5. Conclusions

In this paper, the in-network caching management algorithm over ICN is investigated. A RXI-based caching replacement algorithm customized for streaming delivery over ICN is proposed. We have introduced RXI to offer a unified estimation criteria of possible future request probability for cached chunks. RXI adopts both file-level and chunk-level internal request probability and is estimated according to the dynamically varied request status at each ICN router. We have evaluated the proposed algorithm in two scenarios by comparing with the LRU, LFU and TLP-TTH. The simulation results indicate that the RXI-based algorithm evicts the cached chunk with the minimum RXI in order to maintain high cache utilization, and improves the ICN network performance in several aspects. In addition, the RXI-based algorithm can also avoid additional management overhead, such as control packets transmission. Thus, the proposed algorithm can be deployed in complex network scenarios, as proved by the results. In conclusion, by taking fine-grained request probability and dynamically varied request status into consideration, the customized in-network caching management algorithm specifically for streaming delivery can further improve the ICN network performance.

Acknowledgments: Haipeng Li is supported by the China Scholarship Council (No.20120643003) as a Ph.D. student.

Author Contributions: Haipeng Li contributed to the design of the proposed scheme, the performing of the simulation and the writing of the paper. Hidenori Nakazato contributed to reformation of proposed scheme and the supervision of the work. Syed Hassan Ahmed reviewed and revised the paper. All authors have read and approved the final manuscript.

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

References

1. Cisco Visual Networking Index. *Global Mobile Data Traffic Forecast Update, 2013–2018*; White Paper 2014; Cisco Visual Networking Index: London, UK, 2014.
2. Koponen, T.; Chawla, M.; Chun, B.-G.; Ermolinskiy, A.; Kim, K.H.; Shenker, S.; Stoica, I. A data-oriented (and beyond) network architecture. In *SIGCOMM Computer Communication Review*; ACM: New York, NY, USA, 2007; pp. 181–192.
3. Tarkoma, S.; Ain, M.; Visala, K. The publish/subscribe internet routing paradigm (PSIRP): Designing the future internet architecture. In *Future Internet Assembly*; IOS Press: Amsterdam, The Netherlands, 2009; pp. 102–111.
4. Fotiou, N.; Nikander, P.; Trossen, D.; Polyzos, G.C. Developing information networking further: From PSIRP to PURSUIT. In *Broadnets*; Springer: Berlin, Germany, 2010; pp. 1–13.
5. Dannewitz, C. Netinf: An Information-Centric Design for the Future Internet. In Proceedings of the 3rd GI/ITG KuVS Workshop on The Future Internet, Munich, Germany, 28 May 2009; pp. 1–3.
6. Zhang, L.; Estrin, D.; Burke, J.; Jacobson, V.; Thornton, J.D.; Smetters, D.K.; Zhang, B.; Tsudik, G.; Massey, D.; Papadopoulos, C. Named Data Networking (NDN) Project. *Relatório Técnico NDN-0001, Xerox Palo Alto Res. Center-PARC* **2010**, *1892*, 227–234.
7. 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; ACM: New York, NY, USA, 2009; pp. 1–12.
8. Niebert, N.; Baucke, S.; El-Khayat, I.; Johnsson, M.; Ohlman, B.; Abramowicz, H.; Wuenstel, K.; Woesner, H.; Quittek, J.; Correia, L.M. The way 4WARD to the Creation of a Future Internet. In Proceedings of the

IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2008, Cannes, France, 15–18 September 2008; pp. 1–5.

9. Xylomenos, G.; Ververidis, C.N.; Siris, V.A.; Fotiou, N.; Tsilopoulos, C.; Vasilakos, X.; Katsaros, K.V.; Polyzos, G.C. A survey of information-centric networking research. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 1024–1049.
10. Ahlgren, B.; Dannewitz, C.; Imbrenda, C.; Kutscher, D.; Ohlman, B. A survey of information-centric networking. *IEEE Commun. Mag.* **2012**, *50*, 26–36.
11. Choi, N.; Guan, K.; Kilper, D.C.; Atkinson, G. In-Network Caching Effect on Optimal Energy Consumption in Content-Centric Networking. In Proceedings of the 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012; pp. 2889–2894.
12. Liu, W.X.; Yu, S.Z.; Gao, Y.; Wu, W.T. Caching efficiency of information-centric networking. *IET Netw.* **2013**, *2*, 53–62.
13. Meng, Z.; Luo, H.; Zhang, H. A survey of caching mechanisms in information-centric networking. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1473–1499.
14. Laoutaris, N.; Che, H.; Stavrakakis, I. The LCD interconnection of lru caches and its analysis. *Perform. Eval.* **2006**, *63*, 609–634.
15. Laoutaris, N.; Syntila, S.; Stavrakakis, I. Meta Algorithms for Hierarchical Web Caches. In Proceedings of the 2004 IEEE International Conference on Performance, Computing, and Communications, Hyderabad, India, 20–23 December 2004; pp. 445–452.
16. Psaras, I.; Chai, W.K.; Pavlou, G. Probabilistic in-Network Caching for Information-Centric Networks. In Proceedings of the Second Edition of the ICN Workshop on Information-Centric Networking, Helsinki, Finland, 17 August 2012; ACM: New York, NY, USA, 2012; pp. 55–60.
17. Cho, K.; Lee, M.; Park, K.; Kwon, T.T.; Choi, Y.; Pack, S. Wave: Popularity-Based and Collaborative in-Network Caching for Content-Oriented Networks. In Proceedings of the 2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Orlando, FL, USA; 25–30 March 2012; pp. 316–321.
18. Arianfar, S.; Nikander, P.; Ott, J. Packet-Level Caching for Information-Centric Networking. In Proceedings of the ACM SIGCOMM, ReArch Workshop, New Delhi, India, 30 August–3 September 2010.
19. Bernardini, C.; Silverston, T.; Festor, O. MPC: Popularity-Based Caching Strategy for Content Centric Networks. In Proceedings of the IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013; pp. 3619–3623.
20. Wang, Y.; Li, Z.; Tyson, G.; Uhlig, S.; Xie, G. In Optimal Cache Allocation for Content-Centric Networking. In Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP), Gottingen, Germany, 7–10 October 2013; pp. 1–10.
21. Cui, X.; Liu, J.; Huang, T.; Chen, J.; Liu, Y. A Novel Metric for Cache Size Allocation Scheme in Content Centric Networking. In Proceedings of the National Doctoral Academic Forum on Information and Communications Technology, Beijing, China, 21–23 August 2013.
22. Guan, J.; Quan, W.; Xu, C.; Zhang, H. The Location Selection for Ccn Router Based on the Network Centrality. In Proceedings of the 2012 IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), Hangzhou, China, 30 October–1 November 2012; pp. 568–582.
23. Xu, Y.; Wang, Z.; Li, Y.; Lin, T.; An, W.; Ci, S. Minimizing Bandwidth Cost of Ccn: A Coordinated in-Network Caching Approach. In Proceedings of the 24th International Conference on Computer Communication and Networks (ICCCN), Las Vegas, NV, USA, 3 August–6 August 2015; pp. 1–7.
24. Li, Y.; Xie, H.; Wen, Y.; Zhang, Z.-L. Coordinating in-Network Caching in Content-Centric Networks: Model and Analysis. In Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems (ICDCS), Philadelphia, PA, USA, 8–11 July 2013; pp. 62–72.
25. Wang, J.M.; Zhang, J.; Bensaou, B. In Intra-as Cooperative Caching for Content-Centric Networks. In Proceedings of the 3rd ACM SIGCOMM Workshop on Information-Centric Networking, Hong Kong, China, 12 August 2013; ACM: New York, NY, USA, 2013; pp. 61–66.
26. Ming, Z.; Xu, M.; Wang, D. Age-Based Cooperative Caching in Information-Centric Networking. In Proceedings of the 23rd International Conference on Computer Communication and Networks (ICCCN), Shanghai, China, 4–7 August 2014; pp. 1–8.

27. Yu, Y.; Bronzino, F.; Fany, R.; Westphalz, C.; Gerlay, M. Congestion-Aware Edge Caching for Adaptive Video Streaming in Information-Centric Networks. In Proceedings of the 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2015; pp. 588–596.
28. Belady, L.A. A study of replacement algorithms for a virtual-storage computer. *IBM Syst. J.* **1966**, *5*, 78–101.
29. Mattson, R.L.; Gecsei, J.; Slutz, D.R.; Traiger, I.L. Evaluation techniques for storage hierarchies. *IBM Syst. J.* **1970**, *9*, 78–117.
30. Van Roy, B. A short proof of optimality for the min cache replacement algorithm. *Inf. Process. Lett.* **2007**, *102*, 72–73.
31. Arlitt, M.F.; Friedrich, R.J.; Jin, T.Y. Web Cache Performance by Applying Different Replacement Policies to the Web Cache. U.S. Patent 6272598 B1, 7 August 2001.
32. Romano, S.; ElAarag, H. A Quantitative Study of Recency and Frequency Based Web Cache Replacement Strategies. In Proceedings of the 11th Communications and Networking Simulation Symposium, Ottawa, ON, Canada, 14–17 April 2008; ACM: New York, NY, USA; pp. 70–78.
33. Podlipnig, S.; Böszörmenyi, L. A survey of web cache replacement strategies. *ACM Comput. Surv. (CSUR)* **2003**, *35*, 374–398.
34. Carofiglio, G.; Gallo, M.; Muscariello, L.; Perino, D. Modeling Data Transfer in Content-Centric Networking. In Proceedings of the 23rd International Teletraffic Congress, San Francisco, CA, USA, 6–9 September 2011; pp. 111–118.
35. Sun, Y.; Fayaz, S.K.; Guo, Y.; Sekar, V.; Jin, Y.; Kaafar, M.A.; Uhlig, S. Trace-Driven Analysis of ICN Caching Algorithms on Video-on-Demand Workloads. In Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, Sydney, Australia, 2–5 December 2014; ACM: New York, NY, USA, 2014; pp. 363–376.
36. Chao, F.; HUANG, T.; Jiang, L.; CHEN, J. Y.; LIU, Y. J. Fast convergence caching replacement algorithm based on dynamic classification for content-centric networks. *J. China Univ. Posts Telecommun.* **2013**, *20*, 45–50.
37. Ooka, A.; Eum, S.; Ata, S.; Murata, M. Compact CAR: Low-overhead cache replacement policy for an icn router. *arXiv* **2016**, arXiv:1612.02603.
38. Zhang, G.; Li, Y.; Lin, T. Caching in information centric networking: A survey. *Comput. Netw.* **2013**, *57*, 3128–3141.
39. Lee, J.; Lim, K.; Yoo, C. Cache Replacement Strategies for Scalable Video Streaming in Ccn. In Proceedings of the 19th Asia-Pacific Conference on Communications (APCC), Bali, Indonesia, 29–31 August 2013; pp. 184–189.
40. Li, H.; Nakazato, H. Two-level popularity-oriented cache replacement policy for video delivery over ccn. *IEICE Trans. Commun.* **2016**, *99*, 2532–2540.
41. Fricker, C.; Robert, P.; Roberts, J. A Versatile and Accurate Approximation for Lru Cache Performance. In Proceedings of the 24th International Teletraffic Congress, Krakow, Poland, 4–7 September 2012; p. 8.
42. Afanasyev, A.; Moiseenko, I.; Zhang, L. *Ndnsim: NDN Simulator for NS-3*; University of California: Los Angeles, CA, USA, 2012; p. 4.
43. The China Education and Research Network (CERENT). Available online: http://www.edu.cn/cernet_1377/index.shtml (accessed on 29 June 2016).
44. Rossi, D.; Rossini, G. Caching performance of content centric networks under multi-path routing (and more). Technical Report; *Relatório Técnico Telecom ParisTech* **2011**. Available online: http://perso.telecom-paristech.fr/~drossi/paper/rossi11ccn-techrep1.pdf (accessed on 5 May 2015).