Received May 15, 2021, accepted August 25, 2021, date of publication September 9, 2021, date of current version October 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3111220

Frozen Cache: Mitigating Filter Effect and Redundancy for Network of Caches

SAEID MONTAZERI SHAHTOURI[1], MOSTAFA REZAZAD[2], AND RICHARD T. B. MA[3], (Senior Member, IEEE)
[1]Ab Initio Software Company, Weybridge KT13 0NY, U.K.
[2]School of Computing, Institute for Research in Fundamental Sciences (IPM), Tehran 1953833511, Iran
[3]School of Computing, National University of Singapore, Singapore 117417

Corresponding author: Mostafa Rezazad (mostafa@ipm.ir)

ABSTRACT Information-Centric Networking (ICN) architecture leverages the network of caches' idea to bring content closer to consumers to ultimately reduce the load on content servers and prevent unnecessary packet re-transmissions. Nevertheless, the performance of existing cache management schemes mainly developed for a single cache is inadequate for a network of caches. There are many factors such as data dependencies, data redundancy, the limited size of caches, poor replacement policies, and many other factors that negatively impact a network of caches. Besides, traffic correlation among different caches on the network influences the performance of the network of caches. One of the essential correlations is the edge filtering effect. In the presence of data redundancy, the edge filtering effect even becomes more severe. The cache filtering effect happens when all arriving requests inspect the first cache for data. Therefore, the subsequent caches in the network receive only those requests that could not find data (cache-miss) from the edge cache. In this paper, we propose *Frozen-cache* to mitigate the filtering effect. This policy repeatedly freezes content in a cache to allow subsequent caches to receive popular content. A lightweight coordinated scheme incorporated with Frozen-cache policy to cope with the data redundancy problem. Based on our experiments obtained from realistic scenarios, the Frozen-cache idea highly outperforms state of the art caching schemes. Depending on the network setup, this superiority varies from 25% to 700%.

INDEX TERMS Frozen-cache, network of caches, information centric networking, filter effect, locality of reference, coordinated cache scheme, cache policy.

I. INTRODUCTION

Information-Centric Networking (ICN) [1] has gained considerable attention in the current decade as the one idea which has the potential for being the future of Internet architecture. Although there are different proposals for ICN such as NDN [2] and NetInf [3], they all introduce the same concept of *in-network caching*. Through in-network caching, each router utilizes its memory buffer to store data packets that pass through the router. This *network of caches* in a basic form, caches everything at all nodes. That leads to a poor performance in terms of the overall cache hit rate. The two main reasons for this insignificant performance are *filtering effect* and *data redundancy* [4].

A cache can be considered as a filter, i.e., the cache serves the requests that generate cache-hit and forwards the requests that generate cache-miss. This filtering affects the pattern of requests such that subsequent caches are unable to obtain a high hit rate from the forwarded requests. Thus, the cache's performance at the core of the network depends on how the edge routers perform and handle the traffic. If the edge router has enough space for caching the most so-called "popular content," then caching the same content at its core is futile.

To reduce the filtering effect, Busari and Williamson [4] proposed heterogeneous replacement policies. Later, Ari *et al.* [5] proposed Adaptive Caching using Multiple Experts (ACME), which uses a neural network to find the optimal combination of replacement policies. Although previous studies combined different replacement policies to obtain a higher hit ratio at the core routers, their results reveal that their solution does not entirely remove the filtering effect.

In this paper, by proposing *Frozen-Cache policy*, we tried to reduce the cache dependencies between routers at different levels to enhance the network of caches in terms of increasing

The associate editor coordinating the review of this manuscript and approving it for publication was Nabil Benamar.

the durability of popular content in the network. This policy is a two-state caching scheme where a node either behaves like a traditional cache (content in the cache can be replaced) or freeze the data in its cache for a specific period (frozen content cannot be replaced). When the cache is in the frozen state, it does not accept any data chunks. That gives some time to the cached content to receive their potential hit. This is a very desirable condition in a loaded network and one of the key differences with other cache policies. To enhance our policy, we suggested some modifications and presented various versions of the *Frozen-Cache policy*.

The basic form of the policy is presented as FC0 (Frozen-Cache policy version zero), and we show that this version of the proposed policy and the Least Recently Used (LRU) replacement policy exhibit similar performance under Independent Reference Model (IRM) [6] assumption. To further enhance the performance of the caches, FC1 (Frozen-Cache policy version 1) is proposed. Following that, the same algorithm's extensions can be derived (FC2 to FCn) to reduce the filtering effect of edge routers. Through extensive simulations on real traces, it is shown that FC2 can reach cache hit rate very close to the optimal cache policy, and obtain a higher cache hit rate than some of the selected prominent state of the art cache policies.

Besides, to rectify the content redundancy problem, a coordinated cache mechanism is introduced in this paper. In general, in coordinated cache management, some information from other nodes is required to decide which content to be cached or to be evicted. In overall, there are two types of coordinated caching schemes: explicit coordination and implicit coordination schemes [7]. Coordination is explicit when the caches share their state (or state summaries) with each other [8]. The cost of communication to exchange the state of caches is not insignificant. An implicit coordination scheme, on the contrary, does not exchange information with other nodes. These caching mechanisms such as [9]–[11], might obtain some information from their local cache, the position of the cache in the network, or small piggybacking information from request and data packets. However, for an accurate cache placement decision, the implicit coordinated schemes may still suffer due to insufficient information that they can obtain from the network.

The proposed *coordinated cache scheme* is integrated with *Frozen-Cache* policy FCn (called CFCn) to boost cache hit rate. We report two versions of the coordinated Frozen-Cache policy in this paper (i.e., CFC1 and CFC2). The simulation results from **ndnSIM** [12], [13] (a very well-known simulation in this domain) demonstrate the effectiveness of both versions. These techniques dramatically increase the cache hit rate at the core and edge routers, while keeping communication overhead very low, which makes them able to work at line-speed [14]. CFC2 gains a cache hit rate of 7 times higher than the rival coordinated cache schemes. On top of that, a comparison of CFC2 with the other proposed policies reveal that it reduces traffic load by 3 order of magnitude. On the other hand, CFC1 exhibits similar performance in terms of cache hit rate and traffic load to the state of the art coordinated cache policies. However, it reduces the average eviction rate per cache slot up to 4 order of magnitude. That can be considered as a sign of a more energy savior scheme.

Unlike the other similar schemes that impose extra overhead such as content popularity measurement [15] or sharing information among neighboring caches [16] into packets, CFCn (both version 1 and 2) coordinates caching nodes with piggybacking information through three integer fields in the request and the data packets.

The rest of this paper is organized as follows: A summary of the Named Data Networking paradigm is provided in Section II. Section III describes the design of *Frozen-Cache policy* and its variations for a standalone cache. Section IV introduces the coordinated scheme that integrates with Frozen-Cache. The evaluation of our coordinated schemes is presented in Section V. Finally, Sections VI and VII represent related work and concluding remarks.

II. NAMED DATA NETWORKING SUMMARY

Named Data Networking (NDN) [17] is one of the well-defined networking architectures that falls in Information-Centric Networking (ICN) paradigm [18]. Giving a name to content by ICN paradigms enables caching content at the network level. Since then, various studies have been conducted to optimize caching at the packet level in the network. As a similar attempt, we try to improve the performance of caches in the network in terms of cache hit rate in this paper. The paradigm we used to implement Frozen-Cache policy is the NDN paradigm and its infamous simulation environment named ndnSim [12], [13], which is based on NS3 simulator [19]. To be aligned with other related works, we use the same terminology introduced in NDN paper [17].

Content Store: the cache inside a router is called Content Store or CS.

Data Chunk: is the minimum size of the content that can get an address (name). So, every file is divided into several Data Chunks.

Interest Packet: is a request initiated by a client for a specific Data Chunk.

Data Packet: is the packet carrying Data chunk in its payload to respond to an Interest packet.

Content Publisher: is the source of a file, or the owner of the content announces the availability of the file.

Admission Policy: is a scheme that determines which Data Chunks should be admitted to be cached in a router.

The packet forwarding architecture of NDN is explained in [2]. In this architecture, there are three main tables CS, PIT, and FIB standing for Content Store, Pending Interest Table, and Forwarding Information Base, respectively. An arrived Interest packet to a router first investigates CS for a cached copy of the requesting Data Chunk. In case the content is available in CS, the Data Packet of the requesting Data Chunk will be sent back to the requester through the arriving interface of the Interest Packet. On the other hand, if data is not found from CS, an entry will be created in the PIT table indicating the arrival interface of the requesting data's name. This entry will be used by Data Packet to find its way back to the requester. If an entry for a name is in the PIT table, it will be updated by adding the new interface. Later, the arrived Data Packet for the same name will be forwarded to multiple interfaces. The last table, FIB, provides routing information to forward an Interest packet out toward the content publisher. The default admission policy that NDN architecture is used in its basic form [2] is Leave Copy Everywhere (LCE). This admission policy admits all Data Chunks in every Data Packets that are passing by a router. NDN uses the Least Recently Used (LRU) replacement policy to replace a data chunk that recently has not received any hit in the cache with a newly arrived data chunk to the CS.

III. STAND-ALONE FROZEN-CACHE

The Frozen-Cache policy is a solution for the cache filtering effect in a network of caches. Therefore, in this section first, the cache filtering effect is thoroughly described before delving into the details of the Frozen-Cache policy and its variations.

A. CACHE FILTERING EFFECT

Classical cache replacement policies, such as LRU (Least Recently Used) and LFU (Least Frequently Used) were often designed to maximize the hit ratio of a stand-alone cache. Nevertheless, by capturing the most popular chunks from a workload, they negatively impact the performance of their subsequent caches and cause *filter effect* [4] in multiple levels of caches. Some researchers [20] questioned the gravity of this problem. The argument is that if the edge routers are sufficient to respond to the popular content, the necessity of the network of caches fades away. There are two main objectives for attempts like this study to increase the performance of caches at the core of the network: (I) the limited capacity of a single cache (at the edge) cannot cope with the increasingly high traffic flows, (II) there would be multiple copies of the same popular content at the edge routers that could be served with only one copy of the content at the core. We believe that a redundant copy of data should only be placed in the network for load balancing purposes.

To illustrate the filtering effect, the results of an experiment on ndnSIM simulator [12], [13] with a simple topology are presented in Figure 1. The catalog size (the maximum amount of data in the network) in this simulation setup is set to 1000 Data Chunks, content popularity follows Zipf distribution with parameter $\alpha = 1$ and both routers have the same cache size C varying within the range (0, 1000]. The client issues traffic with an average rate of 10^4 requests/sec that follows Poisson distribution. The simulation time is equal to 2×10^7 requests in each simulation run.

To display the importance of cache policy selection on the system's performance, the replacement policy of the cache at the router 2 is fixed to LRU policy and the replacement policy for the cache at router 1 is switched between LFU and

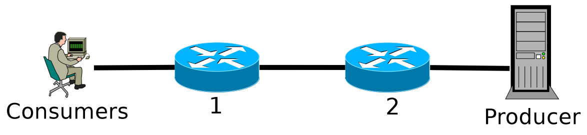
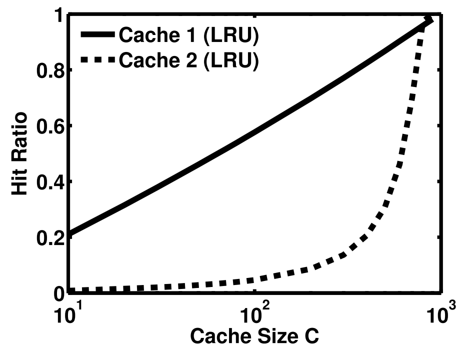


FIGURE 1. A simple topology to illustrate the filter effect.



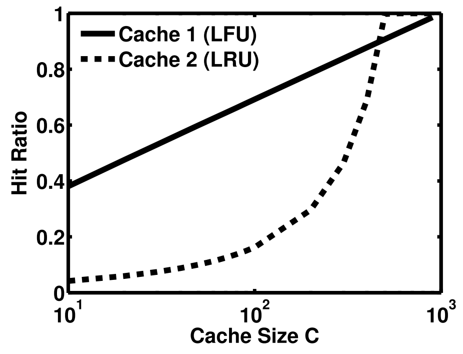


FIGURE 2. Cache hit ratio for two consecutive caches when the replacement policy for both caches are LRU (up) and when the first cache uses LFU and the second cache uses LRU replacement policies (down).

LRU. Figure 2 shows the hit ratio of two caches when LRU and LFU are used for the cache of router 1. One observation is that, when the cache size is much smaller than the catalog size (This is the area in the graph that is more realistic due to the limitations of the current memory technologies [14], [21]), the hit ratio of the cache at router 2 is in order of magnitude lower than the hit ratio of the cache at router 1. Compared to LRU, LFU obtains a higher cache hit ratio at router 1. Theoretically, LFU is an optimal strategy under the Independent Reference Model (IRM) when the content popularity remains constant [6].

Unlike LRU, which replaces cache content more aggressively, LFU keeps a relatively stable working set of content in a cache, which turns out from Figure 2 that it is more beneficial for the performance of subsequent caches. Motivated by this observation, we propose *Frozen-Cache*, a framework to freeze caches, to improve the overall performance of a network of caches when LRU replacement policy is deployed. LFU is not a practical replacement policy due to its high cost in implementation. It requires many counters (one counter for each data chunk) to store the frequency of data's request.

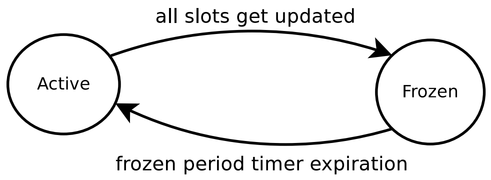


FIGURE 3. State diagram of a Frozen-Cache policy.

B. FROZEN-CACHE POLICY

The idea of Frozen-Cache can be illustrated by the state transition diagram in Figure 3. Based on this figure, a cache can operate at either the *Active state* or the *Frozen state*. The cache executes its regular cache policy when in the Active state. The policy decides whether to cache the passing by traffic or just pass it to the next node without caching it.

However, by the transition to the frozen state, the node locks the content inside the cache (i.e., its content is *frozen*) for a specific period (*frozen period*), during which no new chunks get cached, and no existing chunks get evicted.

Under this framework, three design parameters need to be specified:

- The length of the frozen period (T)
- The cache policy of the active state
- The condition that triggers the transition to the frozen state.

The value of T mainly depends on traffic patterns, which in turn depends on some network parameters such as user demands, routing state, network topology, and some more. Any change in these parameters might affect content popularity. Therefore, this parameter is better set by the network administrator when the network is in a stable condition. For an unstable network, the idea of caching content at the network level might not help much. A wrongly set T value might have a great impact on the caching system's performance. Assume T is the proper frozen period for the current state of the network but the administrator has set it wrongly to $T\prime$ value. If $T\prime > T$ then content remains in the cache for a longer time than it should be. Meaning, the caching system reacts much slower to the changes of the content popularity. When $T\prime < T$, frozen cache policy fails to meet its objective to keep a popular content long enough in the cache to give the content enough time to reach its potential hit rate. We provide some suggestions on how to determine the value of T. First of all, caching unpopular content is futile. Hence, the caching system is supposed to keep as many popular contents as possible. Assuming that the network, mainly the popular range of the content passing through the network, changes every x hours, the value of T should be set to a value less than x hours. On the other hand, caching **large** popular content benefits clients and ISPs more. Thus, one of the most critical targets for network caching is video traffic, and it is shown in [22] that the popularity of Video on Demand (VoD) content does not change within a day. Moreover, it is stated in [23] that routing tables are quite stable for at least 2/3 of the paths on the Internet in a day. Therefore, the lengths of the frozen period can be in the order of hours but less than a day.

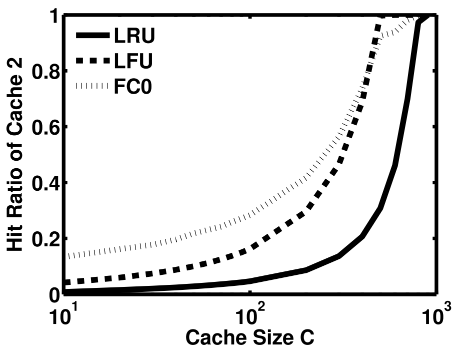


FIGURE 4. Hit ratio of cache 2 under LRU when cache 1 is managed by LRU, LFU and FC0.

The design space of cache policies and triggering conditions for freezing caches are wide open. At the baseline, a cache is assumed to be empty when entering the active state. Transitioning the cache to the frozen state has already cached C distinct newly arrived data chunks. This baseline mechanism is named FC0 (Frozen-Cache version 0). When a node is in the Active state, the LRU replacement policy is used as it is the widely accepted policy in practice.

Considering the simple topology in Figure 1 with T equals to the arrival time of 5×10^4 requests, a comparison plot of hit ratio of cache 2 when LRU manages cache one, LFU, and FC0 cache policies are provided in Figure 4. It is shown that FC0 outperforms the other two policies when the cache size is small compared to the catalog size. One important reason is that FC0 reduces the filter effect for the incoming request to the cache of router two compared to the other two policies. When cache size is small, maybe even the optimum cache policy cannot perform well as the data inside the cache does not stay long enough inside the cache to get hit.

Although LFU, in general, outperforms both LRU and FC0 for stationary traffic, the cost of its implementation is not negligible. Moreover, it fails to adapt itself with a high rate of change in content popularity. Nevertheless, because of its high-performance profile, we will compare its performance with our general design of the Frozen-Cache algorithm in a later section and show they are comparable.

For further explanation on the effectiveness of Frozen-Cache on the hit rate of subsequent caches, we investigate one of the structural characteristics of the inbound traffic, i.e. *reuse distance* [24] on cache 2. Reuse distance is defined as the number of distinctive requested data chunks between two consecutive requests of the same data chunk. If the number of requests defines the notation of time, reuse distance naturally measures the temporal locality of reference of data chunks.

From Figure 5, FC0 has smaller average reuse distance compared to LRU and LFU, especially when C is smaller than the catalog size N. Intuitively, Frozen-Cache keeps a stable working set of content and bypasses some popular content for the subsequent cache. Consequently, it makes the reuse distance of the outbound workload smaller and, therefore, provides a more substantial locality of reference for the subsequent cache's inbound workload.

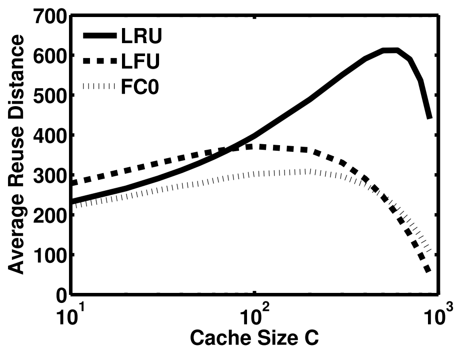


FIGURE 5. Comparison of the reuse distance of cache 2's inbound workload when cache 1 uses LRU, LFU and FC0.

C. FROZEN-CACHE VERSION ONE

In the baseline mechanism, FC0 changes the cache state from active to frozen state when C distinct data chunks arrive. However, there is a concern when *round trip time* (RTT) between the cache and the content provider is not negligible. We need to reserve slots for the first C distinct requested content such that a newly popular content gets into cache before the cache goes to the frozen state because RTT passes before a newly popular content (not present in the cache) gets received by the cache. This constraint might be unfair and prioritize content closer to the consumers over content located at further nodes. To rectify the problem, FC1 makes sure that it has the original C distinct requested content and then transitions to an active state.

The same simulation setup in Figure 1 assesses the algorithm with dynamic content popularity. The popularity index is changed every X random amount of time. X is an exponential random variable with a mean 50 seconds. The change of popularity rank (1 to the catalog size 1000) is determined by a geometric random variable with mean 20 ($p_s = 0.05$). The analysis is based on the stand-alone cache, and the experiment ran 10 times, each lasts for 2×10^5 seconds.

Figure 6 shows the cache hit ratio under LRU, FC0 and FC1 when the RTT varies along the x-axis. The cache size equals to 50 and 100 in the up and down sub-figures, respectively. We observe that all three policies achieve the same cache hit ratio under zero RTT. When RTT increases, the hit ratio under LRU sharply drops till RTT is smaller than the cache characteristic time [25], which is defined as the maximum interarrival time between two consecutive requests for a chunk without a cache miss. Although the hit ratio under FC0 increases with small RTTs, as RTT advances, it sharply drops. The analysis of the two sub-figures reveals that with smaller cache sizes, the performance of the cache is more susceptible to larger RTTs. Nonetheless, with FC1 the cache hit ratio is not sensitive to RTT variation.

D. FROZEN-CACHE THAT FILTERS ONE-TIME REQUESTS

FC1 caches the first C distinct requested chunks regardless of their RTTs, and therefore, it adapts to the popularity changes. Nevertheless, prior works studying web [26] and

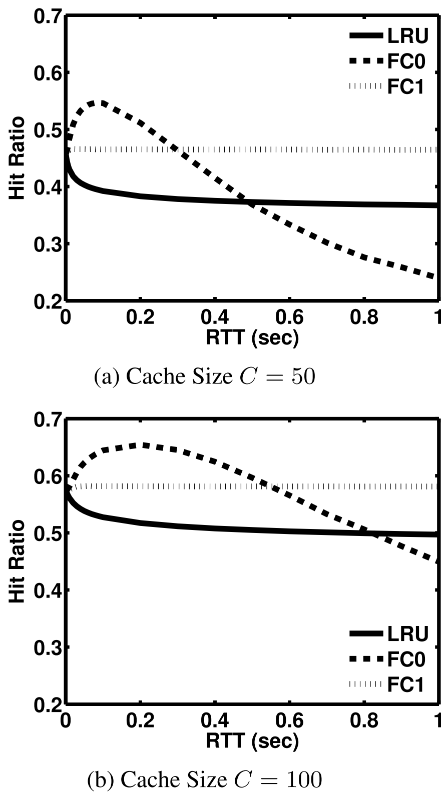


FIGURE 6. Cache hit ratio for LRU, FC0 and FC1 under various RTTs.

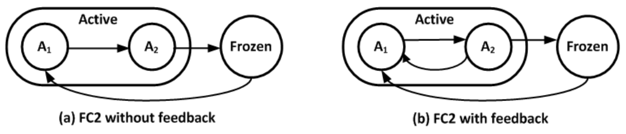


FIGURE 7. State transition diagram of the FC2 policy.

video workload [27] show that up to 50% of the data appear only once during a caching period. Hence, caching them is futile.

To filter out these one-time requests, we add a rule in FC1 policy to cache C district *hit* chunks rather than *requested* chunks to create version two of our policy (FC2). FC2 guarantees that any chunk cached in the frozen state had received a couple of requests during the previous active state. Thus, the frozen chunks are not one-timers.

An intermediate stage is added to the system to achieve the goal. In Figure 7, a cache makes the transition to stage A_2 when all cached data chunks receive at least one request. By the second request, the cache transitions to the frozen state (i.e., with one request stage of the active state changes and with another request (hit), the state transitions from active to frozen). To support the idea, the replacement policy is modified only to evict data chunks that have received zero requests.

Figure 7 illustrates two variations of the state transition diagram of the FC2 policy. The one on the right has a feedback transition from stage A_2 back to stage A_1. This feedback is to reset the system's active state, which is triggered by a time-out

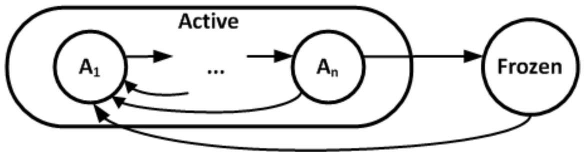
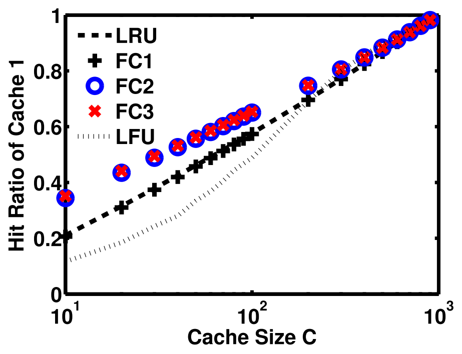
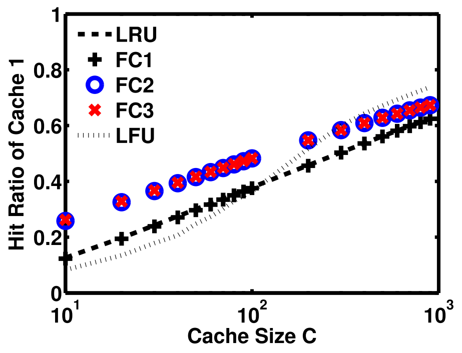


FIGURE 8. State transition diagram of a general FCn policy.



(a) 0% one-timer requests.

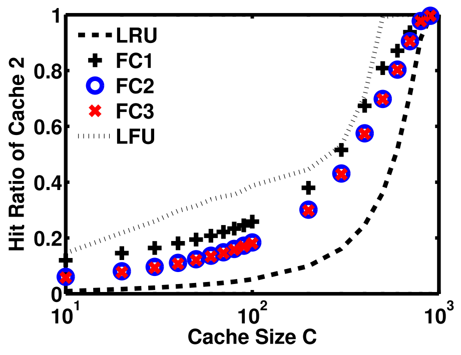


(b) 25% one-timer requests.

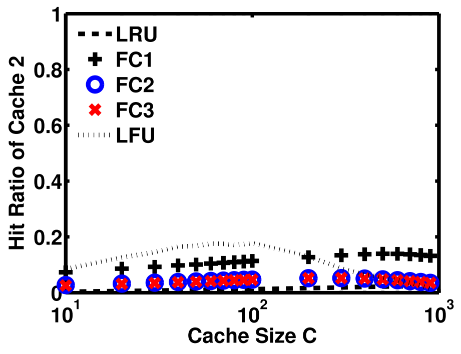
FIGURE 9. Comparison of the hit ratio of cache 1 when cache 1 is managed by LRU, LFU, FC1, FC2 and FC3.

event similar to the triggering condition from the frozen state to the active state. The version with feedback can be seen as a generalization of the version without feedback. When the time-out value is set to infinity, the two versions become equivalent. The system without feedback gets stuck in stage A_2 for too long before capturing C hit chunks and entering the frozen state. To make the cache policy adaptable to the variation of content popularity, the same period of T is set for the time-out from stage A_2 to A_1.

In general, FC2 can be extended to a policy that maintains n stages in the active state, as shown in Figure 8. In particular, FC1 can be regarded as a special case where the active state only maintains a single stage; and therefore, we call such a policy that maintains n stages in the active state as policy FCn. The same experimental settings described in Section III-C is used to assess FCn. Figure 9 plots the hit rate of the first cache managed by LRU, LFU, FC1, FC2 and FC3 under various cache size C. It shows that FC2 and FC3 obtain a higher hit rate than LRU, FC1, and LFU. These results indicate that FC2 can capture popular chunks and adapt to the changes



(a) 0% one-timer requests.



(b) 25% one-timer requests.

FIGURE 10. Comparison of the hit ratio of cache 2 when cache 1 is managed by LRU, LFU, FC1, FC2 and FC3.

in the traffic pattern. Moreover, the figure shows that increasing the number of stages of n from 1 to 2 effectively improves the cache hit rate. However, this improvement is negligible by further increasing n to higher values. Another observation is that increasing the number of one-timer requests reduces the overall performance of the network of caches. Besides, as mentioned earlier, Frozen Cache policy reduces the incoming rate of the data packet to the cache, and that specifically helps when the size of the caches is small.

Figure 10 plots the cache hit rate under LRU at the second cache while the first cache is managed by LRU, LFU, FC1, FC2, and FC3. It shows that the second cache obtains the highest hit rate when the cache policy of the first cache is LFU. Since LFU does not perform well when content popularity is changed, some popular content is captured by LRU at the second cache.

E. IMPLEMENTATION HIGHLIGHTS

To capture the first C distinct data chunks passing by a router, one extra bit, called Reference Bit (RB), is required per cache slot. RBs are all unset at the beginning. When a data chunk at a cache slot gets hit, its corresponding RB will be set. Like Clock (second chance) replacement policy, newly arrived data chunks can be replaced only with the slots whose reference bits are unset. As a result, to move forward from one stage of active state to another stage, the triggering condition is to have all the RBs set. RB in FC0 is set, when writing new content,

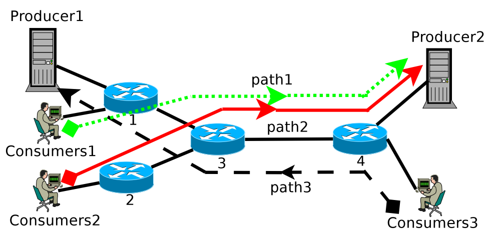


FIGURE 11. Definition of closeness based on various paths.

or when the content in slot hits. In FC1, the reservation is made only by one counter set to zero when a cache transits to the active state. Whenever a request gets missed, and it is forwarded to the content provider or when the content in a slot with an unset RB gets hit, the counter gets increased until it reaches C. After that, the cache only sets a slot RB by writing content. This implementation is sufficient for FC1, but for more intermediate stages, every cache slot needs $[log(n)] + 1$ bits to implement FCn. The value of extra bits of each slot indicates the number of cache hits for the slot.

IV. COORDINATED FROZEN CACHE

As stated in Section III, frozen cache policy enhances the cache hit rate of subsequent caches. However, a local cache management in a network of caches increases the redundant copies of data chunks at different nodes. Data redundancy wastes the overall cache space in the network so that the hit rate of the network of caches drops. To overcome the problem, a lightweight coordinated scheme is introduced in the following subsections. It is shown that the coordinated scheme, together with FCn, highly improves the performance of the network.

To introduce the policy, first, some notions need to be defined:

Path: is a set of routers and links to connect a consumer group to a content producer (or content publisher).

Closeness: is $[\frac{1}{\textit{hop distance to the consumer}}]$. It is calculated by the arrival of a request at a router. The higher the number, the closer the router to the consumer.

Redundant Data: is the presence of multiple instances of a data chunk alongside a **Path**.

For instance, there are three distinct paths in Figure 11 illustrated by different patterns and colors. Path1 consists of routers {1,3,4} that connects consumers1 to the producer2. Path2 involve with routers {2, 3 and 4} to connect consumers2 to the same producer2. The final path, path3, is the opposite of path1 and connects consumers3 to the producer1 through routers{4,3,1}.

Closeness for every path that a router belongs to should be calculated. For example, the closeness of router 4 regarding path3 is 1 while this value is $\frac{1}{3}$ concerning path1 and path2.

Redundant data is usually considered a waste of caching space unless the same copies of data are cached at different *Paths*. For instance, upon requests from consumers 1 and 2 for data X at producer2, the data is cached at all four routers in Figure 11. In this example, the copies of the data at routers 1 and 2 are not *Redundant Data* as they are at different paths and can be used by a different set of consumers. However, the other two copies of X at routers 3 and 4 are redundant. Because all of the requests generated by consumers 1 and 2 are served by router1 and 2, respectively. We try to reduce the wasteful data redundancy.

A. DESIGN GOAL AND PRINCIPLE

The primary goal of the coordinated caching policy is to remove *Redundant Data* in a path. In this regard, the coordinated policy must ensure that only one router in the path between the requester and the source of the content caches the data chunk. Equally important, the policy should decide on the location where the data chunk should be cached. Intuitively, to meet the users' expectation, content should be cached at the closest router to the consumers. However, as the cache space is extremely limited at each router compared to the catalog size of data passing by a router, the design principle is to increase the probability of caching *popular* content at the routers with higher *closeness*. Recall that under the FCn framework, each data chunk in a cache has to pass through n stages before getting frozen, and this is equivalent to get n requests/hits. Thus, from any router's perspective, the instantaneous popularity of a requested chunk can be measured by its current state, i.e., the number of hits accumulated in the active state. If a router is at stage i in the active state, to proceed further into stage $i + 1$ or eventually the frozen state, this router wants to cache content chunks that are popular enough for it, i.e., chunks that have been requested for at least i times, possibly from other routers in a coordinated environment.

Based on the above design goal and principle, Algorithm 1 describes an implementation of the coordinated scheme. The algorithm is used by a router to process arriving requests of data chunks. The router, first, looks up its cache for the name of the data chunk (a.k.a. X). The return value from the cache is either a hit or a miss. In case of a miss, a message is added to the request packet indicating that the router will cache the data chunk if the data chunk at its origin is in stage A_j (i.e., has received j hits). The origin can be a cache of another router or the content publisher. The value of j reduces as the *closeness* of the router in the path decreases. For instance, at the edge router with FCn policy, only contents with $j = n$ are cached; while, the last router in the path (next to the content publisher) caches content with $j = 1$. After the message is prepared, the request will be forwarded according to the routing information provided in the FIB table.

If the outcome of searching the cache is a hit, the data chunk will be returned to the requester using the information of that packet in the PIT table. However, to eliminate data redundancy, the router checks if another router in the return path is willing to cache the content. It is done by matching the hit value of the content in the cache against requested A_j values for the routers in the message. If such equality exists, the data chunk evicts from the cache of the current router.

Algorithm 1 Coordinated Caching Scheme

```
Result: Send Data x if Hit, Forward msg if Miss
msg ← Request for Data x is arrived;
if x is not in current Cache then
    msg ← msg + This router will cache x if it has got $A_j$ hits;
    Path ← FIB(msg.address);
    Send(msg, Path);
else
    Path ← PIT(msg.address);
    routersToCache ← msg.ToCache;
    msg ←Data(x);
    if routersToCache != empty then
        hit ← Data(x).numberOfHits;
        if $Max(routerToCache.hit \leq hit) \neq 0$ then
            Evict(x);
            msg.ToCache ← $A_{hit}$;
        end if
    end if
    Send(msg, path);
end if
```

B. FCn WITH COORDINATION

With coordination, candidates to cache a content add extra information (current active stage of their cache) in the requested packet's header. Based on this information, the router or the content publisher that serves the request specifies which router should cache the content. Augmenting coordination into FCn, requires an extra vector of n Boolean fields $\vec{A} =< A_1, A_2, \ldots, A_n >$ in the header of the requesting packets. Initially all elements of the vector are reset to 0 ($A_1 = A_2 = \ldots = A_n = 0$). A request packet along the path toward the content publisher might visit a router that does not have the cache's content but is willing to cache it when a data packet arrives. Such a router sets the value of A_i that matches the active stage of its cache to one. The router that sets a field of the vector is called *marker*. The action of changing the field is called *marking* a request. When a request packet reaches the content publisher or arrives at a router with the content in its cache, the vector A will be investigated for non-zero fields. Among them, the field with the most significant j value, which is smaller than the current active stage of the router's cache, will be selected as the proper caching node for the content. The reason is explained through an example below:

Suppose when a router receives a request packet, its cache is in stage A_m. Let say j is the smallest index of vector A that has been marked, i.e., $j = min\{i : A_i > 0\}$. The request packet might either get its data from cache or not (hit or miss):

- **Cache Miss (Marking Rules)** Depends on the value of j, the router might react differently:
  - $j = 0$: the vector is in its initial stage. Thus, none of the routers in the path before the current router are interested in caching this content. Therefore, this is the closest router to the consumer that might cache the content.
  - $j > m$: there is at least one router with higher *closeness* value in the path that is interested in caching the content **if and only if** the content has received A_j hits at its origin. So, if the content is not popular enough to satisfy the upstream route requirement, this router can cache it. The current router marks the request packet and leaves the decision of where the content should be cached to the destination node.
  - $j \leq m$: there is another router with higher *closeness* value that is interested in caching the content. Therefore, the current router gives up caching the content due to its less closeness rank and forwards the packet unchanged.

  Marking rule has an inherent characteristic that the marker, which marks a higher A_j field, has higher *closeness* value.
- **Cache Hit (Caching Rules)** FCn operates normally if the request hits the content in the router. Thus, the state of the content becomes A_{m+1}. To prepare the data packet, the node that will cache the content should be determined at this stage.
  - $j = 0$: the vector is in its initial stage. Thus, none of the routers in the path before the current router are interested in caching this content. The content would not be evicted from the current cache.
  - $j > m + 1$: the content in the cache has not received sufficient hits (i.e., it is not popular enough) for any routers in the path. Consequently, the router returns the data packet and keeps the content in its cache. None of the routers will cache this content along the reverse path.
  - $j \leq m + 1$: there exists a router that accepts the content to cache when it is at stage A_{m+1}. In this case, to avoid data redundancy, this router evicts the cache's content even if it is in the frozen state.

When the request reaches the content publisher, and assuming the content at the publisher is always at the initial active stage A_1, the stage of the content is set to A_2.

V. PERFORMANCE EVALUATION

We implemented the frozen cache with coordination in *NDNsim* [12], [13] simulator to evaluate and compare it with other well-known caching schemes. This section first explains the simulation setups, including traffic generation, metrics, and topology. The section follows the simulation results and their analysis.

A. EXPERIMENTAL SETTING

1) BENCHMARK SCHEMES

The performance of Coordinated Frozen Cache policy is evaluated using five real and well-known cache policies which are: Leave Copy Down (LCD), Move Copy Down (MCD), Leave Copy Everywhere (LCE), CCndn and UNI. They all are applicable in the ICN context and representatives of a wide range of existing schemes. In LCD [28], a data chunk on its way back to the consumer is written only into the first cache after the node that it gets hit. LCD is a representative for [11] because it writes the missed content in the network to the farthest router from consumers and moves the content toward the consumers. Similar to LCD, MCD writes the content only into the very first cache in the return path, but it evicts the content from the origin cache. Therefore, MCD reduces data redundancy compared to LCD. LCE is universal caching, and due to its simplicity it is used as the baseline of comparison by many schemes [11], [29], [30]. The main idea behind CCndn (CCndnS) is to spread every data object in the routing path between the consumer and the producer. CCndn breaks data into several segments to place each segment in one router. Since the algorithm at the content publisher determines which segment(s) should be cached in a router, searching other routers' cache for the segment can be escaped with CCndnS. CCndn tries to increase data diversity at each node and engage the core routers more in the caching process by letting them cache a piece of some popular data object. Thus, like most of the cache policies, every piece of data definitely will be cached somewhere in the network with CCndn. However, all nodes might be at freezing state with the frozen-cache policy when some data chunks pass through them, and these data never placed in the network. Besides, the frozen-cache policy is not a deterministic caching scheme, and data cannot be assumed to be cached in any specific node (unlike CCndnS). Moreover, with frozen-cache policy each router individually determines which data should be cached at the router based on its current state. Whilst, this is the content publisher for CCndn that determines which router is responsible to cache which data chunk based on its hop distance to the requesters. Finally, UNI used in [11], which caches each content with a probability inversely proportional to the number of hops between the consumer and the current location of the content. The UNI is a representative of schemes, such as [29], that writes a missed content only in one of the routers on the return path to the consumers.

2) PERFORMANCE METRICS

There are 6 performance metrics used in this study to cover different perspectives from consumers to ISPs.

[i] **System perspective metrics,** represent the performance of the network in its entirety:

1) Overall hit ratio, H, is defined as $H = \frac{N_{hit}}{\sum req}$, where $\sum req$ is the total number of requests entered the network and N_{hit} (i.e., Network hit) is the total number of cache hits from all routers,
2) H_E, average edge routers hit rate,
3) H_C, average core routers hit rate.

[ii] **Consumer perspective metric,** is related to content access time:

4) Average content download time, D, that represents the average latency that consumers experience in downloading contents.

[iii] **ISP perspective metrics,** are related to traffic intensity and cost:

5) Traffic reduction ratio, γ, is defined as $\gamma = \frac{T_{cache}}{T_{cache-less}}$, the fraction of traffic with and without caching capacity in the network.
6) Average eviction rate per cache slot, β, is defined as $\beta = \frac{E}{S \times T_{sim}}$ where E is the total number of evictions in the network, S is the total number of cache slots in the network and T_{sim} is the duration of simulation time. β can be a good estimation of the energy consumption of caching components in the network.

B. ISP TOPOLOGY CAPTURED BY RocketFuel - REQUEST GENERATION

AboveNet, an extensive transit ISP with 6461 Autonomous Systems, is the topology used to assess the performance of the CFCn algorithm. This ISP has 25 edge routers and 77 core routers. All 25 edge routers and 25 of the randomly selected core routers are connected to the content publishers (sources of data). There are three different traffic types generated in this network: i) Video Streaming, ii) Web, and iii) file sharing. The size of each traffic pattern is set based on predicted future traffic [31]. The average size of video streaming is set to 2000 data chunks; the web traffic has the average content size of 100 data chunks; file-sharing has an average content size of 600 data chunks. All of the content sizes are generated based on the geometric distribution [32]. That makes the catalog size of the network around 2×10^7 data chunks in total. The consumers generate content requests based on Poisson distribution with an average rate of 250 requests per second. Prior work [33] has shown that the *session level of Internet traffic* is well modeled by Poisson distribution. Moreover, a window-based request generation at the packet level on the consumer side is used. In this window-based system, request generation starts with $w = 1$ and uses TCP rules, such as linearly increasing the window size by receiving a data packet and dividing window size by half for each packet loss. The popularity of the content follows the Zipf distribution with a slope of one ($\alpha = 1$). The index of file popularity changes with rate $\lambda_c = 0.00066$ (period of 1500 seconds) and it is determined by a geometric random variable with mean 20 ($p_s = 0.05$). Doing that, the location of the popular files dynamically changes through time. The simulation time is set to 6000 seconds, and we run each simulation setup ten times. The frozen period T for both CFC1 and CFC2 is 700 seconds.

1) PERFORMANCE UNDER VARIOUS CACHE SIZES

Hit Ratio: Figure 12 shows that in general, CFC2 has the highest overall hit ratio of H. Because CFC2 obtains the highest hit ratio H_E at the edge routers and has a comparable hit rate of H_C at core routers. CFC2 outperforms LCD

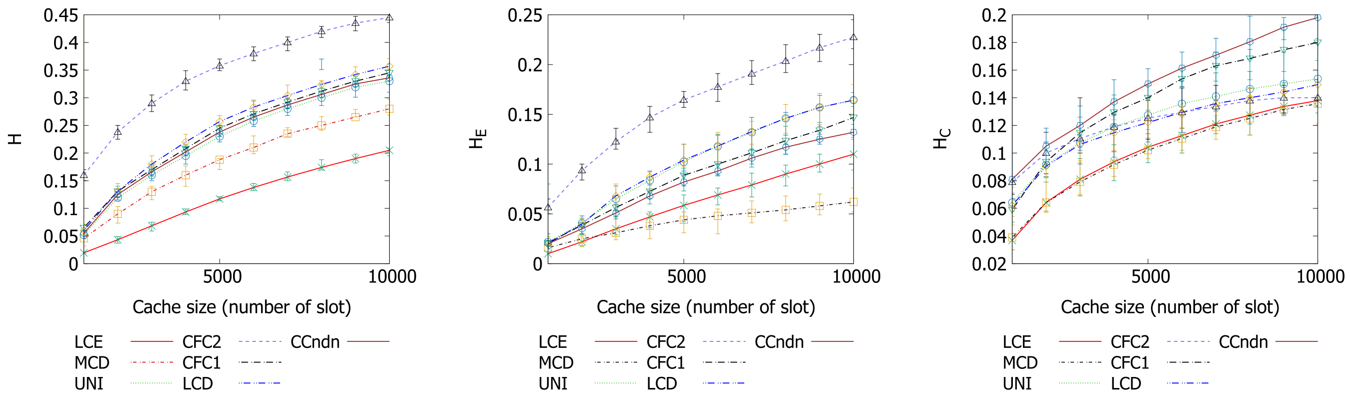


FIGURE 12. Overall hit ratio (*H*), average edge hit ratio (H_E), average core hit ratio (H_C).

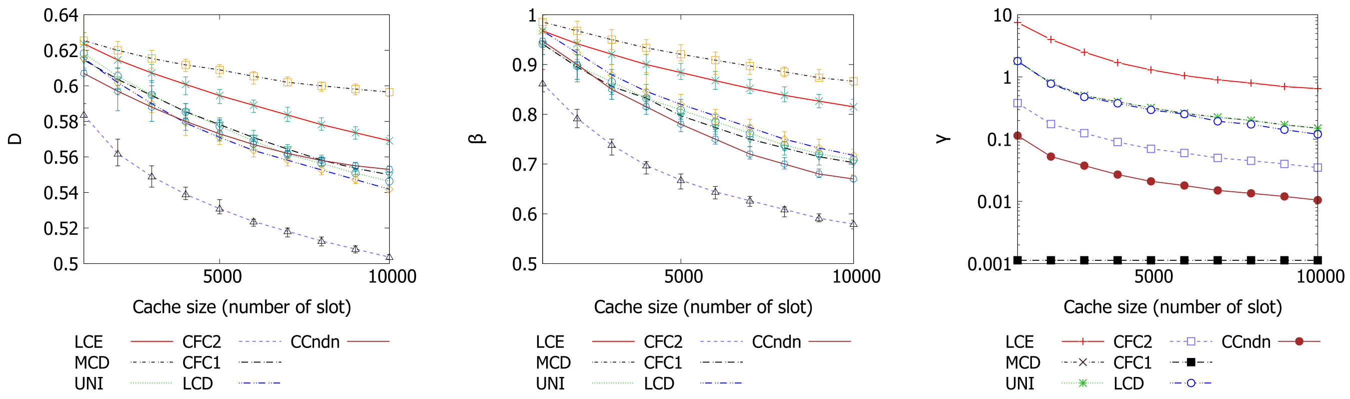


FIGURE 13. Average content download time (*D*), packet reduction ratio (β), eviction rate per slot (γ).

because it brings data closer to the consumers (i.e., prioritizes routers with higher *closeness*). In Figure 11, the edge router 4 caches more content from path three than the other paths under CFC2.

One important result from Figure 12 is that, CFC2 obtains the highest cache hit rate under small cache sizes, regardless of whether it is an edge router or a core router. This area of the curve is more interesting, as catalog size passing by a router increases rapidly while the cache size remains constant. The cache hit rate at core routers drops for CFC2 is that the fixed frozen period of 700 seconds is insufficient to stabilize the cache state when the cache size increases.

Comparing the proposed algorithm with CCndn - the newest cache policy compared to the other policies - shows the superiority of CFC2 over CCndn and all other policies when the overall hit rate is considered. However, CCndn is performing slightly better than the rest of the policies at the core of the network. Unlike the other policies that significantly improve the performance of the edge routers, for CCndn there is no difference between edge and core routers, and the slight improvement at the core routers is due to the more significant number of traffic passing through the core routers.

Average Content Download Time and Traffic Reduction Ratio: Figure 13 shows that CFC2 significantly reduces content download time and provides the traffic volume reduction around 30%, more than other schemes. MCD shows the least improvements in these regards as it removes the popular content from the router that could be an edge router for other consumers. CCndn improves the performance of the core routers, and that is the reason it is more successful than other policies (except CFC2) to reduce traffic load.

Average Eviction Rate Per Slot: Figure 13 also shows that as a trade-off, by using CFC1, the average eviction rate can be decreased up to 4 orders of magnitude, which could help to save the energy consumption in ICN routers. In CFC2, each cache keeps replacing the contents until capturing *C* contents that get at least one hit. On the other hand, each cache in CFC1 can stop replacing after capturing the first *C* distinct contents (either they get hit or not). Therefore, CFC2 replace more content but obtains more popular content compared to CFC1. Since CCndn considers only a fraction of content to cache at each router, the router's catalog size is smaller, and the incoming data rate to the router is less than most of the algorithms. Therefore, the data eviction rate for that is lower.

2) PERFORMANCE VERSUS POPULARITY OF CONTENTS

To test the performance of the frozen cache under various content popularity, we also use a constant cache size of 1000 chunks and change the α parameter of the Zipf distribution from 0.7 to 1.2. Figure 14 depicts a similar trend

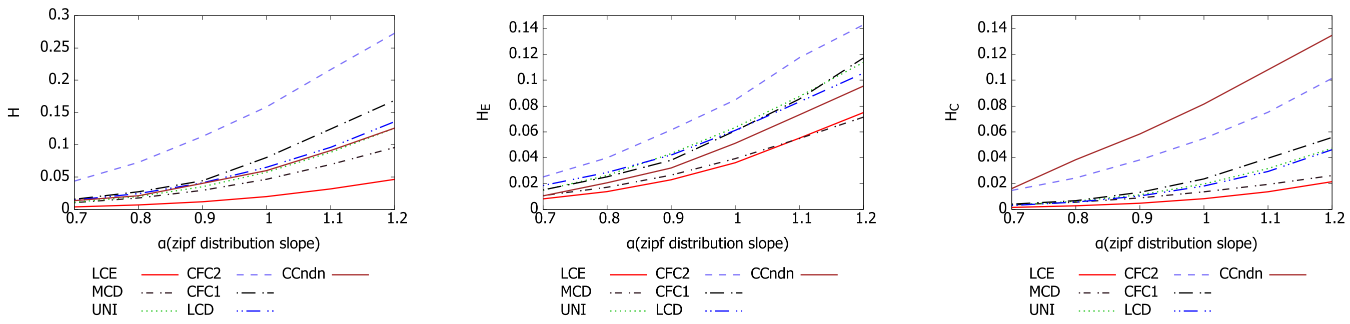


FIGURE 14. Overall hit ratio (H), average edge hit ratio (H_E), average core hit ratio (H_C) when α varies.

under these scenarios. As CCndn promised, it improves the core routers performance whilst, CFC2 wins overall network hit rate.

VI. RELATED WORKS

Caching at Application-level, to reduce traffic load on the links, and to lessen content access time for clients, has been studying and developing for many years [34]. However, Content-Centric Networking [2], [35] was one of the pioneers to propose a practical scheme to utilize cache at the network level. This networking paradigm has been growing vastly in terms of applications such as, smart homes [36], ad-hoc networks [37], [38], vehicle applications [39]–[41], wireless sensor networks [42], [43], IoT (Internet of Things) [44]–[49], connectivity of mobile networks [50], [51], etc. However, the main advantage of CCN or, later, NDN (Named Data Networking) [17] lies in its compatibility with the current infrastructure. A tunneling method to run NDN on top of TCP/UDP protocols is proposed in [52].

Since then, one crucial trend of studies is measuring and improving the performance of a network of caches working at network level.[1] One reason NDN's default en-route caching strategy leaves the core caches cold lies in the lack of cache diversity, i.e., the core contains copies of the content at the edge. One way to reduce this redundancy is for the caches to run some coordination protocol [34], [53]. The coordination may require the measurement of content popularity [54]–[56]. Such schemes increase traffic overheads, add complexity to the routers, and may not adapt fast enough to changes in popularity.

We classify ICN coordinated caching schemes in two categories. The first category either imposes high overhead such as measuring the frequency or makes impractical assumptions, such as having a holistic view of the network. Although these works give us a better understanding, they are not practical due to the technology's current limits, such as lack of fast, inexpensive, and plentiful memories. For instance, algorithms proposed in [16], [57] change the default route of requests by looking at the state of the neighboring caches. Therefore, each cache must maintain extra information about its neighbors' content to update them about how its cache state changes periodically. The updating communication overhead depends on the cache update rate, which is high due to the small cache size of an ICN router compared to the total content available on the Internet. Breadcrumb is a well-known idea that is used to find the best location of the cached node [58]. Still, this approach requires extra memory to track the history to find a node with the content.

As another attempt to reduce the complexity of off-path caching, OpenCache [59] reduces the domain of the cache collaboration into a smaller group of nodes to share the most popular content among them. In an innovative study, a routing algorithm is used to retrieve a copied data in one of the routers in [60]. Their proposed scheme searches for a copy of the requested data using probe packets. The probe packet searches the routers to find the data. However, this idea works well when there are enough resources in the network to cache most of the data in many close nodes to the requesters.

Other coordinated caching schemes with communication overhead, even with on-path caching, are [15], [61], [62]. In an on-path caching, only nodes on the routing path from content publishers toward consumers are considered for caching content.

We are looking for a scheme to improve the performance without adding extra traffic overhead and expect minimum additional complexity at routers. Ideas like those presented in [9], [10], [62] require each router to measure the access frequency, which imposes processing overhead to the ICN router.

The second category has low overhead and does not rely on impractical assumptions. For example, [11], [29] proposed easy-to-implement coordinated caching schemes. WAVE, the scheme proposed by [11], determines the number of packets that should be cached by measuring the content popularity in the producers. However, measuring popularity at the producer may not be very accurate because of intermediate caches. On the other hand, the authors in [29] propose a probabilistic in-network caching scheme. The scheme considers three parameters to calculate the probability to let the content be accepted in a cache: the total cache size in the path from consumer to producer, the number of hops from

[1] The main difference between caching at Application-level and Network-level is in the formation of the topology. Unlike Network-level caching, Application-level caching can define an arbitrary topology.

the previous location of the content, and the number of hops to the consumer. Their idea for probabilistic caching could be useful, but their evaluation is limited to the hierarchical topologies.

Although it is widely believed that popular content must be cached at the edge of the network and less popular at the core routers [63], the introduction of an optimal cache allocation in [64] defies this belief. Wang *et al.* formulated the problem as a linear program to maximize the benefit of caching, equivalent to hop distance minimization. Their finding emphasis that for a heterogeneous topology, a system tends to cache content at core routers, and for a homogeneous topology, content is pushed more toward the edge routers. In this regard, studies like [63], [65], [66] illustrate the benefit of distributing an object through multiple caches and reducing the download time by a parallel download scheme. These are the most relevant studies to our proposal regarding reducing content correlation among routers. However, there are some subtle differences. First they do not consider content popularity in their caching. Frozen-cache assumes that LRU captures the popular file during the active state of the cache. Second, the frozen-cache idea does not force content in a cache to be replaced when the cache is in the frozen state. Third, frozen-cache improves the performance of the edge router which is very precious to the end-users. Besides, our scheme achieves the same distribution goal with a minimum required explicit coordination.

Using local content popularity with pre-filter queues to filter out popular content is proposed in [67]. Since measuring popularity at a local node is a waste of resources, a global popularity measurement is proposed in [68]. Although they have an excellent idea of caching the most popular content at the core router where they can get more hits from more potential requesters, using Zipf distribution at all routers to assess content popularity seams, not a right approach. As mentioned earlier in this paper, cache hits at downstream routers change the popularity patterns at upstream routers.

In another popularity-based approach to reduce redundancy that is inspired by web caching, an age-based cooperative caching scheme is proposed in [15]. In their design, popular content that is cached in routers closer to requesters receive larger age values. In this way, they push the popular content closer to the clients. Like the other caching proposals, measuring popularity is not a straightforward task. Another example of measuring popularity with breadcrumbs is presented in [58]. The paper uses the content popularity to cache them closer to the requester and cache less popular content further from the clients. Caches are using the Betweenness Centrality idea to determine whether to cache content or not. This paper not only has the limitation of an impractical way of popularity measurement, but the cost of calculating betweenness centrality is not negligible. As examples of other researches that cannot be applied here are: Multifactor replacement [69] that uses semantic information provided in the packets to find proper data for a request. Aiming at the IoT domain, a request can be satisfied by other data with some errors. In an off-path caching, data can be cached anywhere in the network. Ideas like using hash function [70]–[72] to find a proper cache to locate the content, interferes with the default routing algorithm of NDN and decouples it from TCP/IP. A similar trend that requires modifying the routing algorithm is proposed in [73]. The paper proposes a controller to rout packets toward the content. In another approach, Xiaoyan *et al.*, in [74] network coding idea along with multipath routing to locate and find data in an off-path router. However, using multipath routing in this way increases the traffic load of the network. A similar idea of network coding for multipath off-path caching is presented in [75].

An application specific caching scheme is proposed in [76]. The algorithm targets the Video caching application for the purpose of energy efficiency in cognitive content centric networking.

VII. CONCLUSION AND FUTURE WORKS

We proposed *frozen cache policy* a new way to manage a network of caches to achieve a high cache hit rate and reduce the filtering effect. We achieved those goals by uniformly distributing requests to all caches. In our proposal, content in a cache is frozen to prevent them from being replaced. That forces the other content to be cached in another node and gives them a chance to cache some popular content. We showed that our basic idea of *frozen cache policy* has the same hit ratio as LRU, which can be modified for a higher hit ratio, and can be used in a coordinated fashion for a network of caches. Our *coordinated frozen cache scheme* obtains an improvement of 7 times of overall hit ratio for small cache sizes and up to 25% for large cache sizes, and decreases the average number of evictions per cache slot by 4 orders of magnitude. Besides, our scheme reduces the average content download time up to 10%, traffic reduction ratio up to 26% compared to existing coordinated schemes with low overhead. Future directions include the design of new triggering conditions for frozen cache for new objectives such as minimizing the Inter-ISP traffic and combining our scheme with the works that consider the content in the neighbors' caches. We believe *frozen cache policy* suits such schemes as it needs minimal cache updates.

REFERENCES

[1] D. Kutscher, S. Eum, K. Pentikousis, I. Psaras, D. Corujo, D. Saucez, T. Schmidt, and M. Waehlisch, "Information-centric networking (ICN) research challenges," Internet Res. Task Force (IRTF), Tech. Rep. rfc7927, 2016.

[2] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th Int. Conf. Emerg. Netw. Exp. Technol. (CoNEXT)*, 2009, pp. 1–12.

[3] B. Ahlgren, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, R. Rembarz, and V. Vercellone, "Design considerations for a network of information," in *Proc. ACM CoNEXT Conf.*, 2008, pp. 1–6.

[4] C. Williamson, "On filter effects in web caching hierarchies," *ACM Trans. Internet Technol.*, vol. 2, no. 1, pp. 47–77, 2002.

[5] I. Ari, A. Amer, and R. Gramacy, "ACME: Adaptive caching using multiple experts," in *Proc. WDAS*, 2002, pp. 143–158.

[6] J. E. G. Coffman and P. J. Denning, *Operating Systems Theory*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1973.

[7] E. J. Rosensweig and J. Kurose, "Breadcrumbs: Efficient, best-effort content location in cache networks," in *Proc. IEEE 28th Conf. Comput. Commun. (INFOCOM)*, Apr. 2009, pp. 2631–2635.
[8] X. Tang and S. T. Chanson, "Coordinated en-route web caching," *IEEE Trans. Comput.*, vol. 51, no. 6, pp. 595–607, Jun. 2002.
[9] A. Ioannou and S. Weber, "Towards on-path caching alternatives in information-centric networks," in *Proc. 39th Annu. IEEE Conf. Local Comput. Netw.*, Sep. 2014, pp. 362–365.
[10] J. Ren, W. Qi, C. Westphal, J. Wang, K. Lu, S. Liu, and S. Wang, "MAGIC: A distributed MAx-gain in-network caching strategy in information-centric networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2014, pp. 470–475.
[11] K. Cho, M. Lee, K. Park, T. T. Kwon, Y. Choi, and S. Pack, "WAVE: Popularity-based and collaborative in-network caching for content-oriented networks," in *Proc. IEEE INFOCOM Workshops*, Mar. 2012, pp. 316–321.
[12] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of ndnSIM: An open-source simulator for NDN experimentation," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 47, no. 3, pp. 19–33, Jul. 2017.
[13] S. Mastorakis, A. Afanasyev, I. Moiseenko, and L. Zhang, "ndnSIM 2: An updated NDN simulator for Ns-3 [R]," UCLA Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0028, Rev. 2, 2016.
[14] S. Arianfar, P. Nikander, and J. Ott, "On content-centric router design and implications," in *Proc. Re-Architecting Internet Workshop (ReARCH)*, New York, NY, USA, 2010, p. 6.
[15] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networks," in *Proc. IEEE INFOCOM Workshops*, Mar. 2012, pp. 268–273.
[16] S. Guo, H. Xie, and G. Shi, "Collaborative forwarding and caching in content centric networks," in *Proc. 11th Int. TC Conf. Netw. (IFIP)*, Prague, Czech Republic. Berlin, Germany: Springer-Verlag, 2012, pp. 41–55, doi: 10.1007/978-3-642-30045-5_4.
[17] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and P. B. Zhang, "Named data networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, Jul. 2014.
[18] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1024–1049, 2nd Quart., 2014.
[19] G. Carneiro, "NS-3: Network simulator 3," in *Proc. UTM Lab Meeting April*, vol. 20, 2010, pp. 4–5.
[20] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker, "Less pain, most of the gain: Incrementally deployable ICN," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*. New York, NY, USA, 2013, p. 147.
[21] D. Rossi and G. Rossini, "On sizing CCN content stores by exploiting topological information," in *Proc. IEEE INFOCOM Workshops*, Mar. 2012, pp. 280–285.
[22] J. Choi, A. S. Reaz, and B. Mukherjee, "A survey of user behavior in VoD service and bandwidth-saving multicast streaming schemes," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 1, pp. 156–169, 1st Quart., 2012.
[23] H. Madhyastha, E. Katz-Bassett, and T. Anderson, "iPlane nano: Path prediction for peer-to-peer applications," in *Proc. NSDI*, Apr. 2009, pp. 137–152.
[24] R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, "Evaluation techniques for storage hierarchies," *IBM Syst. J.*, vol. 9, no. 2, pp. 78–117, 1970.
[25] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1305–1314, Sep. 2002.
[26] A. Mahanti and C. Williamson, "Traffic analysis of a web proxy caching hierarchy," *IEEE Netw.*, vol. 14, no. 3, pp. 16–23, May/Jun. 2000.
[27] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti, "Characterizing web-based video sharing workloads," *ACM Trans. Web*, vol. 5, no. 2, pp. 1–27, May 2011.
[28] M. Zhang, H. Luo, and H. Zhang, "A survey of caching mechanisms in information-centric networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1473–1499, 3rd Quart., 2015.
[29] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. 2nd Ed. ICN Workshop Inf.-Centric Netw. (ICN)*, 2012, pp. 1–6.
[30] Y. Mordjana, M. R. Senouci, and A. Mellouk, "Performance analysis of caching and forwarding strategies in content centric networking," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.
[31] Cisco Visual Networking Index, "Cisco visual networking index: Forecast and methodology, 2011–2016," Cisco, Tech. Rep., 2012.
[32] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: A view from the edge," in *Proc. 7th ACM SIGCOMM Conf. Internet Meas.*, New York, NY, USA, 2007, p. 15.
[33] E. Chlebus and J. Brazier, "Nonstationary Poisson modeling of web browsing session arrivals," *Inf. Process. Lett.*, vol. 102, no. 5, pp. 187–190, May 2007.
[34] W. Wong, L. Wang, and J. Kangasharju, "Neighborhood search and admission control in cooperative caching networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2012, pp. 2852–2858.
[35] N. Lal, S. Kumar, G. Kadian, and V. K. Chaurasiya, "Caching methodologies in content centric networking (CCN): A survey," *Comput. Sci. Rev.*, vol. 31, pp. 39–50, Feb. 2019.
[36] S. H. Ahmed and D. Kim, "Named data networking-based smart home," *ICT Exp.*, vol. 2, no. 3, pp. 130–134, Sep. 2016.
[37] R. A. Rehman, J. Kim, and B.-S. Kim, "NDN-CRAHNs: Named data networking for cognitive radio ad hoc networks," *Mobile Inf. Syst.*, vol. 2015, pp. 1–12, 2015.
[38] Y. Thomas, N. Fotiou, S. Toumpis, and G. C. Polyzos, "Improving mobile ad hoc networks using hybrid IP-information centric networking," *Comput. Commun.*, vol. 156, pp. 25–34, Apr. 2020.
[39] X. Liu, M. J. Nicolau, A. Costa, J. Macedo, and A. Santos, "A geographic opportunistic forwarding strategy for vehicular named data networking," in *Intelligent Distributed Computing IX*. Springer, 2016, pp. 509–521.
[40] A. Boukerche and R. W. L. Coutinho, "LoICen: A novel location-based and information-centric architecture for content distribution in vehicular networks," *Ad Hoc Netw.*, vol. 93, Oct. 2019, Art. no. 101899.
[41] M. Chen, T. Wang, K. Ota, M. Dong, M. Zhao, and A. Liu, "Intelligent resource allocation management for vehicles network: An A3C learning approach," *Comput. Commun.*, vol. 151, pp. 485–494, Feb. 2020.
[42] M. Amadeo, C. Campo, A. Molinaro, and N. Mitton, "Named data networking: A natural design for data collection in wireless sensor networks," in *Proc. IFIP Wireless Days (WD)*, Nov. 2013, pp. 1–6.
[43] G. Jaber and R. Kacimi, "A collaborative caching strategy for content-centric enabled wireless sensor networks," *Comput. Commun.*, vol. 159, pp. 60–70, Jun. 2020.
[44] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, "Caching in named data networking for the wireless Internet of Things," in *Proc. Int. Conf. Recent Adv. Internet Things (RIoT)*, Apr. 2015, pp. 1–6.
[45] M. Meddeb, A. Dhraief, A. Belghith, T. Monteil, K. Drira, and H. Mathkour, "Least fresh first cache replacement policy for NDN-based IoT networks," *Pervas. Mobile Comput.*, vol. 52, pp. 60–70, Jan. 2019.
[46] M. A. Bouras, A. Ullah, and H. Ning, "Synergy between communication, computing, and caching for smart sensing in Internet of Things," *Proc. Comput. Sci.*, vol. 147, pp. 504–511, Jan. 2019.
[47] A. Djama, B. Djamaa, and M. R. Senouci, "Information-centric networking solutions for the Internet of Things: A systematic mapping review," *Comput. Commun.*, vol. 159, pp. 37–59, Jun. 2020.
[48] I. U. Din, S. Hassan, A. Almogren, F. Ayub, and M. Guizani, "PUC: Packet update caching for energy efficient IoT-based information-centric networking," *Future Gener. Comput. Syst.*, vol. 111, pp. 634–643, Oct. 2020.
[49] H. Asmat, I. U. Din, F. Ullah, M. Talha, M. Khan, and M. Guizani, "ELC: Edge linked caching for content updating in information-centric Internet of Things," *Comput. Commun.*, vol. 156, pp. 174–182, Apr. 2020.
[50] F. R. C. Araujo, A. M. de Sousa, and L. N. Sampaio, "SCaN-mob: An opportunistic caching strategy to support producer mobility in named data wireless networking," *Comput. Netw.*, vol. 156, pp. 62–74, Jun. 2019.
[51] R. Ullah, M. A. U. Rehman, M. A. Naeem, B.-S. Kim, and S. Mastorakis, "ICN with edge for 5G: Exploiting in-network caching in ICN-based edge computing for 5G networks," *Future Gener. Comput. Syst.*, vol. 111, pp. 159–174, Oct. 2020.
[52] J. Shi. (2017). *Tunnel Ethernet Traffic Over NDN*. [Online]. Available: https://yoursunny.com/t/2017/tunnel-Ethernet-over-NDN/
[53] L. Dong, D. Zhang, Y. Zhang, and D. Raychaudhuri, "Optimal caching with content broadcast in cache-and-forward networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.
[54] J. Li, H. Wu, B. Liu, J. Lu, Y. Wang, X. Wang, Y. Zhang, and L. Dong, "Popularity-driven coordinated caching in named data networking," in *Proc. 8th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Oct. 2012, pp. 15–26.
[55] H. Wu, J. Li, Y. Wang, and B. Liu, "EMC: The effective multi-path caching scheme for named data networking," in *Proc. 22nd Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2013, pp. 1–7.

[56] C. Bernardini, T. Silverston, and O. Festor, "MPC: Popularity-based caching strategy for content centric networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2013, pp. 3619–3623.

[57] Y. Wang, K. Lee, B. Venkataraman, R. L. Shamanna, I. Rhee, and S. Yang, "Advertising cached contents in the control plane: Necessity and feasibility," in *Proc. IEEE INFOCOM Workshops*, Mar. 2012, pp. 286–291.

[58] Y. Hayamizu, A. Shibuya, and M. Yamamoto, "Effective new cache decision policy for breadcrumbs in content-centric networking," in *Proc. IEEE Int. Workshop Tech. Committee Commun. Qual. Rel. (CQR)*, May 2017, pp. 1–6.

[59] Y. Yang, T. Song, and B. Zhang, "OpenCache: A lightweight regional cache collaboration approach in hierarchical-named ICN," *Comput. Commun.*, vol. 144, pp. 89–99, Aug. 2019.

[60] I. V. Bastos and I. M. Moraes, "A diversity-based search-and-routing approach for named-data networking," *Comput. Netw.*, vol. 157, pp. 11–23, Jul. 2019.

[61] V. Sourlas, P. Flegkas, and L. Tassiulas, "A novel cache aware routing scheme for information-centric networks," *Comput. Netw.*, vol. 59, pp. 44–61, Feb. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613004039

[62] J. Li, B. Liu, and H. Wu, "Energy-efficient in-network caching for content-centric networking," *IEEE Commun. Lett.*, vol. 17, no. 4, pp. 797–800, Apr. 2013.

[63] M. Rezazad and Y. C. Tay, "Decoupling NDN caches via CCndnS: Design, analysis, and application," *Comput. Commun.*, vol. 151, pp. 338–354, Feb. 2020.

[64] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for content-centric networking," in *Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2013, pp. 1–10.

[65] V. Sourlas, P. Georgatsos, P. Flegkas, and L. Tassiulas, "Partition-based caching in information-centric networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr./May 2015, pp. 396–401.

[66] M. Rezazad and Y. C. Tay, "CCndnS: A strategy for spreading content and decoupling NDN caches," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, May 2015, pp. 1–9.

[67] D. Man, Q. Li, Y. Wang, Y. Wu, and X. Du, "An adaptive cache management approach in ICN with pre-filter queues," *Comput. Commun.*, vol. 153, pp. 250–263, Mar. 2020.

[68] V. Young, C.-C. Chou, A. Jaleel, and M. Qureshi, "Ship++: Enhancing signature-based hit predictor for improved cache performance," in *Proc. Cache Replacement Championship (CRC) Held Conjunct Int. Symp. Comput. Archit. (ISCA)*, 2017.

[69] L. Dong and R. Li, "A novel multi-factored replacement algorithm for in-network content caching," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2019, pp. 246–251.

[70] S. Wang, J. Bi, and J. Wu, "Collaborative caching based on hash-routing for information-centric networking," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, Aug. 2013, p. 535.

[71] K. Hasan and S.-H. Jeong, "Efficient caching for delivery of multimedia information with low latency in ICN," in *Proc. 11th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2019, pp. 745–747.

[72] L. Saino, I. Psaras, and G. Pavlou, "Hash-routing schemes for information centric networking," in *Proc. 3rd ACM SIGCOMM Workshop Inf.-Centric Netw. (ICN)*, 2013, pp. 27–32.

[73] E. Aubry, T. Silverston, and I. Chrismen, "Implementation and evaluation of a controller-based forwarding scheme for NDN," in *Proc. IEEE 31st Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2017, pp. 144–151.

[74] X. Hu, S. Zheng, L. Zhao, G. Cheng, and J. Gong, "Exploration and exploitation of off-path cached content in network coding enabled named data networking," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2019, pp. 1–6.

[75] X. Hu, S. Zheng, G. Zhang, L. Zhao, G. Cheng, J. Gong, and R. Li, "An on-demand off-path cache exploration based multipath forwarding strategy," *Comput. Netw.*, vol. 166, Jan. 2020, Art. no. 107032.

[76] M. Zhang, B. Hao, F. Song, M. Yang, J. Zhu, and Q. Wu, "Smart collaborative video caching for energy efficiency in cognitive content centric networks," *J. Netw. Comput. Appl.*, vol. 158, May 2020, Art. no. 102587.

SAEID MONTAZERI SHAHTOURI received the B.Sc. degree in computer engineering from Isfahan University of Technology, in 2002, the M.S. degree in computer engineering from Iran University of Science and Technology, in 2005, and the Ph.D. degree in computer science from the National University of Singapore, in 2015. He is currently an Internal Consultant with Ab Initio Software. His research interests include network of caches, distributed systems, and big data processing.

MOSTAFA REZAZAD received the B.Sc. degree in computer hardware from Azad University, Tehran, Iran, in 2000, the M.Sc. degree in computer architecture from Sharif University of Technology, Tehran, in 2004, and the Ph.D. degree in computer science from the National University of Singapore, Singapore, in 2015. After his Ph.D. and before joining IPM, he was a Postdoctoral Researcher at Singapore University of Technology and Design (SUTD). He is currently a Senior Researcher with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM). At the same time he teaches graduate and under-graduate level courses, such as network security, advanced networks, and operating systems at Sharif University of Technology. In addition to many years working experience as a Network Administrator and a Computer Consultant, he opened his own start-ups in Iran, in 2000 and 2004, working on wireless telecommunication IP-based systems.

RICHARD T. B. MA (Senior Member, IEEE) received the B.Sc. degree (Hons.) in computer science and the M.Phil. degree in computer science and engineering from The Chinese University of Hong Kong, in 2002 and 2004, respectively, and the Ph.D. degree in electrical engineering from Columbia University, in 2010. During the Ph.D. degree, he was a Research Intern with IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, and Telefonica Research, Barcelona. He is currently an Assistant Professor with the Department of Computer Science, National University of Singapore. His current research interests include distributed systems and network economics. He was a co-recipient of the Best Paper Award in the IEEE Workshop on Smart Data Pricing 2015, the IEEE ICNP 2014, and the IEEE IC2E 2013.