HAL open science

Collaborative Forwarding and Caching in Content Centric Networks

Shuo Guo, Haiyong Xie, Guangyu Shi

To cite this version:

Shuo Guo, Haiyong Xie, Guangyu Shi. Collaborative Forwarding and Caching in Content Centric Networks. 11th International Networking Conference (NETWORKING), May 2012, Prague, Czech Republic. pp.41-55, 10.1007/978-3-642-30045-5_4 . hal-01531138

HAL Id: hal-01531138

https://inria.hal.science/hal-01531138v1

Submitted on 1 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Collaborative Forwarding and Caching in Content Centric Networks

Shuo Guo[1], Haiyong Xie[2,3,⋆], and Guangyu Shi[3]

[1] University of Minnesota, Minneapolis, Minnesota 55455
sguo@umn.edu
[2] University of Science and Technology of China, Hefei, China
haiyong.xie@ustc.edu
[3] Central Research Institute, Huawei Technologies, Shenzhen, China
shiguangyu@huawei.com

Abstract. Content caching plays an important role in content-centric networks. The current design of content-centric networks adopts a limited, en-route hierarchical caching mechanism, and caching and forwarding are largely uncoordinated. In this paper, we propose a novel collaborative caching and forwarding design. In this design, collaboration is guided by content popularity ranking, based on which we introduce a collaborative forwarding table to allow coordination between caching and forwarding. We also propose a self-adaptive dual-segment cache division algorithm to deal with dynamic inconsistent content popularity. We evaluate our design via extensive simulations and demonstrate that our design improves content access cost and cache miss rate by at least 30% in a diverse network settings.

Keywords: content-centric network, name-based routing, collaborative forwarding and caching

1 Introduction

Content caching plays an important role in content-centric networks (CCN) [25] or named-data networks (NDN) [38]. With routers being able to cache contents in such networks, it is likely that not only the content distribution costs incurred to the network but also the quality of service experienced by end users are significantly improved. Internet content caching, especially collaborative caching, has drawn much attention (*e.g.*, [8,14,16,20,21,23,26,29,32,35,37]) and some have become commercially successful since more than a decade ago. This leads us to believe that collaborative caching in CCN is a key to success in that the network performance could be significantly improved by letting routers collaborate with each other to optimize overall caching performance. Furthermore, forwarding, if coordinated with caching, is likely to further optimize the network performance.

The current design of CCN adopts a hierarchical caching mechanism allowing only *limited* collaboration in content caching. More specifically, for a given content, caching in CCN takes place only at en-route routers (*i.e.*, routers on

⋆ Corresponding author.

the paths between a requesting host and one or multiple content origins), and thus forms a hierarchical caching mechanism. An en-route router that has the requested content will directly respond with the content from its local content store and then suppress further forwarding the request (*i.e.*, Interest) to the next router in the routing hierarchy. With its unique name-based routing architecture and Interest forwarding, CCN advocates a "*host-to-content*" communication model differing from the "*host-to-host*" model in Internet. In CCN, where content comes from is no longer important to the requesting host[4]. Additionally, not only en-route routers but also routers in the same administrative domain (particularly those nearby en-route routers) could have possibly cached a requested content. These observations suggest that collaborative caching beyond the current limited hierarchical mechanism is feasible and could be beneficial.

However, collaborative caching in CCN, if not well designed, could significantly increase the communication overhead. For instance, control messages exchanged among routers, as an example of such overhead, are necessary to enable collaborations. Such messages normally contain information about what contents are stored in a particular router; due to the enormously large number of distinct contents, such messages could consume a significant portion of the network bandwidth. Additionally, the extra latency of exchanging such messages may further slow down the collaborative decision making process and thus reduce the effectiveness. A naive approach to collaborative caching is to adopt a broadcast mechanism, *i.e.*, each request is forwarded to all routers and only those with the requested content respond with the data. However, such an approach is too costly and inefficient. A key challenge to collaborative caching in CCN is how to make routers know what contents are available from other collaborative routers in an economic and efficient manner. Furthermore, since routers have knowledge about such availability information, routers should leverage it when making forwarding decisions; namely, forwarding and caching should be coordinated and collaborative.

In this paper, we go beyond the en-route caching mechanism and propose a novel name-based distributed collaborative forwarding and caching design (referred to as *CFC* for short) for content-centric networks. More specifically, collaboration is guided by content popularity, and content popularity is measured distributively by content routers. Each router maintains an Availability Information Base (*AIB* for short), estimating which content could be available from which router. Each router also generates a *popularity ranking sequence* periodically through local measurements and propagated such sequences; after aggregating sequences announced by other routers, each router is able to update its AIB and leverage it to optimize forwarding decisions. However, in practice content popularity is likely different when measured from different routers. In order to deal with such inconsistency seen by different routers, we are inspired by the PodNet Project [24] and propose a self-adaptive dual-segment cache division design, using an additional cache space to handle inconsistent content requests and dynamically adjusting cache division based on different levels of inconsistency.

[4] CCN architecture has measures to ensure content security, which is beyond the scope of this paper.

We summarize our contributions as follows. Firstly, to the best of our knowledge, our popularity-ranking based collaborative forwarding and caching scheme for content-centric networks is the first to coordinate forwarding and caching decisions through the availability information base, allowing us to utilize the information of content popularity ranking to reduce the network cost for cache collaboration. When assuming consistent popularity, we theoretically prove the optimality of our design. Secondly, We propose a novel self-adaptive dual-segment cache design to deal with popularity inconsistency. Thirdly, we evaluate the performance of our design via extensive simulations and demonstrate that our design outperforms the hierarchical caching design significantly.

The rest of the paper is organized as follows: Section 2 summarizes the related work. Section 3 introduces the network model. Sections 4 and 5 present our collaborative caching design, followed by the evaluations in Section 6. Section 7 concludes the paper with future work.

2 Related Work

In recent years there has been a line of work on emerging future Internet architectures (see, *e.g.*, [5,6,15,25,38]). In such architectures, content caching becomes an inherent capability of network elements such as routers. Without specifying the details of content caching, these architectures are designed to allow flexible design and implementation of new caching schemes. However, they also pose new challenges to caching schemes; in particular, it remains unclear how content caching should be provisioned (independently or collaboratively), and how it should be implemented efficiently. To the best of our knowledge, our work is among the first attempts to investigate these issues and provide new insights through comparative evaluations.

There is also a large body of literature on content caching in traditional network architectures (see, *e.g.*, [8,14,16,17,20,21,23,26,29,32,33,35,37]). Content caching has been an integral component of Internet-based services for many years, and this has been reflected by the proliferation of content delivery networks (*e.g.*, [3,4,11,27,28,30]). Collaborative caching (or cooperative caching) has been a long-lasting research topic. Researchers have not only investigated the effectiveness of collaborative caching (see, *e.g.*, [21,35]), but also proposed numerous collaborative caching schemes for both general networks and networks with specific structures (see, *e.g.*, [8,14,19,20,23,26,29,32,33,37]); for instance, general Internet-based content distribution (see, *e.g.*, [20,26,34]), delivering content of special types (*e.g.*, [29]), content caching in networks with special topological structures (*e.g.*, [8]), and content caching in special networks such as ad hoc networks (*e.g.*, [24]), content-centric networks (*e.g.*, [10,31,36], and peer-to-peer networks (*e.g.*, [23]),

Our work clearly differs from the above work in that we take advantage of the unique properties of content-centric networks, propose the Availability Information Base guided by the popularity ranking sequence to achieve coordinated forwarding and caching for content routers, and propose the self-adaptive dual-segment division algorithm to deal with inconsistent popularity.

3 Network Model

We consider an autonomous system managed by an administrative domain. The network consists of N content routers. Each router i has a local Content Store (CS) that can cache up to C_i content objects ("contents" for short). The size of each content is u at the largest. We assume that content can be chunked into pieces[5], and each piece fits one cache unit (*i.e.*, the size of each piece is no greater than u). Then, the entire network can cache at most Cu of data, where $C = \sum_{i=1}^{N} C_i$. Users send requesting packets of "Interest" to their nearest routers (see, *e.g.*, [25]).

We use a ranking sequence $\{r_1, r_2, ..., r_C\}$ to denote the most popular C contents, sorted in descending order of popularity. This ranking sequence can be measured in real time as routers receive Interests. All routers may not see the same distribution of content popularity; however, we assume that the ranking sequence $\{r_1, r_2, ..., r_C\}$ measured by different routers have a certain percentage of mismatches or shifts, refer to as the *popularity inconsistency*.

In intradomain, topological information, *e.g.*, link status information and link costs between any pair of adjacent routers i and j (denoted by d_{ij}), is typically distributed by intradomain routing protocols such as OSPF and ISIS. Link costs can correspond to IGP link weights, or other metrics that the content-oriented network cares (*e.g.*, distance, latency). When other metrics are used, they can be distributed across the whole domain via OSPF TLV messages. Other topological information including sizes of Content Stores (C_i's) and the average rate of arriving Interests (*Received Interest Rate* for short), denoted by I_i for router i, can also be distributed in the same way as link costs.

Note that an Interest contributes to a router's Received Interest Rate only if it is sent to this router by a client, rather than another router. We assume that a router can easily distinguish (by, *e.g.*, adding an additional flag bit in the Interest packets) if an Interest comes from a user, or instead from a collaborative router.

4 Ranking-Based Collaborative Forwarding and Caching

In this section we present our design for the collaborative forwarding and caching scheme.

4.1 Overview

We introduce a new component, the *Availability Info Base* (*AIB* for short), to allow us coordinate forwarding and caching in content routers, as shown in Fig. 1. AIB keeps track of content availability information. More specifically, AIB can be thought of as a table, where each entry has two columns, *Name* and *RouterID*, suggesting that a given named content is available from a router. Note that we assume that each router in the network has a name and routers' names

[5] Content objects are segmented into pieces in many content-centric networks (*e.g.*, [25, 38]) as well as in many content-oriented overlay networks (*e.g.*, [7, 18]).

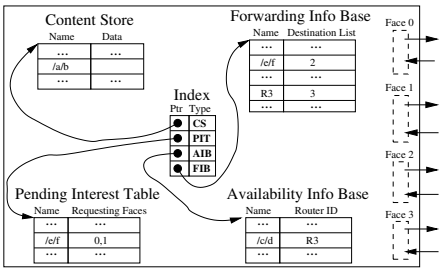


Fig. 1. Content router with collaborative forwarding and caching.

are propagated through the network via intradomain routing protocol such as OSPF. As a result, routers' names are treated in the same way as content names and put in FIB. For instance, the outbound face to reach Router `R3` is face 3, and content `/c/d` is available from `R3`, as shown in Fig. 1.

Each content router periodically announces the pairwise link cost and collaborative forwarding/caching related metrics via OSPF or ISIS intradomain routing protocols. Each router also measures the ranking of incoming Interests, namely, examine the received Interests from the users, and generates its local ranking sequence of the most popular C contents. Each router implements the collaborative forwarding and caching mechanism, namely, a distributed mechanism to make joint decisions for forwarding and caching. Additionally, each router measures the miss rate of the interests in order to further improve the caching/forwarding efficiency.

Upon receiving an Interest, a router first checks whether the content is available and fresh in its local CS. If yes, the router responds with the locally cached content. Otherwise, it looks up the Pending Interest Table (PIT), and either this Interest should not be forwarded if it is already pending in PIT, or it should be forwarded and PIT be updated accordingly. In the latter case, the router looks up AIB to check whether the content is available from other collaborative routers. If not, the Interest should be forwarded using the default policy in content-centric networks (*e.g.*, look up the outbound face in FIB and forward to the designated face; if FIB lookup fails, use a broadcast-like approach to forward the Interest). Otherwise, the Interest should be forwarded to the designated collaborative router. In order to do so, the router needs to look up FIB to determine the outbound face to reach the designated router. Note that most likely retrieving contents from collaborative routers within an autonomous system saves a noticeable time than getting it from the origin, as the latter typically requires traversing multiple autonomous systems and multiple interdomain links.

4.2 Collaborative Forwarding and Caching Mechanism

We next describe the collaborative forwarding and caching mechanism. In our design, each content router keeps track of the most popular C contents. We now assume a consistent popularity model where the most popular C contents at each router results in the same ranking sequence $\{r_1, r_2, ..., r_C\}$; under this assumption, we need to understand how to optimally distribute these C contents in the N caches in the corresponding N routers, whose sizes are $C_1, C_2, ..., C_N$, so that the average content access cost can be minimized in the network.

Recall that I_i denotes the average number of interests received by router i and d_{ij} denotes the link cost of nodes i and j (d_{ij} becomes the cost of accessing content from the local Content Store when $i = j$). Such costs can correspond to either intradomain routing weights or other performance-related metrics such as distance and latency. Then, for any cache unit in router i, the average cost of accessing this content requested by users is

$$cost_i = \frac{\sum_{k=1}^{N} I_k d_{ki}}{\sum_{k=1}^{N} I_k}, \tag{1}$$

where $cost_i$ is the weighted sum of pair-wise access costs from all N routers.

The following theorem states how contents should be optimally distributed to minimize the average content access cost:

Theorem 1. *Suppose the average cost for accessing the content in a cache unit of router i is $cost_i$. Without loss of generality, assume that after sorting, $cost_1 \leq cost_2 \leq ... \leq cost_N$. Also, suppose the ranking sequence is $r_1, r_2, ...r_C$ in descending order of popularity. Then, the solution that minimizes the average cost of accessing the most popular C contents in the network is to let the more popular content be cached at a place that has a lower cost,* i.e., *router 1 caches $r_1, ..., r_{C_1}$, router 2 caches $r_{C_1+1}, ..., r_{C_1+C_2}$, and so on.*

Proof. The proof is straightforward using contradiction. Suppose in the optimal solution that minimizes the average cost of accessing the C most popular contents, there exists two contents $r_i < r_j$ (r_i is more popular than r_j) stored at routers p and q, respectively. Routers p and q follows $cost_p > cost_q$. Then, by swapping content r_i and r_j, we get a smaller average content access cost because the frequency of accessing r_i is higher than r_j.

Theorem 1 sheds light on how we should design the distributed collaborative mechanism. More specifically, with the help of intradomain routing protocol, topological information is generally available to each router; as a result, each router can calculate $\{cost_i | i \in [1, N]\}$ for all collaborative routers in the network and sort all routers using these values. Following Theorem 1, the most popular contents should be stored by the router with the least cost, and the less popular contents by the router with a larger cost. Therefore, for any top-C popular content, each router knows not only which contents it should keep in its local Content Store, but also which collaborative router it can request this content from, if not locally available. Such availability information for the top-C contents is stored in AIB.

Example 1 We illustrate our design using a simple example shown in Fig. 2. In the example, there are three collaborative routers `R1`, `R2` and `R3`. These routers can cache 10 contents in total (the size of these three caches are 3, 4, and 3, respectively). The most popular 10 contents measured by these routers are consistent. Suppose $cost_1 \geq cost_2 \geq cost_3$. Based on Theorem 1, the most popular 3 contents should be cached in router `R3`, the next 4 contents should be cached in `R2`, and the next 3 contents should be cached in `R1`. As shown in the figure, for any incoming Interest requesting for the most popular C contents, AIB tells where it should be forwarded to (the content is either available from the router's local CS or other collaborative routers).

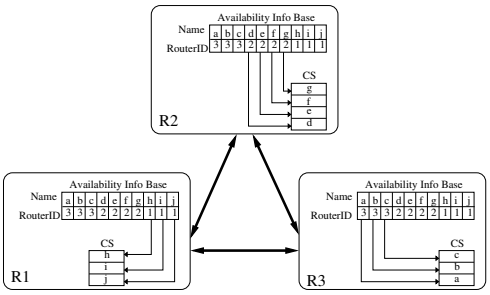


Fig. 2. A design for collaborative forwarding and caching.

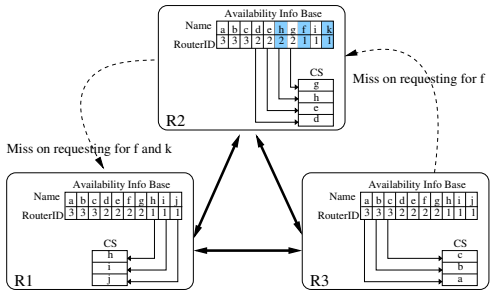


Fig. 3. An example of inconsistent ranking sequences.

4.3 Dealing with Inconsistent Popularity Ranking

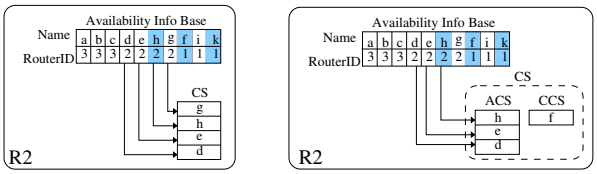
In practice, popularity rankings seen by different routers are more likely inconsistent. In such cases, the efficiency of our CFC scheme will be degraded.

Example 2 We illustrate inconsistency in popularity ranking through an example shown in Fig. 3. In this example, router R2's ranking sequence is slightly different from R1's and R3's. In R2's ranking sequence, the positions of content h and f are swapped and content k replaces j. As a result, router R2 caches contents $\{d, e, h, g\}$; however, routers R1 and R3 expect that R2 caches $\{d, e, f, g\}$ instead. Whenever R1 and R3 forward Interests for content f to R2, such Interests have to be further forwarded towards the origin by R2 (not shown in the figure). Similarly, R2 always forwards Interests for content f and k to R1, resulting in cache misses and further forwarding.

To address the above problem resulted by inconsistent popularity, we adopt a dual-segment cache design, namely, divide a router's Content Store into two segments: (1) the *Advertised Content Store* (*ACS* for short), denoted by C_i', is the regular collaborative cache that is operated the same way as described in the preceding subsection assuming consistent popularity; and (2) the *Complementary Content Store* (*CCS* for short), denoted by C_i'', is the cache space used for adapting to the inconsistency of popularity distribution. The rationale behind this dual-segment design is to leverage CCS to absorb contents that are supposed to be store at a router but are missing in its ACS due to inconsistent popularity.

Upon receiving an Interest, if the requested content is available locally, the router directly responds with the data. Otherwise, if the Interest comes from another collaborative router, the router forwards the Interest towards the content origin and stores the returned data into its Complementary Content Store when the data comes back; if the Interest comes from a requesting host directly, the router applies its knowledge of popularity-ranking sequence and checks whether the ranking of the requesting content is less than C; If yes, it forwards the Interest to the collaborative router designated by the sequence; otherwise, it forwards the Interest towards the origin.

Example 3 Fig. 4 shows an example of this design for router R2 in the previous example. With the single-segment design shown in Fig. 4(a), content f, which is



(a) Single segment (b) Dual segment

Fig. 4. An example of cache space division.

supposed to be cached in R2, is always missing due to the inconsistency of R2's ranking statistics. However, with the dual-segment design shown in Fig. 4(b), R2 only advertise 3 as its cache size. As a result, an extra cache unit can be used to store the missing content f. Therefore, future Interests requesting for f forwarded from routers R1 and R3 can be fulfilled by R2.

5 Adaptive Content Store Division

The impact of the preceding dual-segment design on the performance of collaborative forwarding and caching in content-centric networks could be subtle. On the one hand, a sufficiently large CCS is more favorable to adapt to the popularity inconsistency; and on the other hand, when the total size of the Content Store is fixed, a smaller CCS is more favorable, as the ACS could be larger to store more frequently requested contents in the network. Clearly there exist trade-offs when determining their sizes.

A straightforward solution is fixed division of ACS and CCS, *e.g.*, 90% dedicated to ACS and the remaining to CCS. However, the problem of fixed division is one size does not fit all, namely, routers may experience different levels of popularity inconsistency, and a fixed size may either over-estimate or under-estimate the inconsistency level, thus resulting an inefficient use of the cache space. This can be observed in the examples in Fig. 3 and Fig. 4. Router R1 and R3 experience less popularity inconsistency than R2 does. In fact, they have no cache misses after R2 switches to the dual-segment cache as shown in Fig. 4. As a result, there is no need to allocate any space for CCS in router R1 and R3.

We note that the cache miss rate plays an important role in the efficiency of dual-segment collaborative caching. On the one hand, when the miss rate is low, it implies a potentially oversized CCS and thus a waste of cache space. On the other hand, when the miss rate is high, then contents supposed to be stored in a designated collaborative router are actually not stored in it, resulting in additional costs to forward Interests to the designated router and then towards the origin.

We design a distributed, self-adaptive algorithm to address the above division problem; specifically, we adjust the division of ACS and CCS based on the dynamics experienced by the content routers. We define the *Locking Miss Rate* (LMR for short) to characterize the maximum miss rate (corresponding to a maximum level of popularity inconsistency) that a router would like to tolerate. Every router distributively adjusts its size of CCS to make its miss rate closely approach to LMR.

More specifically, a router starts with a pre-configured initial cache division, *e.g.*, 90% for the Advertised and 10% for CCS. It then begins measuring the

cache miss rate for all Interests received from other collaborative routers. Recall that the size of ACS and CCS are denoted by C_i' and C_i'' respectively.

We denote the measured cache miss rate by MR. If MR $\leq$ LMR, it is likely that the router experiences less popularity inconsistency than expected, we may have an oversized CCS, so the size of CCS C_i'' is halved so that we increase the size of ACS C_i' to cache more contents in the network. If MR $>$ LMR, the popularity inconsistency is likely under-estimated; therefore we should reduce the size of ACS to have a larger CCS in order for accommodating the inconsistency. However, instead of reducing the size of ACS C_i' aggressively, we linearly reduce it and allocate more space to CCS C_i'' based on the following equation:

$$C_i' \leftarrow (C_i' - \Delta C_i'), \tag{2}$$

where $\Delta C_i'$ is the size reduced by ACS C_i' and increased by CCS C_i''. In this equation, the right hand side is a rough estimation of the number of contents expected to be missed (with the current size of CCS C_i'' unchanged) after the size of ACS C_i' is reduced to $(C_i' - \Delta C_i')$. Ideally, these missed contents are expected to store at the extra space allocated to CCS C_i''. With this equation, the percentage the size of ACS C_i' is reduced can be calculated by

$$\frac{\Delta C_i'}{C_i'} = \frac{\mathrm{MR}}{1 + \mathrm{MR}}, \tag{3}$$

And as a result, the new ACS size should be $\frac{C_i'}{1+\mathrm{MR}}$. This algorithm is shown in Algorithm 1.

Algorithm 1 Self-Adaptive Division Algorithm

```
Initialize C_i' and C_i'' such that C_i' + C_i'' = C_i.
τ is a pre-determined threshold, e.g., τ ∈ [0, 0.5].
while TRUE do
    Measure MR for interests from other collaborative routers
    if (MR < (1 − τ) · LMR ) then
        C_i'' ← 0.5C_i''.
        C_i' ← C_i − C_i''.
    else if (MR > (1 + τ) · LMR) then
        C_i' ← C_i'/(1 + MR).
        C_i'' = C_i − C_i'.
    end if
end while
```

6 Evaluations

In this section we systematically evaluate the collaborative forwarding and caching design via simulations.

6.1 Simulation Setup

Network topologies. We use the pop-level network topology of Abilene [2] in simulations. Link costs are approximated by the measured end-to-end average

latencies using PlanetLab and are in the range of $[10, 40]$. Note that we also run simulations on other network topologies (*e.g.*, CERNET [12]); however, the results are consistent and thus we omit them here due to space limit. We include two additional nodes to represent content origins in Europe and Asia. For each of these additional nodes, we choose the geographically closest node in each network and connect them. Thus we have a trans-Atlantic link and a Trans-Pacific link. We refer to the topology with these additional nodes and links as the *extended topology*. In this extended topology, the Trans-Atlantic link cost is 120 and the Trans-Pacific link cost is set to 200. Unless otherwise specified, the cache size of each node is 100 units; we assume contents are chunked into unit-sized pieces, and the total number of content pieces varies from 2000 to 20,000. We assume a 5% LMR by default, and the initial splitting ratio is 90%-10%, *i.e.*, 10% of the cache space is used as the Complementary Cache in the beginning.

Content popularity and requests. To generate inconsistent content popularity distribution, we use a single ranking sequence following the Zipf distribution $[9, 13, 22]$ and randomly inject noises to 10% of the contents by shifting them to new positions up to a distance of 100 from their original positions in the sequence. We also evaluate different levels of inconsistency by shifting a different percentage of nodes in the original sequence in Section 6.4.

Interests are generated following the Zipf distribution. The number of interests each router receives directly from users per unit time is randomly chosen from 100 to 500. The adaptive cache division algorithm is run every 50 unit time. The simulation duration in total is 1000 unit time.

Algorithms. We evaluate three schemes using simulations. The first is our collaborative forwarding and caching scheme, referred to as the *global collaborative forwarding and caching (GCC)*. The second is a hierarchical caching approach [8], denoted as *HIE*, where nodes are divided into two clusters based on their locations and there are two additional nodes serving as the upper-level caches whose size is three times of the size of lower-level caches. The third is a locally clustered variant of GCC, referred to as the *local collaborative forwarding and caching (LCC)*, where nodes are divided into two clusters (in the same way as HIE) and each cluster run its own GCC independently.

Note that when evaluating these algorithms, all nodes have an identical caching size in order to make the comparison fair. However, due to the extra two upper-level caching nodes, the total cache size of HIE is larger than that of GCC and LCC.

Performance Metrics. We quantify the performance using two metrics. The first metric is the average cumulative link costs of accessing content by a user, referred to as the *Average Access Cost*. The second metric is the percentage of interests received by each cache that are not fulfilled locally, referred to as the *Cache Miss Ratio*.

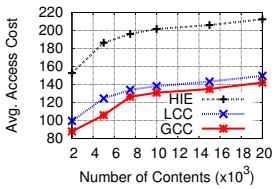


Fig. 5. Average content access cost.

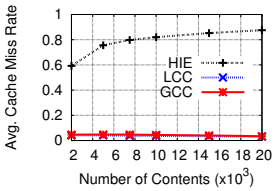


Fig. 6. Average cache miss rate.

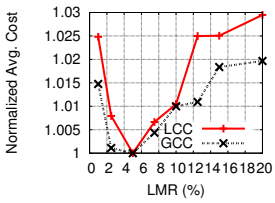


Fig. 7. Normalized average access cost vs. LMR.

6.2 Impacts of Number of Contents

We first quantify the performance by varying the number of delivered contents. Fig. 5 summarizes the results of average content access cost in the Abilene networks. We make the following observations. Firstly, as the number of delivered contents increases, the average access cost increases monotonically. This is because with more contents, the percentage of the contents that can be accessed from caches decreases and more contents have to been accessed directly from content sources, leading to higher costs.

Secondly, our design, GCC, outperforms HIE and cuts the cost by more than 30%. The reason is that GCC can take full advantage of peer caches, while HIE allows only very limited collaboration between the low-tier and high-tier nodes.

Thirdly, the localized collaborative scheme, LCC, enforces a clustered structure, therefore fewer nodes are collaborating with each other, resulting approximately 10% higher cost on average than the global collaborative scheme.

Fig. 6 summarizes the results of average cache miss rate. We observe that with HIE, the cache miss rate increases as the total number of contents increases, and that the miss rates are almost an order of magnitude higher than GCC and LCC. HIE does not use popularity information to guide forwarding and caching; instead, it searches local cache and upper-level cache only. As the number of contents increases, the miss rate increases since the percentage of contents can be stored in cache is limited by the size of cache storage. This explains why we see an increasing curve for HIE. For GCC and LCC, we observe that the miss rates largely remain the same, due to LMR being fixed to 5%.

6.3 Impact of Adaptive Cache Division

We next evaluate the impact of the adaptive cache division algorithm. More specifically, we quantify the performance of global and local collaborative caching by varying LMR.

Fig. 7 plots the impact of cache adaptation on average access cost. Note that the cost is normalized and the total number of contents is 10000 in this figure. We observe that the best performance can be achieved when LMR is approximately 5%. Note that when LMR is in the range of [0.025, 0.1], the average access cost is within 4% of the best performance, suggesting that the performance of GCC and LCC are not sensitive to LMR.

Fig. 8 plots the results of cache miss rate. We observe an increasing miss rate as LMR increases, however, it increases slower when LMR is larger than approximately 10%. With a larger LMR, the cache division algorithm tries to keep the cache miss rate around LMR, in order to make the most efficient use

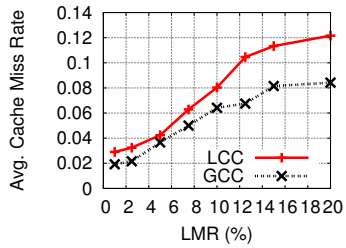


Fig. 8. Average cache miss rate vs. LMR.

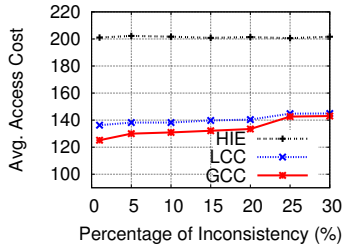


Fig. 9. Average access cost vs. popularity inconsistency.

of the cache space. When LMR increases from 10% to 20%, the cache miss rate only slightly increases because the miss rate is mainly contributed by the inconsistency of content popularity, which is generated by shifting 10% of contents from a given ranking sequence.

6.4 Impact of Popularity Inconsistency

We next evaluate the impact on the average access cost when the inconsistency in popularity changes. More specifically, we change the percentage of popularity inconsistency from 1% to 30%, while setting the total number of contents to 10000 and keeping all the other network settings the same as in Fig. 5. We summarize the results in Fig.9.

We make the following observations. Firstly, the higher the percentage of inconsistency, the higher the average access cost for GCC and LCC. We note that when the percentage of inconsistency is higher, each node allocates larger space to the Complementary Content Store in order to handle requests for contents that are not stored in its Advertised Content Store. The reason these contents are not stored in the Advertised Content Store is that different routers see different sets of contents for a given ranking range. With an increasing percentage of contents shifting from their original positions in the ranking sequence, the union of these different sets becomes larger, leading to a larger Complementary Content Store and a smaller Advertised Content Store, which eventually increases the average content access cost due to the reduced number of unique contents stored in the network.

Secondly, the average access cost of HIE remains almost the same. The reason is that HIE does not rely on consistency of popularity seen by different nodes, and thus is not affected by popularity inconsistency. However, even when the percentage of inconsistency is as high as 30%, GCC and LCC still improve the average access cost by approximately 30%, due to the reason that HIE does not take full advantage of cache available from other routers nearby, leading to an inefficient use of caching.

7 Conclusion

In this paper we propose a novel distributed, popularity-guided collaborative forwarding and caching design for content-centric networks, where we introduce an Availability Information Base to allow coordination between forwarding and

caching in content routers. In order to deal with popularity inconsistency in realistic networks, we also propose a self-adaptive dual-segment cache division algorithm. We evaluate our design via extensive simulations and demonstrate that our design improves content access cost and cache miss rate by at least 30% in a diverse network settings.

There are many avenues to future work. We plan to implement a prototype based on CCNx [1] and conduct medium- to large-scale experiments. This also gives us opportunities to investigate the complexity and feasibility of the proposed framework. We also plan to theoretically model and analyze the impacts of popularity inconsistency on the effectiveness of the proposed design.

Acknowledgements. This work was partially supported by the National Natural Science Foundation of China under Grant No. 61073192, by the Grand Fundamental Research Program of China (973 Program) under Grant No. 2011CB302905, by the New Century Excellent Talents Program, and by the Fundamental Research Funds for Central Universities under Grant No. WK0110000014.

References

1. CCNx. `http://www.ccnx.org`
2. Abilene: Internet2 IP IGP Metrics. `http://noc.net.internet2.edu/i2network/maps--documentation/maps.html`
3. Akamai Technologies: http://www.akamai.com
4. Amazon CloudFront Express: http://www.amazon.com/cloudfront
5. Anand, A., Dogar, F., Han, D., Li, B., Lim, H., Machadoy, M., Wu, W., Akella, A., Andersen, D., Byersy, J., Seshan, S., Steenkiste, P.: XIA: An architecture for an evolvable and trustworthy internet. Tech. Rep. CMU-CS-11-100, Carnegie Mellon University (Feb 2011)
6. Anderson, T., Birman, K., Broberg, R., Caesar, M., Comer, D., Cotton, C., Freedman, M., Haeberlen, A., Ives, Z., Krishnamurthy, A., Lehr, W., Loo, B.T., Mazires, D., Nicolosi, A., Smith, J., Stoica, I., van Renesse, R., Walfish, M., Weatherspoon, H., Yoo, C.: NEBULA - a future internet that supports trustworthy cloud computing. White Paper (2010)
7. BitTorrent: http://www.bittorrent.com
8. Borst, S., Gupta, V., Walid, A.: Distributed caching algorithms for content distribution networks. In: IEEE INFOCOM 2010. San Diego, CA (march 2010)
9. Breslau, L., Cao, P., Fan, L., Phillips, G., Shenker, S.: Web caching and zipf-like distributions: evidence and implications. In: Proc. IEEE INFOCOM 1999. vol. 1 (mar 1999)
10. Carofiglio, G., Gallo, M., Muscariello, L., Perino, D.: Modeling data transfer in content-centric networking. In: ITC '11. pp. 111–118 (2011)
11. CDNetworks: http://www.cdnetworks.com
12. Cernet: Cernet Topology, http://www.edu.cn/20060111/3170194.shtml
13. Cheng, X., Dale, C., Liu, J.: Statistics and social network of youtube videos. In: Proc. IWQoS 2008. pp. 229 –238 (june 2008)
14. Chow, C.Y., Leong, H.V., Chan, A.T.S.: Distributed group-based cooperative caching in a mobile broadcast environment. In: Proc. of Mobile Data Management '05. pp. 97–106 (2005)
15. Dannewitz, C.: NetInf: An information-centric design for the future internet. In: Proc. 3rd GI/ITG KuVS Workshop on The Future Internet (May 2009)

16. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., Weihl, B.: Globally distributed content delivery. IEEE Internet Computing September/October, 50–58 (2002)
17. Dykes, S., Robbins, K.: A vaibility analysis of cooperative proxy caching. In: IEEE INFOCOM 2001. pp. 1205–1214. Anchorage, AK (april 2001)
18. eMule-Project.net: http://www.emule-project.net
19. Erman, J., Gerber, A., Hajiaghayi, M.T., Pei, D., Sen, S., Spatscheck, O.: To cache or not to cache: The 3g case. IEEE Internet Computing 15, 27–34 (2011)
20. Fan, L., Cao, P., Almeida, J., Broder, A.Z.: Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Trans. Netw. 8 (June 2000)
21. Gadde, S., Chase, J., Rabinovich, M.: Web caching and content distribution: a view from the interior. Computer Communications 24(2), 222 – 231 (2001)
22. Gill, P., Arlitt, M., Li, Z., Mahanti, A.: Youtube traffic characterization: a view from the edge. In: Proc. ACM IMC 2007. ACM, New York, NY, USA (2007)
23. Hefeeda, M., Noorizadeh, B.: On the benefits of cooperative proxy caching for peer-to-peer traffic. IEEE Transactions on Parallel and Distributed Systems 21, 998–1010 (2010)
24. Helgason, O., Karlsson, G.: Podnet: A system architecture for opportunistic content distribution. Tech. rep., Royal Institute of Technology (KTH) (Feb 2010)
25. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.: Networking named content. In: ACM CoNEXT 2009. Rome, Italy (Dec 2009)
26. Korupolu, M.R., Dahlin, M.: Coordinated placement and replacement for large-scale distributed caches. IEEE Transactions on Knowledge and Data Engineering 14, 1317–1329 (2002)
27. Level3 Communications: http://www.level3.com
28. Limelight Networks: http://www.limelight.com
29. Ni, J., Tsang, D.: Large-scale cooperative caching and application-level multicast in multimedia content delivery networks. Communications Magazine, IEEE 43(5) (may 2005)
30. Pathan, M., Buyya, R., Vakali, A.: Content delivery networks: State of the art, insights, and imperatives. In: Buyya, R., Pathan, M., Vakali, A. (eds.) Content Delivery Networks, Lecture Notes in Electrical Engineering, vol. 9, pp. 3–32. Springer Berlin Heidelberg (2008)
31. Psaras, I., Clegg, R., Landa, R., Chai, W., Pavlou, G.: Modelling and evaluation of ccn-caching trees. In: NETWORKING 2011, Lecture Notes in Computer Science, vol. 6640, pp. 78–91. Springer Berlin / Heidelberg (2011)
32. Sailhan, F., Issarny, V.: Cooperative caching in ad hoc networks. In: Mobile Data Management, Lecture Notes in Computer Science, vol. 2574, pp. 13–28. Springer Berlin / Heidelberg (2003)
33. Sarkar, P., Hartman, J.H.: Hint-based cooperative caching. ACM Trans. Comput. Syst. 18 (November 2000)
34. Wessels, D., Claffy, K.: Internet cache protocol (icp), version 2 (1997)
35. Wolman, A., Voelker, M., Sharma, N., Cardwell, N., Karlin, A., Levy, H.M.: On the scale and performance of cooperative web proxy caching. In: Proc. ACM SOSP 1999. pp. 16–31 (1999)
36. Xie, H., Shi, G., Wang, P.: TECC: Towards collaborative in-network caching guided by traffic engineering. In: IEEE INFOCOM 2012. Orlando, FL (mar 2012)
37. Yin, L., Cao, G.: Supporting cooperative caching in ad hoc networks. IEEE Transactions on Mobile Computing 5, 77–89 (2006)
38. Zhang, L., Estrin, D., Burke, J., Jacobson, V., Thornton, J.D., Smetters, D.K., Zhang, B., Tsudik, G., kc claffy, Krioukov, D., Massey, D., Papadopoulos, C., Abdelzaher, T., Wang, L., Crowley, P., Yeh, E.: Named data networking (NDN) project. Tech. Rep. NDN-0001, Palo Alto Research Center (PARC) (Oct 2010)