Towards Energy-Efficient Caching in Content-Centric Networking

Hao Wu[1], Bin Liu[1], Yang Li[1], Huichen Dai[1], Yi Wang[2], and Yaxuan Wang[3]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
[2]Huawei Future Network Theory Lab, Hong Kong, 999072, China
[3]Beijing University of Posts and Telecommunications, Beijing, 100876, China
Email: {wu-h11, liyang-12}@mails.tsinghua.edu.cn; {liub, dhc}@mail.tsinghua.edu.cn; wy@ieee.org; 2013210119@bupt.edu.cn

Abstract —Content-Centric Networking (CCN) is regarded as a promising architecture for the future Internet because of its built-in caching capability, which will potentially reduce the energy consumption. In this work, we focus on the cache allocation and replacement strategy in CCN, aiming at improving the energy efficiency and resource utilization within the scope of an ISP. We first explain the Interests aggregation in Pending Interest Table (PIT) is very weak and the in-network energy consumption largely relies on the average bit-level Data download hop count. Then we present a heuristic scheme called Distributed Energy-Efficient (DEE) caching that dynamically replicates Data among the delivery path based on: a) the popularity-hop product; b) the impacts of bias requesting, information inconsistency and Interest timeliness. Finally, we evaluate the performance of DEE. The proposed scheme exhibits constantly superior performance not only in energy efficiency, but also in cache hits and content download delay, which enhances user experience and cache network stability.

Index Terms—Next generation network, Content-Centric Networking, distributed cache allocation scheme

I. Introduction

Today's Internet is primarily associated with the distribution of content, rather than IP-based host-to-host communication. Simultaneously, the explosive growth in Internet traffic, ranging from conventional text, image, and multimedia data to user-generated contents, poses a number of challenges to the network in terms of bandwidth capacity, energy consumption and QoS. To address these problems, *Content-Centric Networking* (*CCN*) has been proposed as a promising future Internet architecture and a variety of instances, such as NDN [1], PURISUIT [2], DONA [3] and COMET [4], have been extensively studied. CCN packets carry content names instead of destination addresses carried in IP packets, which are used by CCN routers and hosts to match user requests (called *Interests*) with the actual content objects (called *Data*). On the other hand, CCN routers are equipped with cache to store the frequently requested data in order to serve the subsequent Interests. The two inherent features decouple content from host location and balance network load, which undoubtedly improves the network performance [1], [5].

A straightforward merit of CCN caching is the improvement of energy efficiency. With Interests being served by in-networks caches, the energy consumption for transporting Data can be significantly reduced because the delivery distance is cut down. In this work, we explore to what extent that caching can improve energy efficiency within the scope of an ISP, which is subject to the specific caching strategy. By default, CCN adopts *Caching Everything Everywhere* (*CEE*) [1] strategy. That is, every router on Data download path is supposed to replicate a copy of passing Data and manage the cached replicas with *Least Recently Used* (*LRU*) replacement policy. Since the subsequent Interests for the same Data will be readily served by the first cache node, the replicas on the rest path become redundancy, which lowers the resource utilization. Compared with the exponentially increasing size of Internet contents, the storage of CCN routers is very limited that we need to carefully decide which Data are qualified to be stored, where to store them, how long should they be hosted and how to evict a cached Data when there is no room to accommodate the new one.

Cache framework, allocation and replacement policies in IP-based networks have been well studied for years. Specifically, web cache uses proxy servers to store static contents without supporting dynamic routing. Peer-to-Peer (P2P) applications and nodes make up a multi-path cache network. Contents are automatically distributed among the online nodes, which is hard to manage in the global view. In contrast, Content Delivery Network (CDN) builds a fully centralized structure that actively "pushes contents" to end users. Thus, caches are widely deployed among the network edge. Caching strategy for CDN is usually subject to the central controller who computes the cache layout [6]. The decoupling of computation and storage brings huge control and computational overheads. On the other hand, the existing

Manuscript received September 24, 2016; revised December 20, 2016.
This work was supported by Huawei Innovation Research Program (HIRP), NSFC (61373143, 61432009, 61602271), China Postdoctoral Science Foundation (2016M591182, 2015T80089), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20130002110084).
Corresponding author email: liub@mail.tsinghua.edu.cn.
doi:10.12720/jcm.11.12.1036-1047

cache networks running on application layer can flexibly place contents and redirect the Interests to the desired cache servers via packet classification or tunneling. However, CCN is deployed in layer-3 and deals with fine-grained chunk level Data whose location changes very frequently among the cache network. In addition, the *Forwarding Information Base* (*FIB*) size of a CCN router is orders of magnitude larger than that of an IP router, because the content name has much larger name space compared with the fixed-length IP address. Thus, it is infeasible and inefficient for a CCN router to frequently modify FIB entries when the cache layout changes, which will incur huge overhead and cache miss. Instead, CCN adopts the lightweight *on-path* caching. That is, a) Data can only be cached by the routers along the download path [1]; b) since FIB keeps unchanged, the CCN router does not know where the desired Data is cached, so it constantly forwards Interests towards the static content server. In this way, the forwarding path for a particular Interest is relatively fixed, so is the Data download path which is the same to the requesting path but in reverse direction. Without the help of FIB, the IP-based caching schemes cannot be directly transplanted to CCN. These new challenges make CCN caching a brand new research area.

Towards the energy-efficient in-network caching in CCN, we first reveal very few Interests can be aggregated in *Pending Interest Table* (*PIT*) in reality. With the assumption that the aggregation is ignored, we then explain the energy consumption roughly depends on the Data download hop count. Guided by the insight, we propose the distributed and energy-efficient heuristic scheme that reduces 30% energy consumption against the default CEE strategy in CCN. In addition to the caching algorithm, we pay attention to systematic implementation issues including information inconsistency problem, bias requesting and aging out stale statistics.

Specifically, our contributions include:

1) We explore the effect of Interests aggregation in PIT and conclude very few Interests can be actually aggregated. With the assumption of the aggregation being ignored, we find the energy consumption largely relies on the average bit-level Data download hop count.

2) We present a heuristic scheme named Distributed Energy-Efficient (DEE) caching that places Data on the download path based on the popularity-hop product. Besides, we realize bias requesting and information inconsistency problems and consider their impacts when making caching decisions. We also present the dilution method to age out inactive Data and accelerate the cache renewal process.

3) We evaluate the performance of DEE under real-life networks. Numerous results demonstrate the proposed scheme is effective, steady and scalable.

The rest of the paper is organized as follows: Section II briefly introduces the background of CCN. Section III explores Interests aggregation in PIT. Then it translates the problem of saving energy to reducing the average hop count. In Section IV, the systematic descriptions of DEE are presented. The simulation settings, parameters and numerical results are discussed in Section V. Then, Section VI reviews the related works. Finally, Section VII concludes the paper.

II. A Brief Overview of Content-Centric Networking

As mentioned, there are two basic kinds of packets in CCN: Interest and Data. Every Data packet is in fixed size and assigned a globally unique *content identifier* (e.g., the hash value of URL), which is used by routers and hosts to identify the Data.

CCN routers are equipped with cache medium that can store the passing Data to serve the future Interests. The cache storage in routers is called *Content Store* (*CS*) (we use the terms *CS* and *cache* interchangeably in this paper). If the Interest is not satisfied by local CS, it is forwarded by the name-based FIB. CCN routers keep track of all the currently unsatisfied Interests in a dedicated data structure called *Pending Interest Table* (*PIT*). Therefore, CCN achieves the connectionless communication with the help of the *stateful* data plane consists of CS, PIT and FIB.

Shown in Fig. 1, the end user who is requesting Data is called *consumer*, while the static content server is called *producer*. In the scope of an ISP, the CCN routers are categorized to the access routers that directly face to consumers; the core routers that only connect to other routers; the gateway routers that can talk to (may not directly connect) outside producers.

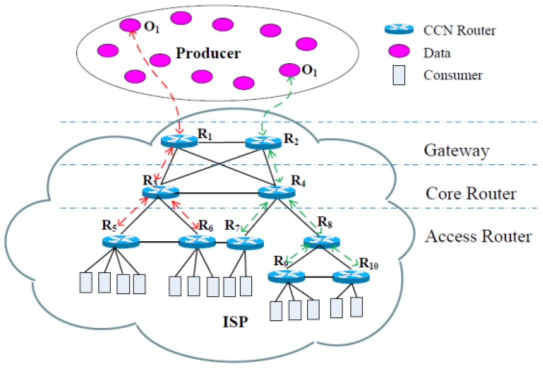


Fig. 1. The gateways interconnect with other ISPs, either peer-ISPs or provider-ISPs. Access routers receive Interests from end users or customer networks. The dash lines characterize the logical forwarding path for Data O_1. The Data requesting and downloading are on the symmetrical path but in opposite directions.

The content requesting and download processes are briefly described as:

1) A consumer sends an Interest attached with the content identifier to the network;

2) When the router receives the Interest, it first checks if the local CS has the requested Data. If so, the router encapsulates the content payload into Data packet, then returns it to the Interest's incoming port and drops the

Interest in the same time. Otherwise, the router looks up local Pending Interest Table (PIT) to check whether the Interest has already been forwarded earlier. If yes, the router adds the Interest's incoming port in the PIT entry and drops the Interest. In this way, the Interest is aggregated in PIT. If no, the router forwards it by looking up Forwarding Information Base (FIB) and creates a new entry in PIT;

3) When the target Data returns, the router decides whether to store a copy of the Data in local CS, according to the currently deployed caching policy. Then, it forwards the Data to the port(s) recorded in the corresponding PIT entry which is deleted soon after. Thus, the Data is delivered along the reverse direction of the previous Interest forwarding path.

As mentioned, the original CCN design adopts the *on-path* caching. That is, the router can only selectively cache those Data passing itself, but will never take the initiative to fetch a Data from some other content source.

III. System Assumptions and Energy Consumption

We explore the Interests aggregation in PIT. Based on the findings, we describe our system assumptions and energy consumption insights.

A. *Explore Interests Aggregation in PIT*

Shown in Section II, PIT has two inherit functions: one is recording the Data delivery path to avoid flooding; the other is aggregating the repetitive Interests for the same content. We first evaluate to what extent the Interests can be aggregated in PIT. We explore PIT's aggregation effect through both the theoretical analysis and the simulation with real trace, under LRU cache replacement and Poisson arrivals. The result reveals very few Interests can be actually aggregated.

1) Interests aggregation in PIT based on the theoretical analysis

Ref. [7] theoretically describes the CS characteristic time and CS hit probability under LRU replacement policy with Poisson arrivals. But it assumes Data is fetched by the router's CS without delay, which is not realistic. Recently, Ref. [8] improves the previous work by involving the non-zero data download delay. It also presents the probability of having an existing entry in PIT when an Interest arrives, namely the Interest's aggregation probability in PIT. But it does not show the overall PIT aggregation effect. We go one step further by evaluating the amount of the aggregated Interests.

Assume there are N distinct contents: O_1, O_2... O_N. λ_k ($1 \le k \le N$) denotes the arrival rate for the Interest for O_k and D_k denotes the Data download delay for O_k. Without loss of generality, each Data packet is in unit size. The CS uses LRU replacement policy with fixed storage C. Interests arrive to CS under Poisson process. According to [8], a PIT entry's TTL (defined as T) after it is created is a constant value in this scenario and is obtained by solving the fixed-point formula

$$\sum_{k=1}^{N} \frac{e^{\lambda_k T} - 1}{\lambda_k \mathbf{E}[D_k] + e^{\lambda_k T}} = C \tag{1}$$

The probability of aggregating an incoming Interest for O_k is

$$p_k = \frac{\lambda_k \mathbf{E}[D_k]}{\lambda_k \mathbf{E}[D_k] + e^{\lambda_k T}} \tag{2}$$

Limited by the computational complexity of solving (1), we set CS size to C=1,000, the content category size to N=10,000. The overall Interests arrival rate is $\lambda = \sum_{k=1}^{N} \lambda_k$ =1,500 Interests/second. The Data download delay is an exponentially distributed variable with the average value $\mathbf{E}[D_k]$ =100 ms ($1 \le k \le N$). Interests popularity follows Zipf distribution [9] with the parameter $\alpha = 0.9$. By solving (1), we get T=10 ms. For each single content, we plot the probability of an Interest being aggregated in PIT in Fig. 2. We further calculate the expected percentage value of the aggregated Interests:

$$\sum_{k=1}^{N} \lambda_k p_k \Big/ \sum_{k=1}^{N} \lambda_k = 0.9\% .$$

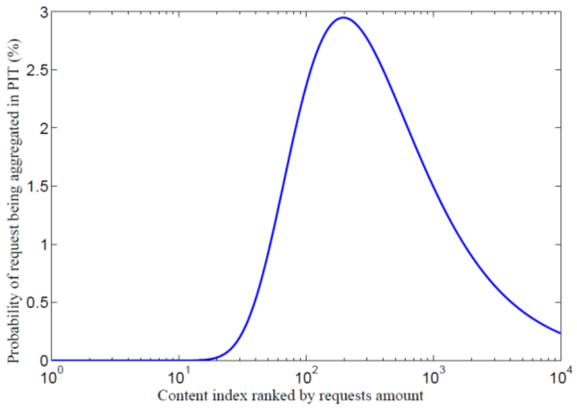


Fig. 2. Interest aggregation probability in PIT

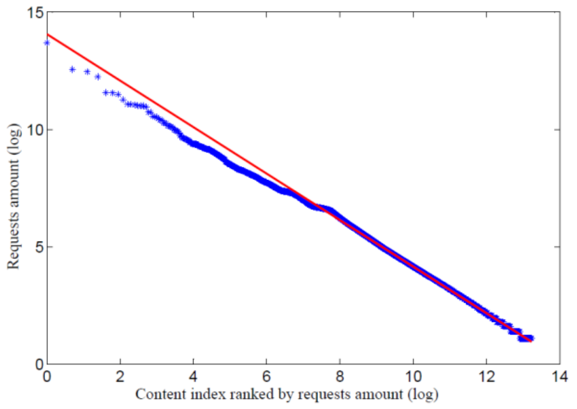


Fig. 3. The rank-amount scatter of real trace and its linear fit (red line), whose slope is the parameter of the Zipf distribution followed by the content popularity.

TABLE I: SIMULATION RESULTS WITH THE REAL TRACE

Percentage of Router CS size w.r.t. Content category size	CS hits	CS miss	CS hit ratio (%)	PIT hits	PIT miss	PIT aggregation ratio (%)
1	7,724,295	7,093,260	52.13	203,606	6,889,654	1.37
2	9,036,597	5,780,958	60.99	148,809	5,632,149	1.00
3	9,806,178	5,011,377	66.18	124,006	4,887,371	0.84
4	10,441,085	4,376,470	70.46	96,481	4,279,989	0.65
5	10,923,375	3,894,180	73.72	77,551	3,816,629	0.52
6	11,266,208	3,551,347	76.03	69,416	3,481,931	0.47
7	11,508,074	3,309,481	77.67	65,958	3,243,523	0.45
8	11,695,159	3,122,396	78.93	63,280	3,059,116	0.43
9	11,860,378	2,957,177	80.04	61,247	2,895,930	0.41
10	12,008,612	2,808,943	81.04	59,389	2,749,554	0.40

We derive the insights: for the popular contents, they are held in CS constantly, so the Interests are served in the CS which locates in front of PIT. For the unpopular contents, the requesting rate is so low that the Interests arrival interval is even larger than the Data download delay (RTT), for which reason the Data will reach the router and delete the PIT entry before the next Interest comes. So there is no Interests aggregation in PIT either. For contents in the moderate popularity, they have duplicated Interests in PIT, considering they have the highest Interests aggregation probability in Fig. 2. However, since the Interests popularity follows Zipf distribution in reality [9], this kind of contents does not have many Interests because most of the Interests are covered by the few hottest contents. As a result, very few Interests can be actually aggregated in PIT.

2) Simulation based on real trace

In addition to the theoretical analysis, we use a 1-hour realistic campus traffic trace collected in Aug 2015 to simulate the Interests aggregation in PIT. There are 14.82 M Interests for 556,896 unique contents. We first assume the Interests popularity follows Zipf distribution. To verify it, we rank the contents by Interests amount and plot the logarithmic rank-amount scatter in Fig. 3. Since the amount of the *i*-th ranked Interest in Zipf distribution is $TotalAmount \cdot i^{-\alpha} / \sum_{k=1}^{N} k^{-\alpha}$, the *linear shape* for the logarithmic rank-amount scatter verifies our assumption of the Zipf distribution in terms of Interest popularity. We plot the linear fit for the scatter and derive the parameter (i.e., the slope for the linear fit line) in the Zipf distribution is $\alpha = 0.9905$.

Similar to the analysis above, we assume the Data download delay follows the exponential distribution with average value $\mu = 100$ ms. We generate 10,000 samples under this distribution. The 95% confidence interval (CI) is [82.84 ms, 122.72 ms], so we set the PIT entry timeout to a large enough value: $122.72 \times 2 = 245.44$ ms. That is, as soon as a new PIT entry is created, we attach it with the Data download delay which is a variable following the exponential distribution with $\mu = 100$ ms. We bind a timer (245.44 ms) to the entry at the same time. The PIT entry is deleted either when the Data returns, or when the timer expires. Thus, a PIT entry's TTL is **min**{*Data download delay*, *timer*}.

With these settings fixed, we change the CS size from 1% to 10% w.r.t. content category size and derive the results in Table I. As is shown, even if the CS budget is 1% of content category, only 1.37% of total Interests are aggregated. As the CS budget increases, the percentage of aggregated Interests rapidly drops.

From both the theoretical analysis and the simulation with real trace, we reveal: though a single Data chunk is supposed to be multiplexed and serve multiple Interests in the original CCN design, almost all the Interests and Data are still following the one-to-one mapping.

B. System Assumptions and Energy Consumption

Inspired by the insights in terms of Interests aggregation in PIT, we can *safely ignore the Interests aggregation*. Then, in the ISP domain, *the bit-level energy consumption only depends on the average bit-level hop count*. The detailed proof is presented in Appendix. Therefore, the objective of minimizing the energy consumption is equivalent to minimizing the average bit-level hop count. We regard the popularity-hop product as the main caching benefit in the heuristic scheme to be introduced in the following section.

IV. DISTRIBUTED ENERGY-EFFICIENT CACHING

Guided by the design insights towards reducing energy consumption, we propose Distributed Energy-Efficient (DEE) caching, a heuristics that makes caching decisions based on the popularity-hop product. In addition, we present dilution methods to age out the stale Data and accelerate the cache renewal process.

Let $\mathcal{R}$ be the set of CCN routers in an ISP, among which $\mathcal{A}$ is the set of access routers that directly receive Interests from consumers. Define $\mathcal{O}$ as the content category. The Interest and Data for content O_k are labeled as $Interest_k$ and $Data_k$ respectively.

A. *Caching Benefit*

We define $POP_{k,i}$ as the accumulated amount of *Interest*$_k$ arriving at R_i, namely the popularity for *Interest*$_k$ in R_i. Recall $h_{k,i}$ is the hop count to download *Data*$_k$ from the content source to R_i. Since we do not consider the energy consumption outside the ISP, the maximal value of $h_{k,i}$ is the hop-level distance between R_i and the gateway on *Data*$_k$'s delivery path, as shown in the model in Fig. 1. Towards reducing the average bit-level Data download hop count, it is straightforward to pull more bits closer to the consumer. Thus, we set the benefit for caching *Data*$_k$ at R_i as the popularity-hop product

$$b_{k,i} = POP_{k,i} \cdot h_{k,i} \tag{3}$$

DEE takes the caching benefit as the quantitative criterion for CS admission and LRU as the cache replacement policy. Each router holds a priority queue *BQ* recording the 3-tuple <content identifier, caching benefit, popularity> for the candidate contents.

B. *Bias Requesting and Fair Aggregation at Core Routers*

Since the core router $R_i \in C_{\mathcal{A}}\mathcal{R}$ (Recall $\mathcal{A} \subseteq \mathcal{R}$ is the set of access routers) does not directly face to consumers, the access routers have to periodically encapsulate their collected Interests popularity into update datagrams and send to R_i. The core router aggregates the update information and renews its priority queue BQ.

An important observation is the *bias requesting* when the core router aggregates multiple descendant routers' popularity datagrams for the same Interest. For example, consider R_2 and R_3 are two access routers in Fig. 4. The popularity for the three Interests sent by R_2 and R_3 are aggregated by R_1. For each router, assume the CS can only accommodate one Data and the hop counts for downloading every Data are same. To avoid redundant replicas, a single Data is supposed to be cached at most once on the delivery path. In this way, every router caches the most beneficial Data locally. The cache layout is shown in Fig. 4(a). However, we observe that though *Data*$_1$ has the maximal benefit in R_1, the components are very biased distributed among the two branches and most of the benefit is seen in R_3. As the key position covering the whole area, R_1 should cache the *Data that fairly benefits all the branches*. Therefore, DEE involves a bias factor *biaf* to refine the aggregation.

Let $biaf = \dfrac{b_{min,agg}}{b_{max,agg}}$, where $b_{max,agg}$ is the largest benefit value in terms of the Data among the branches, while $b_{min,agg}$ is the smallest. In addition to summing the benefit components (say b_{sum}), DEE adds the weight with an extra increment $b_{sum} \cdot biaf$. In the example, the *biaf*-attached benefit in R_1 and the resulting cache layout are shown in Fig. 4(b). The globally beneficial *Data*$_2$ occupies R_1's CS. Compared with Fig. 4(a), R_2 gets more benefit by storing *Data*$_3$ rather than *Data*$_1$ on the path, while R_3's benefit roughly keeps unchanged. Note that R_1's actually derived benefit is still the *basic benefit* because the extra part is only used to add weights in making the decision. As a result, the network-wide gained benefit increases.

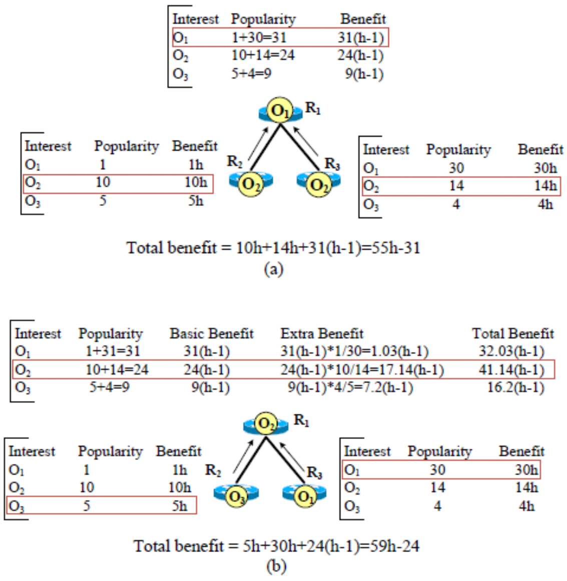

Interest Popularity Benefit
O1 1+30=31 31(h-1)
O2 10+14=24 24(h-1)
O3 5+4=9 9(h-1)
O1 R1
Interest Popularity Benefit
O1 1 1h
O2 10 10h
O3 5 5h
R2 O2
R3 O2
Interest Popularity Benefit
O1 30 30h
O2 14 14h
O3 4 4h
Total benefit = 10h+14h+31(h-1)=55h-31
(a)
Interest Popularity Basic Benefit Extra Benefit Total Benefit
O1 1+31=31 31(h-1) 31(h-1)*1/30=1.03(h-1) 32.03(h-1)
O2 10+14=24 24(h-1) 24(h-1)*10/14=17.14(h-1) 41.14(h-1)
O3 5+4=9 9(h-1) 9(h-1)*4/5=7.2(h-1) 16.2(h-1)
O2 R1
Interest Popularity Benefit
O1 1 1h
O2 10 10h
O3 5 5h
R2 O3
R3 O1
Interest Popularity Benefit
O1 30 30h
O2 14 14h
O3 4 4h
Total benefit = 5h+30h+24(h-1)=59h-24
(b)

Fig. 4. For simplicity, assume the download hop counts for different Data are equivalent at every router: $h_{i,1}=h-1$, $h_{i,2}=h_{i,3}=h$ ($1 \le i \le 3$). R_1 aggregates the Interests popularity sent by the branches. In Fig. 4(a), R_1 only considers the sum of the benefits, and stores the most beneficial one. In Fig. 4(b), in addition to the basic benefit, R_1 also computes an extra benefit that adds more weights to the Data that can fairly benefit the branches. As a result, the later scheme gets more network-wide benefit.

C. *Request the Data*

1) *Process the interest at CCN routers*

Based on the on-path caching and forwarding model explained in Section I, DEE treats the request and download path for the Data as a whole cache space and explores the position with maximal caching benefit to place the Data.

DEE adds the 3-tuple <*rid*, b_{max}, *if_empty*> into Interest packet. Among the routers traversed by *Interest*$_k$, *rid* and b_{max} indicate the most beneficial router's ID and the corresponding benefit value. Flag *if_empty*=TRUE means the chosen router in *rid* has enough CS storage to accommodate *Data*$_k$ without kicking out any existing Data, while *if_empty*=FALSE means the router has to replace another cached Data with *Data*$_k$ in CS. *if_empty* is initialized as FALSE by the consumer. Accordingly, when the Data returns, it has a field *rid*$_{ca}$ recording the selected router to cache the replica. In order to update the hop-level distance information for on-path routers, Data packet has another field *hops* indicating the hop count it has traveled originating from the content source.

Shown in Algorithm 1, when R_i receives *Interest*$_k$, it first checks local CS. If hit, the router drops *Interest*$_k$ and returns the matched Data (Line 2-12). If there is a better position in the downstream nodes for placing *Data*$_k$ (Line

3-6, recall $b_{k,i}$ is the local caching benefit for $Data_k$ recorded by R_i), the cache point will be pulled down: the Data packet will record the candidate router ID so it can be cached there (Line 4). Meanwhile, the replica in current CS is to be deleted in order to avoid duplicate copies (Line 6). $Data_k.hops$ is set to the distance from R_i to the gateway (denoted by $HOPS_{i\rightarrow gw}$) because the Data replica in current CS node is invalid (Line 5). Otherwise, it is initialized as 1 (Line 10) and increasing hop by hop. Since the CS is managed under LRU replacement policy in DEE, it is updated if the matched Data is not deleted (Line 8). If CS misses and $Interest_k$ is already recorded in R_i's PIT, the router adds $Interest_k$'s incoming port and *rid* value in PIT entry and drops $Interest_k$ (Line 13-15). If $Interest_k$ is not found in PIT, the router creates a new entry for it (Line 17). In the case that either R_i has available CS storage (Line 18-22) or there is the replaceable Data in CS (Line 23-26), DEE selects R_i as the new candidate router for caching $Data_k$. Note that DEE preferentially selects the CS node with spare room in order to make the most of the storage on the path (Line 18). Finally, R_i forwards $Interest_k$ to the next router (Line 27).

Algorithm 1 Receive_Interest

```
1   function RECV_INT($R_i$, $Interest_k$)
2       if $Data_k \in R_i.CS$ then
3           if $b_{k,i} < Interest_k.b_{max}$ then
4               $Data_k.rid_{ca} \leftarrow Interest_k.rid$
5               $Data_k.hops \leftarrow HOPS_{i\rightarrow gw}$
6               delete $Data_k$ in $R_i.CS$
7           else
8               CS_UPDATE_WITH_LRU($R_i$, $Data_k$)
9               $Data_k.rid_{ca} \leftarrow$ NULL
10              $Data_k.hops \leftarrow 1$
11          FORWARD($R_i$, $Data_k$)
12          return HIT
13      else if $Interest_k$ is recorded in $R_i.PIT$ then
14          PIT_ADD_INPORT_AND_RID($R_i$, $Interest_k$)
15          return MISS
16      else
17          PIT_ADD_ENTRY($R_i$, $Interest_k$)
18      if $R_i.CS$ has enough storage to cache $Data_k$ then
19          if $Interest_k.if\_empty$=FALSE OR ($Interest_k.if\_empty$=TRUE AND $b_{k,i} > Interest_k.b_{max}$) then
20              $Interest_k.rid \leftarrow i$
21              $Interest_k.b_{max} \leftarrow b_{k,i}$
22              $Interest_k.if\_empty \leftarrow$ TRUE
23      else if There exists $Data_q \in R_i.CS$ satisfying $b_{q,i} < b_{k,i}$ then
24          if $Interest_k.if\_empty$=FALSE AND $b_{k,i} > Interest_k.b_{max}$ then
25              $Interest_k.rid \leftarrow i$
26              $Interest_k.b_{max} \leftarrow b_{k,i}$
27      FORWARD($R_i$, $Interest_k$)
28      return MISS
```

2) *Information inconsistency problem*

As discussed, there exists Data download delay (RTT) from the time when a router decides to cache the Data to the time when the Data packet returns. During this interval, other Interests and Data may reach the router and change the CS profile. So it is possible that the Data cannot be actually cached when it reaches the router because the previously predicted caching condition is no longer met. We refer to this as the *information inconsistency problem*.

To address it, as the router gets the Interest, it records the previously selected router's ID ($Interest_k.rid$) in PIT (Line 14, 17 in Algorithm 1), while in the original CCN design, PIT only records the Interest's incoming port. For the example in Fig. 5, the forwarding path contains 5 routers, from access router R_1 to gateway R_5. Assume the Interest's *rid* changes at R_1, R_2 and R_4, then the whole path is segmented into three sub-paths by the three "change points". By checking the *rid* value recorded at the change point router, we can obtain *the best caching position for the Data on the next sub-path*, which acts as the alternative node if the router on the upstream path fails to accommodate the Data. More details are shown in the following paragraphs.

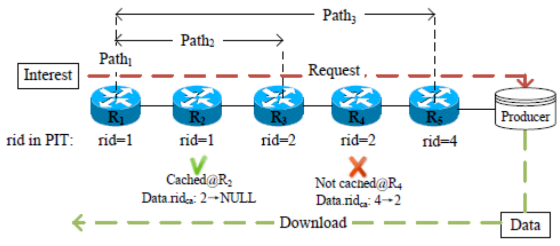


Fig. 5. The "change points" (R_1, R_2, R_4) segment the path into three sub-paths ($Path_1$ only contains R_1). When Data goes towards the consumer, if the selected router fails to store the Data due to the information inconsistency problem, the Data's target CS location can be changed to the alternative node indicated by *rid* in PIT. E.g., if R_4 cannot store the Data, it modifies rid_{ca} in Data to 2, for R_2 is the selected router on $Path_2$.

D. *Data Delivery*

Sent by the outside producer, the Data enters the ISP via the gateway node. The gateway router looks up PIT and writes the ID of the candidate caching router into Data's rid_{ca} field (There may be multiple candidates when the Data is requested by multiple consumers. If so, gateway router is supposed to make multiple Data copies.). If there is not a valid candidate router ID, rid_{ca} is written as *NULL*. The gateway also initializes Data's *hops* field as 1 before forwarding it.

Shown in Algorithm 2, when receiving $Data_k$, R_i increases the Data's *hops* value (Line 2). The router can use this value update the local distance profile periodically. Assume the Data has not been cached on the traveled path and R_i is the candidate caching router (Line 3-9). Since there may be information inconsistency problem as explained above, the router has to *double check* if the caching condition for $Data_k$ (i.e., it has enough storage or the caching benefit is larger than the cacheable threshold) is still met (Line 4). If the condition holds, then R_i adds $Data_k$ in CS and sets the cached flag ($Data_k.if\_cached$) in order to prevent it from being cached again afterwards (Line 5-6). Then the router sets $Data_k.hops$ to 1 because it is the nearest content source w.r.t. downstream routers now (Line 7). If the caching condition is no longer met, there is another chance to cache the Data on the left path: R_i looks up PIT and retrieves the ID of the selected caching router on the following sub-path, as shown in Fig. 5. In this way, *Data*$_k$.*rid*$_{ca}$ is modified to the new router ID (Line 9) so that the Data is likely to be cached there. Finally, R_i forwards the Data to the next node (Line 10).

```
Algorithm 2  Receive_Data
1   function RECV_DATA($R_i$, $Data_k$)
2       $Data_k.hops \leftarrow Data_k.hops+1$
3       if $Data_k.cached$=FALSE AND $Data_k.rid_{ca}=i$ then
4           if $R_i.CS$ has enough storage to cache $Data_k$ OR There exists $Data_q \in R_i.CS$ satisfying $b_{q,i} < b_{k,i}$ then
5               CS_ADD_ENTRY($R_i$, $Data_k$)
6               $Data_k.cached \leftarrow$ TRUE
7               $Data_k.hops \leftarrow 1$
8           else
9               $Data_k.rid_{ca} \leftarrow$ PIT_RETRIEVE_RID($R_i$, $Data_k$)
10      FORWARD($R_i$, $Data_k$)
```

E. The Dilution Methods

With the explosion of user-generated contents in social and video networks, the lifetime of popular contents is getting much shorter. Therefore, a large number of inactive Data exist in the CS not only during network idle hours, but also in the transition periods as the new popular contents emerge, which will lower the caching performance. We introduce two dilution methods in order to age out the stale statistics for caching benefit. As a result, the benefit for continuously beneficial Data keeps dynamic balance while that for inactive/dead Data drastically decreases.

1) The linear dilution

A straightforward way is to dilute benefit linearly. That is, for each item recorded in the router's priority queue BQ, its popularity decreases at the constant rate. In R_i, the caching benefit decreases every Δ_i^{des} time, shown in (4), where BQ_i represents the priority queue of R_i and bq_k is an entry in it.

$$\begin{cases} POP_{k,i} = POP_{k,i} - 1 \\ b_{k,i} = b_{k,i} \cdot \dfrac{POP_{k,i}}{POP_{k,i}+1} \end{cases}, \forall bq_k \in BQ_i \qquad (4)$$

If a cached Data's popularity reaches 0, it will be evicted from CS. In most cases, it is replaced by an incoming Data when the caching benefit drops out of the cacheable zone in the priority queue. The linear dilution is lightweight. However, it synchronously dilutes the effects of all Interests regardless of the arrival order of different Interests, which is unfair to the newly arriving ones.

2) The window-based dilution

To guarantee the fairness, we further propose the window-based dilution. That is, each item in the priority queue is assigned with a window which is implemented by the round-robin queue. The round-robin queue consists of a fixed number of slots. The length of the queue represents the window width. There is an active slot recording the real-time popularity for the content. The active slot slides at the constant rate.

For R_i, there are W_i slots in each content's round-robin queue. Current Interest popularity is recorded in the active slot whose duration is $(1/v_i)$, where v_i is the sliding rate for the active slot. Let $pop_{k,i}^r$ be the counter in the r-th slot, the popularity for *Interest*$_k$ is the sum of counters within the window, $POP_{k,i} = \sum_{r=0}^{W_i-1} pop_{k,i}^r$. Shown in (5), as the active slot slides to the $[(r+1) \bmod W_i]$-th slot from the r-th slot, the new active slot clears its old counter and the dilution is triggered.

$$\begin{cases} b_{k,i} = b_{k,i} \cdot \dfrac{POP_{k,i} - pop_{k,i}^{(r+1) \bmod W_i}}{POP_{k,i}} \\ POP_{k,i} = POP_{k,i} - pop_{k,i}^{(r+1) \bmod W_i} \\ pop_{k,i}^{(r+1) \bmod W_i} = 0 \end{cases}, \forall bq_k \in BQ_i \qquad (5)$$

Obviously, the average effect time of an Interest is $(1/v_i)(W_i - 1)$, because the lifetime for every Interest lasts for $(W_i - 1)$ slots in the round-robin queue. Thus, the effect time is adjusted by the value of v_i (in general, the length of the queue remains unchanged.). Compared with the linear dilution, the window-based dilution has more reliable performance, because the influence of Interests elapses strictly following the order of their arrivals.

V. Performance Evaluation

We simulate the proposed caching schemes and evaluate the performance under real-life topologies.

A. Simulation Setup

The caching schemes are simulated on five real-life ISP topologies in different tiers and scales as summarized in Table II. Without loss of generality, the end users under an access router are merged to one single node which acts as the consumer. For each consumer, we generate the synthetic Interests trace as the input traffic which is described as follows. The popularity for Interests is governed by Zipf distribution [9]. That is, the probability of the i-th ranked popular Interest arrival is $f(i) = i^{-\alpha} / \sum_{k=1}^{|\mathcal{O}|} k^{-\alpha}$ $(0.5 \le \alpha \le 1.3)$, where α is the parameter (skewness factor) of Zipf distribution. The total number of Interests expressed by the consumer is uniformly distributed. The simulator uses *slot* as the time granularity. In each slot, for a consumer, it sends out one Interest to the connected access router; for a router, it processes all packets received in the last slot. All the Interests are assured to be successfully served, either by the on-path CS or by the outside producer. To enhance the effect of inside cache, the distance between the producer and the gateway is assumed to be much longer than that between inside consumers and the gateway. A simulation terminates when all the desired Data arrive at the requesting consumers.

To compute the energy consumed within an ISP, we use the parameters listed in Table III [10]-[13]. Note that currently there is not a prototype of the CCN router, therefore we refer to the energy density of the typical IP router. The energy consumption is derived according to (8-10) in Appendix. We implement DEE scheme with the window-based dilution that is called DEE-WIN (The DEE scheme implemented with linear dilution is also tested and shows worse performance than DEE-WIN in all the metrics below.). In addition, we compare it against ProbCache scheme [14], ProbCache+ scheme [15] and Caching Everything Everywhere (CEE) scheme [1]. Specifically, ProbCache is a probabilistic CCN caching scheme in which each router on the Data downloading path has the probability of 1/(hop count) to cache the passing Data, while ProbCache+ improves ProbCache in terms of fairness. In CEE which is the default caching scheme suggested in the original CCN design [1], the router caches every received Data and manages local CS with LRU replacement policy.

TABLE II: THE NETWORK TOPOLOGY

Network	AS number	Gateways	Nodes
Wisconsin University	AS59	2	41
Tsinghua University	-	1	51
UUNet Alternet	AS701	3	75
AT&T ISP	-	3	154
UNINETT	AS224	2	208

TABLE III: THE ENERGY AND POWER DENSITY

Parameter	Descriptions	Energy/Power Density
P_r	Energy density of the router	2.63×10^{-8}J/bit
P_l	Energy density of the link	1.04×10^{-7} J/bit
P_{ca}	Power density of the cache medium	3.00×10^{-9}W/bit

Since CEE is the proposed caching scheme in the original CCN design, we use it as the baseline. By default, the duration of a single slot is $\tau'=1$ s. The skewness parameter α in Zipf distribution is set to 0.9. The default cache size for every router is set to 0.5% of the content category size. The dilution rate and popularity aggregation interval are initialized as 10% of the Interest arrival rate and 100 slots respectively, and can be adjusted according to the real-time performance. For simplicity, the caching decisions are made within the same slot for popularity aggregation. We test a variety of simulation settings and report the results.

B. Energy Consumption

We first compute the percentage in terms of energy consumed by other schemes w.r.t. CEE. The simulation is conducted with the AT&T ISP. The total Interest amount at every access router is uniformly distributed in $U(200{,}000, 1{,}000{,}000)$.

In Fig. 6, we consider 100,000 different contents and plot the energy consumption as the relative cache size increases from 0.1% to 3.0%. It is reasonable that the performance for all schemes gets steadily improved as the cache size increases. DEE-WIN outperforms the rest and saves around 30% energy against CEE. Since the price of storage medium drops constantly, the router is supposed to afford the larger CS in the future, which is more favorable to DEE.

To test the scalability, we fix the cache size at 0.5% of the content category size and increase the content popularity. As shown in Fig. 7, the curves for all the schemes are smooth. We can infer these schemes have stable performance when dealing with massive contents.

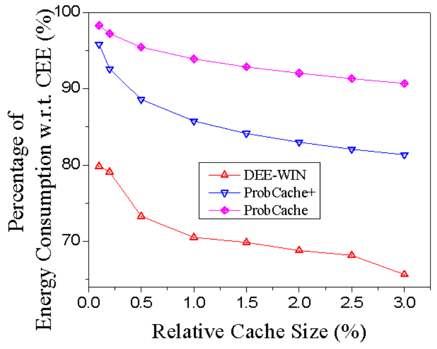


Fig. 6. Energy consumption with cache size (AT&T)

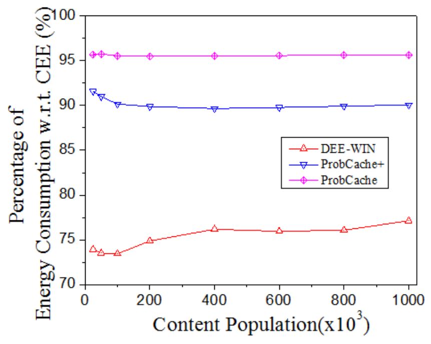


Fig. 7. Energy consumption with content population (AT&T)

C. Cache Hits

With the same topology and configuration settings to the simulation for energy consumption, we then compute the number of cache hits achieved by the schemes w.r.t. CEE in Fig. 8 and Fig. 9.

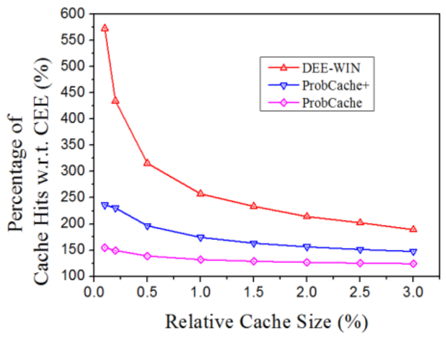


Fig. 8. Cache hits with cache size (AT&T)

Fig. 8 shows the percentage of cache hits as the relative cache size increases. With the smaller cache size, DEE-WIN can achieve five times more cache hits than the original CEE. As the cache size goes larger, the cache performance reaches the stable state. On the other hand, we change the content population and plot the percentage of cache hits in Fig. 9. For different content category sizes, DEE-WIN always derives more cache hits than the two compared schemes.

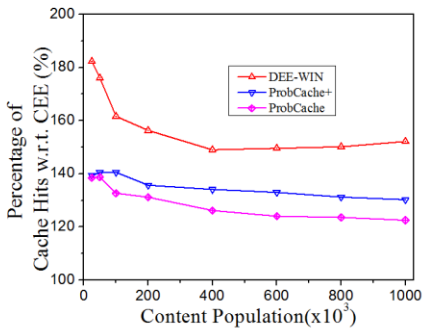


Fig. 9. Cache hits with content population (AT&T)

D. *Average Data Download Delay*

We use the campus network of Tsinghua University to test the average Data download delay under the four caching schemes. The Data download delay is defined in the number of hops that a single Data has traversed from the content source (the CS or producer) to the requesting consumer.

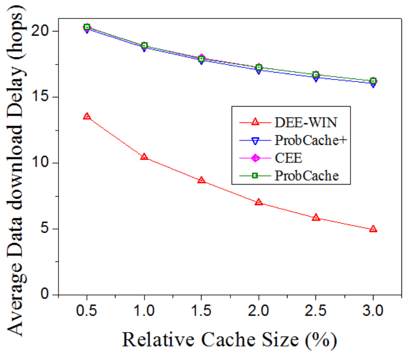


Fig. 10. Average data download delay with cache size (Campus)

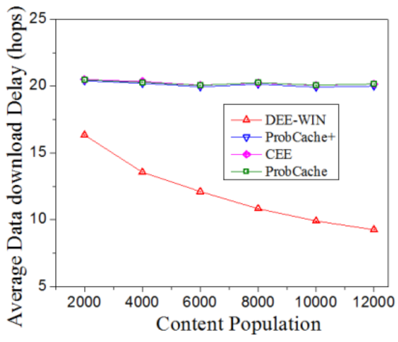


Fig. 11. Average data download delay with content population (Campus)

In Fig. 10, we consider 2,000 different contents. The total Interests amount at every access router is uniformly distributed in U(4,000, 6,000). We increase the relative cache size of every router and derive the average Data download delay curves. It is not surprising that all curves get dropped when more cache storage is given. Since more and more Interests can be served by the in-network cache rather than the outside producer, the Data download delay is shrunk.

We then change the content category size and plot the result in Fig. 11. Note that the total Interest amount in the access router is proportional to the real-time content population. With the fixed relative cache size, DEE-WIN steadily decreases the average download delay while others keep the constant values.

E. *Average Link Load*

In Fig. 12 and Fig. 13, we use the same campus network and configuration settings above to evaluate the average link load. The link load is described as the average number of packets (Interest and Data) per link. Given more Interests are served by CS as the cache size increases, the link load drops in Fig. 12. In Fig. 13, because the Interests amount is proportional to the content category size, there are more loads on links when we use the larger content category. DEE-WIN still generates the least additional load.

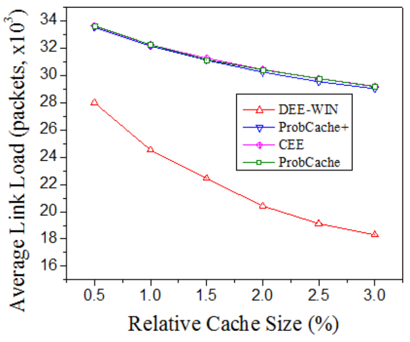


Fig. 12. Average link load with cache size (Campus)

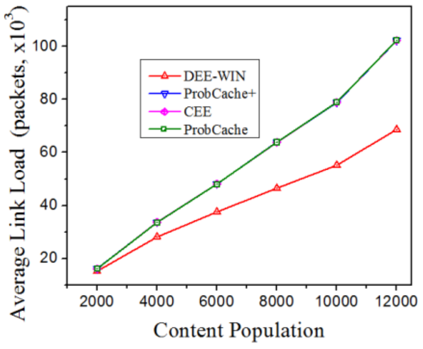


Fig. 13. Average link load with content population (Campus)

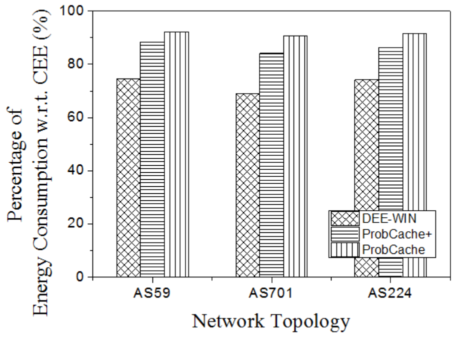


Fig. 14. Energy consumption with topology

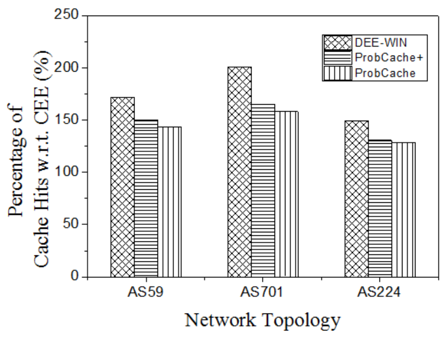


Fig. 15. Cache hits with topology

F. Topology

In addition to the topologies of AT&T and Tsinghua University, we also explore the other three topologies listed in Table II. We plot the results for energy consumption and cache hits in Fig. 14 and Fig. 15. When deployed in these real-life topologies, DEE-WIN outperforms the others in both metrics, which indicates it is insensitive to the topology changes.

VI. Related Work

The cache networks, allocation schemes and replacement policies in IP-based applications have been extensively studied and a large body of works has led to great success. Differing from IP-based caching running on the application layer which is readily to manage and control, CCN caching deals with the contents on the network layer. It needs to consider the name-based routing and make on-path caching decision on-the-fly. Therefore, the existing caching strategies are hard to be transplanted in CCN, which makes caching in CCN a brand new research area.

In recent studies, Ref. [16] colors some nodes in the campus network as CCN routers and proposes a caching scheme aiming at minimizing the network-wide delay, which runs in exponential time in the worst case. Ref. [17] converts the collaborative caching to linear programming problems and solves them through dynamic programming method, but it is constrained by the huge computing complexity. Ref. [18] studies caching in CCN from the economical view. It proposes OC^{3}N caching scheme for the ISP which does not require the content popularity information. Instead, it formulates the costs in terms of caching and retrieval. D.Rossi *et al*. propose a novel simulation study for the caching performance in CCN [19] and present the architecture of content router recently [20]. Ref. [21] also proposes a window-based caching scheme WAVE that pushes hot contents to network edge, but it cannot eliminate cache redundancy. Note that most of these works are based on the assumption that the Data can be delivered through arbitrary paths with the help of a flexible FIB. Instead, we adopt the lightweight on-path caching in CCN. That is, routers would not modify FIB entries no matter whether the cached Data location changes in network, so the forwarding path for an Interest is fixed. Therefore, Data can only be cached along the download path. Besides, Interest hits the CS only on its way towards the producer, which reduces redundant traffic and avoids the huge overhead incurred by CS misses and FIB updates, as suggested in [1].

As for the optimization objective, almost all the related works focus on the increase of cache hit ratio, whereas little effort has dedicated to the energy-efficient caching. Ref. [5] formulates the optimization problem with energy consumption model and proposes an online caching scheme. But it needs to predict the exact time at which the incoming Interest arrives, which is hard to be realized in practice. Besides, it has not considered the implementation issues such as the CS information inconsistency.

Towards the energy-efficient caching strategy within the scope of an ISP in CCN, we reveal very few Interests can be aggregated in PIT in reality, which is valuable to the CCN-related topics but rarely noticed by literatures. Then we conclude the energy consumption largely relies on the average bit-level Data download delay. Regarding the popularity-hop product as the main caching benefit, we develop the heuristic scheme DEE. We also pay attention to the caching framework in this work.

VII. Conclusion

In this paper, we investigate the energy-efficient cache allocation and replacement problems in CCN within the scope of an ISP. With the observation that Interests aggregation in PIT is very weak, we safely ignore it and translate the problem of saving energy to reducing the average bit-level Data download hop count. We then propose a heuristic scheme called Distributed Energy-Efficient (DEE) caching that dynamically places Data across the delivery paths based on the predicted benefit in term of Interests popularity and Data download hop count. We further consider the impact of bias requesting, information inconsistency and Interests timeliness. Through the simulation under real-life topologies, the proposed framework has been proven to be effective, steady and scalable.

Appendix

As mentioned in Section III-B, we prove that the bit-level energy consumption only depends on the average bit-level hop count under the assumption of ignoring the Interests aggregation in PIT.

Let $\mathcal{R}$ be the set of CCN routers in an ISP, among which $\mathcal{A}$ is the set of access routers that directly receive Interests from consumers. Define $\mathcal{O}$ as the content category. The Interest and Data for content O_k are labeled as $Interest_k$ and $Data_k$ respectively. We divide the continuous time into a series of identical time intervals, each of which lasts for τ . t_m (m=0,1,2...) represents the m-th time interval. During a single time interval, the Interests arrival rate remains constant and the location for each individual cached replica keeps unchanged. Thus, the cache network incrementally evolves. This analysis model can approach to the real case as long as the interval duration is small enough. Define $\lambda_{k,i}(t_m)$ as the arrival rate for *Interest*$_k$ in router R_i within t_m, and s_k as the size of *Data*$_k$. The binary indicator $x_{k,i}(t_m)=1$ means *Data*$_k$ is cached in R_i in t_m, otherwise $x_{k,i}(t_m)=0$. In t_m, the total amount of requested bits is

$$S(t_m)=\sum_{R_i\in\mathcal{A}}\sum_{O_k\in\mathcal{O}}\lambda_{k,i}(t_m)\cdot\tau\cdot s_k \tag{6}$$

Originated from R_i, $h_{k,i}(t_m)$ is the hop count that *Interest*$_k$ has traversed till it is served, either by the producer or by the on-path CS. Note that $h_{k,i}(t_m)$ is constant during interval t_m because the cache layout keeps unchanged in each single time interval. Assume we omit the Interests aggregation in PIT and consider the requested Data size equals to that of the delivered Data. Then, we can derive the average bit-level hop count when downloading Data

$$h_{avg}(t_m)=\frac{\sum_{R_i\in\mathcal{A}}\sum_{O_k\in\mathcal{O}}\lambda_{k,i}(t_m)\cdot\tau\cdot s_k\cdot h_{k,i}(t_m)}{S(t_m)} \tag{7}$$

The energy consumption in t_m is contributed by two main components: the transmission energy $E_{tr}(t_m)$ and the caching energy $E_{ca}(t_m)$. The background energy consumption is omitted because it keeps unchanged no matter how much Data is transported and what caching scheme is utilized. To reduce energy consumption, we only consider the changeable components. Define P_l (in J/bit) and P_r (in J/bit) as the transmission energy density of a link and a router respectively. According to [5], the transmission energy is roughly proportional to the transported Data size for both links and routers. So the consumed Data transmission energy is

$$\begin{aligned}E_{tr}(t_m)&=\sum_{R_i\in\mathcal{A}}\sum_{O_k\in\mathcal{O}}\lambda_{k,i}(t_m)\cdot\tau\cdot s_k\cdot[P_l\cdot h_{k,i}(t_m)+P_r\cdot(h_{k,i}(t_m)+1)]\\&=(P_r+P_l)\sum_{R_i\in\mathcal{A}}\sum_{O_k\in\mathcal{O}}\lambda_{k,i}(t_m)\cdot\tau\cdot s_k\cdot h_{k,i}(t_m)+P_r\cdot\sum_{R_i\in\mathcal{A}}\sum_{O_k\in\mathcal{O}}\lambda_{k,i}(t_m)\cdot\tau\cdot s_k\\&=(P_r+P_l)\cdot S(t_m)\cdot h_{avg}(t_m)+P_r\cdot S(t_m)\end{aligned} \tag{8}$$

On the other hand, the energy consumed for caching Data in CS is

$$E_{ca}(t_m)=P_{ca}\cdot S(t_m)\cdot\gamma(t_m)\cdot\tau \tag{9}$$

where P_{ca} is the power density (W/bit) of the CS medium in the CCN router and $\gamma(t_m)$ is the percentage of CS-served Data size over $S(t_m)$. Thus, the total energy consumption is

$$\begin{aligned}E(t_m)&=E_{tr}(t_m)+E_{ca}(t_m)\\&=[(P_r+P_l)\cdot h_{avg}(t_m)+P_r+P_{ca}\cdot\tau\cdot\gamma(t_m)]\cdot S(t_m)\end{aligned} \tag{10}$$

Finally, the bit-level energy consumption is derived by dividing $E(t_m)$ by $S(t_m)$

$$\begin{aligned}E_{bit}(t_m)&=\frac{E(t_m)}{S(t_m)}\\&=(P_r+P_l)\cdot h_{avg}(t_m)+P_r+P_{ca}\cdot\tau\cdot\gamma(t_m)\end{aligned} \tag{11}$$

Compared with the content category size, the router's CS budget is very limited so that we can assume the CS is fully occupied all the time. Thus, $\gamma(t_m)$ is completely depending on the total requested Data size $S(t_m)$, which is a constant value during each interval. Then the average bit-level hop count $h_{avg}(t_m)$ becomes the only variable in (11).

REFERENCES

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th Annu. International Conf. Emerging Networking Experiments and Technologies (CoNEXT '09)*, New York, 2009, pp. 1-12.

[2] P. Jokela, A. Zahemszky, C. E. Rothenberg, S. Arianfar, and P. Nikander, "LIPSIN: Line speed publish/subscribe inter-networking," in *Proc. ACM SIGCOMM Conf. Data Communication*, New York, 2009, pp. 195-206.

[3] T. Koponen, *et al*., "A data-oriented (and beyond) network architecture," in *Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Communications*, New York, 2007, pp. 181-192.

[4] W. K. Chai, *et al*., "Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 112-120, March 2011.

[5] J. Llorca, *et al*., "Dynamic in-network caching for energy efficient content delivery," in *Proc. IEEE INFOCOM*, Turin, 2013, pp. 245-249.

[6] N. Kamiyama, *et al*., "Optimizing cache location and route on cdn using model predictive control," in *Proc. 27th International Teletraffic Congress*, Ghent, 2015, pp. 37-45.

[7] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, pp. 1305–1314, Sep. 2002.

[8] M. Dehghan, B. Jiang, A. Dabirmoghaddam, and D. Towsley, "On the analysis of caches with pending interest tables," in *Proc. 2nd International Conf. Information-Centric Networking*, New York, 2015, pp. 69-78.

[9] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE Trans. on Networking*, vol. 16, no. 6, pp. 1447–1460, 2008.

[10] W. Van Heddeghem and F. Idzikowski, "Equipment power consumption in optical multilayer networks-source data," Technical Report, IBCN-12-001-01.

[11] M. W. M. Nazri, *et al*., "Energy efficient segmentation-link strategies for transparent IP over WDM core networks," *Journal of Communications*, vol. 9, no. 1, pp. 48-55, 2014.

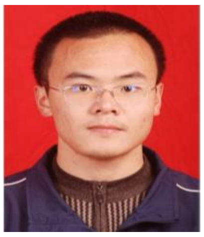
[12] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *Proc. 36th Annual International Symposium on Computer Architecture*, New York, 2009, pp. 2-13.

[13] *Cisco ONS 15501 Erbium Doped Fiber Amplifier Data Sheet*, Cisco Systems, Inc., San Jose, CA, 2003.

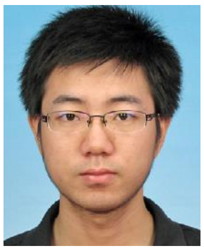
[14] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. 2nd International Conf. Information-Centric Networking*, New York, 2012, pp. 55-60.
[15] I. Psaras, W. K. Chai, and G. Pavlou, "In-Network cache management and resource allocation for information-centric networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 25, no. 11, pp. 2920-2931, Nov. 2014.
[16] M. Shen, B. Chen, X. Zhu, and Y. Zhao, "Towards optimal cache decision for campus networks with content-centric network routers," in *Proc. IEEE Symposium on Computers and Communication*, Messina, 2016, pp. 810-815.
[17] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Design and evaluation of the optimal cache allocation for content-centric networking," *IEEE Trans. on Computers*, vol. 65, no. 1, pp. 95-107, Jan. 2016.
[18] A. Gharaibeh, A. Khreishah, and I. Khalil, "An O(1)-competitive online caching algorithm for content centric networking," in *Proc. IEEE INFOCOM*, San Francisco, 2016, pp. 1-9.
[19] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Technical Report, 2011.
[20] G. Rossini, D. Rossi, M. Garetto, and E. Leonardi, "Multi-Terabyte and multi-Gbps information centric routers," in *Proc. IEEE INFOCOM*, Toronto, 2014, pp. 181-189.
[21] K. Cho, *et al*., "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *Proc. IEEE INFOCOM*, Orlando, 2012, pp. 316-321.

Hao Wu received the B.S. degree in information engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2011. He is currently working toward the Ph.D. degree in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include caching algorithm in CCN and network measurement.

Bin Liu was born in 1964. He is now a Full Professor in the Department of Computer Science and Technology, Tsinghua University. His current research areas include high performance switches/routers, network processors, high speed security and greening the Internet. Bin Liu has received numerous awards from China including the Distinguished Young Scholar of China and won the inaugural Applied Network Research Prize sponsored by ISOC and IRTF in 2011.



Yang Li is currently pursuing the Ph.D. degree in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research areas include network measurement and software-defined networking.



Huichen Dai received the B.S. degree from Xidian University, Xi'an, China, in 2010. He is currently pursuing the Ph.D. degree in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests mainly lie in router architecture, content-centric network, name lookup, and network processor architecture.

Yi Wang is a researcher in Huawei Lab. He received the Ph.D. degree in Computer Science and Technology from Tsinghua University in July, 2013. His research interests include router architecture design and implementation, greening the Internet, fast packet forwarding and Information-Centric Networking.

Yaxuan Wang is currently pursuing the B.E degree in the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China. Her current research focus is on Named Data Network (NDN).