Age-based Cooperative Caching in Information-Centric Networking

Zhongxing Ming*, Mingwei Xu*, Dan Wang†
* Dept. of Comp. Sci. & Tech., Tsinghua Univ., Tsinghua National Laboratory for Information Science and Technology
† Dept. Computing, The Hong Kong Polytechnic University

***Abstract*—Information-Centric Networking (ICN) provides substantial flexibility for users. One of the most important features of ICN is the universal in-network caching. The characteristics of ICN make it substantially different from traditional caching systems. In this paper we propose an age-based cooperative caching scheme in response to the special characteristics of ICN. We leverage the coupling between routing and caching in ICN to develop a light-weight collaboration mechanism that adaptively pushes popular contents to the network edge. We evaluate our approach using real traces and realistic network topology. Results show that our approach can significantly reduce network delay and traffic, and outperforms existing schemes.**

I. Introduction

he information-centric networking (ICN) has attracted great attention in recent years [1, 2]. One of the most important features of ICN designs is the *universal in-network caching*. In-network caching has the potential to improve network efficiency and content distribution performance [3]. Although Caching has been extensively studied in the past decades (e.g., [4, 5]), a number of characteristics make ICN different from traditional caching systems. First, caching becomes an intrinsic property of routers. Content caching and packet routing are performed at the same network layer, which makes caching and routing highly coupled. Second, the universal in-network caches of ICN are arranged as an arbitrary topology graph instead of a hierarchical system. Third, different application can use the same cache space, which makes in-network caching fundamentally different from traditional caching systems such as web caching or Content Dilivery Network (CDN) [6].

In response to the above characteristics, many researchers devote efforts to the study of in-network caching, e.g., [7–9]. The above work either requires extensive calculation that constrains the adoption in routers, or fail to adapt to the on-path coupling effect of caching and routing.

In this paper, we propose an lightweight **A**ge-**B**ased **C**ooperative scheme (ABC) to the design of efficient in-network caching system by fully utilizing the on-path coupling effect of caching and routing in ICN. By lightweight we mean that the scheme do not need extensive computation or message exchange between routers.

ABC adaptively pushes popular content objects to the network edge by using dynamically changing ages that control the lifetime of a content replica in the router. The age of the content object is decided along the routing path and no signalling message between routers or extensive calculation is needed. This scheme spreads popular contents to the network edge and at the same time eliminate unnecessary content replication in the middle of the network.

We conduct extensive evaluation using the topology of China Education and Research Network 2 (CERNET2), the world's largest native IPv6 backbone network. We leverage real traces captured from a commercial website with data sent by 72,517 users from 20 cities of China. We compare ABC to classical caching strategy and newly proposed in-network caching strategy. The results show that our approach achieves significant gains to improve network performance and outperforms the existing schemes. Preliminary work of the paper was presented in [10]

The contribution of this paper is as follows.

- We analyze existing caching schemes to be used in ICN.
- We propose an age-based cooperative in-network caching scheme and design two lightweight algorithms (the second one is a part of the first one) to implement the scheme.
- We evaluate the proposed approach using real trace and realistic network topology. Results show that our approach achieves significant performance gains and outperforms existing caching strategies.

The rest of this paper is organized as follows. Section II describes the the problems of existing caching mechanisms. Section III presents the age-based cooperative caching scheme and develops two algorithms to realize it. Section IV evaluates the proposed approach. Section V discusses the consistency issues. Section V discusses the related work. Section VII concludes the paper.

II. Problem statement

Information-centric networking is inherently organized as a multi-level caching system, with the caches physically scattered across the network and the user requests generated in geographically dispersed nodes. When an object is requested from an intermediate node then if the cache has a copy, the response is returned locally. This leads to the design problems of efficient caching strategies, such as optimal dimensioning of caches and intelligent content placement.

Ideally, it is desirable to distribute contents hierarchically in the network, with most popular contents at the edge, less popular contents near the edge and least popular contents even further. In this way, the aggregated cache hit rate of the network will be maximized, thus the network delay and publisher load will be reduced. Take delay as a criterion, the closer a node with a content copy is to a user the less time the user requires to get the content. Caching strategies can exploit storage capacity to absorb latency by replicating the most popular contents.

The issue of network level cache has been analyzed in the context of web caching and content distribution networks. Classical Web caching algorithms, such as Least Recently Used (LRU), enable popular contents spread around the network. However, they need a lot of book keeping on each memory access to be implemented exactly. After each access the time of the least recently used page must be tracked. This is highly inefficient and requires sophisticated hardware support, which constrains the adoption in routers.

Cooperative mechanisms appear to be more efficient in ICN scenarios. The benefits of cooperative caching have been extensively studied, such as [4, 11–13]. However, these work either focuses on special-purpose applications which put additional constraints on the design (e.g., P2P system), or requires the system to be constructed as a particular type of topology, e.g., a multicast tree. Extensive calculation is often required, which limits their usage in global environment.

Age-based caching presents another alternative. The *age*-based caching is widely explored in web applications (e.g., distributed file systems [14], web caching [5], and DNS [15]). Nevertheless, they have drawbacks in terms of following aspects: (I) Many existing age-based cache schemes adopt a fixed content age which lacks service differentiation. (II) Even different ages are assigned to contents with various priorities, most schemes require manual configurations, which does not fit for the Internet-scale content caching. (III) They usually have no collaboration, which is inefficient to make full use of the aggregated network storage capacity.

Motivated by the above issues, we provide our work on an age-based cooperative caching scheme for Information Centric Network. Section III presents the details.

III. Age-based cooperative caching

The proposed caching scheme uses an age-based cache replacement policy to specify what content chunks to be cached in routers. The principle of the scheme is as follows.

- Each piece of content has an age. The lifetime of a replica in a router is decided by its age .
- The replica obtains its age when it is added into the cache and is removed from the cache when the age expires.
- Routers implicitly collaborate by modifying the age.

Content location and popularity are used to decide the value of the age, which obeys the following rules:

- The further a replica is to the server the larger age it has.
- The more popular a replica is the larger age it has.

This scheme increases the aggregated in-network cache hit probability, thus reduces network delay and publisher load simultaneously.

Figure 1 presents a simple case that intuitively shows how the scheme works. In Figure 1, one client and one server are connected by two routers (R1 and R2). The server serves two contents, one is popular and the other is less popular. The

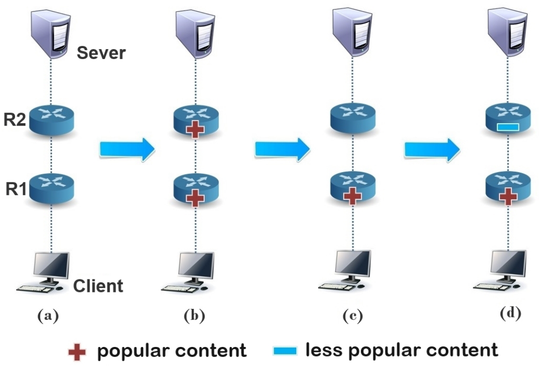


Fig. 1. A case study

client successively requests the two contents according to their popularity. For the sake of simplicity, we assume both routers have the storage capacity of one replica. We start at the initial phase that two routers have no content cached (Figure 1(a)). At a certain moment, the popular content is requested by the client and both routers have the content cached, which leads the system to Figure 1(b). According to our scheme, for the same content, R1 assigns it a longer age than R2. After some time, the popular content in R2 expires but can still be cached by R1 (Figure 1(c)). If at this time, the less popular content is required then R2 will naturally cache the replica. Now, the system migrates to Figure 1(d). This state is optimal, as aggregated network delay and publisher load are minimized. Because we provide the popular content with longer age than the less popular one, the popular content will have less opportunity to be replaced, which tend to maintain the system in this state.

In the next sections, we provide two algorithms to realize the scheme.

A. Age-based cooperation

This algorithm aims at collaborating ICN nodes under content's age. To avoid extensive calculation, we apply an age-based policy. A replica is replaced by other object(s) if both of the following conditions are satisfied: 1) The age of the content has expired and, 2) the cache memory of the node has been full.

We design a lightweight cooperative algorithm by dynamically adjusting content's age at different network nodes. For each content object, the age is determined by the following factors: 1) Distance to the server. The further a content is from a server, the longer age it has. 2) Popularity of the content. The more popular the content is, the longer age it has. Factor 1 enables content chunks to be pushed to the network edge while at the same time releases unnecessary object storage at intermediate nodes (e.g., Figure 1(c)). Factor 2 gives popular contents higher priority to be cached by routers. To realize the above logic, each content packet carries an age field and routers decide a content's age by exploring and adjusting the age field. Many studies have investigated the reliable and efficient estimation of content popularity, such as [16]. In this paper, we assume such schemes are given.

In a word, the algorithm works as follows: when a content object reaches a router, the router first checks its cache memory to see if there is enough space for the object. If yes, the object is cached. Otherwise, the router finds whether there is any expired content. If there is, the router removes it and adds the new object into its cache. Otherwise, the object will not be cached. Upon caching an object, a router performs the *get_new_age()* algorithm to generate the content's age and fill the age into the age field to pass the information to the next router. A *base_age* is set by the first router along the path and a *max_age* value is used to keep the age from infinitely increasing. Algorithm 1 shows the above logic.

The age decision rules stated in Section II are realized by the *get_new_age()* function, which is described by the next algorithm.

Algorithm 1 Age-based cooperation ($content$)

```
1: delete_expired_content();
2: $age$ = get_new_age($content$);
3: if have_enough_space($content$) then
4:     add_to_cache($content, age$);
5: end if
6: set_new_age($content, age$);
7: forward_data($content$);
```

B. Age decision

This algorithm implements the *get_new_age()* function used in algorithm 1, with the objective of spreading popular contents to the network edge while at the same time relieving intermediate nodes from caching unnecessary replicas.

We assume N contents $\{C\}_{i=1}^{N}$ with associated popularity $\{P\}_{i=1}^{N}$, $P_i \in R^{+}, \forall i = 1, ..., N$. The popularity weights, used in the *get_new_age()* function, can be formulated as $weight_i = \frac{P_i}{\sum_{j=1,...,N} P_j}$. The age decision process, detailed in algorithm 2, can guarantee popular contents be held longer throughout the network. We assume routers can use topology knowledge to decide which one is the first node on the path.

Algorithm 2 get_new_age($content$)

```
1: $weight$ = get_weight($content$);
2: if the router is the first node along the path then
3:     $age = base\_age * weight$;
4: else
5:     $age$ = extract_upstream_age($content$);
6:     $age$ = $age * 2$;
7:     if $age > max\_age$ then
8:         $age = max\_age$;
9:     end if
10: end if
11: return $age$;
```

Our algorithms ensure that a router keeps data longer when it serves as a leaf router in one multicast group, and holds data shorter if it serves as an intermediate router in other groups. To a large extent, this avoids the problem that a replica is cached in an intermediate router but is never used (because requests for the replica are satisfied by the downstream router(s)). In such a way, the buffer memory is efficiently utilized and network performance is improved.

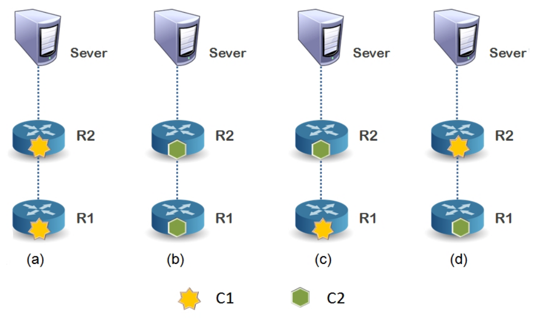


Fig. 2. A simple scenario

The proposed algorithms focus on the cache management of a single content. In practice, contents are often split into a sequence of chunks. However, it is straightforward that our scheme can readily extend to the chunk-level caching, with each chunk having a particular age.

C. Theoretical Analysis

In this section we conduct a simple case to theoretically show the benefits of the age-based cooperative scheme. We want to study ABC's impact on network delay and publisher load. As a comparison, we compare ABC to LRU in the same scenario.

In our case one server (S) and two routers (R_1 and R_2) are deployed with the interrelations shown in Figure 2. The server serves two contents C_1 and C_2 with unit content size. We assume all contents have the same popularity. Each router has a cache size of 1. During the lifetime of the case, R_1 repeatedly receives requests at the speed of α requests per second, with each request randomly asking for C_1 or C_2. The *base_age*of ABC is set to be β, with $\beta >> \alpha$.

It is not difficult to see that under LRU, R_1 will always cache the same content as R_2 (Figure 2(a)-(b)). For ABC, however, 4 types of situation may happen, which is shown in Figure 2(a)-(d). This is because R_1 caches content longer than R_2 for it is the downstream router in this scenario. For example, suppose the current status of the system is as what is shown in Figure 2(a). After some time, C_1 expires in R_2 but can still be kept longer by R_1. If the next request received by R_1 asks for C_2, the required content will go through the path of S - R_2 - R_1 and be cached by R_2. The situation then transforms to Figure 2(c).

We investigate how much delay the network requires to meet end users' demand. For simplicity, we assume the network delay is measured by hops. Denote by ts_i the average network delay of the i_{th} case as shown in Figure 2. Denote by tc_{i1} and tc_{i2} the time for users in case i to get C_1 and C_2 respectively. As stated above, requests for C_1 and C_2 are uniformly distributed, thus ts_i can be formulated as follows:

$$ts_i = \frac{tc_{i1} + tc_{i2}}{2} \quad (1)$$

Case	1	2	3	4
tc_{i1}	2	6	2	4
tc_{i2}	6	2	4	2
ts_i	4	4	3	3

TABLE I
CASE RESULTS

It is straightforward to calculate tc_{i1}, tc_{i2} and ts_i in our scenario with the results shown in Table I.

Denote by D_{ABC} and D_{LRU} the average network delay under ABC and LRU, respectively. If we assume the possible cases for ABC and LRU happen with equal probability then:

$$D_{ABC} = \sum_{i=1}^{4} ts_i \;/\; 4 = 7 \quad (2)$$

$$D_{LRU} = \sum_{i=1}^{2} ts_i \;/\; 2 = 8$$

This means that the expectation of ABC's performance in this scenario is 12.5% better than that of LRU. However, by carefully designing the age increment strategy, we can ideally limit the states of ABC within the scope of case (c) and case (d) and thus we have

$$D_{ABC} = \sum_{i=3}^{4} ts_i \;/\; 2 = 6 \quad (3)$$

This may be explained that ABC is potentially capable to reduce LRU's delay by 25%. Even under the worst situation that ABC only produces the case of 1 and 2, its performance will be the same as that of LRU, if not better.

From the perspective of publisher load, there are significant gains from ABC as it satisfies all requests at network-level which reduces the publisher load by 100% (Figure 2(c)-(d)).

While the case study is extremely simple and may not illustrate real scenarios in the complex network, it does intuitively show ABC's benefits. In the next section, we develop a trace-based evaluation under the realistic network topology to evaluate ABC's performance.

IV. Performance evaluation

In this section, we evaluate the performance of ABC using a trace-driven simulation under a real network topology. We first compare ABC with existing caching schemes, the we study the impact of parameter configuration on ABC's performance, including *cache size*, *base_age* and *max_age*.

A. Caching Performance

1) Methodology: We evaluate the performance of our scheme using the topology of CERNET2, which is the world's largest native IPv6 backbone network [17]. We make use of the data trace on various topics obtained from a commercial website. The topology is in conformity with the user distribution of our trace, in which we configure the simulated users to the exact cities where they lives. In such a way, we can make our simulation be conformance with real scenarios. We use NS2 [18] to build the simulator. Figure 3 shows the CERNET2 topology.

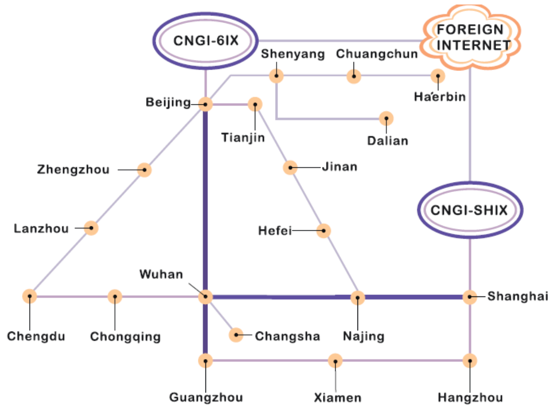


Fig. 3. CERNET2 topology

In Figure 3, each node represents the core router of its associated city (labeled with city name such as *Beijing*, *Tianjin*, etc). Each router is assigned with a cache capacity of 200 MB. For the sake of simplicity, we set the delay of the links to be 10 ms. A shortest path routing protocol is used in the network.

According to the real situation of the commercial website, content items are stored in a single server located in *Beijing*. Subscribers of 20 cities are connected to CERNET2 which are distributed according to Figure 4(b). Each user generates content requests according to a Poisson process of intensity 0.1 req/s. The motivation behind the Poisson assumption comes from the observation that Internet traffic is well modeled at session level by a Poisson process [19]. We evaluation the following targets.

Aggregated network delay: We first study the impact of ABC on the network by measuring the *aggregated network delay* calculated by equation 4. The *aggregated network delay* provides the insight of how ABC can improve network performance from the macroscopic viewpoint.

$$\frac{\sum_{i=1}^{UserCount} Delay\_of\_User_i}{UserCount} \quad (4)$$

End user delay: We then measure the impact of ABC on end users at each city. This factor points out how ABC affects users which reside in various locations of the network. Denote by SIM_TIME the duration that the simulation lasts. Denote by rc_i the count of requests received at the router of a particular city during the interval [i-1, i), i=1,...,N. Then for each city, the *end user delay* can be formulated by equation 5.

$$\frac{\sum_{i=1}^{SIM\_TIME} \sum_{j=1}^{rc_i} delay\_of\_request\_j/rc_i}{SIM\_TIME} \quad (5)$$

Publisher load: We finally study the traffic at the server side during the simulation, which is measured by the number of requests which reaches the server. Due to in-network caching, publisher load can be reduced, thus bandwidth and computing resources of the server can be saved. We want to find out how much this benefit can be achieved.

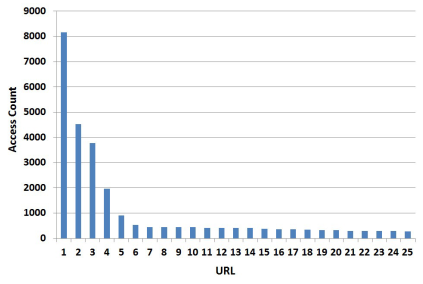
We implement an event-driven simulator to evaluate ABC's performance. The *base_age* is set to be 10 seconds and the *max_age* is set to be 60 seconds. The simulation lasts for 100 minutes. As a comparison, we compare the performance of ABC to that of LRU and WAVE [9]. LRU is a classical caching algorithm that has been proved to be very efficient in web caching systems. WAVE is an in-network caching strategy that enables an upstream router to suggest what content chunks to be cached at its next downstream router. We choose WAVE because it is designed for ICN and has the commonality of leveraging content popularity to make caching decision like ABC.

2) Characteristics of the trace: The trace was captured during a one-month period in 2013 using online statistical tools, which totaled 184,636 URL-requests sent by 72,517 users from 20 cities of China. We filter this trace and retrieve 13,426 distinct URLs and use these URLs as our simulation input. We range the URLs from 1 to 13,426 according to their access count in descending order, and find that the trace follows well the *Pareto Principle*. We make use of the popularity distribution of the trace to determine which content is required when one request occurs in our simulation. Figure 4(a) shows the access count of the top 25 URLs. In our simulation, each distinct URL represents an content item. The sizes of the content objects vary from 376 KB to 3.61 MB.

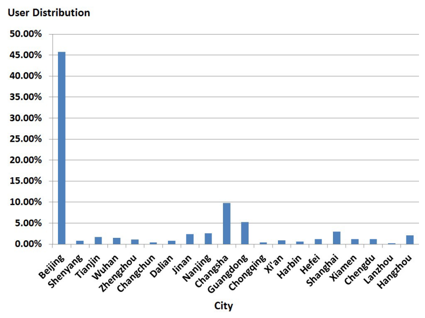
We further investigate the locations of all users in the trace. We distribute the number of users in the network based on this information. Figure 4(b) shows the subscriber location of the trace, in which subscribers in *Beijing* contribute a majority of the population. It's not surprising that the user distribution is not even as the website mainly provides regional services. To the best of our knowledge, this is the first work that focuses on localized websites in Information-Centric Network.

3) Evaluation results: Figure 5 shows how *aggregated network delay* varies as different caching schemes are used. The *aggregated network delay* of ABC is around 16 ms during the simulation, comparing with 21 ms of WAVE and 25 ms of LRU respectively. The figure shows the potential performance gains through the use of ABC.

Table II illustrates the effects of ABC, WAVE and LRU on *end user delay* in 8 cities of the network. The result is in accordance with *aggregated network delay*, such that *ABC* achieves the smallest delay, while LRU has the largest delay. There are, however, different performance gains in different cities. The reason for this phenomena is that the cities have different distances to the server. Another observation is that the further a router is from the server, the more latency reduction ABC provides than LRU and WAVE. For example, ABC reduces 11% more delay than LRU in Beijing, while it reduces 51% more delay than LRU in *Hangzhou*. On the other hand, the *aggregated network delay* of ABC remains almost constant for different cities, while the performances of LRU and WAVE decrease dramatically as distance grows. This is a major benefit of ABC as users that are far away from



(a) Access count per URL (top 25)



(b) User Location Distribution

Fig. 4. Trace information

the content server will not suffer from the increased distance, which provides uniform user experience for consumers at different locations of the network. This matches the scope of our expectation because popular contents are adaptively pushed to the network edge using our mechanism.

The previous two figures present the impact of caching on the network delay. In Figure 6, we show the amount of traffic that is received by the server under various caching schemes. When a subscriber requests one content, the request may be either satisfied by an intermediate router with the content cached or by the publisher server. As routers' caches become gradually filled after the simulation starts, the traffic to the server drops as more content object can be fetched from the in-network cache. We can see that ABC reduces more traffic than WAVE, while LRU has the worst performance, with traffic reduction of 43%, 26% and 18% for ABC, WAVE and LRU respectively.

The results in the figures show that there are significant gains through using ABC. The users will benefit from ABC as the total download latency will be lowered. On the other hand, content providers will be able to greatly reduce the traffic to save bandwidth and computing resources.

During the simulation, we find that adjusting the way that dynamically changes the parameters (e.g., *cache size*, *base_age*, *max_age*) within a rational scope does not change ABC's superiority. However, it does affect ABC with the impact that cannot be ignored. We will study the effects of cache size, *base_age* and *max_age* in the next Section.

B. Parameter Configuration

In this section, we evaluate how the parameter configuration of ABC will influence its performance. Three parameters are

City	Beijing	Shenyang	Wuhan	Changchun	Guangdong	Hefei	Chengdu	Hangzhou
Hops to the server	1	2	2	3	3	4	4	5
LRU	18.1 ms	20 ms	20.2 ms	25.6 ms	25.9ms	31.4 ms	32.1 ms	36.7 ms
WAVE	16.3 ms	18.5 ms	18.3 ms	20.2 ms	20.5 ms	22.7 ms s	22.3 ms	23.8 ms
ABC	16 ms	16.4 ms	16.6 ms	16.8 ms	16.9 ms	17.5 ms	17.2 ms	18.1 ms

TABLE II
END USER DELAY

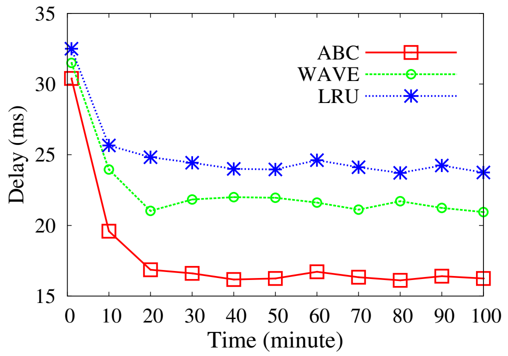


Fig. 5. Aggregated network delay

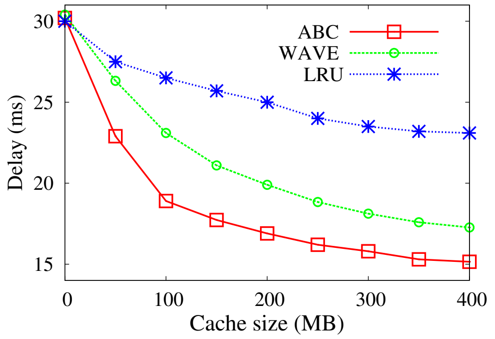


Fig. 7. Impact of cache size on different caching strategies

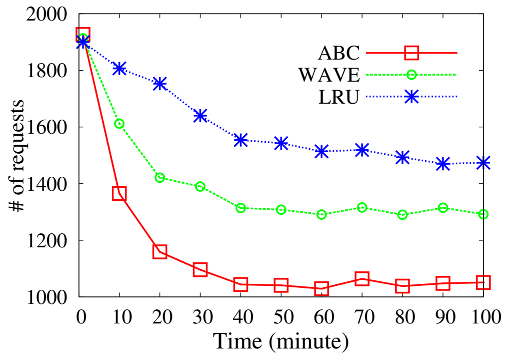


Fig. 6. Server load

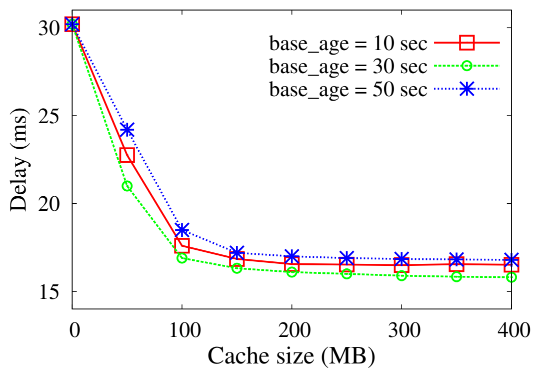
evaluated, they are: *cache size*, *base_age* and *max_age*. We use the aggregated network delay as the performance metric.

1) Cache size: In Figure 7, we show the influence of cache size on the delay reduction. Generally, cache size has a significant impact on the performances of all the algorithms. For example, the network delay of ABC decreases from 30.2 ms to 15.3 ms when cache size grows from 0 MB to 400 MB (reduced by 50%). We see that ABC has the best performance out of the three algorithms despite the change of cache size. It is worth noticing that the impact is more significant when the cache size is small. For instance, when cache size increases from 0 MB to 200 MB, the network delay of ABC is reduced by 13 ms. However, when cache size increases from 200 MB to 400 MB, the network delay of ABC is only reduced by 2 ms. This indicates that a small cache size can lead to a sound performance improvement while a large cache size is not necessary.

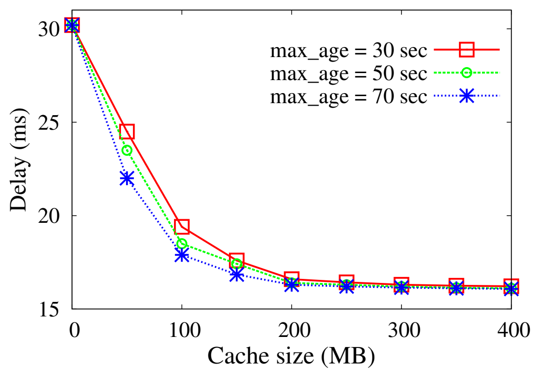
Figure 8 shows how cache size influences the performance of ABC under different *base_ages* and *max_ages*. We see that different combinations of *base_age* and *max_age* lead to different performance. Take *cache size*=50 MB and *max_age*=100 ms (Figure 8(a)), for example, when *base_age* changes from 10 ms to 50 ms, the aggregated delay increases from 21 ms to 24 ms. In the following paragraphs, we will discuss the impacts of *base_age* and *max_age* in detail.

2) Max_age: In Figure 9(a), we show the network delay under various *max_ages*. The cache size is set to be 200 MB. We see that *max_age* has different affects under different *base_ages*. For example, when *base_age*=50 seconds, the network delay decreases from 18.2 ms to 15.5 ms as *max_age* increases from 30 seconds to 100 seconds, while the network delay remains static when *base_age*=10 seconds. The result is likely explained by the on-path filtering effect of ABC. Note that the age of the replica of a content object will increase along the data path until it reaches *max_age*. In our simulation, the average length of the data path from server to the clients is 3 hops. This means that if we set *base_age* to be 10 seconds, the age of a content object will hardly reach *max_age* if *max_age* is greater than 30 seconds. As a contract, when *base_age*=50 seconds, it will never increase as *max_age*=30 seconds, which breaks the principle that the closer a content object is to the network edge the longer it is cached. That explains why the *base_age*=10 seconds achieves more delay reduction than *base_age*=50 seconds. On the other hand, when *max_age* increases to 100 seconds, the age of a replica can increase as it gets closer to the client. Because a popular content object can be cached longer when *base_age*=50 seconds than *base_age*=10 seconds, the performance of *base_age*=50 seconds becomes better than *base_age*=10 seconds when *max_age*=100 seconds.

3) Base_age: In Figure 9(b), we show the average network delay under various base_ages. Similar to the evaluation of *max_age*, the cache size is set to be 200 MB. We see that *base_age* noticeably affects the performance of ABC. Take *max_age*=30 seconds, for example, the network delay increases from 17 ms to 18.2 ms as *base_age* grows from 10

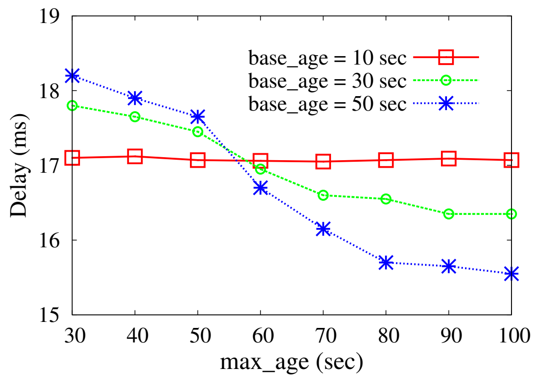


(a) max_age = 100 ms



(b) base_age=30 ms

Fig. 8. Impact of cache size on ABC under various base_ages and max_ages



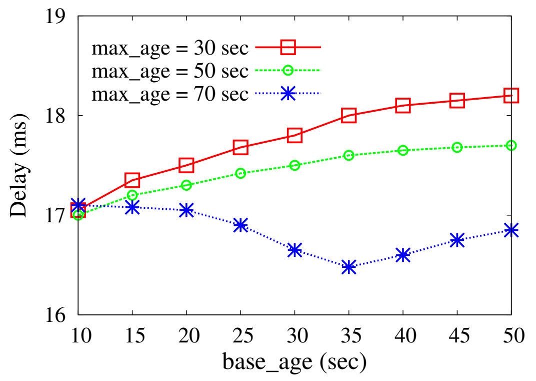


Fig. 9. Impacts of base_age and max_age on ABC

seconds to 50 seconds. We believe that this is because the age of a replica will quickly approach the *max_age* (30 seconds) as *base_age* increases (e.g., after 1 or 2 hops). Another observation is that the larger the gap between *base_age* and *max_age* is the better the performance of ABC may be. For example, the performance of *max_age*=70 seconds is better than that of *max_age*=30 seconds. Further more, when *max_age*=30 seconds, the network delay decreases as *base_age* increases. In contract, the network delay of *max_age*=70 seconds first increases then decreases as *base_age* increases. We believe this is because a sufficiently large gap between *base_age* and *max_age* can efficiently push popular content objects to the network edge while still leave cache space for unpopular content in the network, which effectively leverage the in-network cache.

V. Discussion

When a router caches an object, it has twofold consequences. If the object is "read-only", then the router simply benefits. However, when the object is "read/write", the problem of consistency will arise. writes require the router adopts a consistency maintenance algorithm. The consequence might be that the router may actually incur a performance penalty by caching an object due to the cost of consistency maintenance.

In this paper, we focus on "read-only" contents. Our motivation is based on the following observations. First, a major reason for routers to cache read/write objects is that the object is so important that it must always be readily available for read access. In other words, the "delta" between local and local remote access is so large that it makes sense to maintain replicas of "write" objects despite the additional overhead [20]. In practice, this constraint is hardly so strong, as long as the object can be accessed from its server. Second, the read-only contents are more than enough to fulfill the routers' caches. On one hand, it has been recognized that the majority of today's Internet traffic is occupied by streaming services. In such services, contents will remain static throughout their lifetime. On the other hand, for non-streaming services, there exist a large portion of objects that do not change with time (e.g., static web pages). Other technology instead of caching can be developed for read/write objects, which is beyond the scope of this paper.

VI. Related Work

Many efforts have been devoted to in-network caching. For example, ProbCache [7] proposes a probabilistic in-network caching algorithms with the aim of reducing caching redundancy. Multicache [21] proposes an overlay network architecture that is based on multicast and in-network caching. [22] studies the behavior of content centric networking and proposes a centrality-based caching algorithm to eliminate the uncertainty in the performance of random caching strategy.

For the most relevant work, [8] proposes a cooperative caching strategy with some of the motivations similar to ours. This work aims at halving the cross-domain traffic by creating cluster of caches that work as a federation and maintaining only R copies of the same object in the cluster. In contrast, we focus on optimizing in-network cache by leveraging federation at the network path level by using an age-based approach that is efficient for information-centric networking. WAVE [9] proposes a popularity-based cache placement strategy regardless of specific network topologies. In WAVE, an upstream router explicitly suggests what content chunks to be cached at its next downstream router. While WAVE is an effective caching scheme, it has difficulty in achieving preferable caching performance globally. In contract, ABC enables the routers to make decisions from a content delivery path perspective and optimizes both global and local performance.

VII. Conclusion

This paper has proposed a lightweight age-based cooperative caching scheme (ABC) for information-centric networking. The scheme uses 'age' as the signal to control the lifetime of a content replica in the router and adaptively pushes popular content objects to the network edge. We perform extensive evaluation using real traces and realistic network topology. Results show that our approach can significantly reduce network delay and traffic, and outperforms existing schemes. As for future work , we will formally model the agorithm and theoretically analyze the impacts of *cache size*, *base_age* and *max_age* to optimize ABC's performance.

Acknowledgment

The research is supported by the National Basic Research Program of China (973 Program) under Grant 2012CB315803, the National Natural Science Foundation of China (61133015 and 61161140454), Specialized Research Fund for the Doctoral Program of Higher Education under grant (20120002110060).

References

[1] T. Koponen, B. Chawla, A. Emolinskiy, H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *ACM SIGCOMM*, 2007.

[2] Named data networking (ndn) project. PARC Technical Report NDN-0001. [Online]. Available: http://www.named-data.net/ndn-proj.pdf

[3] H. Lee and A. Nakao, "User-assisted in-network caching in information-centric networks," *Computer Networks*, vol. 57, pp. 273–281, 2013.

[4] H. Che, Z. Wang, and T. Y, "Analysis and design of hierarchical web caching systems," in *IEEE Infocom 2001*, 2001, pp. 1416 – 1424.

[5] E. Cohen and H. Kaplan, "Aging through cascaded caches: Performance issues in the distribution of web content," in *ACM SIGCOMM 01*, 2001.

[6] Y. Xu, Y. Li, T. Lin, Z. Wang, W. Niu, H. Tang, and S. Ci, "A novel cache size optimization scheme based on manifold learning in content centric networking," *Journal of Network and Computer Applications*, vol. 37, pp. 273–281, 2014.

[7] I. Psaras, W. K. Chai, and G. Pavlou, "Probabilistic in-network caching for information-centric networks," in *Proc. of the second edition of the ICN workshop on Information-centric networking*, 2012, pp. 55–60.

[8] Z. Li and G. Simon, "Time-shifted tv in content centric networks: the case for cooperative in-network caching," in *Proc of IEEE Int. Conf. on Communications (ICC)*, 2011.

[9] K. Cho, M. Lee, K. Park, T. Kwon, Y. Choi, and S. Pack, "Wave: Popularity-based and collaborative in-network caching for content-oriented networks," in *IEEE INFOCOM NOMEN Workshop*, 2012.

[10] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networks," in *IEEE INFOCOM NOMEN Workshop*, 2012.

[11] J. Ni and D. H. K. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," in *IEEE Commun. Mag*, 2005, pp. 98 – 105.

[12] I. D. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," in *SIAM J. Comput*, 2008, pp. 1411 – 1429.

[13] P. Sarkar and J. H. Hartman, "Hint-based cooperative caching," in *ACM Trans. Comp. Syst*, 2001, pp. 387 – 419.

[14] V. Cate, "Alexa global file system," in *USENIX File System Workshop*, 1992.

[15] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, "Dns performance and the effectiveness of caching," in *IEEE/ACM Trans. Networking*, 2002.

[16] X. Shi, J. Han, Y. Liu, and M. L. Ni, "Popularity adaptive search in hybrid p2p systems," *Journal of Parallel and Distributed Computing*, 2008.

[17] W. Jianping, C. Yong, L. Xing, and C. Metz, "4over6 for the china education and research network," in *IEEE Internet Computing*, Jun. 2006.

[18] Ns2 website. [Online]. Available: http://www.isi.edu/nsnam/ns/

[19] E. Chlebus and J. Brazier, "Nonstationary poisson modeling of web browsing session arrivals," in *Information Processing Letters*, vol. 102, no. 5, 2007, pp. 187–190.

[20] A. Leff, J. L. Wolf, and P. S. Yu, "Replication algorithms in a remote caching architecture," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 11, pp. 1185–1204, 1993.

[21] K. Katsaros, G. Xylomenos, and G. C. Polyzos, "Multicache: An overlay architecture for information-centric networking," *Computer Networks*, vol. 55, no. 4, pp. 936–947, 2011.

[22] W. K. Chai, D. He, I. Psaras, and G. Pavlou, "Cache less for more in information-centric networks (extended version)," *Computer Communications*, vol. 36, no. 7, pp. 758 – 770, 2013.