Age-based Cooperative Caching in Information-Centric Networks

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 to obtain information. One of the most important commonalities of ICN designs is the universal caching. It is widely accepted that the in-network caching would improve performance. However, there has been no consensus on how to design an efficient caching scheme in ICN network. In this paper we propose an age-based cooperative cache scheme aiming at reducing network delay and publisher load for ICN network. We focus on light-weight collaboration mechanisms that spread popular contents to the network edge while at the same time fully utilize the storage capacity of intermediate nodes. We evaluate the effectiveness of our scheme under real traces and realistic network topology. The results indicate that our scheme achieves significant performance gains.**

I. Introduction

The Information-Centric Network has produced plenty of results in the international Future Internet research activities [1], [2], [3], [4], [5]. One of the most important commonalities of ICN designs is the *universal caching*. It is believable that the universal content caching would improve performance. It is straightforward that the network delay and publisher load can be reduced by having content requests satisfied at network edge as much as possible. Popular contents, therefore, should be cached near the users.

While we can easily keep popular contents at network edge by leveraging traditional Web caching policies, e.g., LRU, most such policies need sophisticated hardware support. Cooperative caching and age-based caching are expected to be more efficient, with the benefits investigated [6], [7], [8], [9], [10], [11], [12]. However, as detailed in section II, they have limitations in different ways which make them not suitable for ICN networks.

By now, there has been no consensus on how to design an efficient caching scheme in ICN network.

Motivated by the above issues, we propose an age-based cooperative scheme to the design of efficient ICN caching system by fully utilizing the benefits of cooperative and age-based mechanisms.

The aim of the scheme is the reduction of both network delay and publisher load. We achieve this by providing light-weight cooperative mechanism under age-based control. Contents' ages are dynamically changed by network nodes in an implicitly cooperative manner. This scheme spreads popular contents to the network edge while at the same time eliminate unnecessary content replication at intermediate network nodes.

We conduct comprehensive simulations using the topology of China Education and Research Network 2 (CERNET2), the world's largest native IPv6 backbone network. We leverage the real traces gathered from a commercial website of China to evaluate our scheme. The results show that our scheme can achieve significant gains to improve network performance.

The contribution of this paper is as follows.

- We analyze the problems of existing mechanisms and propose an age-based cooperative caching scheme for ICN architecture. As far as we know, this is the first work on such type of scheme in ICN research activities.
- We design two light-weight algorithms to realize this scheme.
- We evaluate the impact of the algorithms under real trace and realistic network topology. Results show there are significant performance gains by using the scheme.

The rest of this paper is organized as follows. Section II describes the the problems of existing mechanisms and points out our motivation. Section III presents our age-based cooperative scheme and designs two algorithms to realize the scheme. A case study is conducted to intuitively show the scheme's benefit. Section IV describes our experiment methodology as well as the results. Section V discusses the consistency issues. Section VI concludes our work and outlines further work.

II. Problem statement

The ICN network 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, it is clear that the closer a node with a content copy is to a user, the less time the user requires to get the content. It is observable

The research is supported by the National Natural Science Foundation of China (61073166 and 61133015), the National Basic Research Program of China (973 Program) under Grant 2012CB315803, the National High-Tech Research and Development Program of China (863 Program) under Grants 2011AA01A101.

that 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 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.

Cooperative mechanisms appear to be more efficient in ICN scenarios. The benefits of cooperative caching have been investigated in [6], [7], [8], [9]. 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 [10], web caching [11], and DNS [12]). 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 configuration, which is 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 age-based cooperative scheme dynamically configure content's age in an implicit cooperation manner among ICN routers. The principle of the scheme is as follows.

- Each router has an age. The lifetime of a replica is decided by its age in each router.
- 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 guide the age, which obeys the following rules:

- The closer a replica is to the network edge, the longer age it has.
- The more popular a replica is, the longer age it has.

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

Figure 1 presents a mini network 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 client successively requests the two contents according

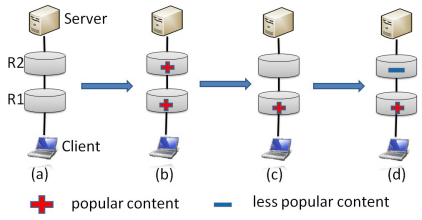


Fig. 1. Case study

to their popularity. For 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 our 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. 2) The cache memory of the node has been full.

We design a light-weight 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 contents to be pushed to the network edge while at the same time release 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 [13]. 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 yes, the router removes it and adds the new object into its cache. Otherwise, the object won't

be cached. Upon caching an object, a router performs the *get_new_age()* function 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 + age * weight;
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.

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.

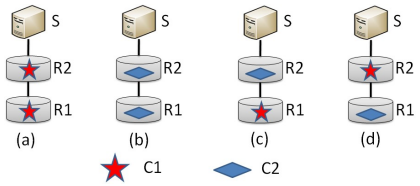


Fig. 2. Case study

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

C. A case study

In this section we conduct a case study with a simple scenario to intuitively show the benefits of the age-based cooperative scheme. For simplicity, we call the scheme *ABC* for short. We want to study *ABC*'s impact on network delay and publisher load. To make comparison, we use *LRU* to compete with *ABC* 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 R1. 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:

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

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* happens 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 simulation under realistic network topology to evaluate *ABC*'s performance.

IV. Performance evaluation

A. Simulation methodology

We evaluate the performance of our scheme using the topology of CERNET2, which is the world's largest native IPv6 backbone network [14]. We make use of the data trace on various topics obtained from a commercial Chinese website that focuses on localized services of Beijing (*www.wdklife.com*). The topology is in conformity with the user distribution of our trace (detailed in section IV-B1), 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. Our work also serves as a reference for the future deployment of in-network caching on CERNET2. Figure 3 shows the CERNET2 topology.

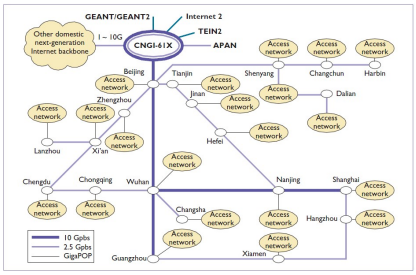


Fig. 3. CERNET2 topology

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 100M. The link weights between the core routers are interpreted as delays (ms), which are set to be 10 ms. A shortest path routing protocol is used in the network.

Content items are stored in a single server located in *Beijing*, which is the same as *www.wkdlife.com* really does. 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 [15]. 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.

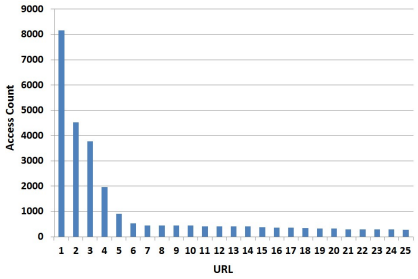
$$\frac{\sum_{i=1}^{SIM\_TIME} \sum_{j=1}^{rc_i} delay\_of\_request\_j / rc_i}{SIM\_TIME} \quad (5)$$

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.

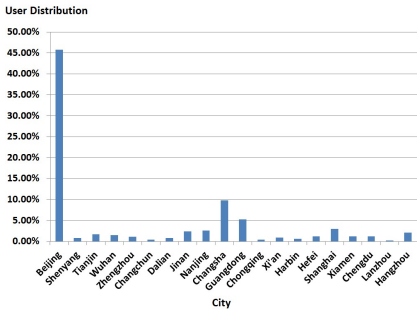
We implement an event-driven simulator to evaluate *ABC*'s performance. The *BASE_AGE* is set to be 10 seconds and the *MAX_AGE* is set to be 60 seconds. The simulation lasts for 100 minutes. For each evaluation target, we make use of the performances of *LRU* and *FIFO* in the same scenario as a comparison. It has long be recognized that the Internet follows the *Pareto Principle*, in which 20% of the contents contribute 80% of the traffic. LRU has been analytically and experimentally proved efficient for web pages whose distribution obeys the *Pareto Principle*. The motivation of using *FIFO* is that it is easy to implement and widely deployed in practical systems. Results are shown in section IV-B.

B. Results

1) Characteristics of the trace: The trace was captured during a one-month period in 2011 using online statistical tools, which totaled 76,414 URL-requests sent by 36,432 users from 20 cities of China. We filter this trace and retrieve 8,369 distinct URLs and use these URLs as our simulation input. We range the URLs from 1 to 8,369 according to their access count



(a) Access count per URL (top 25)



(b) User Location Distribution

Fig. 4. Trace information

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. For the purpose of simplicity, the size of the content is uniformly set to be 1M.

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.

2) Evaluation results: Figure 5(a) shows how *aggregated network delay* varies as different schemes are used. The figure shows the potential performance gains through the use of *ABC*. The *aggregated network delay* of *ABC* is around 16ms during the simulation, comparing with 21ms of *LRU* and 29ms of *FIFO* respectively. There is a dramatic drop for the curves of *ABC* and *LRU* at the beginning of the simulation, while the performance of *FIFO* remains relatively constant. This is because given the power-law-distributed contents, *ABC* and *LRU* can utilize content's popularity in their replacement strategies, which quickly increases the hit probability.

Figure 5(b) illustrates the effects of *ABC*, *LRU* and *FIFO* on *end user delay* in each city in the network. The result is in accordance with *aggregated network delay*, such that *ABC* achieves the shortest delay, while *FIFO* has the largest delay. There are, however, different performance gains for different cities. By looking at the horizontal axis of Figure 5(b) from left to right, we can distinguish the cities into 5 groups, in which each group has approximately the same delay. The reason for this phenomena is that cities in the same group have the same distance to the server under the shortest path routing scheme.

Another observation is that the further a router is from the server, the more latency reduction *ABC* provides than *LRU* and *FIFO*. For example, *ABC* achieves 88.4% delay of *LRU* in *Beijing*, while its delay reduces to 69.7% of *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 *FIFO* decrease dramatically as distance grows. This is a major benefit of *ABC* as users that are far away from the content server won't 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. While the increased age may lead to potential consistency problem, we argue that this will not be a serious issue. Section V discusses the consistency issues in detail.

The previous two figures present the impact of caching on the network delay. In Figure 5(c), 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* saved more traffic than *LRU*, while *FIFO* has the worst performance, with traffic reduction of 50%, 33% and 14% for *ABC*, *LRU* and *FIFO* respectively. There is a drastic decrease in server traffic at the beginning of the curves of *ABC* and *LRU* as they give popular contents more priority to live in cache.

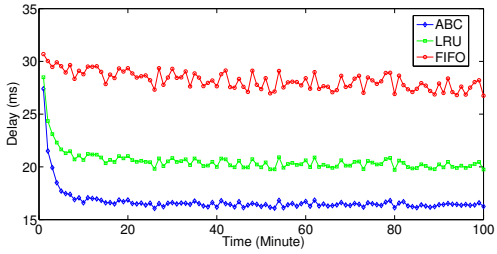
The results in the figures show that there are significant gains through using *ABC*. The users will benefit from *ABC* as the total download latency will be lowered. On the other hand, content providers will be able to greatly reduce the traffic to save bandwidth and computing resources.

There are, however, problems that need to be addressed. 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 investigate how to optimize the parameters to enhance *ABC*'s performance in future work.

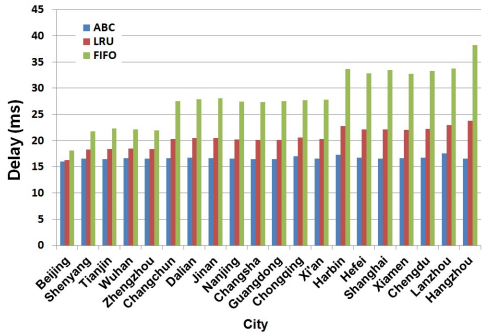
V. Discussion

Consistency is important to the sharing of contents in ICN networks. While our algorithm achieves significant benefits in reducing network delay, it may have drawbacks on maintaining highly dynamic contents, as nodes which are far from the server may take a relatively long time to refresh their contents. The drawback, however, can be viewed from the following aspects:

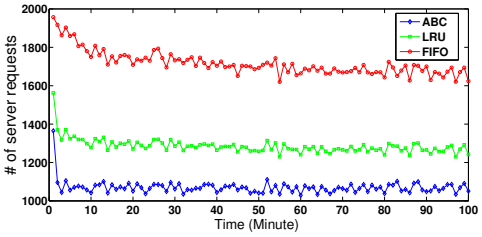
First, it has been widely proved that streaming services occupy the majority of traffic to today's network. In such



(a) Aggregated network delay



(b) End user delay



(c) Server load

Fig. 5. Evaluation results

services, contents will remain static during their lifetime. For instance, the video file of *Harry Potter* will stay unchanged unless the provider removes it from the server. For these applications, there is no consistency problem caused by our scheme.

Second, even for non-streaming services, there still exist a large portion of contents that do not change with time. Take the website which we used to get the traces for example, more than 90% of its web pages won't be modified once they are created. This situation also applies to larger websites such as *yahoo* and *sina*. As a result, we believe our caching scheme can satisfy most users' demands.

Finally, for time critical applications such as VoIP, we can extend our algorithm by explicitly set this type of contents a short, non-increasing, or even zero age to avoid inconsistency. Routers can also periodically poll the server to keep the contents up to date. These mechanisms, however, are beyond the scope of this paper and will be investigated as future work.

We notice that a recent work [16] proposes a cooperative caching strategy for NDN, 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 a same object in the cluster. In contrast, we focus on minimizing access latency and publisher load by leveraging federation at the network path level.

VI. Conclusion

This paper focuses on the design of efficient caching within the ICN architecture. We proposed an age-based cooperative scheme and designed two algorithms to realize it. To the best of our knowledge, this is the first attempt to the study and evaluation of such type of policy in presence of multiple contents sharing the same ICN infrastructure. We performed extensive simulation under real traces and realistic network topology. Simulation results show the benefits deriving from our scheme, with network delay and the publisher load reduced significantly. We plan to further improve the scheme through theoretical analysis and implementation to identify problems in the design and evaluate performances at different scales.

References

[1] M. Gritter and D. R. Cheriton. (2000, Jul.) Triad: A new next-generation internet architecture. [Online]. Available: http://www-dsg.stanford.edu/triad/

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

[3] Z. Lixia. (2010, Oct.) Named data networking (ndn) project. PARC Technical Report NDN-0001. [Online]. Available: http://www.named-data.net/ndn-proj.pdf

[4] V. Jacobson, D. K. Smetters, N. H. Briggs, and P. M. F, "Voccn: Voice-over content-centric networks," in *ReArch'09*, Dec. 2009.

[5] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayannan, I. Stoica, and S. Shenker, "Rofl: Routing on flat labels," in *ACM SIGCOMM*, 2006.

[6] 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.

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

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

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

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

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

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

[13] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *IEEE INFOCOM*, 2010.

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

[15] 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.

[16] L. Zhe and S. Gwendal, "Time-shifted tv in content centric networks: the case for cooperative in-network caching," in *Proc of IEEE Int. Conf. on Communications (ICC)*, 2011.