Cache Management in Content Delivery Networks Using the Metadata of Online Social Networks

Abdorasoul Ghasemi[a,b,*], Amirhosein Ahmadi[a]

[a] *Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran 1631714191, Iran*
[b] *School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran*

Abstract

The number of requests on content delivery networks (CDN) originating from the online social networks (OSN) by sharing the content weblink increases according to recorded data. The sequence of the OSN originated requests shows temporal burstiness with a typical interval shorter than the ordinary requests. We consider CDN and OSN as a multilayer network and exploit the average spreading power of each user in the OSN to predict the temporal pattern of the corresponding consecutive social requests that may originate from this user to improve the underlying cache management mechanism. The traditional least recently used (LRU) content replacement algorithm uses the statistical popularity of contents to increase the cache's hit ratio. We propose LRU-Social, which defers the eviction of social requests for a specific amount of time to take advantage of the possible burstiness in the underlying interval without missing the popular contents' hits. We model the content link sharing by the susceptible-infected-recovered (SIR) spreading process in the underlying OSN to compute the user spreading power. We provide numerical studies for synthetic streams consisting of ordinary requests that follow Zipf's popularity model and social requests to justify the effectiveness of the LRU-Social compared to the LRU.

Keywords: Content delivery networks, Cache management, Online social networks, Multilayer networks

1. Introduction

Content delivery networks (CDNs) like Akamai and Amazon provide efficient delivery of contents to geographically distributed end-users and are responsible for a large amount of total traffic on the Internet [1]. CDNs deploy a cluster of surrogate servers that work as cache and prepare multiple points of pretense over the network. The cache reduces the access delay, bandwidth costs, and the load on the origin server(s) by responding to requests from the best surrogate servers [2]. Considering the limited storage space of surrogate servers, the proper replacement of contents decreases the miss rate of content requests as a crucial performance metric in CDNs. Therefore, many studies focus on designing a content replacement algorithm considering the hierarchical and distributed structure of CDN, the dynamics of content requests, and the regional popularity of contents [3]. The key

[*] Corresponding author
Email address: arghasemi@kntu.ac.ir (Abdorasoul Ghasemi)

distinguishing feature of these algorithms is which item should be evicted and replaced with the new fetched item from the origin server after a miss.

Online social networks (OSNs) facilitate content sharing between users by offering web links to the contents. For example, recorded data show that about one billion content is shared daily via Facebook subscribers, or more than 400 messages linking to Youtube are communicated between Twitter subscribers [4]. Another study reveals that, on average, about 40% of YouTube requests originate from social media [5]. Therefore, the dynamics of content link-sharing, which is a spreading process in OSNs, affects the pattern of content requests, and as a result, the dynamics of content replacement in CDNs. More precisely, we can classify the CDN requests as social and ordinary when the social requests originate from the weblinks shared on the OSNs.

Social networks could help to predict real-world outcomes like revenues of movies by analyzing social media discussions about them [6], the latent sub-communities structure in open source software projects [7], and the geographic area distribution of content sharings [4].

Many research pieces investigate the spreading behavior in the OSNs and multilayer networks [8, 9]; however, less attention is paid to its effects on the content spreading in CDNs. In this work, we consider how the web link sharing in the OSNs as a piece of side information affects the cache management in the CDNs. We use the predictive aspect of the OSNs' structure to estimate the spreading power of the users in content link sharing as a factor for creating a particular type of request for CDNs, called social request. Then, we exploit the burstiness and local distribution features of this kind of request to enhance cache management in CDNs. Using the spreading power of subscribers in OSN as metadata, we develop a least recently used (LRU) based content replacement algorithm (LRU-Social) that defers the eviction of contents that currently have social requests for a specific amount of time.

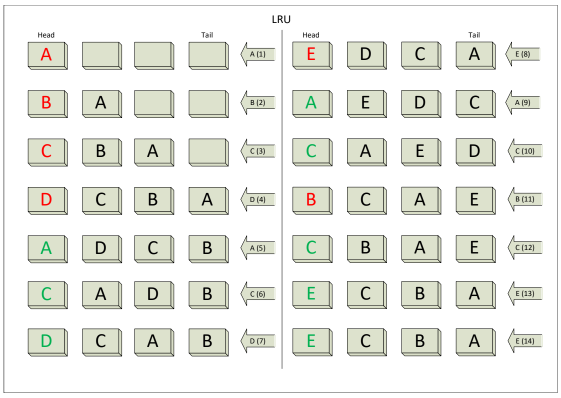
In the rest of this paper, we review some related works and provide the background materials in Section 2. In Section 3, we discuss the solution approach for exploiting the information flow in OSN to improve the cache management in CDNs and propose the LRU-Social scheme. In Section 4, simulation results are provided to justify the effectiveness of the LRU-Social. Conclusions and future works are provided in Section 5.

2. Backgrounds and Related works

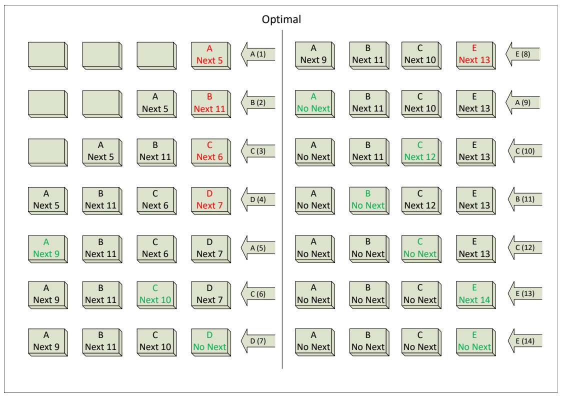
In this section, we first briefly review the required backgrounds and related works on replica management in CDNs. Next, we review backgrounds on the spreading process in OSNs that we use in subsection 3.3 for estimating the spreading power of OSNs subscribers.

2.1. Replica management in CDNs

CDNs use cache-like data warehouses called the surrogate server to bring the popular contents closer to the end-users, which leads to a scalable solution for content delivery and relief of the bottlenecks in their networks. The cache's effectiveness is directly related to its content replacement algorithm, which determines the content that should be evicted and replaced with the new fetched



(a)


Optimal
A Next 5
A (1)
A Next 9, B Next 11, C Next 10, E Next 13
E (8)
A Next 5, B Next 11
B (2)
A No Next, B Next 11, C Next 10, E Next 13
A (9)
A Next 5, B Next 11, C Next 6
C (3)
A No Next, B Next 11, C Next 12, E Next 13
C (10)
A Next 5, B Next 11, C Next 6, D Next 7
D (4)
A No Next, B No Next, C Next 12, E Next 13
B (11)
A Next 9, B Next 11, C Next 6, D Next 7
A (5)
A No Next, B No Next, C No Next, E Next 13
C (12)
A Next 9, B Next 11, C Next 10, D Next 7
C (6)
A No Next, B No Next, C No Next, E Next 14
E (13)
A Next 9, B Next 11, C Next 10, D No Next
D (7)
A No Next, B No Next, C No Next, E No Next
E (14)

(b)

Figure 1: Stored contents and their priority for eviction are shown for (a) the LRU and (b) the Optimal content replacement algorithms. The request stream is *A, B, C, D, A, C, D, E, A, C, B, C, E, E*, and the cache size is 4. Misses and hits are shown in red and green colors. In the Optimal algorithm *Next* variable shows the time distance to the next call of this content in the input request stream.

content after a miss. The hit rate as a performance metric of caches refers to the ratio of successful responses from the cache to the total number of requests. Recency, frequency, and size are the main characteristics used in algorithms [10]. The least frequently used (LFU) and least recently used (LRU) and their many variants are typical algorithms for content replacement in CDNs. The LFU evicts the content with the least number of references, and the LRU evicts the item with the oldest time to reference. In this article, we use the LRU algorithm, which shows good performance in the dynamic environment of web caching and has a simple implementation. The web caching algorithms like LRU-threshold or and LRU-min use the base LRU and the contents' sizes for decision making. [11]. The performance of each algorithm is usually compared to the non-causal algorithm called Optimal, which knows the list of all future requests. The Optimal algorithm exploits this information and evicts the item that would have the farthest future reference as the best choice and provides an upper bound for the hit rate.

Fig.1 helps to understand better how the LRU and Optimal algorithms work by showing stored contents and their priority for eviction when each request of stream *A, B, C, D, A, C, D, E, A, C, B, C, E, E* arrives for the cache with size 4.

Various studies have worked on the characterization and classification of classic content replacement algorithms in caches [3, 10]. Also, many studies have proposed new methods, some in the form of modifications to classical algorithms like LRU and LFU, and some others suggest different approaches. The [12] emphasizes the fact that certain parts of video content are seen more than other parts and offers a new approach based on LRU called Chunk-LRU, which uses partial caching. Note that even for the most popular ones, on average, only 72 percent of each video is watched. The method calculates the audience retention rate parameter for each part of the videos, which indicates the probability of watching that part and making decisions to store them. Also, to evaluate this method's efficiency, the core network traffic has been used as a comparison criterion. Another study [13] first addresses that among the factors on which the cache-hit ratio of a caching system depends, only capacity and replacement algorithm are configurable and controllable, in which the effect of capacity is negligible. It then reviews the Prob algorithm and points out that this algorithm performs well only when the content popularity distribution is static or slowly changes. Finally, it introduces the AProb algorithm, which uses three techniques, including dynamic probabilistic caching, Ghost list, and adaptive probing and protection, to cover the shortcomings of the Prob algorithm in modern caching networks, where the traffic exhibits signs of dynamic content popularity. The [4] takes advantage of the fact that social torrents can propagate in a geographically limited area to discern whether an item is spreading locally or globally. The suggested method extracts this geographic information of social torrents and utilizes them to ensure that content relevant to a social torrent is kept close to its users.

2.2. Spreading process in OSN

The OSNs facilitate the spreading of behaviors and information among the subscribers. The spreading model and the network structure determines how the process takes place over the network [14]. There is two class of spreading models named as the threshold and independent interaction models [15]. In the threshold models introduced by [16], each node becomes active and helps in the spreading of behavior or opinion if the number of its neighbors that adopted the behavior exceeds a certain threshold. The threshold depends on the total number of node's neighbors and the average degree of the networks. The independent interaction model inspires by disease spreading and assume that each interaction of two subscribers may help the spreading with a given probability. The susceptible-infected (SI), susceptible-infected-susceptible (SIS), and susceptible-infected-recovered (SIR) are three extensively studied spreading models in this class [14]. Specifically, in the SIR model, each susceptible node may change to the infected due to an interaction with an infected node with a given probability β. Also, each infected node may recover with a probability γ, and in that state, it would not be infected again in future interactions nor change the state of other nodes. In this model, the fraction of nodes that are never infected and the fraction of recovered nodes are the most important criterion. Also, the spreading power of a node is the fraction of nodes that will be infected in the spreading process, starting with that node.

Note that both the adopted model and the underlying network structure affect the result of the spreading process. Although the threshold models are more popular for explaining the behaviors in OSNs [17], the disease's epidemic models are as well plausible to explain the opinion diffusion in the OSNs. For example, in [18], the SIR model is used to explain the topic diffusion in the web forums and justified on large longitudinal datasets.

3. Solution Approach

Recent studies show that a significant fraction of all content requests initiate from the OSNs. Consider an individual who generates content or finds exciting content in a CDN, motivating him to share the link of this content with his friends in OSN. Since the friends of an individual have the same interests, they may request that content from the CDN and share this link with their friends. This process is like spreading the content weblinks among the subscribers of the OSN.

This spreading process has specific features that help to model its behavior and predict the spreading power of the user, which initially generates or shares the corresponding content link as discussed in subsection 3.3. Furthermore, the analysis of real recorded data shows other geographical and time constraints of content weblink sharing in OSN, which has a practical implication for CDN. For example, the studies on Youtube video sharing in OSNs show two specific features [5, 19]. First, the social originated requests usually confined in a specific geographic region means that these requests would be directed to specific surrogate servers in the CDN. Second, the pattern of these requests over time shows a peak in a limited period means that the interval of social requests to the CDN is shorter than ordinary requests. On average, the interval for social requests is limited and is in the order of cache size [20].

We exploit social requests' geographic and time constraints features alongside each node's estimated spreading power from the SIR model to improve the LRU algorithm. The designed LRU-Social content replacement algorithm has a better adjustment for eviction than the LRU.

3.1. The impact of content requests interval

In the LRU algorithm, the cache saves the most recent contents which users request. The worst-case scenario is when the probability density function (pdf) of contents' popularities has uniform distribution in which each of N contents may be requested with probability $\frac{1}{N}$. The request interval of all contents is then given by the geometric distribution with the expected value of N for two consecutive requests of specific content. Since the cache size is much smaller than the number of contents, caching has no benefit in this scenario. However, different studies show that the web content's popularity pdf does not follow a uniform distribution; a small number of contents are popular with many requests, and a large number of contents are rarely requested. The popularity of web contents are usually modeled by the Zipf distribution in which the probability p_i of requesting content i is proportional to the inverse of its rank, r_i, $p_i \propto \frac{1}{{r_i}^{\alpha}}$, where α is a parameter between 0.64 to 0.83 [21]. The popularity model determines each item's request interval distribution, which affects the performance of recency-based algorithms like the LRU. In Fig.2 (a), the pdf of request interval in a stream of 10000 requests for $N = 100$ contents, which follow uniform and Zipf popularity models, are shown. We observe that the request interval pdf of the Zipf model is more scattered than the uniform model, where the spreading is greater as we increase α. Fig.2 (b) shows the corresponding hit rate value for applying the LRU and Optimal for different models. We observe that as the probability of having consecutive requests with smaller intervals increases, the performance of both LRU and Optimal algorithms increases.

If the number of requests for unique contents between two consecutive requests of the specific content is greater or equal than the cache size, irrespective of current contents in the cache, the LRU algorithm evicts that content before the arrival of its second request, and this decision leads to a miss. However, the Optimal algorithm's behavior depends on the current contents in the cache and indeed misses the second request with a small probability. The Optimal algorithm evicts that content only if between consecutive requests (i) the first request points to a content which is not currently

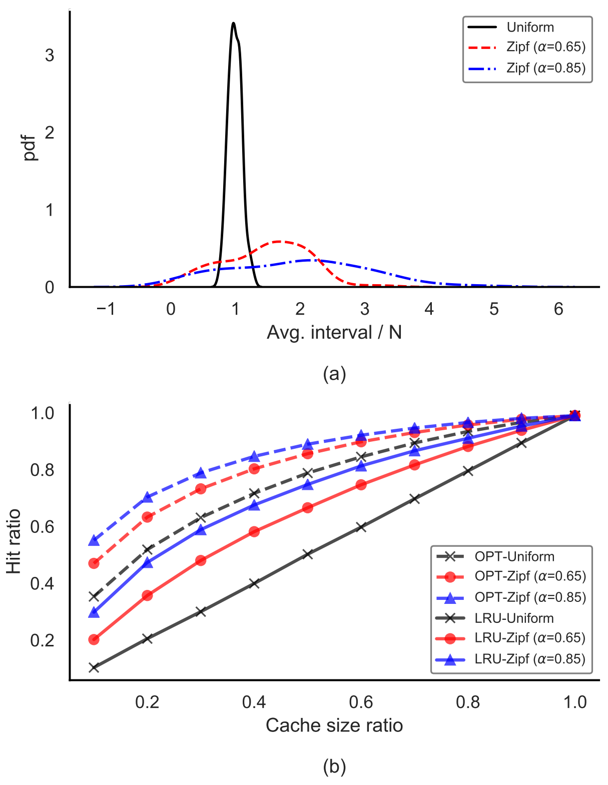


Figure 2: (a) The probability density function of the request interval in a stream of 10000 requests for $N = 100$ contents with different popularity models. The pdf is smoothed by Gaussian kernel over the numerical histogram. The interval for the uniform popularity model follows the geometric distribution highly peaked around N while they are more scattered for the Zipf model. (b) The hit ratio of the LRU and Optimal algorithms are, respectively, shown by solid and dashed lines.

in the cache, and (ii) either all of the contents in the cache or except one of them will be requested in the remaining requests. Therefore, the Optimal algorithm misses the second request with a small probability than the LRU, which misses that request with probability one.

As an illustrative example, assume that the cache size is $C = 4$, and the total number of contents is $N = 10$. Consider a scenario in which the stream of requests consists of two close social requests of a content, denoted by *SR*, and four ordinary requests: *SR*, R_1, R_2, R_3,R_4, *SR*. Irrespective of current cache contents, the LRU algorithm would cache *SR*, R_1, R_2, R_3 and then AHosein~~evicts~~ evict *SR* to store R_4 and hence miss the second social request. However, the Optimal algorithm will evict the second social request only if the current cache contents include *SR* and R_2, R_3,R_4. The Optimal algorithm misses the second request with probability *0.04* assuming uniform distribution for a request to these content. See Appendix A for details of computing this probability. That is, the LRU makes the wrong decision in *96 %* of situations. Note that with increasing the interval between the two consecutive social requests, the miss probability of the Optimal algorithm increases, as shown in Fig.3 for uniform popularity of contents. However, it is still saving a performance margin compared to the LRU even if the interval is multiple times of cache size. The performance gap for the Zipf popularity model in Fig.2 (b) also shows this fact; the LRU algorithm only takes advantage of the requests which their interval are less than the cache size, while there is room to improve the performance even when the interval is greater than the cache size. This suggests that we could improve the performance of the LRU-social if we have an estimate of the interval distribution of social requests.

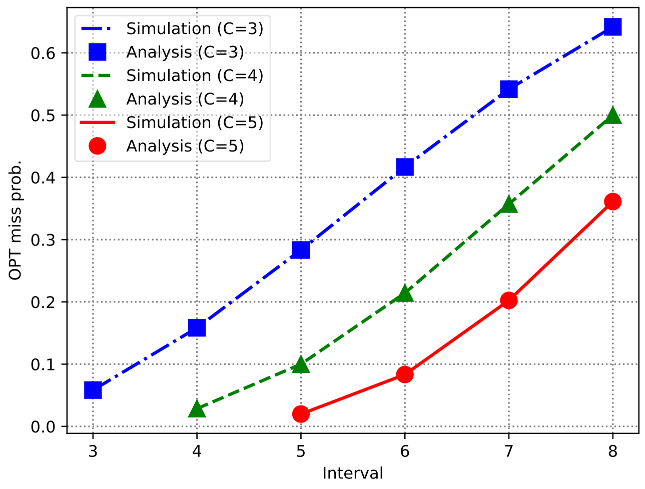


Figure 3: The probability that the Optimal algorithm misses the second content against the different time intervals to its first request is shown. The popularity model is uniform, and the intervals equal to or greater than the cache size are considered. The LRU will miss the second request with probability one.

3.2. LRU-Social

The main idea of the LRU-Social is to exploit the short interval between consecutive social requests and prioritize the related contents by keeping them in the cache for a longer time than the LRU. This strategy will help to capture the hits of consecutive requests with interval spaces greater than the cache size.

We start by introducing a simple label-based implementation of the LRU. We consider a numerical label for each content in the cache and update the labels after a hit or miss. The strategy of assigning the initial label and label updating determines the desired priority for keeping the cache's contents. The initial contents' labels are equal to the cache size C in the naive LRU scheme. After each hit, we update the label of the requested content to C, and decrement the labels of contents that have greater label than the requested content. After each miss, the content with the least label will be evicted, and the labels of other contents will be decremented. The new fetched content will be cached with a label C. This ensures that, as in the LRU, we always evict the farthest cached item.

In the LRU-Social algorithm, we treat the contents related to social requests differently compared to ordinary requests by assigning appropriate labels according to the users' social influence behind those requests measured by the spreading power. Assume that we estimate the total number of future social requests of a social request j, denoted by N_j. Also, let D_0 be an average distance between the two consecutive social requests, which we know is in the order of cache size. Then, the interval between the first and last social request for content j is $D_j = D_0(N_j - 1) + (N_j - 2)$. Therefore, if we set the initial label of the first social request j by $l_j = D_j + C + 1$, we make sure that (i) the corresponding social content j would be kept in the cache until the arrival of the last social request for this content, and (ii) after the arrival of the last social request, the updated label is C, and the algorithm works like the LRU. Algorithm 1 provides the pseudo-code of the LRU-social scheme. Note that if the social requests do not appear for a while, Alg. 1 behaves like the LRU.

Algorithm 1 Pseudocode of LRU-Social

Input: request for a content; C: cache size; SP: spreading power; U: user who requested the content
Output: requested content

```
 1: if requested content is not in the cache then
 2:     fetch requested content from origin server
 3:     decrement labels
 4:     if cache is full then
 5:         evict the content with the least label
 6:     end if
 7:     if request is originated from social sharing then
 8:         set label as $SP_U \times C$
 9:     else
10:         Let $S$ be the number of contents that currently have label greater than $C$
11:         set label as $C - S$
12:     end if
13: else
14:     if label of requested content is greater than $C$ then
15:         decrement labels that are greater than $C$
16:     else
17:         decrement labels that are greater than the label of requested content
18:         Let $S$ be the number of contents that currently have label more than $C$
19:         set label as $C - S$
20:     end if
21: end if
22: return requested content
```

3.3. Estimating the spreading power

Alg. 1 needs the predicted spreading power of the corresponding user for each social request. To implement this algorithm in the real world, users' spreading power can be extracted from data minings performed by social network operators. Some social networks provide public information about users' activity that can be used to estimate users' spreading power, like Views for each post on Instagram. In this article, we use simulation of the spreading process to provide the spreading power of users to the LRU-Social algorithm. In the context of the independent interaction spreading models, each infected subscriber that generates or accesses ordinary content may share the content link with his friends and infect them, motivating them to access the content and share the corresponding link with their friends. Also, this spreading process possesses these properties: (i) once a subscriber receives a content link and possibly shares it with his friends, this subscriber rarely shares this link again in future link sharing of this content by other friends, i.e, an infected node would be recovered and would not affect other nodes, and (ii) after access to the content following a friend link sharing, the subscriber would not request to access this content again from the CDN in possible future link sharing, i.e., a recovered node will not be infected again. Therefore, we use the SIR spreading model to simulate the spreading of content link sharing in OSN. We set the infection and recovery probabilities, respectively, by $\beta = \lambda_c$ and $\gamma = 1$, where $\lambda_c = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}$ is the epidemic threshold of the network and $\langle k \rangle$ is the mean degree of the network.

Having β and γ, we can simulate the spreading process for each user in the OSN to estimate numerically how many other subscribers may request the same content if each user shares a content link.

4. Numerical study and results

We use two data sets for the OSNs. The first one is a piece of Twitter network with 4335 subscribers and 29645 directed links where the link from node i to node j shows that j follows i [22]. The second one is a piece of Facebook social network with 4039 nodes and 88234 undirected links where the link between node i and j shows that i and j are friends of each other [23]. Also, $\lambda_c^{Twitter} = 0.0371$ and $\lambda_c^{Facebook} = 0.0094$.

We consider $N = 100$ contents and generate a stream of $R = 10000$ requests. The stream of requests consists of the ordinary requests with the Zipf popularity model with $\alpha = 0.75$ and social requests. The fraction of social requests and their average intervals are changing in each simulation scenario.

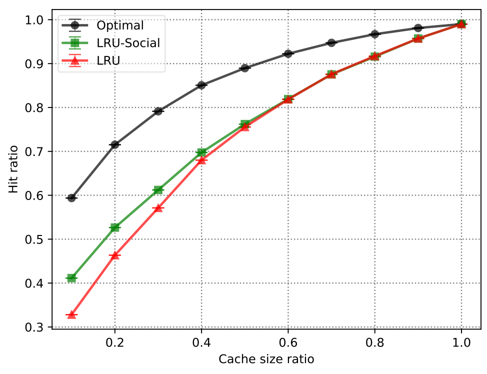
The sequence of social requests is generated by selecting a node randomly and uniformly from the OSN network. Next, we simulate the SIR spreading process to predict the spreading power of this node. We then select random content uniformly. The number of social requests for this content is equal to the corresponding subscriber's spreading power in the OSN. These social requests are then merged with the stream of ordinary requests. Finally, we use a uniform distribution with an average proportional to the cache size to simulate the intervals between two consecutive social requests.

We measure the hit ratio of the LRU, LRU-Social, and Optimal algorithms as the performance metric for different ratios of cache size than N.

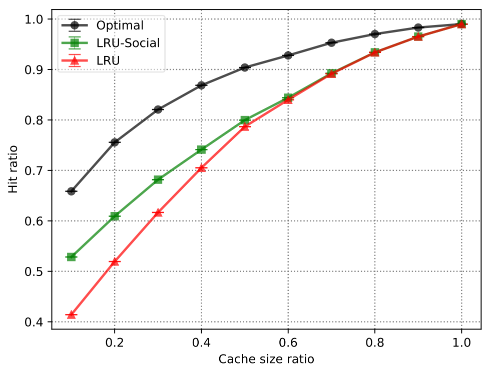
We first evaluate the performance of the LRU-Social against the LRU with different fractions of social requests. Fig. 4 (a) shows the results for a scenario when the fraction of social requests is 0.3, and the interval between consecutive social requests follows a uniform distribution with an average equal to the cache size. The result shows that the LRU-social outperforms the LRU about 5 percent, especially in small cache sizes. Note that in real scenarios, the cache size ratio is much less than one. As expected, by increasing the cache size, the corresponding average interval between two consecutive social requests increases, and the social requests are more similar to the ordinary requests. Therefore, the performance gap between the LRU-social and the LRU decreases and becomes closer to the Optimal strategy. Fig. 4 (b) shows the results for a scenario when the fraction of social requests increases to 0.5. As expected, the LRU-social hits more social requests, and the performance gap with the LRU is increased.

Next, we investigate how much the performance of the LRU-social is sensitive to the average interval between consecutive social requests. In Fig. 5, the interval between consecutive social requests follows a uniform distribution with an average of two cache-size. The result shows that the LRU-Social keeps the performance gap even in larger cache sizes. The reason is that although the occurrence probability of ordinary requests between two social requests increases in this scenario, the LRU hit a few of them.

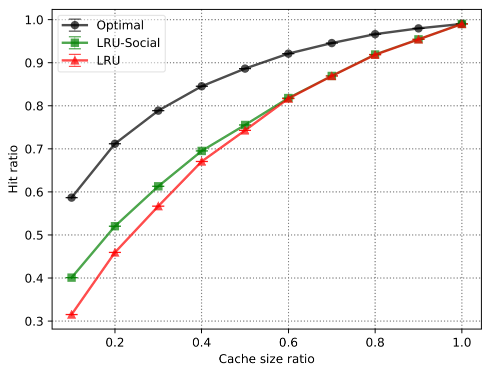
It is also interesting to figure out the performance of the LRU on the merged stream irrespective of the social or ordinary type. See Fig. 6. We expect to have a greater hit ratio as the LRU can take advantage of hitting requests with short consecutive intervals compared to the cache size. However,



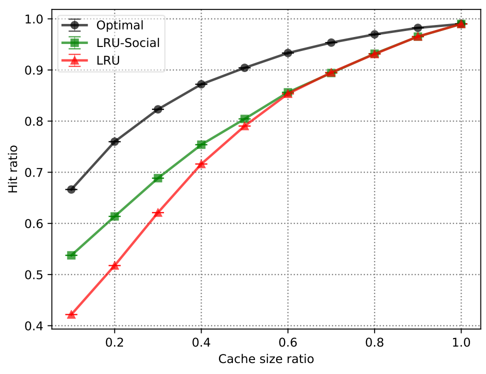
(a)



(b)

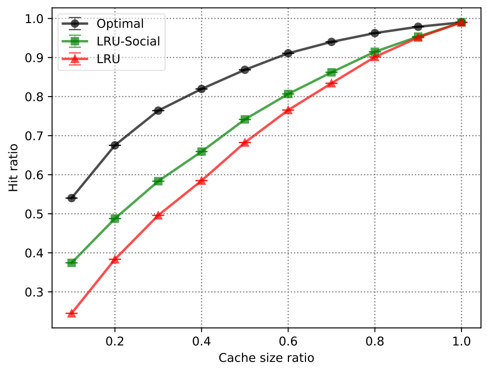


(c)

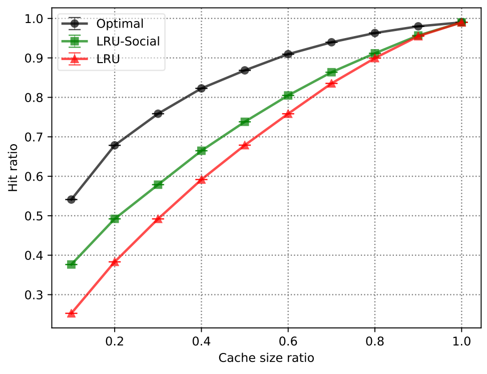


(d)

Figure 4: The hit ratio of LRU, LRU-Social, and Optimal algorithms for different values of cache size. The stream contains $R = 10000$ requests for $N = 100$ contents where the fraction of social requests is 0.3. The interval between consecutive social requests follows a uniform distribution with an average of two cache size. The spreading power of users is predicted by simulation of SIR spreading on (a, c) Twitter network and (b, d) Facebook network.



(a)



(b)

Figure 5: The hit ratio of LRU, LRU-Social, and Optimal algorithms for different values of cache size. The stream contains $R = 10000$ requests for $N = 100$ contents where the fraction of social requests is 0.3. The interval between consecutive social requests follows a uniform distribution with average equal to the cache size. The spreading power of users is predicted by simulation of SIR spreading on (a) Twitter network and (b) Facebook network.

the LRU-social outperforms this strategy as it can hit the requests with greater intervals than the cache size. This result emphasizes that characterizing and treating social requests is beneficial.

Finally, we study the impact of the content sizes heterogeneity on the algorithm performance. Content size heterogeneity decreases the performance of cache management algorithms as they need to evict more than one stale content to store a new large content. Here, we consider a scenario in which the size of 20% of the contents is twice the typical size, and the size of 10% of them is three times the typical size for the same stream of requests. The cache size is the total sum of content sizes. Fig. 7 shows the hit ratios of algorithms for this scenario compared to the scenario in which all contents have homogeneous sizes. The content sizes heterogeneity decreases the hit ratio in both LRU and LRU-social algorithms; however, the LRU-social still outperforms the LRU.

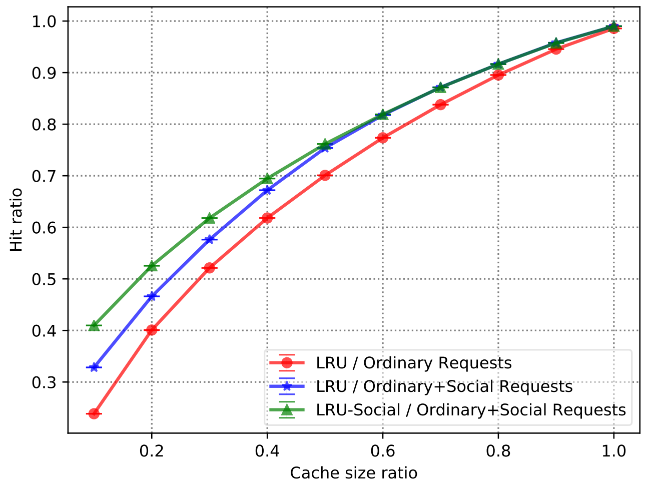


Figure 6: The hit ratio of LRU and LRU-Social for different cache size values with and without merging the requests. The first stream contains $R = 10000$ ordinary requests for $N = 100$ contents. The second stream consists of $R = 10000$ requests merging ordinary and social requests for $N = 100$ contents when the fraction of social requests is 0.3. The interval between consecutive social requests follows a uniform distribution with an average equal to the cache size. The spreading power of users is predicted by a simulation of SIR spreading on the Twitter network.

5. Conclusion

Social networks affect the pattern for content requests in CDNs by spreading the corresponding weblink. We suggested exploiting the temporal and spatial patterns in the consecutive requests originated from the social network to improve the content replacement strategy in cache management using the LRU-social. We use the SIR model to estimate the spreading power of each user on two real data sets. The consecutive stream of requests is then synthesized. We show that the LRU-social outperforms the LRU in terms of hit ratio by characterizing the social requests and treating them differently. The LRU-social keeps social contents, which may have consecutive intervals greater than the cache size. We did not find any real request stream of contents with appropriate labeling, which shows if the request is originated from link-sharing or not. We will try to gather the request stream of a CDN with labeled data to better characterize the temporal patterns in the social requests in future works.

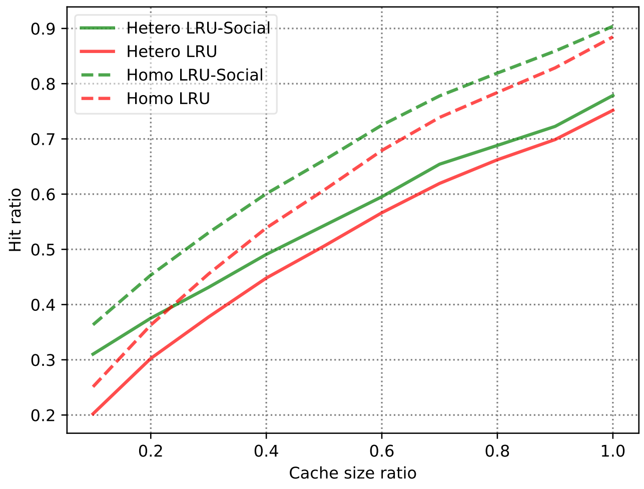


Figure 7: The hit ratio of LRU and LRU-Social for the scenario with heterogeneous content sizes compared to the homogeneous scenario. The stream contains R = 20000 requests for N = 100 contents when the fraction of social requests is 0.3, and the interval between consecutive social requests follows a uniform distribution with an average equal to the cache size. The spreading power of users is predicted by a simulation of SIR spreading on the Twitter network.

Acknowledgement

A. Ghasemi gratefully acknowledges support from the the Alexander von Humboldt Foundation for his visiting research in Germany.

References

[1] A.-M. K. Pathan, R. Buyya, A taxonomy and survey of content delivery networks, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report 4 (2007) 70.

[2] A. Vakali, G. Pallis, Content delivery networks: Status and trends, IEEE Internet Computing 7 (6) (2003) 68–74.

[3] J. Wang, A survey of web caching schemes for the internet, ACM SIGCOMM Computer Communication Review 29 (5) (1999) 36–46.

[4] S. Scellato, C. Mascolo, M. Musolesi, J. Crowcroft, Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades, in: Proceedings of the 20th international conference on World wide web, ACM, 2011, pp. 457–466.

[5] A. Brodersen, S. Scellato, M. Wattenhofer, Youtube around the world: geographic popularity of videos, in: Proceedings of the 21st international conference on World Wide Web, ACM, 2012, pp. 241–250.

[6] S. Asur, B. A. Huberman, Predicting the future with social media, in: Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01, IEEE Computer Society, 2010, pp. 492–499.

[7] C. Bird, D. Pattison, R. D'Souza, V. Filkov, P. Devanbu, Latent social structure in open source projects, in: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, ACM, 2008, pp. 24–35.

[8] D. Centola, The spread of behavior in an online social network experiment, science 329 (5996) (2010) 1194–1197.

[9] M. Salehi, R. Sharma, M. Marzolla, M. Magnani, P. Siyari, D. Montesi, Spreading processes in multilayer networks, IEEE Transactions on Network Science and Engineering 2 (2) (2015) 65–83.

[10] S. Podlipnig, L. Böszörmenyi, A survey of web cache replacement strategies, ACM Computing Surveys (CSUR) 35 (4) (2003) 374–398.

[11] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, E. A. Fox, Caching proxies: Limitations and potentials (1995).

[12] L. Maggi, L. Gkatzikis, G. Paschos, J. Leguay, Adapting caching to audience retention rate, Computer Communications 116 (2018) 159–171.

[13] S. Tarnoi, W. Kumwilaisak, V. Suppakitpaisarn, K. Fukuda, Y. Ji, Adaptive probabilistic caching technique for caching networks with dynamic content popularity, Computer Communications 139 (2019) 1–15.

[14] M. Newman, Networks, Oxford university press, 2018.

[15] S. Pei, H. A. Makse, Spreading dynamics in complex networks, Journal of Statistical Mechanics: Theory and Experiment 2013 (12) (2013) P12002.

[16] M. Granovetter, Threshold models of collective behavior, American journal of sociology 83 (6) (1978) 1420–1443.

[17] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, A. Vespignani, Epidemic processes in complex networks, Reviews of modern physics 87 (3) (2015) 925.

[18] J. Woo, H. Chen, Epidemic model for information diffusion in web forums: experiments in marketing exchange and political dialog, SpringerPlus 5 (1) (2016) 66.

[19] T. Broxton, Y. Interian, J. Vaver, M. Wattenhofer, Catching a viral video, Journal of Intelligent Information Systems 40 (2) (2013) 241–259.

[20] A. Ruhela, S. Triukose, S. Ardon, A. Bagchi, A. Mahanti, A. Seth, The scope for online social network aided caching in web cdns, in: Architectures for Networking and Communications Systems, IEEE, 2013, pp. 37–45.

[21] L. Breslau, P. Cao, L. Fan, G. Phillips, S. Shenker, et al., Web caching and zipf-like distributions: Evidence and implications, in: Ieee Infocom, Vol. 1, INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), 1999, pp. 126–134.

[22] L. Weng, F. Menczer, Y.-Y. Ahn, Virality prediction and community structure in social networks, Scientific reports 3 (2013) 2522.

[23] J. Leskovec, J. J. Mcauley, Learning to discover social circles in ego networks, in: Advances in neural information processing systems, 2012, pp. 539–547.

Appendix A. The miss probability of Optimal strategy

In this appendix we calculate the probability that the Optimal algorithm misses the second social request, *SR*, in an stream of consecutive requests like *SR, R_1, R_2, . . . , R_D, SR* where R_i shows an ordinary request. That is, there are D ordinary requests between the two social requests. Let N and C denote the total number of contents and the cache size. Note that for $D \geq C$, the LRU will miss the second social request, irrespective of the cache's current contents. However, the Optimal scheme does not necessarily miss the second social request depending on the current contents in the cache.

The Optimal algorithm misses the second social request if (i) the current cache contents does not contain the content related to R_1 and (ii) at least $C-1$ requests in the stream $R_2, \ldots, R_{\mathrm{D}}$ do request for the current cache content, i.e., at least $C-1$ of contents in the cache will be requested in the $R_2, \ldots, R_D$. These conditions ensure that upon the arrival or R_1, the corresponding content to the second social request is the farthest one in the future, which leads to the eviction and miss of the second social request.

Therefore, the probability of missing the second social request depends on the current cache content and the requested stream. We could compute the corresponding probability by assuming uniform distribution for the current cache content and the request stream.

We first enumerate the total number of possible combinations for the current cache content and the possible combinations for the request stream $R_1, R_2, \ldots, R_{\mathrm{D}}$. This is equal to possible combinations of selecting C contents from the total N contents, $\binom{N}{C}$, multiplied by the possible combinations of selecting D contents from the total $N-1$ non-social contents $\binom{N-1}{D}$, considering their possible $D!$ permutations. Therefore, the total number of possible combinations counts for the current cache content, and the offered request stream is $\binom{N}{C}\binom{N-1}{D}D!$.

Next, we enumerate the desired possible number of combinations in which conditions (i) and (ii) will meet, and the Optimal algorithm evicts the second social request. We enumerate two possible cases depending on whether the current cache does (N_1) or does not (N_2) contain the social request content. The total number of desired combinations then given by $N = N_1 + N_2$.

Let consider N_1. The possible number of combinations for the current cache content is given by selecting $C-1$ content out of the remaining $N-1$ ones. Next, the possible number of combinations for selecting R_1 to meet condition (i) is given by $N-C$. The possible number of combinations for $R_2, \ldots, R_D$ that the corresponding content does contain in the cache is $\binom{C-1}{C-1}$. The number of combinations for the remaining requests in $R_2, \ldots, R_D$ which their contents does not contain in the cache is $\binom{N-(C+1)}{D-C}$. Finally, the number of permutations for the D ordinary request is $D!$. Thus, we get

$$N_1 = \binom{N-1}{C-1}(N-C)\binom{C-1}{C-1}\binom{N-(C+1)}{D-C}(D-1)!. \tag{A.1}$$

Using the same reasoning we can enumerate the possible number of combinations for N_2. We can select the C ordinary requests in $\binom{N-1}{C}$ combinations and selecting R_1 in $N-C-1$ ways. The number 365 of combinations for the remaining requests in $R_2, \ldots, R_D$ which their contents does not contain in the cache is $\binom{C}{C-1}\binom{N-(C+2)}{D-C} + \binom{C}{C}\binom{N-(C+2)}{D-C-1}$. Taking into account the possible $D!$ permutations we have

$$\begin{aligned} N2 = &\binom{N-1}{C}(N-C-1)\times \\ &\left[\binom{C}{C-1}\binom{N-(C+2)}{D-C} + \binom{C}{C}\binom{N-(C+2)}{D-C-1}\right]\times \\ &(D-1)! \end{aligned} \tag{A.2}$$

Considering the total number of possible combinations and the combinations in which the Optimal algorithm misses the second social request, we conclude:

$$\Pr(\text{miss}) = \frac{N_1 + N_2}{\binom{N}{C}\binom{N-1}{D}D!}. \tag{A.3}$$