Chapter 18
Delivering Social Multimedia Content with Scalability

Irene Kilanioti and George A. Papadopoulos

18.1 Introduction

CDNs aim at overcoming Internet issues and ensuring smooth and transparent content delivery. They principally replicate content in locations as near as possible to the user that is bound to consume it. CDNs handle altogether the major issues of (i) the most efficient placement of surrogate servers in terms of high performance and less infrastructure cost; (ii) the best content diffusion placement, namely the decision of which content will be copied in the surrogate servers and to what extent; and (iii) the temporal diffusion, related to the most efficient timing of the content placement [15]. They are, however, very dissimilar in terms of the services provided and their geographic coverage. The optimization of their overall efficiency, as far as user is concerned, is practically achieved with the automatic detection of the medium (either computer or mobile -smartphone/tablet), optimized management of the browser cache, server load-balancing, the consideration of specific nature of the content of the media provider (video on demand, live videos, geo-blocked content, etc.,) or features of certain operators, such as real-time compression, session management, etc.

Utilization of CDNs is likely to have profound effects on large data download through enhanced performance, scalability, and cost reduction. Extended use of OSNs, and the increasing popularity of streaming media are the factors that drive the HTTP traffic growth [4]. The amount of traffic generated on a daily basis by online multimedia streaming providers is multiplied by the transmission over OSNs (with more than 400 tweets per minute including a YouTube video link [3] being

I. Kilanioti (✉) · G.A. Papadopoulos (✉)
Department of Computer Science, University of Cyprus, 1 University Avenue,
P.O. Box 20537, 2109 Nicosia, Cyprus
e-mail: ekoila01@cs.ucy.ac.cy

G.A. Papadopoulos
e-mail: george@cs.ucy.ac.cy

F. Pop et al. (eds.), *Resource Management for Big Data Platforms*,
Computer Communications and Networks, DOI 10.1007/978-3-319-44881-7_18

published per minute). Subsequently, CDN users can benefit from an incorporated mechanism of social awareness over the CDN infrastructure. In [15, 16] Kilanioti and Papadopoulos introduce a dynamic mechanism of preactive copying of content to an existing validated CDN simulation tool and propose an efficient copying policy based on prediction of demand on OSNs along with its variations.

18.1.1 A Toy Example of Our Approach

Let us consider Bob, located in London and assigned to the London CDN servers of an OSN service. Most of Bob's social friends are geographically close to him, but he also has a few friends in Europe and Australia assigned to their nearest servers. Bob logs into the OSN and posts a video that he wants to share. Pushing the video content to all other geographically distributed servers immediately before any requests occur would be the naive way to ensure that this content is as close as possible to all users. Aggregated over all users, pushing can lead to traffic congestion, and users would experience latency in accessing the content, which, moreover, could not be consumed at all. The problem of caching would be intensified when Alice, the only friend of Bob in Athens, would be interested in that content, and with many such Alices in various places.

Rather than pushing data to all surrogates, we can proactively distribute it only to friends of Bob likely to consume it and only at the time window that signifies a non-peak time for the upload in London area and a non-peak-time for the download in Athens area, thus taking advantage of the timezone differences of our geo-diverse system. The content will be copied only under certain conditions (content with high viewership within the media service, copied to geographically close timezones where the user has mutual friends with high influence impact). This would contribute to smaller response times for the content to be consumed (for the users) and lower bandwidth costs (for the OSN provider).

18.1.2 Contributions

In this work we modify the Social Prefetcher algorithm [15, 16] to incorporate best performing caching mechanisms. We conduct experiments over a large corpus of YouTube videos and use Twitter, where information propagates via retweeting across multiple hops in the network [19]. Social cascades are directly analyzed, as the real dataset of User Generated Content (UGC) used includes multimedia links over the OSN. The Twitter dataset contains geographic locations, follower lists and tweets for 37 million users, spreading of more than one million YouTube videos over this network, a corpus of more than 2 billions messages and approximately 1.3 million single messages with an extracted video URL. The wide popularity and massive user base of YouTube and Twitter allow us to obtain safe insights regarding user navigation

behavior on other similar media and microblogging platforms, respectively. The implemented variations are incorporated in a validated simulator for CDNs, and restrictions of the CDN infrastructure (storage issues, network topology) are taken into account.

The present work goes beyond the Social Prefetcher algorithm [16] in terms of performance. The latter surpasses performance improvement of similar works in pull-based methods, that are employed by most CDNs, whereas moreover uses more refined topology of data centers and does not neglect storage issues. Storage costs are still a significant challenge despite the proliferation of cloud computing. In this work we examine which caching schemes in the surrogate server affect the CDN metrics the most. We further enhance the performance of Social Prefetcher algorithm, an optimization analysis of which is presented by the authors in [15]. The findings of present work can be exploited for future policies complementary to existing CDN solutions or incorporated to OSN providers mechanisms, to handle larger scale data.

The rest of this paper is structured as follows: Sect. 18.2 reviews previous related work. Section 18.3 formally describes the addressed problem. The proposed algorithm is described in Sect. 18.4. Section 18.5 gives an outline of the methodology, along with the preparation of the employed datasets. Our main findings are presented in Sect. 18.6. Section 18.7 concludes the paper and discusses directions for future work.

18.2 Related Work

Systems that leverage information from OSNs with various research goals, such as the decision for copying content or improvement of policy for temporary caching, include [20, 21, 23]. Traverso et al. [23] improve QoS by exploiting time differences among sites and the access patterns that users follow. Sastry et al. [20] analyze social cascades and access to social profiles via a third-party page.

As long as the behavior of users in different media services is concerned, the traffic on YouTube is described in several studies [4, 10–12, 18], with emphasis on the characteristics of media content, such as file size, bitrate, usage patterns, and popularity. In [11], the authors study the YouTube workload to discover that there are many similarities between traditional Web and media streaming workloads. The authors in [7] find a strong correlation among YouTube videos, because the links to related videos generated by uploaders depict small-world characteristics. In [9] the authors analyze how the popularity of individual YouTube videos evolves.

Authors in [16] extend the Social Prefetcher algorithm proposed in [15] to include information about peak-time of various timezones of a geo-diverse system, as well as contextual information about the viewership of video content within the media service. The basic algorithm gives a near-optimal solution to the problem of content delivery and addresses memory usage issues related to the very large graph dataset accommodated. The suggested mechanism added to a CDN simulator overcomes the testing limitations of other existing CDN platforms, such as the blackbox treatment of CDN policies or the need for the participation of third users.

18.3 Problem Description

We aim at mitigating the inherent Internet performance issues by improving the CDN infrastructure mechanisms. We aim at the reduction of the response time for the user, increase of the hit ratio of our request, as well as restriction of the cost of copying from the origin server to surrogate servers. We consider the network topology, the server location, and restrictions in the cache capacity of the server. Taking as input data from OSNs and actions of users over them, we want to recognize objects that will eventually be popular in the realm of the OSN platform.

We search a policy such that given a graph $G(V, E)$, a set of R regions, where the nodes of the social network are distributed, and the posts P of the nodes, it will recognize the set of objects O that will be popular only in a subset of the regions (Table 18.1), where the content is likely to be copied. The policy is represented by the function $Put(n_i, Predict(G, P, R, O))$, which takes as input a surrogate server $n_i \in N$ and the results of function *Predict* (set of g objects that will be globally popular and λ objects that will be locally popular), such that

$$\frac{Q_{hit}}{Q_{total}} \tag{18.1}$$

is maximum, whereas constraint

$$\sum_{\forall i \in O} S_i f_{ik} \leq C_k \tag{18.2}$$

is fulfilled, where:

$$f_{ik} = \begin{cases} 1 & \text{if object } i \text{ exists in the cache of surrogate server } k \\ 0 & \text{if object does not exist} \end{cases} \tag{18.3}$$

Function $Put(n_i, Predict(G, P, R, O))$ returns the set of objects $o \in O$ that have to be placed in surrogate server $n_i \in N$.

18.4 Proposed Dynamic Policy

The proposed algorithm encompasses an algorithm for each new request arriving in the CDN and an algorithm for each new object in the surrogate server. Internally, the module communicates with the module processing the requests and each addressed server separately (Fig. 18.1).

Table 18.1 Notation overview

$G(V, E)$	Graph representing the social network
$V = \{V_1, \ldots, V_n\}$	Nodes representing the social network users
$E = \{E_{11}, \ldots, E_{1n}, \ldots, E_{nn}\}$	Edges representing the social network connections, where E_{ij} stands for friendship between i and j
$R = \{r_1, r_2, \ldots, r_\tau\}$	Regions set
$N = \{n_1, n_2, \ldots, n_u\}$	The surrogate servers set. Every surrogate server belongs to a region r_i
$C_i, i \in N$	Capacity of surrogate server i in bytes
$O = \{o_1, o_2, \ldots, o_w\}$	Objects set (videos), denoting the objects users can ask for and share
$S_i, o_i \in O$	Size of object i in bytes
i_κ	Object accessed at the κ-th iteration, κ: the counter maintained and incremented each time there is a request for an object
ΔT_{i_κ}	Number of accesses since the last time object i was accessed
Π_i	Popularity of object i, $i \in O$
$q_i = \{t, V_\psi, o_x\}, 1 < x < w, 1 < \psi < n$	Request i consists of a timestamp, the id of the user that asked for the object, and the object id
$P = \{p_{12}, p_{13}, \ldots, p_{nw}\}$	User posts in the social network, where p_{ij} denotes that node i has shared object j in the social network
$pts_i, pte_i, 1 < i < \tau$	peak time start and peak time end for each region in secs
$Q = \{q_1, q_2, \ldots, q_\zeta\}$	Object requests from page containing the media objects, where q_i denotes a request for an object of set O
Q_{hit}, Q_{total}	Number of requests served from surrogate servers of the region of the user/total number of requests
$X, Y \in R$	Closest timezones with mutual followers/with highest centrality metric (HITS) values

18.4.1 For Every New Request in the CDN

The main idea is to check whether specific time has passed after the start of the cascade, and then define to what extent the object will be copied. Initially, we check whether it is the first appearance of the object. The variable *o.timestamp* depicts the timestamp of the last appearance of the object in a request and helps in calculating the timer related to the duration of the cascade. If it is the first appearance of the object, the timer for the object cascade is initialized and *o.timestamp* takes the value of the timestamp of the request. If the cascade is not yet complete (its timer has not surpassed

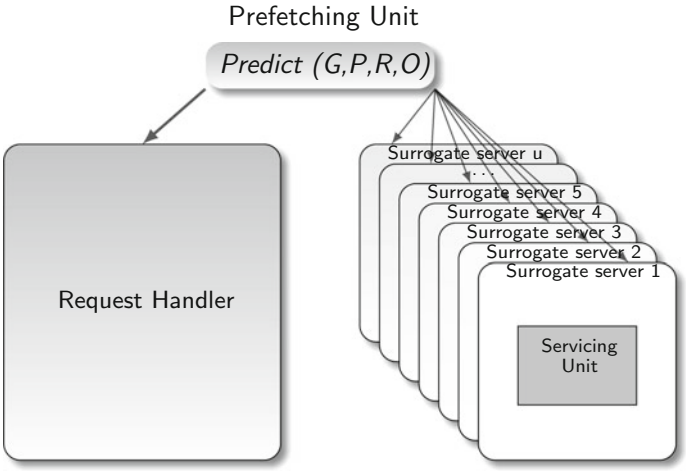

Fig. 18.1 The prefetching unit

a threshold), we check the importance of the user applying the Hubs Authorities (HITS) algorithm and checking its authority score, as well as the viewership of the object in the media service platform (Fig. 18.2).

For users with a high authority score, we copy the object to all surrogate servers of the user's timezone and to the surrogate servers serving the timezones of all

```
1:  if o.timestamp == 0 then
2:      o.timer = 0;
3:      o.timestamp = request_timestamp;
4:  else if o.timestamp != 0 then
5:      o.timer = o.timer + (request_timestamp - o.timestamp);
6:      o.timestamp = request_timestamp;
7:  end if
8:  if o.timer > time_threshold then
9:      o.timer = 0;
10:     o.timestamp = 0;
11: else if o.timer < time_threshold and user.authority_score > authority_threshold then
12:     copy object o to surrogate that serves user's V_i timezone;
13:     for all user V_y that follows user V_i do
14:         find surrogate server n_j that serves V_y's timezone;
15:         copy object o to n_j;
16:     end for
17: else if o.timer < time_threshold and o.Π_i > Π_i_threshold then
18:     copy object o to surrogates n_j that Subpolicy I decides;
19: end if
```

Fig. 18.2 Algorithm for every new request (*timestamp*, V_i, o) in the CDN

1: find X timezones where (user V_i has mutual followers **and** they are closer to user's V_i timezone);
2: find the $Y \subseteq X$ that (belong to X **and** have the highest HITS score);
3: **for all** timezones that belong to Y **do**
4: find surrogate server n_j that serves timezone;
5: copy object o to n_j;
6: **end for**

Fig. 18.3 Subpolicy I

followers of the user (global prefetching). Otherwise, selective copying includes only the surrogates that the subpolicy decides (local prefetching).

Centrality is measured with the HITS algorithm [17], a link analysis algorithm that rates web pages. Twitter uses a HITS style algorithm to suggest to users which accounts to follow [13], as well. A so-called good hub represents a page that points to many other pages, whereas a good authority represents a page that is linked by many different hubs. Memory usage issues for the very large graph dataset accommodated led to calculation of HITS with the MapReduce technique. Subpolicy (Fig. 18.3) checks the X closest timezones where a user has mutual friends and out of them, the Y with the highest value of the centrality metric as an average. Highest value of the metric means that the object is likely to be asked for more times. Copying is performed to the surrogate servers that serve the above timezones.

18.4.2 For Every New Object in the Surrogate Server

Surrogate servers keep replicas of the web objects on behalf of content providers. In the case that the new object does not fit in the surrogate server's cache, we define the *time_threshold* as the parameter for the duration that an object remains cached.

We check for items that have remained cached for a period longer than the *time_threshold* and we delete those with the largest timestamp in the cascade. In case there exist no such objects or all objects have the same timestamp, we apply various policies for the removal of objects

- *Least Recently Used (LRU)* In the most straightforward extension of LRU for handling nonhomogeneous-sized objects we prune the least recently used items first. The algorithm keeps track of what was used when, to make sure that it discards the least recently used item.
- *Least Frequently Used (LFU)* The algorithm keeps track of the number of times an object is referenced in memory. When the cache is full and more room is required, it purges the item with the lowest reference frequency. We simply employ an LFU algorithm by assigning a counter to every object that is loaded into the cache. Each time a reference is made to that object the counter is increased by one. When the cache reaches capacity and a new object arrives, the system will search for the object with the lowest counter and remove it from the cache.

Table 18.2 Applied caching schemes

Name	Primary key	Secondary key
LRU	Time since last access	
LFU	Frequency of access	
SIZE	Size	Time since last access

- *Size-adjusted LRU (SIZE)* The optimization model devised in [1] to generalize LRU is approximately solved by a simple heuristic and the policy is called Size-adjusted LRU or SIZE. In this policy the objects are removed in order of size with the largest object removed first. In case two objects have the same size, objects longer cached since their last access are removed first. Objects in the cache are reindexed in order of increasing values of $S_i \cdot \Delta T_{i_\kappa}$ and highest index objects are greedily selected and purged from the cache until the new object fits in.

Varying algorithms depending on the caching scheme used (Table 18.2) are depicted in Figs. 18.4, 18.5 and 18.6. The heuristics applied in our approach are based on the following observations [15]: Users are more influenced by geographically close friends, and moreover by mutual followers, with the most popular users acting as authorities. Social cascades have a short duration, and in our prefetching algorithm we take into account the observation that the majority of cascades end within 24 h. However, we introduce a varying time threshold for the cascade effect and the time that an object remains in cache. Values given in the time threshold variable also include 48 h, as well as threshold covering the entire percentage of requests.

```
1:  if o.size + current_cache_size ≤ total_cache_size then
2:      copy object o to cache of surrogate n_k;
3:  else if o.size + current_cache_size > total_cache_size then
4:      while o.size + current_cache_size > total_cache_size do
5:          for all object o' in current_cache do
6:              if (current_timestamp - o'.timestamp) + o'.timer > time_threshold then
7:                  copy o' in CandidateList;
8:              end if
9:              if CandidateList.size>0 and CandidateList.size != total_cache_size then
10:                 find o' that o'.timestamp is maximum and delete it;
11:             else if CandidateList.size==0 or CandidateList.size==total_cache_size then
12:                 use LRU to delete any object o ∈ O;
13:             end if
14:         end for
15:     end while
16:     put object o to cache of surrogate n_k;
17: end if
```

Fig. 18.4 VariationA—Algorithm for every new object o in the surrogate server n_k

```
1: if o.size + current_cache_size ≤ total_cache_size then
2:     copy object o to cache of surrogate n_k;
3: else if o.size + current_cache_size > total_cache_size then
4:     while o.size + current_cache_size > total_cache_size do
5:         for all object o' in current_cache do
6:             if (current_timestamp - o'.timestamp) + o'.timer > time_threshold then
7:                 copy o' in CandidateList;
8:             end if
9:             if CandidateList.size>0 and CandidateList.size != total_cache_size then
10:                find o' that o'.timestamp is maximum and delete it;
11:            else if CandidateList.size==0 or CandidateList.size==total_cache_size then
12:                use LFU to delete any object o ∈ O;
13:            end if
14:        end for
15:    end while
16:    put object o to cache of surrogate n_k;
17: end if
```

Fig. 18.5 VariationB—Algorithm for every new object o in the surrogate server n_k

```
1: if o.size + current_cache_size ≤ total_cache_size then
2:     copy object o to cache of surrogate n_k;
3: else if o.size + current_cache_size > total_cache_size then
4:     while o.size + current_cache_size > total_cache_size do
5:         for all object o' in current_cache do
6:             if (current_timestamp - o'.timestamp) + o'.timer > time_threshold then
7:                 copy o' in CandidateList;
8:             end if
9:             if CandidateList.size>0 and CandidateList.size != total_cache_size then
10:                find o' that o'.timestamp is maximum and delete it;
11:            else if CandidateList.size==0 or CandidateList.size==total_cache_size then
12:                use SIZE to delete any object o ∈ O;
13:            end if
14:        end for
15:    end while
16:    put object o to cache of surrogate n_k;
17: end if
```

Fig. 18.6 VariationC—Algorithm for every new object o in the surrogate server n_k

Prinicipally we check whether specific time has passed after the start of cascade and, only in the case that the cascade has not ended, define to what extent the object will be copied (*algorithm for every new request*). This check is also performed in *algorithm for every new object*, where we define the *time_threshold*. The latter roughly expresses the average cascade duration, as it defines the duration that an object remains cached.

18.5 Experimental Evaluation

For the experimental evaluation, we used a stand-alone CDN simulator for CDNs. The configuration of the simulation values is depicted in Table 18.3. For the extraction of a reliable output, we had to conclude to a specific network topology, as well as make assumptions regarding the input dataset. The simulator takes as input files describing the underlying CDN and the traffic in the network, and provides an output of statistical results, discussed in the next Section.

- *Network Topology* There follows a short description of the process defining the nodes in the topology. These nodes represent the surrogate servers, the origin server, and the users requesting the objects (Fig. 18.7). For an in-depth analysis you can refer to [15]. To simulate our policy and place the servers in a real geographical position, we used the geographical distribution of the Limelight network [14]. For the smooth operation of the simulator the number of surrogate servers was reduced by a ratio of 10 %, to ultimately include 423 servers (Table 18.4). Depending on the closer distance between the surrogate region defined by Limelight and each of the timezones defined by Twitter (20 Limelight regions, 142 Twitter timezones), we decided where the requests from each timezone will be redirected. The population of each timezone was also taken into consideration. The INET generator [4] allowed us to create an AS-level representation of the network topology. Topology coordinates were converted to geographical coordinates with the NetGeo tool from CAIDA [5], a tool that maps IP addresses and Autonomous System (AS) coordinates to geographical coordinates [22], and surrogate servers were assigned to topology nodes.
  After grouping users per timezone (due to the limitations the large dataset imposes), each group of users was placed in a topology node. We placed the user groups in the nodes closer to those comprising the servers that serve the respective timezone requests, contributing this way to a realistic network depiction.
- *Number of Requests* 1 million requests were considered sufficient, with the number of objects being the dominant factor increasing the memory use of the simulation tool. Also similar concept approaches use similar number of requests ([23] on a daily basis and [21]), and same number of distinct videos for generation of requests.

Table 18.3 Simulation characteristics

Number of nodes in the topology	3500
Redirection policy	Cooperative environment (closest surrogate)
Number of origin servers	1
Number of surrogate servers	423
Number of user groups	162
Bandwidth	100 Mbit/s

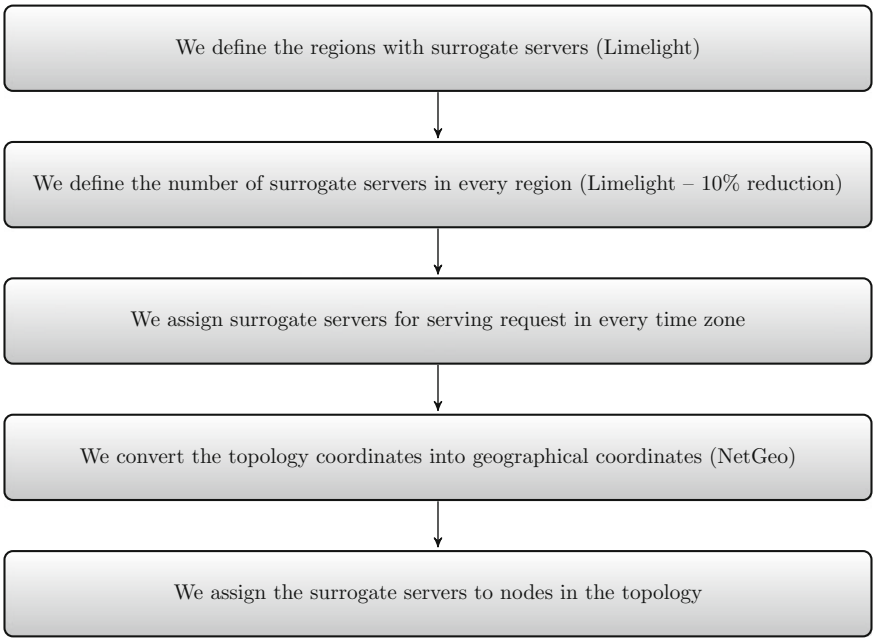

Fig. 18.7 Methodology followed

Table 18.4 Distribution of servers over the world for the experimental evaluation

City	Servers	City	Servers
Washington DC	55	Toronto	12
New York	43	Amsterdam	20
Atlanta	11	London	30
Miami	11	Frankfurt	31
Chicago	37	Paris	12
Dallas	19	Moscow	10
Los Angeles	52	Hong Kong	8
San Jose	37	Tokyo	12
Seattle	15	Changi	5
Phoenix	3	Sydney	1

- *Cache Size* The requests generated from the generator follow a long-tail distribution, thus 15 % of the whole catalog size was considered to be sufficient.
- *Threshold Values* Experimenting was conducted for time thresholds of 24 h and 48 h, as well as for the time threshold that covered all the requests. The threshold for media service viewership was set at 402.408 (average media viewership in the dataset). The authority threshold score was tested for various values (0.006/0.02/0.04).

18.6 Main Findings

The statistic reports produced by the simulator are used to evaluate the proposed policy. There follows a short explanation of the metrics used in our experiments for the extraction of statistical results.

18.6.1 Client Side Metrics

They refer to activities of clients, i.e., the requests for objects.

- *Mean Response Time* indicates how fast a client is satisfied. It is defined as

$$\frac{\sum_{i=0}^{M-1} t_i}{M}$$

where M is the number of satisfied requests and t_i is the response time of the ith request. It starts at the timestamp when the request begins and ends at the timestamp when the connection closes.

18.6.2 Surrogate Side Metrics

They are focused on the operations of the surrogate servers.

- *Hit Ratio*: is the percentage of the client-to-CDN requests resulting in a cache hit. High values indicate high quality content placement in the surrogate servers.
- *Byte Hit Ratio*: is the hit ratio expressed in bytes, counting the corresponding bytes of the requests. High values indicate optimized space usage and lower network traffic.

18.6.3 Network Statistics Metrics

They run on top of TCP/IP and concern the entire network topology.

- *Active Servers* refers to the surrogate servers being active serving clients.
- *Mean Utility of the Surrogate Servers* is a value that expresses the relation between the number of bytes of the served content against the number of bytes of the pulled content (from the origin server or other surrogate servers). It is bounded to the range [0, 1] and provides an indication about the CDN performance. High net utility values indicate good content outsourcing policy and improved mean response times for the clients.

We conducted a multitude of experiments (55 for each caching scheme and time threshold combination). Table 18.5 presents the average values of four parameters for six cases of testing. The lowest mean response times appear for the cases of the time threshold covering all requests for all caching schemes. We observe that LFU scheme outperforms LRU and SIZE in terms of mean response times and hit ratios achieved.

- *Hit Ratio* To begin with, Fig. 18.8 illustrates how the hit ratio of the requests is affected by modifying the number of timezones with highest centrality metric examined. The caching scheme of LFU appears to perform better than the LRU scheme. LRU and LFU offer comparable results, whereas they both outperform SIZE. We come to the conclusion that there is a realistic room for performance improvement by implementing various web caching characteristics in a CDN infrastructure, even though the social cascading mechanisms have already been activated to improve its performance.
- *Mean Utility of the Surrogate Servers* For a fixed number of 10 closest timezones with mutual followers LRU scheme appears to depict the highest mean utility of the surrogate servers, followed by LFU and SIZE (Fig. 18.9).

Table 18.5 Average metric values for $X = 10$ timezones of close mutual friends

	Mean response time (Avg, 10^-2 s)	Hit ratio (Avg, %)	Active servers	Mean utility (Avg, %)
LFU—24-h	1.1383	32.81	326	96.01
LFU—48-h	1.1352	33.08	325	96.01
LFU—all-h	1.1112	34.69	324	96.01
SIZE—24-h	1.1541	32.10	327	95.94
SIZE—48-h	1.146076	32.03	326	95.98
SIZE—all-h	1.1274	33.17	326	96.00
LRU—24-h	1.1412	32.12	326	95.99
LRU—48-h	1.1377	32.42	325	96.02
LRU—all-h	1.1181	34.16	325	96.04

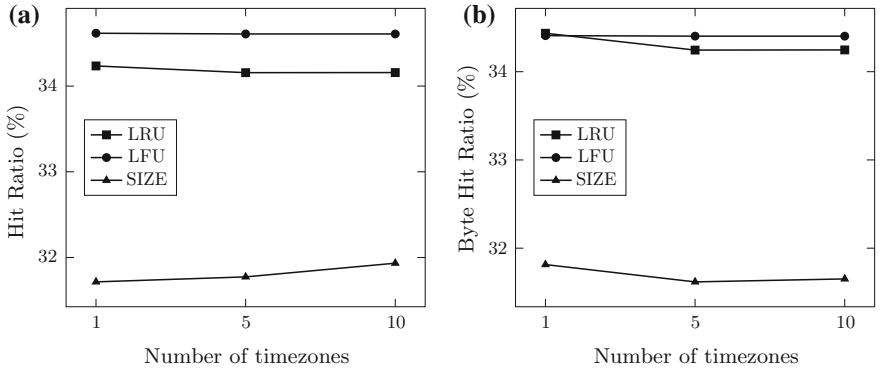

Fig. 18.8 Effect of timezones used as Y on **a** Hit Ratio and **b** Byte Hit Ratio for $X = 10$ closest timezones with mutual followers for LRU, LFU and SIZE

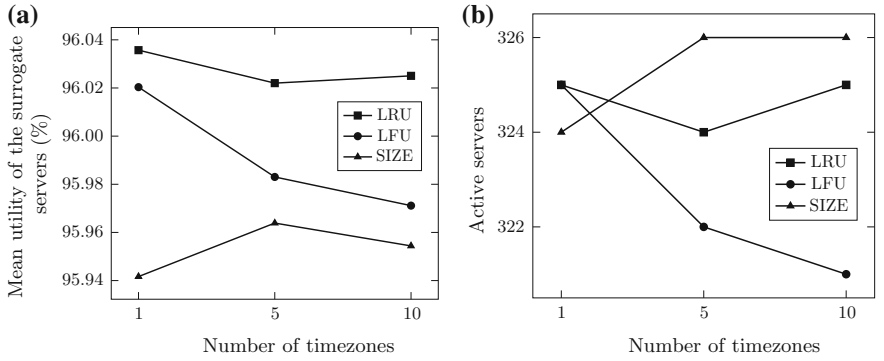

Fig. 18.9 Effect of timezones used as Y on **a** Mean Utility of the Surrogate Servers and **b** Active Servers for $X = 10$ closest timezones with mutual followers for LRU, LFU, and SIZE

- *Active Servers* For a fixed number of 10 closest timezones with mutual followers LFU appears to use less active servers after the first timezone of highest centrality in the scenario of time threshold covering all the requests. SIZE depicts higher values of active servers for the cases of 5 and 10 timezones examined (Fig. 18.9).
- *Mean Response Time* For the most representative case of all requests for LRU and LFU schemes, the trade-off between the reduction of the response time and the cost of copying in servers is expressed with a decrease of the mean response time as the timezones increase, and a point after which the mean response time starts to increase again (Figs. 18.10 and 18.11). This decrease in the mean response time occurs with approximately five timezones out of the 10 used for LRU scheme (for a fixed number of closest timezones with mutual followers), and with seven timezones for LFU. After this point the slight increase in the mean response time is attributed to the delay for copying content to surrogate servers. The cost for every copy is related to the number of hops among the client asking for it and

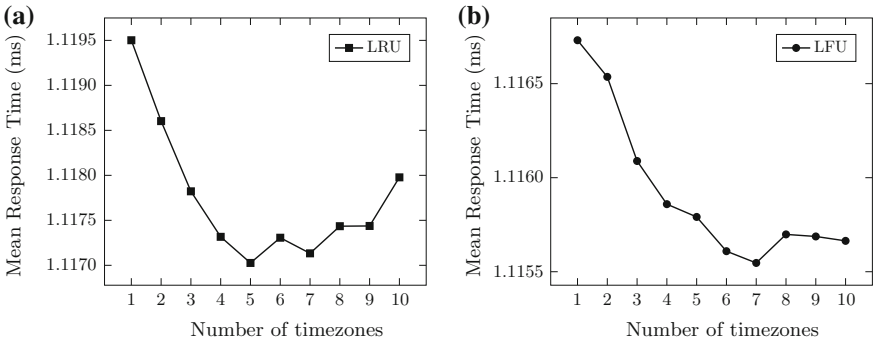

Fig. 18.10 Effect of timezones used as Y on Mean Response Time for $X = 10$ closest timezones with mutual followers for **a** LRU and **b** LFU

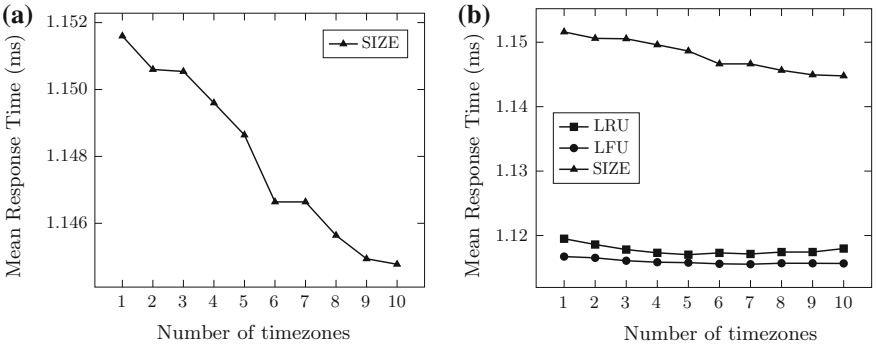

Fig. 18.11 Effect of timezones used as Y on Mean Response Time for $X = 10$ closest timezones with mutual followers for **a** SIZE and **b** all caching schemes

the server where copying is likely to be made, according to the *Put* function. We observe that SIZE depicts a poor performance, since it does not take advantage of the frequency skew.

18.7 Conclusions

CDN infrastructures rapidly deliver multimedia content cached on dispersed geographical servers to Web browsers worldwide. The growing demands for quick and scalable delivery, also due to HTTP traffic increase, can be satisfied with efficient management of the content replicated in CDNs. Specifically, we need Web data caching techniques and mechanisms on CDNs, as well as policies recognizing the patterns of social diffusion of content, to ensure satisfying performance in a constantly changing environment of continuing data volume growth.

In the present work, we further extended a dynamic policy of OSN content prefetching implemented with temporal and other contextual parameters, to depict how various caching schemes can affect the content delivery infrastructure. Bandwidth-intensive multimedia delivery over a CDN infrastructure is experimentally evaluated with realistic workloads, that many works in the related literature lack. While recognizing that we used one media service and one OSN platform for our experimentation, we believe that our results are generally applicable, with a potentially high impact for large-scale systems where traffic is generated by online social services and microblogging platforms. We aim to generalize our proposed policies in the future, to deal with multiple OSN platforms, as well as mobile CDN providers.

Acknowledgments For the development of algorithms and conducting of the accompanying experiments, the cloud infrastructure of the Department of Computer Science of the University of Cyprus, as well as Amazon Web Services, were used.

References

1. Aggarwal, C., Wolf, J.L., Yu, P.S.: Caching on the World Wide Web. IEEE Trans. Knowl. Data Eng. **11**(1), 94–107 (1999). doi:10.1109/69.755618
2. Bakshy, E., Rosenn, I., Marlow, C., Adamic, L.A.: The role of social networks in information diffusion. In: Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, 16–20 April 2012, pp. 519–528 (2012). doi:10.1145/2187836.2187907
3. Brodersen, A., Scellato, S., Wattenhofer, M.: YouTube around the world: geographic popularity of videos. In: Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, 16–20 April 2012, pp. 241–250 (2012). doi:10.1145/2187836.2187870
4. Cha, M., Kwak, H., Rodriguez, P., Ahn, Y., Moon, S.B.: I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system. IN: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC 2007, San Diego, California, USA, 24–26 Oct 2007, pp. 1–14 (2007). doi:10.1145/1298306.1298309
5. Center for Applied Internet Data Analysis. https://www.caida.org. Accessed 30 Jun 2016
6. Chard, K., Caton, S., Rana, O., Bubendorfer, K.: Social Cloud: cloud computing in social networks. In: Proceedings of the 3rd IEEE International Conference on Cloud Computing, CLOUD 2010, Miami, FL, USA, 5–10 July 2010, pp. 99–106 (2010). doi:10.1109/CLOUD.2010.28
7. Cheng, X., Dale, C., Liu, J.: Statistics and social network of YouTube videos. In: Proceedings of the 16th International Workshop on Quality of Service, IWQoS 2008, University of Twente, Enskede, The Netherlands, 2–4 June 2008, pp. 229–238 (2008). doi:10.1109/IWQOS.2008.32
8. Easley, D.A., Kleinberg, J.M.: Networks, Crowds, and Markets—Reasoning About a Highly Connected World. Cambridge University Press (2010)
9. Figueiredo, F., Benevenuto, F., Almeida, J.M.: The tube over time: characterizing popularity growth of YouTube videos. iN: Proceedings of the 4th International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, 9–12 Feb 2011, pp. 745–754 (2011). doi:10.1145/1935826.1935925
10. Finamore, A., Mellia, M., Munafò, M.M., Torres, R., Rao, S.G.: YouTube everywhere: impact of device and infrastructure synergies on user experience. In: Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement, IMC 2011, Berlin, Germany, 2–4 Nov 2011, pp. 345–360 (2011). doi:10.1145/2068816.2068849
11. Gill, P., Arlitt, M., Li, Z., Mahanti, A.: YouTube traffic characterization: a view from the edge. In: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC

2007, San Diego, California, USA, 24–26 Oct 2007, pp. 15–28 (2007). doi:10.1145/1298306.1298310
12. Gill, P., Arlitt, M., Li, Z., Mahanti, A.: Characterizing user sessions on YouTube. In: Proceedings of the SPIE Multimedia Computing and Networking Conference, MCN 2008, San Jose, California, USA, 30–31 Jan 2008, pp. 6818060–6818068 (2008). doi:10.1117/12.775130
13. Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., Zadeh, R.: WTF: the who to follow service at Twitter. In: Proceedings of the 22nd International World Wide Web Conference, WWW 2013, Rio de Janeiro, Brazil, 13–17 May 2013, pp. 505–514 (2013). doi:10.1145/2488388.2488433
14. Huang, C., Wang, A., Li, J., Ross, K.W.: Measuring and evaluating large-scale CDNs. In: Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement, IMC 2008, Vouliagmeni, Greece, 20–22 Oct 2008, pp. 15–29 (2008)
15. Kilanioti, I.: Improving multimedia content delivery via augmentation with social information. The Social Prefetcher approach. IEEE Trans. Multimedia **17**(9), 1460–1470 (2015). doi:10.1109/TMM.2015.2459658
16. Kilanioti, I., Papadopoulos, G.A.: Socially-aware multimedia content delivery for the cloud. In: Proceedings of the 8th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2015, Limassol, Cyprus, 7–10 Dec 2015, pp. 300–309 (2015). doi:10.1109/UCC.2015.48
17. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM (JACM) **46**(5), 604–632 (1999). doi:10.1145/324133.324140
18. Mitra, S., Agrawal, M., Yadav, A., Carlsson, N., Eager, D.L., Mahanti, A.: Characterizing web-based video sharing workloads. TWEB **5**(2), 8 (2011). doi:10.1145/1961659.1961662
19. Rodrigues, T., Benevenuto, F., Cha, M., Gummadi, P.K., Almeida, V.A.F.: On word-of-mouth based discovery of the web. In: Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement, IMC 2011, Berlin, Germany, 2–4 Nov 2011, pp. 381–396 (2011). doi:10.1145/2068816.2068852
20. Sastry, N., Yoneki, E., Crowcroft, J.: Buzztraq: predicting geographical access patterns of social cascades using social networks. In: Proceedings of the 2nd ACM EuroSys Workshop on Social Network Systems, SNS 2009, Nuremberg, Germany, 31 March 2009, pp. 39–45 (2009). doi:10.1145/1578002.1578009
21. Scellato, S., Mascolo, C., Musolesi, M., Crowcroft, J.: Track globally, deliver locally: improving content delivery networks by tracking geographic social cascades. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28–April 1, 2011, pp. 457–466 (2011). doi:10.1145/1963405.1963471
22. Torres, R., Finamore, A., Kim, J.R., Mellia, M., Munafò, M.M., Rao, S.G.: Dissecting video server selection strategies in the YouTube CDN. In: Proceedings of the 31st International Conference on Distributed Computing Systems, ICDCS 2011, Minneapolis, Minnesota, USA, 20–24 June 2011, pp. 248–257 (2011). doi:10.1109/ICDCS.2011.43
23. Traverso, S., Huguenin, K., Trestian, I., Erramilli, V., Laoutaris, N., Papagiannaki, K.: Tailgate: handling long-tail content with a little help from friends. In: Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, 16–20 April 2012, pp. 151–160 (2012). doi:10.1145/2187836.2187858