Simulation Modelling Practice and Theory 71 (2017) 114–133

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Content delivery simulations supported by social network-awareness

Irene Kilanioti*, George A. Papadopoulos

Department of Computer Science, University of Cyprus, 1 University Avenue, Nicosia, Cyprus

ARTICLE INFO

Article history:
Received 7 August 2016
Revised 2 December 2016
Accepted 2 December 2016

Keywords:
Social video sharing
Social cascading
Simulations
Content distribution networks
Internet measurements
Social prediction
Prefetching

ABSTRACT

In this study we conduct experiments on a modified content delivery simulation framework, as we aspire to compare miscellaneous policies for dynamic OSN-aware content delivery. The incorporation of an OSN-aware dynamic mechanism becomes indispensable for CDN services, since (i)significantly large proportion of Internet traffic results from -easily produced via online media services and transmitted over OSNs- bandwidth-intensive multimedia content and (ii)multimedia content providers, such as YouTube, often rely on ubiquitous Content Distribution Networks (CDNs) infrastructures. Our policies take patterns of user activity over OSNs and exploit geo-social properties of users participating in retransmissions of items over OSNs (social cascades), proceed to incorporate various caching schemes of the underlying infrastructure, different policies for the handling of OSN data and various approaches that take into account the efficient timing of prefetching. The simulation framework we introduce can serve as the basis of further parameterized content delivery experimentation that exploits information transmission over OSNs and decreases replication costs by selectively copying items to locations where items are likely to be consumed.

1. Introduction

1.1. CDNs

The volume of content exchanged over the Internet nowadays has made content delivery integral part of the digital ecosystem, and infrastructures like Content Distribution Networks (CDNs) have flourished as they mitigate the challenges of content delivery. Content Distribution Network (CDN) services are being increasingly used to ensure the smooth delivery of multimedia content for a wide range of innovative services, including multimedia social networks, P2P video streaming, IPTV, interactive online games, cloud multimedia content delivery and content-centric networks. The amount of Internet traffic generated every day by online multimedia streaming providers such as YouTube [1] has reached huge numbers. These providers often rely on CDNs [2] to distribute their content from storage servers to multiple locations over the planet. CDNs are ubiquitous by their presence. A CDN is essentially an overlay network with the task of improving Internet service quality via replication of the content from place of origin to surrogate servers scattered over the Internet. Requests are served from

* Corresponding author.
E-mail addresses: ekoila01@cs.ucy.ac.cy (I. Kilanioti), george@cs.ucy.ac.cy (G. A. Papadopoulos).

http://dx.doi.org/10.1016/j.simpat.2016.12.003
1569-190X/

surrogate servers closer to the clients' location. CDN servers exchange content in a cooperative way to maximize overall efficiency.

There exist many commercial CDNs, including Akamai, NTT Communications CDN, Limelight, Microsoft Azure CDN, CacheFly, MaxCDN, etc. with Akamai infrastructure deploying 170.000 servers in 102 countries [3] and Limelight deploying 18.000 servers in over 80 locations [4]. There is also a number of non-commercial ones [5,6]. CDNs may be very dissimilar in terms of the services provided and their geographic coverage. The optimization of their overall efficiency, as far as the user is concerned, is practically achieved with the automatic detection of the medium (computer/ smartphone / tablet), optimized management of the browser cache, server load-balancing, the consideration of the specific nature of the content of the media provider (video content may include video on demand, live videos, geo-blocked content, etc.) or features of certain operators, such as real-time compression, session management, etc. CDNs address in general the major issues of (i) the most efficient placement of surrogate servers in terms of high performance and low 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 [7].

Copy policies for CDNs include:

- Push-based copy policy: content is proactively prefetched to all surrogate servers, with minimum response time and maximum copying content cost. Push-based copy policy is typically used for information that is in high demand by users, e.g. large files or static assets that do not frequently change as often.
- Pull-based copy policy: content is forwarded to the surrogate server at the moment the user asks for it, with minimum copying cost and maximum response time. Pull-based copy policy is typically used for personalized information, e.g. small objects with inherent virality and limited duration.

Most modern CDNs deploy both pull and push zones, with pulling being the most dominant case.
As far as the cooperation of the CDN infrastructure elements is concerned, there exist:

- Cooperative CDNs: surrogate servers are cooperating with each other in case of cache misses to reduce replication and update cost, adopting various policies (closest server, random selection, load balancing, etc.) for the mapping between content request and surrogate server. Server selection decisions can be made on the client-side (e.g. probing, statistical estimator), server-side (e.g. server push) or by the network infrastructure (e.g. DNS, anycast-enabled routers). An empirical evaluation on client-side server selection algorithms in [8] has found that simple dynamic probing outperforms other common clientside approaches, but individual client probing does not scale in an Internet-wide information-centric setting. Server-side selection in view of anycast includes application and network-layer solutions [9–12] which may be either restricted to specific content (e..g web services, wireless ad hoc networks)or may need offline pre-computation as a preparation to serve content requests in real time.
- Uncooperative CDNs: content is asked either from the local surrogate server or the origin server.

Next generation ubiquitous CDNs will be leveraged in an array of ways to overcome the challenges of providing a seamless customer experience across multiple devices with varying connectivity and corresponding to the call for enterprise application delivery. They will have to go beyond efficient resource discovery and retrieval tasks of the established CDNs and support refined mechanisms for data placement, replication and distribution for a large variety of resource types and media formats. OSNs on the other hand create a potentially transformational change in user navigation and from this angle the rapid proliferation of OSNs sites is expected to reshape the architecture and design of CDNs. The challenges and opportunities highlighted in the interdisciplinary field of OSN-aware content delivery are bound to foster some interesting future developments, including innovative cache replacement strategies as a product of the systematic research of temporal, structural and geographical properties of social cascades. In this work, we implement and experimentally evaluate dynamic policies of content prefetching and have in our hands proof that OSNs can affect the content delivery infrastructure.

1.2. Motivation

Widespread use of ubiquitous OSNs has led to a radical change in public information sharing. The popularity of relatively data heavy multimedia applications, such as YouTube, has also risen [13–15], and the volume of User Generated Content (UGC) has exploded across all media platforms (more than 300 hours of video content are uploaded in YouTube every minute [16]). OSN users increasingly repost links they have received from others and a large proportion of bandwidth-intensive media is distributed via OSN links (for example YouTube videos links through retweets in Twitter), that contribute significantly to Internet traffic [17]. Although it is difficult to estimate the proportion of traffic generated by OSNs, it is observed that 320 million monthly active Twitter users send 500 million tweets per day, of which more than 400 tweets per minute include a YouTube link [18–20].

Video streaming operators like YouTube own content delivery infrastructure consisting of geo-distributed servers and caches all over the world. Cache selection mechanisms implemented within the delivery infrastructure of video operators aim to satisfy the growing demand by directing users to nearby servers that contain the data. Transmission Control Protocol (TCP), though, the protocol via which video data delivery is typically conducted, is subject to delay jitter and throughput variations and clients are required to preload a playout buffer before staring the video playback. Due to the use of TCP, that

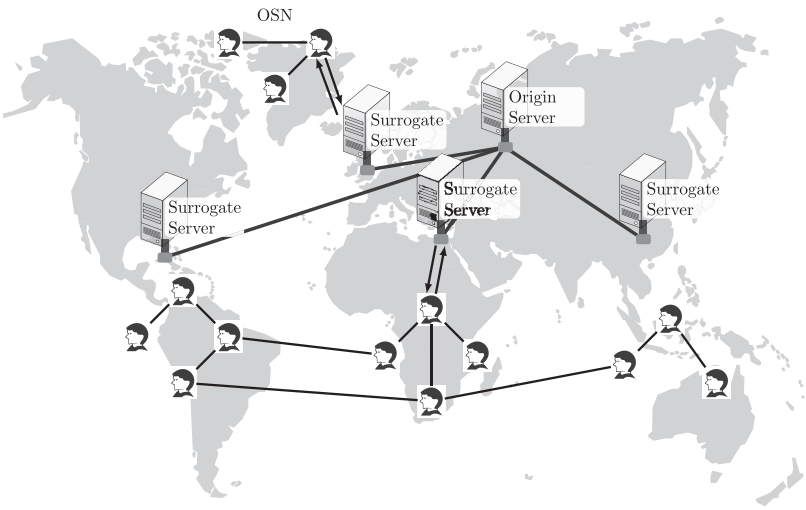

Fig. 1. An example of a CDN of an OSN.

via error recovery and congestion control ensures lack of packet losses, the quality of experience (QoE) of YouTube users is primarily determined by stalling effects on application layer as opposed to image degradation in User Datagram Protocol (UDP)-based video streaming [21]. Cache server selection is also highly Internet Service Provider (ISP)-specific for the YouTube case with geographical proximity not being the primary criterion and DNS level redirections for load-balancing purposes occur quite frequently and can considerably increase the initial startup delay of the playback. Hence, from a QoE management perspective, the smooth playback of the video rather than visual image quality is the key challenge for YouTube, and several network-level and client-level approaches are focused on the detection of interruptions, that have a dramatic impact on the user experience [21].

With the growing popularity of OSNs and the increased traffic due to outspread of information through the latter, the improvement of user experience through scaling bandwidth-demanding content largely depends on the exploitation of usage patterns found in OSNs. The motivation of our study, thus, is to support SNA tasks (such as cascades characterization) that accommodate large volumes of data for the improvement of user experience, e.g. via prefetching in the framework of a CDN infrastructure that streaming providers own. We aim to apply social information in situations where traditional bandwidth-intensive content scaling is infeasible. This happens because global replication demanded by traditional CDNs for the voluminous content produced becomes expensive. Moreover, user-generated content is especially difficult due to its long tail nature, with each item probably not popular enough to be replicated globally, but with the long-tail altogether getting sufficient accesses [22].

1.3. Why is our approach necessary: an Example

Let us consider Bob, who is living in London and is assigned to the London CDN servers of an Online Social Network service (Fig. 1). Most of Bob's social contacts 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 uploads a video that he wants to share. A naive way to ensure that this content is as close as possible to all users before any accesses occur would be to push the content to all other geographically distributed servers immediately.

Aggregated over all users, pushing can lead to a traffic spike, and users will experience latency in downloading the content. Indeed, content delivery is subject to delay jitter and clients are required to preload a playout buffer before staring the video playback. Moreover, the content may not be consumed at all. The problem of caching multimedia content is further exacerbated when Alice, for example, who is the only friend of Bob in Athens, is interested in that content; there are many such Alices in various places.

Rather than pushing data to all surrogates, we can proactively distribute it only to friends of Bob, who is depicted as B in Fig. 2, likely to consume it. The content will be copied only under certain conditions (e.g. geographically close user zones where Bob has mutual friends with high centrality, at non-peak times for the transfer according to the friend's zone), and users would not experience unnecessary delays in downloading the content. Thus, users such as Alice, depicted as A in Fig. 2, would not have to wait for the content to be downloaded, and the OSN provider would experience lower bandwidth costs, as traffic congestion would be avoided. With the use of evolving graph sequence analysis [23], this approach can be applicable as an in-house solution of OSN providers. The necessary data will not have to be calculated on the fly, but only across snapshots of gradually evolving graphs, reducing bandwidth and storage costs.

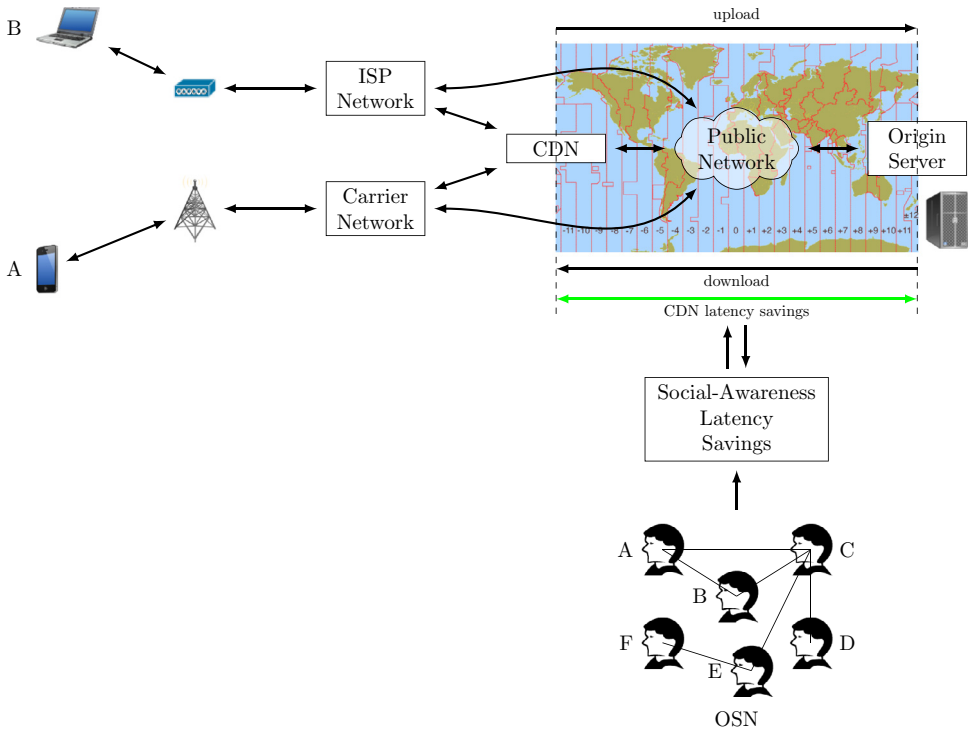

Fig. 2. Latency savings through the proposed mechanism.

1.4. Contributions

Overall, this paper provides the following contributions:

- In the paper we conduct a multitude of experiments, and we reach some fruitful conclusions regarding how OSNs can affect the content delivery performance. The Social Prefetcher policy surpasses performance improvement (30%) of similar works in pull-based methods [24], that are employed by most CDNs, and depicts better response times when various approaches are implemented within the suggested simulation framework (e.g., [24,25], as we present for first time in this work. Our policies use more refined topology of data centers and do not neglect storage issues [25], as storage costs are still a significant challenge despite the proliferation of cloud computing. 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 and we believe that they are generally applicable with a potentially high impact for large-scale systems with traffic generated by online social services and microblogging platforms.
- Our policies take patterns of user activity over OSNs and exploit geo-social properties of users participating in retransmissions of items over OSNs (social cascades). They proceed to incorporate various caching schemes of the underlying infrastructure, different predictive models for the circulation of multimedia over OSN data, and various approaches that take into account the efficient timing of prefetching and decrease replication costs by selectively copying items to locations where items are likely to be consumed. The simulation framework we introduce can serve as the basis of further parameterized content delivery experimentation that exploits information transmission over OSNs.
- We conduct experiments over a large corpus of YouTube videos transmitted over Twitter data. 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 billion messages and approximately 1.3 million single messages with an extracted video URL. The large user base of YouTube and Twitter allow us to obtain safe insights regarding user navigation behavior on other similar media and microblogging platforms, respectively, whereas restrictions of the CDN infrastructure (storage issues, network topology) are taken into account in the well-validated CDN simulator we use.

The rest of this paper is structured as follows. Section 2 reviews previous related work. Section 3 formally describes the addressed problem. The proposed policies along are described in Section 4. Section 5 gives an outline of the methodology, along with the preparation of the employed datasets. Our main findings are presented in Section 6. Section 7 concludes the paper and discusses directions for future work.

2. Related work

2.1. Metrics for characterization of social cascades

Local cascades affect only a relatively small number of individuals and typically terminate within one or two steps of the initiator. The size of local cascades is therefore determined mostly by the size of an initiator's immediate circle of influence, not by the size of the network as a whole. Global cascades affect many individuals, propagate for many steps, and are ultimately constrained only by the size of the population through which they pass.

Different approaches follow for the characterization of the extent a cascade will receive. Some of them are related to the extent that nodes are influenced by their neighbours on a *microscopic* level, such as the "vulnerability" that Watts [26] introduces, some to factors that function as obstacles to the spread of a cascade on a *macroscopic* level, such as those that Kleinberg and Easley [27] or Ver Steeg et al. [28] introduce. Dave et al. [29] combine microscopic and macroscopic level properties to identify how empirical factors like user's and their neighborhood's influencing ability or a specific action's influencing capability and come to the conclusion that action dominates in the prediction of the spread of the action.

In order to study the cascades a synopsis of the most indicative metrics follows. It includes some of the *structural properties* of the cascades, namely their *size*, which is the number of participants, including the initiator, their *length* [30], denoting the height of the cascade tree, the *time delay* between two consecutive steps of the cascade [31], the *time duration*, and the *rate of the cascade* [[32]]. *Geographical properties* are: the *geodiversity*, denoting the geometric mean of the geographic distances between all the pairs of users in the cascade tree, and the *georange*, denoting the geometric mean of the geographic distances between the users and the root user of the cascade tree [31].

The field of predicting the appearance of social cascades is very active ([30,33–35], etc.) Many studies focus on the prediction of the amount of aggregate activities (e.g. aggregate daily hashtag use [36]), whereas others focus either on the prediction of user-level behaviour, like retransmission of a specific tweet/ URL [37,38] or on the prediction of growth of the cascade size [33].

Although our work focuses solely on video sharing, we identify the following methods for virality prediction in general with our approach falling into the first category: (i) Feature-based methods are based on content, temporal and other features, and the learning algorithms schemes they use are based on simple regression analysis [33,39], regression trees [30], content-based methods [40], binary classification [41–43] etc. They do not focus, though, on the underlying network infrastructure, and often encounter difficulty in extracting all the necessary features due to the large volume of accommodated graphs. (ii) Time-series analysis works [44,45], on the other hand, argue that patterns of a resource's growth of popularity are indicative for its future retransmissions.

2.2. Structure characteristics of OSNs and measurement studies

The main characteristics of OSNs found by different studies are the following: Social networks are *power-law* [46], and, moreover *scale-free* [47], and exhibit the *small-networks property* [48]. Low diameter and high clustering coefficient, as well as power-laws for in- and out-degree distributions were confirmed for the Twitter social graph by Java et al. [49]. The graphs representing the interaction among the nodes of a social network, known as "interaction graphs" also exhibit the above properties [50], but lower levels of the "small-world" properties than their social graph counterparts.

Kumar et al. [51] observed that OSNs consist of the following three parts: *singletons* (nodes not participating in the network), the *giant component* (a subgraph that contains highly active nodes or to be put differently a dense core of low, actually shrinking, diameter, consistent with the observations in [52]) and the *middle region* (the remainder, that consists of various isolated communities interacting with one another but not with the overall network).

The role the topological features play concerning the information diffusion in a network is shown in various works: Indicatively, Draief et al. [53] show the effect of the network topology in the size of the population that eventually becomes infected, showing that if the ratio of cure to infection rates is larger than the spectral radius of the graph, a small initially infected population leads to a small also finally infected population and in the opposite case, in specific studied models, the final infected population is large.

Concerning the temporal evolution of OSNs, Leskovec et al. [52] highlight two additional characteristics, *densification power laws* (average degree increases with $E(t)$ and $V(t)$ denoting the edges of the social graph and its nodes at time t, respectively, and $a \in (1, 2)$, so that they obey: $E(t) \propto V(t)^a$) and *shrinking diameters* (diameter decreasing as the network grows), which are at odds, for example, with the preferential attachment model, in the notion that it generates graphs with average node degree constant over time and slowly growing diameters. Forest Fire Model [52] is proposed instead to simulate these properties.

In terms of *user workloads* in OSNs, Benevenuto et al. [54] collected traces from a social network aggregator website in Brazil, enabling connection to multiple social networks with a single authentication, and, thus, studied Orkut, MySpace, Hi5 and Linked. Benevenuto et al. presented a clickstream model to characterise users' interactions, frequency and duration of connection, as well as frequency of users' transition to activities, such as browsing friends' profiles, sending messages etc., with their analysis showing that browsing, which cannot be identified from visible traces, is the most dominant behavior (92%). They also, reinforced the social cascade effect, since more than 80% of bandwidth-intensive-media content like videos and photos was found through a 1-hop friend.

2.3. Systems, applications and techniques

In Buzztraq [25], Sastry et al. build a prototype system that takes advantage of the knowledge of the users' friends' location and number, to generate hints for the placement of replicas closer to future accesses. Comparing their strategy with location based placement, which instead uses the geographical location of recent users, they find substantial decrease of cost, when requests as part of cascades are more than random accesses of content. Furthermore, their system reacts faster when there is a new region shift, since it starts counting friends of previous users in a new region, even before a request comes from that region. The key concept of Buzztraq is to place replicas of items already posted by a user closer to the locations of friends, anticipating future requests. The intuition is that social cascades are rapidly spread through populations as social epidemics. The experimental results indicated that social cascade prediction can lower the cost of user access compared to simple location-based placement. Buzztrack is a simple system that only provides hints as to where to place objects. Other more complex constraints that the present work covers, such as server bandwidth and storage, are not taken into account. Moreover, social cascade is indirectly analyzed because there has to be a third-party page where users connect to view the videos and have access to their social profile.

Compared to Buzztraq, the novelty that this work introduces is that it takes into account the parameters of a CDN infrastructure, such as storage issues, and it also applies heuristics introduced from more recent works. In our approach, social cascades are directly analyzed and access to social profiles is not conducted via a third-party page, but rather with a real dataset from multimedia links spread over an OSN.

In the direction of distributing long-tailed content while lowering bandwidth costs and improving QoS, although without considering storage constraints, Traverso et al. in [24] exploit the time differences between sites and the access patterns that users follow. Rather than naively pushing UGC immediately, which may not be consumed and contribute unnecessarily to a traffic spike in the upload link, the system can follow a pull-based approach, where the first friend of a user in a Point of Presence (PoP) asks for the content. Moreover, rather than pushing content as soon as a user uploads, content can be pushed at the local time that is off-peak for the uplink and be downloaded in a subsequent time bin, also off-peak for the downlink. The larger the difference is between the content production bin and the bin in which the content is likely to be read, the better is the performance of the system.

In Tailgate the authors make the non-realistic assumption that once a video is delivered to a site, all future requests for that content originating from users of that site will be locally served. In other words, content is moved between sites only once. Tailgate moreover has the limitation that authors assume read patterns follow a diurnal trend similar to write patterns. Traverso et al. are more interested in time-of-day effects and more sophisticated read patterns with respect to content interest, quality of content etc. are ignored.

In [31], Scellato et al. study how Twitter can be used to examine social cascades of UGC from YouTube and discover popular objects for replication. They improve the temporary caching policy by placing content after accounting for the distance between users. For the model CDN system constructed and tested, Scellato et al. used the Limelight network properties with 19 clusters of servers worldwide. To test the system, two different video weights were used: geosocial, in which node locality values are calculated from all the users that have posted a message about the item (even without being involved in a cascade), and geocascade, in which node locality values are calculated from the users participating in the item's social cascade. It was shown that the model improved performance against a no weight policy, with geocascade weight performing better. Scellato et al. find that social cascades tend not to expand geographically, a notion which we also apply as a heuristic in this work.

The sequence of content requests created to the CDN directly from the Twitter messages within the dataset in [31] is based on the assumption that every video contained in a Twitter message is requested by each follower of the author with a certain probability. This assumption can be far from reality, as the authors explicitly state. In this work non-synthetic workloads are obtained via the request generator we use. A non-synthetic dataset from multimedia links spread over an OSN platform is used to directly analyze social cascades.

Zhou et al. [55] leverage the connection between content exchange and geographic locality (using a Facebook dataset they identify significant geographic locality not only concerning the connections in the social graph, but also the exchange of content) and the observation that an important fraction of content is "created at the edge" (*is user-generated*), with a web based scheme for caching using the access patterns of friends. Content exchange is kept within the same Internet Service Provider (ISP) with a drop-in component, that can be deployed by existing web browsers and is independent of the type of content exchanged. Browsing users online are protected with k-anonymity, where k is the number of users connected to the same proxy and are able to view the content.

Instead of optimizing the performance of UGC services exploiting spatial and temporal locality in access patterns, Huguenin et al., in [56], show on a large (more than 650,000 videos) YouTube dataset that content locality (induced by the related videos feature) and geographic locality are in fact correlated. More specifically, they show how the geographic view distribution of a video can be inferred to a large extent from that of its related videos, proposing a UGC storage system that proactively places videos close to the expected requests. Such an approach is extended in our work with the leverage of information from OSNs.

An interesting approach [57] applicable to the realm of content delivery is based on an architecture which combines global learning and local caches with small population. It is shown that age-based thresholds can timely exploit time-varying popularities to improve caching performance. Moreover, the caching efficiency is maximized by a combination of

Table 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
$N = \{n_1, n_2, \ldots, n_s\}$	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 κth 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 geographic zones with mutual followers/ with highest centrality metric (lobby-index / HITS) values
A_{u2v}	Number of actions where u influenced v
α, β, γ	Coefficients of feature set variables
U	Vector of YouTube interests of user u
V	Vector of Twitter interests of user v
$Score(u, t)$	Score of node u at time t
$dScore = dScore(u, t)/dt$	Derivative of Score of node u at time t
$content\_dist$	Content distance

global learning and clustering of access locations, accompanied by score mechanisms to help with practical issues at local caches. Practical considerations include, though, the size of the content that circulates over OSN and the long-tail effect, since the goal of the authors is first to learn a good estimate at the global point and then feed it back to the local caches in the form of content scores, thus, making the approach possibly prohibitive for OSN-aware content delivery.

3. Problem description

We endeavor to mitigate 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 dynamic 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 1), where the content 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:

$$\frac{Q_{hit}}{Q_{total}} \tag{1}$$

is maximum, whereas constraint

$$\sum_{\forall i \in O} S_i f_{ik} \leq C_k \tag{2}$$

is fulfilled, where:

$$f_{ik} = \begin{cases} 1 & \text{if object } i \text{exists in the cache of surrogate server } k \\ 0 & \text{if object does not exist} \end{cases} \tag{3}$$

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 $n_i \in N$.

4. Proposed policies

Our approach 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. 3).

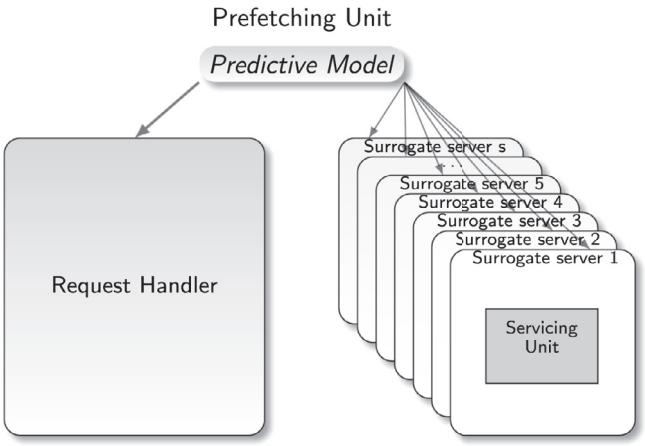

Fig. 3. The social-aware CDN mechanism.

4.1. For every new request in the CDN

The central idea in all policies for every new request (*timestamp*, $V_i(t)$, o) in the CDN 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 its first appearance. 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 a threshold), we check the importance of the user applying the Hubs Authorities (HITS) algorithm and checking its authority score (Algorithm 1). Additionally we check alternatively the viewership of the object in the media service platform (Algorithm 4) or if the time of the transmission is not within the peak-time range of the region of the user (Algorithm 5) [58].

Algorithm 1 Social Prefetcher.

```
if o.timestamp == 0 then
    o.timer = 0
    o.timestamp = request_timestamp
else if o.timestamp != 0 then
    o.timer = o.timer + (request_timestamp - o.timestamp)
    o.timestamp = request_timestamp
end if
if o.timer > time_threshold then
    o.timer = 0
    o.timestamp = 0
else if o.timer < time_threshold and˜user.authority_score > authority_threshold then
    copy object o to surrogate that serves user's V_i geographic zone
    for user V_y that follows user V_i do
        find surrogate server n_j that serves V_y's geographic zone
        copy object o to n_j
    end for
else if o.timer < time_threshold then
    copy object o to surrogates n_j that Subpolicy Lobby-index or Subpolicy HITS decides
end if
```

For users with a high authority score, we copy the object to all surrogate servers of the user's geographic zone and to the surrogate servers serving the geographic zones of all followers of the user (global prefetching). Otherwise, selective copying includes only the surrogates that the subpolicy decides (local prefetching). Subpolicies check the X closest geographic zones where a user has mutual friends and out of them, the Y with the highest value of the centrality metric as an average, denoting that the object is likely to be asked for more times. Copying is performed to the surrogate servers that serve the above geographic zones. Subpolicies for the plain algorithm, i.e. Subpolicy Lobby-index and Subpolicy HITS (Algorithms 2 and 3), include lobby-index and HITS ranking of geographic zones, respectively. For modified policies (Algorithms 4–6) only

HITS ranking, that performed better in the plain simulation framework experiments, is applied (Subpolicies: Algorithms 3 and 7).

Algorithm 2 Subpolicy Lobby-index.

find X geographic zones where (user V_i has mutual followers **and˜**they are closer to user's V_i geographic zone)
find the $L \subseteq X$ that have the highest $Lobby - index$ score
for geographic zones that belong to L **do**
  find surrogate server n_j that serves geographic zone
  copy object o to n_j
end for

Algorithm 3 Subpolicy HITS.

find X geographic zones where (user V_i has mutual followers **and˜**they are closer to user's V_i geographic zone
find the $H \subseteq X$ that have the highest $HITS$ score
for geographic zones that belong to H **do**
  find surrogate server n_j that serves geographic zone
  copy object o to n_j
end for

Algorithm 4 Popular Items.

if $o.timestamp$ == 0 **then**
  $o.timer$ = 0
  $o.timestamp$ = $request\_timestamp$
else if $o.timestamp$!= 0 **then**
  $o.timer$ = $o.timer$ + ($request\_timestamp$ - $o.timestamp$)
  $o.timestamp$ = $request\_timestamp$
end if
if $o.timer > time\_threshold$ **then**
  $o.timer$ = 0
  $o.timestamp$ = 0
else if $o.timer < time\_threshold$ **and˜**$user.authority\_score > authority\_threshold$ **then**
  copy object o to surrogate that serves user's V_i geographic zone
  for user V_y that follows user V_i **do**
    find surrogate server n_j that serves V_y's geographic zone
    copy object o to n_j
  end for
else if $o.timer < time\_threshold$ **and˜**$o.\Pi_i > \Pi_i\_threshold$ **then**
  copy object o to surrogates n_j that Subpolicy HITS decides
end if

The heuristics applied in our approach are based on the observations that users are more influenced by geographically close friends, and moreover by mutual followers, social cascades have a short duration, whereas the majority of cascades end within 24 h [7]. 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 hours, as well as threshold covering the entire percentage of requests.

Towards the direction of studying the evolution of social cascades that could lead to reduction schemes for the storage of whole sequences of large social graphs and the reduction of their processing time, we devised a simple Predictive Model for efficient calculation of the number of retweets of a video [59] and we proceed to incorporate it into our simulation framework (Algorithm 6). The number of retweets including a video link is associated within the model with a score depicting the influence of its uploader in the Twitter dataset ($Score(u, t)$), the increasing or decreasing trend the score depicts ($dScore = dScore(u, t)/dt$) as well as the distance of content interests among users of the YouTube and Twitter community ($content\_dist$). The validity of the predictors is analyzed in [59], whereas the intuition for their selection is based on the notion, that, the higher influence score a node depicts, the more influence it is expected to exert on other nodes of the social graph. Moreover, the $dScore/dt(u, t)$ expresses the popularity rise / fall of the node, and, lastly, the content distance associates the resource with the user context.

Detection of interests of users was conducted through analysis of the interests of the users they follow against directory information from http://wefollow.com, a website listing Twitter users for different topics along with a value indicating the

Algorithm 5 Efficient Timing.

if *o.timestamp* == 0 **then**
  o.timer = 0
  o.timestamp = *request_timestamp*
else if *o.timestamp* != 0 **then**
  o.timer = *o.timer* + (*request_timestamp* - *o.timestamp*)
  o.timestamp = *request_timestamp*
end if
if *o.timer* > *time_threshold* **then**
  o.timer = 0
  o.timestamp = 0
else if *o.timer* < *time_threshold* **and˜***user.authority_score* > *authority_threshold* **then**
  copy object *o* to surrogate that serves user's V_i geographic zone
  for user V_y that follows user V_i **do**
    find surrogate server n_j that serves V_y's geographic zone
    copy object *o* to n_j
  end for
else if o.timer < time_threshold **then**
  if $o.timestamp \ni (pts_{r_{V_i}}, pte_{r_{V_i}})$ **and˜**$o.timestamp \ni (pts_{r_{n_j}}, pte_{r_{n_j}})$ **then**
    copy object *o* to surrogates n_j that Subpolicy HITS decides
  end if
end if

Algorithm 6 Predictive Model.

if *o.timestamp* == 0 **then**
  o.timer = 0
  o.timestamp = *request_timestamp*
else if *o.timestamp* != 0 **then**
  o.timer = *o.timer* + (*request_timestamp* - *o.timestamp*)
  o.timestamp = *request_timestamp*
end if
if *o.timer* > *time_threshold* **then**
  o.timer = 0
  o.timestamp = 0
else if *o.timer* < *time_threshold* **and˜***user.Score* > *Score_threshold* **then**
  copy object *o* to surrogate that serves user's $V_i(t)$ geographic zone
  for user $V_y(t)$ that follows user $V_i(t)$ **do**
    find surrogate server n_j that serves $V_y(t)$'s geographic zone
    copy object *o* to n_j
  end for
else if *o.timer* < *time_threshold* **then**
  copy object *o* to surrogates n_j that Subpolicy Score decides
end if

Algorithm 7 Subpolicy Score.

find the *Y* geographic zones that depict the highest average values of Predictive Model(*Score*, *dScore*, *content_dist*) for user $V_i(t)$
for geographic zones that belong to *Y* **do**
  find surrogate server n_j that serves geographic zone
  copy object *o* to n_j
end for

importance of the user in the respective field. Interests of users in YouTube were calculated with a mapping of YouTube video categories to http://wefollow.com categories, as well, according to [60].

User score is calculated combining the number n_f of its followers, reduced by a factor of 100 to compensate the wide range of followers in the dataset from zero to more than a million, a quantity *b* catering for users with reciprocal followership, calculated by taking an average of number of a user's followers to the number of users he follows, as well as the effect *e* of a user's tweet, measured by multiplying average number of retweets with number of user's tweets and normalizing it

to correspond to the total number of tweets. The distribution of these combined metrics depicts large variance and we have applied a logarithmic transformation in order to avoid the uneven leverage of extreme values.

$$Score = \log\left(n_f + \left(\left(\frac{b}{100}\right) \times n_f\right) + e\right) \tag{4}$$

Content distance is calculated using cosine similarity between vectors of user's u YouTube (U) and user's v Twitter video interests (V), as follows:

$$content\_dist = 1 - \frac{U \cdot V}{\|U\|\|V\|} \tag{5}$$

Our model is expressed as the linear combination of $Score(u, t)$, $dScore(u, t)/dt$ and $content\_dist$, and its predicted output (total number of predicted values) expressed by A_{u2v} is:

$$A_{u2v} = 0.1460 \times Score(u,t) + 0.0200 \times \frac{dScore(u,t)}{dt} + 0.1656 \times content\_dist \tag{6}$$

For users with Score surpassing a threshold (average value: 1.2943 in the dataset), we copy the object to all surrogate servers of the user's geographic zone and to the surrogate servers serving the geographic zones of all user's followers. Otherwise, selective copying includes only the surrogates that Subpolicy Score decides. Subpolicy Score checks the Y geographic zones with the highest value of the combined feature set (Predictive Model(*Score, dScore*, *content_dist*)) as an average. Copying is performed to the surrogate servers that serve the Y geographic zones of highest combined feature set value, according to the coefficients derived from our analysis.

Algorithm 8 Caching scheme LRU / LFU / SIZE.

```
if o.size + current_cache_size ≤ total_cache_size then
    copy object o to cache of surrogate n_k
else if o.size + current_cache_size > total_cache_size then
    while o.size + current_cache_size > total_cache_size do
        for object o' in current_cache do
            if (current_timestamp - o'.timestamp) + o'.timer > time_threshold then
                copy o' in CandidateList
            end if
            if CandidateList.size > 0 and˜CandidateList.size! = total_cache_size then
                find o' that o'.timestamp is maximum and delete it
            else if CandidateList.size == 0 ∨ CandidateList.size == total_cache_size then
                use LRU/LFU/SIZE to delete any object o ∈ O
            end if
        end for
    end while
    put object o to cache of surrogate n_k
end if
```

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 o in the surrogate server n_k 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 (Algorithm 7):

- **Least Recently Used (LRU):** In the most straightforward extension of LRU for handling non-homogeneous 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.
- **Size-adjusted LRU (SIZE):** The optimization model devised in [61] 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

Table 2
Twitter dataset tweet information.

Id	Tweet id
Text	Tweet content
Created_at	Time of creation
Retweeted	If it is retweet
In_reply_to_status_id	Status id of the tweet to which it replies
In_reply_to_user_id	User id of the tweet to which it replies
Urls	Urls included
Retweet_count	Number of retweets

Table 3
Twitter dataset user information.

Id	User id
Verified	If user has verified email
Followers_count	Number of user's followers
Protected	If user's information is private
Listed_count	How many tweets refer the user
Statuses_count	How many tweets the user has published
Friends_count	How many users the user follows
Location	Explicit location of the user
Geo_enabled	If the service denoting the user location along with tweet is enabled
Lang	User language
Favourites_count	How many tweets user has added to favourites
Created_at	Time of creation
Time_zone	Timezone of the user

removed first. Objects in the cache are reindexed in order of increasing values of $S_i \cdot \Delta T_{i_k}$ and highest index objects are greedily selected and purged from the cache until the new object fits in.

We selected the caching schemes applied among a variety of caching algorithms: modified LRU (mLRU), scoring based caching algorithm (SC), Cache Management based on Temporal Pattern Solicitation (CMTPS) algorithm, etc. Our selection was based on the criteria of time complexity and ease of implementation within the CDN simulator framework [62].

LRU supports the fundamental locality principle that if a process visits a location in the memory, it will probably revisit the location and its neighbourhood soon, and being implemented by a double linked list, it gives operations of an $O(1)$ complexity (When a page in the list is accessed, it will be moved from its place to the end of the list. When a new page is accessed, it will be put in the end of the list and the oldest page is taken out of the list.). It poses, though, the problem of what will happen when a process scans a huge surrogate server cache. LFU overcomes this problem as it takes into account the advanced locality principle, that claims that the probability of revisiting will be increased if the number of the visits is bigger. LFU is implemented by a heap and thus has a logarithmic complexity for adding / removing a page from the heap, as well as updating the position of a page in the heap. Stale objects can remain a long time in the memory, while popular objects can be mistakenly removed [63].

A brief categorization [61] of cache replacement schemes for the web includes following categories:

- (i) Direct extensions of traditional policies: such as Least Frequently Used (LFU) and First In First Out (FIFO) besides LRU, which do not in general pay sufficient attention to object sizes.
- (ii) Key-based policies: objects are sorted based on a primary key, break ties based on a secondary key, break remaining ties based on a tertiary key and so on. The idea in using the key-based policies is to prioritize some replacement factors over others, although this prioritization may not always be the best case.
- (iii) Function-based replacement policies: They employ a function of various factors such as time since last access, entry time of the object in the cache, transfer time cost, object expiration time, etc. For example, Least Normalized Cost Replacement (LNC-R) algorithm employs a rational function of the access frequency, the transfer time cost and the size.

5. Experimental evaluation

- **Dataset:** Experimentation is conducted on a Twitter dataset containing geographic locations, follower lists and tweets for 37 million users. We take into account the 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 a video link extracted (information contained in dataset depicted in Tables 2, 3). Information for the Youtube users and videos contained in the dataset is depicted in Tables 4 and 5, respectively.
  Although our simulations are focused only on YouTube and Twitter, their wide popularity and massive user base allow us to obtain insights regarding user navigation behavior on other similar media and microblogging platforms, respectively.

Table 4
YouTube dataset user information.

Id	User id
num_youtube_videos_shared	Number of YouTube videos shared by the user
num_of_views_for_videos_shared_by_this_user	Number of views of videos shared by the user
num_of_comments	Number of comments of the user
categories	categories the user is interested in

Table 5
YouTube dataset video information.

id	video id
video_title	title of the video
video_uploader	id of the uploader
video_views_number	number of the video views
video_favorited_times	number of the times the video was favorited
video_raters_number	number of the video raters
video_likes_number	video likes number
video_dislikes_number	video dislikes number
video_total_number_comments	total number of comments
video_uploaded_time	uploading time
category	category of the video
Freebase_topic_id	list of Freebase topics of the video

Table 6
(a) Simulation characteristics and **(b)** Distribution of servers over the world for the experimental evaluation.

Topology nodes	3500
Redirection policy	Closest surrogate
Origin servers	1
Surrogate servers	423
User groups	162
Bandwidth	100 Mbit/sec

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

Vine [64], for example, which is a short-form video sharing service acquired from Twitter that allows users to record and edit five- to six-second-long looping video clips, can also exploit such a mechanism for sharing videos.

- **Simulation configuration:** The configuration of the simulation values for the stand-alone CDN simulator is depicted in Table 6(a). 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. 4). To simulate our policy and place the servers in a real geographical position, we used the geographical distribution of the Limelight network [65]. 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 6(b)). Depending on the closer distance between the surrogate region defined by Limelight and each of the geographic zones defined by Twitter (20 Limelight regions, 142 Twitter geographic zones), we decided where the requests from each geographic zone will be redirected. The population of each geographic zone was also taken into consideration. The INET generator [66] 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 [67], a tool that maps IP addresses and Autonomous System (AS) coordinates to geographical coordinates [68], and surrogate servers were assigned to topology nodes.

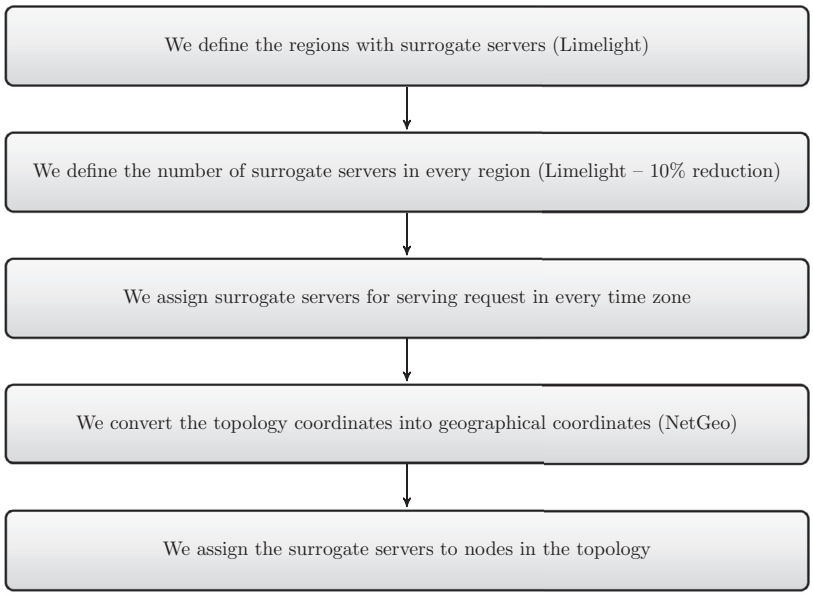

Fig. 4. Methodology followed.

After grouping users with explicit location provided by Twitter according to their geographic zone (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 geographic zone 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 ([24] on a daily basis and [31]), and same number of distinct videos for generation of requests.
- **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).
- **Classification methods:** To compare Predictive Model Algorithm with other methods we assumed that if A_{u2v} crosses a threshold, e.g. 30%, i.e., if more than 30% tweets of user u are retweeted by others users, then user u can be considered as a popular user, hence we handled our problem as a binary classification problem. Comparative classification was conducted with: Linear Regression, i.e., the suggested Predictive Model, Random Forest, Naive Bayes, Support Vector Machines (SVM), Stochastic Gradient Descent (SGD) and K-Nearest Neighbours (KNN) and we experimented with the respective confidence scores of all the aforementioned classifiers in Predictive Model Algorithm. Thus, modifications include the replacement of the geographic zones depicting the highest average values of Predictive Model(*Score, dScore*, *content_dist*), with those being derived from the application of Naive Bayes, Random Forest, SVM, SGD, and KNN schemes.
  - **SVM**: is a supervised learning model with associated learning algorithm that analyzes data used for classification and regression analysis. Given a set of training examples, each marked to belong to one of the two categories (popular/ non-popular user), the SVM training algorithm builds a model that assigns new examples into each of the categories, acting as a non-probabilistic binary linear classifier.
  - **SGD**: is a gradient descent optimization method for minimizing an objective function written as a sum of differentiable functions. It encompasses a popular algorithm for training a wide range of models in machine learning, including linear support vector machines, logistic regression and graphical models. Its use for training artificial networks is motivated by the high cost of running backpropagation algorithm over the full training set, as SGD overcomes this cost and still leads to fast convergence.
  - **KNN**: is a method classifying objects based on closest training examples in the feature space. The input consists of positive, typically small, integer -15 in our case- of closest training examples in the feature space. In KNN classification, the output is a class membership (popular/ non-popular user), whereas an object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its K-Nearest Neighbours.

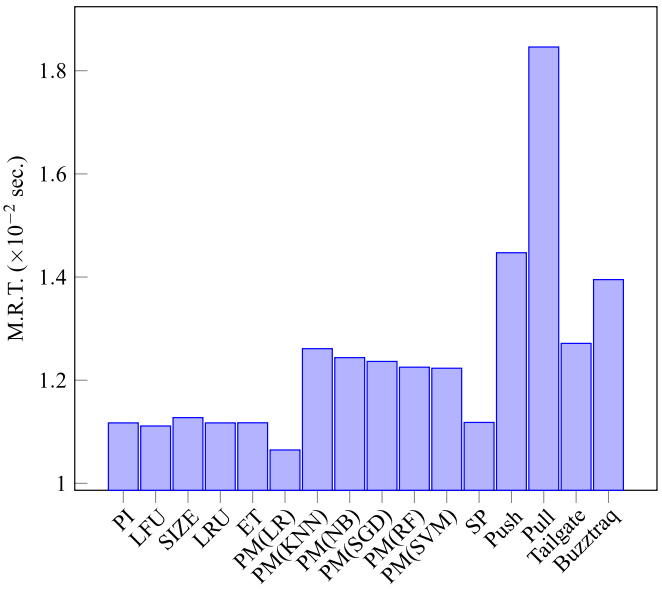

Fig. 5. Summary - Average M.R.T. Values for $X = 10$ Closest Timezones of Mutual Friends for threshold covering all requests.

6. Main findings

The statistic reports produced by the simulator are used to evaluate the proposed policies. There follows a short explanation of the metrics used in our experiments for the extraction of statistical results. For detailed results for selected cases, results for varying number of geographic zones, as well as additional server-side and network-side statistics you can refer to [7,59].

Client side measurements

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

- **Mean response time (M.R.T.)**: 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 i^{th} request. It starts at the timestamp when the request begins and ends at the timestamp when the connection closes.

Surrogate side measurements

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 of the surrogate servers.

Network statistics

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

- **Mean surrogate servers utility**: 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 M.R.T.s for the clients.

6.1. Impact on metrics

Table 7 presents the average values for all cases of testing and Fig. 5 the summary of average M.R.T. Values for $X = 10$ Closest Timezones of Mutual Friends for threshold covering all requests. As the time threshold increases from 24 to 48 h and to hours covering the entire set of requests, we observe that the Mean Response Time decreases steadily. At the same time Hit Ratio and Mean Utility of Surrogate Servers increase. Here, we present indicative values for the 10 closest geographic

Table 7
Average metric values for $X = 10$ timezones of close mutual friends.

Policy	Mean response time (Avg, 10^{-2}sec.)	Hit ratio %	Mean surrogate server utility %
Popular Items - 24-h	1.1383	32.81	96.01
Popular Items - 48-h	1.1352	33.08	96.01
Popular Items - all-h	1.1172	34.58	96.04
LFU - 24-h	1.1121	32.87	96.17
LFU - 48-h	1.1117	33.26	96.23
LFU - all-h	1.1112	34.61	96.60
SIZE - 24-h	1.1261	32.10	95.42
SIZE - 48-h	1.1270	32.39	95.98
SIZE - all-h	1.1274	33.78	95.99
LRU - 24-h	1.1158	32.77	96.00
LRU - 48-h	1.1166	33.06	96.01
LRU - all-h	1.1172	34.42	96.01
Efficient Timing - 24-h	1.1411	32.74	95.97
Efficient Timing - 48-h	1.1376	32.61	95.99
Efficient Timing - all-h	1.1174	34.38	96.01
Predictive Model (LR)- 24-h	1.0666	33.80	97.00
Predictive Model (LR)- 48-h	1.0662	33.84	97.10
Predictive Model (LR)- all-h	1.0647	34.92	97.38
Social Prefetcher 24-h	1.1412	32.12	95.98
Social Prefetcher 48-h	1.1377	32.42	96.00
Social Prefetcher all-h	1.1181	34.16	96.01
Predictive Model (SGD)- 24-h	1.2401	32.28	95.90
Predictive Model (SGD)- 48-h	1.2389	32.54	95.99
Predictive Model (SGD)- all-h	1.2364	33.99	96.00
Predictive Model (SVM)- 24-h	1.2268	32.25	95.86
Predictive Model (SVM)- 48-h	1.2249	32.26	95.91
Predictive Model (SVM)- all-h	1.2232	33.83	95.94
Predictive Model (RF)- 24-h	1.2270	32.23	95.81
Predictive Model (RF)- 48-h	1.2266	32.25	95.89
Predictive Model (RF)- all-h	1.2252	33.81	95.94
Predictive Model (NB)- 24-h	1.2463	32.21	95.77
Predictive Model (NB)- 48-h	1.2451	32.22	95.79
Predictive Model (NB)- all-h	1.2437	33.76	95.63
Predictive Model (KNN)- 24-h	1.2655	32.17	95.61
Predictive Model (KNN)- 48-h	1.2637	32.21	95.70
Predictive Model (KNN)- all-h	1.2611	33.68	95.79
Plain CDN Simulator - Push	1.4471	31.10	94.03
Plain CDN Simulator - Pull	1.8460	30.97	94.42
Tailgate	1.2714	32.11	95.57
Buzztraq	1.3950	31.00	94.98

zones of mutual followers and experiments were conducted for varying subsets of 1, 5 and 10 geographic zones with the highest influence metric, respectively, where copying will ultimately be performed for all policies. The lowest M.R.T.s appear for the cases of the time threshold covering all requests for all the policies. In general, between Popular Items and Efficient Timing we observe a better performance in terms of M.R.T.s achieved for the Popular Items, where the viewership within the YouTube platform is considered. All policies perform better than the plain Social Prefetcher approach apart from SIZE policy, which however performs better than plain CDN Simulator Push and Pull policies. 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.

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 M.R.T. (and increase of Hit Ratio and Mean Surrogate Servers Utility, respectively) as the geographic zones increase, and a point after which the M.R.T. starts to increase again. This decrease in the M.R.T. occurs with approximately 5 geographic zones out of the 10 used for LRU scheme (for a fixed number of closest geographic zones with mutual followers), and with 7 geographic zones for LFU. After this point the slight increase in the M.R.T. 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 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.

The trade-off between the reduction of the response time and the cost of copying in servers is expressed for all schemes used with an M.R.T. decrease as number of examined X geographic zones increase, and a point after which the MRT starts

to increase again [59]. For the scheme augmented with our Predictive Model, namely the Linear Regression, for example, this shift occurs with approximately 6 geographic zones out of the 10 used (for a fixed number of closest geographic zones with mutual followers), whereas the system is faster burdened for Popular Items and Efficient Timing (switchpoint with 4 geographic zones). After this point the slight increase in the MRT 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 the server where copying is likely to take place. We observe that Linear Regression outperforms all the other schemes, depicting MRTs smaller than their respective. We note here that geographic zones with highest average values for each scheme, that Subpolicy defines, are pre-calculated, in order to reduce computational burden in the simulations.

6.2. Comparison with other works

Concerning comparison with related works, storage issue is not addressed in [24] and its performance is measured with a ratio of download times for buffering stage of videos, that do not coincide with the response times a CDN measures. We note, though, that with the policy proposed herein, there is a significant improvement of the Social Prefetcher framework as a whole over their respective improvement (30%) in pull-based methods employed by most CDNs. Moreover, in our approach, we use a more refined topology of data centers and take storage issues into account. Although the proliferation of cloud services has led to a reduction in storage costs over the past years, storage costs still remain a significant factor that can be reduced under certain conditions.

In terms of performance, we note that with even with the plain Social Pefetcher policy proposed herein, there is a significant improvement over the respective improvement (39.43% only for the plain Social Prefetcher, upto 42.32% for Predictive Model(LR), whereas 30% in [24]) of other methods in pull-based methods employed by most CDNs, even though they moreover overlook storage issues of the distributed infrastructure.

Social Prefetcher algorithm depicts smaller response times compared to the implementation of other algorithms for content delivery within the Social Prefetcher framework, e.g. plain Location Based Placement (LBP) algorithm for $k_{re} = 3$ locations incorporated in our framework depicts a M.R.T. of 17.32023 ms., Buzztraq [25] incorporated in our framework depicts a M.R.T. of 13.95011 ms., and Tailgate algorithm [24] depicts a M.R.T. of 12.71411 ms. for the case of Full Information/ Perfect Reads, that the system has access to the social graph.

We note here that LBP uses the geographical location of recent users to place replicas, whereas social cascade prediction [25] places replicas in regions where the social cascade is densest, as determined by the average number of friends who have accessed the UGC. These strategies are evaluated in the specific case where the UGC provider is allowed to place a fixed number, k_{re}, of replicas (k_{re}=3). Hence, the LBP strategy amounts to placing the replicas in the top k_{re} regions ranked by number of recent users, whereas social cascade prediction ranks regions by the number of friends of previous users and places replicas in the top k_{re} regions. Buzztraq is expected to work better if there is a strong social cascade component driving the user accesses, since it uses the history of social cascades (i.e. how many friends of a visitor also accessed the content) in contradiction to the history of previous accesses to predict future accesses (LBP).

6.3. Comparison with static approaches

Concerning comparison with static approaches, namely those that do not include the social-awareness component, Plain CDN Simulator - Push case (M.R.T. 14.471 ms.) refers to proactive prefetching of content to all surrogate servers. In this case the M.R.T. becomes minimum for the plain simulator, since every request is bound to be satisfied, whereas the content copying cost is maximum. In the Plain CDN Simulator - Pull case (M.R.T. 18.460 ms.) content is forwarded to the surrogate server at the moment the user asks for it, hence, the copying cost becomes minimum, but the response time is maximum for the plain simulator.

Most observed systems are assumed to be of either pure-push or pure-pull type with pure-push systems being, in fact, pure-broadcast and pure-pull systems, on the other hand, being pure-unicast [69]. Practically, however, most systems combine both forms of data communication, push and pull. The pull policy is typically used for personalized information, while the push policy deals with information that is in high demand by users. Most modern CDNs still deploy both pull and push zones, with pulling being the most dominant case. Pull zones are more frequently used simply because they more easily automatically cache assets of the clients. However, push zones are still used, but they usually concern dealing with hosting large files or static assets that do not frequently change. Small objects of inherent virality and limited duration should ideally be pulled by the CDN.

We notice that Social Prefetcher approach outperforms both Plain CDN Simulator - Push and Plain Simulator - Pull policies for the case of 15% of the whole catalog size used in our experiments. The performance of pull and push strategies varies in regard to the load with pull strategies achieving a lower M.R.T. under high loads and push strategies being superior under low to medium loads [70–72].

Plain CDN Simulator - Push case refers to proactive prefetching of content to all surrogate servers. In this case the M.R.T. becomes minimum for the plain simulator, since every request is bound to be satisfied, whereas the content copying cost is maximum. In the Plain CDN Simulator - Pull case content is forwarded to the surrogate server at the moment the user asks for it, hence, the copying cost becomes minimum, but the response time is maximum for the plain simulator. Most

observed systems are assumed to be of either pure-push or pure-pull type. Practically, however, most systems combine both forms of data communication, push and pull. The pull policy is typically used for personalized information, while the push policy deals with information that is in high demand by users. Most modern CDNs still deploy both pull and push zones, with pulling being the most dominant case. Pull zones are more frequently used simply because they more easily automatically cache assets of the clients. However, push zones are still used, but usually when dealing with hosting large files or static assets that dont frequently change as often. Small objects with inherent virality and limited duration should ideally be pulled by the CDN.

6.4. Caching schemes performance

In our example LFU performs best, and SIZE depicts a poor performance compared with the two other schemes, as it is unable to take advantage of the frequency skew. In our study the objects' sizes follow a zipf-distribution. As studies [61] depict, the effect of cost correlation with size has the greatest effect on the performance of SIZE algorithm, which discriminates against larger objects, -affecting the storage needs of our infrastructure- and the least effect on the LRU scheme, the performance of which remains unchanged. Consequently, when cost correlation with size increases, the cost savings are reduced as well, even though the hit ratios are the same.

When we consider which objects to replace when a new object enters a web cache we should consider not only the relative frequency but also factors such as object sizes, transfer time savings and expiration times. Thus, additional suggested features could be: the handling of object expiration times, enforcement of an admission control policy which will decide if object will be cached or not in the first place, that will eliminate the disruption caused from the addition of an object in cache and the necessary removal of other objects, and is significant for caching of non-uniform sized objects.

7. Conclusions

In the present work, we comparatively study modified dynamic policies of OSN content prefetching. The modifications concern parameters of temporal or contextual nature, as well as fundamental parameters associated with the content delivery infrastructure. Bandwidth-intensive multimedia delivery over a CDN infrastructure is experimentally evaluated with laborious simulations, taking into account features of the network that works in the related literature overlook. Although we recognize the focused scope of this work and the limitations of its conduction solely with Twitter and YouTube data, the scale of the media allows us to make assumptions for generalization across different OSNs and microblog platforms. 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. Future extensions include experimentation with various score assignment formulas, as well as subsequent verification in the realm of content delivery. We believe that the simulation framework we introduce can serve as the basis of further parameterized content delivery experimentation that exploits information transmission over OSNs, and we hope that our findings will broaden the view on the spread of information in the web.

In the future, we could implement and experiment on variations of dynamic policy that aim to improve it with more complex functions that incorporate more refined handling of the time differences (enriched with diurnal trends) and exploit load-balancing among surrogate servers.Whereas our study is limited to a specific OSN and media service, our results are generally applicable with a potentially high impact for large-scale systems with traffic generated by online social services and microblogging platforms. As the number of internet users increases dramatically and OSNs open new perspectives in the improvement of Internet-based content technologies, new issues in the architecture, design and implementation of existing CDNs arise. Our research agenda includes the generalization of proposed policy to deal with multiple OSN platforms and mobile CDN providers.

Acknowledgments

This research work was partially supported by the ICT COST Action IC1406 High-Performance Modelling and Simulation for Big Data Applications (cHiPSeT).

References

[1] YouTube, https://www.youtube.com-yt-about, [online; accessed 1-sept-2016].
[2] G. Peng, CDN: Content distribution network technical report tr-125, experimental computer systems lab, stony brook university.
[3] Akamai, https://www.akamai.com/de/de/about/facts-figures.jsp, [online; accessed 1-sept-2016].
[4] Limelight., http://www.limelight.com, [online; accessed 1-sept-2016].
[5] M.J. Freedman, E. Freudenthal, D. Mazieres, Democratizing Content Publication with Coral, in: NSDI, Vol. 4, 2004, p. 18.
[6] L. Wang, K. Park, R. Pang, V.S. Pai, L.L. Peterson, Reliability and Security in the Codeen Content Distribution Network, in: USENIX Annual Technical Conference, General Track, 2004, pp. 171–184.
[7] I. Kilanioti, Improving multimedia content delivery via augmentation with social information, The Social Prefetcher Approach., Multimedia, IEEE Trans. 17 (9) (2015) 1460–1470, doi:10.1109/TMM.2015.2459658.
[8] S.G. Dykes, K.A. Robbins, C.L. Jeffery, An empirical evaluation of client-side server selection algorithms, in: INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol. 3, IEEE, 2000, pp. 1361–1370.

[9] E.W. Zegura, M.H. Ammar, Z. Fei, S. Bhattacharjee, Application-layer anycasting: a server selection architecture and use in a replicated web service, IEEE/ACM Trans. Networking 8 (4) (2000) 455–466.
[10] V. Lenders, M. May, B. Plattner, Density-based anycast: a robust routing strategy for wireless ad hoc networks, IEEE/ACM Trans. Networking 16 (4) (2008) 852–863.
[11] D. Katabi, J. Wroclawski, A framework for scalable global ip-anycast (gia), ACM SIGCOMM Comput. Commun. Rev. 30 (4) (2000) 3–15.
[12] A. Beben, J.M. Batalla, W.K. Chai, J. Śliwiński, Multi-criteria decision algorithms for efficient content delivery in content networks, Ann. Telecommun.-annales des telecommun. 68 (3–4) (2013) 153–165.
[13] Internet society. global internet report mobile evolution and development of the internet., http://www.internetsociety.org/globalinternetreport/, 2015.
[14] L. Plissonneau, G. Vu-Brugier, Mobile data traffic analysis: How do you prefer watching videos? in: Proceedings of the 22nd International Teletraffic Congress (ITC), IEEE, 2010, pp. 1–8.
[15] G. Maier, A. Feldmann, V. Paxson, M. Allman, On dominant characteristics of residential broadband internet traffic, in: Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference IMC, ACM, 2009, pp. 90–102.
[16] YouTube Statistics, https://www.youtube.com/yt/press/statistics.html, [online; accessed 1-sept-2016].
[17] Alexa, http://alexa.com-topsites, [online; accessed 1-sept-2016]}.
[18] Twitter, https://about.twitter.com-company, [online; accessed 1-sept-2016].
[19] Twitter official blog, https://blog.twitter.com-, [online; accessed 1-sept-2016].
[20] A. Brodersen, S. Scellato, M. Wattenhofer, YouTube Around the World: Geographic Popularity of Videos, in: Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16–20, 2012, pp. 241–250, doi:10.1145/2187836.2187870.
[21] T. Hoßfeld, R. Schatz, E. Biersack, L. Plissonneau, Internet video delivery in YouTube: From traffic measurements to quality of experience, in: Data Traffic Monitoring and Analysis, Springer, 2013, pp. 264–301.
[22] C. Anderson, L. Tail, The, revised and updated edition: why the future of business is selling less of more, hyperion, 2008.
[23] C. Ren, E. Lo, B. Kao, X. Zhu, R. Cheng, On querying historial evolving graph sequences, Proc. VLDB Endowment, 4(11) (2011).
[24] S. Traverso, K. Huguenin, I. Trestian, V. Erramilli, N. Laoutaris, K. Papagiannaki, TailGate: Handling Long-tail Content with a Little Help from Friends, in: Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16–20, 2012, 2012, pp. 151–160, doi:10.1145/2187836.2187858.
[25] N. Sastry, E. Yoneki, J. Crowcroft, Buzztraq: Predicting Geographical Access Patterns of Social Cascades Using Social Networks, in: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems, SNS 2009, Nuremberg, Germany, March 31, 2009, 2009, pp. 39–45, doi:10.1145/1578002.1578009.
[26] D.J. Watts, A simple model of global cascades on random networks, Proc. Nat. Acad. Sci. 99 (9) (2002) 5766–5771.
[27] D.A. Easley, J.M. Kleinberg, Networks, Crowds, and Markets - Reasoning about a Highly Connected World, Cambridge University Press, 2010.
[28] G.V. Steeg, R. Ghosh, K. Lerman, What Stops Social Epidemics? in: Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17–21, 2011, 2011.
[29] K.S. Dave, R. Bhatt, V. Varma, Modelling Action Cascades in Social Networks, in: Proceedings of the 5th International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17–21, 2011, 2011.
[30] E. Bakshy, J.M. Hofman, W.A. Mason, D.J. Watts, Everyone's an Influencer: Quantifying Influence on Twitter, in: Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9–12, 2011, 2011, pp. 65–74, doi:10.1145/1935826.1935845.
[31] 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, WWW 2011, Hyderabad, India, March 28, - April 1, 2011, 2011, pp. 457–466, doi:10.1145/1963405.1963471.
[32] M. Cha, A. Mislove, B. Adams, K.P. Gummadi, Characterizing social cascades in Flickr, in: Proceedings of the 1st workshop on Online social networks, ACM, 2008, pp. 13–18.
[33] J. Cheng, L.A. Adamic, P.A. Dow, J.M. Kleinberg, J. Leskovec, Can Cascades Be Predicted? in: 23rd International World Wide Web Conference, WWW 2014, Seoul, Republic of Korea, April 7–11, 2014, 2014, pp. 925–936, doi:10.1145/2566486.2567997.
[34] H. Kwak, C. Lee, H. Park, S.B. Moon, What Is Twitter, a Social Network or a News Media? in: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26–30, 2010, 2010, pp. 591–600, doi:10.1145/1772690.1772751.
[35] P.A. Dow, L.A. Adamic, A. Friggeri, The anatomy of large Facebook cascades, in: Proceedings of the 7th International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, 2013.
[36] Z. Ma, A. Sun, G. Cong, On predicting the popularity of newly emerging hashtags in twitter, JASIST 64 (7) (2013) 1399–1410, doi:10.1002/asi.22844.
[37] S. Petrovic, M. Osborne, V. Lavrenko, RT to Win! Predicting Message Propagation in Twitter, in: Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17–21, 2011, 2011.
[38] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, W. Kellerer, Outtweeting the Twitterers - Predicting Information Cascades in Microblogs, 3rd Workshop on Online Social Networks, WOSN 2010, Boston, MA, USA, June 22, 2010, 2010.
[39] G. Szabó, B.A. Huberman, Predicting the popularity of online content, Commun. ACM 53 (8) (2010) 80–88, doi:10.1145/1787234.1787254.
[40] O. Tsur, A. Rappoport, What's in a Hashtag?: Content Based Prediction of the Spread of Ideas in Microblogging Communities, in: Proceedings of the Fifth International Conference on Web Search and Web Data Mining, WSDM 2012, Seattle, WA, USA, February 8–12, 2012, 2012, pp. 643–652, doi:10.1145/2124295.2124320.
[41] M. Jenders, G. Kasneci, F. Naumann, Analyzing and Predicting Viral Tweets, in: 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13–17, 2013, Companion Volume, 2013, pp. 657–664.
[42] L. Hong, O. Dan, B.D. Davison, Predicting Popular Messages in Twitter, in: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28, - April 1, 2011, Companion Volume, 2011, pp. 57–58, doi:10.1145/1963192.1963222.
[43] A. Kupavskii, L. Ostroumova, A. Umnov, S. Usachev, P. Serdyukov, G. Gusev, A. Kustarev, Prediction of Retweet Cascade Size over Time, in: 21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012, 2012, pp. 2335–2338, doi:10.1145/2396761.2398634.
[44] J. Yang, J. Leskovec, Patterns of Temporal Variation in Online Media, in: Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9–12, 2011, 2011, pp. 177–186, doi:10.1145/1935826.1935863.
[45] S. Yang, H. Zha, Mixture of Mutually Exciting Processes for Viral Diffusion, in: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16–21 June 2013, 2013, pp. 1–9.
[46] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, B. Bhattacharjee, Measurement and analysis of online social networks, in: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, ACM, 2007, pp. 29–42.
[47] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[48] J.M. Kleinberg, Navigation in a small world, Nature 406 (6798) (2000) 845.
[49] A. Java, X. Song, T. Finin, B. Tseng, Why we Twitter: understanding microblogging usage and communities, in: Proceedings of the 9th WebKDD and 1st SNA-KDD workshop on Web mining and social network analysis, ACM, 2007, pp. 56–65.
[50] C. Wilson, B. Boe, A. Sala, K.P. Puttaswamy, B.Y. Zhao, User interactions in social networks and their implications, in: Proceedings of the 4th ACM European conference on Computer systems, Acm, 2009, pp. 205–218.
[51] R. Kumar, J. Novak, A. Tomkins, Structure and evolution of online social networks, Link Mining: Models, Algorithms, Appl. (2010) 337–357.
[52] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph evolution: densification and shrinking diameters, ACM Trans. Knowl. Discovery Data (TKDD) 1 (1) (2007).
[53] M. Draief, A. Ganesh, L. Massoulié, Thresholds for virus spread on networks, in: Proceedings of the 1st International Conference on Performance evaluation methodolgies and tools, ACM, 2006, p. 51.

[54] F. Benevenuto, T. Rodrigues, M. Cha, V. Almeida, Characterizing user behavior in Online Social Networks, in: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, ACM, 2009, pp. 49–62.
[55] F. Zhou, L. Zhang, E. Franco, A. Mislove, R. Revis, R. Sundaram, WebCloud: Recruiting social network users to assist in content distribution, in: Proceedings of the 11th IEEE International Symposium on Network Computing and Applications, Cambridge, MA, USA, 2012.
[56] K. Huguenin, A.-M. Kermarrec, K. Kloudas, F. Taïani, Content and geographical locality in user-generated content sharing systems, in: Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video, ACM, 2012, pp. 77–82.
[57] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, S. Chouvardas, Placing Dynamic Content in Caches with Small Population, in: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9, doi:10.1109/INFOCOM.2016.7524380.
[58] I. Kilanioti, G.A. Papadopoulos, Socially-aware Multimedia Content Delivery for the Cloud, in: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), 2015, pp. 300–309, doi:10.1109/UCC.2015.48.
[59] I. Kilanioti, G.A. Papadopoulos, Predicting video virality on Twitter, in: J.K.F. Pop, B.d. Martino (Eds.), Resource Management for Big Data Platforms: Algorithms, Modelling and High-Performance Computing Techniques, Springer Computer Communications and Networks Series, Springer, 2016. Ch. 20
[60] A. Abisheva, V.R.K. Garimella, D. Garcia, I. Weber, Who Watches (and Shares) What on YouTube? and When?: Using Twitter to Understand Youtube Viewership, in: 7th ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24–28, 2014, 2014, pp. 593–602, doi:10.1145/2556195.2566588.
[61] C. Aggarwal, J.L. Wolf, P.S. Yu, Caching on the world wide web, IEEE Trans. Knowl. Data Eng. 11 (1) (1999) 94–107, doi:10.1109/69.755618.
[62] I. Kilanioti, G.A. Papadopoulos, Delivering social multimedia content with scalability, in: J.K.F. Pop, B.d. Martino (Eds.), Resource Management for Big Data Platforms: Algorithms, Modelling and High-Performance Computing Techniques, Springer Computer Communications and Networks Series, Springer, 2016. Ch. 18
[63] M. Garetto, E. Leonardi, V. Martina, A unified approach to the performance analysis of caching systems, ACM Transactions on Modeling and Performance Evaluation of Computing Systems 1 (3) (2016) 12.
[64] Vine, https://vine.co, [online; accessed 1-sept-2016].
[65] J.L.C. Huang, A. Wang, K. Ross, Measuring and Evaluating Large-scale CDNS, in: Internet Measurement Conference (IMC), 2008, 2008, pp. 15–29.
[66] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, S.B. Moon, 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 2007, San Diego, California, USA, October 24–26, 2007, 2007, pp. 1–14, doi:10.1145/1298306.1298309.
[67] Center for Applied Internet Data Analysis, https://www.caida.org, [online; accessed 1-sept-2016].
[68] R. Torres, A. Finamore, J.R. Kim, M. Mellia, M.M. Munafò, S.G. Rao, Dissecting Video Server Selection Strategies in the YouTube CDN, in: 2011 International Conference on Distributed Computing Systems, ICDCS 2011, Minneapolis, Minnesota, USA, June 20–24, 2011, 2011, pp. 248–257, doi:10.1109/ICDCS.2011.43.
[69] J. Xu, D.-L. Lee, Q. Hu, W.-C. Lee, Data Broadcast, Handbook of Wireless Networks and Mobile Computing, John Wiley & Sons, Inc., 2002. Pp. 243–265
[70] R. Mirchandaney, D. Towsley, J.A. Stankovic, Adaptive load sharing in heterogeneous distributed systems, J. Parallel Distrib. Comput. 9 (4) (1990) 331–346.
[71] D.L. Eager, E.D. Lazowska, J. Zahorjan, A comparison of receiver-initiated and sender-initiated adaptive load sharing (extended abstract), in: Proceedings of the 1985 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 1985, ACM, New York, NY, USA, 1985, pp. 1–3, doi:10.1145/317795.317802.
[72] W. Minnebo, B.V. Houdt, Pull versus Push Mechanism in Large Distributed Networks: Closed Form Results, in: Proceedings of the 24th International Teletraffic Congress, ITC 2012, International Teletraffic Congress, 2012. 9:1–9:8