University of Cyprus

DEPARTMENT OF COMPUTER SCIENCE

IMPROVING CONTENT DELIVERY WITH OSN-AWARENESS

EIRINI T. KOILANIOTI

A Dissertation Submitted to the University of Cyprus in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

December, 2016

VALIDATION PAGE

Doctoral candidate: Eirini T. Koilanioti

Doctoral Thesis Title: Improving content delivery with OSN-awareness

The present Doctoral Dissertation was submitted in partial fulfillment of the requirements for the Degree of the Doctor of Philosophy at the ***Department of Computer Science*** *and was approved on the /12/2016 by the members of the* ***Examination Committee****.*

Examination Committee:

Research Supervisor:_____
Professor George A. Papadopoulos

Committee Member:_____
Professor Charalambos D. Charalambous

Committee Member:_____
Assistant Professor Michael Sirivianos

Committee Member:_____
Assistant Professor Dimitrios Tsoumakos

Committee Member:_____
Assistant Professor Vasos Vassiliou

DECLARATION OF DOCTORAL CANDIDATE

The present doctoral dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes or any other statements.

Eirini T. Koilanioti

_____

ABSTRACT IN GREEK

Τα Δίκτυα Παράδοσης Περιεχομένου (CDN) χρειάζεται συχνά να ανακαλέσουν περιεχόμενο απαιτητικό σε εύρος ζώνης για καταναλωτές πληθώρας εφαρμογών (ηλεκτρονικής μάθησης, υγείας, κοινωνικής δικτύωσης, έξυπνων πολέων κ.ά.). Μεγάλο μέρος της κίνησης στο Internet οφείλεται στους επιγραμμικούς παρόχους πολυμεσικού περιεχομένου και η κίνηση επιτείνεται από τη διάδοση περιεχομένου από τα ευρέως διαδεδομένα Επιγραμμικά Κοινωνικά Δίκτυα (OSNs). Στην παρούσα διδακτορική διατριβή ισχυριζόμαστε ότι η πληροφορία από τα κοινωνικά δίκτυα μπορεί να χρησιμοποιηθεί για προανάκληση περιεχομένου και να συνεισφέρει στη βελτίωση των υποδομών παράδοσης περιεχομένου.

Ο ισχυρισμός μας υποστηρίζεται από τρεις μελέτες, που εφαρμόζουν τη χρήση της πληροφορίας από τα κοινωνικά δίκτυα σε καταστάσεις όπου η συμβατική αντιγραφή περιεχομένου είναι αδύνατη (λόγω του όγκου των δεδομένων και της φύσης του παραγόμενου από χρήστες περιεχομένου, που μπορεί να μην είναι τόσο δημοφιλές για να αντιγραφεί στο σύνολο της υποδομής, αλλά συνολικά ευθύνεται για μεγάλο αριθμό αιτήσεων χρηστών). Η προσέγγισή μας βασίζεται σε πειραματισμό με εμπειρικά δεδομένα για την αναγνώριση των σχετικών ιδιοτήτων από τα κοινωνικά δίκτυα, και την ενσωμάτωση των τελευταίων σε σύγχρονα CDN.

Οι ελάχιστες προσεγγίσεις που έχουν προταθεί χρησιμοποιούν συνθετικά δεδομένα και αγνοούν: θέματα αποθηκευτικών χώρων της υποδομής, την αντιπροσωπευτικότερη τοπολογία της υποδομής, κ.ά..

Στην πρώτη μελέτη σκοπός μας είναι να εκμεταλλευτούμε τη δραστηριότητα των χρηστών των κοινωνικών δικτύων προκειμένου να βελτιώσουμε το μηχανισμό προανάκλησης περιεχομένου που ενσωματώνουμε σε προσομοιωτή CDN κίνησης, ενώ το κόστος αντιγραφής στους υποκατάστατους εξυπηρέτες του CDN λαμβάνεται υπόψη. Για την αναβίβαση του περιεχομένου χρησιμοποιούμε γεωχωρικά χαρακτηριστικά των χρηστών που συμμετέχουν στις αναμεταδόσεις περιεχομένου πάνω από τα Κοινωνικά Δίκτυα. Μη δημοφιλή συνολικά αντικείμενα συχνά διαδίδονται επανειλημμένα και εντοπίζονται σε συγκεκριμένες περιοχές του OSN. Προτείνουμε τον Social Prefetcher, που μειώνει το κόστος αναπαραγωγής με την επιλεκτική αντιγραφή αντικειμένων σε τοποθεσίες όπου πιθανόν θα ζητηθούν.

Επεκτάσεις της μελέτης περιλαμβάνουν πειραματισμό με παραλλαγές ως προς τα σχήματα caching στους υποκατάστατους εξυπηρετητές και περαιτέρω διερεύνηση του

θέματος της πλέον αποδοτικής χρονικά αντιγραφής (γνώση των ωρών αιχμής για την αναβίβαση / καταβίβαση περιεχομένου). Επίσης ενσωματώνουμε και άλλη πληροφορία συγκειμένου, π.χ. αριθμό θεάσεων μέσα στην πολυμεσική υπηρεσία. Οι παραλλαγές μας αποδεικνύεται πειραματικά ότι συνεισφέρουν στη μεγιστοποίηση της απόδοσης των CDN με ταυτόχρονη εξοικονόμηση του κόστους αντιγραφής.

Η δεύτερη μελέτη αφορά στο συνδυασμό δεδομένων σχετικά με τους χρήστες από το Twitter με δεδομένα σχετικά με βίντεο από την πλατφόρμα YouTube. Στοχεύουμε στο συσχετισμό της προβλεψιμότητας του διαμοιρασμού των βίντεο με το κοινωνικό συγκείμενο των χρηστών. Συνδυάζοντας τα δύο σύνολα δεδομένων πραγματοποιούμε διαπιστώσεις που κανένα από τα δύο δεν προσφέρει μεμονωμένα. Αναπτύσσουμε ένα ακριβές μοντέλο για την πρόβλεψη της μελλοντικής δημοφιλίας ενός πόρου-βίντεο βάσει χαρακτηριστικών από το κοινωνικό δίκτυο του χρήστη που το μετάδωσε αρχικά. Το σύνολο των χαρακτηριστικών που προτείνουμε στηρίζεται στην αποτίμηση της επιρροής ενός χρήστη, τη διακύμανσή της μέσα στο χρόνο, και την απόσταση των ενδιαφερόντων των χρηστών στην πλατφόρμα και το κοινωνικό δίκτυο.

Ολοκληρώνουμε ενσωματώνοντας τη δεύτερη μελέτη σε μια υποδομή παράδοσης περιεχομένου, προκειμένου να αποκομίσουμε οφέλη ως προς την ποιότητα εμπειρίας του χρήστη και τα κόστη των παρόχων.

Η διατριβή ολοκληρώνεται με ανακεφαλαίωση των κυριότερων συνεισφορών της και προτάσεις για μελλοντική έρευνα.

ABSTRACT IN AN INTERNATIONAL LANGUAGE

Content Distribution Networks (CDNs) often need to prefetch bandwidth-intensive content for consumers of a plethora of applications, including elearning, healthcare, social activities, and smart cities. Internet traffic generated by online multimedia streaming providers has exploded, and circulation of content over ubiquitous Online Social Networks (OSNs) intensifies the traffic growth. Our thesis is that information extracted from OSNs can be harnessed to facilitate proactive content caching decisions, and thereby build better CDN infrastructures.

Our claim is supported by three case studies, which apply social information in situations where traditional content scaling is infeasible: global replication demanded by traditional CDNs for the voluminous content produced becomes expensive, and 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 items altogether getting sufficient accesses-. We extensively experiment with empirical traces to identify relevant social properties and incorporate the latter into current CDNs.

The challenging endeavor of engineering of general OSN-aware content placement applications over a CDN infrastructure has received less attention. The very few existent approaches lack evaluation with nonsynthetic workloads and ignore storage issues of the infrastructure, the refined topology of data centers, etc.

In our first study the pursuit lies in exploiting the user activity extracted from OSNs to incorporate a dynamic mechanism to a CDN traffic simulator. The cost of copying to surrogate servers is taken into consideration. For the staging of content we use geo-social properties of users participating in social cascades, as unpopular long-tail user-generated items often go viral and are localized in a part of the OSN. We propose the Social Prefetcher, an approach that decreases replication costs by selectively copying items to locations where items are likely to be consumed.

Extensions of this work include caching scheme variations as well as further study of the issue of temporal diffusion, related to the most efficient timing of the content placement.

We exploit the knowledge of peak times for upload and download, and incorporate contextual information, such as the viewership within the media service. Our variations are experimentally proven to contribute toward maximization of CDN performance with regard to client-perceived metrics and content replication costs.

The second study examines merging of Twitter user-centric data with YouTube video-centric data. We aim at investigating the ties between predictability of video sharing and the social context of video uploaders, and obtain insights than neither employed dataset individually gives. We develop an accurate model to predict the future popularity of a video resource given features of the underlying network of its initiator: the notion of user influence, its fluctuation through time, as well as the distance of content interests among users for both datasets.

We finish by discussing a study integrating the second study with a Content Delivery infrastructure to reap possible benefits in terms of user experience and providers costs.

The Thesis concludes with a discussion of its main contributions and a suggestion of key topics for future research.

ACKNOWLEDGEMENTS

I am profoundly indebted to my supervisor, Professor George A. Papadopoulos, for his scientific guidance, continuous encouragement and prompt advice. As my Thesis supervisor, Professor Papadopoulos supported me in all stages of the research and was very generous with his time and knowledge despite his busy agenda. His contribution to this Thesis was decisive.

I would like here to thank the members of my examination Committee for their valuable comments and suggestions.

I express my sincere gratitude to the Greek State Scholarships Foundation, that supported me financially through years 2012-2015, for the Ph.D. scholarship (Lifelong Learning Programme, Customized Evaluation Process) that gave me the opportunity to pursue my Thesis in the Department of Computer Science, University of Cyprus, and without which this Thesis would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

This Thesis discusses the challenges which are inherent in developing content delivery applications that leverage information from Online Social Networks (OSNs), and proceeds to propose novel algorithms that efficiently handle the media circulating over OSNs within the framework of a Content Distribution Network (CDN) infrastructure. Video streaming sites comprise a considerable part of the Internet data volume ([156], [136]) and the recent surge of popularity of video content intensified by circulation over OSNs calls for investments of the respective operators in the realm of content delivery. The video streaming operators 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. However, transmission via the Transmission Control Protocol (TCP), 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 via error recovery and congestion control ensures lack of packet losses, the quality of experience (QoE) of YouTube is primarily determined by stalling effects on the application layer as opposed to image degradation in UDP-based video streaming [99].

1.1 Foundations

Our aim in this chapter is to establish a common understanding of the basic concepts used in this Thesis, concepts related to content delivery, social networks and the transmission of media objects over them. In this respect, a few relevant definitions are first provided to

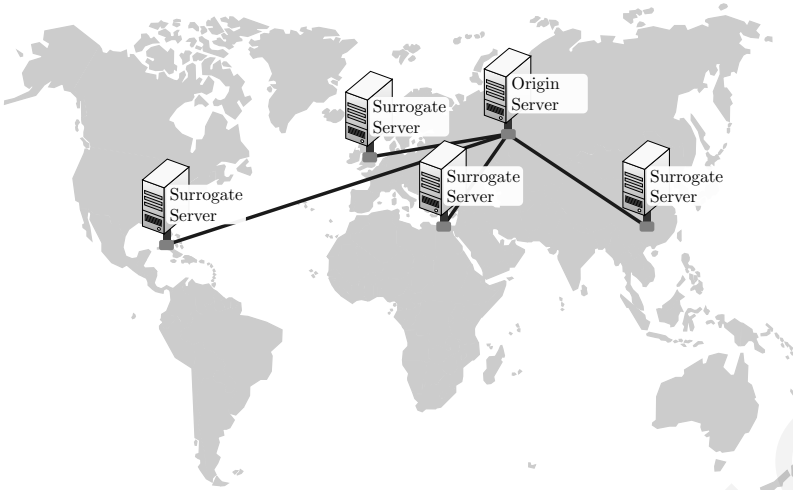


Figure 1: A CDN.

facilitate the establishment of a common frame of reference for understanding the problem investigated in this Thesis.

Content Distribution Networks

The amount of Internet traffic generated every day by online multimedia streaming providers such as YouTube [24] has reached huge numbers. These providers often rely on Content Distribution Networks (CDNs) [154] to distribute their content from storage servers to multiple locations over the planet. A CDN is 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 (Fig.1). Requests are served from surrogate servers closer to the clients' location. CDN servers exchange content in a cooperative way to maximize overall efficiency.

The general principle of CDNs is to replicate data dynamically in various places of the world as close as possible to the user that will consume it. They are mainly used for sites streaming large video files, sites which consist of mainly large media files like image sites, sites with known heavy traffic in different countries, sites with many tablet and mobile users, whereas they are not suitable for sites that have their main traffic in one geographic area or region.

They are, however, 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 (either computer or mobile -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.

There are 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 [1] and Limelight deploying 18.000 servers in over 80 locations [10]. There is also a number of non-commercial ones [83], [182].

CDNs, often operated as SaaS in cloud providers (Amazon CloudFront, Microsoft Azure CDN, etc.) aim at addressing the problem of smooth and transparent content delivery. A CDN actually drives cloud adoption through enhanced performance, scalability and cost reduction. With the limitation for both CDNs and cloud services being the geographic distance between a user asking for content and the server where the content resides, cloud acceleration and CDN networks are both complementary to achieving a goal of delivering data in the fastest possible way. Although cloud mainly handles constantly changing and, thus, not easily cached dynamic content, utilization of CDNs in cloud computing is likely to have profound effects on large data download [131].

In a manner that is complementary to the above, 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 [110].

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, e.g. MaxCDN, EdgeCast, Amazon CloudFront, BitGravity, Akamai, CDNetworks, CacheFly, ChinaCache, MaxCDN, CDN77, etc., 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.
- Uncooperative CDNs: content is asked either from the local surrogate server or the origin server.

Online Social Networks

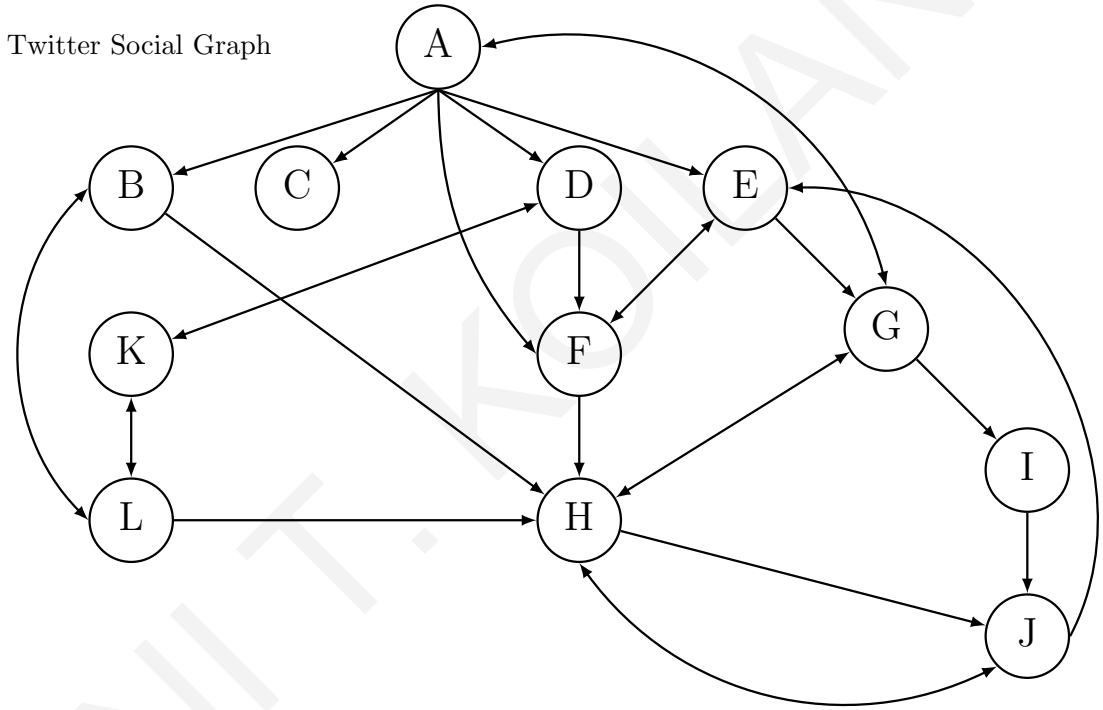


Figure 2: An example of the Twitter social graph.

Formally, an *OSN* is depicted by a directed graph $G = (V, E)$, where V is the set of the vertices of the graph representing the nodes of the network and E are the edges between them, denoting the various relationships among the edges of the graph [76]. An example of a social graph for Twitter OSN is depicted in Fig. 2. The semantics of these edges are different for different social networks: for Facebook, for example, edges are usually translated into personal acquaintance, whereas in LinkedIn they mean business contact. As far as the directionality of the edges of the social graph is concerned, it depends on the OSN the graph depicts. For Facebook, an edge denotes mutual friendship between the endpoints

of a link. For Twitter, if the edge between A and B points at B, B is a follower of A, meaning that A's posts (tweets) appear in B's main Twitter page. We say that the neighbours of a node are defined as the nodes that are in a 1-hop away distance from it in the social graph.

A node centrality is indicative of the importance of a node within a social network. It is given in terms of a real-valued function on the vertices of a graph, where the values produced are expected to provide a ranking which identifies the most important nodes [48], [49].

Twitter is an OSN with a microblogging service as its core functionality, that allows users to exchange real-time posts of up to 140 characters (tweets) either via a web or mobile interface (apps, SMSs). Twitter users subscribe to the content of other users, which they can repost (retweet). The retweet feature, moreover, helps in the acknowledgement of reposts. Twitter was launched in 2006, and counts 320 million monthly active Twitter users, that send 500 million tweets per day ([19], [20]).

In this Thesis we refer to the term *OSN-aware* system / algorithm/ mechanism to refer to a system/ algorithm/ mechanism that takes information from OSNs into consideration. The information incorporated in OSNs either involves their general structural properties or information exchanged over them.

Social Network Analysis SNA is the study of nodes and their ties in a network with the ultimate purpose to mine useful patterns between them [183].

Bandwidth-intensive Media

A fusion of bandwidth- and storage-demanding media, which may include text, graphics, audio, video, and animation is characterized as *Bandwidth-intensive Media*. Bandwidth-intensive media generated from the users and circulating over OSNs is currently ubiquitous due to the proliferation of smartphones, video editing software and cheap broadband connections. The *volume* (refers to the amount of data), *variety* (refers to the number of types of data) and *velocity* (refers to the speed of data processing needed) of the data allows us to characterise it also as *Big Data*. Multimedia content delivery technologies are nowadays essential 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.

Scaling refers to the fact that the throughput of the proposed systems / algorithms / policies / mechanisms remains unchanged with the increase in the data input size, such as the large datasets that social graphs comprise and the social cascades that amplify the situation.

A commonplace observation is the *Long-Tail effect* happening with the content generated from OSNs users, where many objects with a small number of individual requests may in aggregate account for a big contribution in the total object requests [34]. The cost of scaling such content can be expressed in different ways: For a CDN, or example, can be the number of replicas needed for a specific source, may take into account the optimal use of memory and processing time of a OSN-aware built system, etc.

Information Diffusion

In his classic work Rogers [161] defines *information diffusion* as the process in which an innovation is communicated through certain channels over time among the members of a social system. In this context, the innovation is defined as the first spread of an information from an originator. Rogers indicates that innovation and imitation are the two basic factors that drive the spread of a new idea: whereas the innovators' motive is their desire to try an innovation, imitators are influenced.

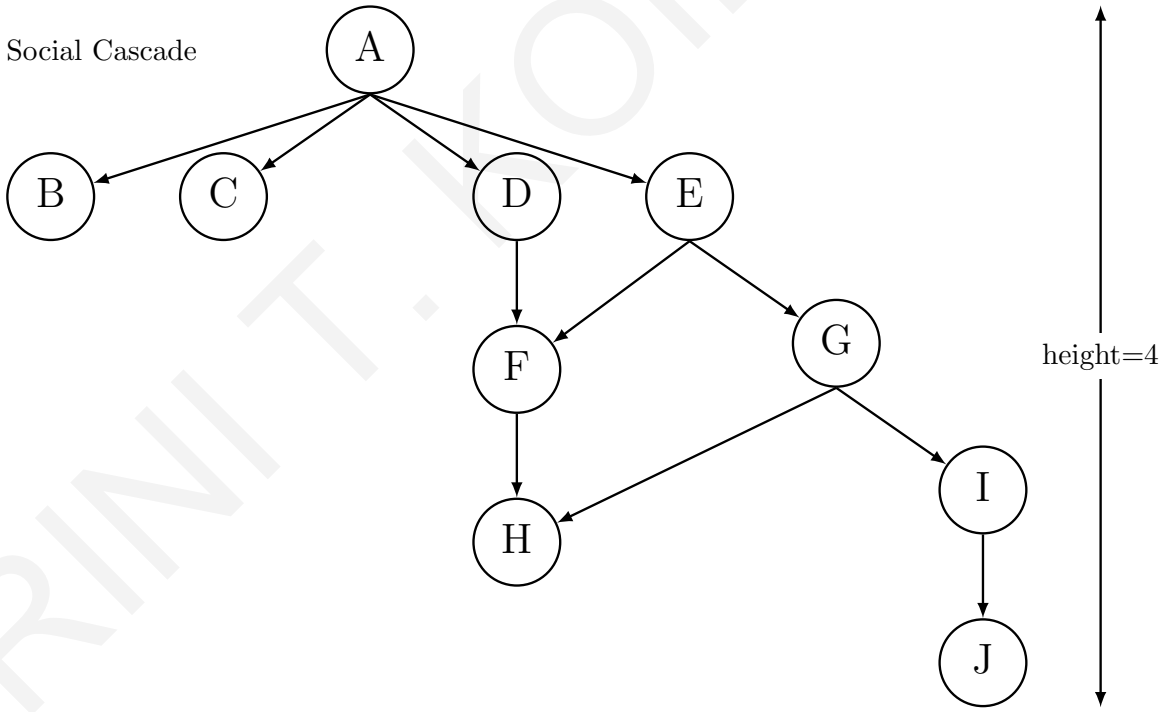


Figure 3: The evolution of a social cascade in Twitter

A specific case of information diffusion is an *information* or *social cascade* (also mentioned as "herding"). We say that a social cascade has occurred when somebody, regardless of his own private information (e.g. concerning the selection of a restaurant in an unfamiliar town), finds it rational to join the crowd [41]. Practically, a cascade happens in a social

network, when a piece of information is extensively retransmitted after its initial publication from a user. Cascades can be represented as rooted directed trees where the initiator of the cascade is the root of the tree [39]. The *length* of the cascade is the height of the resulting tree. Each vertex in the cascade tree can have the information of the user, and the identity of the item replicated in the cascade.

Figure 3 depicts an example of the evolution of a social cascade in a directed graph. The cascade follows the arrows' direction. In the Twitter social network, for example, B, C, D, E are followers of A, denoted by the directional links from A to B, C, D, E. The adopters of a new information piece could be the nodes, that after having been exposed in a piece of information, the link of a YouTube video, for example, they retweet it. User A tweets the URL of a YouTube video and exposes B, C, D and E to this piece of information. B, C, D and E adopt the information. B and C have no followers, but the cascade procedes with D and E. The cascade procedes with nodes F and G and so on so forth until it stops with node J. For F that receives the same information from D and E, we assume that its influence is ascribed to the one (between D and E) that tweeted the piece of information earlier.

The *diffusion* of information in a network is essentially interweaved with whether a piece of information will become eventually popular or its spread will die out quickly. A large proportion of bandwidth-intensive media is distributed via OSNs' links (for example YouTube videos links through retweets in Twitter), that contribute significantly to Internet traffic [2]. Facebook and Twitter users increasingly repost links they have received from others. Thus, they contribute to *social cascades* phenomena, a specific case of information diffusion that occurs in a social network, when a piece of information is extensively retransmitted after its initial publication from an originator. Therefore, it would be beneficial to know when such cascades happen in order to proactively replicate popular items (*prefetching*) via CDN infrastructures. More precisely, it would be useful to know if they are expected to depict a *global* or *local* range, so that this content can be replicated either in the geographic local area of the user or in the whole network.

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 ([9], [156], [136]), 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 [26]). The wide use of OSNs and the growing popularity of streaming media have been characterized as the primary causes behind the increases in Hypertext Transfer Protocol (HTTP) traffic observed in measurement studies 2.5.2. 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 [2]. 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 ([19], [20] [52]).

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 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 [99]. 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 [99].

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

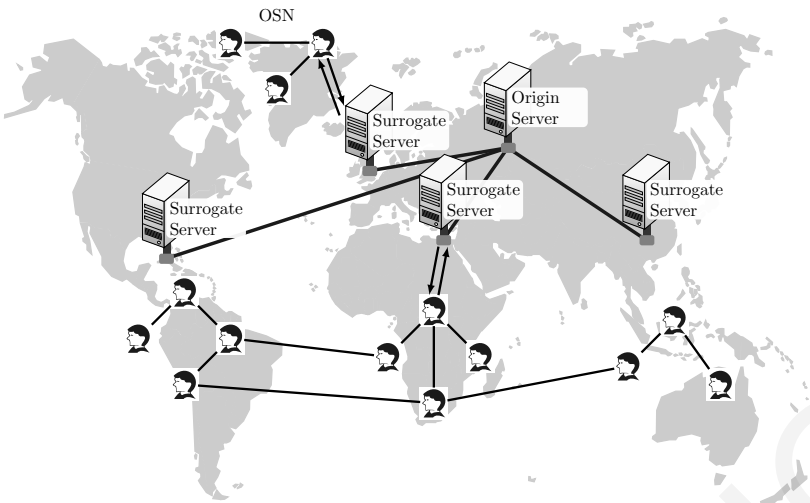


Figure 4: An example of a CDN of an OSN.

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 [34].

1.2.1 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. 4). 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. 5, 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

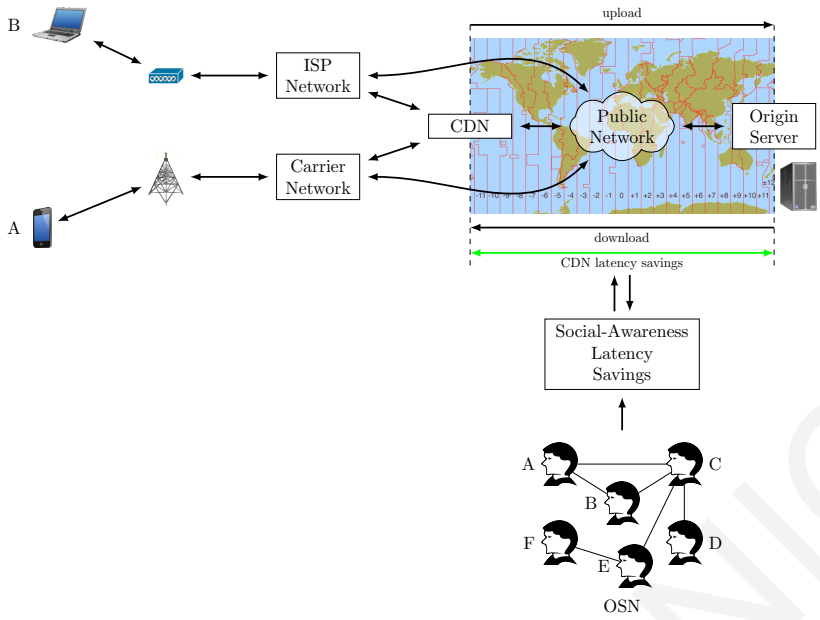


Figure 5: Latency savings through the proposed mechanism.

as Alice, depicted as A in Fig. 5, 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 [158], 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.

1.3 Challenges

In order to utilize information diffusion over CDN infrastructure, a diffusion amplified by the proliferation of smartphones and cheap broadband connections, we have to efficiently deal with the size of accommodated social graphs. The amount of information in OSNs is an obstacle, since elaborate manipulation of the data may be needed. An open problem is the efficient handling of graphs with billions of nodes and edges [190]. Facebook, for example, reported that 1.13 billion daily active users on average for June 2016 [5].

In order to generate aggregations and analyses that have meaning, the Facebook custom built-in data warehouse and analytics infrastructure [21] has to apply ad-hoc queries and custom MapReduce jobs [68] in a continuous basis on over half a petabyte of new data every 24 hours, with the largest cluster containing more than 100PB of data and the process needs surpassing the 60.000 queries in Hive, the data warehouse system for Hadoop [16] and Hadoop compatible file systems. (MapReduce [68] is Google's framework (programming

model and implementation) for processing of parallelizable tasks over large datasets by a large number of computers / nodes (referred to as a cluster or grid depending on their location and homogeneity of resources), based on the division of a problem to smaller sub-problems (map phase) and the composition of their outputs (reduce phase), that hides the details of parallelization, and encompasses fault tolerance, locality optimization, and load balancing.)

The desired scaling property refers to the fact that the throughput of the proposed approaches should remain unchanged with the increase in the data input size, such as the large datasets that social graphs comprise and the social cascades phenomena that amplify the situation. Cost of scaling such content can be expressed in different ways. For instance, in case of CDNs, can be the number of replicas needed for a specific source, may take into account the optimal use of memory and processing time of a OSN-aware built system, etc.

1.4 Thesis Statement

The main point in our thesis is that information extracted from Online Social Networks (OSNs) can be used to facilitate proactive content caching decisions, and thereby build better Content Delivery infrastructures.

This claim is supported by three case studies, which apply social information in situations where traditional bandwidth-intensive content scaling is infeasible. Our approach is based on experimentation with empirical traces to identify relevant social properties and is followed by incorporation of the latter into current Content Delivery infrastructures.

The very few approaches that have been proposed lack experimental evaluation with non-synthetic workloads for the development of OSN-aware content staging mechanisms, and moreover ignore storage issues of the infrastructure and matters such as refined topology of data centers. However, the engineering of general OSN-aware content placement over a CDN infrastructure is an important and challenging endeavor that has received less attention.

In this work, we address the following question: "Can we find an efficient content prefetching policy that takes as input a non-synthetic workload of data from OSNs and actions of users over them and that recognizes objects likely to be popular?" To answer this question, we perform experiments over a large corpus of YouTube videos. For capturing the social cascading effects, we use Twitter, one of the most popular OSNs with its core functionality centered around the idea of spreading information by word-of-mouth [159]. Specifically, Twitter provides mechanisms such as retweet (act of forwarding other people's

tweets), which enable users to propagate information across multiple hops in the network through cascading.

Overall, this Thesis provides the following contributions:

- We present a taxonomy of existing OSN-aware approaches that can be leveraged for the scaling of bandwidth-demanding media content in CDNs [112]. We take into account phenomena related with media content and its outspread via OSNs, measurement studies on OSNs that could provide valuable insight into CDN infrastructure decisions for the replication of the content, as well as systems built with the leverage of OSN data.

- The Thesis introduces the Social Prefetcher algorithm and theoretically analyzes it through an optimization analysis to prove that it provides a near-optimal solution. Addressing memory usage issues related to the very large graph dataset accommodated, it incorporates the algorithm implementation in a validated simulator for CDNs, suggesting the accompanying framework within a content delivery infrastructure. In the Thesis we conduct a multitude of experiments, and we reach some fruitful conclusions regarding how OSNs can affect the content delivery performance. The Social Prefetcher surpasses performance improvement (30%) of similar works in pull-based methods [176], that are employed by most CDNs, and depicts better response times when various approaches are implemented within the suggested framework [164]. Moreover our algorithm uses more refined topology of data centers and does not neglect storage issues [164], 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. We aim in the future to generalize our proposed policies to deal with multiple OSN platforms, as well as mobile CDN providers.

- Experimentation is conducted on a Twitter dataset [64] 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, 1).

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 1: Twitter dataset tweet 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

Table 2: Twitter dataset user information

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. Vine [22], 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.

- The sequence of content requests created to the CDN directly from the Twitter messages within the dataset in [165] 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. [1] A non-synthetic dataset from multimedia links spread over an OSN platform is used to directly analyze social cascades and access to social profiles is not conducted via a third-party page [164].

- Our approach consists a paradigm of augmentation of an existing stand-alone tool for CDN simulations with an OSN-awareness mechanism. With the extension proposed herein, we can experiment on social patterns without 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. To the best of our knowledge, no similar extensions exist towards this direction.

- We furthermore [115] modify the Social Prefetcher algorithm [110] to incorporate best performing caching mechanisms. We examine which caching schemes in the surrogate server affect the CDN metrics the most. We further examine which parameters (number of timezones examined, time threshold duration etc.) affect the CDN metrics the most. Works extending the Social Prefetcher algorithm [110] to include information about peak-time of various timezones of our geo-diverse system, as well as contextual information about the viewership of video content within the media service [111] go beyond the Social Prefetcher algorithm in terms of performance. 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 diffusion of video content is fostered by the ease of producing online content via media services and mainly happens via ubiquitous OSNs, where users increasingly repost links they have received from others (social cascades). If we knew beforehand when a social cascade will happen or to what range it will evolve, we could exploit this knowledge in various ways, sparing bandwidth in the area of content delivery infrastructure, where popular items would be proactively replicated. Our work focuses on video virality over an OSN and combines detailed information both of the OSN

[1] G.Christodoulou implemented the accommodated generator, described in the Appendix.

and the media service. We propose an accurate prediction model with a small and easily extracted feature set that performs better than methods like Support Vector Machines (SVM), Stochastic Gradient Descent (SGD) and K-Nearest Neighbours (KNN), among others [113]. We, furthermore, proceed to incorporate the model into a mechanism for content delivery depicting substantial improvement for the user experience.

1.4.1 Publications

- I. Kilanioti, Improving multimedia content delivery via augmentation with social information. The Social Prefetcher approach. Multimedia, IEEE Transactions on, vol. 17, no. 9, pp. 14601470, 2015. [110]

- I. Kilanioti, C. Georgiou, and G. Pallis, On the impact of online social networks in content delivery, in Advanced Content Delivery and Streaming in the Cloud, M. Pathan, R. Sitaraman, and D. Robinson, Eds. Wiley, 2014. [112]

- I. Kilanioti and G. A. Papadopoulos, Socially-aware multimedia content delivery for the cloud, in 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), Dec 2015, pp. 300309. [111]

- I. Kilanioti and G. A. Papadopoulos, Efficient content delivery through popularity forecasting on social media, in Proceedings of the 7th IEEE International Conference on Information, Intelligence, Systems and Applications, IISA 2016, Chalkidiki, Greece, July 13-15, 2016, pp. 1319. [113]

- I. Kilanioti and G. A. Papadopoulos, Delivering social multimedia content with scalability, in Resource Management for Big Data Platforms: Algorithms, Modelling and High-Performance Computing Techniques, Springer Computer Communications and Networks Series, J. K. Florin Pop and B. di Martino, Eds. Springer, 2016, ch. 18. [115]

- I. Kilanioti and G. A. Papadopoulos, Predicting video virality on Twitter, in Resource Management for Big Data Platforms: Algorithms, Modelling and High-Performance Computing Techniques, Springer Computer Communications and Networks Series, J. K. Florin Pop and B. di Martino, Eds. Springer, 2016, ch. 20. [116]

- I. Kilanioti, G.A. Papadopoulos, Content Delivery Simulations supported by Social Network-awareness, Simulation Modelling Practice and Theory Journal - SIMPAT Elsevier, ChipSet Special Issue, under review [114]

1.5 Thesis Structure

Network infrastructures such as CDNs often need to prefetch bandwidth-intensive content for consumers of a plethora of applications, including elearning, healthcare, ecommerce, gaming, social activities, smart cities and environment management. The performance of existent solutions can be insufficient in terms of latency the consumers of content experience, as well as bandwidth the media providers need.

Our aim in this chapter is to establish a common understanding of the basic concepts related to content delivery used in this Thesis in general, as well as social networks and the circulation of media objects over them in particular. In this respect, a few relevant definitions are first provided, followed by a characterization of social cascades features. These facilitate the establishment of a point of reference for understanding the problems investigated in this Thesis.

In our first study (Chapter 3) the pursuit lies in exploiting the user activity extracted from OSNs to improve the content prefetching mechanism of multimedia user-generated content, as a significantly large proportion of HTTP traffic results from bandwidth-intensive multimedia content circulating through OSNs and multimedia content providers, such as YouTube, often rely on CDN infrastructures. Aiming to reduce bandwidth usage, we incorporated a dynamic mechanism to a stand-alone CDN traffic simulator. This mechanism is based on a dynamic policy that takes patterns of information transmission over OSNs into account. Thereby, we demonstrate that the performance of CDNs can be improved, and the cost of copying to surrogate servers is taken into consideration. For the staging of content we use geo-social properties of users participating in social cascades that prove to be of great importance toward improving the performance of CDNs and cloud, in the long-term. Unpopular long-tail user-generated items often go viral and are localized in a part of the OSN, whereas at the same time social cascades are observed to have a short duration. Exploiting the above, we propose the Social Prefetcher, an approach that decreases replication costs by selectively copying items to locations where items are likely to be consumed, thus saving bandwidth and time.

Extensions of this work include caching schemes variations (Chapter 5) as well as further study of the issue of temporal diffusion, related to the most efficient timing of the content placement (Chapter 4). We exploit the knowledge of peak times for upload and download, so that content is prefetched in the hours with less traffic. We also incorporate other contextual information, such as the viewership within the media service, to ensure performance

optimization. Our variations are experimentally proven to contribute toward maximization of CDNs performance and reduction of content replication costs.

The second study examines merging of user-centric data from Twitter, one of the most popular OSNs with its core functionality centered around the idea of spreading information with mechanisms such as retweet, with video-centric data from YouTube. We aim at investigating the ties between predictability of video sharing and the social context of video uploaders. With the combination of social media datasets used, we obtain insights than neither employed dataset individually gives. We develop an accurate model to predict future popularity of a video resource given features of the underlying network of its initiator. The set of features we propose and analysed are based on the notion of influence score of a user and its fluctuation through time, as well as the distance of content interests among users for both datasets. We discover that the latter feature plays an important role in video popularity prediction, suggesting high dependence of video sharing via Twitter on the video content itself.

We finish by discussing a study integrating the second study with a Content Delivery infrastructure to reap possible benefits in terms of user experience and providers' costs.

The Thesis concludes with a discussion of its main contributions and with a suggestion of key topics for future research.

Chapter 2

Related Work

Our aim in this chapter is to present existing OSN-aware approaches that can be leveraged for the scaling of bandwidth-demanding media content in CDNs. In particular, we present a taxonomy of the relative research, taking into account phenomena related with media content and its outspread via OSNs, measurement studies on OSNs that could provide valuable insight into CDN infrastructure decisions for the replication of the content, as well as systems built with the leverage of OSN data.

The remainder of the chapter is organized as follows: A categorization of most predominant models for the depiction of the information diffusion process in a social network is presented in Section 2.1 (*Models of Information Diffusion*). Section 2.2 (*Tools*) discusses various tools for experimenting with scaling of bandwidth-intensive-media content using OSNs. Metrics for the characterization of the outspread of cascades are described in Section 2.4 (*Metrics*). The association of bandwidth-intensive media content diffusion with social networks is given in Section 2.5 (*Bandwidth-intensive media content and Social Networks*), where measurements for social networks are also presented with some generic assumptions on the OSNs structure (*Structure of OSNs*), and various studies that corroborate them (*Measurement studies*). Moreover, the way topological features of an OSN may affect the information diffusion is discussed. The association of bandwidth-intensive media content with social networks, as well as a description of works studying patterns of user requests over OSNs for various Video-on-Demand (VOD) systems and OSNs follow in subsection 2.5.3 (*Bandwidth-intensive media content diffusion over OSNs*). The Section gives finally the outline of existent works (*Applications and techniques*) that exploit information extracted from OSNs for the improvement of user experience.

2.1 Models of Information Diffusion

This section describes the most predominant models for the depiction of the information diffusion process in a social network, on which most of the existent algorithmic solutions are built. The main algorithmic problems studied in the bibliography are related with the discovery of nodes that are most prone to diffuse an information to the greatest extent, and the categorization of nodes according to their influence degree. The categorization of the models is depicted in Figure 6. The first-level discrimination of models is based on whereas they take the structure of the network into consideration (*network-aware*) or not (*holistic*). In other words the discrimination criterion is if they incorporate knowledge about underlying associations of the nodes (friendships) or, to the contrary, follow an aggregate-level approach.

In our Thesis we propose a simple feature-based empirical model (6.3), which combines data from a user-centric and a video-centric platform, whereas the main proposed algorithm (3.3) includes heuristics derived from empirical models and augments them with geographic information.

2.1.1 Holistic view models

Rogers' theory [161] (Section 1.1) is quantified by the Bass model [43]. The Bass model is based on the notion that "the probability of adopting by those who have not yet adopted is a linear function of those who had previously adopted" (F.Bass). It predicts the number of adopters $n(t) \in N$ of an innovation at time t (in the information diffusion scenario the number of retransmitters of an information piece):

$$n(t) = pM + (q-p)N(t) - q/M\,(\,N(t))^2 \tag{1}$$

where N(t) is the cumulative number of adopters by time t, M is the potential market (the ultimate number of adopters), $p \in [0,1]$ is the coefficient of innovation (the external influences, expressing the individuals influenced by the mass media), and q is the coefficient of imitation (internal influence, expressing the individuals influenced by the early adopters). This approach, however, largely ignores the underlying network structure.

Models under the same concept of holistic view of the social behavior make use of differential equations, and include, among others, the multi step flow model by Katz and Lazarsfeld [107], the Daley-Kendall rumours model [66], and also, more recent ones, such as, the Van den Bulte and Joshi model of influentials and imitators [180].

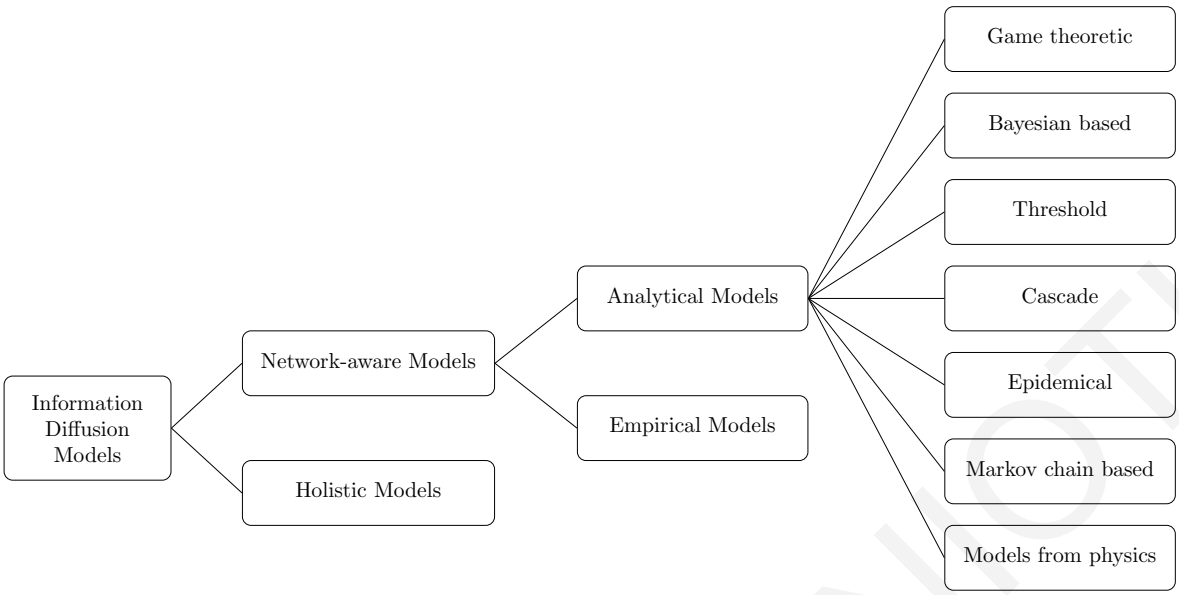


Figure 6: Information Diffusion Models

2.1.2 Network-aware models

These include completely novel models, but also variations of the afore-mentioned (holistic) models, such as the Nekovee variation [150] of the Daley-Kendall model, and are separated in the two following categories, based on whether they are mathematically formulated (*Analytical models*) and then applied or are the outcome of empirical methods, such as regression, regression trees etc. (*Empirical models*).

2.1.2.1 Analytical models

The first mathematical models based on nodes' thresholds for the depiction of information diffusion were developed by Schelling [166] and Granovetter [91]. There follows a categorization of the most predominant ones.

Game theoretic models

In [117], Kleinberg proposes a simple networked coordination games model. The model is based on the notion that for each individual $v \in V$ in the graph G=(V,E) the benefits of adopting a new behaviour increase as more of its neighbours adopt the new behaviour. We assume that there are two behaviours a node can follow, the "old" one (A) and a "new" one (B). At discrete time steps t=1,2,3, ... each node updates its choice of A or B according to the behaviour of its neighbours. The objective of the nodes is to switch each time to the behaviour that reaps the maximum benefit for them. This means that v, based on his

own interest, should adopt the new behaviour after a satisfactory fraction of its neighbours have already adopted. For the nodes v and w there is a motivation for behaviour matching, expressed in the following way, where parameter q is a real number $0 < q < 1$:

-if v and w both choose behaviour A, they both receive a q payoff

-if v and w both choose behaviour B, they both receive a 1-q payoff

-if v and w choose different behaviours, they both receive a 0 payoff

The overall payoff for v playing the game with its neighbours in G is the sum of the individual (pairwise) payoff; q is actually the threshold expressing the fraction of adopting neighbours, since it easily results that v should adopt behaviour B if $d_v^B > qd_v$, and A if $d_v^B < qd_v$, where d_v is the degree of the node, d_v^A the number of its neighbours with behaviour A and d_v^B the number of its neighbours with behaviour B. (v's payoff for choosing A is qd_v^A and for choosing B is $(1-q)d_v^B$)

Initially there is a set S of nodes adopting behaviour B and $h_q(S)$ is the set of nodes adopting B after one round of updating with threshold q, $h_q^k(S)$ the nodes adopting B after k successive rounds. A set S is *contagious* (with respect to h_q) if "a new behaviour originating at S eventually spreads to the full set of nodes" and the contagion threshold of a social network G is "the maximum q for which there exists a finite contagious set" (how big can q be and still small sets impose new behaviours to all).

The technical issue of *progressive* or *non-progressive processes* (monotonous or non-monotonous as referred to later on in the present study) refers to the fact that when in time step t a node v following till then the behaviour A updates to behaviour B, it will be following B in all subsequent time steps. Although, intuitively, we would expect progressive processes to give finite contagious sets more easily (because of lack of early adopters setbacks that would hinder the cascade), Kleinberg points out that both the progressive and non-progressive models have the same contagion thresholds [145], which in both cases is at most 1/2 ("a behaviour can't spread very far if it requires a strict majority of your friends to adopt it") [145].

More game-theoretic models can be found in the work of Arthur [35], who proposes a simple cascade model of sequential decisions with positive externalities, manifested by a term that adds to the payoff of a decision. Namely in the scenario of two competing products, the latter become more valuable as more users use them (for a social media site or a smartphone, for example, it will aquire better third-party applications and support as its users grow). Also game-theoretic models are introduced by Banerjee [41] and Bikhchandani et al. [47], that are based on influence not due to positive externalities, but because of information

conveyed from earlier decisions. The proposed game-theoretic models, however, have the drawback of not taking heterogeneity into consideration, in the notion that all nodes have the same threshold, and all their neighbours contribute the same in making a node change its behaviour.

The basic propagation models on which most generalizations for information diffusion are based are the Linear Threshold Model (LTM) [91], [166], [185] and the Independent Cascade Model (ICM) [89] with many proposed extensions (LTM: [185], [109], ICM: [89], [109], [93]) and also a proposed unification [109].

Linear Threshold Model

Based on the assumption that some node can be either active (adopts a new idea / transmits a piece of information) or inactive and taking into account the monotonicity assumption, namely that nodes can turn from inactive to active with the pass of time but not the opposite, we can say that the LTM is based on the following notion: Each node v has a predefined activation threshold $\theta_v \in [0,1]$, which expresses how difficult it is for the node to be influenced when its neighbors are active ("the weighted fraction of the neighbors of node that must become active in order for node to become active"), and is influenced by each one of its neighbors w according to a weight b_{vw}, so that $\sum_{w\in\Gamma(v)} b_{vw} \leq 1$. The thresholds can be produced randomly with a uniform distribution, but some approaches investigate a uniform threshold for all the nodes of the network, e.g. [46].The process takes place in discrete steps and the nodes that satisfy the constraint $\sum_{w\in\Gamma(v)} b_{vw} > \theta_v$ are gradually added as active to the initial set of nodes. It's worth mentioning that LTM can result as a modification of the networked coordinations game referred in the previous paragraph with the differentiation of payoffs for different pairs of nodes.

LTM expresses the idea that the influence of the neighbours of a node is additive, but when the rule of influence can not be expressed by a simple weighed sum, for example a node becomes active when one of its acquaintances and two of its co-workers do so, the arbitrary function g_v substitutes the weighed sum. In the *General Threshold Model* for time steps $t = 1,2,3...$ a node v becomes active if the set of active neighbours at t satisfy $g_v(X) > \theta_v$.

Independent Cascade Model

Under the ICM model [89], there is also a set of initially active nodes, the process takes place in discrete steps, but when node v becomes active, it has only one chance of activating each of its inactive neighbors w until the end of the process with a probability p_{vw} independent of the activations history and with an arbitrary order.

Exact evaluation of activation probabilities is exponential to the number of edges of the graph. Improving the performance of the works in [91] and [167], there are works studying the calculation of these probabilities such as [90](based on a General Threshold Model with the assumption that each parent's influence is fixed), or [69] (based on the ICM). In the latter, sampling from the twiter dataset is conducted in an efficient Markov-Chain Monte Carlo fashion using the Metropolis-Hastings algorithm [62] and the problem is tackled with two differentiations, one of which considering the past paths of data known (retweets for the twitter dataset) and one considering only the past path endpoints known (hashtags and urls) and joint probabilities are taken into consideration, reflecting also model uncertainty.

Epidemical models

In the case of epidemical models a single activated ("infected") node causes the change of state of a neighbour susceptible node, whereas in the afore-mentioned threshold and game-theoretic models a node has to interact with multiple neighbour nodes to evolve (complex contagion).

Epidemical models were introduced on the assumption that information would propagate like diseases. They constitute another category with an almost straightforward pairing with the ICM. The ICM captures the notion of contagion more directly, and also allows us to incorporate the idea that a node's receptiveness to influence does not depend on the past history of interactions with its neighbors.

Epidemical models variations include the simple branching processes model, where a node infects a number of nodes and the contagion proceeds in subsequent waves with a probability π. This model is characterized by the basic reproductive number of the disease $R_0 = k\pi$, where k is the number of new people somebody meets, which expresses the anticipated number of new cases of the disease that a single node will cause.

Extensions of the epidemical models are the SIR, SIS, and SIRS models: S stands for susceptible nodes, nodes that have not been infected yet and have no immunity to the contagion. I stands for infected nodes, nodes contagious to their susceptible neighbours, and R

Name	Meaning	Scope
$n(t) \in \mathbb{N}$	number of adopters of an innovation at time t	Bass Model
$N(t)$	cumulative number of adopters at time t	
M	potential market (ultimate number of adopters)	
$p \in [0,1]$	coefficient of imitation	
v,w	nodes participating in the game	Kleinberg game theoretic model
q	payoff for the game	
d_v^A	degree of node A	
d_v^B	degree of node B	
b_{vw}	influence of node v by neighbour w	Linear Threshold Model
θ	activation threshold	
X	set of active neighbours at t	
g_v	generalization of sum function in the case of LTM	general threshold model
p_{vw}	probability that active node v will activate inactive neighbour w	Independent Cascade Model
R_0	basic reproductive number	epidemical model
k	number of new people somebody meets	
π	probability that the contagion proceeds in subsequent waves	
K, B	states of the world	Bayes based model
p	probability that world is placed in state K	
v_g, v_b	payoffs for node v adopting a good/ bad idea respectively	
H, L	signals denoting that adoption is a good or bad idea respectively	
G_r	ordering of common shapes , r frequency ranking	
Ξ=$\xi_1, \xi_2, ..., \xi_r$	set of possible states	Markov chain models
p_{ij}	probability of transition from state ξ_i to state ξ_j	
$G(V, w, \tau)$	G contains a set V of n nodes and E edges between nodes representing the information diffusion paths. w denotes the set of the edges' weights ,τ the set of the time delay on the information diffusion paths.	Continuous-Time Markov Chain Model

Table 3: Models - Notation Overview 1

stands for recovered nodes, with the recovery considered as permanent in SIR and temporary in the case of SIRS [123]. The sequence of the letters in the acronyms of the models explains

the flow of the epidemic. In SIR model nodes pass from the state of being susceptible to the state of being infected and then recover. In SIS model nodes are immediately susceptible once they have recovered (like in the case of common cold, recovery does not imply immunity that lasts for long). In the SIRS model recovered nodes free of infection may rejoin the susceptible nodes.

Bayes based models

Combining nodes' private information and their observations of earlier adoptions, in [76], Kleinberg and Easley present a Bayes based model to formulate information cascades, answering questions such as "What is the probability this is the best restaurant given the reviews I have read and the crowds I see there?".

$$Pr[A|B] = \frac{Pr[A]\,Pr[B|A]}{Pr[B]} \tag{2}$$

Three factors are taken into consideration:

- The states of the world;
- Payoffs; and
- Signals.

The first factor expresses whether an option is good or bad (if a new restaurant is a good or a bad choice). Supposing that the two options of the world are K (the option is a good idea) and B (the option is a bad idea), the world is placed in K with probability p and in B with probability $1-p$ ($Pr[K] = p$, $Pr[B] = 1 - Pr[K] = 1 - p$). Payoffs for a node v are defined as follows: -If v rejects the option, the payoff is 0.
-If v adopts a good idea, it receives a positive $v_g > 0$ payoff.
-If v adopts a bad idea, it receives a negative $v_b > 0$ payoff.
-If v adopts without any prior knowledge, the payoff is 0.

The signals refer to private information each individual gets about the benefit or not of a decision: a high signal (H) suggests that adoption is a good idea, whereas a low signal (L) suggests that it is a bad idea. If accepting is indeed a good idea, then $Pr[H|K] = q > \frac{1}{2}$ and $Pr[H|K] = 1 - q < \frac{1}{2}$. In the restaurant example the private information could be a review

that an individual reads about the first restaurant, with a high signal corresponding to a review comparing it favorably to restaurant B. If choosing the first restaurant is indeed good, there should be a higher number of such reviews, so $Pr[H|K] = q > \frac{1}{2}$.
Kleinberg and Easley [76] consider how individual decisions are made using (Eq.2.1.2.1) when they get a sequence of independently generated signals consisting of a number of high signals and a number of low signals, thus making interesting observations about situations where individuals can observe others' earlier decision, but do not have access to their knowledge.

Markov chain models

Markov chains [70] are used to describe transitions from one state of a system to another in a finite set of possible states. Their memoryless nature (*Markov property*) has to do with the fact that the next state each time is independent of the preceding states. More formally: With a set of states $\{\xi_1, \xi_2, .., \xi_r\}$ the process moves successively from one state to another in so-called *steps*, and specifically from state ξ_i to state ξ_j with a probability p_{ij} (*transition probability*) independent of the previous states of the chains, or remains in the same state with a probability p_{ii}. A particular state is picked from Ξ as the initial state. Markov chains are usually depicted with a directed graph, where the edges' labels denote the transition probabilities.

Markov models are widely used for analysing the web navigation of users. PageRank [51] is based on a Markov model and is used for ranking of information in the Word Wide Web. By assigning weights that denote the relative importance of an hyperlinked document in a set of documents, the likelihood that a person will reach a specific page through random clicks is, essentially, represented.

In [167], Song et al. use a Continuous-Time Markov Chain Model (CTMC), namely a Markov model that describes the transition among states after some time of stay in a particular state. This time is exponentially distributed and does not affect the transition probability to the next state. The information diffusion model is introduced on a network $G(V, w, \tau)$. G contains a set V of n nodes and E edges between nodes representing the information diffusion paths. w denotes the set of the edges' weights ("amount of information to flow from one node to another") and τ the set of the time delay on the information diffusion paths. Thus, the representation of the graph matches the CTMC in the notion that each node represents a

state, each weight a transition probability and the delay is represented as the time-to-stay in each state.

Name	Meaning	Scope
$f_{t+1}(v)$, initial assignment f_0		Voter model
$\sigma_i = \pm 1$	state of an atom in the network	Ising model
$\sigma = (\sigma_1, ..., \sigma_N)$	configuration for N atoms	
H, s_g	total energy of the system, lowest energy configuration	
J	"external magnetic field" coefficient	
E	"nearest-neighbours interaction" coefficient	
f	probability a node is initially active	bootstrap percolation
k (k = 2, 3, ...)	number of nearest neighbors required to be active	
w, v	w is a child of v if w appends its name in the copy of the letter directly below v	chain-letter Liben-Nowell and Kleinberg model
t	time each recipient waits before action	
β	probability with which a recipient can group-reply to his corecipients	
c_i	cost function of the number of followers per individual i	Bakshy
a	acquisition cost	
c_f	cost per follower	
f_i	number of followers	
$I_u(l)$	number of mentions of an information for a node u l time units after the node u adopted the information (at t_u)	Yang and Leskovec model for the global influence of a node through the whole network
$V(t)$	number of nodes that mention the information at time t	
x_i	time series	
$A(t)$	nodes that got activated before t	

Table 4: Models - Notation Overview 2

Voter model

The basic voter model introduced by Clifford and Sudbury [65] and Holley and Liggett [96], is defined in an undirected network and allows the spread of two opinions. In discrete

time steps, a node adopts the opinion of a randomly chosen neighbour. For a node $v \in V$ in graph $G = (V, E)$, $\Gamma(v)$ is the set of neighbors of v in G and initially the nodes are arbitrarily endowed with a 0/1 state. At time step t each node adopts the opinion of one uniformly picked neighbour. With an initial assignment $f_0 : V \to \{0, 1\}$ inductively we define

$$f_{t+1}(v) = \begin{cases} 1, with \quad probability \quad a \\ 0, with \quad probability \quad b \end{cases} \tag{3}$$

where $a = \frac{|\{u \in \Gamma(v) : f_t(u) = 1\}|}{|\Gamma(v)|}$ and $b = \frac{|\{u \in \Gamma(v) : f_t(u) = 0\}|}{|\Gamma(v)|}$.

Even-Dar and Shapira [78] argue that it is one of the most natural probabilistic models to capture the information diffusion in a social network. It is suitable for depicting the spread of a technological product, as it is proved that under this model consensus is reached with probability 1. Even-Dar and Shapira refer to the (almost) consensus of products such as Google as a search engine, YouTube as a video-sharing website etc.

Models from physics

Models from physics include the Ising model [101] serving for the description of magnetic systems, and bootstrap percolation [31] serving for the description of magnetic systems, neuronal activity, glassy dynamics, etc.

Ising model The Ising model [101] was first proposed in statistical physics and encompasses the notion of a ground state (in physics the state with the minimum energy), and that of the "self-optimizing" nature of the network.

Similarly to the basic voter model, there can be two competing "opinions", in favour of or against a subject, let's say depicted by a "+1" and a "-1", which in physics express the correspondence of an atom forming a network to a spin variable (can be considered as the basic unit of magnetization) state $\sigma_i = \pm 1$. The total energy of the system under this model (Hamiltonian) is defined as:

$$H = H(\sigma) = -\sum_{<i,j>} E\sigma_i\sigma_j - \sum_i J\sigma_i \tag{4}$$

for each configuration $\sigma = (\sigma_1, ..., \sigma_N)$, with the parameter J associated with an "external magnetic field" and E with the "nearest-neighbours interaction", N the number of the atoms.

Name	Meaning	Scope
$G_\cap$	largest common subgraph of all snapshots in cluster C	Ren et al.
$G_\cup$	smallest common supergraph of all snapshots in C	
ges	graph edit similarity between two graphs	
$\tilde{P}_*(u,v)$ in a graph G_*, where $* = 1,2,...,n,\cap,\cup$	shortest-path between the vertices v and u	
Δ	difference between two consecutive snapshots	
$SM1(C), SM2(C), SM\_FVF(C)$	storage schemes	
n_i	node of the social network	Problem formulation
$\Gamma(n_i)$	set of neighbours of node n_i	
$C_z(m, n_i, O_k, T_c, G_{t_l}, M)$	Cascade of topic T_c initiated by node n_i on object O_k, G_{t_l}: the first snapshot where the cascade was detected, under model of diffusion m, M: array of cascade metrics	
ϕ	Fraction of the network that the object is bound to spread	
ω	Threshold above which a cascade is characterized as global, and under which as local (constant)	
$\theta_i(C)$	Activation threshold of a specific node i per cascade	
O_k	Object to be replicated	
$s_{t_x} = \{0,1\}$	State of a node (inactive/active) at time point t_x	
N_{ser}	Number of surrogate servers	
N_c	Number of clients	
$S = [(S_1,l_1),(S_2,l_2),...,(S_{N_{ser}},l_{N_{ser}})]$	Topology of surrogate servers, with $l_1, l_2, ..., l_{N_{ser}}$ their locations	

Table 5: Models - Notation Overview 3

The ground state is the lowest energy configuration s_g (in physics the zero temperature configuration), so that $s_g \in argmin_s H(s)$. In a social network can be seen as the state with the most likely opinion, minimizing conflicts among its members (atoms).

Bootstrap percolation In the standard bootstrap percolation process [31] a node is initially either active with a given probability f or inactive. It becomes active if k (k = 2, 3, ...) of its nearest neighbors are active. In that notion it resembles the k-core problem of random graphs [134], where k-core is the maximal subgraph within which all vertices have at least k neighbors, but whereas bootstrap percolation starts from a subset of seed vertices according to the above-mentioned activation rule, the k-core of the network can be found by a subsequent pruning of vertices which have less than k neighbors.

Name	Meaning	Scope
$P(O_k, C_z) = [P_1, P_2, ..., P_w]$	Places of proactive replication	
$t_p(N_{ser}, N_c, S)$	Amount of traffic traversing path p between client and the server finally serving the request	
$c = f_2(t_p, \phi, N_c)$	Cost of replicating a single object, N_c is taken into account when $\phi < \omega$	
f_1	classification function for a cascade C_z:	Metrics for characterization of cascades
ϖ	threshold for characterization	
p_k	degree distribution of the graph comprising of n nodes	Watts
z	average node degree	
$f(\phi)$	threshold distribution	
$reach^a(u)$	the number of cascades a user can reach with a specific action α	Dave et al.
$\vec{P}^a(u)$	the propagation set of u, consisting of all his immediate neighbours u_i, such that there was an action propagation from u to u_i	
δ	density of a cluster	Kleinberg and Easley

Table 6: Models - Notation Overview 4

2.1.2.2 Empirical models

Before the advent of machine-readable traces, the potential of networks in the transmission of information and messages was stated already by Milgram in his renowned experiment [139] or Christakis [82], who suggested in a study of 12000 participants that risks, such as the risk of becoming obese or benefits, such as stopping of smoking, are propagated through social ties. However, it is large scale and time-resolved machine-readable traces that, through the step-by-step track of interactions in OSNs (although not compulsorily easily accessible/ collectible), have driven to the formulation of a plethora of empirical models.

Some generic observations concerning the empirical models are the following. Many of them lack insight of information content, unlike works, such as that of Huberman et al. [37], who formulate a model taking into consideration solely the features of an information item (a news item in Twitter). Sometimes the discovered patterns in empirical models are at odds with the predictions based on theoretical (analytical) models. For example, in unison with the epidemical model, Leskovec et al. in [130] claim that cascades (depicting the blogosphere information diffusion) are mostly tree-like. More specifically, they notice that the number of edges in the cascade increases almost linearly with the number of nodes, suggesting that

Name	Meaning	Scope
R_0	basic reproductive number	Ver Steeg et al.
β	transmission rate	
γ	infection duration	
k	node degree	
σ_0	the probability that a person will adopt the shared piece of information	
s	size	
λ	length	[39]
t_d	time delay	[165]
d	time duration	
r	rate	[57]
t_f	time to the first step of the cascade (infector's view)	[57]
t_e	duration of exposure to an item before infection (infectee's view)	[57]
g_d	geodiversity	[165]
g_r	georange	[165]
α	exponent of Zipf distribution	
$g(v)$	betweenness centrality of a node v	
σ_{st}	total number of shortest paths from node s to node t	
$\sigma_{st}(v)$	number of paths that pass through v	
$\varepsilon(v)$	eccentricity of node v	
$G(n,p)$	new edges in graph of n nodes are constructed at random with an independent probability p	Gilbert model for random network
$G(n,M)$	collection of all graphs which have n nodes and M edges	Erdos−Renyi model for random network
$\delta > 1$	power-law coefficient	power-law networks
μ	minimum number of interaction events	interaction graph
t_w	time window t of interactions' occurrence	

Table 7: Models - Notation Overview 5

the average degree in the cascade remains constant as the cascade grows (a trees property). Moreover, Leskovec et al. claim that these trees are balanced, as they notice that the cascade diameter increases logarithmically with the size of the cascade. In contradiction to the above, the trees derived from the chain-letter diffusion model of Liben-Nowell and Kleinberg in [132] are incosistent with the epidemic model, as they are very narrow and deep, with the majority of their nodes having one child and a median distance from their root to the their leaves being of hundreds steps.

Precisely, in [132] the spread of a chain-letter is represented by a tree. Copies of the chain-letter represent paths through the tree, the root represents the originator and the leaves represent the recipients of a message (w is a child of v if w appends its name in the copy of the letter directly below v). In order to produce trees with the characteristics mentioned in the previous paragraph, the probabilistic model suggested (i) incorporates asynchrony: after receiving a message, each recipient waits for a time t before acting on it, and if it receives more copies of the item in this time interval, it acts upon only one of them, and (ii) encompasses a back-rate β, as a node can either forward the message to its neighbours with probability $1-\beta$ or group-reply to his corecipients with a probability β.

In [39], Bakshy et al. attempt to model the information diffusion in Twitter with the use of regression trees. Twitter is convenient for information diffusion modeling, since it is explicitly diffusion-oriented: users subscribe to the content of other users. The retweet feature, moreover, helps in the acknowledgement (though does not guarantee it) of reposts. Seeders are users posting original (not retweeted) content and reposting instead of the conventional retweeting (RT @username) is taken into account. Infuence is measured in terms of the size of the whole diffusion tree created, and not just the plain number of explicit retweets. The three different cases studied ascribe the influence to the first one having posted a link, the most recent one or follow a hybrid approach.

The predictors used include, for the seed users: the number of followers, number of friends, number of tweets and date of joining, and regarding the past influence of seed users: the average,minimum and maximum total influence and average, minimum and maximum local influence (local refers to the average number of reposts by a user's immediate friends over a period of one month and total to the average total cascade size over that period).

Bakshy et al. [39] come to the conclusion that although large cascades have in their majority previously successful individuals with many followers as initiators, individuals with these characteristics are not necessarily bound to start a large cascade. Thus, because of the fact that estimations cannot be made at an individual level, marketers should rely on the average performance. By studying the return on investment, on the whole, with a cost function of the number of followers per individual i: $c_i = ac_f + f_i c_f$, where a is acquisition cost c_f cost per follower and f_i is the number of followers, they conclude that relatively ordinary users of average influence and connectivity are most cost-efficient.

Content-related features are, also, according to Bakshy et al. not expected to discriminate initiators of large cascades from non-successful ones, due to the large number of non-successes. In order to take content into account the regression analysis is repeated encompassing the following features: rated interestingness, perceived interestingness to an average person, rated positive feeling, willingness to share via email, IM, Twitter, Facebook or Digg, some indicator variables for type of URL, and some indicator variables for category of content.

Moreover, Lerman et al. [127] claim that exploiting the proximity of users in the social graph can serve as an adding-value factor for the prediction of information diffusion. They discriminate proximity as coming from conservative or non-conservative processes (denoting that the amount of spread information in the network remains or not constant, respectively). For the case the underlying network is not fully known [148], Najar et al. focus on predicting the final activation state of the network when an initial activation is given. They find the correspondence between the initial and final states of the network without considering the intermediate states. Their work is based on the analogy between predictive and generative approaches for discrimination or regression problems (predictive models depicting a better performance, when the real data distribution can't be captured).

In [190], Yang and Leskovec use a time series model for modeling the global influence of a node through the whole network. For each node u, an influence function $I_u(l)$ is the number of mentions of an information l time units after the node u adopted the information (at t_u), and with $V(t)$ being the number of nodes that mention the information at time t, it applies:

$$V(t+1) = \sum_{u \in A(t)} I_u(t - t_u) \tag{5}$$

where A(t) are the nodes that got activated before t, $t_u \leq t$. For the modeling of the influence functions a non-parametric formulation followed allows greater accuracy and deviation, as no assumptions are made.

A study of the social news aggregator Digg [71] crawling data from the site, story, user and social network perspective, suggests the presence of previously unconsidered factors for the steering of information spread in OSNs. Doerr et al. suggest, that, beyond the bare OSN topology two factors matter: the temporal alignment between user activities (i.e. whether users are visiting the site in the same narrow time window) and a hidden logical layer of interaction patterns occurring in their majority outside the social graph.

In the direction of studying the information diffusion as social graphs evolve, Ren et al. [158] study the evolution steps for shortest paths between two nodes, (so that they can ascribe them to a disjoint path, a short-circuiting bridge or a new friend between them), and furthermore, metrics such as closeness centrality, and global metrics, like the graph diameter, across snapshots of gradually evolving graphs. To this end, they adopt an efficient algorithm and an efficient storage scheme.

Firstly, they cluster (in an incremental procedure not requiring all snapshots to be present in memory) successive graphs exploiting their many resemblances (daily snapshots). As $G_\cup$ and $G_\cap$ essentially "bound" the graphs in the cluster, with $G_\cap$ being the intersection (the largest common subgraph) of all snapshots in cluster C, and $G_\cup$ the union (the smallest common supergraph) of all snapshots in C, grouping of snapshots into clusters can be based in the idea of the graph edit similarity between these two graphs ($G_\cup$, $G_\cap$). The graph edit similarity to capture the similarity requirement of a cluster is defined as:

$$ges(G_a, G_b) = \frac{2 \mid E(G_a \cap G_b) \mid}{\mid E(G_a) \mid + \mid E(G_b) \mid} \tag{6}$$

Secondly, they exploit the idea that, denoting the shortest-path between the vertices v and u, by $\tilde{P}_*(u,v)$ in a graph G_*, where $* = 1,2,...,n,\cap,\cup$, the solution can easily be found by the intersection or union (two graphs) of graphs in the cluster, or be "fixed" using these two graphs, and they propose a "finding-verifying-fixing framework".

As far as the storage schemes variations are concerned, Δs consist a small fraction of the snapshot, and their size depends on the threshold value used for clusters' similarity. The penalty of decompression overheads needed is surpassed by savings in I/O. Variations of the storage schemes include the following:

$$SM1(C) = \{G_\cap, \Delta(G_\cup, G_\cap), \Delta(G_i, G_\cap) | 1 \leq i \leq k\} \tag{7}$$

$$SM2(C) = \{G_\cap, \Delta(G_\cup, G_\cap), \Delta(G_1, G_\cap), \mathscr{D}(G_i, G_{i-1}) | 2 \leq i \leq k\} \tag{8}$$

$$SM\_FVF(C) = \{\mathscr{D}(G_\cap, G_{p\cap}), \Delta(G_\cup, G_\cap), \Delta(G_1, G_\cap), \mathscr{D}(G_i, G_{i-1}) | 2 \leq i \leq k\} \tag{9}$$

2.2 Tools

Various tools for experimenting with scaling of bandwidth-intensive-media content using "OSN-awareness" include:

- Tools for general Social Network Analysis tasks, with the help of which valuable information for the properties of studied OSNs and the patterns of information diffusion through OSNs may derive,
- Tools for CDNs simulation, so that OSN-aware policies can be evaluated prior to the release of a system or their incorporation in an OSN in-house application, and
- Tools for efficient management of the large graphs that social graphs comprise.

Some of their representative characteristics are depicted in Tables 8, 9, and 10, respectively. The Thesis extensively leverages and modifies one of the described CDN simulators to simulate the CDN infrastructure framework.

2.2.1 Tools for SNA

Name	Purpose	Built on	GUI	Mode
SNAP	analytical	C++	available through graphical front-end NodeXL	single workstation
NetMiner	analytical	Python	yes	single workstation
igraph	analytical	R, Python	visualization capabilities	single workstation
NetworkX	analytical	Python	visualization capabilities	single workstation
NetEvViz	visualization of temporal differences	C#	yes	single workstation
XRime	analytical	Java	no	MapReduce

Table 8: SNA Tools

For the augmentation of measurements concerning the OSNs with further results, tools for SNA [183] are used. SNA is the study of nodes of a network as its members are being represented (membership can express the membership on a social networking website such as Twitter or Facebook), and the relationships among them, with the ultimate purpose to mine useful patterns among them.

Studied attributes can include, node-level attributes, for example, betweenness centrality (Eq. 2.2.1). Betweenness centrality denotes the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v.

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{10}$$

Network-level attributes can include the diameter or the radius of the graph, namely the maximum and minimum eccentricity respectively of any graph vertex in a graph. The eccentricity $\varepsilon(v)$ of a graph vertex v in a connected graph G is the maximum graph distance between v and any other vertex u of G. For a disconnected graph, all vertices are defined to have infinite eccentricity.

Existing SNA platforms are either task-oriented (IdiroSNA [6]) or general-purpose (NodeXL [14]), and include software built for visualisation exclusively (NetEvViz [11] captures also the dynamic evolution of graphs) or software that encompasses analytical features, too (which can be used for link-prediction tasks, for example).

Scripting tools used for network analysis include the free open-source packages: SNAP [18] package (C++), NetMiner [12](GUI, Python scripting engine), igraph [7], (visualisation capabilities, R and Python), and the NetworkX [13] library for Python (visualisation capabilities).

Most SNA tools function in single workstation mode, with the exception of libraries of scalable algorithms, such as XRime [23]. XRime consists of Map-Reduce programs, dedicated to do raw data pre-processing and transformation, calculate SNA metrics, e.g. vertex degree statistics, weakly and strongly connected components, bi-connected components, ego-centric network density, breadth first search / single source shortest paths (BFS/SSSP), K-core, maximal cliques, pagerank, hyperlink-induced topic search (HITS), minimal spanning trees, etc., as well as give some graph visualization.)

Name	**Purpose**	**Built on**	**GUI**
CDNSim	solely for CDNs	OMNET++, INET	yes
NS-2	general purpose	C++, simulation scenarios in Object Tcl	no
NS-3	general purpose	C++, python	no
PlanetLab	general purpose	Linux vserver as node provisioning mechanism, migrating to LXC - the implementation of container-based virtualization in the Linux kernel	yes

Table 9: Tools for CDN Simulation

2.2.2 Tools for CDNs

CDNSim [168] is a simulation tool for CDNs. It is designed to monitor CDN services prior to or after their public release in a controlled simulated environment, and overcomes typical problems of CDN simulators, that may require voluntary involvement of many users. The simulation concerns the CDN surrogate servers, and the main CDN functions. Outcomes of experimentation for the evaluation of specific CDN installations are client, server and network statistics.

The tool provides a graphic user interface for setting of the parameters of a certain simulation, and, also, utilities and interfaces for simulation of peer-to-peer services. Moreover, it is open-source, modular and open-architecture parallel discrete-event trace-driven tool, based on OMNeT++, and its extension INET for the support of network protocols, such as TCP/IP. Request routing, content distribution and management are simulated by the tool itself.

Some of its features include: support of various policies (cooperative push based content management policy, non-cooperative push based content management policy, cooperative pull based content management policy, non-cooperative pull based content management policy, LRU cache replacement policy, static cache policy), TCP / IP networking support, various utilities (for executing unattended simulations, for automatically generating results' reports, for extracting statistics related to net-utility, for converting Apache log files into CDNsim trace files), and extensibility (implementation of modules in libraries form).

Other tools include the Network Simulator NS-2 [15], a discrete event simulator targeted at networking research, that supports simulation of TCP, routing, and multicast protocols, and the real-time testbed PlanetLab [17].

2.2.3 Tools for Graphs

The amount of information in social networks can be an obstacle, since elaborate manipulation of the data may be needed. An open problem is the efficient handling of graphs with billions of nodes and edges. Facebook, for example, has one billion monthly active users as of October 2012, 584 million daily active users on average in September 2012, and 604 million monthly active users who used Facebook mobile products as of September 2012 [5]. In order to generate aggregations and analyses that have meaning, the Facebook custom built-in data warehouse and analytics infrastructure [21] has to apply ad-hoc queries and custom MapReduce [68] jobs in a continuous basis on over half a petabyte of new data every

24 hours, with the largest cluster containing more than 100PB of data and the process needs surpassing the 60.000 queries in Hive, the data warehouse system for Hadoop and Hadoop compatible file systems.

Below we describe systems for the efficient calculation of graph features (PEGASUS, GBASE), an approach for the efficient management of distributed graphs through replication of nodes (Mondal and Deshpande approach), and an architecture for better scheduling and managing of the underlying MapReduce jobs over an OSN (Facebook Corona).

Name	**Purpose**	**Built-on**	**Features**
PEGASUS	graph storage, graph mining	Java	indexing, inference, spectral analysis, node-centralized computation exclusively
GBASE	graph storage, graph mining	MapRedude framework	node-centralized or edge-centralized computation
Mondal Deshpande approach	dynamic replication of nodes based on read-write frequencies	Apache Couch-DB key-value store	exploits clustering
Facebook Corona	improvement of Apache Hadoop scalability	Java	separate central cluster manager and multiple job trackers, task scheduling in push model

Table 10: Graph Tools

PEGASUS [106] is a Java-written Peta-scale graph mining system, which runs in distributed manner on top of Hadoop [16] (the open source implementation of MapReduce framework), implementing algorithms for estimating graph features, such as PageRank, Random Walk with Restart (RWR), radius, Connected Components. It encompasses, also, an efficient graph storage schema and indexing methods, inference and spectral analysis of large graphs (with a distributed belief propagation algorithm which infers the states of unlabeled nodes given a set of labeled nodes, an eigensolver for computing top k eigenvalues and eigenvectors of the adjacency matrices and a tensor decomposition algorithm.)

For the same purposes GBASE [105] is used to approach problems of: storage (in distributed settings: splitting the edges into smaller units, grouping the units into files), efficient algorithms for graph applications, and exploitation of the above for query optimization.

In the direction of reducing network bandwidth consumption and specifically tailored for the large-scale dynamically changing graphs that OSNs comprise, Mondal and Deshpande propose a distributed graph data management system in [144]. The system is based on dynamic replication of nodes defined by their read-write frequencies, and a clustering-based

approach to amortize the costs of making these replication decisions, which are dictated by a fairness criterion, and is implemented as middleware on top of the open-source CouchDB keyvalue store.

Facebook's open-source Corona [21] surpasses some inherent limitations of an Apache Hadoop MapReduce implementation, which can be summarized as follows: With the scheduling framework consisting of a single job tracker, that manages both the cluster resources and user jobs' scheduling, and many task trackers (one per worker machine) that have to execute the tasks the job tracker assigns them, cluster utilization suffers due to jobs' plethora. Also, task trackers periodically inform the job tracker about their status in a pull-based fashion, thus resulting in latency, especially observable for the smaller jobs. Last, the static configuration responsible for the division of the cluster in a predefined number of map and reduce slots results in a scheduling not based on actual resource requirements. To address these issues the proposed framework introduces a scalable architecture of a separate central cluster manager that allocates slots to job trackers and keeps track of cluster utilization (machines in the cluster, resources to jobs), and separate multiple job trackers that track one job each, as well as a task scheduling working in push model (requests itinerary: job tracker, cluster manager, resource grants back to job tracker).

2.3 Taxonomy of Content Delivery over OSNs

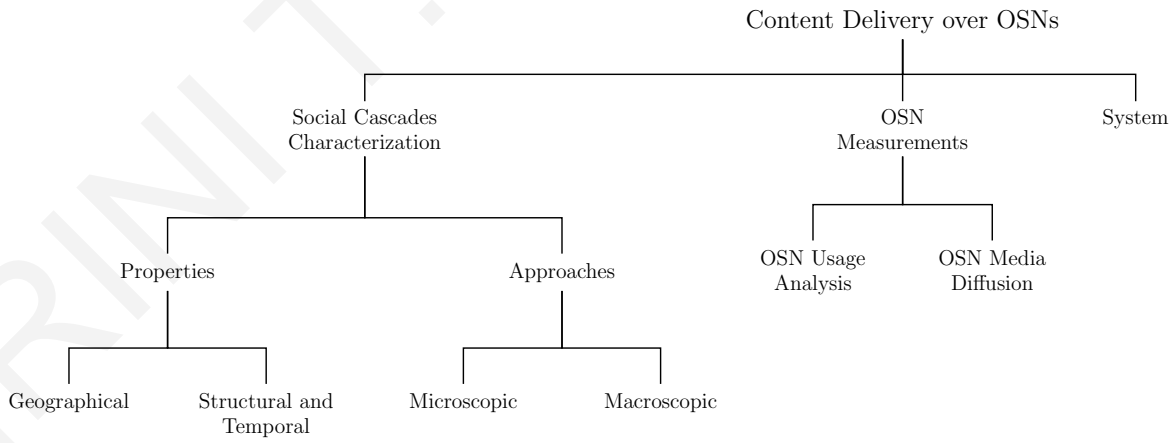


Figure 7: A taxonomy of content delivery over OSNs.

The presented taxonomy of Content Delivery over OSNs is outlined in Figure 7. The taxonomy presents the various properties and approaches in the literature for the characterization of cascades. The branching of topics in our presented taxonomy continues with OSN measurement works that focus on phenomena and measurement studies providing valuable

insights into usage analysis and media diffusion. The last dimension of our taxonomy consists of content delivery systems built based on OSNs' data.

2.4 Metrics for characterization of social cascades

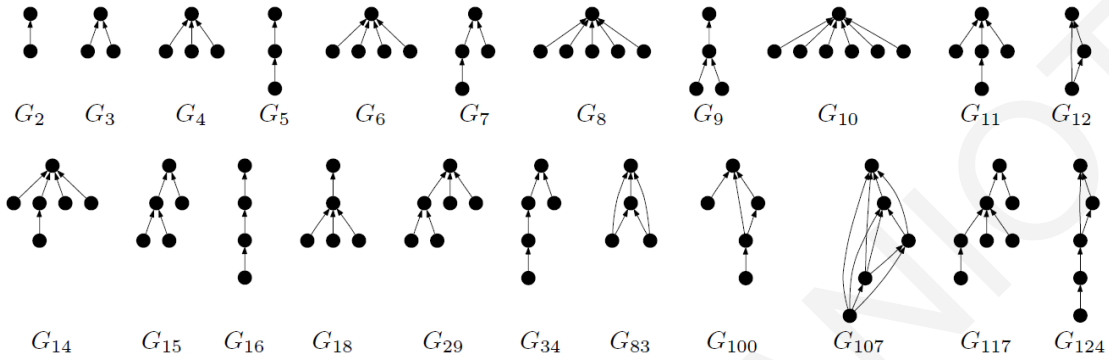


Figure 8: Common cascade shapes. Ordering of common shapes of cascades in the blogosphere [130] by frequency, with r the frequency ranking of G_r.

A cascade is characterized as *local* if it spreads in a fraction of the network lower than a threshold. Otherwise the cascade is characterized as *global*. Local cascades affect only a relatively small number of individuals and typically terminate within one or two steps of the initiator (Figure 8). 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. In global cascades the opposite happens: they affect many individuals, propagate for many steps, and are ultimately constrained only by the size of the population through which they pass.

A cascade is local if it spreads in a fraction φ of the network lower than a threshold ω or else we say that the cascade is global. We define the classification function f_1 for a cascade $C_z, z \in N$:

$$f_1(G, C_z) = \begin{cases} 0 & \text{if } \varphi < \omega \text{ local cascade,} \\ 1 & \text{if } \varphi > \omega \text{ global cascade.} \end{cases} \tag{11}$$

Formally, a cascade should be characterized as global or local with the maximum accuracy $\alpha = \frac{\sigma}{\nu} \cdot 100$, σ is the number of the correctly classified cases and ν is the number of all sampled cases, such that the cost of replicating a simple object c is minimized: $\min \sum_{C_z} c = f_2(t_p, \varphi, N_{cl}, C_z)$ for all cascades $C_z \in C$, where φ is the fraction of the network that the object is bound to spread, N_{cl}, $cl \in N$ is the number of clients requesting a specific

object and t_p the amount of traversing path p between client and the server finally serving the request.

Even though a percentage of the occurred information flow in cascades is ascribed to homophily, namely the tendency of individuals to associate with similar others, as "similarity breeds connection" [137], and research has been conducted for the discrimination between the two cases (homophily or influence) (see [33], [125], etc.), there exist specific metrics for the quantification and characterisation of the social cascades, to which we refer in this section. Below different approaches are presented 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 [185] 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 [76] or Ver Steeg et al. [170] introduce. Dave et al. [67] 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* [39], denoting the height of the cascade tree, the *time delay* between two consecutive steps of the cascade [165], the *time duration*, and the *rate of the cascade* [57]. The latter, for the epidemiological model of [57] is the basic reproductive number $R_0 = \rho_0 \overline{k^2}/\overline{k}^2$, where $\rho_0 = \beta\gamma\overline{k}$, β is the transmission rate, γ is the infection duration, and k the node degree. With σ_0 the probability that a person will adopt the shared piece of information (under the assumption that duration infection is equal to the timelife of the user, much larger than duration of the cascade, and, thus, the information will be definitely shared among connections) it applies $\rho_0 = \sigma_0\overline{k}$, and σ_0 can empirically be estimated by identifying an infected node and counting the fraction of its connected nodes subsequently becoming infected. Other properties related to the *susceptibility* of the network to new items are: the time to the first step of the cascade from infector's point of view and the duration of exposure to an item before infection from infectee's view [57]. *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 [165].

In [185], Watts defines a global cascade as a "sufficiently large cascade", covering practically more than a fixed fraction of a large network. Watts introduces a simple model for cascades on random graphs.

- The network comprises of n nodes with threshold distribution $f(\varphi)$, and the degree distribution of the graph is p_k, namely each node is connected to k neighbours with probability p_k; z is the average node degree ($\bar{k} = z$).
- The initial state of each node is state 0 (inactive) and each node is characterized by a threshold φ. If at least a threshold fraction φ of the node's k neighbours acquire state 1 (active), the node will switch from inactive to active.
- Nodes with $k \leq \lfloor 1/\varphi \rfloor$ are said to be "vulnerable" and will switch state if just one of their neighbours becomes active. Otherwise, nodes are called "stable".

Watts uses percolation theory [169], the theory studying how connected clusters behave in a random graph, to investigate the conditions under which a small initial set of seed nodes can cause a finite fraction of infinite nodes to switch from inactive to active. Percolation in this case is interpreted as follows: a global cascade is said to occur, when the vulnerable vertices percolate. Namely the largest connected vulnerable cluster of the graph must occupy a finite fraction of the infinite network. For infinite Poisson random graphs Watts defines a region, inside which a finite fraction of an infinite network would switch from inactive to active state if at least one arbitrarily selected node switched from inactive to active state. Simulations on finite graphs of 10000 nodes give similar results.

Watts makes the observation that the frequency of global cascades is related to the size of the vulnerable component, with the larger the component, the higher the chance for the cascade to be global. He also states that the average size of a global cascade is governed by the connectivity of the network as a whole. In sparsely connected networks, cascades are limited by the global connectivity of the network, and in dense networks cascades are limited by the stability of individual nodes.

In [67], Dave et al. quantify the *reach* of a user as the number of cascades it can reach with a specific action α as:

$$reach^{\alpha}(u) = \begin{cases} \sum_{u_i \in \vec{P}^a(u)} 1 + \frac{1}{2} \sum_{u_j \in \vec{P}^a(u_i)} (reach^{\alpha}(u_j)) \\ 0, \text{ otherwise} \end{cases} \tag{12}$$

where a user gets the complete credit for action propagation to his immediate neighbours, or a decaying factor for non-immediate neighbours. $\vec{P}^{\alpha}(u)$ is the propagation set of u, consisting of all his immediate neighbours u_i, such that there was an action propagation from u to u_i.

Dave et al. [67] identify how empirical factors like user's and their neighborhood's influencing ability, a specific action's influencing capability and other user and network characteristics affect the $reach^{\alpha(u)}$ quantity, coming to the conclusion that action dominates in the prediction of the spread of the action.

In [76], Kleinberg and Easley claim that clusters are obstacles to cascades, and, moreover, that they are the *only* obstacles to cascades: "Considering a set of initial adopters of behavior A, with a threshold of q for nodes in the remaining network to adopt behavior A: (i) If the remaining network contains a cluster of density greater than $1-q$, then the set of initial adopters will not cause a complete cascade. (A cluster of density p is a set of nodes, so that each node in the set has at least a p fraction of its network neighbors in the set.) (ii) Moreover, whenever a set of initial adopters does not cause a complete cascade with threshold q, the remaining network must contain a cluster of density greater than $1-q$."

In [170], Ver Steeg et al. find another two factors related with the multiple exposure of users to stories due to the highly clustered nature of Digg, that drastically limit the cascade size in Digg. The reproductive number R_0 of the epidemical model used, which by intuition expresses the average number of people infected by a single infected person, is the product of the average number of fans times the transmissibility. As far as the first factor is concerned, it is implied that only the number of new fans (those that have not already been exposed to a story) should be taken into account. In addition for the second factor, transmissibility for actual cascades is observed to remain constant until about a number of people have voted, and then begin to decline (maybe due to decay of novelty [188] or decrease in visibility [95] as a consequence of new stories being submitted to Digg). From this point of view, cascades are limited.

The field of predicting the appearance of social cascades is very active ([59], [124], [171], [155], [73], [39], etc.) Many studies focus on the prediction of the amount of aggregate activities (e.g. aggregate daily hashtag use [135]), whereas others focus either on the prediction of user-level behaviour, like retransmission of a specific tweet/ URL ([155], [84]) or on the prediction of growth of the cascade size [59].

Although our Thesis focuses solely on video sharing, we identify the following methods for virality prediction in general. Feature-based methods and time series analysis methods.

They are both based on the empirical observation of social cascades. Our approach in the thesis falls into the first category.

Feature-based methods are based on content, temporal and other features, and the learning algorithms schemes they use are based on simple regression analysis ([59, 172]), regression trees [39], content-based methods [178], binary classification ([97,104], [122]) 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.

Time-series analysis works ([191, 192]), on the other hand, argue that patterns of a resource's growth of popularity are indicative for its future retransmissions.

Finally, we should mention that one branch of virality research is based on study of the evolution of cascades during a specific time-window ([135], [122], [178]), whereas there exist works that examine the cascades continuously over their entire duration [59].

Regarding the aspect of cascade analysis, more recent works explore the dynamic nature of cascades over social networks, including works such as [147].

2.5 Bandwidth-intensive media content and Social Networks

In this section we present the main findings about the structure of OSNs (static properties and temporal evolution), and moreover how these topological characteristics can affect the information diffusion in an OSN. Findings on structure are corroborated by measurement studies, that depict also findings in terms of OSNs users' workloads.

Moreover, we discuss the association of bandwidth-intensive media content with social networks, since we would like to know for the delivery of content prior to requests, how the number of requests is bound to evolve. Next, follows a description of works studying patterns of user requests over OSNs for various video-on-demand systems and OSNs.Finally, we present (*Applications and techniques*) existent applications and techniques that could provide valuable insights concerning the exploitation of information extracted from OSNs for scaling of content diffused via OSNs.

2.5.1 Structure Characteristics of OSNs

The main characteristics of OSNs found by different studies are the following: Social networks are *power-law* [142], and, moreover *scale-free* [42], and exhibit the *small-networks property* [119]. The graphs representing the interaction among the nodes of a social network,

known as "interaction graphs" also exhibit the above properties [187], but lower levels of the "small-world" properties than their social graph counterparts. Wilson et al. [187] analyze interaction graphs derived from Facebook user traces and depict that these graphs have fewer nodes with extremely high degree, and overall significant increase of the network diameter. An interaction graph [187] is parameterized by a minimum number of interaction events n and a time window t of interactions' occurrence and is defined as the subset of the social graph where for each link interactivity between the link's endpoints is greater than a rate defined by n and t. Wilson et al. [187] formulate interaction graphs with the implicit assumption that the majority of user interactions take place across social links, which is true for the Facebook dataset they study, thus defining them as subsets of the social graph, but in the general case interaction graphs can instead be completely new graphs based only on the interaction data.

Kumar et al. [121] 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 [129]) 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. [74] 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. [129] 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 [129] is proposed instead to simulate these properties.

2.5.2 Measurement Studies on Large-scale OSNs

A first large-scale analysis of multiple OSNs data encompassing Flickr, YouTube, LiveJournal, and Orkut, social networks for sharing photos, videos, blogs and profiles, respectively, by Mislove et al. [142] highlighted the difficulties of crawling a social network, and came to the conclusions that: Although node degrees in the studied OSNs varied by orders of magnitude, key findings are the same. The studied OSNs are power-law, small-world, scale-free, the in-degree matches out-degree distribution (due to link symmetry, an observation at odds with the web graph, that increases OSNs' network connectivity and reduces their diameter), there is a densely connected core of high degree nodes surrounded by small clusters of low-degree nodes, the average distances are lower, and clustering coefficients higher than those of the web graph (studied OSNs clustered 10.000 more times than random graphs, 5–50 times more than random power-law graphs). The clustering coefficient of a graph is defined as the average of the clustering coefficients of the nodes of the graph. For a node i with degree k_i the clustering coefficient is given as:

$$C_i = \frac{2|(v,w)|,(i,v),(i,w),(v,w) \in E}{k_i(k_i-1)} \tag{13}$$

The core consists of 1–10% of nodes, is necessary for the connectivity of 90% of users, with most short paths passing through it, could be used for quick information diffusion, whereas the highly clustered remainder could be used for sharing of information of local interest.

Mislove et al. performed crawls with a picked known user and all of his friends until all known users were crawled, performing a BFS of the graph, facing the challenge of quick finishing of the crawl, due to the changes in the underlying social network, and for the crawling to be complete they needed to estimate the crawl coverage percentage (for Flickr: 27%, but remaining users had very few links, LiveJournal: 95%, Orkut: sub-crawl, but representative, YouTube: unable to find percentage of fraction covered), since there exist users not connected or small clusters.

The concerns about the validity of crawling method used (underestimation of the number of low-degree nodes and correct capture of the scaling behaviour of degree distribution only with a sampling ratio above a certain threshold) are confirmed by Ahn et al. [30]. The latter study, of the complete CyWorld dataset (by that time the largest SN in South Korea) and small parts of MySpace and Orkut, also confirms power-law and small-world properties.

Wilson et al. [187] conducted the first large-scale analysis of Facebook, by crawling and use of 'networks' (15% of total 10M users, and 24M. interactions). In [121], Kumar et al. study Flickr and Yahoo!360, finding they follow power-law degree distributions. In [128], Leskovec et al. analyze Microsoft Messenger instant-messaging (IM) service users (240M users, 30 billion conversations) in terms of the average path length of Messenger users (6.6) and graph topology, finding that it is well-connected and robust against node removal.

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. [103]. In [124], Kwak et al. study Twitter using its API with 20 whitelisted accounts (41.7M user profiles, near-complete at the time,1.47B following relationships, 4262 trending topics, 106M tweets mentioning trending topics), finding that following is reciprocal only for 22.1% of nodes, compared to 68% in Flickr, 84% in Yahoo and 77% in Cyworld guestbook messages, and that it resembles more a news media site, in the notion that users discuss timely topics, and only few users have a local influence.

In terms of *user workloads* in OSNs, Benevenuto et al. [45] 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.

2.5.3 Impact of bandwidth-intensive media content diffusion over OSNs

Zhou et al., in [195], explore the popularity of photos in Facebook, noting that the request pattern follows a Zipf distribution, with an exponent $\alpha = 0.44$, significantly lower than that of traditional distributions (ranging from 0.64 to 0.83 [50]). They interprete this as shift of interest from popular items to items in a long tail. Long-tailed nature of multimedia content is also taken into account in our Thesis.

In the same context, Yu et al. [193] analyse PowerInfo, a VOD system deployed by China Telecom and note that the top 10% of the videos account for approximately 60% of accesses, and the rest of the videos (the 90% in the tail) for 40%. Unaccessible via the

official distribution channels (television networks or record companies) independent video content generated by the users (UGC) becomes available to a wide number of viewers via services as YouTube [25] or the US-based Vimeo. Cha et al. [56] investigate the long tail opportunities in the UGC services, such as YouTube video content, taking into account the fluctuation of the viewing patterns due to the volatile nature of the videos (videos may appear and disappear) and the various sources that direct to the content (recommendation services, RSS feeds, web reviews, blogosphere etc.)

In [191], Yang and Leskovec examine the temporal variations of Twitter hashtags and quotations in blogs, creating time series' of the number of mentions of an item i at time t, thus measuring the popularity given to the item i over time. By grouping together items so that item i is in the same group have a similar shape of the time series x_i with a clustering algorithm, they infer items with similar temporal pattern of popularity, and find that temporal variation of popularity of content in online social media can be accurately described by a small set of time series shapes, with most press agency news depicting a very rapid rise and a slow fading.

In the direction of addressing the open problem of efficient server provisioning especially for bandwidth-intensive-media content (such as streaming video, that has relatively strict delivery constraints concerning latency requirements for example [175]), a problem intensified by the advent of bandwidth-intensive-media UGC and the proliferation of smartphones, leveraging information from social networks, Scellato et al. [165] study the relation between YouTube video sharing service where, for example, a wide number of viewers has access to a diversity of under other circumstances unaccessible via the official distribution channels independent video content, and the microblogging service Twitter, specifically tailored for the distribution of information content. Scellato et al. find that social cascades tend not to expand geographically, a notion which we also apply as a heuristic in our main algorithm in the Thesis.

Another corpus of the literature studies users' behaviors in different media services. Several studies ([56], [81] , [88], [153], [143]) have been conducted to investigate the traffic characteristics of YouTube users. These works focus on the characteristics of YouTube content, such as file size, bitrate, usage patterns and popularity. After an extensive analysis of the YouTube workload in [88], the authors found that there are many similarities between traditional Web and media streaming workloads. From another perspective, the authors in [61]

found a strong correlation among YouTube videos because the links to related videos generated by uploaders depict small-world characteristics. In [80], Figueirido et al. analyze how the popularity of individual YouTube videos evolves since the upload time of the video.

2.5.4 Systems, applications and techniques

In [102], Jacobson et al. introduce Content Centric Networking (CCN), noting that network use has evolved to be dominated by content distribution and retrieval. CCN has no notion of host at its lowest level - a packet "address" names content, not location, while simultaneously preserving the design decisions that make TCP/IP simple, robust and scalable. Content is treated as a primitive, and with new approaches, Jacobson et al. simultaneously achieve scalability and performance.

Furthermore, at the application level, there have been efforts to incorporate the information extracted from OSNs in the way that users share content and in how the content ultimately reaches the users. Some of these works use the information directly from OSNs, whereas others use such information indirectly. The research goals vary: the decision for copying content, improvement of policy for temporary caching, etc. For an extensive taxonomy of content delivery over OSNs in various directions, readers can refer to [112]. The presented taxonomy of the relative research includes systems built with the leverage of OSNs' data. It takes into account phenomena related to bandwidth-intensive media content and its outspread via OSNs, as well as measurement studies on OSNs that could provide valuable insights into CDN infrastructure decisions for replicating the content. Some indicative systems, applications and techniques built with the leverage of OSNs' data are presented in the following.

Herein we present existent solutions (systems, applications and techniques) that could provide valuable insights concerning the exploitation of information extracted from OSNs for scaling of content diffused via OSNs. As far as the proposed applications and techniques are concerned, cost for scaling of content tailored for a small number of costumers can be expressed in terms of: required bandwidth for quick access to the content, required storage capacity for caching of content etc., and can be quantified, for CDNs for example, by the number of replicas needed. Presented solutions span from components within browsers to Cloud architecture variations, whereas in the present Thesis we remain focused on the Content Distribution infrastructure.

To share resources within the context of a social network with the use of the cloud business model, Chard et al. in [58] propose the SocialCloud architecture. Users register in cloud services (computational capacity, photo storage etc.), and their friends can consume and provide these services through a Facebook application. The allocation of resources (trading or reciprocal use between friends) is conducted by an underlying market infrastructure, whereas the Social Cloud application passes a SLA to the service. The advertisement of the service, so that it can be included in the market is done with XML based metadata stored in Globus Monitoring and Discovery System (MDS).

In Buzztraq [164], 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 Thesis 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 [176] 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 [165], 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.

Zhou et al. [195] 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.

In [98], Hoque and Gupta propose a technique for putting with a logical addressing scheme together in the disk blocks containing data from friends. The large scale of OSNs and the predominant tail effect do not allow use of techniques such as those used in multimedia file systems or web servers, where items are globally popular, and, techniques keeping

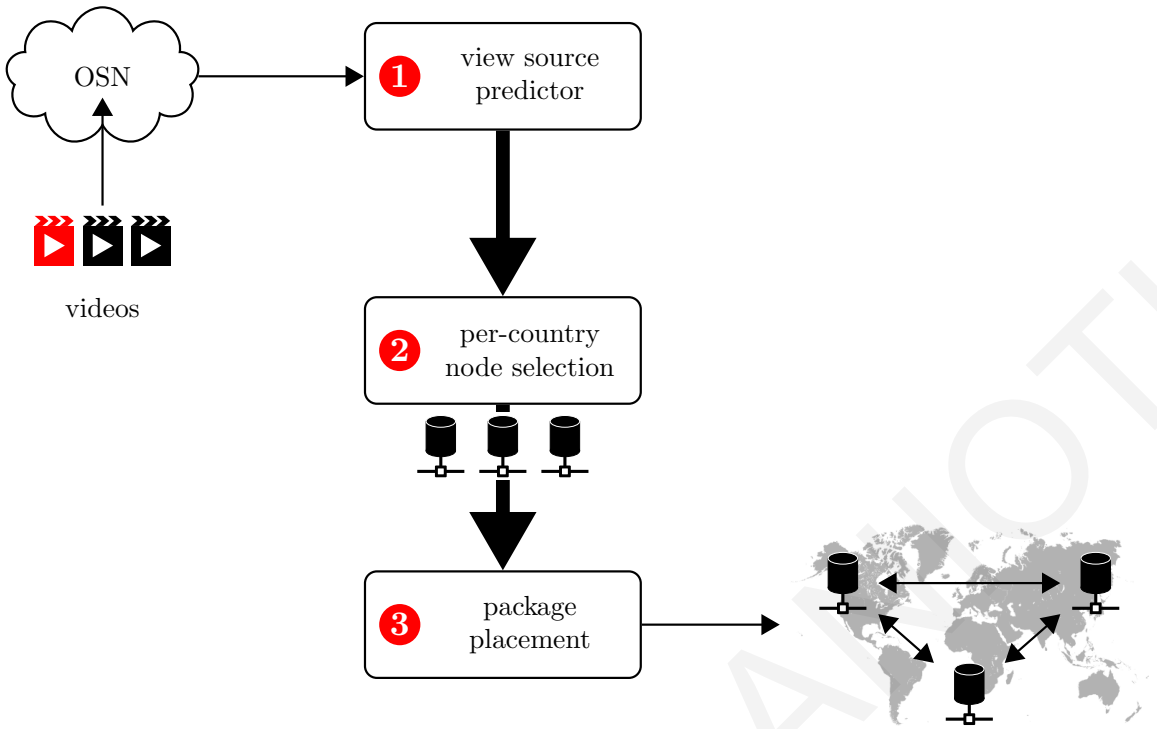


Figure 9: Overview of VOD service placement strategy leveraging OSNs.

related blocks together tracking the access pattern of blocks, respectively. To this purpose, in [98] the social graph is divided into communities, and the organization of blocks in the disk is conducted with a greedy heuristic that finds a layout for the users within the communities and organizes the different communities on the disk by considering inter-community tie strength. The system is implemented on top of the Neo4j graph database as a layout manager.

Instead of optimizing the performance of UGC services exploiting spatial and temporal locality in access patterns, Huguenin et al., in [100], 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 could be extended with the leverage of information from OSNs, in the way that Figure 9 depicts.

An interesting approach [126] 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

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.

Chapter 3

The Social Prefetcher

A significantly large proportion of HTTP traffic results from bandwidth-intensive multimedia content circulating through OSNs. With multimedia content providers, such as YouTube, often relying on CDN infrastructures, the pursuit lies in exploiting the user activity extracted from OSNs to improve the content prefetching mechanism. Aiming to reduce bandwidth usage, we incorporated a dynamic mechanism to CDNsim, a stand-alone CDN traffic simulator. This mechanism is based on a dynamic policy that takes patterns of information transmission over OSNs into account. Herein, we demonstrate that the performance of CDNs can be improved, and the cost of copying to surrogate servers is taken into consideration. The chapter is organized as follows. Section 3.1 gives an example of the necessity of the Social Prefetcher approach, and epitomizes the contributions of the present work [110]. Section 3.2 formally describes the addressed problem. The proposed algorithm is described in Section 3.3. Section 3.4 provides an outline of the methodology, followed by the preparation of the employed datasets. Our main findings are presented in Section 3.5. Section 3.6 concludes the chapter and discusses directions for future work.

3.1 Introduction

The three major issues that the area of CDNs faces concerning the maximization of their overall efficiency are as follows:

- (i) the most efficient placement of surrogate servers with maximum performance and minimum infrastructure cost;
- (ii) the best content diffusion placement, either in a global or in a local scale, i.e., which content will be copied in the surrogate servers and to what extent, given the placement

of the surrogate servers, because this requires memory, time and computational cost; and

- (iii) the temporal diffusion, related to the most efficient timing of the content placement.

In this work [110], we focus on (ii), although we do not completely overlook the matter of the efficient placement of servers and the issues of temporal diffusion. We show how leveraging information from OSNs can lead to a better content diffusion placement. More specifically, we incorporate a dynamic mechanism of preactive copying of content to an existing validated CDN simulation tool and propose an efficient copying policy. The latter can be based on demand prediction in social networks.

Indeed, users can ultimately benefit from the scaling of bandwidth-demanding content based on usage patterns over OSNs. Measurement studies, such as [56], attribute the recent increases in HTTP traffic to the extended use of OSNs [40], [58], [76] and the increasing popularity of streaming media. The amount of Internet traffic generated every day by online multimedia streaming providers such as YouTube cannot be neglected. Although we cannot exactly estimate the proportion of traffic generated by OSNs, it is observed that there are more than 400 tweets per minute with a YouTube video link [52]. Because media streaming providers often rely on CDNs to distribute their content from storage servers to multiple locations scattered over the planet, leverage of information diffusion analysis over OSNs can improve the CDN user experience.

The cost of scaling bandwidth-intensive content in CDNs can be expressed in different ways. For example, it might be the number of replicas needed for a specific source, or it may take into account the optimal use of memory and processing time of a OSN-aware built system. In this work, we aim to improve user experience via prefetching the objects most likely to be asked to the necessary extent.

3.2 Problem Description

Our goal is to improve the performance of the content delivery network infrastructure in terms of reducing the response time and possibly improving the hit ratio of our request, whereas the cost of copying from the origin server to surrogate servers remains restricted. We take into account restrictions in the cache capacity of the server, the server location, and the network topology (Table 11). We seek effective content pre-fetching policies in the surrogate servers, taking into account data from OSNs and actions of users over them, such

that we can recognize objects that will eventually be popular and preactively copy them in surrogate servers.

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 recognizes the set of objects O that will be popular only in a subset of the regions, where the content is likely to be copied.

$G(V,E)$	Graph representing the social network
$V = \{V_1, ..., V_n\}$	Nodes representing the social network users
$E = \{E_{11}, ..., E_{1n}, ..., E_{nn}\}$	Edges representing the social network connections, where E_{ij} stands for friendship between i and j
$R = \{r_1, r_2, ..., r_\tau\}$	Regions set
$N = \{n_1, n_2, ..., n_u\}$	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
$O = \{o_1, o_2, ..., o_w\}$	Objects set (videos)
$S_i, i \in O$	Size of object i in bytes
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
$P = \{p_{12}, p_{13}, ..., p_{nw}\}$	User posts in the social network, where p_{ij} denotes that node i has shared object j in the social network
$Q = \{q_1, q_2, ..., q_\zeta\}$	Object requests, where q_i denotes a request for an object of set O
Q_{hit_i}, $Q_{total_i}, i \in N$	Number of requests served from surrogate servers of the region of the user/ total number of requests
$X, Y, L, H \in R$	Closest timezones with mutual followers/ with highest centrality metric/ with highest lobby values/ with highest HITS values

Table 11: Chapter 3 - Notation Overview

We create a function $Predict(G,P,R,O)$ such that given the social graph, the posts, the distribution of users in regions, and the set of objects, it can predict which g objects $\{o_1,o_2,...,o_g\}$ will be globally popular and which λ objects $\{o_1,o_2,...,o_\lambda\}$ will be locally popular and to which regions. Our interest is focused on the λ objects that will be locally popular.

Our optimization strategy is based on the use of the set of popular objects and the set of surrogate servers. It decides about the placement of objects in the surrogate servers, taking into account the capacity of each server and the size of each object, and it aims at minimizing the response time. This is likely to occur when surrogate servers of a region can serve a very large fraction of the requests Q of nodes in the region.

A good policy for prefetching objects from the origin to the surrogate servers 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). It returns the set of objects $o \in O$ that have to be placed in surrogate server $n_i \in N$. For $x, 1 < x < w$, objects that will be copied in the surrogate server and the capacity Cn_i of n_i, (1) has to be fulfilled:

$$S_1 + S_2 + S_3 + ... + S_x \leq Cn_i(1) \tag{14}$$

We thus define the function $Put(n_i,Predict(G,P,R,O))$, $i \in N$, which returns the set of objects that will be placed in each surrogate server of every region r_i such that

$$\frac{Q_{hit_i}}{Q_{total_i}}, i \in N \tag{15}$$

is maximum, whereas constraint

$$\sum_{\forall i \in O} S_i f_{ik} \leq C_k \tag{16}$$

is fulfilled, where:

$$f_{ik} = \begin{cases} 1 & \text{if object } i \text{ exists in the cache of surrogate server } k \\ 0 & \text{if object does not exist} \end{cases} \tag{17}$$

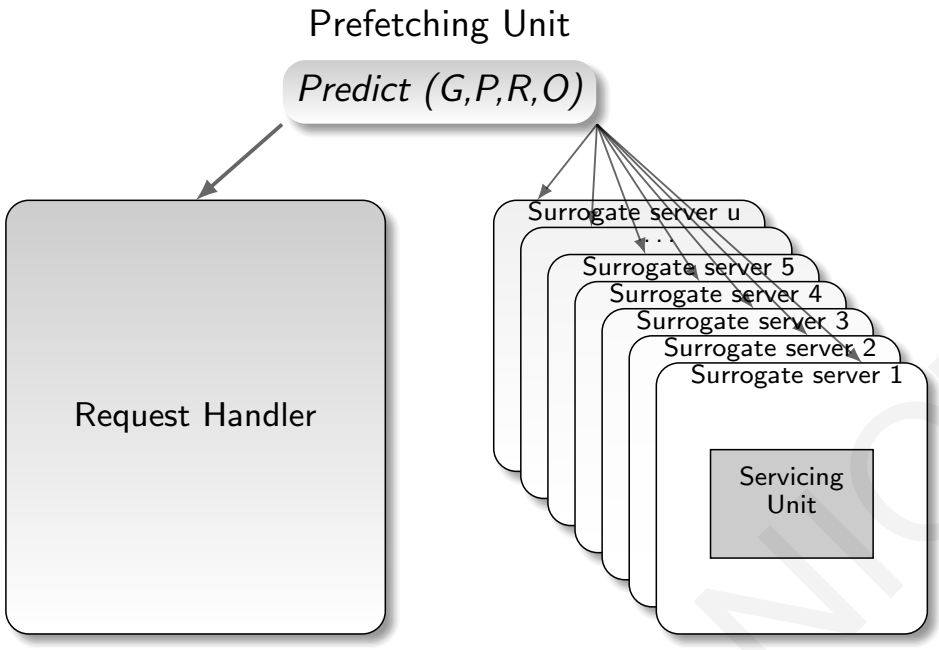


Figure 10: Chapter3 - The Prefetching Unit

3.3 Proposed Dynamic Policy

The description of the proposed algorithm follows. It consists of two parts: 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. 10).

3.3.1 For every new request in the CDN:

The main idea is to check whether specific time has elapsed after the start of the cascade and then define to what extent the object will be copied (Fig. 7). 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 that defines 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 has not completed (its timer has not surpassed a threshold), we check the importance of the user (applying the HITS algorithm and checking its authority score).

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

Figure 11: Chapter 3 - Algorithm for every new request ($timestamp$, V_i, o) in the CDN

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 **then**
18: copy object o to surrogates n_j that Subpolicy I or Subpolicy II decides;
19: **end if**

Subpolicies for local copy differ in the selection of the influence measurement of users. Centrality measurements for the various subpolicies include the Lobby-index (Subpolicy I, Fig. 8) and Hubs Authorities (HITS) algorithms (Subpolicy II, Fig. 9), which are described in Section 3.4. Note that Twitter uses a HITS style algorithm to suggest to users which accounts to follow [94]. Subpolicies check the X closest timezones where a user has mutual friends and out of them, the H or L with the highest value of a centrality metric as an average (HITS or Lobby-index, respectively), which means that the object is likely to be asked for more times. Copying is performed to the surrogate servers that serve the above timezones.

3.3.2 For every new object in the surrogate server:

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 (Fig. 5). We find the oldest objects and delete them. In the case that there are no such objects, we delete

Figure 12: Chapter 3 - Subpolicy I

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

Figure 13: Chapter 3 - Subpolicy II

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

those with the largest timestamp in the cascade. In all other cases, the LRU policy is applied for the removal of objects.

3.3.3 Heuristics

The heuristics applied in our approach are based on the following observations:

The power of social cascade is analyzed in [177], where it is estimated that users viewing a link in the social network are 7.35-fold more likely to share it than are the users not viewing it in the social network. One in 12.5 links is shared again, which is not unexpected if we consider the number of acquaintances of each user.

In [165] the authors reached the conclusion that users are more influenced by geographically close friends. Another insight [64] is the influence of mutual followers. The video retweet likelihood appears to be 6 times larger for users that are mutual followers. Although a very small fraction of users have an extremely large number of followers, the majority of users have only a few followers. The most popular users act as authorities and are generally either public figures or media sources.

Figure 14: Chapter 3 - 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 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
```

Social cascades have a short duration ([165], [60]). The percentage of cascades proceeding for days may not be directly attributed to the influence that a social contact exerts, as the video can appear in the newsfeed of the user for a longer time or a new cascade of this specific object may have started. However, in our prefetching algorithm we experiment with varying time thresholds of 24 hours, 48 hours, and threshold covering the entire percentage of requests (threshold for the cascade effect and the time that an object remains in cache), as well as indicatively with thresholds lower than 24 hours.

The idea is to check whether specific time has elapsed after the start of cascade and, only in the case that the cascade has not ended, to 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*, which roughly expresses the average cascade duration, as the parameter for the duration that an object remains cached.

3.3.4 Short optimization analysis

We search for values of function f_{ik}, which shows whether an object i will be redirected to surrogate server k. A brutal search would have a computational complexity of 2^{w+u}, where w is the number of objects and u is the number of surrogate servers. Hence, we perform the search through simulated annealing. We start from an initial configuration and perform annealing by lowering the temperature. Starting from 0.2 (hot temperature) and cooling slowly to 1.4 (cold temperature), we search for the optimal solution using the Metropolis-Hastings algorithm.

By generating a random number, the new configuration for f_{ik} is generated. We search through all possible allocations of objects in servers assigned to various regions, and we compare the energy, which is expressed as the summation of response time (from the user's point-of-view) and server capacity (from the system engineer's point-of-view) distributions. If the new configuration is energetically favorable, we keep it; otherwise, we continue generating random numbers.

Fig. 15 depicts the various energy values for 2500 iterations of the Metropolis algorithm with different values for f_{ik} (we plot only the different configurations in this figure, sampling over approximately 100 values of iterations). Our solution is shown as the straight line. Because the simulation does not provide a lower energy than our approach, the solution is considered to be near-optimal.

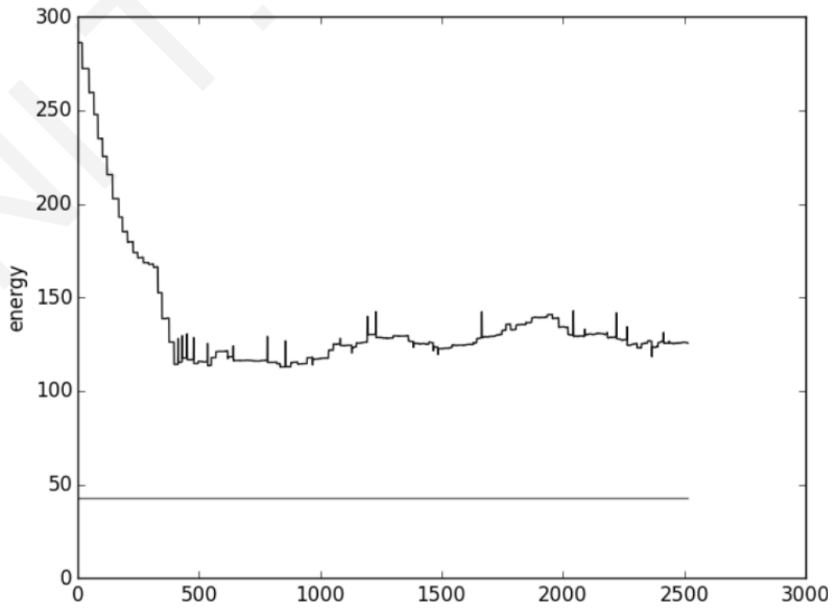


Figure 15: Optimization Analysis

For the experimental evaluation, we used the CDNsim simulator for CDNs [168]. The configuration of the simulation values is shown in Table 12. We conducted a multitude of experiments(330) in which the threshold values and the influence measurement metrics used were varied. For the extraction of 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 it provide an output of statistical results, which are discussed in the next Section.

Number of nodes in the topology	3500
Redirection Policy	Cooperative Environment (closest surrogate)
Number of origin servers	1
Number of surrogate servers	423
User groups	162
Bandwidth	100 Mbit/sec

Table 12: Chapter 3 - Simulation Characteristics

3.4 Experimental evaluation

3.4.1 Network Topology

We followed the process described below to define which nodes would represent the surrogate servers, the origin servers, and the users making the object requests.

First, we had to define the number of servers simulating our policies and their placement in a real geographical position. For this purpose, we used the geographical distribution of the Limelight [54] network. The large initial topology in the tool according to the Limelight distribution included 4879 servers. For the simulator to respond satisfyingly in terms of memory usage and simulation time needed (the way the tool is affected by a varying network size is described in [168]), we reduced the volume of nodes to 10% of the initial volume. Thus, we used 3500 nodes in total. The number of surrogate servers was reduced by the same ratio, ultimately resulting in 423 servers, including one origin server. They are depicted in Table 13.

Based on this placement, we defined the surrogate servers that will initially receive the user requests of each timezone. We calculated the distance between the coordinates of the

142 Twitter timezones and the 20 regions where the surrogate servers are placed according to the Limelight network. Depending on which surrogate region is closer to each timezone, we decided where the requests from this timezone will be redirected. To make this decision, we took the population of each timezone into account.

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

Table 13: Chapter 3 - Distribution of Servers over the World for the Experimental Evaluation

The next step was to place the surrogate servers in the network topology. Because the geographical coordinates do not coincide with the coordinates of the network topology, we used the NetGeo tool from CAIDA, which maps IP addresses and Autonomous System (AS) coordinates to geographical coordinates [174]. To find the latitude and longitude of an IP address or an AS, NetGeo initially has to find the element in its database, and then it searches in ARIN/APNIC/RIPE servers.

After extracting the geographical coordinates of the nodes of the network topology, the distances between servers were calculated. For example, to place the 55 surrogate servers in Washington DC, we found the 55 closest servers in the topology by comparing their geographical coordinates with those of Washington DC. Thus, we managed to obtain a realistic representation.

It would be impossible to define nodes representing users for every user crawled due to the large dataset and time required for each decision separately. Moreover, the set of users posing requests varies each time we run the request generator. To obtain a non-variant placement of the users in the topology, we had to represent all users of the same timezone as one in the topology. Therefore, all users from the timezone of London, for example, have

a new unique identity. This matching ultimately provides us with 162 zones (user groups), as the 142 zones of Twitter include some generic characterizations (7), e.g., Eastern Time and Central Time. We distributed the users belonging to these generic zones as well as users that have not defined a region in their account to unique codes, taking into account their population as a fairness criterion (3 zones for Pacific Time, 1 for International Time, 1 for Westafrica Time, 1 for Midatlantic Time, 4 for Eastern Time, 3 for Central Time, and 14 zones without region annotation, 27 in total).

After grouping users per timezone, each team of users was placed in a topology node. The users had to be placed in the nodes closer to those comprising the servers that serve the respective timezone requests, thus achieving a realistic network depiction. The distances among timezones of real ids of users and influence measurement metrics corresponding to the real ids of users were precalculated such that the simulation was not burdened.

The INET generator [56] allowed us to create an AS-level representation of the network topology. It took as input the number of nodes we wanted to exist in the topology, and it provided as output the file containing the topology. In this file appears the node id and its coordinates in the topology, as well as the edges among the nodes. In our scenarios, we used 3500 nodes.

3.4.2 Number of requests

The number of requests that the request generator provided us was set at approximately 1 million because CDNsim handles satisfyingly up to 1 million requests in general, with the number of objects being the dominant factor increasing the memory use of the tool. We used the same number of distinct videos for the generation of requests as used in [165].

3.4.3 Cache size

Because requests generated from the generator follow a long-tail distribution, as observed by plotting the data on a log-log graph, with the axes being rank order and frequency (Fig.16), 15 % of the whole catalog size was considered to be sufficient for experimentation.

3.4.4 Threshold values

Experimenting was conducted for time thresholds of 24 hours and 48 hours, as well as for the time threshold that covered all the requests. The authority threshold score covering

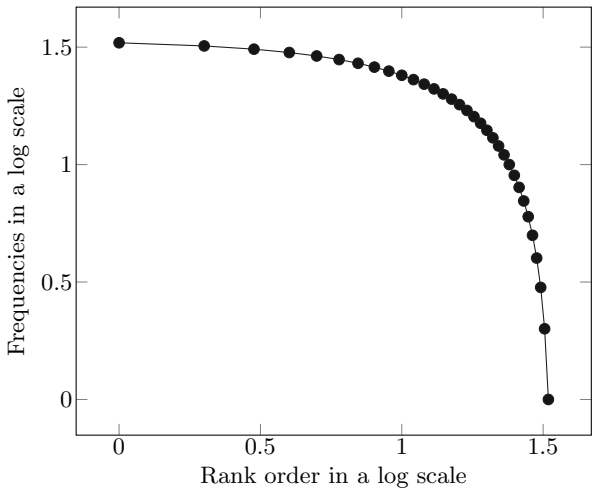


Figure 16: Rank orders - frequencies of requested objects in a log scale

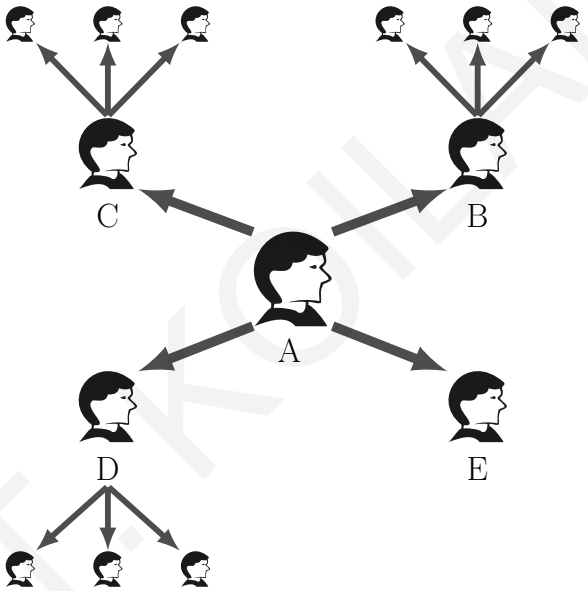


Figure 17: Graph sample

different percentages of authorities in our file was tested for various values (0.006 / 0.02 / 0.04).

3.4.5 Influence Measurement Metrics

- **Lobby-index**:

  The Lobby-index or l-index [120] of a node v is the largest integer l such that v has l neighbors with a degree of at least l. Extended for directed graphs, such as our dataset, rather than neighbors in this definition, we count followers. In Fig.17, for example, A is followed by 4 nodes: B, C, D and E. B, C, and D have 4 neighbors each (A and 3 followers), and E is a neighbor only with A. Thus, $l(A)$ here is 3.

- **HITS**:

  Hyperlink-Induced Topic Search (also known as Hubs and Authorities algorithm) [118] is a link analysis algorithm that rates web pages. A so-called good hub represents a page that points to many other pages, and a so-called good authority represents a page that is linked by many different hubs. In our sample graph, this corresponds to our intuition that node A is the least authoritative because it is the only node without incoming edges and that nodes B, C, D and E are equally important authorities. Indeed, the values here are $hub\_score_A = 0,999985$, $authority\_score_A = 0$, $hub\_score_B = 0,00317$, $authority\_score_B = 0,499990$, $hub\_score_C = 0,003171$, $authority\_score_C = 0,499990$, $hub\_score_D = 0,003171$, $authority\_score_D = 0,499990$, $hub\_score_E = 0$, and $authority\_score_E = 0,499990$.

For both Lobby-index and HITS metrics, we had to address memory usage issues for the very large graph dataset accommodated. HITS was calculated using the MapReduce technique.

3.5 Main findings

The statistic reports produced by the simulator are used to evaluate the proposed policy. A short explanation of the metrics used in our experiments for extracting statistical results using the simulator follows. They are described in detail in [168], along with various other metrics.

3.5.1 Metrics used

3.5.1.1 Client side measurements

They refer to activities of clients, namely, 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 i^{th} request. It starts at the timestamp when the request begins and ends at the timestamp when the connection closes.

3.5.1.2 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.
- **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.

3.5.1.3 Network statistics

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

- **Active surrogate servers**: refers to the servers being active serving clients.
- **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 mean response times for the clients.

Compared to Buzztraq [164], the novelty of this work 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, as in Buzztraq, but rather with a real dataset from multimedia links spread over an OSN.

The novelty in this case lies in the fact that no other CDN stand-alone tool has been augmented with a similar mechanism. With the extension proposed herein, we can experiment on social patterns without 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. To the best of our knowledge, no similar extensions exist toward this direction. By implementing and experimentally evaluating it, we prove that OSNs can affect content delivery.

Table 14 presents the average values of three parameters for six cases of testing after conducting a multitude of experiments (330) with varying threshold values and different influence measurement metrics. The lowest mean response times appear for the cases of the time threshold covering all requests for both centrality metrics. However, the hit ratio percentage is higher in the case where the time threshold covers all requests and the Lobby-index metric is used. In general, the use of the Lobby-index score metric results in higher hit ratio percentages compared to the equivalent HITS score cases.

Cases	H.Ratio%	Surrogates	M.R.T. (x10^{-2} sec.)
Lobby 48h	32,342	325	1,14
Lobby 24h	32,630	326	1,13
Lobby all requests	34,306	325	1,11
HITS 48h	32,304	325	1,14
HITS 24h	32,495	325	1,13
HITS all requests	34,290	324	1,11

Table 14: Chapter 3 - Average Values of Simulation Results

Concerning comparison with related works, storage issue is not addressed in [176] 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.

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 [164] incorporated in our framework depicts a M.R.T. of 13.95011 ms., and Tailgate algorithm [176] 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 [164] 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).

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 mean response time 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.

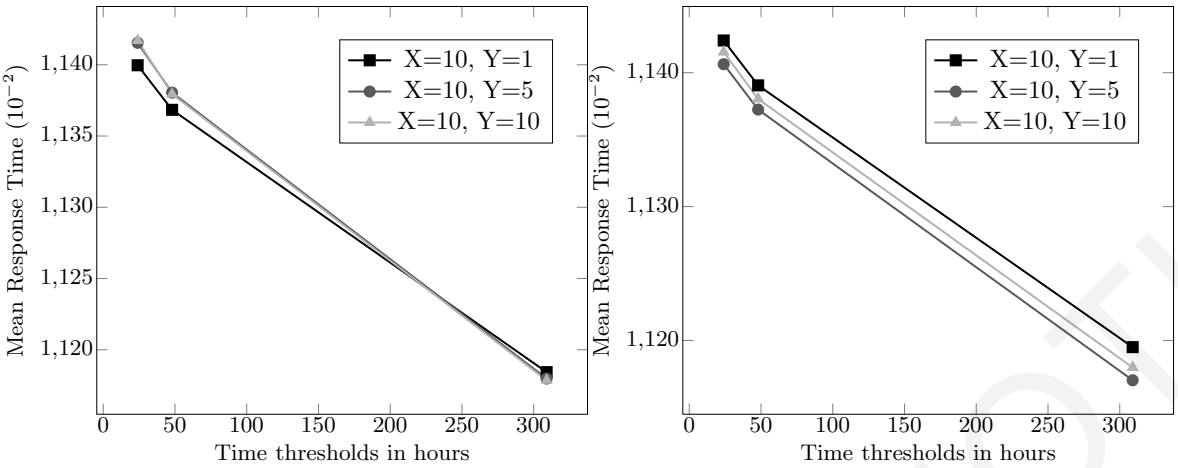
Most observed systems are assumed to be of either pure-push or pure-pull type with pure-push systems being, in fact, pure-broadcast and pure-pull systems, on the other hand, being pure-unicast, they use the broadcast channel just as a return path for point-to-point communication [189]. Practically, however, 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 mean response time under high loads and push strategies being superior under low to medium loads ([141], [75], [140]).

3.5.2 Impact of time threshold duration

- **Mean response time**:

  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, and in all cases of our experiments, this is less than the mean response time of the simulator without



(a) Lobby (b) HITS

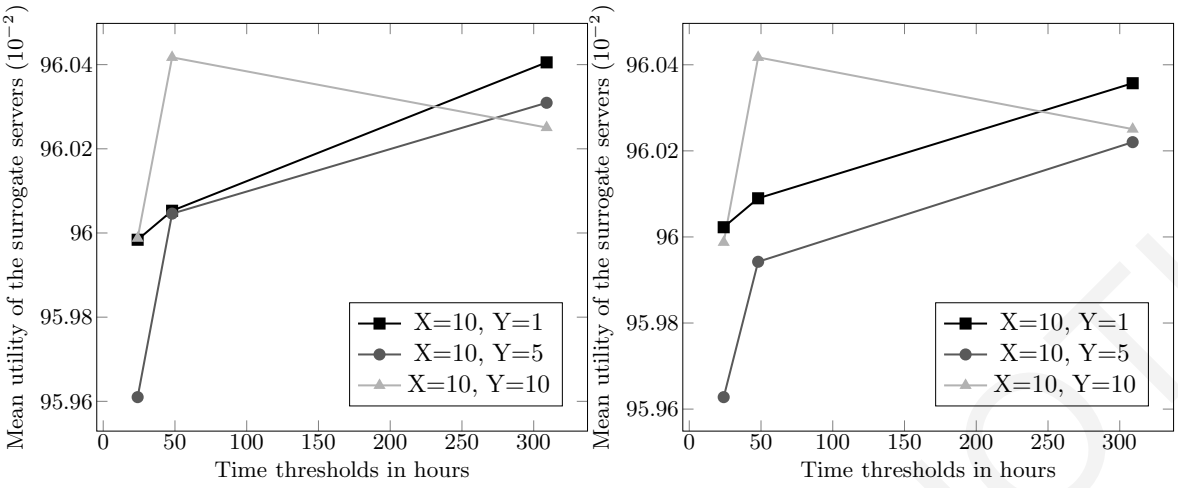
Figure 18: Chapter 3 - Effect of time threshold duration on mean response time for X closest timezones with mutual followers and Y timezones with the highest metric, where copying is ultimately performed

the dynamic policy we applied (1.846021059 x 10-2 sec.). For thresholds smaller than 24-h there is also substantial difference compared to the plain CDN simulation (1.4135 x 10-2 sec. for 12-h), although not better than the performance of our system for higher thresholds. Here, we present indicative values for the 10 closest timezones of mutual followers and varying subsets of 1, 5 and 10 timezones with the highest influence metric (both Lobby and HITS), respectively, where copying will ultimately be performed (Fig. 18) .

- **Mean utility of the surrogate servers**:

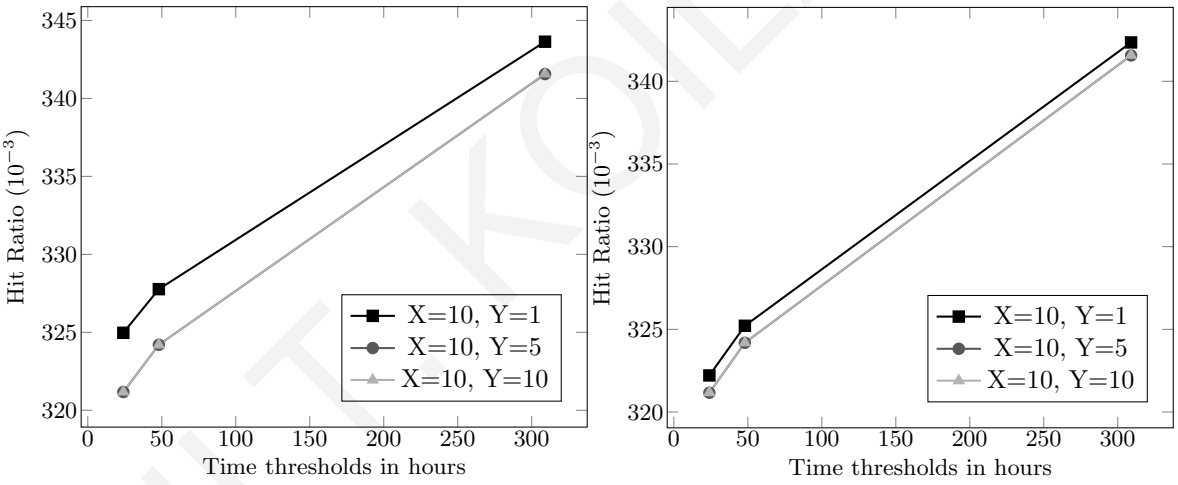
  For the time threshold of 48 h and the case of 10 timezones where copying is ultimately performed, the mean utility of the surrogate servers shows a peak for both influence measurement metrics and decreases for the hours covering the entire set of requests. Here, we present indicative values for the 10 closest timezones of mutual followers and varying subsets of 1, 5 and 10 timezones with the highest influence metric (both Lobby and HITS), respectively (Fig. 19).

- **Hit Ratio**: As the time threshold increases from 24 to 48 h and to hours covering the entire set of requests, we observe that the hit ratio increases. Surpassing the plain simulator hit ratio of 16.36% in all cases, the ratio steadily increases as the time threshold becomes larger. This result is not unexpected because more requests are examined and more copies are likely to be performed (Fig. 20).



(a) Lobby (b) HITS

Figure 19: Chapter 3 - Effect of time threshold duration on mean utility of the surrogate servers for X closest timezones with mutual followers and Y timezones with the highest metric, where copying is ultimately performed



(a) Lobby (b) HITS

Figure 20: Chapter 3 - Effect of time threshold duration on hit ratio for X closest timezones with mutual followers and Y timezones with the highest metric, where copying is ultimately performed

3.5.3 Impact of the number of timezones

- **Mean response time**:

  Concerning the mean response time, the primary metric for depicting performance improvement in our system, we observe an improvement of up to 40% (Table 37). Scellato et al. observed a 55%-70% performance improvement for a substantially smaller percentage of the catalog, state, although that performance improvement is

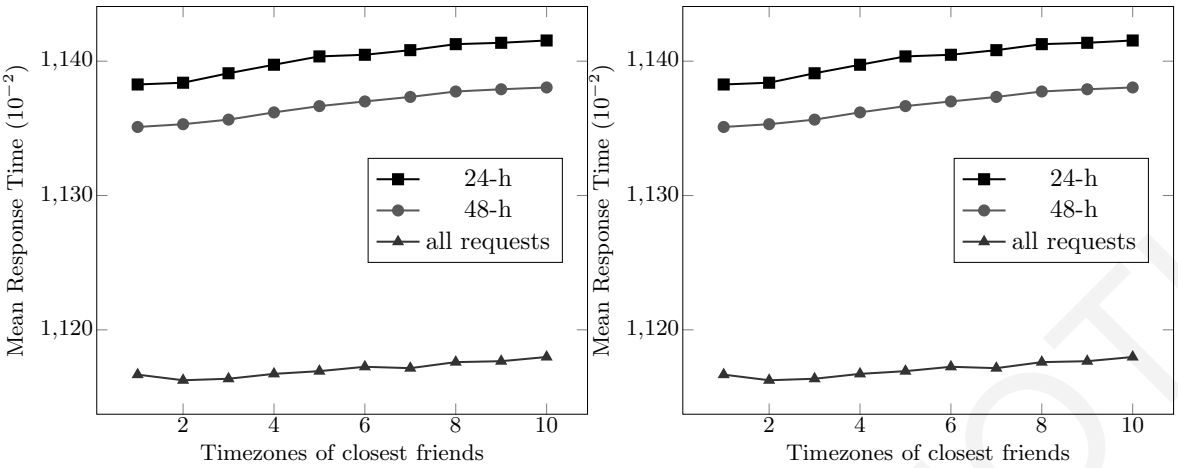
smaller as the cache becomes larger. The cost of their approach also includes the calculation of the geocascade metric.

Number of timezones	**Mean response time (*10^{-2} sec.)**
1	1,11950164004
2	1,11860275262
3	1,11782253597
4	1,11731668785
5	1,11702639155
6	1,11730670104
7	1,11713312804
8	1,11743489586
9	1,11743746926
10	1,11797775248

Table 15: Chapter 3 - Simulation Results

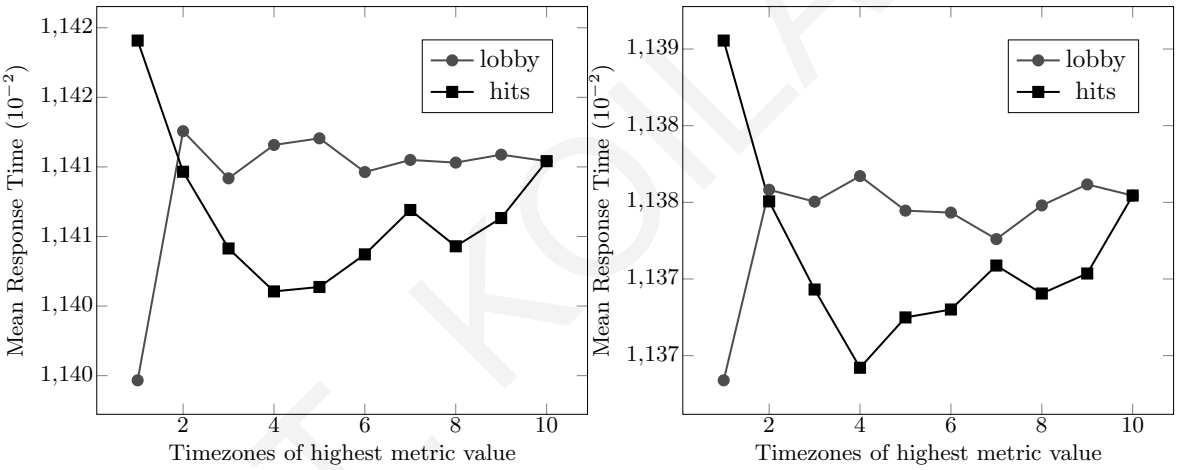
Expressing the trade-off between the reduction of the response time and the cost of copying in servers, we observe a point where the mean response time starts to increase again. The cost for every copy is related to the number of hops among the client asking for it and the server where copying is likely to be made, according to the *Put* function. For this most representative case (HITS, all requests), the values of mean response time for various numbers of timezones with the highest metric value are depicted in Table 15.

- **Number of closest timezones of mutual followers**: In Fig. 21, we observe that the mean response time increases as the number of closest timezones with mutual followers increases. We assume that the number of timezones with the highest metric value is the maximum allowed for each case, i.e., the number of closest timezones with mutual followers. Lower times are observed when a threshold covering all the requests is used. As naturally expected, there exists no difference for different influence measurement metrics because all closest timezones of mutual friends are taken into consideration. Figures 22 and 23 depict the effect of influence measurement on mean response time and active servers for $X = 10$ closest timezones with mutual followers respectively.



(a) Lobby (b) HITS

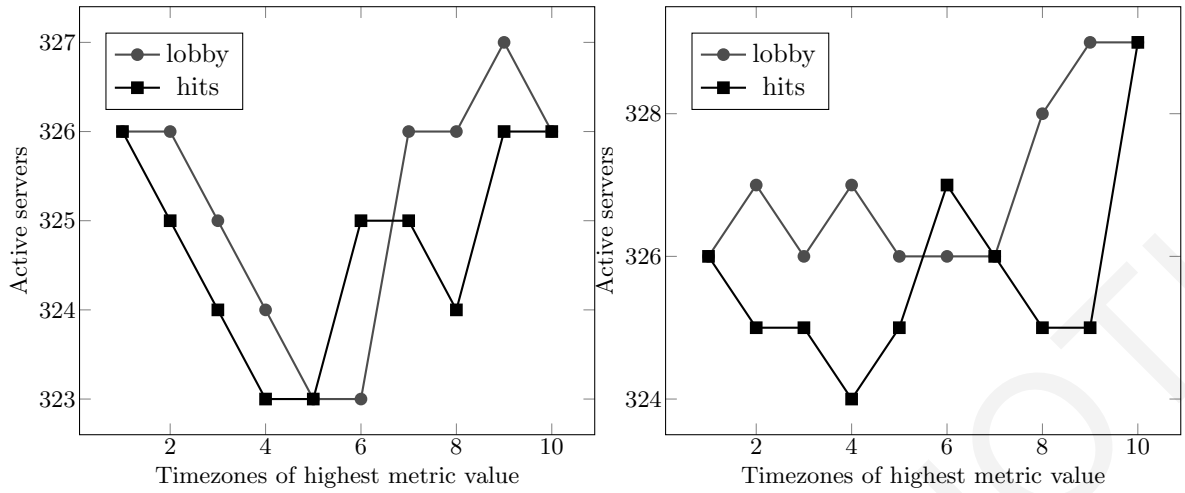
Figure 21: Chapter 3 - Effect of timezones used as X on mean response time for X closest timezones with mutual followers



(a) 24-h (b) 48-h

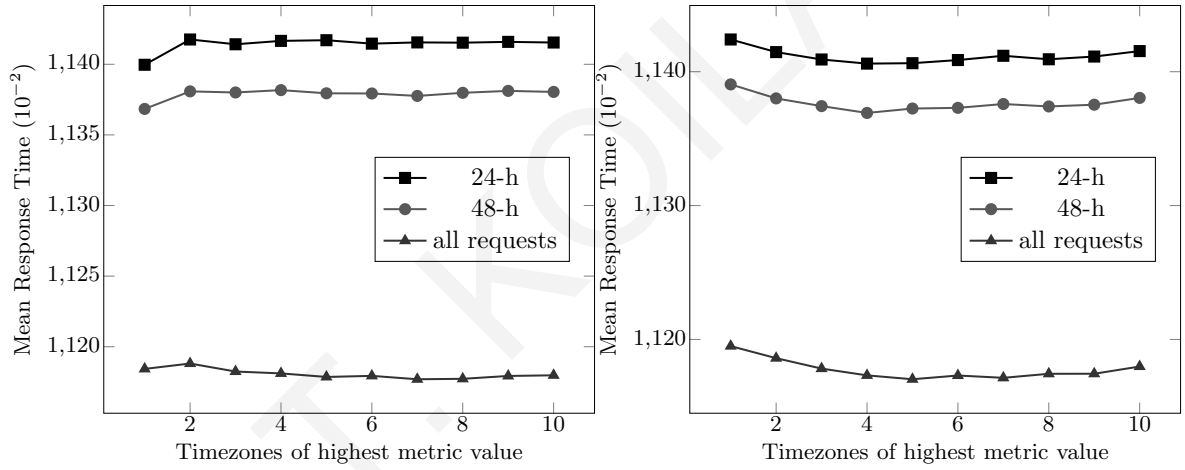
Figure 22: Chapter 3 - Effect of influence measurement on mean response time for $X = 10$ closest timezones with mutual followers

- **Number of timezones with highest metric value**: For both metrics, we observe a decrease in the mean response time as timezones with higher metric values increase (for a fixed number of closest timezones with mutual followers). In Fig. 24, we observe that this decrease occurs with approximately 5 timezones out of the 10 used. After this point, there is slight increase in the mean response time, which is attributed to the delay for copying content to surrogate servers. For the HITS case, this point-of-change is well depicted in Fig. 25.



(a) 24-h (b) 48-h

Figure 23: Chapter 3 - Effect of influence measurement on active servers for $X = 10$ closest timezones with mutual followers



(a) Lobby (b) HITS

Figure 24: Chapter 3 - Effect of timezones used as Y on mean response time for $X = 10$ closest timezones with mutual followers

3.5.4 Impact of authority threshold duration

The authority threshold score covering different percentages of authorities in our file was tested for various values (0.006 /0.02 / 0.04), although not depicting the worth-mentioning difference. This occurs because the percentage of authorities included in the set of nodes of our set of requests does not substantially change for the precalculated set of requests obtained from our request generator.

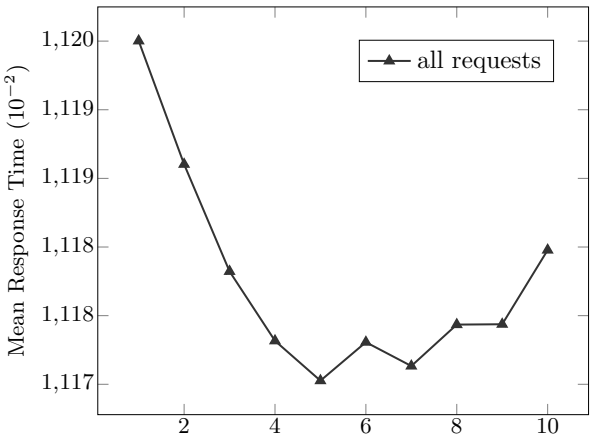


Figure 25: Chapter 3 - Mean response time for X=10 closest timezones with mutual followers and all possible Y values, $Y \in [1, 10]$

3.6 Discussion and Conclusions

Our approach in the implementation of Social Prefetcher was CDN-centric. In case we used a video-centric or user-centric approach, we could alternatively use other metrics to measure the quality of user experience.

Due to the use of TCP, that 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 [99]. 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 [99].

Hence, alternative metrics include, for example, the proportion of a file that is downloaded during the initial buffering stage, after which the playout of the video is smooth. We say the playout is smooth if the download rate for a file drops by a percentage of what was the original rate. On average, the playback is smooth after 15% of a file is downloaded, therefore we can note the delay in terms of time it takes for the first 15% of the file to be downloaded [176] instead of the total M.R.T..

If alternatively Q_{hit}, the number of requests served from surrogate servers of the region of the user was leveraged within the Social Prefetcher algorithm, we could use it to balance the algorithm in each step. The knowledge of Q_{hit} as well as its ratio to the total number of requests to the surrogate server (Q_{total_i}), could support the selection of the servers surpassing a threshold of hit ratio for user requests. Selection could be enforced with assignment of relative weights. Different variations of this load-balancing component might include the distance of the surrogate servers along with their hit ratio, leading to possible reduction of bandwidth and copy costs. Thus, although in the Social Prefetcher algorithm, we aim at maximizing the hit ratio with the application of various heuristics, we could also possibly add hit ratio as a feedback parameter that will calibrate the algorithm.

Understanding the effects of social cascading on bandwidth-intensive multimedia content over the Web is of great importance toward improving the performance of CDNs. In the related literature, many works lack non-synthetic workloads for controlling and evaluating the proposed systems, whereas with the described generator, we ensure the reliability of the data used in our experiments.

In this work, we implemented and experimentally evaluated a dynamic policy of content prefetching, and thus, we have in our hands proof that OSNs can affect the content delivery infrastructure. The dynamic policy described here required a change of the CDNsim simulator concerning request manipulation, object placement, and the object replacement policy after the arrival of new objects.

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. Consequently, we have to search for new ways to process the large volume of multimedia data created on a daily basis. The dynamic policy proposed herein can be further enriched with contextual information. In the future, we could implement and experiment on variations of dynamic policy that aim to improve it with more complex functions, incorporating timezone differences, to prefetch the content in the hours with less traffic as well as incorporate information about the popularity of videos. Thus, it is likely to provide better results toward maximization of CDN performance and reduction of content replication costs.

Chapter 4

Efficient Timing of Content Placement

With multimedia content providers requiring CDN services to enable the delivery of bandwindth-demanding media to end-users, and the growth of HTTP traffic due to media files circulating over OSNs, a OSN-awareness mechanism over a CDN becomes essential, to mitigate the considerable weight placed on bandwidth. A social awareness mechanism augmented to a stand-alone CDN traffic simulator addresses the issue of which content will be copied in the surrogate servers of a CDN infrastructure and to what extent, and, hence, ensures an optimized content diffusion placement (Chapter 3). Herein we further address the issue of temporal diffusion, related to the most efficient timing of the content placement. We exploit the knowledge of peak times for upload and download, so that content is prefetched in the hours with less traffic. We also incorporate other contextual information,such as the viewership within the media service, to ensure performance optimization. Our variations are experimentally proven to contribute toward maximization of CDNs performance and reduction of content replication costs [111].

The remainder of this Chapter is organized as follows. Section 4.1 gives an example of necessity of the approach presented in this Chapter. Section 4.2 formally describes the addressed problem. The proposed algorithm is described in Section 4.3. Section 4.4 gives an outline of the methodology, along with the preparation of the employed datasets. Our main findings are presented in Section 4.5. Section 4.6 concludes the Chapter and discusses directions for future work.

4.1 Introduction

The amount of traffic generated on a daily basis by online multimedia streaming providers is multiplied by the transmission over OSNs [52]. Extended use of OSNs [76], [40], [58],

and the increasing popularity of streaming media are the factors that determine the internet traffic growth [56]. Hence, CDN users can benefit from an incorporated mechanism of OSN-awareness over the CDN infrastructure. In [110] Kilanioti incorporates a dynamic mechanism of preactive copying of content to an existing validated CDN simulation tool and proposes an efficient copying policy. The latter can be based on prediction of demand in social networks.

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).

4.1.1 Contributions

A real dataset of UGC is used. It includes YouTube links over an OSN platform, thus social cascades are directly analyzed. Real restrictions of a CDN infrastructure (storage issues, network topology) are taken into account. The proposed algorithm also suggests a mechanism that overcomes the testing limitations of other existing CDN platforms, that either treat CDN policies as black boxes or need third users for experimentation. This work extends the Social Prefetcher algorithm [110] to include information about peak-time of various timezones of our geo-diverse system, as well as contextual information about the viewership of

video content within the media service. It implements extensions in two variations and incorporates them in a validated simulator for CDNs. A multitude of experiments shows improved metrics for performance measurement over content delivery.

Experimentation is conducted on a Twitter dataset containing 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.

As for the main findings of our work, they can be exploited for future policies complementary to existing CDN solutions or incorporated to OSN providers mechanisms, to handle larger scale data. In this work we examine which parameters (number of timezones examined, time threshold duration) affect the CDN metrics the most. The optimization of our algorithm is proved in [110], whereas herein the incorporation of peak hours and popularity of circulating objects information are examined to furthermore enhance its performance.

4.2 Problem Description

We aim at improving the performance of the CDN infrastructure in terms of reducing the response time, improving the hit ratio of our request, as well as restricting 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, knowledge of peak times for upload and download and the viewership within the media service, we aim at recognizing 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, the peak time start and peak time end for each region $pts_i, pte_i, 1 < i < \tau$, the posts P of the nodes, and the popularity $\Pi_i, 1 < i < w$ of objects, it recognizes the set of objects O that will be popular only in a subset of the regions (Table 16). In these regions is the content likely to be copied. The policy is represented by the function $Put(n_i, Predict(G,P,R,pts_i,pte_i,\Pi_i,O))$, 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:

$G(V,E)$	Graph representing the social network
$V = \{V_1, ..., V_n\}$	Nodes representing the social network users
$E = \{E_{11}, ..., E_{1n}, ..., E_{nn}\}$	Edges representing the social network connections, where E_{ij} stands for friendship between i and j
$R = \{r_1, r_2, ..., r_\tau\}$	Regions set
$N = \{n_1, n_2, ..., n_u\}$	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
$O = \{o_1, o_2, ..., o_w\}$	Objects set (videos), denoting the objects users can ask for and share
S_i, $o_i \in O$	Size of object i in bytes
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
$P = \{p_{12}, p_{13}, ..., p_{nw}\}$	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
$Q = \{q_1, q_2, ..., q_\zeta\}$	Object requests, where q_i denotes a request for an object of set O
Q_{hit_i}, $Q_{total_i}, i \in N$	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

Table 16: Chapter 4 - Notation Overview

$$\frac{Q_{hit_i}}{Q_{total_i}}, , i \in N \tag{18}$$

is maximum, whereas constraint

$$\sum_{\forall i \in O} S_i f_{ik} \leq C_k \tag{19}$$

is fulfilled, where:

$$f_{ik} = \begin{cases} 1 & \text{if object } i \text{ exists in the cache of surrogate server } k \\ 0 & \text{if object does not exist} \end{cases} \tag{20}$$

It returns the set of objects $o \in O$ that have to be placed in surrogate server $n_i \in N$. For $x, 1 < x < w$, objects that will be copied in the surrogate server and the capacity Cn_i of n_i, (2) has to be fulfilled:

$$S_1 + S_2 + S_3 + ... + S_x \leq Cn_i(1) \tag{21}$$

4.3 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. 26).

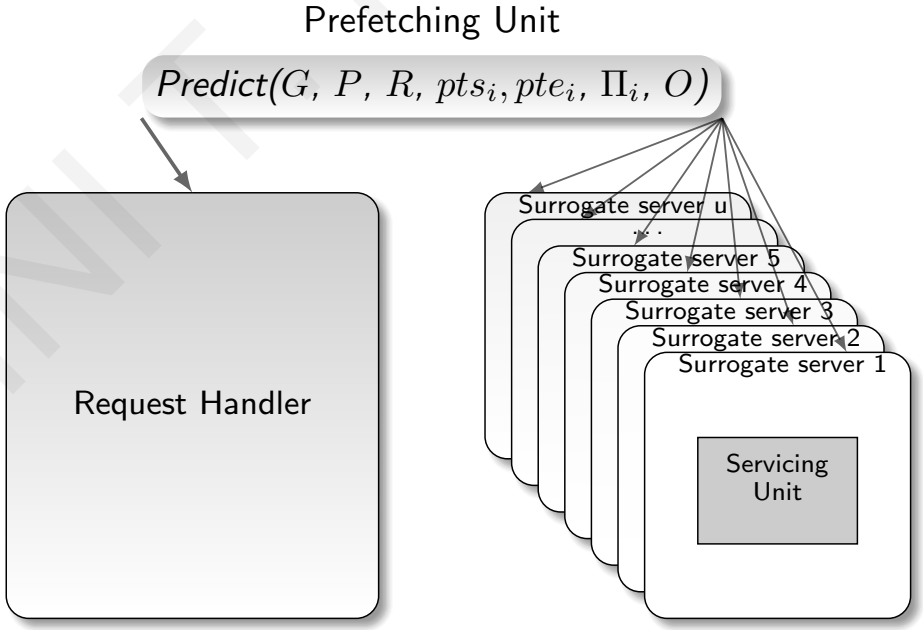


Figure 26: Chapter 4 - The Prefetching Unit

4.3.1 For Every New Request in the CDN:

The main idea is to check whether specific time has elapsed 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 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 (*Variation-1*, Fig. 27). In *Variation-2* (Fig. 28)

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** $o.\Pi_i > \Pi_i\_threshold$ **then**
18: copy object o to surrogates n_j that Subpolicy I decides;
19: **end if**

Figure 27: Chapter 4 - Variation-1 - Algorithm for every new request ($timestamp$, V_i, o) in the CDN

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 **then**
18: **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**
19: copy object o to surrogates n_j that Subpolicy I decides;
20: **end if**
21: **end if**

Figure 28: Chapter 4 - Variation-2 - Algorithm for every new request ($timestamp$, V_i, o) in the CDN

we check the importance of the user, as well as if the time of the transmission is not within the peak-time range of the region of the user ([4]).

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

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**

Figure 29: Chapter 4 - Subpolicy I

Centrality is measured with the HITS algorithm, described in Section 4.4. Subpolicy (Fig. 29) 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.

4.3.2 For Every New Object in the Surrogate Server:

For both variations, 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 find the oldest objects and delete them. In the case that there are no such objects, we delete those with the largest timestamp in the cascade. In all other cases, the Least Recently Used (LRU) policy is applied for the removal of objects. The above are depicted in Fig. 30.

The heuristics applied in our approach are based on the following observations [110]: 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 ([165], [60]). The percentage of cascades proceeding for days may not be directly attributed to the influence that a social contact exerts, as the video can appear in the newsfeed of the user for a longer time or a new cascade of this specific object may have started. However, in our prefetching algorithm we experiment with varying time thresholds of 24 hours, 48 hours, and threshold covering the entire percentage of requests (threshold for the cascade effect and the time that an object remains in cache).The idea is to check whether specific time has elapsed 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.

```
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
```

Figure 30: Chapter 4 - Algorithm for every new object o in the surrogate server n_k

Number of nodes in the topology	3500
Redirection Policy	Cooperative Environment (closest surrogate)
Number of origin servers	1
Number of surrogate servers	423
User groups	162
Bandwidth	100 Mbit/sec

Table 17: Chapter 4 - Simulation Characteristics

4.4 Experimental Evaluation

For the experimental evaluation, we used the CDNsim simulator for CDNs (Chapter 3). The configuration of the simulation values is shown in Table 17. We conducted a multitude of experiments (110, 55 for each variation), in which the time thresholds varied. For the extraction of 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.

4.4.1 Network Topology

There follows a short description of the process to define the nodes in the topology. These nodes represent the surrogate servers, the origin servers, and the users making the object requests (Fig. 31). For an in-depth analysis you can refer to [110]. To simulate our policy and place the servers in a real geographical position, we used the geographical distribution of the Limelight network [54]. For the smooth operation of the simulator the number of surrogate servers was reduced by the ratio of 10%, to ultimately include 423 servers (Table 18). Depending on which surrogate region of the 20 the Limelight network defines is closer to each of the 142 Twitter timezones, we decided where the requests from this timezone will be redirected. The population of each timezone was also taken into consideration. The INET generator [56] 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, a tool that maps IP addresses and AS coordinates to geographical coordinates [174], and surrogate servers were assigned to topology nodes.

After grouping users per timezone (due to the limitations the large dataset imposes), each team of users was placed in a topology node. We placed the users in the nodes closer to those comprising the servers that serve the respective timezone requests, contributing this way to a realistic network depiction.

4.4.2 Number of Requests

1 million requests were considered sufficient, as CDNsim handles satisfyingly up to so many in general, with the number of objects being the dominant factor increasing the memory use of the tool. Also similar concept approaches use similar number of requests ([176] on a daily basis and [165]), and same number of distinct videos for generation of requests.

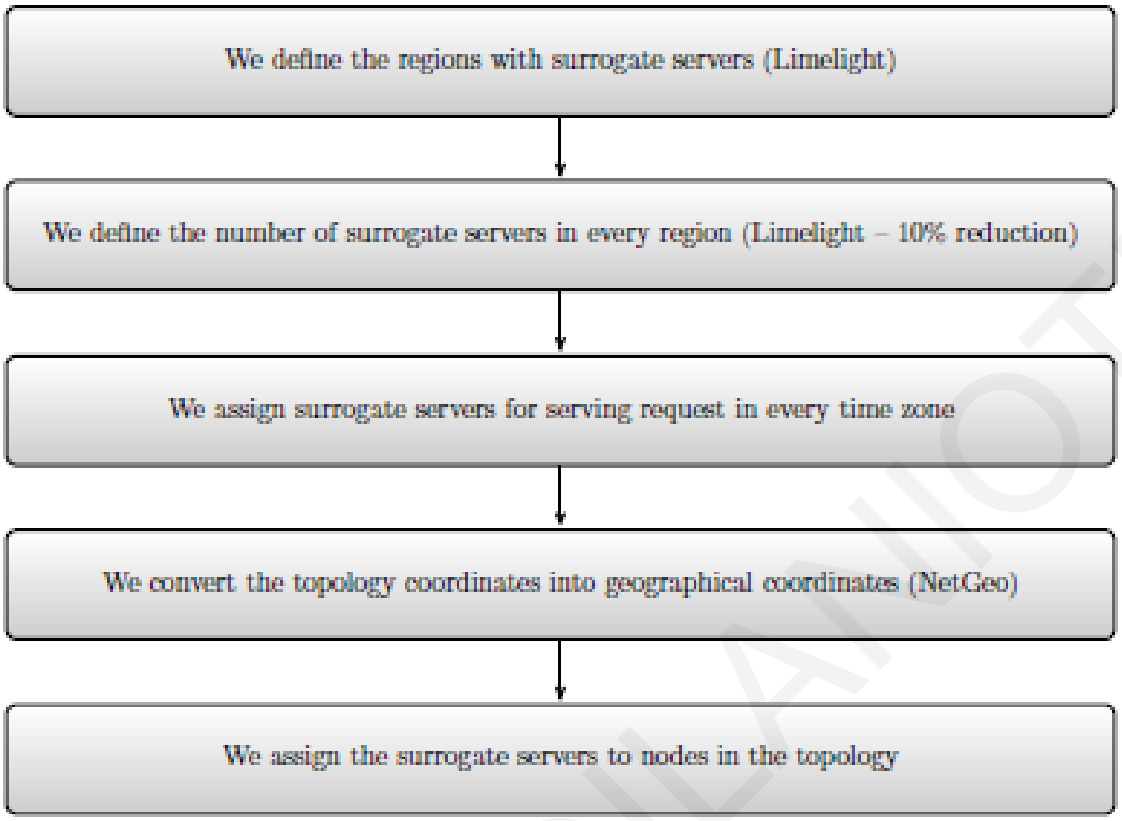


Figure 31: Chapter 4 - Methodology followed

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

Table 18: Chapter 4 - Distribution of Servers over the World for the Experimental Evaluation

With the requests generated from the generator following a long-tail distribution, 15 % of the whole catalog size was considered to be sufficient.

4.4.3 Threshold Values

Experimenting was conducted for time thresholds of 24 hours and 48 hours, as well as for the time threshold that covered all the requests. The threshold value for media service viewership was moderately chosen as 402408 (average media viewership in the dataset). The authority threshold score was tested for various values (0.006 / 0.02 / 0.04).

4.4.4 Influence Measurement Metrics

HITS algorithm [118] is a link analysis algorithm that rates web pages. Twitter uses a HITS style algorithm to suggest to users which accounts to follow [94], as well. A so-called good hub represents a page that points to many other pages, and a so-called good authority represents a page that is linked by many different hubs. We had to address memory usage issues for the very large graph dataset accommodated, and HITS was calculated using the MapReduce technique.

4.5 Main Findings

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

4.5.1 Metrics Used

4.5.1.1 Client Side Measurements

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 i^{th} request. It starts at the timestamp when the request begins and ends at the timestamp when the connection closes.

4.5.1.2 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 in the surrogate servers.

4.5.1.3 Network Statistics

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

- *Active Surrogate Servers*: refers to the servers being active serving clients.
- *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 mean response times for the clients.

After conducting a multitude of experiments (55 for each variation) with varying threshold values, we reached the following conclusions.

Table 19 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 both variations. In general, we observe a better performance in terms of mean response times and hit ratios achieved for the Variation-1, where the viewership within the YouTube platform is considered. Both variations perform better than the Social Prefetcher approach.

	Mean response time (Avg, 10^-2 sec.)	**Hit ratio (Avg, %)**	**Active servers**	**Mean utility (Avg, %)**
Variation-1 - 24-h	1.1383	32.81	326	96.01
Variation-1 - 48-h	1.1352	33.08	326	96.01
Variation-1 - all-h	1.1172	34.58	325	96.04
Variation-2 - 24-h	1.1411	32.13	325	95.98
Variation-2 - 48-h	1.1376	32.43	326	96.00
Variation-2 - all-h	1.1174	34.38	324	96.03
Social Prefetcher 24-h	1.1412	32.12	325	95.98
Social Prefetcher 48-h	1.1377	32.42	326	96.00
Social Prefetcher all-h	1.1181	34.16	325	96.01

Table 19: Chapter 4 - Average Metric Values for $X = 10$ Timezones of Close Mutual Friends

As we surpass the Social Prefetcher performance, the herein proposed policy outperforms related works like [176], where, moreover, storage issue is not addressed -although the proliferation of cloud services has led to a reduction in storage costs over the past years,

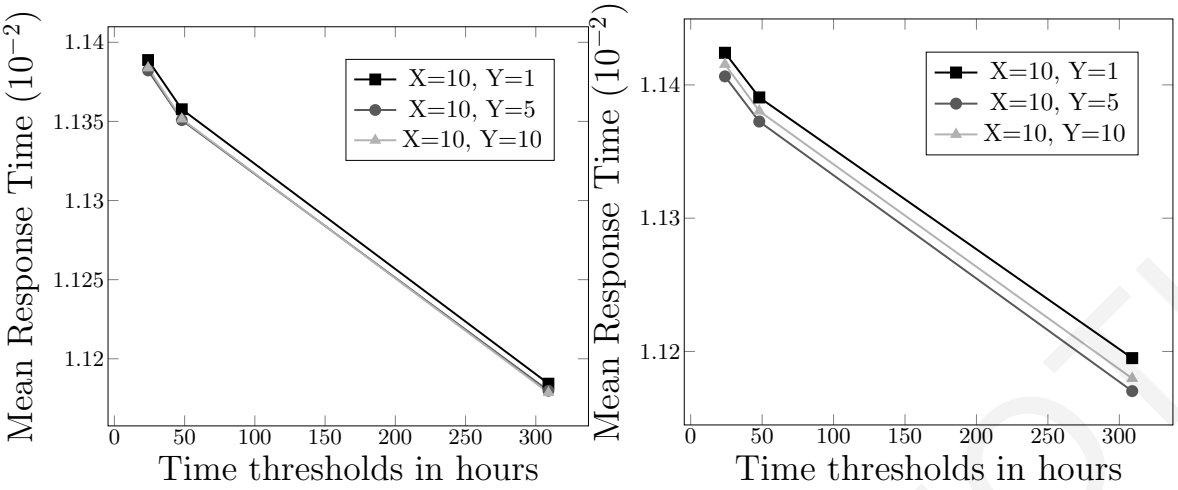


Figure 32: Chapter 4 - Effect of time threshold duration on mean response time for X closest timezones with mutual followers and Y timezones with the highest metric, where copying is ultimately performed for (i)Variation-1 and (ii)Variation-2

storage costs still remain a significant factor that can be reduced under certain conditions-(improvement over their respective improvement (30%) in pull-based methods employed by most CDNs) or [164], since better response times are depicted as well after incorporation of other social prediction algorithms for content delivery in our plain framework (1.395011 ms.).

4.5.2 Impact of Time Threshold Duration

- *on Mean Response Time*:

  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. Here, we present indicative values for the 10 closest timezones of mutual followers and varying subsets of 1, 5 and 10 timezones with the highest influence metric, respectively, where copying will ultimately be performed (Fig. 32) for both variations.

- *on Mean Utility of the Surrogate Servers*:

  With the exception of time threshold of 48 h for Variation-2, the mean utility of the surrogate servers shows a peak for both variations for the hours covering the entire set of requests. Here, we present indicative values for the 10 closest timezones of mutual followers and varying subsets of 1, 5 and 10 timezones with the highest influence metric (Fig. 33).

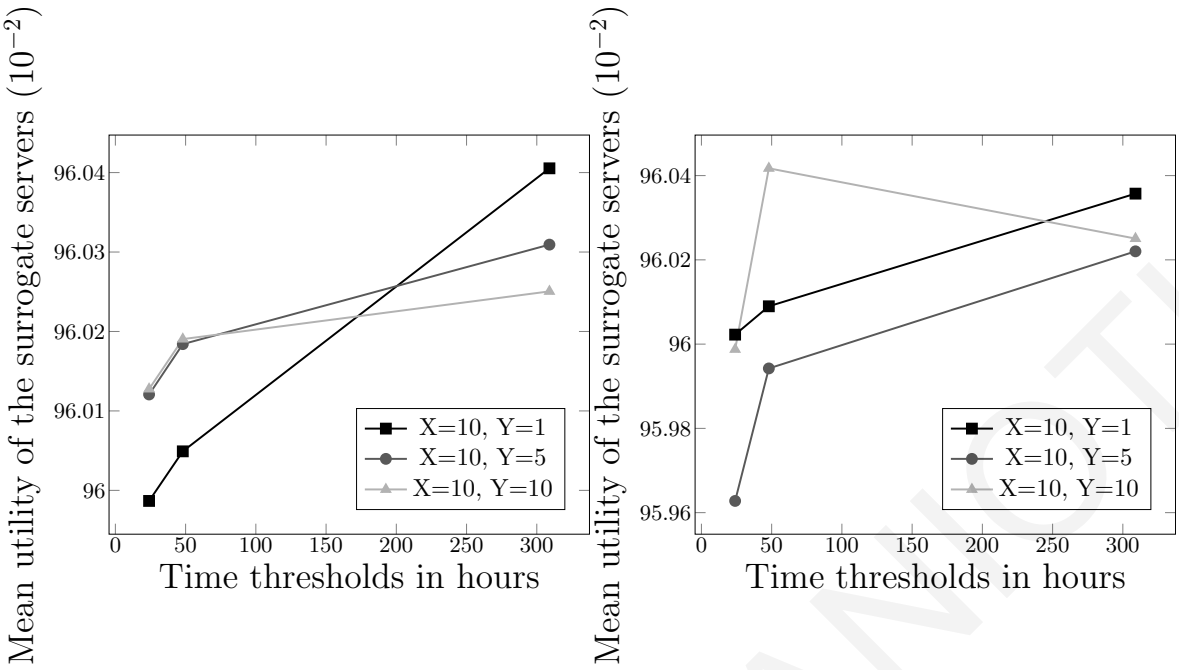


Figure 33: Chapter 4 - Effect of time threshold duration on mean utility of the surrogate servers for X closest timezones with mutual followers and Y timezones with the highest metric, where copying is ultimately performed for (i)Variation-1 and (ii)Variation-2

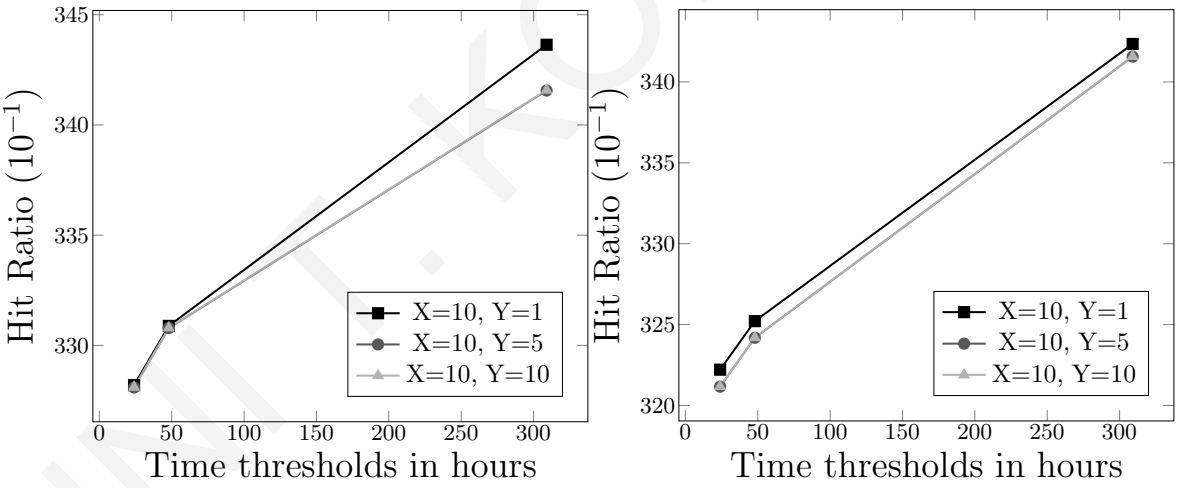


Figure 34: Chapter 4 - Effect of time threshold duration on hit ratio for X closest timezones with mutual followers and Y timezones with the highest metric, where copying is ultimately performed for (i)Variation-1 and (ii)Variation-2

- *on Hit Ratio*: As the time threshold increases from 24 to 48 h and to hours covering the entire set of requests, we observe that the hit ratio steadily increases for both variations. This result is not unexpected because more requests are examined and more copies are likely to be performed (Fig. 34).

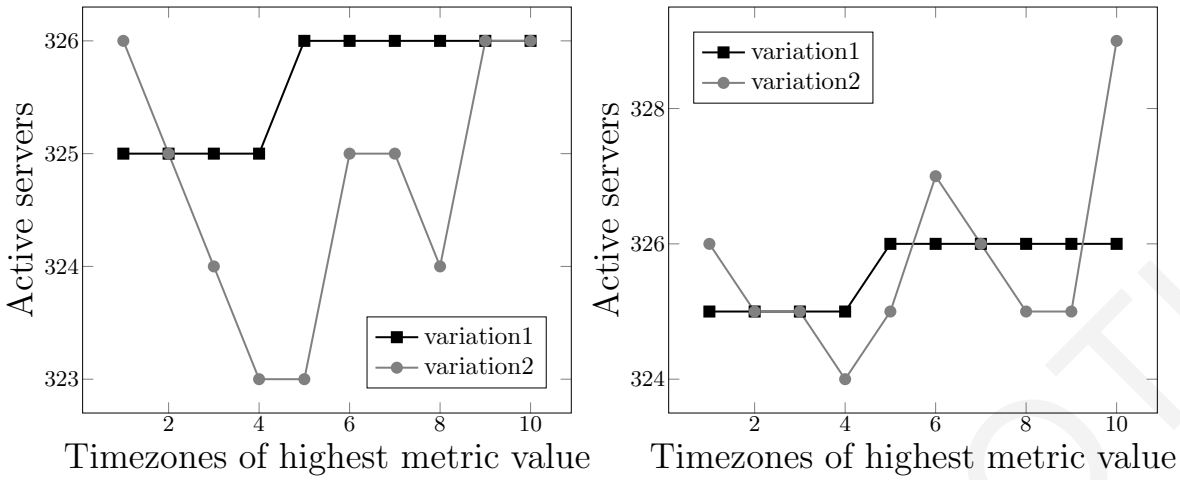


Figure 35: Chapter 4 - Effect of timezones used as Y on active servers for $X = 10$ closest timezones with mutual followers for (i)24-h and (ii)48-h

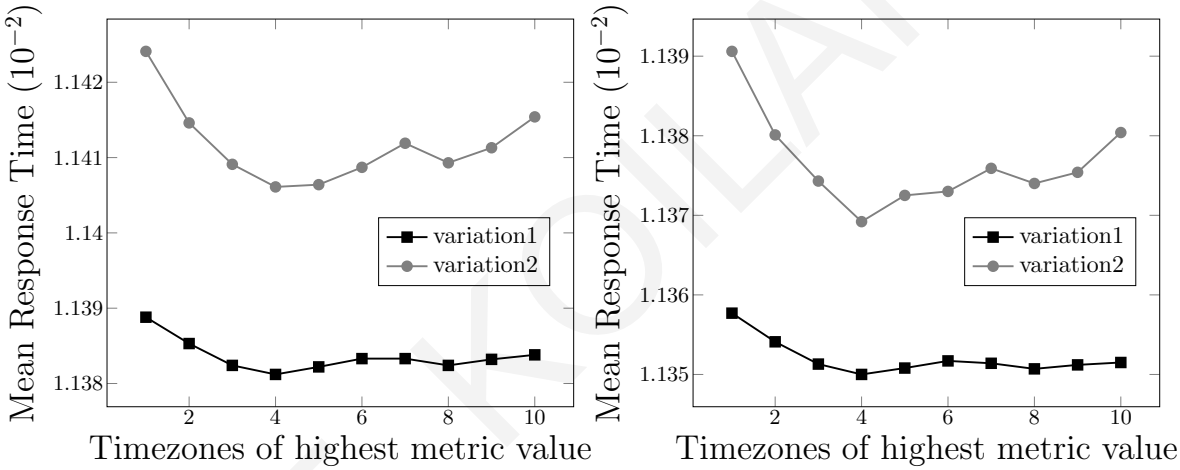


Figure 36: Chapter 4 - Effect of timezones used as Y on mean response time for $X = 10$ closest timezones with mutual followers for (i)24-h and (ii)48-h

4.5.3 Impact of the Number of Timezones

- *on Active Servers*:

  For a fixed number of 10 closest timezones with mutual followers Variation-1 appears to use less active servers after the first timezone of highest centrality in the 24-h scenario. In the 48-h scenario Variation-1 depicts higher values than Variation-2 for the cases of 1, 6 and 10 timezones examined (Fig. 35).

- *on Mean Response Time*: 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 (Fig. 36 and Fig. 37).

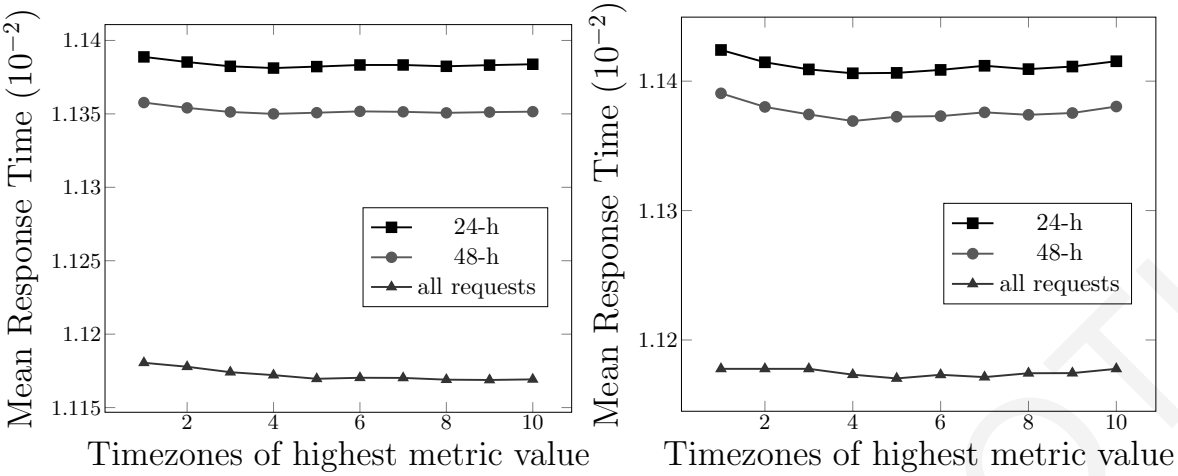


Figure 37: Chapter 4 - Effect of timezones used as Y on mean response time for $X = 10$ closest timezones with mutual followers for (i)Variation-1 and (ii)Variation-2

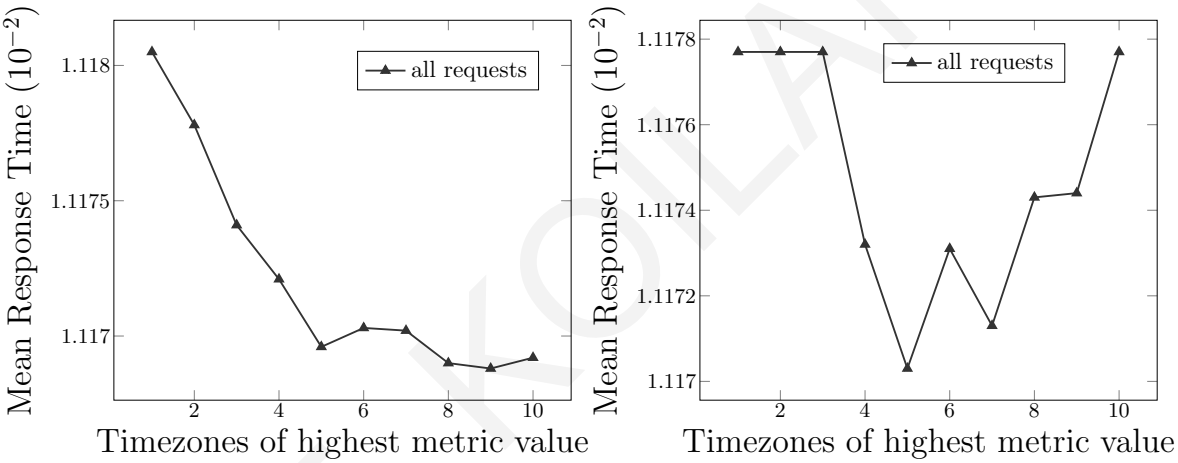


Figure 38: Chapter 4 - Mean response time for X=10 closest timezones with mutual followers and all possible Y values, $Y \in [1, 10]$ for (i)Variation-1 and (ii)Variation-2

For both variations this decrease in the mean response time occurs with approximately 4 timezones out of the 10 used (for a fixed number of closest timezones with mutual followers). 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 the server where copying is likely to be made, according to the *Put* function. This point-of-change is also depicted for our variations in Fig. 38 for the most representative case of all requests.

4.6 Conclusions

In this work, we further extended a dynamic policy of OSN content prefetching with temporal and other contextual parameters, depicting how OSNs can affect the content delivery infrastructure. We have presented how geo-social properties of users participating in social cascades prove to be of great importance toward improving the performance of CDNs and cloud, in the long-term. Bandwidth-intensive multimedia delivery over a CDN infrastructure is experimentally evaluated with non-synthetic workloads, that many works in the related literature lack.

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.

Chapter 5

Caching Schemes

The Social Prefetcher mechanism (Chapter 3) augmented to a stand-alone CDN traffic simulator addresses the issue of which content will be copied in the surrogate servers of a CDN infrastructure and to what extent, and, hence, ensures an optimized content diffusion placement. At the same time, it mitigates the considerable weight on bandwidth, that Big Data generated and circulated via OSNs places. Herein, we further address the issue of studying complementary efficient caching policies in the surrogate servers of the CDN infrastructure. Our variations are experimentally proven to outperform the plain mechanism behaviour in terms of user experience [116]. The rest of this chapter is structured as follows. Section 5.2 formally describes the addressed problem. The proposed algorithm is described in Section 5.3. Section 5.4 gives an outline of the methodology, along with the preparation of the employed datasets. Our main findings are presented in Section 5.5. Section 5.6 concludes the chapter and discusses directions for future work.

5.1 Introduction

Utilization of CDNs is likely to have profound effects on large data download through enhanced performance, scalability and cost reduction. Extended use of OSNs [76], [40], [58], and the increasing popularity of streaming media are the factors that drive the HTTP traffic growth [56]. 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 [52] being published per minute). Subsequently, CDN users can benefit from an incorporated mechanism of OSN-awareness over the CDN infrastructure. In [110], [111] 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.

5.1.1 Contributions

In this work [115] we modify the Social Prefetcher algorithm [110], [111] 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 [159]. Social cascades are directly analyzed, as the real dataset of 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 [111] in terms of performance. The latter surpasses performance improvement (30%) 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. The optimization of our algorithm is proved in Chapter 3 ([110]), and its performance is further enhanced. 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.

5.2 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, inrease 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, the peak time start and peak time end for each region $pts_i, pte_i, 1 < i < \tau$, the posts P of the nodes, and the popularity $\Pi_i, 1 < i < w$ of objects, it recognizes the set of objects O that will be popular only in a subset of the regions (Table 20). There is the content likely to be copied. The policy is represented by the function $Put(n_i, Predict(G,P,R,i_\kappa,\Delta T_{i_\kappa},O))$, 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:

$$\frac{Q_{hit_i}}{Q_{total_i}}, i \in N \tag{22}$$

is maximum, whereas constraint

$$\sum_{\forall i \in O} S_i f_{ik} \leq C_k \tag{23}$$

is fulfilled, where:

$$f_{ik} = \begin{cases} 1 & \text{if object } i \text{ exists in the cache of surrogate server } k \\ 0 & \text{if object does not exist} \end{cases} \tag{24}$$

It returns the set of objects $o \in O$ that have to be placed in surrogate server $n_i \in N$. For $x, 1 < x < w$, objects that will be copied in the surrogate server and the capacity Cn_i of n_i, (2) has to be fulfilled:

$$S_1 + S_2 + S_3 + \cdots + S_x \leq Cn_i(1) \tag{25}$$

$G(V,E)$	Graph representing the social network
$V = \{V_1, ..., V_n\}$	Nodes representing the OSN users
$E = \{E_{11}, ..., E_{1n}, ..., E_{nn}\}$	Edges representing the OSN connections
$R = \{r_1, r_2, ..., r_\tau\}$	Regions set
$N = \{n_1, n_2, ..., n_u\}$	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
$O = \{o_1, o_2, ..., o_w\}$	Objects set (videos)
$S_i, i \in O$	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
$P = \{p_{12}, p_{13}, ..., p_{nw}\}$	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
$Q = \{q_1, q_2, ..., q_\zeta\}$	Object requests, where q_i denotes a request for an object of set O
Q_{hit_i}, $Q_{total_i}, i \in N$	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/ with highest lobby values/ with highest HITS values

Table 20: Chapter 5 - Notation Overview

5.3 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. 39).

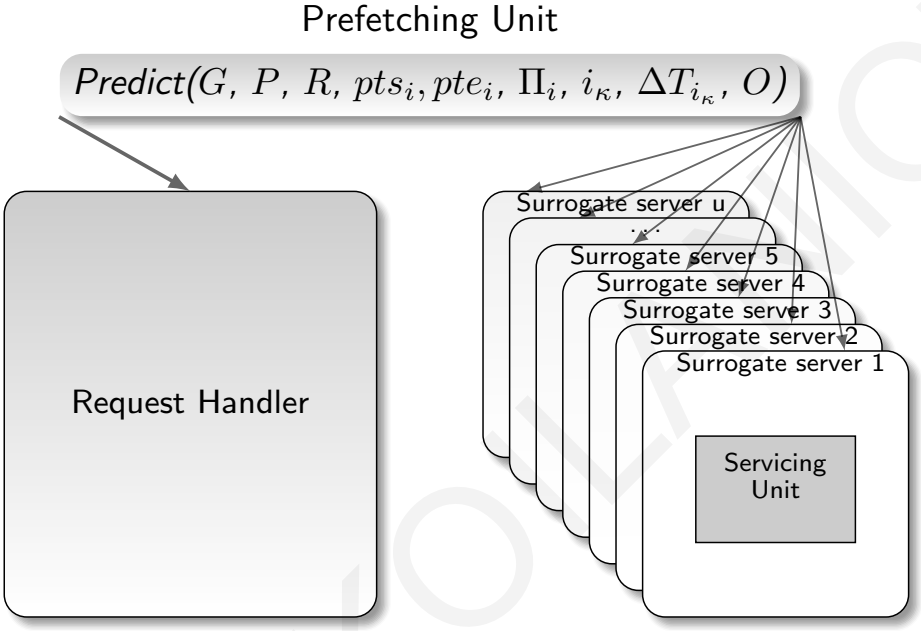


Figure 39: Chapter 5 - The Prefetching Unit

5.3.1 For Every New Request in the CDN

The main idea is to check whether specific time has elapsed 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 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. 40).

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 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 [118], a link analysis algorithm that rates web pages. Twitter uses a HITS style algorithm to suggest to users which accounts to follow [94], as well. A so-called good hub represents a page that points to many other pages, and a so-called 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 I 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.

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** $o.\Pi_i > \Pi_i\_threshold$ **then**
18: copy object o to surrogates n_j that Subpolicy I decides;
19: **end if**

Figure 40: Chapter 5 - Algorithm for every new request ($timestamp$, V_i, o) in the CDN

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

Figure 41: Chapter 5 - Subpolicy I

if $o.size + current\_cache\_size \leq total\_cache\_size$ **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 all 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 **or** $CandidateList.size$==$total\_cache\_size$ **then**
                use LRU to delete any object $o \in O$;
            end if
        end for
    end while
    put object o to cache of surrogate n_k;
end if

Figure 42: Chapter 5 - VariationA - Algorithm for every new object o in the surrogate server n_k

5.3.2 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 find the oldest objects, that are less likely to be asked for again, and delete them. In the case that

```
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
```

Figure 43: Chapter 5 - VariationB - Algorithm for every new object o in the surrogate server n_k

Name	Primary Key	Secondary Key
LRU	Time Since Last Access	
LFU	Frequency of Access	
SIZE	Size	Time Since Last Access

Table 21: Applied Caching Schemes

there are no such objects, we delete those with the largest timestamp in the cascade. In all other cases, various policies are applied for the removal of objects:

- *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. This algorithm keeps track of what was used when, to make sure that it discards the least recently used item.

```
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
```

Figure 44: Chapter 5 - VariationC - Algorithm for every new object o in the surrogate server n_k

- *Least Frequently Used (LFU):* In this method the system keeps track of the number of times an object is referenced in memory. When the cache is full and requires more room the system 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 [29] to generalise 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 with higher time since last access are removed first. Objects in the cache are reindexed in order of 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.

Varying algorithms depending on the caching scheme used (Table 21) are depicted in Fig. 42, Fig. 43 and Fig. 44. The heuristics applied in our approach are based on the following observations [110]: 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 ([165], [60]). The percentage of cascades proceeding for days may not be directly attributed to the influence that a social contact exerts, as the video can appear in the newsfeed of the user for a longer time or a new cascade of this specific object may have started. However, in our prefetching algorithm we experiment with varying time thresholds of 24 hours, 48 hours, and threshold covering the entire percentage of requests (threshold for the cascade effect and the time that an object remains in cache).

Prinicipally we check whether specific time has elapsed 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.

5.4 Experimental Evaluation

For the experimental evaluation, we used the CDNsim simulator for CDNs (Chapter 3). The configuration of the simulation values is shown in Table 22. For the extraction of 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 to define the nodes in the topology. These nodes represent the surrogate servers, the origin servers, and the users making the object requests. For an in-depth analysis you can refer to Chapter 3. To simulate our policy and place the servers in a real geographical position, we used the geographical distribution of the Limelight network [54]. For the smooth operation of the simulator the number of surrogate servers was reduced by the ratio of 10%, to ultimately include 423 servers (Table 13). Depending on which surrogate region of

the 20 the Limelight network defines is closer to each of the 142 Twitter timezones, we decided where the requests from this timezone will be redirected. The population of each timezone was also taken into consideration. The INET generator [56] 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 [4], a tool that maps IP addresses and AS coordinates to geographical coordinates [174], and surrogate servers were assigned to topology nodes.

After grouping users per timezone (due to the limitations the large dataset imposes), each team of users was placed in a topology node. We placed the users 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, as CDNsim handles satisfyingly up to so many in general, with the number of objects being the dominant factor increasing the memory use of the tool. Also similar concept approaches use similar number of requests ([176] on a daily basis and [165]), and same number of distinct videos for generation of requests.

- *Cache size:* With the requests generated from the generator following a long-tail distribution, 15 % of the whole catalog size was considered to be sufficient.

- *Threshold values:* Experimenting was conducted for time thresholds of 24 hours and 48 hours, as well as for the time threshold that covered all the requests. The threshold value for media service viewership was moderately chosen as 402408 (average media viewership in the dataset). The authority threshold score was tested for various values (0.006 / 0.02 / 0.04).

5.5 Main Findings

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

Client Side Metrics

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

Number of nodes in the topology	3500
Redirection Policy	Cooperative Environment (closest surrogate)
Number of origin servers	1
Number of surrogate servers	423
User groups	162
Bandwidth	100 Mbit/sec

Table 22: Chapter 5 - Simulation Characteristics

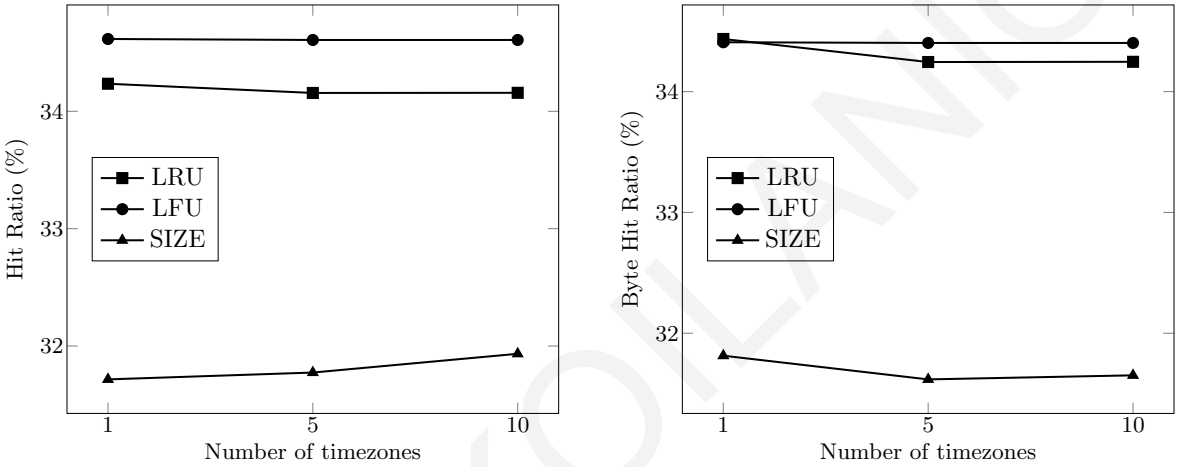


Figure 45: Chapter 5 - Effect of timezones used as Y on (a)Hit Ratio and (b)Byte Hit Ratio for $X = 10$ closest timezones with mutual followers for (i)LRU, (ii)LFU and (iii)SIZE

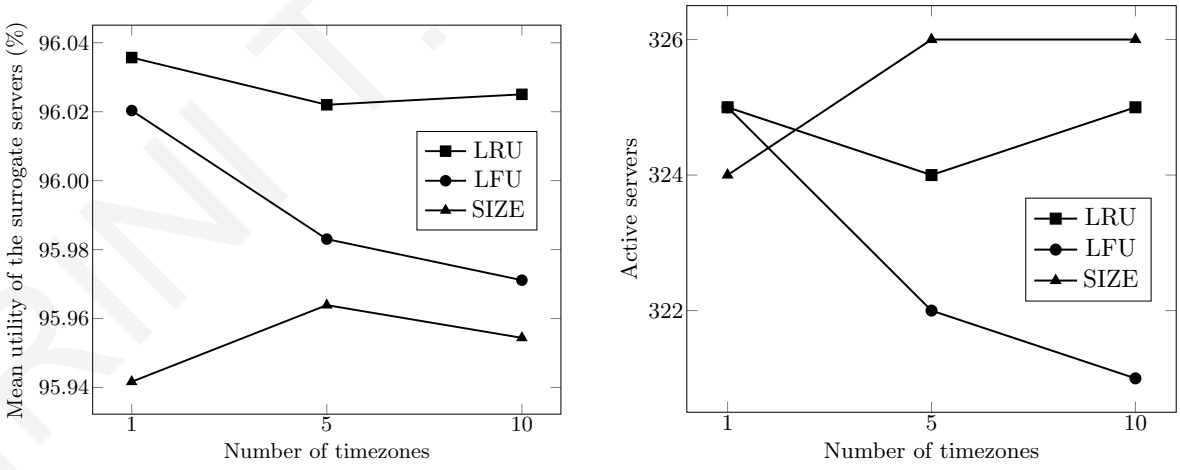


Figure 46: Chapter 5 - Effect of timezones used as Y on (a)Mean utility of the surrogate servers and (b)Active servers for $X = 10$ closest timezones with mutual followers for (i)LRU, (ii)LFU and (iii)SIZE

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

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

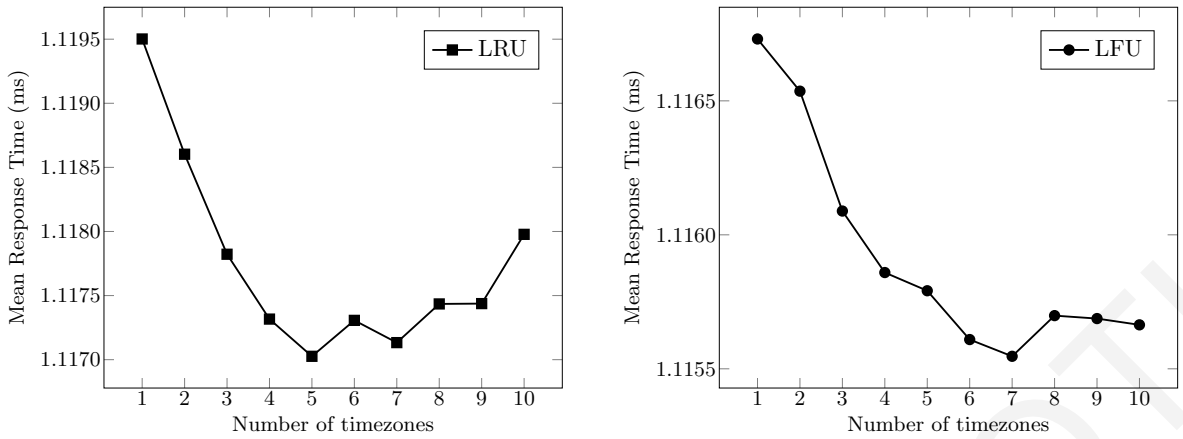


Figure 47: Chapter 5 - Effect of timezones used as Y on Mean Response Time for $X = 10$ closest timezones with mutual followers for (i)LRU and (ii)LFU

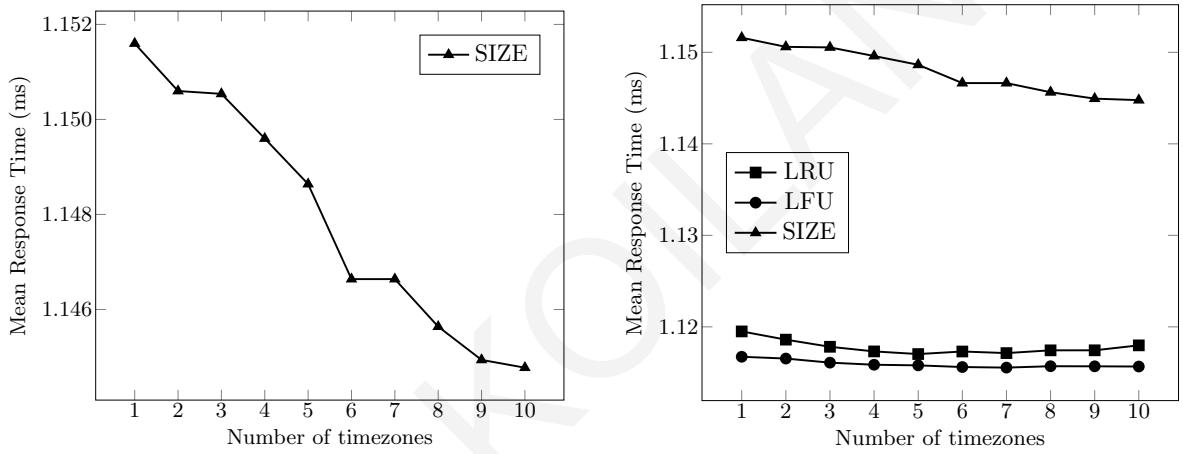


Figure 48: Chapter 5 - Effect of timezones used as Y on Mean Response Time for $X = 10$ closest timezones with mutual followers for (i)SIZE and (ii)all caching schemes

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 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.

Network Statistics Metrics

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

- *Active Surrogate Servers:* refers to the servers being active serving clients.

- *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 mean response times for the clients.

	Mean response time (Avg, 10^-2 sec.)	**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

Table 23: Chapter 5 - Average Metric Values for $X = 10$ Timezones of Close Mutual Friends

We conducted a multitude of experiments (55 for each caching scheme and time threshold combination). Table 23 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. 45 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 outperform SIZE. We also 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. 46).

- *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 1, 5 and 10 timezones examined (Fig. 46).

- *Mean Response Time:* For the most representative case of all requests the trade-off between the reduction of the response time and the cost of copying in servers is expressed for LRU and LFU schemes 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 (Fig. 47 and Fig. 48). For LRU scheme this decrease in the mean response time occurs with approximately 5 timezones out of the 10 used (for a fixed number of closest timezones with mutual followers), and for the LFU with 7 timezones . 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 the server where copying is likely to be made, according to the *Put* function. We observe that SIZE depicts a poor performance compared with the two other schemes, as it is unable to take advantage of the frequency skew.

Performance of caching schemes

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.

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 [86].

A brief categorization [29] 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.

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 [29] 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.

In addition to non-homogeneous object sizes there are several other special features of the web which can be taken into account: Besides the hit ratio, which may not be the best possible measure for evaluating the quality of a web caching algorithm, since

for example the transfer time cost for transfering a large object is more than that for a small object, depending also on the distance of the object from the web server. Moreover, web objects will typically have expiration times. So when considering which objects to replace when a new object enters a web cache we must consider not only the relative frequency but also factors such as object sizes transfer time savings and expiration times. 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, significant for caching of non-uniform sized objects, that will eliminate the disruption caused from the addition of an object in cache and the necessary removal of other objects.

5.6 Conclusions

CDN infrastructures rapidly deliver multimedia content cached on dispersed geographical servers to web browsers worldwide. The growing demands for quick and scalale 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 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 non-synthetic workloads, that many works in the related literature lack. While recognising 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 with traffic generated by online social services and microblogging platforms. We aim in the future to generalize our proposed policy to deal with multiple OSN platforms, as well as mobile CDN providers.

Chapter 6

Predicting Video Virality on Twitter

In this work we merge user-centric data from Twitter with video-centric data from YouTube to investigate the ties between predictability of video sharing and the social context of video uploaders [115]. It provides a combination of social media datasets, giving insights than neither dataset (social network and media service) individually gives. We develop an accurate model to predict future popularity of a video resource given features of the underlying network of its initial sharer. The set of features we propose and analyse are based on the notion of influence score of a user and its fluctuation through time, as well as the distance of content interests among users for both datasets.We discover that the latter feature plays an important role in video popularity prediction, suggesting high dependence of video sharing via Twitter on the video content itself. We proceed to incorporate our prediction model into a mechanism for content delivery resulting in substantial improvement for the user experience.

The remainder of this chapter is organized as follows. Section 6.2 formally describes the addressed problem. Section 6.3 provides an outline of the methodology, followed by the preparation of the employed datasets. Our main findings are presented in Section 6.4, where also a validation is conducted. Section 6.5 investigates the incorporation of the proposed model into a content delivery mechanism. Section 6.6 concludes the work and discusses directions for future work.

6.1 Introduction

The diffusion of video content is fostered by the ease of producing online content via media services. It mainly happens via ubiquitous OSNs, where social cascades can be observed when users increasingly repost links they have received from others. Twitter is one

of the most popular OSNs with its core functionality centered around the idea of spreading information by word-of-mouth [159]. It provides mechanisms such as retweet (forwarding other people's tweets), which enable users to propagate information across multiple hops in the network.

If we knew beforehand when a social cascade will happen or to what range it will evolve, we could exploit this knowledge in various ways. For example, in the area of content delivery infrastructure, with content prefetching by replication of popular items, so that bandwidth could be spared. The knowledge of the evolution of social cascades could also lead to reduction schemes for the storage of whole sequences of large social graphs as well as the reduction of their processing time.

Towards this direction, in this work we present a model for efficiently calculating the number of retweets of a video, associating it with a score depicting the influence of its uploader in the Twitter dataset, the increasing or decreasing trend this score shows, as well as the distance of content interests among users in the YouTube and Twitter community.

Our work [116] focuses on video virality over an OSN. Study of social cascades is active aiming at the prediction of the aggregate popularity of a resource or the individual behaviour of a user. Few works, however, combine detailed information both of the OSN and the media service with a small and easily extracted feature set. Our study proposes a prediction model that performs better than methods like Support Vector Machines (SVM), Stochastic Gradient Descent (SGD) and K-Nearest Neighbours (KNN), among others, and we, furthermore, proceed to incorporate our prediction model into a mechanism for content delivery resulting in substantial improvement for the user experience.

6.2 Problem Description

We consider a directed graph $G(t) = (V(t), E(t))$ representing a social network at time t consisting of V vertices and E edges, that evolves through time. Edges between the nodes of the graph denote friendship in case of a social network (for Twitter B is a follower of A if there is an edge between B and A pointing at A.)

Our problem is stated as follows. We want to associate the number of retransmits of a video link by a user $v \in V$ after $u \in V$ has transmitted the link. User v is a follower of u.

We express this number, intuitively, as a combination of the following features: the $Score(u,t)$ of node u, $dScore(u,t)/dt$ of node u, and content distance between the content interests of the involved users both in the OSN and the media service. The validity of the

$G(t) = (V(t), E(t))$	OSN graph G at time t of V vertices and E edges
A_{u2v}	number of actions where u influenced v
$\widehat{A_{u2v}}$	predicted output
M	total number of predicted values
α, β, γ	coefficients of feature set variables
U	vector of YouTube interests of user u
V	vector of Twitter interests of user v
Features Set	
$Score(u,t)$	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

Table 24: Chapter 6 - Notation Overview

predictors is analysed in this chapter. 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.

With A_{u2v} the output, $\widehat{A_{u2v}}$ the predicted output, and M the total number of predicted values, we want to find the values α, β, γ, so that both equations apply:

$$A_{u2v} = \alpha * Score(u,t) + \beta * \frac{dScore(u,t)}{dt} + \gamma * content\_dist \tag{26}$$

and

$$\sqrt{\frac{1}{M}\sum_{i=1}^{M}(\widehat{A_{u2v}} - A_{u2v})^2} \tag{27}$$

is minimum.

6.3 Proposed Methodology

6.3.1 Dataset

Interests of users were analysed in [28] against directory information from `http://wefol-low.com`, a website listing Twitter users for different topics, including Sports, Movies, News & Politics, Finance, Comedy, Science, Non-profits, Film, Sci-Fi/ Fantasy, Gaming, People, Travel, Autos, Music, Entertainment, Education, Howto, Pets, and Shows.

The activity of Twitter users was quantified, and a variety of features were extracted, such as their number of tweets, the fraction of tweets that are retweets, the fraction of tweets containing URLs, etc. Aggregated features of YouTube videos shared by a user included in the dataset comprise the average view count, the median inter-event time between video upload and sharing, etc.

A sharing event in the dataset is defined as a tweet containing a valid YouTube video ID (with a category, Freebase topics and timestamp). We augmented the provided dataset with Tweet content information about the 15 million video sharing events included in the dataset, as well as information about the followers of the 87K Twitter users.

6.3.2 User Score Calculation

A user score is calculated combining the number n of its followers (reduced by a factor of 1000 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 followers of a user to the number of users he follows), as well as the effect e of a tweet by user (measured by multiplying average number of retweets with number of tweets of the user 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.

$$Score = \log\left(n + \left(\left(\frac{b}{100}\right) * n\right) + e\right) \tag{28}$$

6.3.3 Content Distance

The content distance $content\_dist$ expresses a measure of similarity of user's u YouTube and his follower's v Twitter interests. Content distance is calculated using cosine similarity between vectors of user's u YouTube and user's v Twitter video interests, as follows:

$$content\_dist = 1 - \frac{U \cdot V}{\|U\|\|V\|} \tag{29}$$

6.4 Experimental Evaluation

By combining user ids, followership information, user features and tweet context we build a measure of A_{u2v}, expressing the number of times a user's u tweet is retweeted by his followers v. We aim to associate the independent variables of the features set (X dataframe) with the series depicting A_{u2v} (y).

Dep. Variable	A_{u2v}	R-squared	0.396
Model	OLS	Adj. R-squared	0.396
Method	Least Squares	F-statistic	1.570e+04
Prob (F-statistic)	0.00	Log-Likelihood	-8576.9
No. Observations	71952	AIC	1.716e+04
Df Residuals	71949	BIC	1.719e+04
Df Model	3	Covariance Type	nonrobust

Table 25: Chapter 6 - Regression Results (i)

	coef	std err	t	$P > \|t\|$	95%	Conf.Int.
Score	5.79e-05	3.78e-06	15.300	0.000	5.05e-05	6.53e-05
dScore	4.36e-05	4.51e-06	9.667	0.000	3.48e-05	5.25e-05
con_dist	0.389	0.002	213.060	0.000	0.386	0.393

Table 26: Chapter 6 - Regression Results (ii)

Omnibus	2091.840	Durbin-Watson	1.723
Prob(Omnibus)	0.000	Jarque-Bera (JB)	2323.421
Skew	0.408	Prob(JB)	0.00
Kurtosis	3.333	Cond. No.	746.

Table 27: Chapter 6 - Regression Results (iii)

6.4.1 Selection of Predictors

The regression summary of Table 26 shows that coefficients of all predictors are significant ($P > |t|$ is significantly less than 0.05). Therefore, *Score*, *dScore* and *content_dist* can be considered as good predictors. We note that t here refers to $t - statistic$, denoting the quotient of the coefficient of dependent variable divided by coefficient's standard error. P refers to the $P - value$, a standard statistical method for testing an hypothesis. $P - value < 0.05$ means we can reject the hypothesis that the coefficient of a predictor is zero, in other words the examined coefficient is significant.

The selection of the above predictors comes as a result of comparing the $P - values$ of various metrics in the dataset and the combination of those with the lowest $P - value$. The metrics included the number of distinct users retweeted, fraction of the users tweets

that were retweeted, average number of friends of friends, average number of followers of friends, number of YouTube videos shared, the time the account was created, the number of views of a video, etc., among many others.

6.4.2 Effect of Outliers

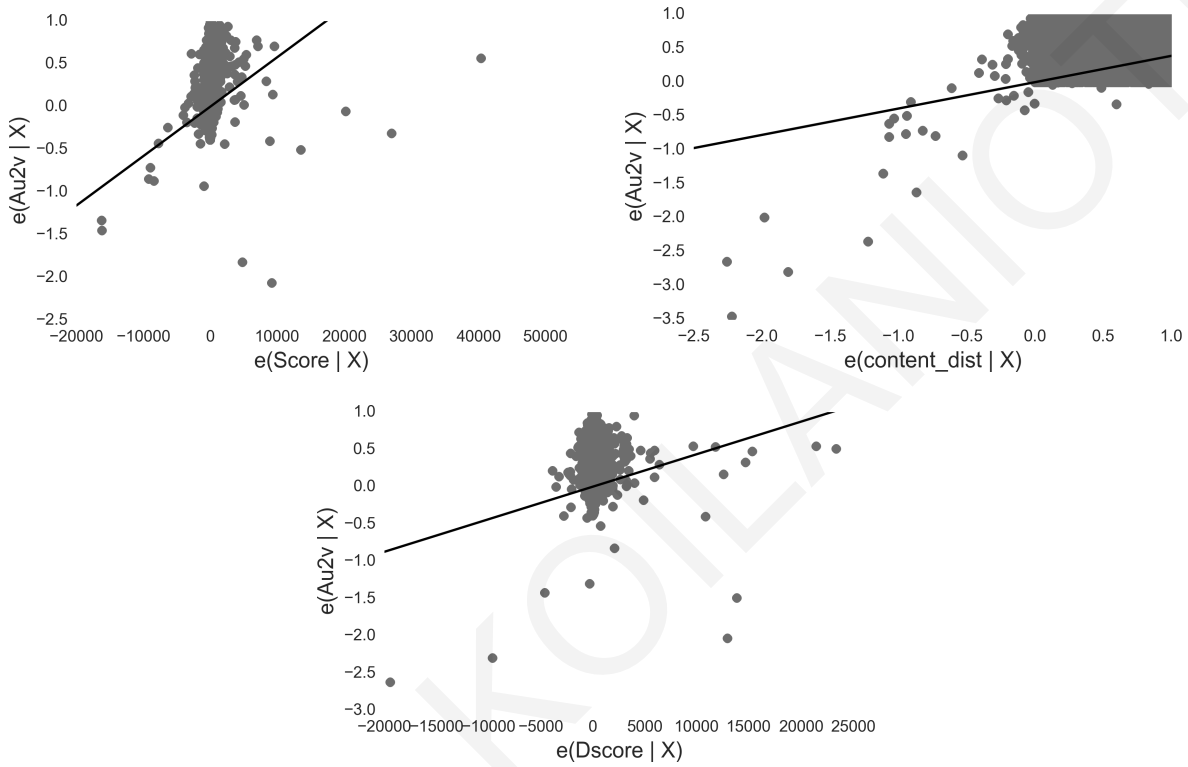


Figure 49: Chapter 6 - Regression plots for each independent variable.

The regression plots for each predictor in Fig. 49 show the effect of outliers on the estimated regression coefficient. Regression line is pulled out of its optimal tracjectory due to the existent outliers. The detailed regression plots for individual predictors (*Score*, *dScore* and *content_dist*) appear in Figures 50, 51, and 52 respectively. As for the fitted (predicted) values of A_{u2v} and the prediction confidence for each independent variable, they appear in Figures 53, 54, and 55. We observe that fitted values are quite close to the real values of A_{u2v} with the exception of the outliers. This suggests that removal of outliers would yield a better estimate, since it is obvious that the plot is skewed due to their presence.

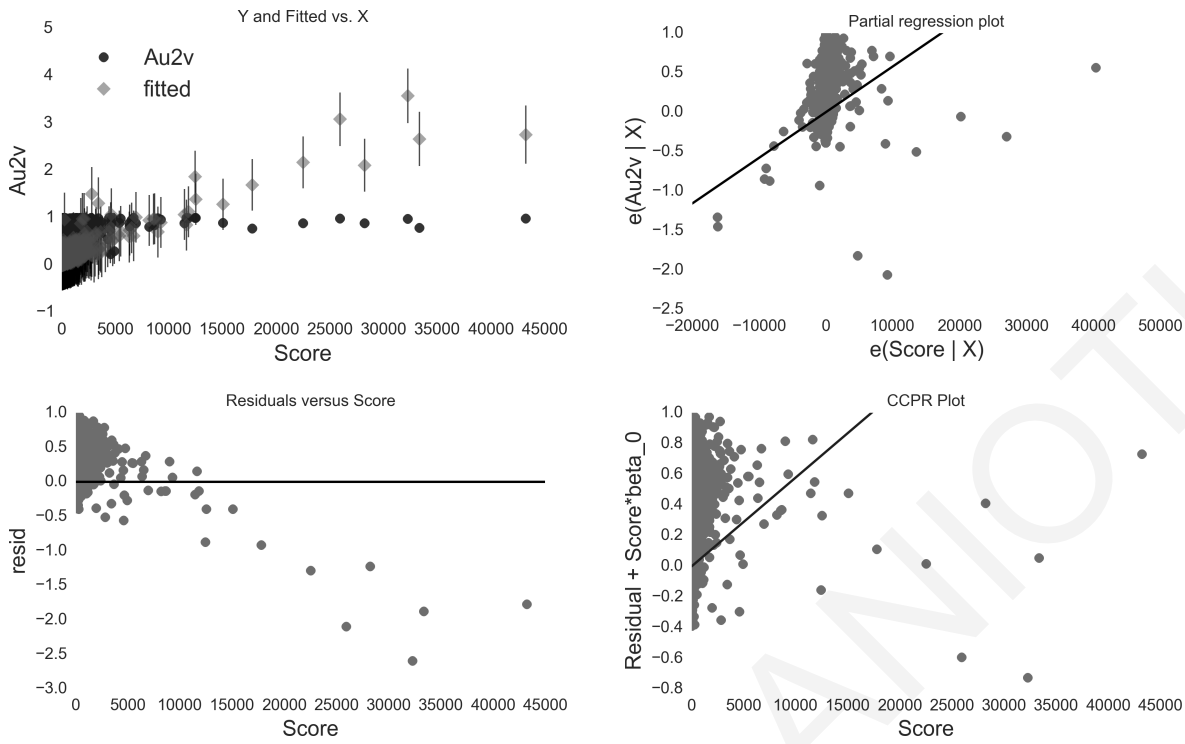


Figure 50: Chapter 6 - Regression plots for *Score*.

A rough estimate of detecting outliers can be based on the quantile distributions of each independent variable in Table 28. Observing Table 28 with an overview of data ditsribution we surmise that we could take values of *Score* and *dScore* only upto 10 and 5 respectively. The quantiles appearing on the table are calculated when data is rearranged in ascending order and divided into 4 equal sized parts. Thus, interpreting the second quantile we notice that 50% of Score values are less than 2.562232. In the table, we notice that for Score and dScore we have huge maximum values but 75% of the data are below 6.902750 and 2.308805 respectively. Thus, we select 10 and 5 as values to take most of the data and exclude data points with extremely large (outliers) out of general range values.

Results of regression model on data obtained after removing outlier data points appear in Tables 29, 30, and 31. The results show considerable improvement with respect to regression with presence of outliers (Tables 25, 26 and 27). Also, Durbin-Watson statistic close to 2 confirms normality assumption of residuals, verifying the normality of error distribution, one of the assumptions of linear regression.

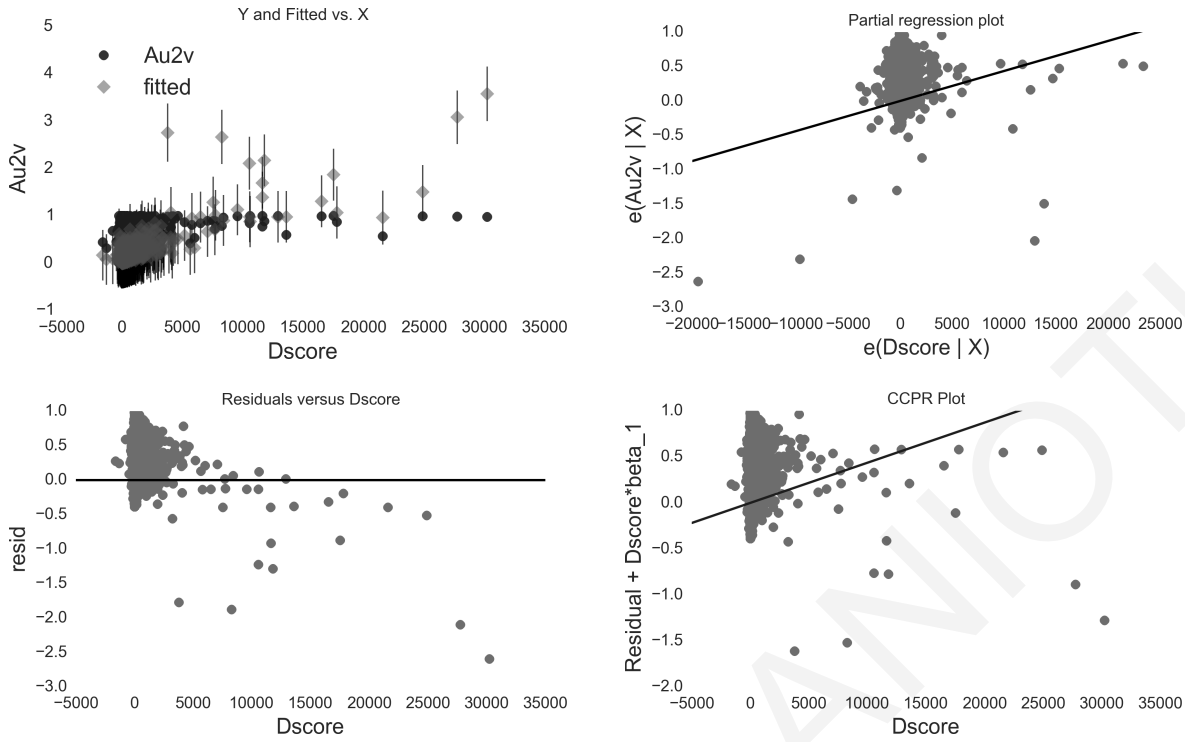


Figure 51: Chapter 6 - Regression plots for *dScore*.

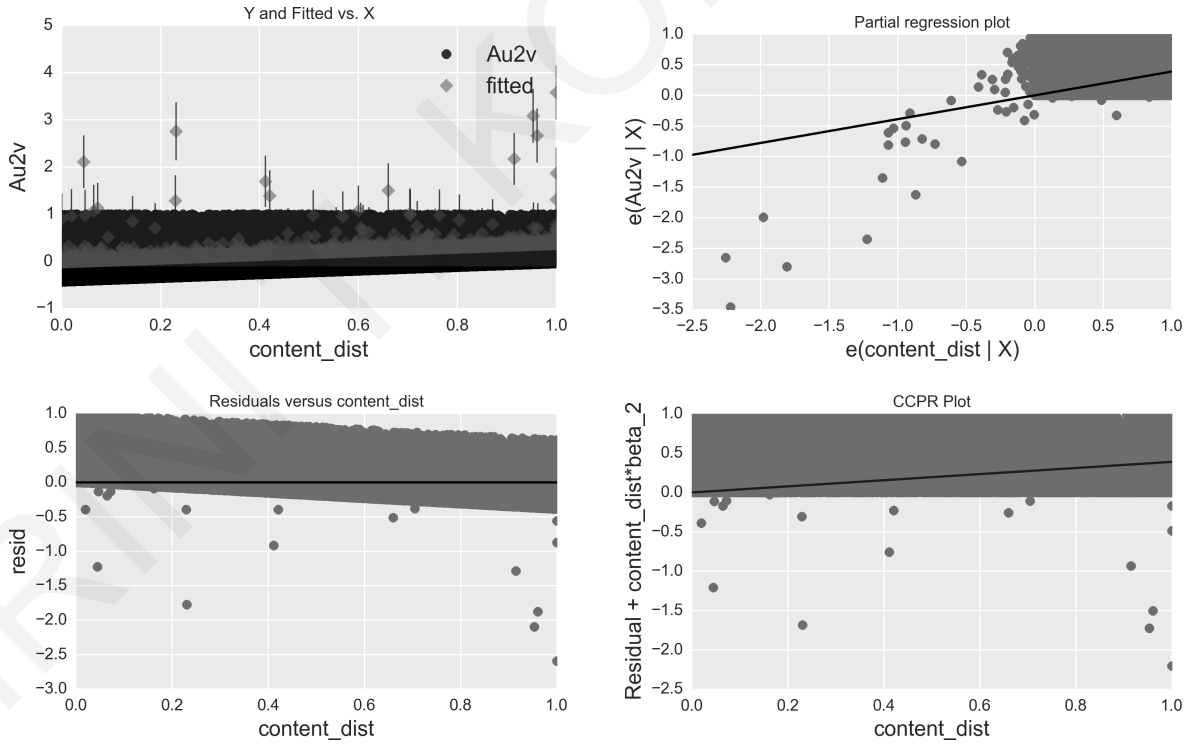


Figure 52: Chapter 6 - Regression plots for *content_dist*.

Fig. 58 plots reinforce the argument that after removing outliers we get a better fit of regression line on each independent variable. Namely, the removal of outliers leads to better alignment of the path of regression line to the optimal path.

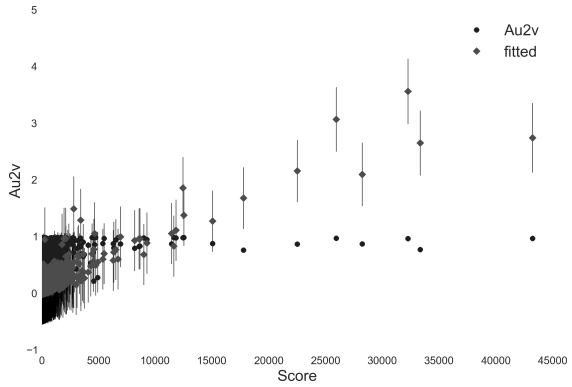


Figure 53: Chapter 6 - Fitted values of A_{u2v} versus *Score*.

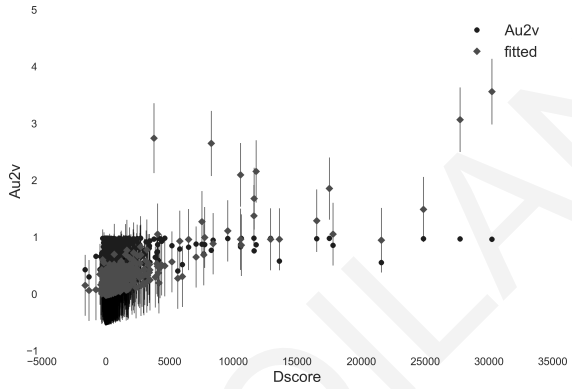


Figure 54: Chapter 6 - Fitted values of A_{u2v} versus *dScore*.

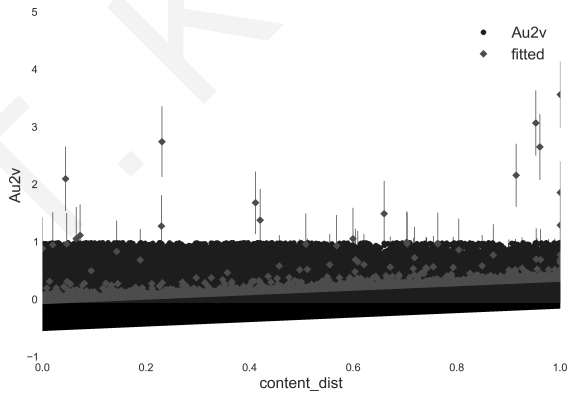


Figure 55: Chapter 6 - Fitted values of A_{u2v} versus *content_dist*.

6.4.3 10-fold Cross-Validation

We performed a 10-fold cross validation on the dataset, fitting the regressor to 90% of the data and validating it on the rest 10% for the prediction of A_{u2v} dependent variable from Score, dScore and content-dist independent variables. Predictive modeling was conducted after removing outliers from the data. The results of the predictive modeling using linear regression show that we achieve a root mean squared error of 0.1873 (across all folds), which means that our prediction varies by 0.1873 amount from the real values of A_{u2v} . This shows

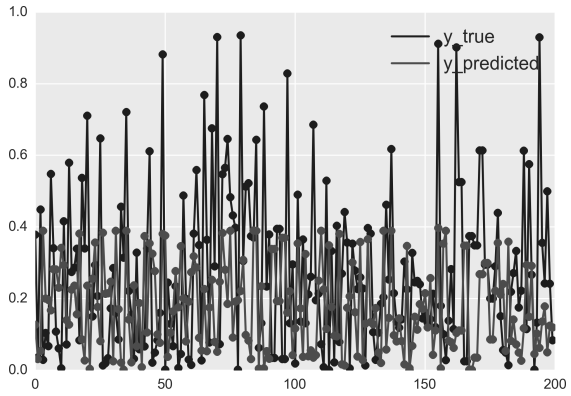


Figure 56: Chapter 6 - 10-fold Cross-Validation of A_{u2v}.

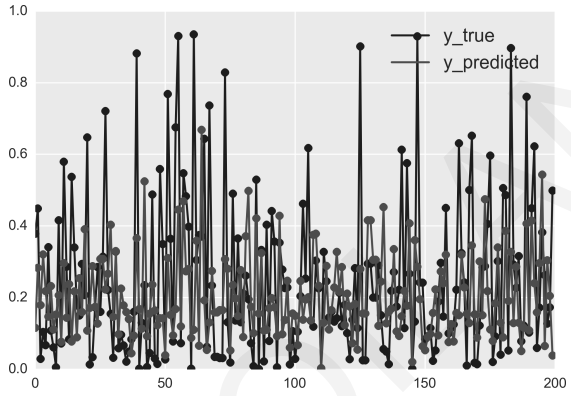


Figure 57: Chapter 6 - 10-fold Cross-Validation of A_{u2v} without outliers.

	Score	*dScore*	*content_dist*
count	71952.000000	71952.000000	71952.000000
mean	21.227703	15.880803	0.459111
std	349.102908	292.727717	0.315837
min	0.000000	-1610.253490	0.000000
25%	0.787060	0.000140	0.172534
50%	2.562232	0.526050	0.415394
75%	6.902750	2.308805	0.724986
max	43262.678131	30235.027960	1.000000

Table 28: Chapter 6 - Outliers thresholds

a considerable improvement in prediction error than modeling with original data (with outliers), where a root mean squared error of 0.2728 (across all folds) was achieved. Plots in both cases appearing in Figure 56 and 57 depict how close our predictions are to the real values of the dependent variable.

Dep. Variable	A_{u2v}	R-squared	0.629
Model	OLS	Adj. R-squared	0.629
Method	Least Squares	F-statistic	3.072e+04
Prob (F-statistic)	0.00	Log-Likelihood	13947.
No. Observations	54473	AIC	-2.789e+04
Df Residuals	54470	BIC	-2.786e+04
Df Model	3	Covariance Type	nonrobust

Table 29: Chapter 6 - Regression Results without Outliers (i)

	coef	std err	t	$P > \|t\|$	95%	Conf.Int.
Score	0.1460	0.001	145.244	0.000	0.144	0.148
dScore	0.0200	0.001	25.819	0.000	0.018	0.022
con_dist	0.1656	0.003	65.690	0.000	0.161	0.171

Table 30: Chapter 6 - Regression Results without Outliers (ii)

Omnibus	10848.216	Durbin-Watson	1.966
Prob(Omnibus)	0.000	Jarque-Bera (JB)	22428.486
Skew	1.183	Prob(JB)	0.00
Kurtosis	5.070	Cond. No.	5.19

Table 31: Chapter 6 - Regression Results without Outliers (iii)

6.4.4 Classification and Comparison with other Models

We predict a user popularity as follows. 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. This can also be interpreted as "if a user u's tweets will be popular or not, given user u's *Score*, *dScore*, and *content_dist* (measure of his YouTube's interest similarity with the Twitter interests of his followers)".

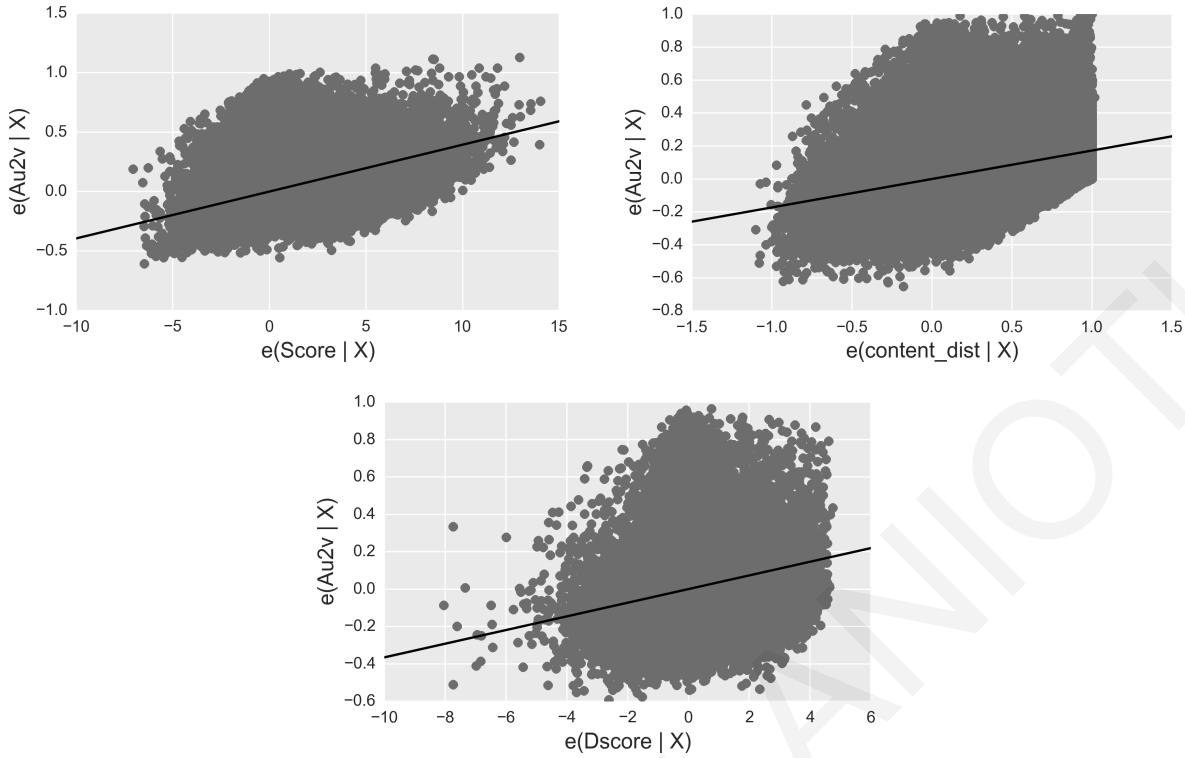


Figure 58: Chapter 6 - Regression plots for each independent variable.

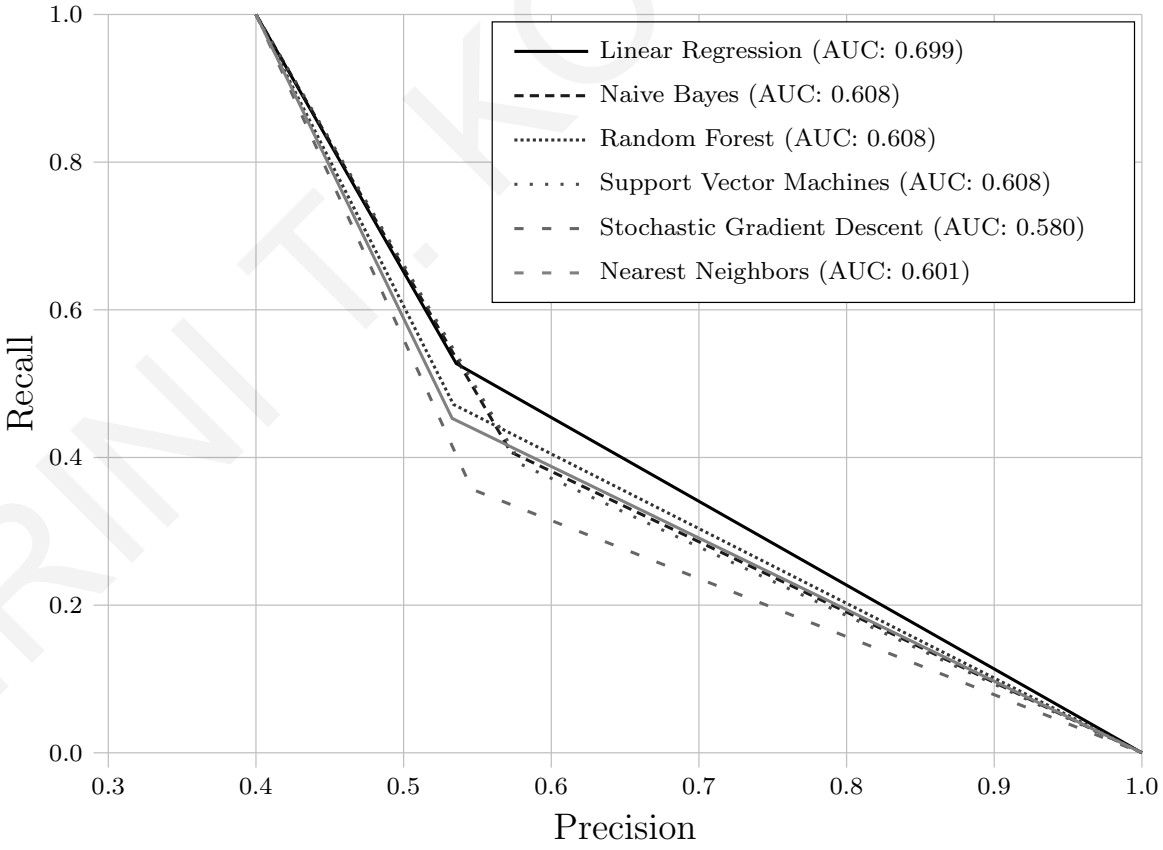


Figure 59: Chapter 6 - Comparison with other models.

Classification was conducted initially with three different methods: Linear Regression (the Predictive Model we present in this study), Random Forest and Naive Bayes methods. Area Under the Curve (AUC) is a score that computes average precision (AP) from prediction scores. This average precision score corresponds to the area under the precision-recall curve and the higher AUC represents better performance. Plots in Figure 59 correspond to computed precision-recall pairs for different probability thresholds and the AUC score computes the area under these curves. Best performance is achieved by Linear Regression (0.699), followed by Naive Bayes (AUC:0.608) and Random Forest (AUC:0.608). Complementary methods tested were Support Vector Machines (SVM), Stochastic Gradient Descent (SGD) and K-Nearest Neighbours (KNN).

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 for belonging 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.

Next classification model was Stochastic Gradient Descent (SGD), a gradient descent optimization method for minimizing an objective function written as a sum of differentiable functions, and 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.

The last classifier implemented here was K-Nearest Neighbours (KNN), 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.

After plotting the results of computed precision-recall pairs for various probability thresholds we observe that best performance is still noticed in the case of our Predictive Model, followed by SVM (AUC:0.608), KNN (AUC:0.601), and, lastly, SGD (AUC:0.580).

6.5 Incorporation into Content Delivery Schemes

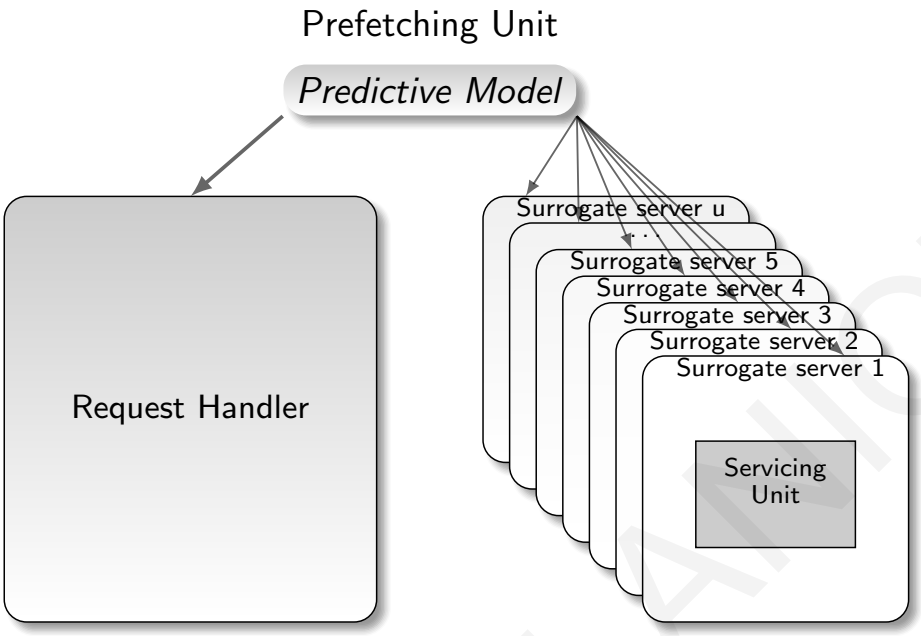


Figure 60: Chapter 6 - The OSN-aware CDN mechanism.

CDNs aim at improving download of large data volumes with high availability and performance. Content generated by online media services circulates and is consumed over OSNs (with more than 400 tweets per minute including a YouTube video link [52] being published per minute). This content largely contributes to internet traffic growth [56]. Consequently, CDN users can benefit from an incorporated mechanism of OSN-awareness over the CDN infrastructure. In [111] Kilanioti and Papadopoulos introduce various efficient copying policies based on prediction of demand on OSNs for the dynamic mechanism of preactive content copying.

Rather than pushing data to all surrogates, they proactively distribute it only to social connections of the user likely to consume it. The content is copied only under certain conditions (content with high viewership within the media service, copied to geographically close timezones of the geo-diversed system used where the user has mutual social connections of high influence impact). This contributes to smaller response times for the content to be consumed (for the users) and lower bandwidth costs (for the OSN provider). Herein, we incorporate the proposed Predictive Model in the suggested policy [111] and prove that it further improves its performance.

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. 60).

- *For Every New Request in the CDN*

  Prinicipally we check whether specific time has elapsed 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. We introduce the *time_threshold*, that roughly expresses the average cascade duration. Initially, we check whether it is the first appearance of the object (Fig. 61). The variable *o.timestamp* depicts the timestamp of the last appearance of the object in a request. 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 its Score.

  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 timezone and to the surrogate servers serving the timezones of all followers of the user. Otherwise, selective copying includes only the surrogates that the subpolicy decides. Subpolicy (Fig. 62) checks the X closest timezones where a user has mutual friends and out of them, the Y 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 timezones of highest combined feature set value, according to the coefficients derived from our analysis. We note here that variations of the Subpolicy include the replacement of the timezones 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.

- *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, *time_threshold* is the parameter for the duration that an object remains cached. We find the oldest objects and delete them. In the case that there are no such objects, we delete those with the largest timestamp in the cascade. We use LRU for handling non-homogeneous sized objects and prune the least recently used items first.To ensure that least recently used items are discarded, the algorithm keeps track of their usage.

$G(V,E)$	Graph representing the social network
$V = \{V_1, ..., V_n\}$	Nodes representing the social network users
$E = \{E_{11}, ..., E_{1n}, ..., E_{nn}\}$	Edges representing the social network connections, where E_{ij} stands for friendship between i and j
$R = \{r_1, r_2, ..., r_\tau\}$	Regions set
$N = \{n_1, n_2, ..., n_u\}$	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
$O = \{o_1, o_2, ..., o_w\}$	Objects set (videos)
$S_i, i \in O$	Size of object i in bytes
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
$P = \{p_{12}, p_{13}, ..., p_{nw}\}$	User posts in the social network, where p_{ij} denotes that node i has shared object j in the social network
$Q = \{q_1, q_2, ..., q_\zeta\}$	Object requests, where q_i denotes a request for an object of set O
Q_{hit_i}, $Q_{total_i}, i \in N$	Number of requests served from surrogate servers of the region of the user/ total number of requests
$X, Y, L, H \in R$	Closest timezones with mutual followers/ with highest centrality metric/ with highest lobby values/ with highest HITS values

Table 32: Chapter 6 - Notation Overview

The nodes representing the surrogate servers, the origin servers, and the users making the object requests (Fig. 64) in the simulated network topology are analysed in detail in Chapter 3. To simulate our policy and place the servers in a real geographical position, we used the geographical distribution of the Limelight network. For the smooth operation of the simulator the number of surrogate servers was reduced by the ratio of 10%, to ultimately include 423 servers. Depending on which surrogate region of the 20 the Limelight network defines

```
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.Score > Score_threshold then
12:    copy object o to surrogate that serves user's V_i(t) timezone;
13:    for all user V_y(t) that follows user V_i(t) do
14:        find surrogate server n_j that serves V_y(t)'s timezone;
15:        copy object o to n_j;
16:    end for
17: else if o.timer < time_threshold then
18:    copy object o to surrogates n_j that Subpolicy decides;
19: end if
```

Figure 61: Chapter 6 - Algorithm for every new request ($timestamp$, $V_i(t)$, o) in the CDN

```
1: find X timezones where (user V_i(t) has mutual followers and they are closer to user's
   V_i(t) timezone);
2: find the Y ⊆ X that (belong to X and depict the highest average values of Predictive
   Model(Score, dScore, content_dist));
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
```

Figure 62: Chapter 6 - Subpolicy

is closer to each of the 142 Twitter timezones, we decided where the requests from this timezone will be redirected. The population of each timezone was also taken into consideration. The INET generator [56] allowed us to create an AS-level representation of the network

```
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
```

Figure 63: Chapter 6 - Algorithm for every new object o in the surrogate server n_k

topology. Topology coordinates were converted to geographical coordinates with the NetGeo tool from CAIDA, a tool that maps IP addresses and AS coordinates to geographical coordinates, and surrogate servers were assigned to topology nodes. After grouping users per timezone (due to the limitations the large dataset imposes), each team of users was placed in a topology node. We placed the users in the nodes closer to those comprising the servers that serve the respective timezone requests, contributing this way to a realistic network depiction.

The heuristics applied in [111] are based on the observation that users are more influenced by geographically close friends, and moreover by mutual followers, as well as on the short duration of social cascades ([165], [60]).

We examine Mean Response Times (MRT), a client-side metric associated with user experience in CDNs that indicates how fast a client is satisfied, for the most representative case of time thresholds covering all the examined requests of our dataset. The trade-off between the reduction of the response time and the cost of copying in servers is expressed for all schemes used (Linear Regression, Naive Bayes, Random Forest, SVM, SGD, KNN) with a decrease of the mean response time as the timezones increase and a point after which

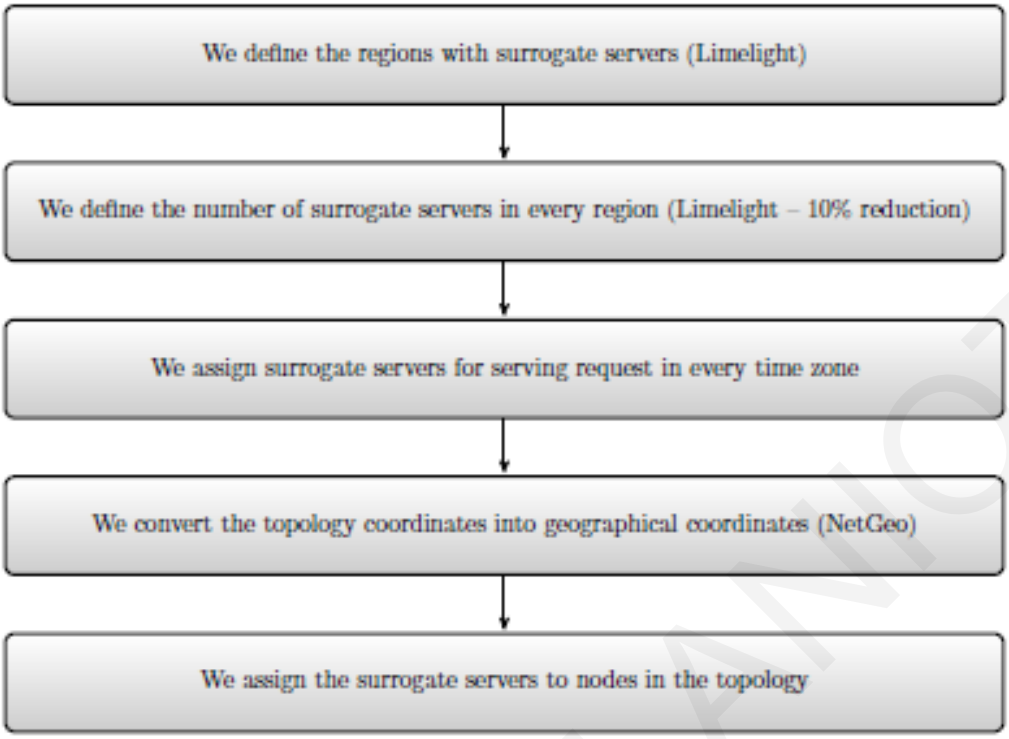


Figure 64: Chapter 6 - Methodology followed

the MRT starts to increase again (Fig. 65). For the scheme augmented with our Predictive Model, namely the Linear Regression, this shift occurs with approximately 6 timezones out of the 10 used (for a fixed number of closest timezones with mutual followers). 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 be made. We observe that Linear Regression outperforms all the other schemes, depicting MRTs smaller than their respective. We note here that timezones with highest average values for each scheme, that Subpolicy defines, are pre-calculated, in order to reduce computational burden in the simulations.

6.6 Conclusions

We come to the conclusion that video sharings over an OSN platform can be predicted with a small set of features combined from both the platform and the media service. Despite the focused scope of this work and the limitations of its conduction solely with Twitter and YouTube data, the scale of the medium allows us to make assumptions for generalisation across different OSNs and microblog platforms. We plan to extensively analyse this generalisation in the future. Future extensions also include experimentation with variations of

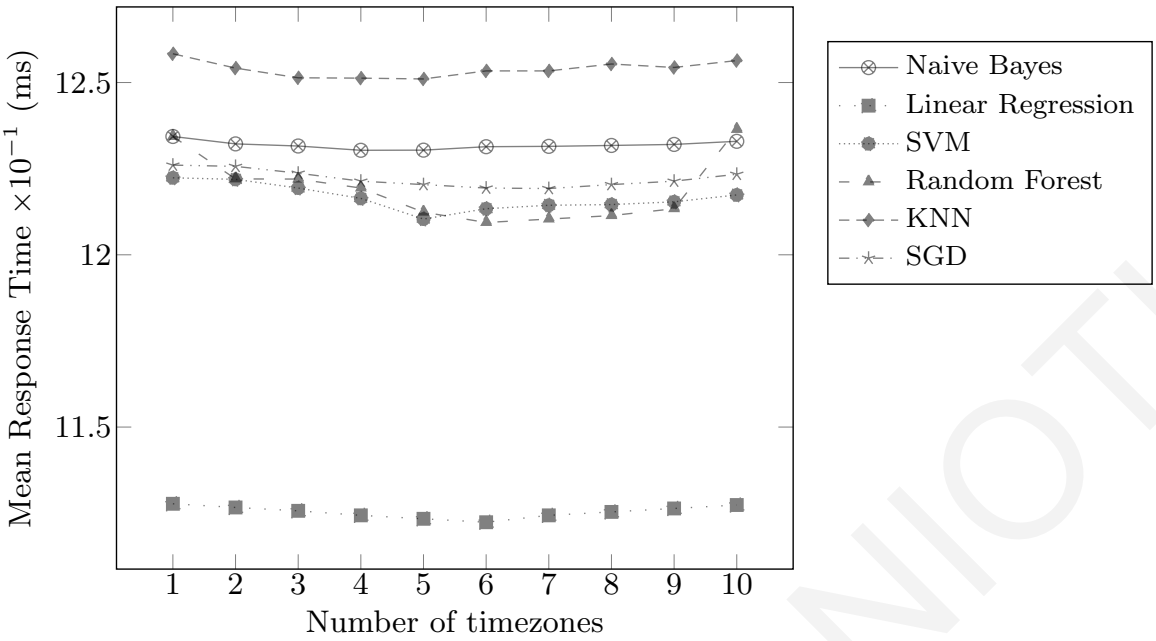


Figure 65: Chapter 6 - Effect of timezones used as Y on Mean Response Time for various schemes ($X = 10$ closest timezones with mutual followers).

content distance interpretation among users, with various score assignment formulas, as well as subsequent verification in the realm of content delivery. We hope that our findings will broaden the view on the spread of information in web today.

Chapter 7

Efficient Content Delivery via Forecasting Popularity on Social Media

Ubiquitous social networks have in recent years become significant for sharing of content generated in online video platforms. Our work investigates how the predictability of video sharing is associated with the underlying social network of the initial sharer of the video and the context of the media platform it was uploaded. In particular we combine user-centric data from Twitter with video-centric data from YouTube to give insights that neither dataset (social network and media service dataset) individually gives. We propose a simple model to predict future popularity of a video resource with a small and easily extracted feature set, based on the notion of influence score of a user and its fluctuation through time, as well as the distance of content interests among users for both datasets. We further demonstrate how the incorporation of our prediction model into a mechanism for content delivery results in considerable improvement of the user experience [113].

The remainder of this chapter is organized as follows. Section 7.2 formally describes the addressed problem. Section 7.3 provides an outline of the methodology, followed by the preparation of the employed datasets. Our main findings are presented in Section 7.4, where also a validation is conducted. Section 7.5 investigates the incorporation of the proposed model into a content delivery mechanism. Section 7.6 concludes the work and discusses directions for future work.

7.1 Introduction

The diffusion of video content is fostered by the ease of producing online content via media services. It mainly happens via ubiquitous OSNs, where users increasingly repost

links they have received from others (social cascades). If we knew beforehand when a social cascade will happen or to what range it will evolve, we could exploit this knowledge in various ways. For example, in the area of content delivery infrastructure, where popular items would be proactively replicated, so that bandwidth could be spared. Our work focuses on video virality over an OSN and combines detailed information both of the OSN and the media service with a small and easily extracted feature set. It proposes a prediction model that performs better than methods like Support Vector Machines (SVM), Stochastic Gradient Descent (SGD) and K-Nearest Neighbours (KNN), among others. We, furthermore, proceed to incorporate it into a mechanism for content delivery depicting substantial improvement for the user experience.

7.2 Problem Description

$G(t) = (V(t), E(t))$	OSN graph G at time t of V vertices and E edges
A_{u2v}	number of actions where u influenced v
$\widehat{A_{u2v}}$	predicted output
M	total number of predicted values
α, β, γ	coefficients of feature set variables
U	vector of YouTube interests of user u
V	vector of Twitter interests of user v
Features Set	
$Score(u,t)$	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

Table 33: Chapter 7 - Notation Overview

We consider a directed graph $G(t) = (V(t), E(t))$ representing a social network at time t consisting of V vertices and E edges, that evolves through time. Edges between the nodes of the graph denote friendship in case of a social network (for Twitter B is a follower of A if there is an edge between B and A pointing at A.) We want to associate the number of retransmits of a video link by a user $v \in V$ after $u \in V$ has transmitted the link. User v is a follower of u.

We express this number, intuitively, as a combination of the following features: the $Score(u,t)$ of node u, $dScore(u,t)/dt$ of node u, and content distance between the content

interests of the involved users both in the OSN and the media service. The validity of the predictors is analysed in this work. 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.

With A_{u2v} the output, $\widehat{A_{u2v}}$ the predicted output, and M the total number of predicted values, we want to find the values α, β, γ, so that both equations apply:

$$A_{u2v} = \alpha * Score(u,t) + \beta * \frac{dScore(u,t)}{dt} + \gamma * content\_dist \tag{30}$$

and

$$\sqrt{\frac{1}{M}\sum_{i=1}^{M}(\widehat{A_{u2v}} - A_{u2v})^2} \tag{31}$$

is minimum.

7.3 Proposed Methodology

7.3.1 Dataset

Interests of users were analysed in [28] against directory information from `http://wefol-low.com`, a website listing Twitter users for different topics, including Sports, Movies, News & Politics, Finance, Comedy, Science, Non-profits, Film, Sci-Fi/ Fantasy, Gaming, People, Travel, Autos, Music, Entertainment, Education, Howto, Pets, and Shows.

Twitter is one of the most popular OSNs centered around the idea of spreading information by word-of-mouth [159]. It provides mechanisms such as retweet, which enable users to propagate information across multiple hops in the network.

The activity of Twitter users was quantified, and a variety of features were extracted, such as their number of tweets, the fraction of tweets that are retweets, the fraction of tweets containing URLs, etc. Aggregated features of YouTube videos shared by a user included in the dataset comprise the average view count, the median inter-event time between video upload and sharing, etc.

A sharing event in the dataset is defined as a tweet containing a valid YouTube video ID (with a category, Freebase topics and timestamp). We augmented the provided dataset with Tweet content information about the 15 million video sharing events included in the dataset, as well as information about the followers of the 87K Twitter users.

7.3.2 User Score Calculation

A user score is calculated combining the number n of its followers (reduced by a factor of 1000 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 followers of a user to the number of users he follows), as well as the effect e of a tweet by user (measured by multiplying average number of retweets with number of tweets of the user 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.

$$Score = \log\left(n + \left(\left(\frac{b}{100}\right) * n\right) + e\right) \tag{32}$$

7.3.3 Content Distance

The content distance $content\_dist$ expresses a measure of similarity of user's u YouTube and his follower's v Twitter interests. Content distance is calculated using cosine similarity between vectors of user's u YouTube and user's v Twitter video interests, as follows:

$$content\_dist = 1 - \frac{U \cdot V}{\|U\|\|V\|} \tag{33}$$

7.4 Experimental Evaluation

By combining user ids, followership information, user features and tweet context we build a measure of A_{u2v}, expressing the number of times a user's u tweet is retweeted by his followers v. We aim to associate the independent variables of the features set with the series depicting A_{u2v}.

The regression summary of Table 35 shows that coefficients of all predictors are significant ($P > |t|$ is significantly less than 0.05). Therefore, *Score*, *dScore* and $content\_dist$ can be considered as good predictors. We note that t here refers to $t - statistic$, denoting the quotient of the coefficient of dependent variable divided by coefficient's standard error. P refers to the $P - value$, a standard statistical method for testing an hypothesis. $P - value < 0.05$ means we can reject the hypothesis that the coefficient of a predictor is zero, in other words the examined coefficient is significant.

The selection of the above predictors comes as a result of comparing the $P - values$ of various metrics in the dataset and the combination of those with the lowest $P - value$.

The metrics included the number of distinct users retweeted, fraction of the users tweets that were retweeted, average number of friends of friends, average number of followers of friends, number of YouTube videos shared, the time the account was created, the number of views of a video, etc., among many others.

Dep. Variable	A_{u2v}	R-squared	0.629
Model	OLS	Adj. R-squared	0.629
Method	Least Squares	F-statistic	3.072e+04
Prob (F-statistic)	0.00	Log-Likelihood	13947.
No. Observations	54473	AIC	-2.789e+04
Df Residuals	54470	BIC	-2.786e+04
Df Model	3	Covariance Type	nonrobust

Table 34: Chapter 7 - Regression Results without Outliers (i)

Score	0.1460	0.001	145.244	0.000	0.144	0.148
dScore	0.0200	0.001	25.819	0.000	0.018	0.022
con_dist	0.1656	0.003	65.690	0.000	0.161	0.171

Table 35: Chapter 7 - Regression Results without Outliers (ii)

Results of regression model on data obtained after removing outlier data points appear in Tables 34 and 35. We discover that *content_dist* feature plays an important role in video popularity prediction, suggesting high dependence of video sharing via Twitter on the video content itself. Fig. 66 plots depict an improved alignment of the path of regression line to the optimal path after the removal of outliers.

7.4.1 10-fold Cross-Validation

We performed a 10-fold cross validation on the dataset, fitting the regressor to 90% of the data and validating it on the rest 10% for the prediction of A_{u2v} dependent variable from *Score*, *dScore* and *content_dist* independent variables. Predictive modeling was conducted after removing outliers from the data. The results of the predictive modeling using linear regression show that we achieve a root mean squared error of 0.1873 (across all folds), which means that our prediction varies by 0.1873 amount from the real values of A_{u2v}. Plot in Fig. 67 depicts how close our predictions are to the real values of the dependent variable.

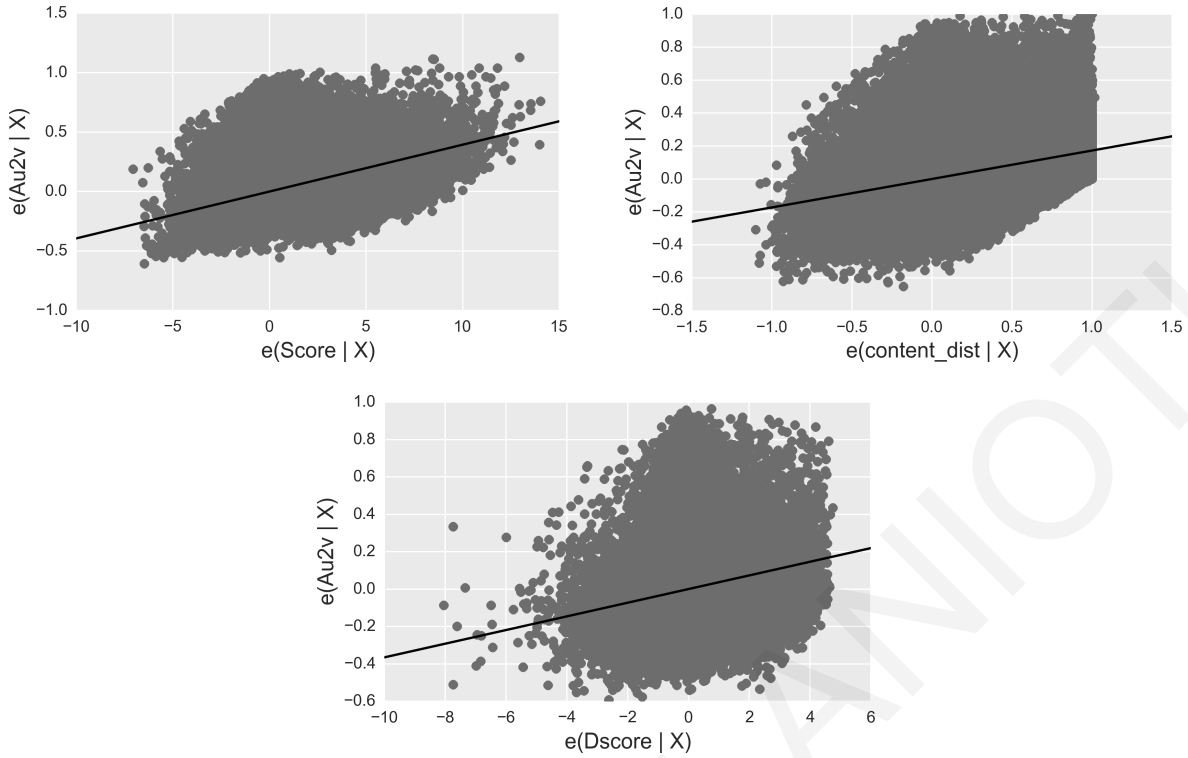


Figure 66: Chapter 7 - Regression plots for each independent variable.

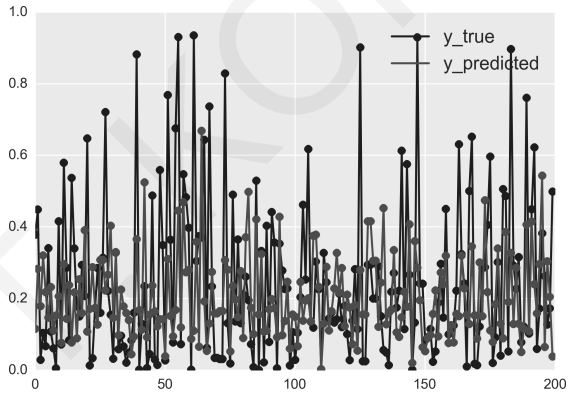


Figure 67: Chapter 7 - 10-fold Cross-Validation of A_{u2v} without outliers.

7.4.2 Classification and Comparison with other Models

We predict a user popularity as follows. 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. This can also be interpreted as "if a user u's tweets will be popular or not, given user u's *Score*, *dScore*, and *content_dist* (measure of his YouTube's interest similarity with the Twitter interests of his followers)".

Classification was conducted initially with three different methods: Linear Regression (the Predictive Model we present in this study), Random Forest and Naive Bayes methods.

Area Under the Curve (AUC) is a score that computes average precision (AP) from prediction scores. This average precision score corresponds to the area under the precision-recall curve and the higher AUC represents better performance. Plots in Figure 68 correspond to computed precision-recall pairs for different probability thresholds and the AUC score computes the area under these curves. Best performance is achieved by Linear Regression (0.699), followed by Naive Bayes (AUC:0.608) and Random Forest (AUC:0.608). Complementary methods tested were Support Vector Machines (SVM), Stochastic Gradient Descent (SGD) and K-Nearest Neighbours (KNN).

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 for belonging 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.

Next classification model was Stochastic Gradient Descent (SGD), a gradient descent optimization method for minimizing an objective function written as a sum of differentiable functions, and 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.

The last classifier implemented here was K-Nearest Neighbours (KNN), 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.

After plotting the results of computed precision-recall pairs for various probability thresholds we observe that best performance is still noticed in the case of our Predictive Model, followed by SVM (AUC:0.608), KNN (AUC:0.601), and, lastly, SGD (AUC:0.580).

7.5 Incorporation into Content Delivery Schemes

CDNs aim at improving download of large data volumes with high availability and performance. Content generated by online media services circulates and is consumed over

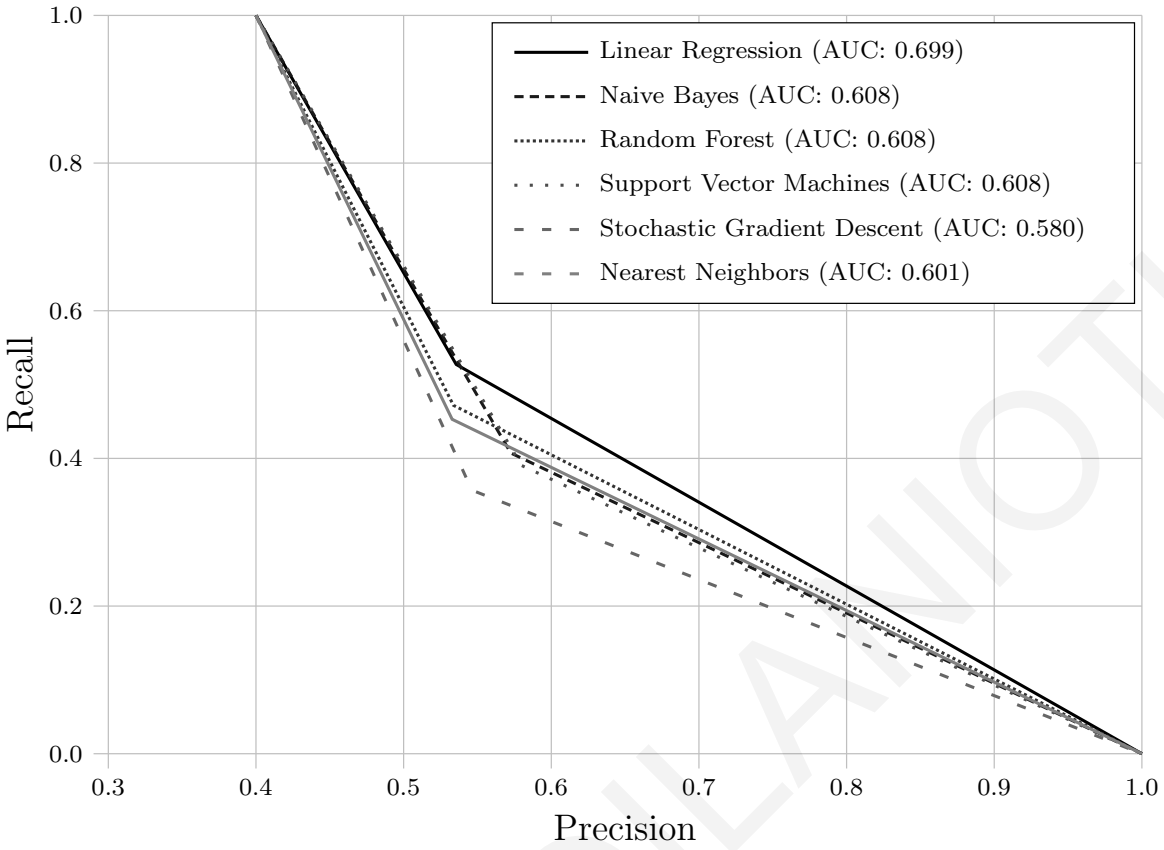


Figure 68: Chapter 7 - Comparison with other models.

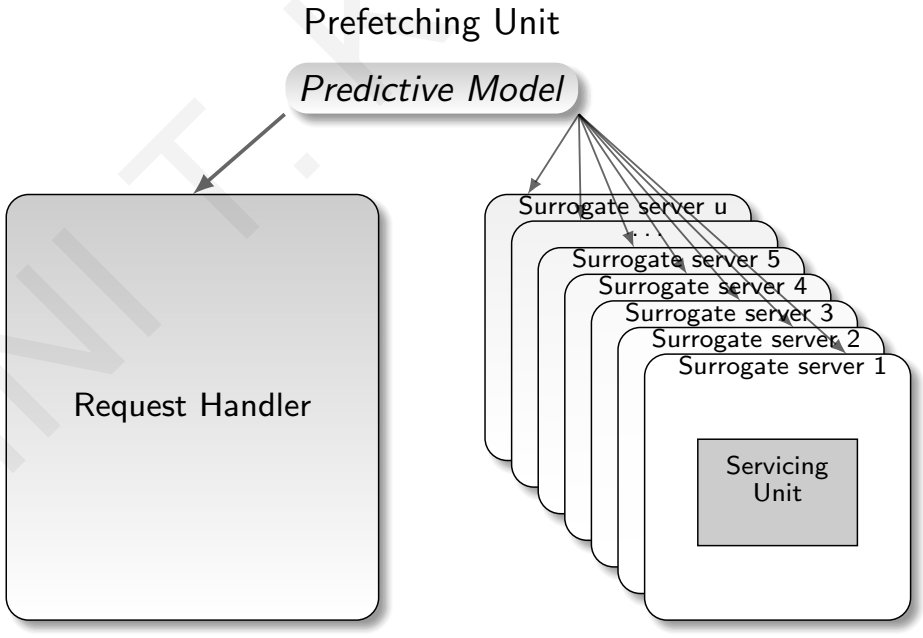


Figure 69: Chapter 7 - The OSN-aware CDN mechanism

OSNs (with more than 400 tweets per minute including a YouTube video link [52] being published per minute) and largely contributes to internet traffic growth [56]. Consequently, CDN users can benefit from an incorporated mechanism of OSN-awareness over the CDN

infrastructure. In [111] Kilanioti and Papadopoulos introduce various efficient copying policies based on prediction of demand on OSNs for the dynamic mechanism of preactive content copying.

Rather than pushing data to all CDN surrogates, they proactively distribute it only to social connections of the user likely to consume it. The content is copied only under certain conditions (content with high viewership within the media service, copied to geographically close timezones of the geo-diversed system used where the user has mutual social connections of high influence impact). This contributes to smaller response times for the content to be consumed (for the users) and lower bandwidth costs (for the OSN provider). Herein, we incorporate the proposed Predictive Model in the suggested policy [111] and prove that it further improves its performance.

The proposed herein algorithm encompasses an algorithm for each new request arriving in the CDN and an algorithm for each new object in the surrogate server. We use LRU for handling non-homogeneous sized objects in the surrogates and prune the least recently used items first. To ensure that least recently used items are discarded, the algorithm keeps track of their usage in a scheme described in [111]. Internally, the module implementing the algorithm communicates with the module processing the requests and each addressed server separately (Fig. 69).

7.5.1 For Every New Request in the CDN

To begin with, we check whether specific time has elapsed after the start of cascade and, only in the case of an uncompleted cascade, define to what extent the object will be copied. We introduce the *time_threshold*, that roughly expresses the average cascade duration. Initially, we check whether it is the first appearance of the object (Fig. 70).

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 its Score.

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 timezone and to the surrogate servers serving the timezones of all followers of the user. Otherwise, selective copying includes only the surrogates that the subpolicy decides. Subpolicy (Fig. 71) checks the Y timezones with

the highest value of the combined feature set (Predictive Model(*Score*, *dScore*, *content_dist*)) for the user $V_i(t)$ as an average. Copying is performed to the surrogate servers that serve the Y timezones of highest combined feature set value, according to the coefficients derived from our analysis. We note here that variations of the Subpolicy include the replacement of the timezones 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.

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.Score* > *Score_threshold* **then**
12: copy object *o* to surrogate that serves user's $V_i(t)$ timezone;
13: **for all** user $V_y(t)$ that follows user $V_i(t)$ **do**
14: find surrogate server n_j that serves $V_y(t)$'s timezone;
15: copy object *o* to n_j;
16: **end for**
17: **else if** o.timer < time_threshold **then**
18: copy object *o* to surrogates n_j that Subpolicy decides;
19: **end if**

Figure 70: Chapter 7 - Algorithm for every new request ($timestamp$, $V_i(t)$, o) in the CDN.

1: find the Y timezones that depict the highest average values of Predictive Model(*Score*, *dScore*, *content_dist*) user $V_i(t)$;
2: **for all** timezones that belong to Y **do**
3: find surrogate server n_j that serves timezone;
4: copy object o to n_j;
5: **end for**

Figure 71: Chapter 7 - Subpolicy

The realistic network depiction for the simulation of nodes representing the surrogate servers, the origin servers, and the users making the object requests is analysed in detail in Chapter 3, along with information about the dataset and the configuration settings of the simulations.

We examine Mean Response Time (MRT), the most significant client-side metric associated with user experience in CDNs. MRT indicates how fast a client is satisfied. For the most representative case of time thresholds covering all the examined requests of our dataset we observe a trade-off between the reduction of the response time and the cost of copying in servers. This is expressed for all schemes used (Linear Regression, Naive Bayes, Random Forest, SVM, SGD, KNN) with a decrease of the MRT as the timezones increase and a point after which the MRT starts to increase again (Fig. 72). For the scheme augmented with our Predictive Model, namely the Linear Regression, this shift occurs with approximately 7 timezones out of the 10 used. 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 be made. We observe that Linear Regression outperforms all the other schemes, depicting MRTs smaller than their respective. The proposed model performs better than the algorithm suggested in [110], and its variations [111], depicting an average MRT of 1.0647 msec. We note here that timezones with highest average values for each scheme, that Subpolicy defines, are pre-calculated, in order to reduce computational burden in the simulations.

$G(V,E)$	Graph representing the social network
$V = \{V_1, ..., V_n\}$	Nodes representing the social network users
$E = \{E_{11}, ..., E_{1n}, ..., E_{nn}\}$	Edges representing the social network connections, where E_{ij} stands for friendship between i and j
$R = \{r_1, r_2, ..., r_\tau\}$	Regions set
$N = \{n_1, n_2, ..., n_u\}$	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
$O = \{o_1, o_2, ..., o_w\}$	Objects set (videos)
$S_i, i \in O$	Size of object i in bytes
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
$P = \{p_{12}, p_{13}, ..., p_{nw}\}$	User posts in the social network, where p_{ij} denotes that node i has shared object j in the social network
$Q = \{q_1, q_2, ..., q_\zeta\}$	Object requests, where q_i denotes a request for an object of set O
Q_{hit_i}, $Q_{total_i}, i \in N$	Number of requests served from surrogate servers of the region of the user/ total number of requests
$X, Y, L, H \in R$	Closest timezones with mutual followers/ with highest centrality metric/ with highest lobby values/ with highest HITS values

Table 36: Chapter 7 - Notation Overview

7.6 Conclusions

Circulation via OSNs further intensifies the problem of HTTP traffic caused by online generated content. We conclude herein that video sharings over an OSN platform can be predicted with a small set of features combined from both the platform and the media service. Despite the focused scope of this work and the limitations of its conduction merely

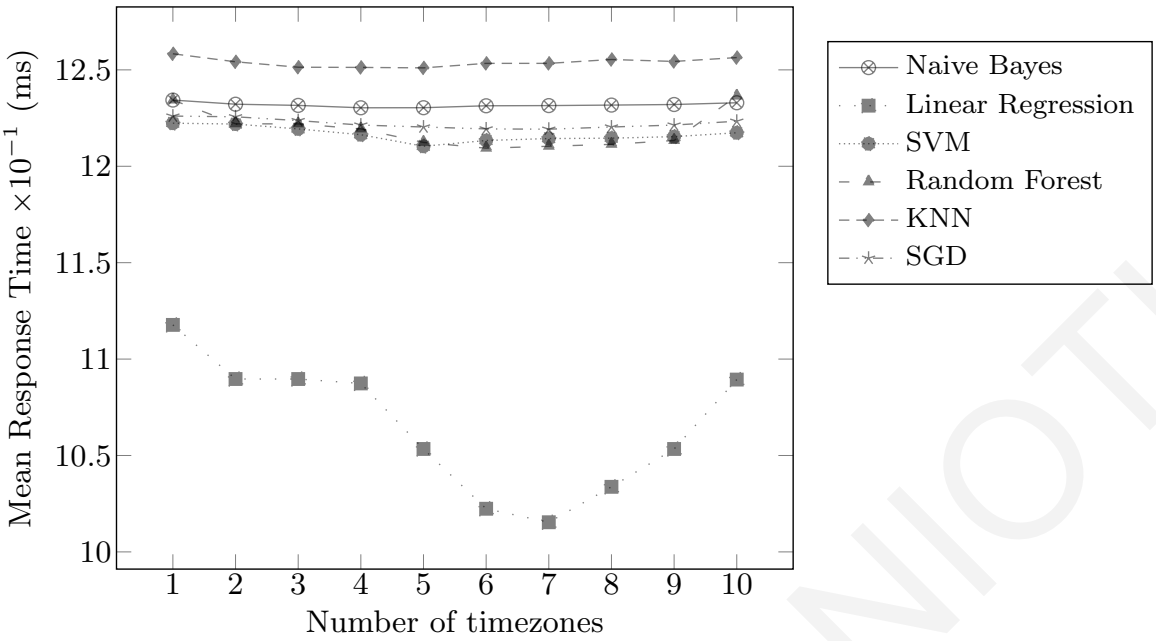


Figure 72: Chapter 7 - Effect of timezones used as Y on Mean Response Time for various schemes

with Twitter and YouTube data, the scale of the medium allows us to make assumptions for generalisation across different OSNs and microblog platforms, which we plan to extensively analyse in the future. Our results have direct application to the optimal placement of online video content. We believe that our approach can serve as a useful starting point for extensive experimentation with OSN-aware content delivery schemes.

Chapter 8

Conclusions and Future Research

8.1 Conclusions

Multimedia big data from entertainment and social media, medical images, consumer images, voice and video drives research and development of related technologies and applications and is steadily becoming a valuable source of information and insights. Multimedia content providers such as YouTube strive to efficiently deliver multimedia big data to a large amount of users over the Internet, with currently more than 300 hours of video content being uploaded to the site every minute. Traditionally, these content providers often rely on CDN infrastructures. However, some measurement studies depict that a significantly large proportion of HTTP traffic results from bandwidth-intensive multimedia content circulating through OSNs. Consequently we can exploit the user activity extracted from OSNs to reduce the bandwidth usage. By incorporating patterns of information transmission over OSNs into a simulated CDN infrastructure, we demonstrated in this Thesis that the performance of CDNs can be remarkably improved.

Especially today that HTTP traffic ascribed to media circulating over OSNs has grown, an OSN-awareness mechanism over a CDN has become essential 3. This mechanism, introduced in the Thesis, aims to exploit patterns of social interactions of the users to reduce the load on the origin server, the traffic on the Internet, and ultimately improve the user experience. By addressing the issue of which content will be copied in the surrogate servers of a CDN, it ensures a near-optimal content diffusion placement. At the same time, it moderates the impact on bandwidth that the Big Data transmitted via OSNs has, offering scalable solutions to existing CDNs or OSNs providers. Furthermore, we have experimented with variations on caching schemes, timing of content delivery and context of the OSN and the media platform.

Table 37 sums up the average values for all cases of testing. 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. Here, we present indicative values for the 10 closest timezones of mutual followers and experiments were conducted for varying subsets of 1, 5 and 10 timezones with the highest influence metric, respectively, where copying will ultimately be performed for all variations. The lowest mean response times appear for the cases of the time threshold covering all requests for all the variations (Fig. 73). In general, between Variation-1 and Variation-2 we observe a better performance in terms of mean response times achieved for the Variation-1, where the viewership within the YouTube platform is considered. All variations perform better than the plain Social Prefetcher approach apart from SIZE variation, 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 mean response time as the timezones increase, and a point after which the mean response time starts to increase again. This decrease in the mean response time occurs with approximately 5 timezones out of the 10 used for LRU scheme (for a fixed number of closest timezones with mutual followers), and with 7 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 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.

In terms of performance, we note that 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 Variation-3(LR, Chap. 7), whereas 30% in [176]) of other methods in pull-based methods employed by most CDNs, even though they moreover overlook storage issues of the distributed infrastructure.

Mean response time	**(Avg, $10^{-}2$ sec.)**
Variation-1 - 24-h	1.1383
Variation-1 - 48-h	1.1352
Variation-1 - all-h	1.1172
LFU - all-h	1.1112
SIZE - all-h	1.1274
LRU - all-h	1.1172
Variation-2 - 24-h	1.1411
Variation-2 - 48-h	1.1376
Variation-2 - all-h	1.1174
Variation-3 (LR)- all-h	1.0647
Variation-3 (KNN)- all-h	1.2611
Variation-3 (NB)- all-h	1.2437
Variation-3 (SGD)- all-h	1.2364
Variation-3 (RF)- all-h	1.2252
Variation-3 (SVM)- all-h	1.2232
Social Prefetcher 24-h	1.1412
Social Prefetcher 48-h	1.1377
Social Prefetcher all-h	1.1181
Plain CDN Simulator - Push	1.4471
Plain CDN Simulator - Pull	1.8460

Table 37: Summary - Average Metric Values for $X = 10$ Timezones of Close Mutual Friends

Plain CDN Simulator - Push case refers to proactive prefetching of content to all surrogate servers. In this case the mean response time 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.

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,

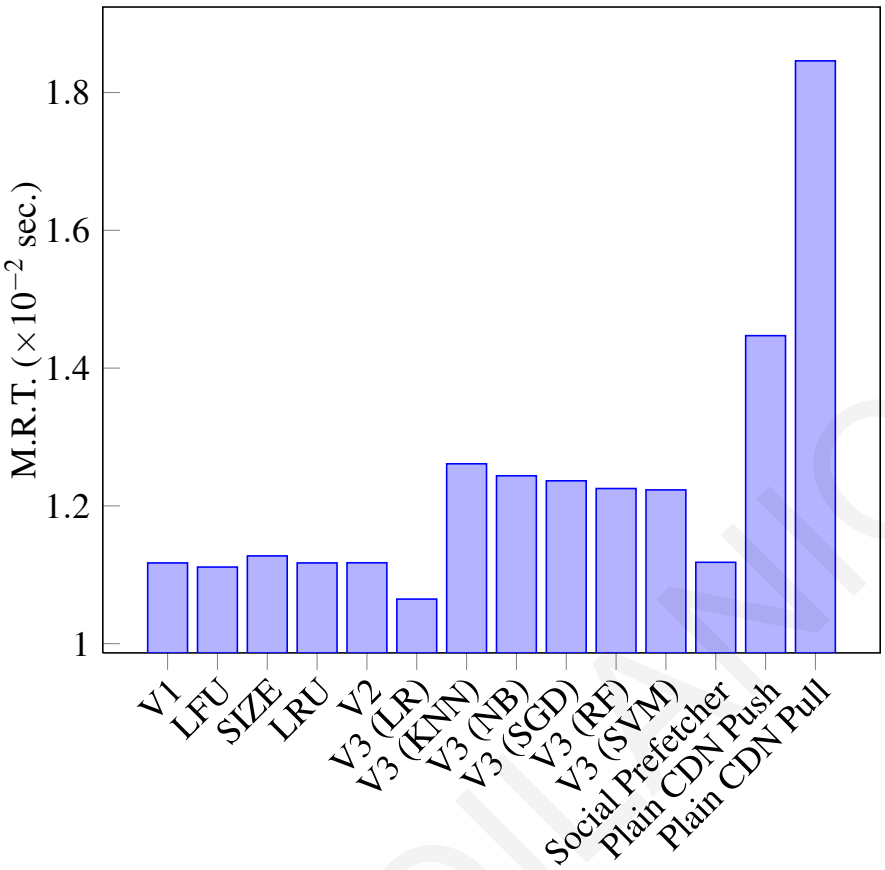


Figure 73: Summary - Average metric values for $X = 10$ timezones of close mutual friends for threshold covering all requests

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 mean response time under high loads and push strategies being superior under low to medium loads ([141], [75], [140]).

8.2 Future research directions

Understanding the effects of social cascading on bandwidth-intensive multimedia content over the Web is of great importance toward improving the performance of CDNs. Next generation CDNs are being 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 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 implemented and experimentally evaluated a dynamic policy of content prefetching, and thus, we have in our hands proof that OSNs can affect the content delivery infrastructure.

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. Consequently, we have to search for new ways to process the large volume of multimedia data created on a daily basis. The dynamic policy proposed herein can be further enriched with contextual information. 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.

Herein, we discuss some key areas of interest concerning the diffusion of bandwidth-intensive media content over OSNs.

8.2.1 Large-scale Datasets

In order to harness the power of social networks diffusion over CDN infrastructure, the key areas of interest that need to be explored include the large size of the graphs, and also

the fact that diffusion of links is multiplied through dissemination over sites like YouTube, and amplified by the proliferation of smartphones and cheap broadband connections. The amount of information in OSNs is an obstacle, since elaborate manipulation of the data may be needed. An open problem is the efficient handling of graphs with billions of nodes and edges.

The desired scaling property refers to the fact that the throughput of the proposed approaches should remain unchanged with the increase in the data input size, such as the large datasets that social graphs comprise and the social cascades phenomena that amplify the situation. Cost of scaling such content can be expressed in different ways. For instance, in this Thesis it matched with the number of replicas needed for a specific source. Future experimentations may take into account the optimal use of memory and processing time of a OSN-aware built system.

8.2.2 OSN Evolution

Existent works examine valuable insights into the dynamic world by posing queries on an evolving sequence of social graphs (e.g. [158]) and time evolving graphs tend to be increasingly used as a paradigm also for the emerging area of OSNs ([79]). However, the ability to scalably process queries concerning the information diffusion remains to a great extent unstudied. With the exception of sporadic works on specialized problems, such as that of inference of dynamic networks based on information diffusion data [160], we are not aware of relative studies on the information diffusion through OSNs under the prism of graphs dynamicity.

8.2.3 Semantic annotation

The introduction of concrete, unified metrics for the characterization of the extent of the social dissemination (local or global cascades phenomena) is an open issue. A systematic incorporation of this quantified knowledge into the existent underlying content delivery infrastructure would be salutary for proactive steps towards the improvement of user experience.

Last but not least, of more concurrent cascades happening it would be interesting to know which of them will evolve as global and which of them will evolve as local, possibly making some associations with their content or context features. It is challenging to discover contextual associations among the topics, which are by nature implicit in the user-generated

content exchanged over OSNs and spread via social cascades. In other words we would like to derive semantic relations. This way the identification of a popular topic can be conducted in a higher, more abstract level with the augmentation of asemantic annotation. While we can explicitly identify the topic of a single information disseminated through an OSN, it is not trivial to identify reliable and effective models for the adoption of topics as time evolves ([85], [133]) characterized with some useful emergent semantics.

8.2.4 Mobile CDNs and the Cloud

CDN services are increasingly being used to enable the delivery of bandwidth-demanding large media data to end-users of multimedia content providers and extend the capabilities of the Internet by deploying massively distributed infrastructures to accelerate content delivery. Mobile computing has created enormous demand for online experience, that OSN-aware CDNs are required to satisfy. Almost ubiquitous Wi-Fi coverage and rapid extension of mobile-broadband (around 78 active subscriptions per 100 inhabitants in Europe and America) provide undisrupted connectivity for mobile devices, whereas 97% of the world's population is reported to own a cellular subscription in 2015 [8]. Mobile-specific optimizations for applications along with drastically simplified and more intuitive use of devices (e.g. with multi-touch interactions instead of physical keyboards) contribute to mobile applications becoming the premium mode of accessing the Internet, at least in the US [9].

Cloud service providers have added CDN services in order to lower costs while increasing simplicity. CDNs, often operated as SaaS in cloud providers (Amazon CloudFront, Microsoft Azure CDN, etc.) aim at addressing the problem of smooth and transparent content delivery. A CDN actually drives cloud adoption through enhanced performance, scalability and cost reduction. With the limitation for both CDNs and cloud services being the geographic distance between a user asking for content and the server where the content resides, cloud acceleration and CDN networks are both complementary to achieving a goal of delivering data in the fastest possible way. Although cloud mainly handles constantly changing and, thus, not easily cached dynamic content, utilization of OSN-aware CDNs in cloud computing is likely to have profound effects on large data download.

8.2.5 Recommender Systems

Recommender systems aim at providing users with recommendations of new products / friends / events / activities and may also be associated with content delivery, e.g. within

multimedia content delivery platforms, where the product may be a multimedia resource (e.g. more than 66% of Netflix movies are recommended [55]). Moreover, OSNs today collect voluminous information from users' social contacts and their daily interactions (co-rating of products, co-commenting of posts, etc.) to provide them with useful recommendations. Hence, this Thesis could be extended in the future to evaluate the prefetching of multimedia content within recommender system platforms, as well as the exploitation of the social-awareness component to improve recommendation suggestions.

The challenges for the provision of recommendation within OSNs for the state-of-the-art recommendation algorithms, including collaborative filtering, semantic enhanced algorithms, etc., in the realm of multimedia OSNs include: (i) the heterogeneity of nodes, as nodes of the graphs may include not only users, but also activities, events, enhancement with geo-location information and information deriving from daily interactions with social contacts,(ii) the cold-start problem for newly-registered users or users with insufficient information, (iii) the necessary fast response due to frequent change of users' location and the energy limitations of their devices.

References

[1] "Akamai." https://www.akamai.com/de/de/about/facts-figures.jsp, [Online; accessed 1-Sept-2016].

[2] "Alexa," http://alexa.com/topsites, [Online; accessed 1-Sept-2016].

[3] "CDNsim," http://oswinds.csd.auth.gr/CDNsim, [Online; accessed 1-Sept-2016].

[4] "Center for Applied Internet Data Analysis," https://www.caida.org, [Online; accessed 1-Sept-2016].

[5] "Facebook Newsroom," http://newsroom.fb.com/Key-Facts, [Online; accessed 1-Sept-2016].

[6] "Idiro," http://www.idiro.com/products-services/sna-for-social-networks/, [Online; accessed 1-Sept-2016].

[7] "IGraph," http://igraph.sourceforge.net/, [Online; accessed 1-Sept-2016].

[8] "International Telecommunication Union. 2015. ict facts and figures. the world in 2015." https://goo.gl/cqOSXA, [Online; accessed 1-Sept-2016].

[9] "Internet Society. Global Internet Report 2015. Mobile Evolution and Development of the Internet." http://goo.gl/wUMJ8y, [Online; accessed 1-Sept-2016].

[10] "Limelight." http://www.limelight.com, [Online; accessed 1-Sept-2016].

[11] "Netevviz," https://wiki.cs.umd.edu/cmsc734_11/index.php?title=NetEvViz, [Online; accessed 1-Sept-2016].

[12] "NetMiner," http://www.netminer.com/index.php, [Online; accessed 1-Sept-2016].

[13] "NetworkX," http://networkx.github.com/, [Online; accessed 1-Sept-2016].

[14] "NodeXL," http://nodexl.codeplex.com/, [Online; accessed 1-Sept-2016].

[15] "NSSim," http://www.isi.edu/nsnam/ns/, [Online; accessed 1-Sept-2016].

[16] "Open source implementation of MapReduce." http://hadoop.apache.org/.

[17] "PlanetLab," http://www.planet-lab.org/, [Online; accessed 1-Sept-2016].

[18] "SNAP: Standford Network Analysis Project," http://snap.stanford.edu/, [Online; accessed 1-Sept-2016].

[19] "Twitter," https://about.twitter.com/company, [Online; accessed 1-Sept-2016].

[20] "Twitter official blog," https://blog.twitter.com/, [Online; accessed 1-Sept-2016].

[21] "Under the Hood: Scheduling MapReduce jobs more efficiently with Corona, Facebook engineering," goo.gl/XJRNN, [Online; accessed 1-Sept-2016].

[22] "Vine," https://vine.co, [Online; accessed 1-Sept-2016].

[23] "X-Rime: Hadoop based large scale social network analysis," http://xrime.sourceforge.net/, [Online; accessed 1-Sept-2016].

[24] "YouTube," https://www.youtube.com/yt/about/, [Online; accessed 1-Sept-2016].

[25] "YouTube official blog," http://youtube-global.blogspot.gr/2012_12_01_archive.html, [Online; accessed 1-Sept-2016].

[26] "YouTube Statistics," https://www.youtube.com/yt/press/statistics.html, [Online; accessed 1-Sept-2016].

[27] A. Abhari and M. Soraya, "Workload generation for YouTube," *Multimedia Tools Appl.*, vol. 46, no. 1, pp. 91–118, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11042-009-0309-5

[28] A. Abisheva, V. R. K. Garimella, D. Garcia, and 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. [Online]. Available: http://doi.acm.org/10.1145/2556195.2566588

[29] C. Aggarwal, J. L. Wolf, and P. S. Yu, "Caching on the world wide web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 94–107, Jan 1999.

[30] Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, "Analysis of topological characteristics of huge online social networking services," in *Proceedings of the 16th International Conference on World Wide Web*. ACM, 2007, pp. 835–844.

[31] M. Aizenman and J. L. Lebowitz, "Metastability effects in Bootstrap Percolation," *Journal of Physics A: Mathematical and General*, vol. 21, no. 19, p. 3801, 1999.

[32] N. Alon, M. Feldman, A. D. Procaccia, and M. Tennenholtz, "A note on competitive diffusion through social networks," *Information Processing Letters*, vol. 110, no. 6, pp. 221–225, 2010.

[33] A. Anagnostopoulos, R. Kumar, and M. Mahdian, "Influence and correlation in social networks," in *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge discovery and data mining*. ACM, 2008, pp. 7–15.

[34] C. Anderson, *Long Tail, The, Revised and Updated Edition: Why the Future of Business is Selling Less of More*. Hyperion, 2008.

[35] W. B. Arthur, "Competing technologies, increasing returns, and lock-in by historical events," *The economic journal*, vol. 99, no. 394, pp. 116–131, 1989.

[36] S. Asmussen, *Applied probability and queues*. Springer Science & Business Media, 2008, vol. 51.

[37] R. B. S. Asur, R. Bandari, and B. Huberman, "The pulse of news in social media: Forecasting popularity," *Association for the Advancement of Artificial Intelligence*, vol. 1202, 2012.

[38] B. Bahmani, R. Kumar, M. Mahdian, and E. Upfal, "PageRank on an evolving graph," in *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, 2012, pp. 24–32. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339539

[39] E. Bakshy, J. M. Hofman, W. A. Mason, and 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. [Online]. Available: http://doi.acm.org/10.1145/1935826.1935845

[40] E. Bakshy, I. Rosenn, C. Marlow, and L. A. Adamic, "The role of social networks in information diffusion," in *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, 2012, pp. 519–528. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187907

[41] A. V. Banerjee, "A simple model of herd behavior," *The Quarterly Journal of Economics*, vol. 107, no. 3, pp. 797–817, 1992.

[42] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[43] F. M. Bass, "A new product growth for model consumer durables," *Management Science*, vol. 15, no. 5, pp. 215–227, 1969.

[44] G. Baxter, S. Dorogovtsev, A. Goltsev, and J. Mendes, "Bootstrap percolation on complex networks," *Physical Review E*, vol. 82, no. 1, p. 011103, 2010.

[45] F. Benevenuto, T. Rodrigues, M. Cha, and 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.

[46] E. Berger, "Dynamic monopolies of constant size," *Journal of Combinatorial Theory, Series B*, vol. 83, no. 2, pp. 191–200, 2001.

[47] S. Bikhchandani, D. Hirshleifer, and I. Welch, "A theory of fads, fashion, custom, and cultural change as informational cascades," *Journal of political Economy*, pp. 992–1026, 1992.

[48] P. Bonacich, "Power and centrality: A family of measures," *American journal of sociology*, pp. 1170–1182, 1987.

[49] S. P. Borgatti, "Centrality and network flow," *Social Networks*, vol. 27, no. 1, pp. 55 – 71, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378873304000693

[50] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *INFOCOM'99. Eighteenth Annual*

Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 1. IEEE, 1999, pp. 126–134.

[51] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer networks and ISDN systems*, vol. 30, no. 1, pp. 107–117, 1998.

[52] A. Brodersen, S. Scellato, and 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*, 2012, pp. 241–250. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187870

[53] E. Brynjolfsson, Y. J. Hu, and M. Smith, "From niches to riches: Anatomy of the Long Tail," *Sloan Management Review*, vol. 47, no. 4, pp. 67–71, 2006.

[54] J. L. C. Huang, A. Wang and K. Ross, "Measuring and evaluating large-scale CDNs," in *Internet Measurement Conference (IMC), 2008*, 2008, pp. 15–29.

[55] O. Celma, "Music recommendation," in *Music Recommendation and Discovery*. Springer, 2010, pp. 43–85.

[56] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and 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. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298309

[57] M. Cha, A. Mislove, B. Adams, and K. P. Gummadi, "Characterizing social cascades in Flickr," in *Proceedings of the 1st workshop on Online social networks*. ACM, 2008, pp. 13–18.

[58] K. Chard, S. Caton, O. Rana, and K. Bubendorfer, "Social Cloud: Cloud computing in social networks," in *IEEE International Conference on Cloud Computing, CLOUD 2010, Miami, FL, USA, 5-10 July, 2010*, 2010, pp. 99–106. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2010.28

[59] J. Cheng, L. A. Adamic, P. A. Dow, J. M. Kleinberg, and 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. [Online]. Available: http://doi.acm.org/10.1145/2566486.2567997

[60] J. Cheng, L. A. Adamic, J. M. Kleinberg, and J. Leskovec, "Do cascades recur?" in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 671–681. [Online]. Available: http://dx.doi.org/10.1145/2872427.2882993

[61] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of YouTube videos," in *16th International Workshop on Quality of Service, IWQoS 2008, University of Twente, Enskede, The Netherlands, 2-4 June 2008.*, 2008, pp. 229–238. [Online]. Available: http://dx.doi.org/10.1109/IWQOS.2008.32

[62] S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings algorithm," *The American Statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[63] F. Chierichetti, J. M. Kleinberg, and A. Panconesi, "How to schedule a cascade in an arbitrary graph," in *Proceedings of the 13th ACM Conference on Electronic Commerce*. ACM, 2012, pp. 355–368.

[64] G. Christodoulou, C. Georgiou, and G. Pallis, "The role of Twitter in YouTube videos diffusion," in *Web Information Systems Engineering - WISE 2012 - 13th International Conference, Paphos, Cyprus, November 28-30, 2012. Proceedings*, 2012, pp. 426–439. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35063-4_31

[65] P. Clifford and A. Sudbury, "A model for spatial conflict," *Biometrika*, vol. 60, no. 3, pp. 581–588, 1973.

[66] D. Daley and D. G. Kendall, "Stochastic rumours," *IMA Journal of Applied Mathematics*, vol. 1, no. 1, pp. 42–55, 1965.

[67] K. S. Dave, R. Bhatt, and 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. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2741

[68] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[69] L. Dickens, I. Molloy, J. Lobo, P.-C. Cheng, and A. Russo, "Learning stochastic models of information flow," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*. IEEE, 2012, pp. 570–581.

[70] Y. Dodge, D. Cox, D. Commenges, A. Davison, P. Solomon, and S. Wilson, *The Oxford dictionary of statistical terms*. Oxford University Press, USA, 2006.

[71] C. Doerr, N. Blenn, S. Tang, and P. Van Mieghem, "Are friends overrated? a study for the social news aggregator digg. com," *Computer Communications*, vol. 35, no. 7, pp. 796–809, 2012.

[72] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge discovery and data mining*. ACM, 2001, pp. 57–66.

[73] P. A. Dow, L. A. Adamic, and 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, July 8-11, 2013.*, 2013. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/6123

[74] M. Draief, A. Ganesh, and 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.

[75] D. L. Eager, E. D. Lazowska, and 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*, ser. SIGMETRICS 1985. New York, NY, USA: ACM, 1985, pp. 1–3. [Online]. Available: http://doi.acm.org/10.1145/317795.317802

[76] D. A. Easley and J. M. Kleinberg, *Networks, Crowds, and Markets - Reasoning About a Highly Connected World*. Cambridge University Press, 2010. [Online]. Available: http://www.cambridge.org/gb/knowledge/isbn/item2705443/?site_locale=en_GB

[77] P. Erdös and A. Rényi, "On random graphs, i," *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.

[78] E. Even-Dar and A. Shapira, "A note on maximizing the spread of influence in social networks," *Inf. Process. Lett.*, vol. 111, no. 4, pp. 184–187, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.ipl.2010.11.015

[79] A. Fard, A. Abdolrashidi, L. Ramaswamy, and J. A. Miller, "Towards efficient query processing on massive time-evolving graphs," in *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2012, Pittsburgh, PA, USA, October 14-17, 2012*, 2012, pp. 567–574. [Online]. Available: http://dx.doi.org/10.4108/icst.collaboratecom.2012.250532

[80] F. Figueiredo, F. Benevenuto, and J. M. Almeida, "The tube over time: characterizing popularity growth of YouTube videos," 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. 745–754. [Online]. Available: http://doi.acm.org/10.1145/1935826.1935925

[81] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "YouTube everywhere: impact of device and infrastructure synergies on user experience," in *Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement, IMC '11, Berlin, Germany, November 2-, 2011*, 2011, pp. 345–360. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068849

[82] J. Fowler and N. Christakis, "Connected: The surprising power of our social networks and how they shape our lives," *HarperCollins Publishers*, 2009.

[83] M. J. Freedman, E. Freudenthal, and D. Mazieres, "Democratizing content publication with Coral." in *NSDI*, vol. 4, 2004, pp. 18–18.

[84] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, and W. Kellerer, "Outtweeting the Twitterers - Predicting Information Cascades in Microblogs," in *3rd Workshop on Online Social Networks, WOSN 2010, Boston, MA, USA, June 22, 2010*, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863190.1863193

[85] A. García-Silva, J.-H. Kang, K. Lerman, and O. Corcho, "Characterising emergent semantics in Twitter lists," in *Proceedings of the 9th International Conference on The Semantic Web: research and applications (ESWC)*, Heraklion, Greece, 2012.

[86] M. Garetto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 1, no. 3, p. 12, 2016.

[87] E. N. Gilbert, "Random graphs," *The Annals of Mathematical Statistics*, pp. 1141–1144, 1959.

[88] P. Gill, M. F. Arlitt, Z. Li, and A. Mahanti, "YouTube traffic characterization: a view from the edge," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement 2007, San Diego, California, USA, October 24-26, 2007*, 2007, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298310

[89] J. Goldenberg, B. Libai, and E. Muller, "Talk of the network: A complex systems look at the underlying process of word-of-mouth," *Marketing letters*, vol. 12, no. 3, pp. 211–223, 2001.

[90] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *Proceedings of the third ACM International Conference on Web search and data mining*. ACM, 2010, pp. 241–250.

[91] M. Granovetter, "Threshold models of collective behavior," *American journal of Sociology*, pp. 1420–1443, 1978.

[92] P. Grindrod, M. C. Parsons, D. J. Higham, and E. Estrada, "Communicability across evolving networks," *Physical Review E*, vol. 83, no. 4, p. 046120, 2011.

[93] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins, "Information diffusion through blogspace," in *Proceedings of the 13th International Conference on World Wide Web*. ACM, 2004, pp. 491–501.

[94] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh, "WTF: the who to follow service at Twitter," in *22nd International World Wide Web Conference, WWW 2013, Rio de Janeiro, Brazil, May 13-17, 2013*, 2013, pp. 505–514. [Online]. Available: http://dl.acm.org/citation.cfm?id=2488433

[95] T. Hogg and K. Lerman, "Stochastic models of user-contributory web sites," in *Proceedings of the 3rd International Conference on Weblogs and Social Media, ICWSM 2009, San Jose, California, USA, May 17-20, 2009*, 2009. [Online]. Available: http://aaai.org/ocs/index.php/ICWSM/09/paper/view/148

[96] R. A. Holley and T. M. Liggett, "Ergodic theorems for weakly interacting infinite systems and the voter model," *The annals of probability*, pp. 643–663, 1975.

[97] L. Hong, O. Dan, and 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. [Online]. Available: http://doi.acm.org/10.1145/1963192.1963222

[98] I. Hoque and I. Gupta, "Disk layout techniques for online social network data," *Internet Computing, IEEE*, vol. 16, no. 3, pp. 24–36, 2012.

[99] T. Hoßfeld, R. Schatz, E. Biersack, and 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.

[100] K. Huguenin, A.-M. Kermarrec, K. Kloudas, and 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.

[101] E. Ising, "Beitrag zur Theorie des Ferromagnetismus," *Zeitschrift für Physik A Hadrons and Nuclei*, vol. 31, no. 1, pp. 253–258, 1925.

[102] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. Briggs, and R. Braynard, "Networking named content," *Commun. ACM*, vol. 55, no. 1, pp. 117–124, 2012. [Online]. Available: http://doi.acm.org/10.1145/2063176.2063204

[103] A. Java, X. Song, T. Finin, and 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.

[104] M. Jenders, G. Kasneci, and 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. [Online]. Available: http://dl.acm.org/citation.cfm?id=2488017

[105] U. Kang, H. Tong, J. Sun, C.-Y. Lin, and C. Faloutsos, "Gbase: a scalable and general graph management system," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge discovery and data mining, ACM*, 2011, pp. 1091–1099.

[106] U. Kang, C. E. Tsourakakis, and C. Faloutsos, "PEGASUS: A peta-scale graph mining system," in *ICDM 2009, The 9th IEEE International Conference on Data Mining, Miami, Florida, USA, 6-9 December 2009*, 2009, pp. 229–238. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2009.14

[107] E. Katz and P. F. Lazarsfeld, *Personal Influence, The part played by people in the flow of mass communications*. Transaction Publishers, 1966.

[108] D. Kempe, J. Kleinberg, and É. Tardos, "Influential nodes in a diffusion model for social networks," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2005, pp. 1127–1138.

[109] D. Kempe, J. M. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. ACM, 2003, pp. 137–146.

[110] I. Kilanioti, "Improving multimedia content delivery via augmentation with social information. The Social Prefetcher approach." *Multimedia, IEEE Transactions on*, vol. 17, no. 9, pp. 1460–1470, 2015. [Online]. Available: http://dx.doi.org/10.1109/TMM.2015.2459658

[111] I. Kilanioti and G. A. Papadopoulos, "Socially-aware multimedia content delivery for the cloud," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, Dec 2015, pp. 300–309.

[112] I. Kilanioti, C. Georgiou, and G. Pallis, "On the impact of online social networks in content delivery," in *Advanced Content Delivery and Streaming in the Cloud*, M. Pathan, R. Sitaraman, and D. Robinson, Eds. Wiley, 2014.

[113] I. Kilanioti and G. A. Papadopoulos, "Efficient content delivery through popularity forecasting on social media," in *Proceedings of the 7th IEEE International Conference on Information, Intelligence, Systems and Applications, IISA 2016, Chalkidiki, Greece, July 13-15, 2016*, pp. 13–19.

[114] ——, "Content delivery simulations supported by social network-awareness." Elsevier, 2016, vol. ChipSet Special Issue, under review.

[115] ——, "Delivering social multimedia content with scalability," in *Resource Management for Big Data Platforms: Algorithms, Modelling and High-Performance Computing Techniques, Springer Computer Communications and Networks Series*, J. K. Florin Pop and B. di Martino, Eds. Springer, 2016, ch. 18.

[116] ——, "Predicting video virality on Twitter," in *Resource Management for Big Data Platforms: Algorithms, Modelling and High-Performance Computing Techniques, Springer Computer Communications and Networks Series*, J. K. Florin Pop and B. di Martino, Eds. Springer, 2016, ch. 20.

[117] J. M. Kleinberg, "Cascading behavior in networks: Algorithmic and economic issues," in *Algorithmic game theory (N. Nisan, T. Roughgarden, E. Tardos, V. Vazirani, Eds.)*. Cambridge University Press, 2007, pp. 613–632.

[118] ——, "Authoritative sources in an hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, 1999. [Online]. Available: http://doi.acm.org/10.1145/324133.324140

[119] ——, "Navigation in a Small World," *Nature*, vol. 406, no. 6798, pp. 845–845, 2000.

[120] A. Korn, A. Schubert, and A. Telcs, "Lobby index in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 11, pp. 2221–2226, 2009.

[121] R. Kumar, J. Novak, and A. Tomkins, "Structure and evolution of Online Social Networks," *Link Mining: Models, Algorithms, and Applications*, pp. 337–357, 2010.

[122] A. Kupavskii, L. Ostroumova, A. Umnov, S. Usachev, P. Serdyukov, G. Gusev, and 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. [Online]. Available: http://doi.acm.org/10.1145/2396761.2398634

[123] M. Kuperman and G. Abramson, "Small world effect in an epidemiological model," *Physical Review Letters*, vol. 86, no. 13, pp. 2909–2912, 2001.

[124] H. Kwak, C. Lee, H. Park, and 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. [Online]. Available: http://doi.acm.org/10.1145/1772690.1772751

[125] T. La Fond and J. Neville, "Randomization tests for distinguishing social influence and homophily effects," in *Proceedings of the 19th International Conference on World Wide Web*. ACM, 2010, pp. 601–610.

[126] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[127] K. Lerman, S. Intagorn, J.-H. Kang, and R. Ghosh, "Using proximity to predict activity in social networks," in *Proceedings of the 21st International Conference on World Wide Web*. ACM, 2012, pp. 555–556.

[128] J. Leskovec and E. Horvitz, "Planetary-scale views on a large instant-messaging network," in *Proceeding of the 17th International Conference on World Wide Web*. ACM, 2008, pp. 915–924.

[129] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, 2007.

[130] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst, "Patterns of cascading behavior in large blog graphs," in *Proceedings of SIAM International Conference on Data Mining (SDM) 2007*. SIAM, 2007.

[131] Y. Li, Y. Shen, and Y. Liu, "Utilizing content delivery network in cloud computing," in *Computational Problem-Solving (ICCP), 2012 International Conference on*, Oct 2012, pp. 137–143.

[132] D. Liben-Nowell and J. Kleinberg, "Tracing information flow on a global scale using Internet chain-letter data," *Proceedings of the National Academy of Sciences*, vol. 105, no. 12, pp. 4633–4638, 2008.

[133] C. X. Lin, Q. Mei, Y. Jiang, J. Han, and S. Qi, "Inferring the diffusion and evolution of topics in social communities," *mind*, vol. 3, no. d4, p. d5, 2011.

[134] T. Łuczak, "Size and connectivity of the K-core of a random graph," *Discrete Mathematics*, vol. 91, no. 1, pp. 61–68, 1991.

[135] Z. Ma, A. Sun, and G. Cong, "On predicting the popularity of newly emerging hashtags in Twitter," *JASIST*, vol. 64, no. 7, pp. 1399–1410, 2013. [Online]. Available: http://dx.doi.org/10.1002/asi.22844

[136] G. Maier, A. Feldmann, V. Paxson, and M. Allman, "On dominant characteristics of residential broadband internet traffic," in *Proceedings of the 9th ACM SIGCOMM Internet Measurement Conference IMC, pages=90–102, year=2009, organization=ACM*.

[137] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, pp. 415–444, 2001.

[138] Q. Mei, D. Cai, D. Zhang, and C. Zhai, "Topic modeling with network regularization," in *Proceeding of the 17th International Conference on World Wide Web*, 2008, pp. 101–110.

[139] S. Milgram, "The Small World problem," *Psychology today*, vol. 2, no. 1, pp. 60–67, 1967.

[140] W. Minnebo and B. Van Houdt, "Pull versus push mechanism in large distributed networks: Closed form results," in *Proceedings of the 24th International Teletraffic Congress*, ser. ITC 2012. International Teletraffic Congress, 2012, pp. 9:1–9:8. [Online]. Available: http://dl.acm.org/citation.cfm?id=2414276.2414288

[141] R. Mirchandaney, D. Towsley, and J. A. Stankovic, "Adaptive load sharing in heterogeneous distributed systems," *Journal of parallel and distributed computing*, vol. 9, no. 4, pp. 331–346, 1990.

[142] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and 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.

[143] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. L. Eager, and A. Mahanti, "Characterizing web-based video sharing workloads," *TWEB*, vol. 5, no. 2, pp. 1–27, 2011. [Online]. Available: http://doi.acm.org/10.1145/1961659.1961662

[144] J. Mondal and A. Deshpande, "Managing large dynamic graphs efficiently," in *Proceedings of the ACM SIGMOD International Conference on Management of*

Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012, 2012, pp. 145–156. [Online]. Available: http://doi.acm.org/10.1145/2213836.2213854

[145] S. Morris, "Contagion," *The Review of Economic Studies*, vol. 67, no. 1, pp. 57–78, 2000.

[146] E. Mossel and S. Roch, "On the submodularity of influence in social networks," in *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM, 2007, pp. 128–134.

[147] S. A. Myers and J. Leskovec, "The bursty dynamics of the Twitter information network," in *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014*, 2014, pp. 913–924. [Online]. Available: http://doi.acm.org/10.1145/2566486.2568043

[148] A. Najar, L. Denoyer, and P. Gallinari, "Predicting information diffusion on social networks with partial knowledge," in *Proceedings of the 21st International Conference on World Wide Web*. ACM, 2012, pp. 1197–1204.

[149] R. Narayanam and Y. Narahari, "A shapley value-based approach to discover influential nodes in social networks," *IEEE Transactions on Automation Science and Engineering*, no. 99, pp. 1–18, 2010.

[150] M. Nekovee, Y. Moreno, G. Bianconi, and M. Marsili, "Theory of rumour spreading in complex social networks," *Physica A: Statistical Mechanics and its Applications*, vol. 374, no. 1, pp. 457–470, 2007.

[151] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functionsI," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[152] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press Cambridge, 2007, vol. 1.

[153] Z. L. P. Gill, M. Arlitt and A. Mahanti, "Characterizing user sessions on YouTube," in *ACM/SPIE Multimedia Computing and Networking Conference (MMCN '08), San Jose, USA, 2008*, 2008.

[154] G. Peng, "CDN: Content distribution network technical report tr-125," *Experimental Computer Systems Lab, Stony Brook University*, 2003.

[155] S. Petrovic, M. Osborne, and 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. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2754

[156] L. Plissonneau and 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.

[157] A. Rao, A. Legout, Y. Lim, D. Towsley, C. Barakat, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the 2011 Conference on Emerging Networking Experiments and Technologies, Co-NEXT '11, Tokyo, Japan, December 6-9, 2011*, 2011, p. 25. [Online]. Available: http://doi.acm.org/10.1145/2079296.2079321

[158] C. Ren, E. Lo, B. Kao, X. Zhu, and R. Cheng, "On querying historial evolving graph sequences," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, 2011.

[159] T. Rodrigues, F. Benevenuto, M. Cha, P. K. Gummadi, and V. A. F. Almeida, "On word-of-mouth based discovery of the web," in *Proceedings of the 11th ACM SIGCOMM Conference on Internet Measurement, IMC '11, Berlin, Germany, November 2-, 2011*, 2011, pp. 381–396. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068852

[160] M. G. Rodriguez, J. Leskovec, and B. Schölkopf, "Structure and dynamics of information pathways in online media," in *Proceedings of ACM International Conference on Web Search and Data Mining (WSDM)*, Rome, Italy, 2013.

[161] E. M. Rogers, *Diffusion of innovations*. Simon and Schuster, 1995.

[162] K. Saito, M. Kimura, K. Ohara, and H. Motoda, "Learning continuous-time information diffusion model for social behavioral data analysis," *Advances in Machine Learning*, pp. 322–337, 2009.

[163] K. Saito, R. Nakano, and M. Kimura, "Prediction of information diffusion probabilities for Independent Cascade Model," in *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, 2008, pp. 67–75.

[164] N. Sastry, E. Yoneki, and 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. [Online]. Available: http://doi.acm.org/10.1145/1578002.1578009

[165] S. Scellato, C. Mascolo, M. Musolesi, and 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. [Online]. Available: http://doi.acm.org/10.1145/1963405.1963471

[166] T. C. Sendling, "Micromotives and macrobehavior," *New York and London: Norton*, 1978.

[167] X. Song, Y. Chi, K. Hino, and B. L. Tseng, "Information flow modeling based on diffusion rate for prediction and ranking," in *Proceedings of the 16th International Conference on World Wide Web*. ACM, 2007, pp. 191–200.

[168] K. Stamos, G. Pallis, A. Vakali, D. Katsaros, A. Sidiropoulos, and Y. Manolopoulos, "CDNsim: A simulation tool for Content Distribution Networks," *ACM Trans. Model. Comput. Simul.*, vol. 20, no. 2, pp. 1–40, 2010. [Online]. Available: http://doi.acm.org/10.1145/1734222.1734226

[169] D. Stauffer and A. Aharony, *Introduction to Percolation Theory*. CRC, 1994.

[170] G. V. Steeg, R. Ghosh, and 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. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/2781

[171] B. Suh, L. Hong, P. Pirolli, and E. H. Chi, "Want to be retweeted? large scale analytics on factors impacting retweet in Twitter network," in *Proceedings of the 2010 IEEE Second International Conference on Social Computing, SocialCom / IEEE International Conference on Privacy, Security, Risk and Trust, PASSAT 2010, Minneapolis, Minnesota, USA, August 20-22, 2010*, 2010, pp. 177–184. [Online]. Available: http://dx.doi.org/10.1109/SocialCom.2010.33

[172] G. Szabó and B. A. Huberman, "Predicting the popularity of online content," *Commun. ACM*, vol. 53, no. 8, pp. 80–88, 2010. [Online]. Available: http://doi.acm.org/10.1145/1787234.1787254

[173] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat, "MediSyn: a synthetic streaming media service workload generator," in *Network and Operating System Support for Digital Audio and Video, 13th International Workshop, NOSSDAV 2003, Monterey, CA, USA, June 1-3, 2003, Proceedings*, 2003, pp. 12–21. [Online]. Available: http://doi.acm.org/10.1145/776322.776327

[174] R. Torres, A. Finamore, J. R. Kim, M. Mellia, M. M. Munafò, and 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. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2011.43

[175] D. Towsley, A. Rao, Y.-S. Lim, C. Barakat, A. Legout, and W. Dabbous, "Network characteristics of video streaming traffic," in *Proceedings of the 7th Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, New York, NY, USA, 2011.

[176] S. Traverso, K. Huguenin, I. Trestian, V. Erramilli, N. Laoutaris, and 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. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187858

[177] S. Triukose, Z. Wen, and M. Rabinovich, "Measuring a commercial Content Delivery Network," in *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, 2011, pp. 467–476. [Online]. Available: http://doi.acm.org/10.1145/1963405.1963472

[178] O. Tsur and 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. [Online]. Available: http://doi.acm.org/10.1145/2124295.2124320

[179] D. Vallet, S. Berkovsky, S. Ardon, A. Mahanti, and M. A. Kafaar, "Characterizing and predicting viral-and-popular video content," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: ACM, 2015, pp. 1591–1600. [Online]. Available: http://doi.acm.org/10.1145/2806416.2806556

[180] C. Van den Bulte and Y. V. Joshi, "New product diffusion with influentials and imitators," *Marketing Science*, vol. 26, no. 3, pp. 400–421, 2007.

[181] N. Vlajic, C. D. Charalambous, and D. Makrakis, "Performance aspects of data broadcast in wireless networks with user retrials," *IEEE/ACM Transactions on Networking (TON)*, vol. 12, no. 4, pp. 620–633, 2004.

[182] L. Wang, K. Park, R. Pang, V. S. Pai, and L. L. Peterson, "Reliability and security in the CoDeeN Content Distribution Network." in *USENIX Annual Technical Conference, General Track*, 2004, pp. 171–184.

[183] S. Wasserman and K. Faust, *Social Network Analysis: Methods and applications*. Cambridge university press, 1994, vol. 8.

[184] D. Watts and S. Strogatz, "The Small World problem," *Collective Dynamics of Small-World Networks*, vol. 393, pp. 440–442, 1998.

[185] D. J. Watts, "A simple model of global cascades on random networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 9, pp. 5766–5771, 2002.

[186] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: finding topic-sensitive influential Twitterers," in *Proceedings of the third ACM International Conference on Web search and data mining*. ACM, 2010, pp. 261–270.

[187] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and 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.

[188] F. Wu and B. A. Huberman, "Novelty and collective attention," *Proceedings of the National Academy of Sciences*, vol. 104, no. 45, pp. 17 599–17 601, 2007.

[189] J. Xu, D.-L. Lee, Q. Hu, and W.-C. Lee, "Handbook of wireless networks and mobile computing." New York, NY, USA: John Wiley & Sons, Inc., 2002, ch.

Data Broadcast, pp. 243–265. [Online]. Available: http://dl.acm.org/citation.cfm?id=512321.512332

[190] J. Yang and J. Leskovec, "Modeling information diffusion in implicit networks," in *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*. IEEE, 2010, pp. 599–608.

[191] ——, "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. [Online]. Available: http://doi.acm.org/10.1145/1935826.1935863

[192] S. Yang and 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. [Online]. Available: http://jmlr.org/proceedings/papers/v28/yang13a.html

[193] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," in *ACM SIGOPS Operating Systems Review*, vol. 40, no. 4. ACM, 2006, pp. 333–344.

[194] N. J. Yuan, Y. Zhong, F. Zhang, X. Xie, C.-Y. Lin, and Y. Rui, "Who will reply to/retweet this tweet?: The dynamics of intimacy from online social interactions," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, ser. WSDM '16. New York, NY, USA: ACM, 2016, pp. 3–12. [Online]. Available: http://doi.acm.org/10.1145/2835776.2835800

[195] F. Zhou, L. Zhang, E. Franco, A. Mislove, R. Revis, and 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.

Appendix A

Request Generator

The media workload generator developed in the context of the leverage of OSN information takes as input controllable parameters and produces traces from desired distributions.

After crawling of Twitter in [64] (users, tweets), tweets were collected with links to YouTube videos and extracted real video links, ultimately obtaining posts for 1,384,758 users and information for 1,113,294 unique videos that Twitter users have shared.

For the construction of the request generator, the request distributions in hours of the day and days of the week [58] were taken into account. For example, the requests increase dramatically at 12 noon, remain high until the night, and decrease significantly after midnight. There are significantly fewer numbers of requests during weekends and holidays than during working days in the week. Timestamps are counted in seconds after the start of the simulation. The generator provides an output for every one of the 142 timezones defined by Twitter.

The requests produced from the generator to evaluate our content prefetching policy should be realistic. Other systems declared in their majority that the user will ask for an object it has viewed in a social network with an arbitrary probability. The time that users ask for content is not taken into account (there exist more requests in the rush hours, and rush hours differ for users in different timezones). This information is taken into account by the proposed generator, which can also be applied to other social networks and non-multimedia content.

1. The unique users per day graph in [88] are used. The measurement is performed at a university campus with a known population, thus enabling the scale up of the amount of requests according to the users in each region.

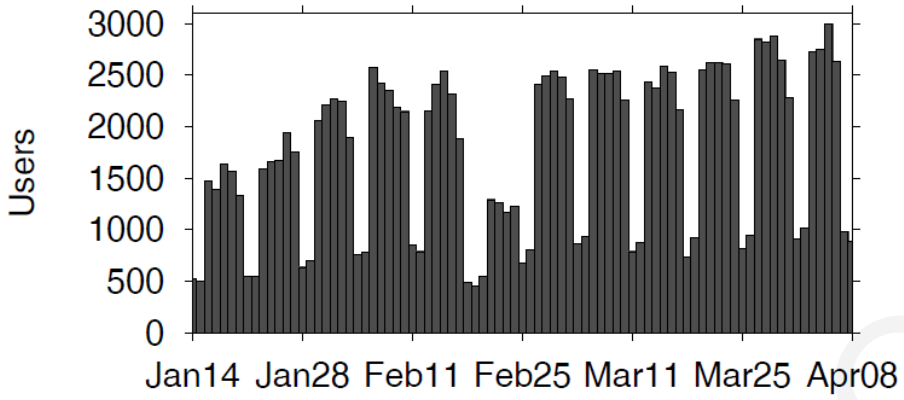


Figure 74: Unique users per day in [88]

2. The user sessions are distributed into hourly timeslots according to the hourly YouTube patterns by [88].

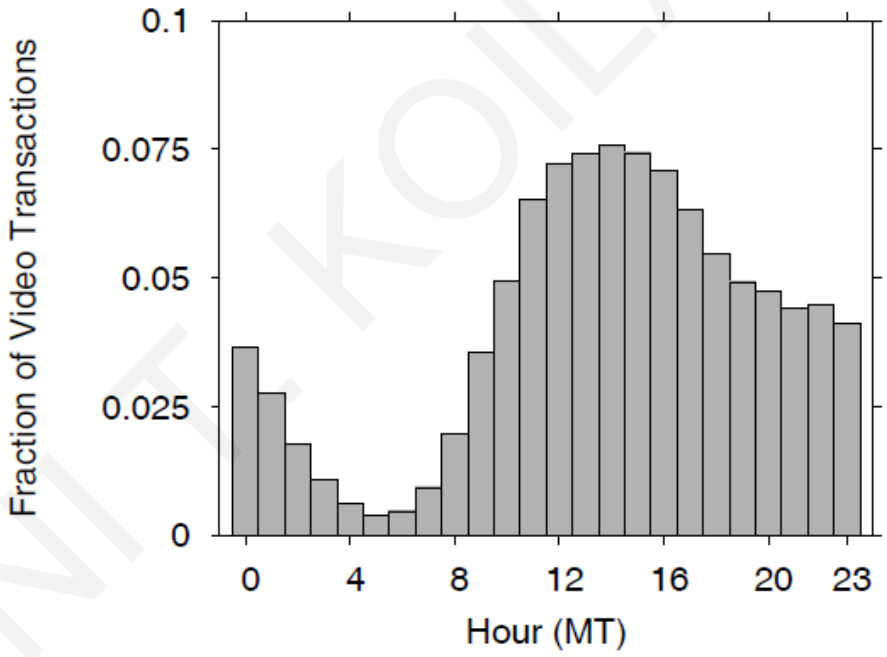


Figure 75: User sessions in [88]

3. Each session is assigned a duration according to the duration CDF presented in [153].

4. For each session starting at moment η: We select a video for that session creating the appropriate request, and the session duration is decreased by the duration of the video. The session is set to start again when the video ends.

   To choose the video to be viewed, the approach described in [27] is used, denoting VC: view count of each video, σ: sum of the view counts of all videos in the catalog,

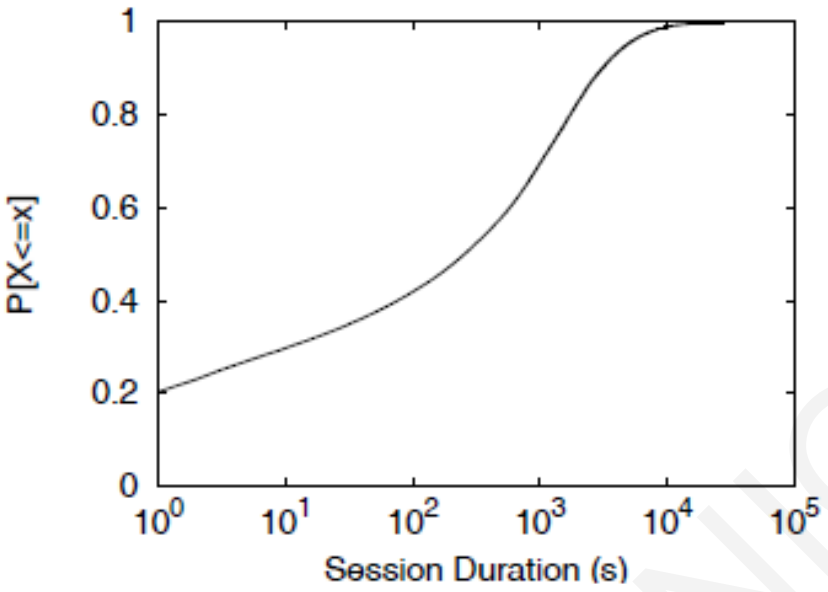


Figure 76: Session durations in [153]

and ν: the number of videos. To select a video, a random number μ between 1 and σ is created. Then, the video j $(1 \leq j \leq \nu)$ is found with $VC_j > \mu$ and $VC_{j-1} < \mu$. This gives a probability for a video proportional to its View Count, and thus, its actual popularity. If a video is also part of a social cascade for the current user, its VC is multiplied by a number to increase its popularity. The suggested probability increase is 7.35, as given in [40].

Appendix B

Acronyms

The following table describes the significance of various abbreviations and acronyms used throughout the Thesis. The page on which each one is defined or first used is also given.

AS	Autonomous System	64	SGD	Stochastic Gradient Descent	15
CDN	Content Distribution Network	1	SIR	Susceptible Infected Removed	23
HTTP	Hypertext Transfer Protocol	8	SIRS	Susceptible Infected Removed Susceptible	23
ICM	Independent Cascade Model	22	SIS	Susceptible Infected Susceptible	23
ISP	Internet Service Provider	51	SNA	Social Network Analysis	5
KNN	K-Nearest Neighbours	15	SVM	Support Vector Machines	15
LTM	Linear Threshold Model	22	TCP	Transmission Control Protocol	1
MRT	Mean Response Time	67	UDP	User Datagram Protocol	1
OSN	Online Social Network	1	UGC	User Generated Content	7
QoE	Quality of Experience	1	VOD	Video-on-Demand	18

Table 38: Acronyms used in Thesis