Low Complexity Content Replication through Clustering in Content-Delivery Networks

Lazaros Gkatzikis[a], Vasilis Sourlas[b], Carlo Fischione[c], Iordanis koutsopoulos[d]

[a] *Huawei Technologies, Paris, France.*
[b] *Dept. of Electronic and Electrical Engineering, University College London (UCL), UK.*
[c] *Dept. of Electrical Engineering, KTH, Stockholm, Sweden.*
[d] *Dept. of Informatics, AUEB, Athens, Greece.*

Abstract

Contemporary Content Delivery Networks (CDN) handle a vast number of content items. At such a scale, the replication schemes require a significant amount of time to calculate and realize cache updates, and hence they are impractical in highly-dynamic environments. This paper introduces cluster-based replication, whereby content items are organized in clusters according to a set of features, given by the cache/network management entity. Each cluster is treated as a single item with certain attributes, *e.g.,* size, popularity, *etc.* and it is then altogether replicated in network caches so as to minimize overall network traffic. Clustering items reduces replication complexity; hence it enables faster and more frequent caches updates, and it facilitates more accurate tracking of content popularity. However, clustering introduces some performance loss because replication of clusters is more coarse-grained compared to replication of individual items. This tradeoff can be addressed through proper selection of the number and composition of clusters. Due to the fact that the exact optimal number of clusters cannot be derived analytically, an efficient approximation method is proposed. Extensive numerical evaluations of time-varying content popularity scenarios allow to argue that the proposed approach reduces core network traffic, while being robust to errors in popularity estimation.

Keywords: Content replication, Content Clustering, Content-Delivery Networks, Coordinated caching.

1. Introduction

Content Delivery Networks (CDNs) currently account for 36% of the Internet traffic [1], and they are expected to carry more than half of such a traffic by 2018. In order to meet the growing demand for content, CDN providers deploy

Email addresses: lazaros.gkatzikis@huawei.com (Lazaros Gkatzikis), v.sourlas@ucl.ac.uk (Vasilis Sourlas), carlofi@kth.se (Carlo Fischione), jordan@aueb.gr (Iordanis koutsopoulos)

cache servers worldwide that host replicas of content. Each content request is redirected to the closest replica rather than being served by the back-end/origin server. Thus, through replication, content requests are served locally and this improves both user Quality of Experience (QoE) (*i.e.,* access latency) and minimizes core network traffic.

Current content delivery services operated by large CDN providers like Akamai [2], Limelight [3] and Netflix [4] can exert enormous strain on ISP networks [5]. This is mainly due to that CDN providers control both the placement of content in surrogate caches/servers spanning different geographic locations, as well as the decision on where to serve client requests from [6]. These decisions are taken without knowledge of the precise network topology and traffic load, and they can result in network performance degradation, thus affecting the experience of end users. To address this issue, the studies in [7][8] propose the notion of an Internet Service Provider (ISP) CDN. An ISP deploys caches over network nodes and manages the resulting limited-capacity distributed CDN service within its network. In contrast to CDN providers, ISPs have global knowledge about the utilization of their network, which causes the problem of optimal content replication in a network of caches[1].

Existing replication schemes rely on the assumption that the popularity of content items is static or changes slowly. Thus, given an estimate of content popularity, caching can be performed at item-level granularity. For instance, in [9] authors proposed the greedy replication algorithm, which has a computational complexity of NV^2C computations, where N is the number of content items, V the number of caches/nodes and under the assumption that all nodes have the same storage capacity C, where C is the maximum number of items to be stored in the cache. However, in reality content popularity changes from time to time and given the vast number of items circulated over the network and that an ISP's network consists of numerous nodes/caches, even such a polynomial complexity algorithm cannot be applied frequently enough or is extremely costly[2].

Consider for example a realistic content catalog of size $N = 10^9$ items and a domain of $V = 100$ nodes. We depict in Table 1 the amount of time required by the five most powerful non-distributed computers (according to `www.top500.org`) for the computation of a new replication assignment according to the greedy algorithm of [9], when each cache/node can hold 10% of the content catalog. We also depict the performance if the computation was parallelized over 10^5 typical Virtual Machines in a public or private cloud. We observe that even the most powerful computer requires more than 7 hours to compute a new replication assignment, whereas in the parallel execution approach, even if we neglect the communication delay and that the greedy algorithm is not

[1]Note that the proposed replication schemes are not limited for ISP deployed caches but can be used in any network of caches.

[2]Netflix, for example, performs a nightly push of the nationally (US) most popular movies to all its regional caches [10].

Table 1: Replica assignment computation time using the five most powerful computers and 10^5 parallel virtual machines ($V = 100, N = 10^9, C = 0.1N$)

Name	Proc. Cap. (*petaflops*)	Repl. time
Tianhe-2	33.86	≈7h
Titan	17.59	≈15h
Sequoia	17.13	≈16h
K Computer	10.51	≈26h
Mira	8.586	≈32h
10^5 cluster cores	$\approx 0.5 \cdot 10^{-3}$ per core	≈5h

fully parallelizable, more than 5 hours would be required. This implies that the problem complexity is substantial, and *the time required to calculate a new cache assignment matches or even exceeds the time scales dictated by the dynamics of content popularity*. This issue is further amplified by fragmentation of items into equally sized chunks, which is a requirement of many replication mechanisms, such as [11][12]. Thus, novel replication schemes of lower complexity are required.

Here, we propose the alternative of *content aggregation through clustering* to reduce the complexity of replication and thus enabling frequent cache updates according to popularity variations. Clustering is a machine learning technique [13], which groups items into clusters based on a certain similarity metric. In our context, item clustering significantly reduces the input size (dimension) of the replication problem. By selecting the number of clusters one may finely tune replication complexity. A cluster of content items is treated as a single item of certain attributes, and replication decisions are taken for the whole cluster as being one item.

Whereas existing works have demonstrated the potential of clustering to reduce complexity [14][15], *this is the first work that provides an efficient method to calculate the optimal number of clusters*. In detail, our original contributions are as follows.

- We model the impact of computational complexity of the underlying replication scheme on the overall network performance under content popularity dynamics.
- We characterize the tradeoff between the time required to calculate a new replication configuration and the sub-optimality of the corresponding replication decisions.
- We propose an optimization-based approach to compute the optimal number of clusters to form, so that overall network traffic is minimized.
- We propose a replication-aware clustering scheme that takes into account the spatial diversity of content popularity.
- We compare the proposed clustering scheme against a replication-agnostic clustering scheme, as well as various item-level partially coordinated caching

schemes, assuming different levels of coordination (from fully coordinated to totally uncoordinated).

The rest of the paper is organized as follows. In Section 2 we survey related work, whereas in Section 3, we present the system architecture and identify the impact of content popularity dynamics on replication decisions. In Section 4, we introduce the replication-aware clustering scheme, as well as an optimization-based approach to compute the optimal number of clusters, and we describe the alternative of partially-coordinated caching mechanisms. We evaluate numerically the performance of those alternatives for realistic network topologies and traffic data, and we demonstrate that clustering enhances the robustness of replication to content popularity variations in Section 5. Finally, Section 6 concludes our study.

2. Related Work

2.1. Replication in static environments

The problem of optimal content replication and placement in a network of distributed caches has received significant interest lately [11][16]. It is an NP-hard problem [9][17], and as such, approximation algorithms of polynomial complexity have been proposed. In [18], authors model the cache assignment problem as a distributed selfish replication (DSR) game in the context of distributed replication groups (DRG). Under the DRG abstraction, nodes utilize their caches to replicate information items and make them available to local and remote users with the objective of minimizing the overall network traffic. The pairwise distance of the nodes in [18] (transfer cost between any two nodes) is assumed to be the same and thus no network characteristics are taken into account. In the context of DRG and under the same distance assumption of [18], a 2-approximation cache management algorithm is presented in [19]. In [11] the authors develop a cache management algorithm for maximizing the traffic volume served by the caches and hence for minimizing network bandwidth cost. They focus on a set of distributed caches, either connected directly, or via a parent node, and they formulate the content placement problem as a linear program to benchmark the globally optimal performance. In the popular area of information-centric networks, a set of offline cache planning and replica assignment algorithms are proposed in [20], whereas in [21] a distributed cache management architecture is presented that enables dynamic reassignment of content items to caches in order to minimize overall network traffic.

The above mentioned replication schemes rely on the assumption that the popularity of content items is either static or changes slowly. In reality, content popularity changes significantly over time [22][23][24] and the design of a content replication scheme that updates caches following closely those changes is a challenging management task, since replication is a time-consuming process.

2.2. Reducing the complexity of replication process

In order to reduce the computational complexity of the replication process, partially coordinated replication was introduced in [25]. Nodes use a fraction of their cache capacity to store content in a coordinated manner (CDN-like replication) and the remaining capacity is used to cache the locally most popular content. The authors in [25] provide a method to optimally provision the storage capacity to be used for coordinated caching in each node, so as to balance the tradeoff between network performance and the provisioning/computational cost. Here, in order to reduce the complexity of the underlying replication schemes, we follow a totally different approach by aggregating content through clustering.

Clustering of web content based on popularity and replication at cluster-level was first considered in [14], where the validity of such an approach was demonstrated through extensive numerical evaluations of existing clustering and replication algorithms. In particular, the k-split algorithm of [26] was used to group together similar contents so as to minimize the maximum intra-cluster distance. In [27], the authors use clustering to reduce the complexity of the cache placement problem. Clustering for replication purposes has also been considered in the context of grid computing in [28]. Items are clustered together whenever they are frequently accessed by the same process within a small period of time. The problem of clustering is cast as a graph partition problem, and a greedy algorithm is proposed. Once the clusters have been determined, the problem is posed as an integer linear programming (ILP), which is solved numerically.

In [29] authors apply clustering to partition a network domain in sub-domains in an attempt to enable efficient hash routing techniques in the area of Information-Centric Networks. The aforementioned approaches are complementary to this work, given that our scheme relies on aggregating content items, neither users nor network nodes.

2.3. Benefits and risks of clustering for replication purposes

In general, clustering of content based on certain similarity metrics leads to a more succinct but less accurate representation of the system state compared to the fine-grained but time-consuming item-level replication. On the other hand, by reducing the problem size and thus its computational requirements, content replication at cluster-level can be applied more often, and hence content popularity dynamics can be tracked more accurately. Particularly, using a larger number of clusters (*i.e.,* clusters with a small number of items) ensures that only very similar contents are clustered together, the average cluster size and its variance become smaller, and hence the loss of treating clusters as a single item is reduced. However, a large number of clusters leads to a more complex optimization problem with larger computational complexity, and hence a less accurate tracking of content popularity dynamics. Thus, deriving the optimal clusters in terms of size and contents is a challenging task that needs to address this inherent tradeoff.

A first approach of clustered content replication for hierarchical cache networks was presented in [15], where cluster-level replication was compared against

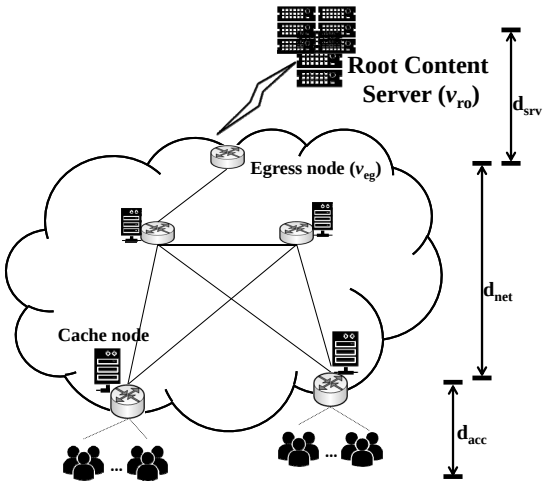

Figure 1: Architecture of the ISP-managed Content Delivery Network.

traditional replication schemes. However, no methodology to compute the optimal number of clusters was provided, which is the main contribution of this article. Besides, similarly to [18], we assumed in [15] that the pairwise distance of the nodes is the same, whereas here a generic network topology is considered. Additionally, in this work, the actual computational complexity of the replication scheme (at item or cluster-level) and the temporal and spatial variations of the content popularity are incorporated in the decision making. The proposed cluster-level replication scheme is compared against a broader set of item-level replication schemes (partially coordinated caching schemes) and the k-split clustering scheme presented in [14]. The k-split clustering scheme is replication-agnostic, and its objective during the formation of clusters is the selection of a set of representatives so that a loss function, *e.g.,* average distance of clustered items from the closest representative, is minimized.

3. System Model and Problem Formulation

3.1. System Model

We consider the interplay of content clustering and replication in a network of arbitrary topology, as the one depicted in Figure 1, which can be represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Let $\mathcal{V}$ denote the set of cache enabled routers/nodes and $\mathcal{E}$ the set of communication links connecting them. We use the calligraphic letters to denote sets and capitals for cardinality (*e.g.,* $|\mathcal{V}| = V$).

We are interested in the network of a single administrative domain, where a set of routers with storage capabilities serve requests for content from users. Node v_{ro} corresponds to the root content server where all content items are

stored. The root content server is an abstraction of multiple origin servers[3]. Additionally, we denote by v_{eg} the egress node through which a request is forwarded to the origin server that lies outside the administrative domain. We also assume that all nodes use the same egress node to access the root server, and each node $v \in \mathcal{V}$ has a storage capacity of C_v bytes.

Let $\mathcal{N}$ denote a given fixed set of N content items that have to be delivered over the network, and let s_n the size (in bits) of item n. Content requests are generated by users attached to network nodes according to their popularity. Access requests that cannot be satisfied locally trigger the transfer of the requested item from a remote node or from the root server. Throughout the paper we assume that the underlying content delivery mechanism always directs requests to the closest replica according to a cost metric. Thus, a request for item n generated at node i, incurs an aggregate traffic cost equal to $s_n \cdot h_{ij}$, if served by node j. Parameter h_{ij} captures the network cost per bit of transferred content from node j to node $i \neq j$, possibly in multihop fashion. In this work, we consider core network traffic as the performance metric of interest and hence h_{ij} captures the per bit required amount of network resources according to the shortest path from node j to node i. Latency is a second important aspect, which as depicted in Figure 1 depends on where the content is retrieved from.

A summary of the system model notations as well as the additional notations used in the Evaluation section is provided in Table 2.

3.2. Cache Management System Architecture

In this section, we briefly present a cache management architecture which, given a high-level optimization objective decides on the placement of the item/clusters in the caches of the network. We adopt a system architecture similar to the one applied in CDNs. In particular, we assume that distributed cache managers are assigned at each cache node of the network. Each cache manager monitors and reports content popularity fluctuations of the managed node to a central entity. The latter is responsible for acquiring all the necessary information such as request rates, popularity/locality of information items, current cache configurations and network topology, and it performs the aggregation of the items in clusters and the computation of placement of items at nodes/caches of the network.

Alternatively in a less coordinated approach (*i.e.,* [25]), managers could base their caching decisions only on a local view of the users demand, in an attempt to minimize the communication and computational complexity by caching the locally most popular content. Generally, the underlying cache management system architecture could be designed according to the used replication algorithm and vice versa, resulting a centralized architecture when traditional offline replication algorithms are applied, or to a distributed autonomic one when approaches like the ones presented in [21] are used.

[3]This assumption is made only for ease of presentation. Exactly the same analysis holds when more servers are present.

Table 2: Summary of the system model notations.

C_v	Storage capacity of cache v.
$\mathcal{E}$	Set of E links interconnecting the caches.
$\mathcal{M}$	Set of M clusters.
$\mathcal{N}$	Set of N content items.
$\mathcal{R}$	Set of replication algorithms ($\rho \in \mathcal{R}$ a given alg.).
S	Size of a cluster in number of content items.
$T_{\text{rep}}(\rho)$	computations and realization time of replication alg. ρ.
T_{obs}	duration of the observation period.
$\mathcal{V}$	Set of V cache enabled nodes.
a	Popularity alteration factor.
c	Proportion of the coordinated caching [25].
d_{acc}	Delay between a user and the network node that is attached.
d_{net}	Delay between two peer nodes in the network.
d_{srv}	Delay between egress node and root content server.
h_{ij}	Per bit traffic cost for fetching an item from cache j to node i.
r_v^n	Total number of requests for item n at node v.
s_n	Size in bits of item n.
t	fraction of the replication computation time (*i.e.*, $t = T_{\text{rep}}(\rho)/T_{\text{obs}}$).
v_{ro}	Root content server.
v_{eg}	Egress node towards content server.
z	Exponent of popularity distribution.

3.3. Impact of content popularity dynamics on replication

Content popularity captures the expected number of requests within a given period. Let r_v^n denote the estimated aggregate incoming request rate (in requests per unit of time) at node v for item n. Thus, vector $\boldsymbol{r}_v = \{r_v^1, \ldots, r_v^N\}$ is an estimate of the actual request rate based on observed, historical content access data within an appropriately chosen time window. This estimate is used as a prediction for the future number of requests addressed to each node. This estimation can be performed with an approach similar to [30], using an exponential moving average function in each measurement window, thus enabling the distributed managers described previously to monitor content popularity dynamics. This prediction could be enhanced with information from other sources like other ISPs, as well as CDNI [31].

We approximate item popularity through a Zipf distribution of exponent z, since it has been shown that file popularity in the Internet follows this distribution [22][32][33][34]. Generally, the popularity of each content item may differ from place to place, a phenomenon that is referred to as locality of interest [35] (*spatial skew* in [36]). In our model, this is captured through a localized request generation model, where aggregate request pattern $\boldsymbol{r}_v$ is different across regions represented by nodes $v \in \mathcal{V}$ in the network. We assume V different

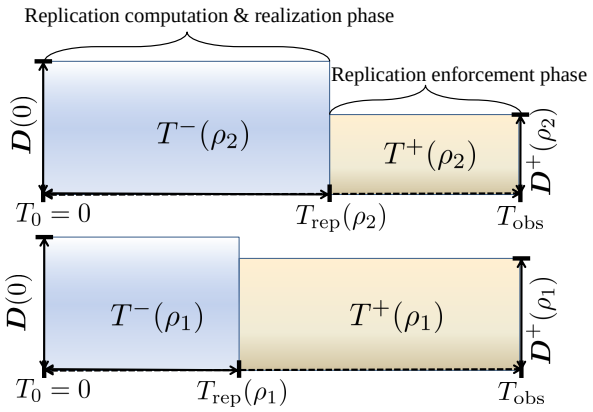

Figure 2: The two phases of content replication. ρ_2 is a replication policy of higher complexity that guarantees lower network traffic (*e.g.,* algorithm presented in [12]) than replication policy ρ_1 (*e.g.,* greedy algorithm in [9]). $\boldsymbol{D}(0)$ and $\boldsymbol{D}(\rho)$ are the per unit of time overall network traffic given by Eq. (1) and Eq.(2) accordingly.

regions, each served by a network router/node. All regions are characterized by the same value for the Zipf distribution exponent which captures the global popularity of items, but in each region the ranking/order of the items within the Zipf distribution is different, which captures the locality of interest.

Content popularity is changing over time [22][23][24], and based on the request rates observed within a time window of duration T_{obs}, the manager/central entity of the network may detect substantial variations in popularity, say at time $T_0 = 0$. We model this modification of request vectors through a popularity alteration factor a in the sense that the ranking of the items within the Zipf popularity distribution at each node is altered by a factor a; *i.e.,* $a \cdot N$ items have a different ranking at T_0 than the most recent one that was used to derive the current replication decision. Then, a reassignment of items on the cache nodes of the network has to be applied so as to minimize the expected network traffic for the upcoming period.

A key decision concerns the selection of the replication algorithm/scheme $\rho \in \mathcal{R}$, where $\mathcal{R}$ is the set of possible replication algorithms. In the end, any replication algorithm $\rho \in \mathcal{R}$ results in a feasible replication decision, *i.e.,* a placement of items in caches that respects cache capacity constraints. However, replication algorithms are characterized by different complexity and performance. Let computation and realization of the new replication decision of algorithm ρ, requires an amount of time equal to $T_{\text{rep}}(\rho)$ time units. Then, if we assume that the content popularity is the same throughout an *"observation period"* T_{obs}, we may define the two main phases depicted in Figure 2. Any request generated in the first phase will be served according to the current replica configuration, which is based on outdated information. Once the cache contents have been updated, in the second phase all the new requests will be served accordingly.

In this context, the optimal content replication strategy has to be decided so as to minimize the overall network traffic over T_{obs}. Let $\mathcal{A}_j^{n-}$ denote the set of nodes retrieving item n through its replica at node j according to the initial (outdated) cache assignment. We call this phase as pre-cache update period. Also, let $\mathcal{A}_j^{n+}(\rho)$ be the corresponding set according to updated cache contents determined by ρ. We call this phase as post-cache update period. Then, the overall network traffic during the first phase according to strategy ρ is given by

$$T^{-}(\rho) = T_{\text{rep}}(\rho)\boldsymbol{D}(0) = T_{\text{rep}}(\rho)\sum_{n=1}^{N}\sum_{j\in\mathcal{V}}\sum_{i\in\mathcal{A}_j^{n-}} r_i^n s_n h_{ij}\,. \tag{1}$$

Notice that the duration of this first period/phase, is determined by the replication algorithm. In a similar way, we define the expected overall network traffic due to requests generated within the second phase as,

$$T^{+}(\rho) = (T_{\text{obs}} - T_{\text{rep}}(\rho))\boldsymbol{D}^{+}(\rho) = (T_{\text{obs}} - T_{\text{rep}}(\rho))\sum_{n=1}^{N}\sum_{j\in\mathcal{V}}\sum_{i\in\mathcal{A}_j^{n+}(\rho)} r_i^n s_n h_{ij}\,, \tag{2}$$

where $\boldsymbol{D}(0)$ and $\boldsymbol{D}^{+}(\rho)$ are the total network traffic per unit of time before and after the cache update respectively. Our objective is to minimize the overall core network traffic, defined as $L(\rho)$, which takes into account the delayed cache updates due to the computational complexity of the underlying replication scheme ρ. Formally, this can be expressed as the following optimization problem

$$\min_{\rho\in\mathcal{R}} L(\rho) = T^{-}(\rho) + T^{+}(\rho)\,. \tag{3}$$

$L(\cdot)$ corresponds to the total area of the rectangles depicted in Figure 2, where the area of each phase is given by Eq.(1) and Eq.(2). Intuitively the optimal replication of content items would minimize the overall network traffic during the second phase, *i.e.,* the height of the second rectangle. However, given that content replication is an NP-hard problem [9], a significant amount of time T_{rep} would be required to calculate the optimal solution and update caches, increasing hence the period that caches remain outdated (*i.e.,* duration of pre-cache phase). This tradeoff between performance (*i.e.,* network traffic) and complexity (duration of pre-cache period) can be addressed through the selection of the replication algorithm/scheme.

Instead of exploring the performance of different replication heuristic algorithms, we propose alternative approaches that explicitly address the aforementioned complexity-optimality tradeoff in a managed and controllable manner. In particular, we propose schemes that reduce the dimension of the problem by controlling the number of clusters M, which determines the computational load and hence the relative size of the pre- and post-cache update intervals. Thus, in our clustering setting all the above formulas still hold, but we need to replace ρ by M.

3.4. Replication algorithm

We assume that the 2-approximation greedy placement algorithm of [9] and [20] is used by the network management entity as the replica placement algorithm. The algorithm initially assumes empty caches, and at each iteration it replicates the item to the cache that yields the maximum traffic gain. In particular, in the first round the algorithm evaluates the traffic gain if each of the N items is cached in each of the V caches. Out of the NV available options, the item-cache pair that yields the maximum traffic gain is selected. Given the previous step decision, in the second round, an additional item-cache pair has to be selected, namely the one that yields the maximum core traffic savings. The greedy algorithm is repeated until all the available storage capacity has been used. The greedy replication algorithm has a computational complexity of $NV \sum_{v=1}^{V} C_v = NV^2C$ computations, assuming that all nodes have the same storage capacity ($C_v = C$, $\forall v \in \mathcal{V}$).

4. Content Replication Schemes of Tunable Complexity

In this section, we present two alternative approaches to tackle the optimization problem in Eq. (3). In particular, we describe how content clustering and partially coordinated replication enable a fine-grained control of the computational complexity of replication and eventually of network traffic.

4.1. Cluster-level Replication

Clustering of N content items into M clusters (*i.e.,* groups), with $M \ll N$, facilitates replication at cluster-level, which can substantially reduce the dimensionality of the replication problem and consequently the replication complexity. The formation of content clusters results in suboptimal replication decisions, when compared to item-level replication, since content items in a cluster are treated as a single item when replicated. Next, we present the two causes of performance loss as compared to item-level replication.

- **Diversity loss: Miss-classification of items due to spatial variation of popularity and coarse-grained replication.** This loss results from two main characteristics of the specific application scenario, namely spatial variation of content popularity [35][36], and the diversity of the traffic cost h_{ij}. Content clustering is performed according to the content popularity $\boldsymbol{r}_v$ across all caches. Thus, any two items assigned to the same cluster should always be cached together. There may be places (caches) though, that it would be preferable to split a cluster, so as to cache only a part of it along with other content items. In order to demonstrate the impact of spatial variation of popularity, consider a set of 3 items $\{1, 2, 3\}$ and two locations i and j with corresponding content request rates $r_i^1 > r_i^2 > r_i^3$ and $r_j^1 > r_j^3 > r_j^2$. If $r_i^1 = r_j^1$, $r_i^2 - r_i^3 > r_j^3 - r_j^2$, and all traffic costs are equal, cluster $\{1, 2\}$ would be formed and stored in both caches. Instead, in item-level caching items $1, 2$ would be cached at

cache i and items $1, 3$ at cache j. The same phenomenon could arise under unequal costs, even if content request patterns are identical from place to place. Diversity loss is generally decreasing in the number of clusters M.

- **Slack loss: Non-integral multiple of cluster size.** This loss arises when part of the capacity of a cache remains unallocated, since no uncached cluster fits there. If item-level caching were applied instead, a subset of the items of an uncached cluster would have been cached. The unallocated capacity at cache j resulting from cluster-level caching, is upper-bounded by the maximum size of a cluster. In particular, a tighter bound holds, namely unallocated capacity is smaller than the smallest uncached cluster. The corresponding loss is equal to the traffic cost of retrieving those items from the closest replica. This loss could be further reduced by assuming partial cluster caching (caching a portion of a cluster), but this is left for future investigation. Such a scheme would require additional effort for the computation of the right cluster(s) that should be partially cached.

4.1.1. ***Replication-aware clustering***

The problem of clustering a set of N items in M clusters is an integer programming one and it is NP-hard [37]. Here, we derive a low-complexity clustering scheme that addresses the aforementioned replication-related issues.

In order to minimize the unused capacity (*i.e.,* slack loss), we suggest that equally sized clusters should be formed. In addition, we select cluster size $S = \sum_{n\in\mathcal{N}} s_n/M$ to be a common divisor of all cache capacities, which translates into the following set of additional constraints

$$\frac{C_v}{S} \in \mathbb{Z}^+ (\text{the set of positive integers}) \; \forall v \in \mathcal{V}\,. \tag{4}$$

This ensures that the resulting clusters fit perfectly to caches. Any approximate divisor can also be used, resulting only to a limited amount of storage being unexploited. Thus, the set of feasible numbers of clusters, M is restricted. It has been shown that introducing specific constraints in cluster cardinality improves performance of heuristic clustering approaches [38][39]. Our extensive numerical evaluations verify that this holds in our scenario as well.

On the other hand, increased traffic stems from cache misses, due to content aggregation (*i.e.,* diversity loss). This, is an inherent characteristic of clustering and hence cannot be avoided. We devise a clustering approach that for a given number of clusters M pursues to minimize the corresponding performance loss.

Initially, a representative similarity index I (metric) has to be derived to drive clustering decisions. In order to address the spatial variations of request rates $\boldsymbol{r}_v$ we calculate content similarity of any two items n_1 and n_2 according to the inverse of pairwise Euclidean distance of their overall request rate vectors,

i.e.,

$$I_{\text{eucl}}(n_1, n_2) = \left(\sqrt{\sum_{v \in \mathcal{V}} (r_v^{n_1} - r_v^{n_2})^2} \right)^{-1} . \quad (5)$$

Euclidean distance of content popularity transforms the Euclidean space of popularity into a metric space. The Euclidean distance of the popularity vectors quantifies how close (similar) are the spatial request patterns of two items over the network caches. This metric can be then used to construct the clusters according to the greedy clustering scheme presented in Appendix A. Initially, a cluster that contains the two most similar items is formed. Then, more items are added to the cluster one by one until the selected size of cluster is reached. At each iteration of the algorithm, the most similar item to those already in the cluster is chosen to be included in it. Once an item is allocated to a cluster, it is excluded from the candidate items set. Thus, clusters are created and filled until all items have been assigned to a cluster. In the Evaluation section we refer to this clustering scheme as *Euclidean distance clustering*.

4.1.2. ***Approximation of the optimal number of clusters***

From the discussion above, it becomes clear that clustering introduces a new degree of freedom to tackle Eq.(3), namely the number of clusters M, identically to replication strategy ρ. Parameter M enables us to trade between *i*) the time required to calculate a cache assignment, which determines the pre-cache update interval in Eq.(1) and *ii*) increased traffic load due to suboptimal placement of content according to Eq.(2). Next, we propose a methodology to estimate the optimal number of clusters, given that the proposed clustering scheme and greedy replication are applied. This requires deriving a closed form expression of $L(M) = T^-(M) + T^+(M)$.

T_{rep} in Eq.(1) is a function of M. For equal clusters of size S the computational complexity of cluster-level replication is MV^2C/S computations, which is smaller by a factor of $(1/S)^2$ compared to item-level greedy replication [9]. Given also that $S = N/M$ we have,

$$T_{\text{rep}}(M) = \gamma \frac{V^2 C}{N} M^2, \quad (6)$$

where multiplicative factor γ captures the average time required for the execution of a single replication computation and can be easily calculated for a specific system. For reference consider the values presented in Table 1, where γ is the time that each system requires for the computation of a single cluster assignment (*i.e.,* usually a multiple of the inverse processing capacity). Notice that given the placement of items in the caches and the new request rate vector, the total network traffic per unit of time before the cache update, $\boldsymbol{D}(0)$, can be numerically calculated and does not depend on M. Eventually $T^-(M)$ of Eq.(1) can be expressed as a function of the number of clusters M (*i.e.,* due to T_{rep}).

Similarly, in Eq.(2) post-cache period network traffic is a function of M,

i.e., $T^+(M) = (T_{\text{obs}} - T_{\text{rep}}(M))\boldsymbol{D}^+(M)$. However, analytically calculating the exact impact of M on $\boldsymbol{D}^+(M)$ (the summation term of Eq.(2)) would require solving the replication problem for different values of M and selecting the one that minimizes Eq.(3). This approach is impractical, and an estimate of the overall network traffic $T^+(M)$ under clustered replication has to be derived. Since slack loss can be easily avoided be proper selection of cluster size, loss is introduced only due to diversity. Then, given that content popularity follows a Zipf distribution, the total traffic per unit of time over M, $\boldsymbol{D}^+(M)$, can be approximated by a power function,*i.e.,*

$$\boldsymbol{D}^+(M) = \beta M^{-\lambda}\,. \tag{7}$$

This approximation comes naturally since as Eq.(2) suggests $\boldsymbol{D}^+(M)$ is a linear combination of request rates $\boldsymbol{r}_v$ which follow a power law[4]. It is also supported by our extensive numerical results. In the example of Figure 3, we depict a simulated instance and its approximation power function. However, the exact values of β and λ depend on numerous system parameters.

A straightforward way to approximate $D^+(M)$ is through its evaluation in at least two points. One such point can be derived by solving the cluster-level replication problem for a small feasible value of M, say M_1. For example, we may select the cluster size so that each cache can store only one cluster $S = C$. An additional point that is easy to calculate corresponds to the solution of the Linear Programming (LP) relaxation of the item-level replication problem which provides a lower bound of $T^+(N)$ (*i.e.,* each cluster consists of one item) [11][16] or the outcome of any of the proposed heuristic item-level replication algoirthms [17]. Thus, β and λ can be calculated as the numerical solution of the following system of equations

$$\beta M_1^{-\lambda} = D^+(M_1), \tag{8}$$

$$\beta N^{-\lambda} = D^+(N). \tag{9}$$

If additional points can be derived, curve fitting could be applied instead, by using any of the known interpolation mechanisms presented in [40]. Notice that the schemes in [40] are also applicable when the popularity distribution is not even exponential. Estimation of those parameters could be also assisted by historical data, which is beyond the scope of this work.

Given the derived expressions of Eq.(6) and Eq.(7), one may numerically solve the following unconstrained optimization problem to approximate the op-

[4]Similar approximations can be applied for other popularity distributions, *e.g.,* exponential.

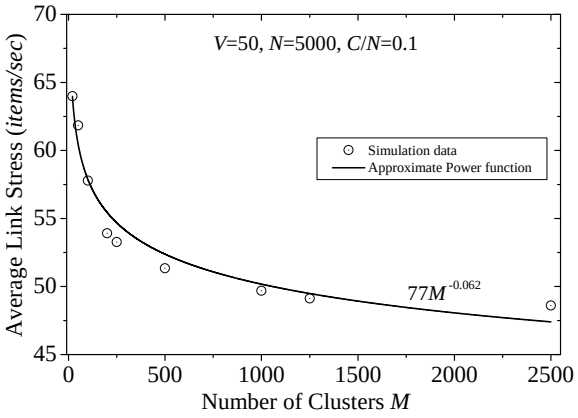

Figure 3: The performance of the cluster-level replication scheme for a given network setup is approximated by the power function $77M^{-0.062}$.

timal number of clusters

$$\begin{aligned}\min_M L(M) =& T_{\text{rep}}(M)\boldsymbol{D}(0) + (T_{\text{obs}} - T_{\text{rep}}(M))\boldsymbol{D}^+(M) \\ =& \gamma\frac{V^2C}{N}M^2\boldsymbol{D}(0) + (T_{\text{obs}} - \gamma\frac{V^2C}{N}M^2)\beta M^{-\lambda}\,.\end{aligned} \tag{10}$$

Since the derived solution may be non-integer, rounding to the nearest feasible M has to be applied.

Notice that an alternative approach to calculate the optimal number of clusters would be to perform a binary search over the set of feasible values according to Eq.(4). By exploiting our knowledge on $T_{\text{rep}}(M)$ and $T^+(M)$, a nearly optimum value of M can be derived by calculating only a limited number of cluster-level replication instances, typically three to five instances.

4.2. Partially Coordinated Content Replication

Another method to reduce the computational complexity of the greedy replica placement algorithm is to apply partially-coordinated replication. In this case, nodes use only a fraction $0 \leq c \leq 1$ of their cache capacity to store content in a coordinated manner (*i.e.,* item-level replication), and the rest of the capacity is used to cache the locally most popular items. This model captures the fact that user preferences in a particular region is likely to be a mix of globally-popular content and regionally-popular content, as shown in [41]. Two extreme cases of interest are the fully coordinated item-level replication for $c = 1$, and the uncoordinated replication for $c = 0$, where each node caches the locally most popular items regardless of the caching decisions of the other nodes. In the latter case, the computational complexity is negligible, since each node independently adapts to the new request pattern by updating its cache with the locally most popular items. Note also that this case is also known as Least Frequently Used,

Table 3: Evaluated Replication Schemes

Replication Scheme	**Comments**
Euclidean distance clustering	Proposed cluster-level replication Section 4.1.1
k-split clustering	Cluster-level repl. using clustering alg. from [14]
Item-level	Greedy Replication with $c = 1$ in Section 4.2
Coordinated 25%	Greedy Replication with $c = 0.25$ in Section 4.2
Coordinated 50%	Greedy Replication with $c = 0.5$ in Section 4.2
Coordinated 75%	Greedy Replication with $c = 0.75$ in Section 4.2
Uncoordinated	Cache the locally most popular items (*i.e.,* LFU)

LFU (within a given time window) replacement strategy, usually employed on push-based caching schemes.

Let each node v contribute $c \cdot C_v$ of its storage capacity for coordinated replication according to the item-level greedy replication algorithm. Coordination level c linearly reduces computational complexity of replication, leading to a smaller scale problem instance of controllable size. Indicatively, for $c = 0.25$ the computational complexity/delay assuming the greedy replication algorithm is $1 - c = 75\%$ smaller than assuming fully coordinated replication.

A similar approach was introduced in [25], in the context of Information-Centric Networking. In contrast to our work, in [25] coordination targets the selection of where each item should be uniquely cached (no replication) with the objective of minimizing latency and under the assumption that the intra-domain delay between peer caches is the same for every pair of nodes.

5. Numerical Evaluation

5.1. Evaluation setup

In this section, we use a custom-built discrete event simulator to evaluate the performance of the proposed replication scheme, namely the cluster-level replication *"Euclidean dist. clustering"* presented in Section 4.1.1, along with the partial coordinated replication scheme, for $c \in \{0, 0.25, 0.5, 0.75, 1\}$, and the generic clustering scheme *"k-split clustering"* proposed in [14]. The k-split algorithm clusters the content items into k clusters and its objective is to minimize the maximum intra-cluster distance, but it is replication-agnostic. Table 3 summarizes the replication schemes used in the Evaluation section.

Throughout this section, we assume that all items are of equal size and all nodes have the same storage capacity ($C_v = C$, $\forall v \in \mathcal{V}$). We normalize the size of each item to one unit with respect to node's storage capacity ($s_n = s = 1$, $\forall n \in \mathcal{N}$) and hence each node can hold up to C different unit sized items. Note that fragmentation of items into equally sized chunks is a requirement of many replication mechanisms, *e.g.,* [11][12]. Content segmentation is also present in various content distribution systems, such as BitTorrent, which implies that our equally sized items assumption is reasonable. Regarding the network topology, we use a topology with $V = 50$ nodes from the Internet Topology Zoo dataset [42].

Although our objective here is to minimize the core network traffic load, user-perceived latency is also an important performance metric. For this reason, we need also to capture the incurred latency, which depends on which cache hosts each requested item. We denote by d_{ij} the latency between two neighbouring nodes $i, j \in \mathcal{V}$ and $(i, j) \in \mathcal{E}$. Typical latency values are ≈ 10 ms for d_{acc} (see Figure 1) in cable and ADSL access networks [25]. The latency between caches in the same administrative domain, d_{net}, typically ranges from a few up to 50 ms more than d_{acc}, depending on the geographical coverage of the network. Finally, d_{srv} typically ranges from $50 - 100$ ms. Throughout our simulations, we use the following values:

$$d_{ij} = \begin{cases} d_{\text{acc}} = 10\,\text{ms}, \\ d_{\text{net}} \in [1,\ 50]\,\text{ms}, \\ d_{\text{srv}} = 100\,\text{ms}\,. \end{cases}$$

We also assume that at each node a total of 100 requests per second are generated. Thus, the request rate for each item at each node varies from 0-100 reqs/sec depending on item popularity and ranking. We consider a scenario where $N = 5,000$ content items have to be replicated (and clustered). Recent measurement-based studies indicate that a small number of items often account for a large portion of traffic, especially for users located in certain areas (*e.g.,* a university campus [43]), or embedded in a social network [44]. This advocates that a small portion of the population of the items available in the network is actually requested. For instance in [22] authors found through the analysis of a video on demand dataset that the 10% of the most popular videos attracted more than 80% of the views. Additionally, the use of Zipf distributions for the items' popularity (also found in the dataset of [22]) implies that $5,000$ items may account for the $\approx 90\%$ of the total demand considering an information space of size of 10^6 items approximately. On these grounds, our choice for the size of the items' population and their popularity distribution can be considered fair. The rest of the items are served directly from the root content server and are not considered in the replication and the clustering process, as suggested in [22].

For the evaluation of the proposed replication schemes, we initially assume that for a period of 12 hours, the request pattern (Zipf exponent and ranking) at each node is unchanged or differently the manager of the network performs the estimation of request patterns in 12 hours intervals. According to the observed request pattern each one of the examined replication schemes assigns replicas to the caches of the network. We call this period as the *"initialization period"*. At the end of this period we assume that the ranking of the items at each node has changed by a popularity alteration factor a. Then, at the beginning of the *"observation period"* T_{obs}, which is also set equal to 12 hours, each one of the examined replication schemes initializes the reassignment of the items in the caches.

Let $t = T_{\text{rep}}/T_{\text{obs}}$ denote the fraction of the time for the computation of the new replication assignment over the whole observation time. We also refer to t as the *replica computation latency*. The computation latency t used in the fol-

lowing experiments refers to the item-level greedy replication scheme when the storage capacity of each node is 10% of the total item population. Accordingly, the computation latency of the rest of the examined replication schemes corresponds to a fraction of factor t (see Section 4.1.2 for the cluster-based replication schemes and Section 4.2 for the partially coordinated replication schemes). Note here that the computation latency used throughout the evaluation section is a normalized value based on the computational times presented in Table 1, and not on the actual used values for the items' population used in the simulator.

Our evaluation is based on the following metrics:

- The *Average Link Stress* (in items/sec) is the mean number of items that traverse each link of the network per second. This is our main metric for the core network traffic load.
- The *Cache Hit Ratio* is the ratio of the content requests that were served by the cache network, *i.e.,* they found the requested item cached within the domain and not at the root server, over the total number of requests issued during the observation period.
- The *Average Retrieval Latency* (in msec) is the mean latency for the retrieval of an item by a user during the observation period and is a user perceived QoS metric.
- The *Maximum Link Stress* (in items/sec) is the maximum number of items that traverse the most constrained/congested link of the network. This metric along with the Average Link Stress metric are indicative of the load balancing capabilities of each scheme.

For better readability of the results, we also depict the *Link Stress gain/loss against item-level replication*, which captures the gain or the loss of of a specific replication scheme over the benchmark item-level replication scheme.

In cluster-level replication, the clustering process requires a non negligible amount of offline computations. This implies that clustering is a process that should be executed in a different time scale compared to the replication process, since otherwise it might compromise the low complexity computation process of the new content replication scheme. Here, we assume that the aggregation of items into clusters occurs only once and never changes both during the initialization and the observation periods. We additionally assume, that the replication process is executed at the end of the initialization period, based on the content popularity observed by the cache managers during this period, whereas clustering was done based on the popularity observed at some time before the initialization period and does not change afterwards. This allows the cluster-based replication schemes to compute the new replica assignment in a very small fraction of time compared to the item-level schemes.

We evaluate the potential of this approach in Figure 4. Whereas item-level replication spends a significant amount of time to calculate and update cache contents (*i.e.,* $T_{\text{rep}}(N) = 0.25T_{\text{obs}}$), in our cluster-based approach the updates are very fast(*i.e.,* $T_{\text{rep}}(M) = 0.01T_{\text{obs}}$). Thus, although the updated cache

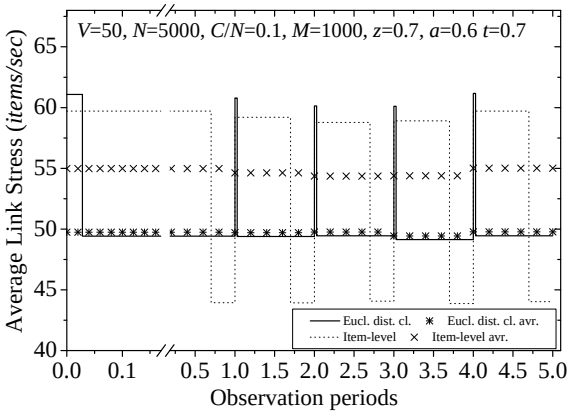

Figure 4: Evolution of core network traffic as content popularity changes over different observation periods: Cluster-level vs. item-level replication.

configuration results to higher traffic in the network, if compared to item-level replication, the average traffic over time is more than 10% lower.

5.2. Impact of number of clusters

Figure 5 depicts the impact of the number of clusters M on the performance of the considered schemes. Naturally, only the newly proposed clustering scheme and the k-split algorithm are affected by the number of clusters. The rest of the schemes are depicted for comparison purposes, since all of them perform replication at item-level and not at cluster-level.

From the comparison of the two clustering schemes (Figure 5(b)), we observe that the newly proposed replication-aware Euclidean clustering scheme, with equally sized clusters, performs between 22% and 52% better than the k-split clustering algorithm [14], depending on the size of the formed clusters. The k-split algorithm, despite having the same complexity with the proposed Euclidean distance clustering algorithm, generally forms clusters of different sizes that increases the Slack loss effect described in Section 4.1. Therefore, even when it manages to replicate all clusters within the domain (for the default cache size at each router), there are items that should be fetched from distant routers, which increases the retrieval latency (Figure 5(c)) and puts increased stress at the links of the network (Figure 5(b) and (d)).

From Figure 5(a) we also observe, that an increase in the number of clusters decreases the possible losses due to miss-classification of the items that arise from their spatial variations. On the other hand, increasing the number of clusters increases the complexity for the computation of the new replica assignments. Therefore, based on the observed network dynamics a network manager should weight the pros and cons of the selected clustering level before applying it to the network, *e.g.,* according to the approach provided in Section 4.1.2. In terms of network traffic, we observe that for any option above $M = 200$ clusters

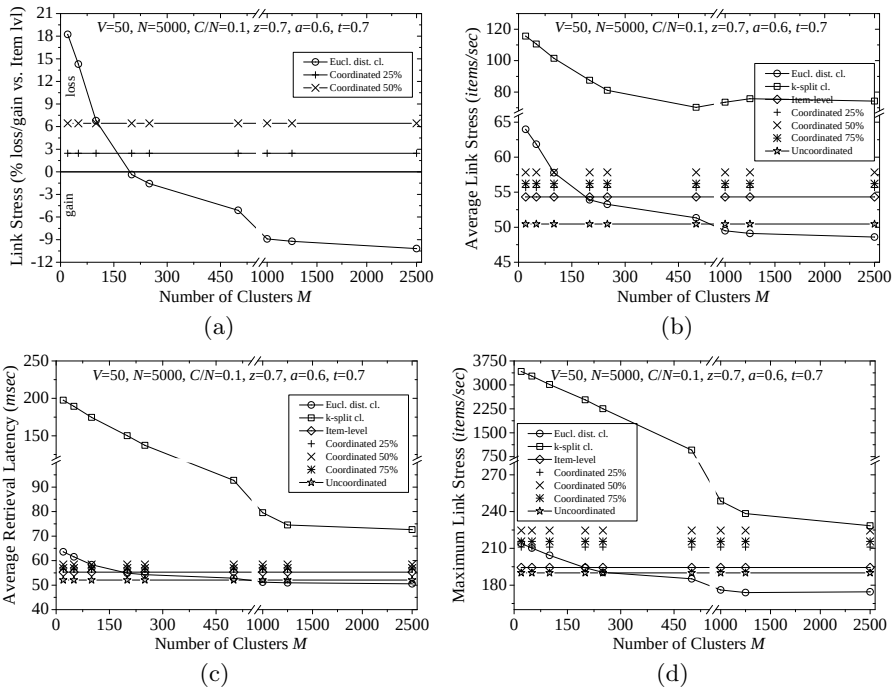

Figure 5: The impact of the total number of clusters formed in the performance of the examined schemes.

the proposed cluster-based replication scheme outperforms item-level replication (Figure 5(b)). The findings regarding average latency are similar. For example, an increase in the number of clusters from $M = 50$ to $M = 250$ decreases the average latency by 11%, whereas an increase from $M = 1000$ to $M = 2500$ decreases the latency only by 2% (Figure 5(c)).

For the system parameters used in this experiment the Cache Hit Ratio of every scheme is equal to 100%. This means, that even the k-split algorithm manages to replicate all items within the caches of the domain. Only for smaller cache capacities the cache hit ratio of all the replication schemes is less than 100%, as we show in the next section.

From Figure 5 it is obvious that a network manager could also apply a hierarchical clustering and replication scheme. Starting from an initial number of clusters, we may split them in each iteration until we reach a core network traffic that is smaller than a given threshold or than a competitive replication scheme. For instance, based on the selected default values for the various system parameters, we observe that a number of $M = 1000$ clusters is sufficient for our clustering scheme to significantly outperform item-level replication by almost 10% regarding Avg. Link Stress. We use this as the default value for the number of clusters in the rest of the Evaluation section.

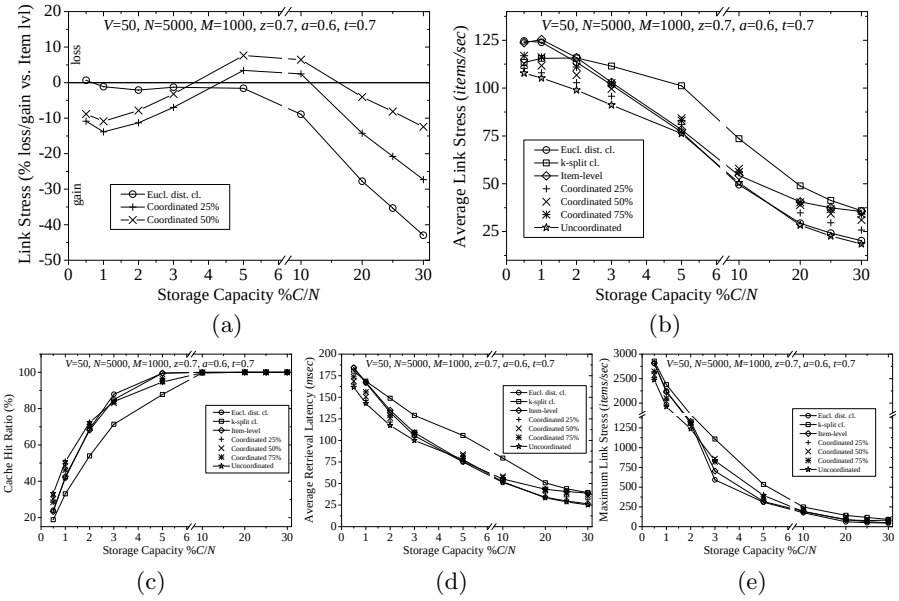

Figure 6: The impact of the cache capacity of each network router/node in the performance of the examined schemes.

5.3. Impact of cache capacity

Figure 6 depicts the impact of available cache capacity, expressed as the fraction of content items that can be stored in the cache of a node. We observe that for the selected number of clusters (*i.e.,* $M = 1000$) the proposed cluster-based replication performs better than item-level for almost all the evaluated cache sizes (Figure 6(a)). Of course when the storage capacity of each node is less than 10% of the item population, item-level replication still performs considerably well. In those scenarios, the incurred computation latency is not significant and any losses during the execution of the item-level replication algorithm (*i.e.,* T_{rep} interval) are reversible within the remaining period ($(1-t)T_{\text{obs}}$). On the other hand, when the storage capacity is larger than 10% cluster-based replication achieves up to 45% reduced traffic due to its considerably lower complexity. Notice that this occurs despite the possible clustering losses described in Section 4.1. Additionally, the partially coordinated schemes that compute the replica assignment in a fraction of time compared to item-level scheme, perform up to 30% better when the storage capacity of each node is large enough (Figure 6(a) and (b)).

In Figure 6(c), we depict the cache hit performance, *i.e.,* the portion of requests served by any of the caches, and not by the origin server. We observe that when $C/N \geq 10\%$ all the examined algorithms serve requested items inside the domain, eliminating thus the access to the origin server and its incurred delay. In those cases only the latency between the client issuing the request and the closest router/node holding a replica of the requested item is apparent. Therefore, their performance difference lies on the efficiency of their replication

scheme and the impact of the computation latency in its realization. Since in the rest of the performed experiments the cache capacity of each node is assumed equal to 10% of the items population and the cache hit ratio of the examined replication schemes is always $\approx 100\%$ we are not depicting the cache hit performance figures.

Also in Figure 6(e) we depict the max link stress metric. The results are in line with the average traffic/link stress performance findings, *i.e.,* the item-level scheme performs worse than the rest (apart the k-split) when the cache of each node can hold more than 10% of the content population. The most interesting finding is that the k-split scheme performs significantly worse than every other scheme, which supports our claim that specifically designed replication-aware clustering schemes are necessary. The same also holds and for the retrieval latency metric depicted in Figure 6(d).

As mentioned above, the k-split algorithm forms clusters of different sizes that might not fit in the cache of a node. Therefore, for small caching capacities (*i.e.,* $1\% \leq C/N \leq 5\%$) some relatively larger clusters that contain popular items cannot be cached within the domain and should be fetched from the distant content server, thus minimizing the cache hit ratio and increasing the delay and the network traffic. For that reason, the proposed Euclidean clustering scheme performs up to 35% better than the k-split algorithm regarding the average incurred latency (Figure 6(d)) and significantly better regarding both link stress metrics (Figure 6(b) and (e)). Only when the cache capacities of each node are very small (*i.e.,* $C/N \leq 1\%$) the k-split algorithm is slightly better regarding the stress metrics and this is mainly to the low hit ratio that every scheme performs, and the corresponding redirection of the requests to the root server. In such scenarios, the limited cache is better utilized by the k-split algorithm and the corresponding Slack loss is minimal.

5.4. Impact of computation latency

Figure 7 depicts the impact of computation latency t on the performance of the examined replication schemes. For the specific system parameters, the performance of the two clustering schemes are not affected by the computation latency, since even when $t = 1$ the cluster-level replication algorithms require less than 0.5 hours out of the 12 hours of the observation period to compute the new replication assignment. In particular, the item-level algorithm outperforms every other algorithm as long as $t < 0.4$ in terms of core network traffic (Figure 7(b)). From that point and beyond, the incurred computation latency is so large that the fine-grained replica assignment at item-level is outperformed by our cluster-based scheme. We also observe similar behaviour in terms of average retrieval latency (Figure 7(c)). In general, the partially coordinated schemes are linearly affected by the computation latency, *i.e.,* the larger the coordination factor c the larger the impact of the computation latency on the performance of each scheme.

From Figure 7 we observe that even if we assume that the impact of the computational complexity is zero ($t = 0$) the proposed clustering scheme performs only 12% worse than the item-level replication algorithm (leftmost points

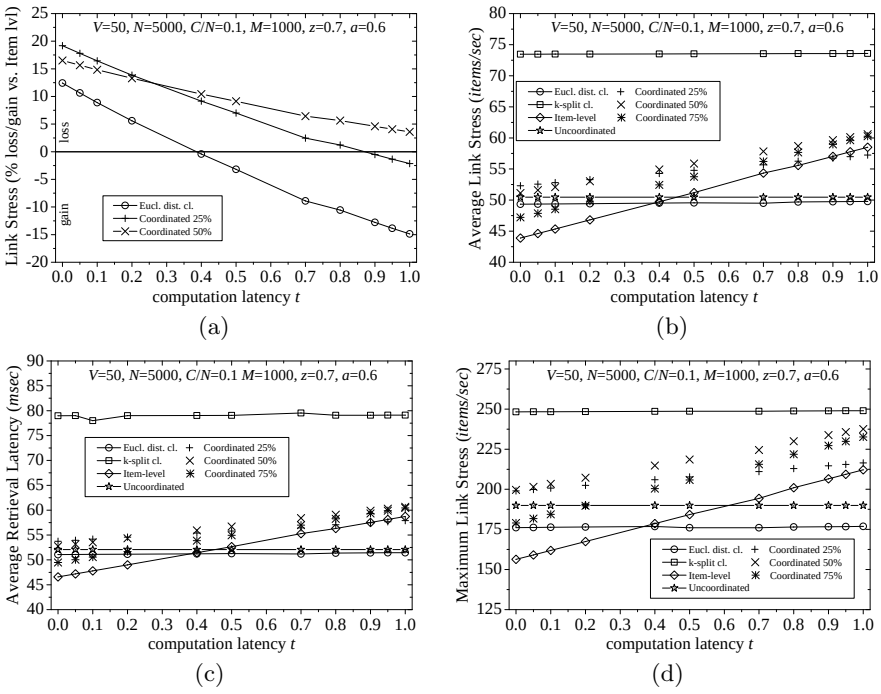

Figure 7: The impact of the computation latency t (*i.e.,* replication computational complexity) in the performance of the examined schemes.

in the performance plots). So, the tradeoff of performance over computational complexity is definitely against item-level replication, since in most cases it requires more than 96% additional computations in order to achieve a performance improvement of $\approx 10\%$.

Regarding the link stress metrics (Figure 7(b) and (d)) we observe that the uncoordinated scheme performs close enough to the proposed clustering scheme. The uncoordinated scheme, by caching the locally most popular items, intuitively tries to minimize the traffic/load that each node puts in the network. This is generally suboptimal, but for the specific simulation parameters performs significantly well (*i.e.,* mainly due to the small size and the connectivity of the used network topology).

5.5. Impact of content popularity

In the above scenarios we assumed a specific value for the Zipf exponent of the items' popularity. Measurement-based studies, such as [32], suggest that the Zipf exponent z for web traffic lies in the range of $0.64 - 0.84$, while other types of traffic (*e.g.,* P2P or video) may follow different popularity patterns [33][34]. For example, in [34] authors found that the distribution of the user access to video content is a Zipf-like with exponent parameter $z \approx 1.5$.

In Figure 8 a wider range of values for the Zipf distribution is examined. We observe that for small values of z our proposed clustering algorithm performs almost identical to the item-level replication scheme, and it outperforms every

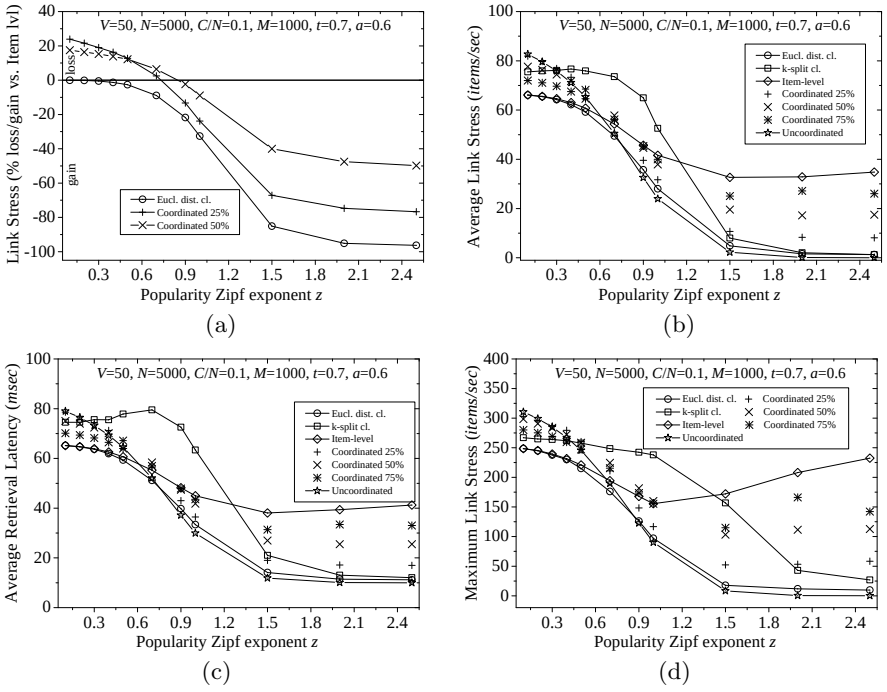

Figure 8: The impact of the content popularity distribution in the performance of the examined schemes.

other examined replication scheme (Figure 8(a) and (b)). For $z \leq 0.5$, the variation of content popularity is marginal and the outdated replica assignment (due to the computation latency described previously) cannot diminish the performance of the fine grained item-level caching. In other words, the losses during the computational period of time T_{rep} are negligible and can be saved after the exploitation of the new assignment, since also the old replica assignment can efficiently satisfy requests for content that is of significant popularity.

When $z \geq 0.7$, the newly proposed clustering scheme outperforms the item-level replication scheme by $20\% - 95\%$, whereas for $z > 1.5$ all replication schemes outperform item-level replication.

Regarding our main metric (average link stress), even the k-split algorithm which is usually 20% worse than the item-level manages to outperform it when $z > 1.5$ (Figure 8(b)). When the exponent of the Zipf distribution is large enough the set of the items that account for the majority of the requests is very small, and even the most simplistic uncoordinated algorithm manages to minimize the incurred traffic. For example when $z = 1.5$ the $0.1 \cdot N$ items that fit in the cache of a node account for the $\approx 98\%$ of the locally generated traffic, which means that an uncoordinated replication scheme performs almost optimally and there is no need for more sophisticated replication schemes of higher complexity.

From the comparison of the two clustering schemes, we observe that the k-split algorithm manages to perform similarly to our clustering scheme only

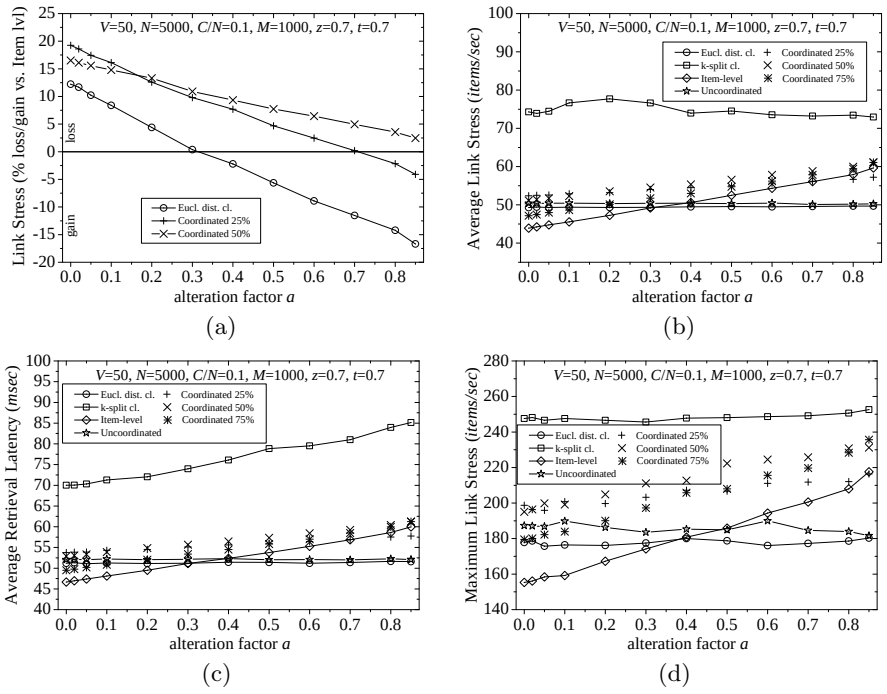

Figure 9: The impact of the content popularity alteration factor a in the performance of the examined schemes.

for large values of z ($z \geq 2$), even for the maximum link stress metric, where it performs up to 60% worse for smaller values of the popularity exponent (Figure 8(d)). For large values of z, the k-split algorithm forms some small clusters that contain the most popular items, which can fit in the caches of the nodes and the remaining items are classified into larger clusters. In this case, the k-split replication scheme manages to replicate within the domain all the popular items similarly to the other schemes.

5.6. Impact of popularity alteration factor

In the scenarios above we assumed a specific value for the popularity alteration factor a. Figure 9 depicts the impact of this factor, which captures the popularity dynamics, on the performance of the examined replication schemes. For scenarios where content popularity is almost static, *i.e.,* small values of a, item-level replication is the natural choice regarding network traffic performance. Instead in dynamic environments, clustering outperforms item-level replication schemes significantly (Figure 9(a) and (b)).

In more details, we observe that for small values of factor a the item-level replication scheme outperforms the rest of the schemes, despite its high computation latency. On the other, hand when $a \geq 0.3$ the proposed clustering scheme performs better than the item-level algorithm mainly due to its small computational complexity (25 times smaller than item-level for the parameters

used in the experiment of Figure 9), which results into a short first phase (*i.e.,* T_{rep} in Figure 2).

Additionally, the three partially coordinated replication schemes are not competitive mainly due to the poor performance of their uncoordinated caching part $(1-c)C$, despite the fact that they decrease their difference from the item-level algorithm with the increase of the alteration factor. Only, when $a \geq 0.4$ the uncoordinated replication algorithm outperforms the item-level replication scheme, since it has zero computation latency and the losses of the item-level caching during the update phase defined in Eq.(1) are irreversible in the remaining time.

The findings regarding the retrieval latency metrics (Figure 9(c)) are in perfect alignment with the findings regarding the link stress ones, with the cluster-based replication schemes and the low rate coordination schemes (especially the uncoordinated one) being insusceptible to the changes of factor a, due to the small T_{rep} period.

6. Conclusions

Content popularity dynamics and the large population of content items to be handled with caching and replication introduce the need for low-complexity replication schemes. In this paper, we showed that, in dynamic environments, the performance of a content replication scheme strongly depends on its complexity. Given the vast number of items circulated over the network, we showed how to significantly reduce the dimensionality of the problem by grouping content items into clusters. We proposed a replication-aware content clustering scheme, which enables control of replication complexity and facilitates timely tracking of content popularity. Our approach is generic and independent of the replication algorithm used. However, since the exact optimal number of clusters cannot be derived analytically, we proposed a systematic methodology to approximate it. Our numerical results show that the proposed replication-aware clustering scheme outperforms significantly generic replication-agnostic clustering schemes, since the latter tend to form unequally sized clusters. Finally, the proposed cluster-level replication scheme requires up to 96% less computation time compared to the fine grained item-level replication, performing significantly better than the straightforward approach of item-level replication for large networks and content catalogues.

Acknowledgments

I. Koutsopoulos acknowledges the support of AUEB funding through the program "Action 1 - Reinforcement of Research" (2015-2017). V. Sourlas work is supported by the EC H2020 UMOBILE project (GA no. 645124) and the the EC H2020 ICN2020 project (GA no. 723014).

Appendix A. Replication-aware clustering algorithm

Input: N: number of unit sized content items,
S: size of a cluster in unit sized items (same for all clusters),
M: number of clusters ($M = N/S$),
I: matrix with pairwise similarity metrics of items (n_1, n_2)
Ensure: The contents of each cluster
m_i // i^{th} item of cluster m
n_j // item j
for $m = 1$ to M **do**
FIND min(I) // $I(n_i, n_j)$ minimum similarity distance
PLACE $m_1 \leftarrow n_i$ // n_i is the first item of cluster m
PLACE $m_2 \leftarrow n_j$ // n_j is the second item of cluster m
$I(m_1, m_2) = \infty$ // exclude from clustering the added items
$I(m_2, m_1) = \infty$
for $s = 3$ to S **do**
$L = [0, 0, \ldots, 0]$
for $l = 1$ to N **do**
for $k = 1$ to $s - 1$ **do**
$L(l) = L(l) + I(n_l, m_k)$
end for
end for
FIND min(L) // item n_z with minimum distance from all items already in cluster m
PLACE $m_s \leftarrow n_z$ // n_z is the s^{th} item of cluster m
for $k = 1$ to $s - 1$ **do**
$I(m_k, m_s) = \infty$
$I(m_s, m_k) = \infty$
end for
end for
end for

References

[1] Cisco visual networking index: Forecast and methodology, `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html`.

[2] Akamai cdn., `http://www.akamai.com`.

[3] Limelight networks cdn., `http://www.limelight.com`.

[4] Netflix CDN, `http://www.netflix.com`.

[5] W. Jiang, R. Zhang-Shen, J. Rexford, M. Chiang, Cooperative Content Distribution and Traffic Engineering in an ISP Network, in: Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '09, ACM, 2009, pp. 239–250.

[6] B. Frank, I. Poese, G. Smaragdakis, S. Uhlig, A. Feldmann, Content-aware Traffic Engineering, in: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '12, ACM, 2012, pp. 413–414.

[7] D. Tuncer, M. Charalambides, R. Landa, G. Pavlou, More control over network resources: An ISP caching perspective, in: Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), 2013, pp. 26–33.

[8] D. Tuncer, V. Sourlas, M. Charalambides, M. Claeys, J. Famaey, G. Pavlou, F. D. Turck, Scalable Cache Management for ISP-Operated Content Delivery Services, IEEE Journal on Selected Areas in Communications 34 (8) (2016) 2063–2076.

[9] J. Kangasharju, J. Roberts, K. W. Ross, Object Replication Strategies in Content Distribution Networks, Comput. Commun. 25 (4) (2002) 376–383.

[10] Netflix OpenConnect Appliance Deployment Guide, vol. 3.7, April 2015.

[11] S. Borst, V. Gupta, A. Walid, Distributed Caching Algorithms for Content Distribution Networks, in: Proceedings of the 29th Conference on Information Communications, INFOCOM'10, IEEE Press, 2010, pp. 1478–1486.

[12] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, G. Caire, FemtoCaching: Wireless Content Delivery Through Distributed Caching Helpers, IEEE Transactions on Information Theory 59 (12) (2013) 8402–8413.

[13] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On Clustering Validation Techniques, J. Intell. Inf. Syst. 17 (2-3) (2001) 107–145.

[14] Y. Chen, L. Qiu, W. Chen, L. Nguyen, R. H. Katz, Efficient and adaptive Web replication using content clustering, IEEE Journal on Selected Areas in Communications 21 (6) (2003) 979–994.

[15] L. Gkatzikis, V. Sourlas, C. Fischione, I. Koutsopoulos, G. Dn, Clustered content replication for hierarchical content delivery networks, in: 2015 IEEE International Conference on Communications (ICC), 2015, pp. 5872–5877.

[16] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, K. K. Ramakrishnan, Optimal Content Placement for a Large-scale VoD System, in: Proceedings of the 6th International COnference, Co-NEXT '10, ACM, New York, NY, USA, 2010, pp. 4:1–4:12.

[17] I. D. Baev, R. Rajaraman, Approximation Algorithms for Data Placement in Arbitrary Networks, in: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '01, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001, pp. 661–670.

[18] N. Laoutaris, O. Telelis, V. Zissimopoulos, I. Stavrakakis, Distributed selfish replication, IEEE Transactions on Parallel and Distributed Systems 17 (12) (2006) 1401–1413.

[19] S. Zaman, D. Grosu, A Distributed Algorithm for the Replica Placement Problem, IEEE Transactions on Parallel and Distributed Systems 22 (9) (2011) 1455–1468.

[20] V. Sourlas, P. Flegkas, G. S. Paschos, D. Katsaros, L. Tassiulas, Storage planning and replica assignment in content-centric publish/subscribe networks, Computer Networks 55 (18) (2011) 4021 – 4032.

[21] V. Sourlas, L. Gkatzikis, P. Flegkas, L. Tassiulas, Distributed Cache Management in Information-Centric Networks, IEEE Transactions on Network and Service Management 10 (3) (2013) 286–299.

[22] Z. Li, J. Lin, M.-I. Akodjenou, G. Xie, M. A. Kaafar, Y. Jin, G. Peng, Watching Videos from Everywhere: A Study of the PPTV Mobile VoD System, in: Proceedings of the 2012 Internet Measurement Conference, IMC '12, ACM, New York, NY, USA, 2012, pp. 185–198.

[23] S. Traverso, M. Ahmed, M. Garetto, P. Giaccone, E. Leonardi, S. Niccolini, Unravelling the Impact of Temporal and Geographical Locality in Content Caching Systems, IEEE Transactions on Multimedia 17 (10) (2015) 1839–1854.

[24] M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latr, G. Pavlou, F. D. Turck, Hybrid multi-tenant cache management for virtualized {ISP} networks, Journal of Network and Computer Applications 68 (2016) 28 – 41.

[25] Y. Li, H. Xie, Y. Wen, Z.-L. Zhang, Coordinating In-Network Caching in Content-Centric Networks: Model and Analysis, in: Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems, ICDCS '13, IEEE Computer Society, 2013, pp. 62–72.

[26] T. F. Gonzalez, Clustering to minimize the maximum intercluster distance, Theoretical Computer Science 38 (1985) 293 – 306.

[27] T. V. Nguyen, F. Safaei, P. Boustead, C. T. Chou, Provisioning overlay distribution networks, Computer Networks 49 (1) (2005) 103 – 118.

[28] H. Sato, S. Matsuoka, T. Endo, File Clustering Based Replication Algorithm in a Grid Environment, in: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09, IEEE Computer Society, 2009, pp. 204–211.

[29] V. Sourlas, I. Psaras, L. Saino, G. Pavlou, Efficient Hash-routing and Domain Clustering Techniques for Information-Centric Networks, Computer Networks 103 (2016) 67 – 83.

[30] T. Janaszka, D. Bursztynowski, M. Dzida, On popularity-based load balancing in content networks, in: Proceedings of the 24th International Teletraffic Congress, ITC '12, 2012, pp. 12:1–12:8.

[31] Content Delivery Networks Interconnection (CDNI), `http://tools.ietf.org/wg/cdni/draft-ietf-cdni-framework/`.

[32] L. Breslau, P. Cao, L. Fan, G. Phillips, 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, 1999, pp. 126–134 vol.1.

[33] G. Dán, N. Carlsson, Power-law Revisited: Large Scale Measurement Study of P2P Content Popularity, in: Proceedings of the 9th International Conference on Peer-to-peer Systems, IPTPS'10, USENIX Association, 2010, pp. 12–12.

[34] Y. Sun, S. K. Fayaz, Y. Guo, V. Sekar, Y. Jin, M. A. Kaafar, S. Uhlig, Trace-driven analysis of icn caching algorithms on video-on-demand workloads, in: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, CoNEXT '14, ACM, 2014, pp. 363–376.

[35] Z. Li, G. Xie, J. Lin, Y. Jin, M. A. Kaafar, K. Salamatian, On the geographic patterns of a large-scale mobile video-on-demand system, in: IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 397–405.

[36] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, S. Shenker, Less Pain, Most of the Gain: Incrementally Deployable ICN, in: Proceedings of the ACM SIGCOMM, SIGCOMM '13, ACM, 2013, pp. 147–158.

[37] D. Aloise, A. Deshpande, P. Hansen, P. Popat, NP-hardness of Euclidean Sum-of-squares Clustering, Mach. Learn. 75 (2) (2009) 245–248.

[38] A. Banerjee, J. Ghosh, Scalable Clustering Algorithms with Balancing Constraints, Data Min. Knowl. Discov. 13 (3) (2006) 365–395.

[39] S. Zhu, D. Wang, T. Li, Data clustering with size constraints, Knowledge-Based Systems 23 (8) (2010) 883 – 889.

[40] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing, Cambridge University Press, 1992.

[41] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, A. Ashkan, Cache content-selection policies for streaming video services, in: IEEE INFOCOM - The 35th Annual IEEE International Conference on Computer Communications, 2016, pp. 1–9.

[42] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, M. Roughan, The Internet Topology Zoo, IEEE Journal on Selected Areas in Communications 29 (9) (2011) 1765–1775.

[43] M. Zink, K. Suh, Y. Gu, J. Kurose, Characteristics of YouTube network traffic at a campus network Measurements, models, and implications, Computer Networks 53 (4) (2009) 501 – 514.

[44] X. Bao, Y. Lin, U. Lee, I. Rimac, R. R. Choudhury, DataSpotting: Exploiting naturally clustered mobile devices to offload cellular traffic, in: 2013 Proceedings IEEE INFOCOM, 2013, pp. 420–424.