Scalable Cache Management for ISP-operated Content Delivery Services

Daphne Tuncer, Vasilis Sourlas, Marinos Charalambides, Maxim Claeys, Jeroen Famaey, George Pavlou, and Filip De Turck

***Abstract*—Content Delivery Networks (CDNs) have been the prevalent method for the efficient delivery of content across the Internet. Management operations performed by CDNs are usually applied based only on limited information about Internet Service Provider (ISP) networks, which can have a negative impact on the utilization of ISP resources. To overcome these issues, previous research efforts have been investigating ISP-operated content delivery services, by which an ISP can deploy its own in-network caching infrastructure and implement its own cache management strategies. In this paper, we extend our previous work on ISP-operated content distribution and develop a novel scalable and efficient distributed approach to control the placement of content in the available caching points. The proposed approach relies on parallelizing the decision-making process and the use of network partitioning to cluster the distributed decision-making points, which enables fast reconfiguration and limits the volume of information required to take reconfiguration decisions. We evaluate the performance of our approach based on a wide range of parameters. The results demonstrate that the proposed solution can outperform previous approaches in terms of management overhead and complexity while offering similar network and caching performance.**

***Index Terms*—Cache Management, Content Placement, Network Partitioning, Content Delivery Networks.**

I. Introduction

The consumption of video content has increased dramatically over the last few years and currently dominates the network traffic. As indicated by reports from Cisco [1], video had a 67% share of the global IP traffic in 2014 and is expected to reach 80% in the next 3 years. Content Delivery Networks (CDNs) have been the prevalent method for delivering video content (content and video content are used interchangeably hereafter) across the Internet. In order to meet the growing demand, CDN providers deploy massively distributed storage infrastructures that host content copies of contracting content providers and maintain business relationships with Internet Service Providers (ISPs). Surrogate servers are strategically placed and connected to ISP network edges [2] so that content can be closer to clients, thus reducing both access latency and the consumption of network bandwidth for content delivery.

Current content delivery services operated by large CDN providers like Akamai [3] and Netflix [4] can exert enormous strain on ISP networks [5]. This is mainly attributed to the fact that CDN providers control both the placement of content in surrogate servers spanning different geographic locations, as well as the decision on where to serve client requests from (*i.e.,* server selection) [6]. These decisions are taken without knowledge of the precise network topology and state in terms of traffic load, and can result in network performance degradation affecting the service quality experienced by the end users.

To address this problem, in previous work we proposed a cache management approach that enables ISPs to have more control over their network resources [7] [8]. In contrast to CDN providers, ISPs have a global knowledge about the utilization of their network and can thus carefully configure the management operations to perform. Exploiting the decreasing cost of storage modules, the approach previously proposed involves operating a limited capacity CDN service within ISP domains by deploying caching points in the network. Such a service can cache popular content, specific to an ISP, and serve most client requests from within the network instead of fetching content items from surrogate/origin servers. Empowering ISPs with caching capabilities can allow them to implement their own content placement and server selection strategies, which will result in better utilization of network resources. In addition, there are economic incentives for an ISP to adopt this approach given that traffic on inter-domain links can be significantly decreased.

While the utilization of network resources is affected by both content placement and server selection operations, this work concentrates on the former. Research CDNs, such as Coral [9], have proposed distributed management approaches [10]. However, commercial CDNs have been traditionally using centralized models for managing the placement of content in distributed surrogate servers. Both centralized and distributed management approaches have their pros and cons in terms of configuration optimality, implementation complexity, computation and communication overhead, as well as reconfiguration timescales. For example, centralized solutions would require collecting information from multiple locations to build a global view of the user demand, which can incur

This research was funded by the EPSRC KCN project (EP/L026120/1) and by the Flamingo Network of Excellence project (318488) of the EU Seventh Framework Programme. Vasilis Sourlas work is supported by the European Commission through the FP7-PEOPLE-IEF INTENT project, Grant Agreement no. 628360, and Maxim Claeys is funded by a grant of the Agency of Innovation by Science and Technology in Flanders (IWT).

D. Tuncer (corresponding author), V. Sourlas, M. Charalambides and G. Pavlou are with the Department of Electronic and Electrical Engineering, University College London, London WC1E 7JE, UK (e-mail: d.tuncer@ee.ucl.ac.uk; v.sourlas@ucl.ac.uk; marinos.charalambides@ucl.ac.uk; g.pavlou.ucl.ac.uk).

M. Claeys and F. De Turck are with the Department of Information Technology, Ghent University - iMinds, Belgium (e-mail: maxim.claeys@intec.ugent.be; Filip.DeTurck@intec.ugent.be).

J. Famaey is with the Department of Mathematics and Computer Science, University of Antwerp - iMinds, Belgium (e-mail: jeroen.famaey@uantwerpen.be).

significant traffic overhead. In addition, computationally intensive algorithms are executed in an off-line fashion, producing longer-term configurations (in the order of days), and do not allow for adaptation to changing user demand. However, they are simple to implement and can result to optimal placement decisions. On the other hand, distributed solutions involve less computational overhead since they employ simpler placement algorithms and usually operate on local views of the demand. This allows them to execute in short timescales (*e.g.,* in the order of minutes or hours) and track the demand, but at the cost of global optimality. Their implementation is more complex since a coordination mechanism may need to be in place to maintain configuration integrity among decision-making points, which will also incur some communication overhead.

Although the release date and the expected popularity of some video content could be known in advance, there are cases that these cannot be predicted. A prime example concerns disaster scenarios (*e.g.,* earthquakes, terrorist attacks) in the event of which news channels create video reports and updates. In the German Wings plane crash in France (24 March 2015), the official BBC report on the tragic event[1] gathered nearly 100,000 hits during the first 24 hours. A centralized management system, operating with long reconfiguration timescales, would not cache this content soon after its release thus forcing user requests to be served from the origin server. This increases the access latency but also causes unnecessary usage of network resources for delivering the content. We believe that cache management systems should be more reactive to user demand, or at least implement short reconfiguration cycles.

To this end, we extend our previous work on pro-active cache management [7] [8] and focus on solutions that can enable the fast execution of the content placement algorithm and the reduction of the communication overhead. In [8], we considered the content management problem in the context of a multi-tenant scenario. We focused on the joint optimization of content placement and server selection, which is formulated as an Integer Linear Programming problem and may have scalability limitations as both the content catalogue and the caching infrastructure increase. To overcome these limitations, we propose a new distributed content management approach. In contrast to the strategies proposed in [7] which rely on sequential decisions, here we develop parallelized decision making processes that provide better scalability and lower management complexity. We also develop a network partitioning approach that reduces the amount of information that needs to be exchanged between decision making entities. The results of our experiments demonstrate that the proposed solution can achieve network and cache performance similar to the ones obtained with the previous approaches while significantly reducing the management/communication overhead and complexity.

The remainder of the paper is organized as follows. Section II presents the cache management framework considered in this work. In Section III, we describe the proposed content placement approach and in Section IV, we present two network partitioning methods. We evaluate the performance of the placement algorithm in Section V. Section VI discusses related work and finally, conclusions and future directions are provided in Section VII.

[1]https://www.youtube.com/watch?v=UYk3NE4Bbiw

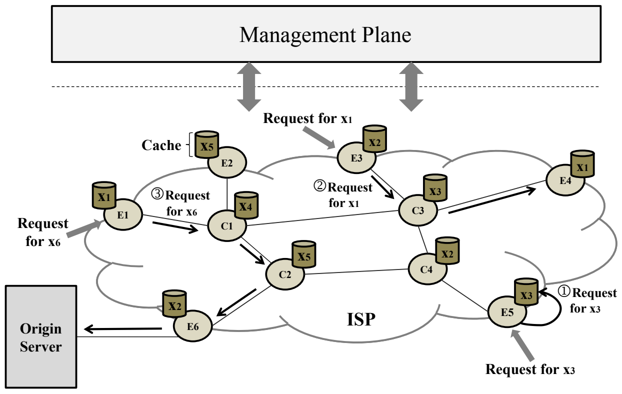


Fig. 1. Overview of the proposed caching infrastructure.

II. Cache Management Framework

In this section, we describe the cache management framework considered in this work.

A. In-Network Caching

We consider a scenario where an ISP operates its own caching service by deploying a set of caching points within its network, as depicted in Fig. 1. The caching locations can be associated with network edge nodes, which represent access networks connecting multiple users in the same region, or with core nodes, interconnecting the access networks. Unlike the heavyweight caching infrastructure maintained by CDN providers, the proposed caching service relies on a simple and lightweight solution. As can be observed in Fig. 1, each network node is associated with caching capabilities - represented by a local cache - which is used to locally store a set of video content items. Local caches can be implemented as external storage modules attached to the routers or, with the advent of flash drive technology, integrated within the routers themselves.

Based on this service, the ISP can cache popular content (specific to its users) and, as such, serve most client requests from within the network instead of fetching content items from origin servers. In practice, the total caching space available in the network will not permit, however, to store all possible content items. A subset of them needs therefore to be preselected, for example based on content popularity, so that the volume of selected items does not exceed the total caching capacity. More specifically, three cases can be considered when deciding how to satisfy a content request, as illustrated in Fig. 1. In the first case, the request for content X_3 received at edge node E5 is served locally given the availability of the item in the cache associated with node E5. In the second case, content X_1 requested at edge node E3 is not available in the local cache but can be found in the caches associated with nodes E1 and E4. To optimize the average retrieval delay, the item is fetched from the closest location, *i.e.,* node E4. Finally, in the third case, the request for content X_6 received at edge node E1 needs to be redirected to the origin server since the item is not stored in any of the network caches.

While serving a content request from the local cache does not incur any cost in terms of network footprint and minimizes the delay in satisfying the request, retrieving a content from a remote location has an impact both on the network utilization and retrieval latency. Intelligently managing the placement of the content items in the different caches, under specific user demand characteristics, is therefore essential to optimize the use of network resources, and, as such, the quality of service/experience offered to the users.

B. Management Plane

In the proposed framework, the caching infrastructure is managed through a management plane, as shown in Fig. 1, which implements the logic of cache management applications (*e.g.,* content placement strategies and request redirection approaches). From an architectural point of view, this can follow a centralized approach, where management operations are executed by a centralized management system that operates on a global view of the network. It can also rely on a decentralized model, where a set of distributed managers is responsible for performing management operations. In this case, each manager is in charge of a subset of the network caches[2]. To take management decisions, the manager can use locally available information about the resources it supervises or obtained from other managers, using for instance the management substrate structure and communication protocols proposed in [11] and [12].

The choice of implementation of the management plane depends on the timescale at which management operations are executed. In general, centralized solutions are well-suited in the case of applications for which the time between each execution is significantly greater than the time to collect, compute and disseminate results. In contrast, distributed management approaches are usually considered to implement the logic of applications executed at short timescales [13]. In this paper, we focus on a content management application that can be executed at short timescales (for example every 30 minutes) to better track changes in the user demand and focus on decentralized management plane solutions, where one cache manager is associated with a single caching location.

III. Content Distribution Strategy

In this section, we describe a content distribution approach to control the configuration of each cache.

A. Content Placement

We first formalize the content placement problem and introduce some notations. Let $\mathcal{M}$ represent the set of M network caches. These are divided into the set of caches $\mathcal{M}_E$, associated with network edge nodes, and $\mathcal{M}_C$, associated with core nodes, so that $\mathcal{M} = \mathcal{M}_E \cup \mathcal{M}_C$ and $\mathcal{M}_E \cap \mathcal{M}_C = \emptyset$. We denote c_m as the capacity of cache $m \in \mathcal{M}$. Let $\mathcal{X}$

[2]Each cache is under the supervision of one manager only.

TABLE I
Summary of the content placement problem notations.

$\mathcal{M}$	Set of M caches
$\mathcal{M}_E$	Set of caches attached to edge nodes
$\mathcal{M}_C$	Set of caches attached to core nodes
c_m	Capacity of cache m
$\mathcal{X}$	Set of X video content items
s^x	Size of content item x
r^x_m	Number of requests for item x received at cache location m
$\boldsymbol{r}^\mathrm{x}$	Total number of requests for item x in the network
a^x_m	Binary variable for content placement of item x at cache m
S_m	Remaining capacity available at cache m

represent the set of the X video content items requested in the network. We denote s^x as the size of content $x \in \mathcal{X}$ in bits. In addition, we denote r^x_m as the number of requests for item x received at cache location m. In the case of edge caches, this represents the number of requests coming from the users connected to the relevant access network. For core caches, this is the aggregate of the requests for the content that transit through the considered core node. We define $\boldsymbol{r}^\mathrm{x}$ as the total number of requests for x received in the network, so that we have: $\forall x \in \mathcal{X}$, $\boldsymbol{r}^\mathrm{x} = \sum_{m \in \mathcal{M}_E} r^\mathrm{x}_\mathrm{m}$. Finally, we denote a^x_m as the binary variable equal to 1 if item x is cached at location m and 0 otherwise. For all $m \in \mathcal{M}$, we denote S_m as the remaining capacity available at cache m, so that we have: $S_\mathrm{m} = c_\mathrm{m} - \sum_{\mathrm{x} \in \mathcal{X}} a^\mathrm{x}_\mathrm{m} \cdot s^\mathrm{x}$. A summary of these notations is provided in Table I.

The objective of the content distribution algorithm is to determine the number of copies of each content to store in the network, as well as the location of each copy, so that the placement of content items in each individual cache satisfies the cache capacity constraint. Formally, this can be defined as finding, $\forall m \in \mathcal{M}$ and $\forall x \in \mathcal{X}$, the values of a^x_m under the cache capacity constraint, *i.e.,* $\forall m \in \mathcal{M}$, $\sum_{\mathrm{x} \in \mathcal{X}} a^\mathrm{x}_\mathrm{m} \cdot s^\mathrm{x} \leq c_\mathrm{m}$.

Based on the resulting cache configuration, a mechanism is then used to decide from where to serve each incoming request, *i.e.,* either locally or redirected to a remote location. In this work, we assume that content items are retrieved from the closest available location. As such, placement decisions coupled with strategies to decide from where to serve the requests can lead to a reduction of both the utilization of network resources and the average content delivery latency.

In the rest of this paper, we assume that all content items have the same size. We believe that this a realistic assumption as the fragmentation of items into equally sized chunks is a requirement of many replication mechanisms, *e.g.,* [14], [15], and is also present in various content distribution systems such as BitTorrent and modern streaming technologies (*e.g.,* Apple HLS, MPEG DASH) that usually operate on fixed duration segments. In addition, we also assume that all caches have the same capacity, which is a common assumption in the literature, *e.g.,* [14] [16]. Determining the correlation between the content size and the user interest for the content, as well as taking into account different cache sizes, are challenging research issues which are outside the scope of this paper. As such, we have:

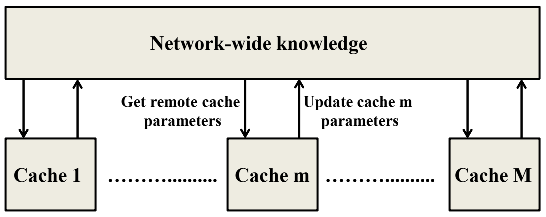


Fig. 2. Parallelization of the placement decisions.

$\forall x \in \mathcal{X}$, $s^{x} = s$ and $\forall m \in \mathcal{M}$, $c_{m} = c$.

In our previous work, we developed two placement strategies [7] to decide on the allocation of the content items in the different caching locations. These strategies follow an iterative process design, so that the placement decisions are taken iteratively: the decision of a single cache manager (*i.e.,* per content and per location) is communicated to the next manager at each iteration. Although such sequential processes have the advantage of preventing the occurrence of conflicting decisions, they have inherent limitations in terms of scalability as the number of items and caching locations increase. To overcome these limitations, we propose a new strategy which relies on the parallelization of the placement decisions. In contrast to our previous approach, the managers compute cache configurations concurrently for multiple content items, which results to a shorter decision-making process. We elaborate on the details of the approach in the next subsection.

B. Proposed Approach

The objective of the proposed algorithm is to cache the most popular content items in the network and to place them in close proximity to the locations they are requested the most. In other words, placement decisions are driven both by the global popularity of the content (in terms of number of requests) and the geographical location of the interests for each content.

As explained in Section III-A, the algorithm relies on parallelizing the decision-making process, so that one instance of the reconfiguration process is executed per caching location, as depicted in Fig. 2. More specifically, the reconfiguration process consists of two phases. The objective of the first phase is to maximize the number of unique content items to cache in the network, which, as a result, can lead to a reduction of the number of redirections to the origin server. In this phase, the decision process executed at each caching location relies on a common network-wide knowledge shared between the different reconfiguration instances. The objective of the second phase is to replicate the content items in order to maximize the local cache hit ratio (i.e., to maximize the chances of finding the requested content locally). In addition, this also increases the probability of being able to serve the incoming requests from the near vicinity, which can lead to a reduction in terms of network resource utilization. In this case, the decisions taken for each caching point rely only on local information regarding the location controlled by the relevant reconfiguration instance. The knowledge used in both phases is based on information about the interests for each content at each caching location, which can be obtained by prediction strategies, such as the one proposed in our previous work [8].

Content	**List of caching locations**
1	$< \ 1_1 \ \ldots \ j_1 \ \ldots \ M_1 \ >$
...	... ...
i	$< \ 1_i \ \ldots \ j_i \ \ldots \ M_i \ >$
...	... ...
X	$< \ 1_X \ \ldots \ j_X \ \ldots \ M_X \ >$

Fig. 3. Structure T.

C. A Two-Phase Content Placement Process

In this subsection, we elaborate on the details of each phase of the reconfiguration process.

1) First Phase - Maximizing Content Diversity: In this phase, decisions are taken based on global information about all caching locations. In particular, this represents (i) the number of requests for each content item received at each caching location, from which the global popularity of the item (*i.e.,* total number of requests in the network for the content item) and the interest of each location for the considered item can be determined, and (ii) the configuration of the cache associated with each caching location (*i.e.,* list of items available locally).

Based on this information, the algorithm can determine the global popularity rank of each content x with respect to all the content items requested in the network. The ranks are in the interval $[1;X]$, so that the content with the highest popularity has a rank equal to 1 and the one with the lowest popularity has a rank equal to X. In addition, we can also compute, for each content item x, the rank of each caching location m in terms of interest for x with respect to all other caching points. In this case, the ranks are in the interval $[1;M]$, so that the location with the highest interest has a rank equal to 1 and the one with the lowest interest has a rank equal to M. Based on the rank values, we define the structure T as shown in Fig. 3, where j_i represents the rank of cache j with respect to item i. Content items in T are sorted by decreasing order of global popularity, while the caching locations are sorted, for each item, by decreasing order of interest for the item. The configuration of each cache is represented by the binary variables a_{m}^{x}, defined in Section III-A, which are equal to 1 if content x is cached at location m and to 0 otherwise. In addition, the binary variable u^{x} is defined for each content item x and is equal to 1 if the content is cached in at least one caching location in the network and to 0 otherwise. By definition, the value of u^{x} is driven by the value of the a_{m}^{x} variables. We note $\mathcal{K}$ the knowledge of (i) the structure T, (ii) the values of all the a_{m}^{x} and (iii) all the parameters c_{m}. In this phase, the reconfiguration process executed for each caching location is associated with an instance of the knowledge $\mathcal{K}$, so that all processes can operate on the same global view of the system.

More specifically, the first phase of the algorithm relies on an iterative process, so that, at each iteration, the following operations are performed in parallel for each caching location. We denote $\mathcal{K}(i)$ as knowledge $\mathcal{K}$ considered at iteration i.

1) Based on $\mathcal{K}(i)$, determine for all items x, u^{x}, and for all caches m, S_{m}.

2) Define $S(i)$ as the normalized caching space available remotely (*i.e.,* in the other caches). $S(i)$ is expressed in units.
3) Consider iteratively the p first content items in T so that $u^{\mathrm{p}} = 0$ (*i.e.,* the items not already cached in the network) and $p \leq S(i)$.
4) For each considered item, do:
    a) If the cache for which the process is executed is attached to the location with the highest interest for the item and has enough capacity to accommodate it, then place the content in that cache.
    b) Otherwise, disregard the content.

At the end of each iteration i, the values of $a^{\mathrm{x}}_{\mathrm{m}}$ are updated and a new instance of the knowledge $\mathcal{K}(i+1)$ is passed to each process (note that this concerns only the updated $a^{\mathrm{x}}_{\mathrm{m}}$). The algorithm terminates if at least one of the three following conditions is satisfied: a) all the available caching space has been consumed, b) all requested content items have been cached or c) there are no more feasible placements. The latter occurs in case caching space is still available at a location but none of the remaining content items to cache is requested at that location, or in case the rank of the first content to consider (driven by the parameter p) is larger than the total remaining caching space (expressed in units).

In this phase, the placement decisions computed for each cache rely on an implicit coordination between the processes, which limits the number and type of items to accommodate locally. This forces the algorithm to maximize the number of unique items cached in the network, and, as such, to minimize the number of redirections to the origin server. It is worth noting that although the first phase of the algorithm involves sequential decisions, these are only serialized at the group of content level (caches operate in parallel), which limits the number of iterations. This is in contrast to purely iterative approaches, which consider sequential decisions per content and per caching location, thus leading to much longer execution time.

2) Second Phase - Content Replication: The objective of the second phase of the algorithm is to perform replication in order to optimize the cache hit ratio. In this case, placement decisions are taken independently for each caching location. The algorithm tries to fill up the remaining capacity by copying the locally most popular content items not already cached in the considered location[3]. Each placement instance operates on information regarding the number of requests received locally for each item.

The pseudo-code of the content placement process is presented in Fig. 4. We quantitatively evaluate the convergence of the algorithm in Section V.

D. Decentralized Decision-Making

From an implementation point of view, the different reconfiguration processes are distributed across a set of cache managers, which have local knowledge about the resources they supervise. To build the required global knowledge, the managers need to communicate between themselves. The volume of information to exchange is driven by the degree of distribution of the management plane (*i.e.,* number of managers). In addition, this also depends on the stage of execution of the algorithm. Initially, each manager needs to acquire information about the number of requests received for each content item at each caching location, which is used to locally build the structure T. In addition, the managers need to communicate the value of their local caching space (*i.e.,* the c_{m}), as well as the configuration of their associated cache (*i.e.,* the $a^{\mathrm{x}}_{\mathrm{m}}$). In contrast, the information which needs to be exchanged at each subsequent iteration concerns only the updated value of $a^{\mathrm{x}}_{\mathrm{m}}$, from which the values S_{m} and u^{x} can easily be deduced. To further reduce the overhead incurred by the coordination between the managers, the exchanged information can be compressed using simple encoding schemes such as Bloom filters [17]. It is worth noting that the second phase of the algorithm does not have a cost in terms of coordination overhead.

From a complexity point of view, the cost and convergence of the algorithm is affected by both the number of requested content items and the number of caching locations. In the worst case, the algorithm operates in a similar fashion to a purely iterative process and the complexity is in the order of $O(M \cdot X)$. This could happen, in particular, in the case of uniform user demand distributions. In a real scenario, however, the demand is unlikely to follow such a distribution [18]. We further discuss and evaluate the scalability and complexity of the proposed approach in Section V-B1.

[3]An item is cached in a location only if the content is requested at that location.

```
Parallelized Content Placement Process
▷ Build initial knowledge $\mathcal{K}$.
▷ Enter process phase 1.
while Ending conditions of phase 1 non satisfied do
    Execute phase 1 algorithm.
    Update knowledge $\mathcal{K}$.
end while
▷ Enter process phase 2.
```

Fig. 4. Pseudo-code of the proposed parallelized content placement process.

IV. Network Partitioning

While the parallelization of the decision process enables parallelism at the network level, issues may arise with respect to the volume of information that needs to be exchanged in order to maintain a global network view in the case of a decentralized implementation. To limit the scope of knowledge and, as such, the volume of information to exchange and process, we propose to partition the network into regions. More specifically, network nodes are partitioned into a set of N clusters so that caching decisions (*i.e.,* placement and request redirection) are taken at the cluster level. Nodes in each cluster have only visibility about other nodes in the same cluster. To limit the complexity of the proposed approach, coordination between clusters is not enabled. As such, upon receiving a content request, each caching location checks whether it can be satisfied from within the cluster, and if not, the request is redirected to the origin server.

In this paper, we use ideas from the area of machine learning to leverage the volume of information which needs to be exchanged by partitioning the network topology, while maintaining a level of network and cache performance similar to the one that can be achieved when considering a single network domain. We investigate two well known partitioning methods proposed in the literature: the k-split approach (*e.g.,* [19] [20]), which aims at minimizing the maximum intercluster distance, and the k-medoids strategy (*e.g.,* [21]), which aims at minimizing the distance of each node in a cluster from a designated node representing the center of the cluster.

Partitioning a single network domain of M caches[4] into N partitions facilitates content placement at the partition level. In addition to minimizing the volume of information that needs to be exchanged at each reconfiguration cycle, network partitioning can also lead to a decrease in content retrieval latency given that the distance between the requesting and the replying nodes will, on average, be smaller. However, the application of the placement strategy onto a smaller set of caches may have an impact on the overall cache hit ratio. In this case, a smaller subset of the requested items is likely to be replicated within each cluster. Due to the absence of coordination between the clusters, this may result in a large volume of redirections to the origin server, leading to a degradation of the cache hit ratio.

A. k-split network partitioning

The problem of partitioning a set of caches in N partitions is generally NP-hard [22]. Let $\mathcal{N}$ represent the set of computed clusters/partitions. In this section, we consider the use of the k-split clustering/partitioning algorithm proposed in [19] and [20] to partition the network domain (*i.e.,* topology). The k-split algorithm partitions the network into k sub-domains. Its objective is to minimize the maximum intercluster distance based on a similarity metric. This means that initially, a representative similarity index L (defined based on the considered partitioning metric), has to be derived for each network cache.

In this work, we use the actual topological latency/distance between two domain caches as the clustering metric. In this case, the similarity of two caches m_1 and m_2 is captured by their topological distance $d_{\mathrm{m_1 m_2}}$, which can be defined based for example on the hop count, the latency, *etc*. The usage of the above metric alongside with the k-split clustering algorithm produces a set of clusters, where the distance of a cache m, that belongs to a cluster $n \in \mathcal{N}$, to the other caches within cluster n is always smaller than the distance of cache m to any other cache in any of the other clusters $\mathcal{N} \setminus \{n\}$.

Since user requests are not directly received at the caches associated with core nodes (*i.e.,* the core caches), the underlying network partitioning method should avoid the formation of clusters consisting of core caches only. To ensure that core caches are clustered with at least one edge cache, we slightly modify the traditional k-split algorithm. In particular, clusters are formed iteratively, so that, at each iteration, the core caches are assigned to the most appropriate cluster (in terms of similarity) which has not already been assigned with core caches during that iteration. To implement this procedure, the core cache assignment is based on a round robin mechanism. The modification of the partitioning algorithm is applied to the core cache assignment only, which results to the maximum of core caches per cluster being $\lceil |\mathcal{M}_C|/N \rceil$.

[4]As stated in Section II-A, there is one cache per network node.

B. k-medoids network partitioning

In this section, we consider another well defined low-complexity algorithm for the partition of the domain, called the k-medoids clustering algorithm [21]. The k-medoids method aims at minimizing the distance between caches labeled to be in a cluster and a cache designated as the center of the considered cluster. The traditional k-medoids algorithm randomly selects k of the M network caches as the medoids/centers of the clusters and associates each cache to the closest medoid based on the considered clustering metric. In a similar fashion to the k-split approach, we use the actual topological latency/distance between two caches as the clustering metric. The procedure is repeated for every combination of k out the M caches and the configuration with the lowest cost is selected.

Due to its exhaustive search approach, the k-medoids algorithm can significantly affect the complexity of the network partitioning procedure. To limit this complexity, we derive a method by which the centers of the clusters are preselected based on criteria of "importance". In this paper, we define the "importance" of a cache according to the graph-based metric of betweenness centrality, which is used to determine the k cluster centers. More specifically, the betweenness centrality is an indicator of the centrality of a cache in a network. It is equal to the number of shortest paths between all pairs of vertices that pass through the node associated with that cache. A cache with high betweenness centrality has a large influence on the transfer of items through the network, under the assumption that item transfer follows the shortest paths (an assumption also made in this paper). After electing the k caches with the highest betweenness centrality and assigning them as the center of each cluster, we associate each of the remaining $(M-k)$ caches with the closest medoid based on the topological distance.

In addition, we further modify the k-medoids algorithm with respect to both the assignment of core caches to clusters and the balance degree of each cluster in terms of size. In particular, the core cache assignment is similar to the procedure described for the k-split method in Section IV-A. Regarding the cluster size, we control the number of caches assigned to each cluster by introducing a constraint on the maximum number of caches to be included in a cluster. In the proposed approach, each cluster is assigned around $\lceil M/N \rceil$ caches. More precisely, the procedure works as follows: cache m is assigned to cluster n if the distance between m and the medoid of cluster n is smaller than the distance of m to the medoids of any other clusters and if cluster n has been assigned less than $\lceil M/N \rceil$ nodes. The second closest medoid is otherwise selected until all the caches have been assigned to a cluster. The modification of the cluster assignment ensures that the complexity of the placement decision-making process (both in terms of management overhead and volume of information to be maintained locally) is equivalent for each cluster. It should be noted that this is not the case with the k-split algorithm which can produce clusters of different sizes. However, nodes in the clusters computed by the k-split algorithm are generally closer to each other compared to the result obtained with the modified k-medoids algorithm. As such, there exists a trade-off between the computational/communication complexity of the placement procedure and its corresponding efficiency, which we investigate in detail in Section V-B1.

V. Evaluation

In this section, we evaluate the performance of the proposed placement algorithm based on a wide range of parameters. The objective is to evaluate the performance in terms of network and caching costs, as well as in terms of management overhead (*i.e.,* scalability and complexity).

A. Experiment Settings

We implemented the considered scenario in a Matlab-based simulator. The simulator relies on a set of parameters which can be tuned to control the configuration of the system. An important aspect of this work concerns the evaluation of the complexity and scalability of the proposed approach. In the absence of a sufficiently real large dataset (and associated topology) to test these characteristics, we resort to using synthetic traces.

We evaluate the performance of our approach based on the Interoute network topology [23] as provided through the Zoo topology dataset[5]. The Interoute network has 110 nodes and 147 bidirectional links, and essentially spans over the European continent. In the considered scenario, each node is associated with a cache and we assume that, by default, all caches have the same capacity, defined as a percentage of the content catalog size. We select 88 nodes as edge nodes and 22 nodes (*i.e.,* 20%) as core nodes (similar proportions have been considered in previous work, *e.g.,* [24] [25]). We also define a transmission delay for each link. The values of the delay are drawn from a random distribution and the delay of each link ranges from 1 ms up to 200 ms based on the results reported in [26] and [27]. The path between two nodes is based on the shortest path computed according to the transmission delay. In addition, we consider a single origin server which is attached to one of the edge nodes and maintains the whole content catalog.

While we want to test the scalability of our approach, the number of content items to consider is also limited by the scalability of the simulator. In [28], Sun et al. indicate that, in their VoD dataset, the size of the catalog of watched videos amounts to $500K$ unique items. To satisfy the scalability constraint of our simulator while preserving a good approximation of a realistic content catalog, we consider a list of $100K$ unit sized video content items.

To generate user demand, we use the demand model presented in our previous work [7]. In the proposed model, the user demand is characterized by a) the total volume of requests for each content in the network, which defines the global content popularity (GCP), and b) the number of caching locations where each content is requested, defining the geographic distribution of the interests (GDI). The GCP is represented by a Zipf distribution of parameter z, which gives the total number of requests for each content item in the network. The GDI is represented by a function f_β of parameter β [7], which provides, for each item, the number of locations (*i.e.,* edge nodes) from where it is requested. For each content, the locations are randomly selected among the 88 edge nodes and the number of requests for the item is then equally divided between these locations. We select the value of z based on the characteristics of the dataset used in [28] and set it to 1.174. The value of β is set to 2, since according to [7] such a value increases the heterogeneity of the GDI. The parameter settings are summarized in Table II.

[5]http://www.topology-zoo.org/dataset.html

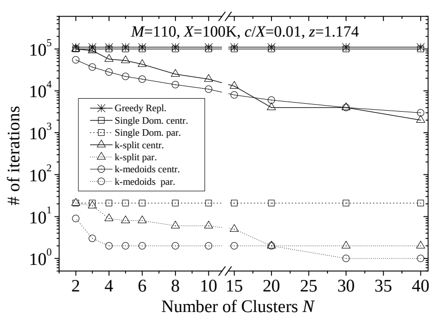
TABLE II
Summary of the evaluation parameter settings.

Parameter	**Value**
Number of nodes	110
Number of links	147
Number of caches	110
Number of items	10^5
z	1.174
β	2

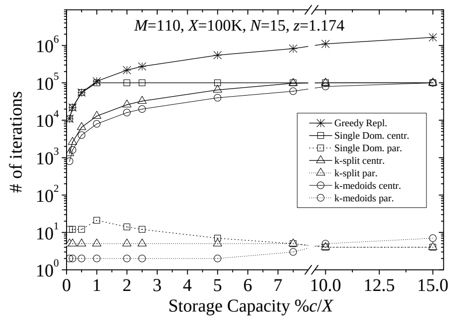
B. Performance Evaluation

We compare the performance of six different schemes. To show the impact of parallelization, we first compare the performance of the proposed approach (denoted Parallel in the plots) to the Global-Popularity driven Strategy (GPS) presented in [7] (denoted Centralized in the plots). The GPS algorithm follows an iterative process where one placement decision is taken per content and per caching location at a time. For both approaches, we also investigate the influence of network partitioning by considering scenarios with and without node clustering. In the case without clustering, the visibility of each node is defined at the network level and we refer to this scenario as Single Domain. In the other case, decisions are taken at the cluster level and we consider here two scenarios: (i) the network is partitioned according to the proposed k-split method and (ii) the network is partitioned according to the proposed k-medoids method. Finally, we also implement the greedy content placement algorithm presented by Qiu et al. in [29], which we use as a performance benchmark for the Single Domain scenario. The greedy algorithm is an iterative approach which aims at minimizing the average retrieval latency. It was shown that solutions of high quality can be achieved with this algorithm. In particular, its median performance is within a factor of $1.1-1.5$ of the optimal and around a factor of 4 for the maximum cases [16].

1) Scalability and Complexity: An important aspect of the proposed approach concerns its scalability and complexity. In

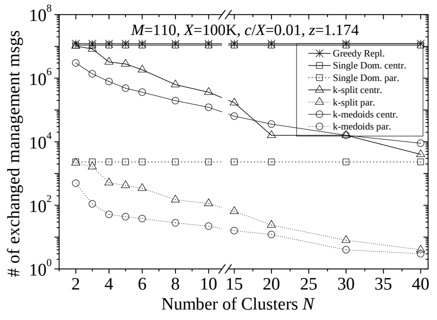


(a) Number of Clusters.

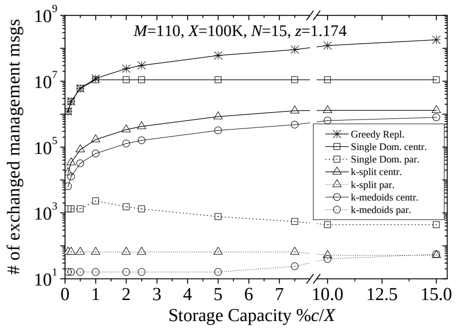


(b) Storage Capacity.

Fig. 5. Convergence of the considered placement schemes.



(a) Number of Clusters.



(b) Storage Capacity.

Fig. 6. Communication costs of the considered placement schemes.

this section, we report the number of iterations required by each scheme to terminate and investigate how this can be affected by the number of clusters and the storage capacity available in the network. The results are presented in Fig. 5[6]. As expected, the largest number of iterations is taken by the Greedy scheme and the Single Domain GPS approach given that they both follow a full iterative process (one decision per content and per location).

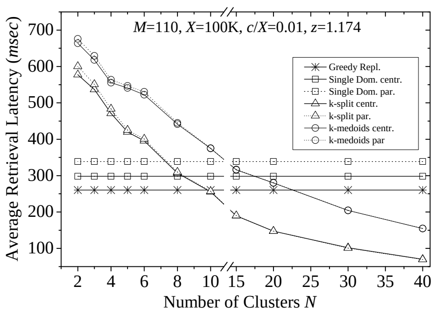
As can be observed, the use of clustering leads to a significant decrease in the number of iterations required by the GPS scheme (reduced by a factor 10), which uniformly diminishes as the number of clusters increases. With the proposed Parallel approach, the number of iterations is further reduced (by a factor 10^4) and it takes around 20 iterations for the algorithm to terminate. The speed of execution of the algorithm is further increased when using network partitioning. It can be observed that the smaller number of iterations can be achieved when nodes are clustered based on the k-medoids method compared to the k-split method. This is because the k-medoids method has been altered to produce clusters of the same size, whereas the k-split method may compute some very large clusters, depending on the actual characteristics of the underlying topology (*i.e.,* similarity metric). Also it can be observed that, in contrast to other schemes, the convergence of the Parallel approach (with and without partitioning) does not depend on the available storage space in the network.

Finally, we plot in Fig. 6 the volume of coordination messages that need to be exchanged at each reconfiguration cycle. The observations formulated with respect to the number of iterations apply here as well. In particular, it can be concluded that the use of a parallel decision-making process in conjunction with network partitioning can significantly improve the performance of the content management application in terms of overhead and complexity.

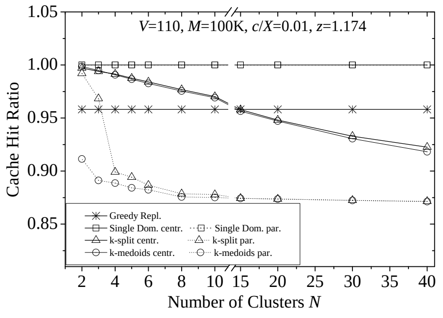
[6]In the cases with network partitioning, the number of iterations is driven by the cluster with the larger number of iterations.

2) Network and Caching Performance: In the previous subsection, we showed that the proposed placement strategy coupled with network partitioning outperforms all other approaches in terms of communication overhead and complexity. We now focus on the performance of the different schemes in terms of network and caching costs based on three performance indicators: (i), the average retrieval latency (*i.e.,* average delay for fetching the requested content from the closest available location), (ii), the network cache hit ratio (*i.e.,* ratio of requests that can be served from within the domain), and (iii), the average link stress (*i.e.,* average number of items transiting a link per second, which can be used as an indicator of the average network link load). For each scheme, we evaluate the influence of the number of clusters N, the storage capacity c available at each caching location and the value of the Zipf distribution parameter z. In the rest of this section, unless otherwise stated, we consider the following reference parameter values: $N = 15$, $c = 0.01 \cdot X$ and $z = 1.174$

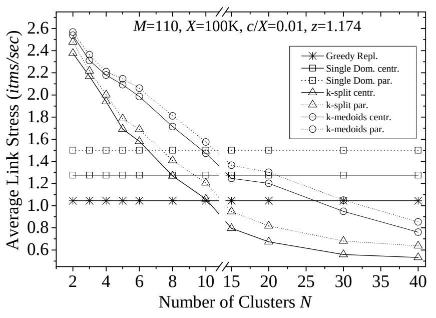
Impact of the number of clusters: We investigate the impact of the number of clusters on the network and caching performance. Although the single domain schemes are not affected by the number of clusters, their performances are reported for comparison purposes. The results are shown in



(a) Average Retrieval Latency.

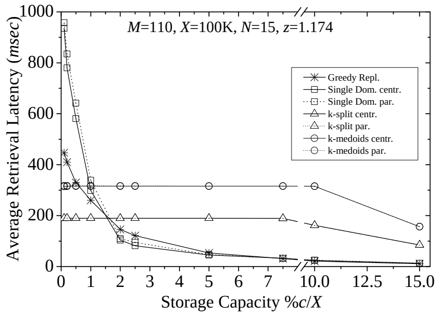


(b) Cache Hit Ratio.

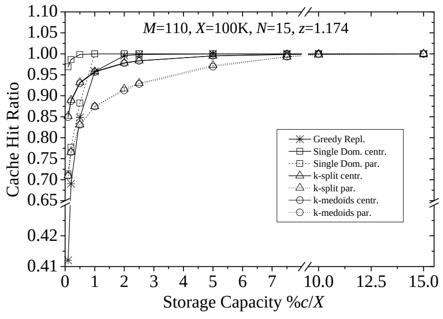


(c) Average Link Stress.

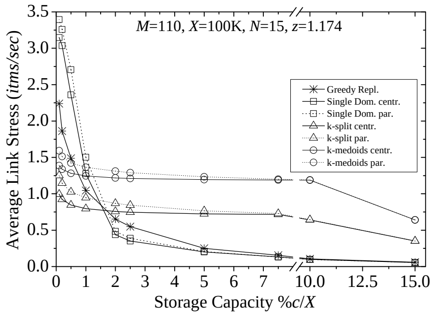
Fig. 7. Performance of the considered placement schemes vs. the total number of clusters.



(a) Average Retrieval Latency.



(b) Cache Hit Ratio.



(c) Average Link Stress.

Fig. 8. Performance of the considered placement schemes vs. the storage capacity of each node.

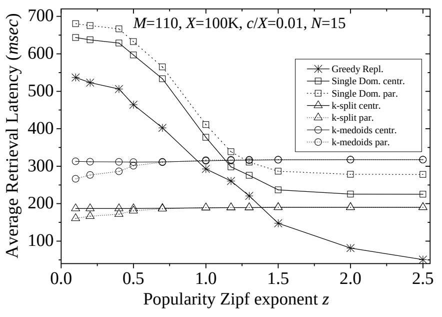
Fig. 7 for the number of clusters N varying from 2 to 40. It is worth noting that we did not further increase the number of clusters given the constraint on the allocation of core nodes to a cluster. However, the extreme case with clusters of one node only ($N = 110$) would represent a fully distributed solution where placement decisions consist in caching the most popular content items locally.

As can be observed, the value of the three metrics decreases for all clustering scenarios as the number of clusters increases. When more clusters are formed, the number of network locations from where a content can be fetched is restricted (*i.e.,* an item is either retrieved from within the cluster or fetched directly from the origin server). Due to the absence of coordination between the clusters, this leads to a drop in terms of cache hit ratio since the probability of retrieving the requested content item from within a cluster with limited caching space is lower. However, as depicted in Fig. 7b, this drop is limited and the value of the cache hit ratio tends towards 0.87 as the number of clusters increases. This relatively high value shows that with the cache capacity considered in this experiment (*i.e.,* $c = 0.01 \cdot X$), the available caching space in most clusters is sufficient for accommodating locally the most popular content items, *i.e.,* the ones incurring significant volume of traffic in the network. As such, content items tend to be retrieved from a closer location on average, which leads to a decrease in the average retrieval latency and average link stress. In addition, it can be observed that the Parallel scheme is more sensitive than the GPS centralized approach with respect to the cache hit ratio, which drops from 1 to around 0.90 when the number of clusters increases up to 4. It should however be noted that the decrease is subsequently very slow (from 0.90 with 4 clusters to around 0.87 with 40 clusters).

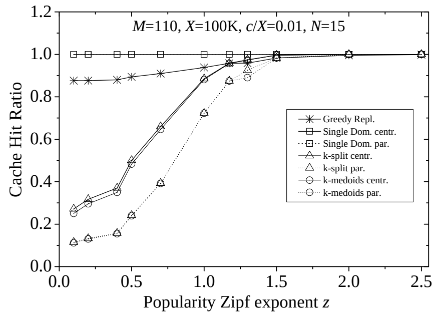
Since the objective of the Greedy approach is to minimize the retrieval delay, it generally obtains better performance in terms of average retrieval latency compared to the rest of the examined schemes. Only when the number of clusters is large, the corresponding placement schemes manage to surpass its performance (Fig. 7a). In contrast, the best performance in terms of cache hit ratio is obtained with the Single Domain GPS Centralized approach. In this experiment, the storage capacity available at each caching location is so that all requested content items can be cached in the network, and, as such, a cache hit ratio of 1 is achieved with GPS. To minimize the average delay, the Greedy algorithm may prefer to fetch some items from the origin server. This can happen in particular if the origin server is closer than a remote network cache location or if the items have lower popularity. Those items are thus not replicated within the domain, which leads to a lower cache hit ratio. It can also be observed that the performance of the Single Domain Parallel scheme is similar to that of the Single Domain GPS scheme.

Finally, while slightly better performance can be achieved in terms of average retrieval latency and average link stress with the *k*-split partitioning method compared to the *k*-medoids one, the actual clustering mechanism has negligible influence on the cache hit ratio.

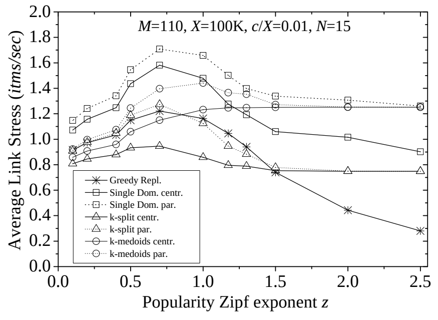
Impact of the cache size: We investigate the impact of the storage capacity available at each caching location. The results are shown in Fig. 8 for storage capacity varying from 0.1% to 15% of the content catalogue X.



(a) Average Retrieval Latency.

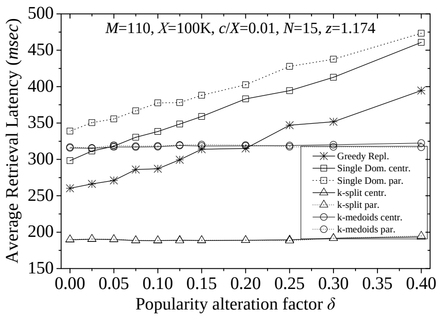


(b) Cache Hit Ratio.

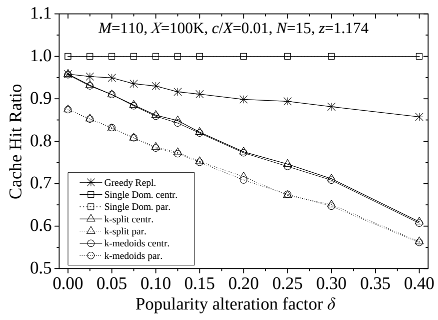


(c) Average Link Stress.

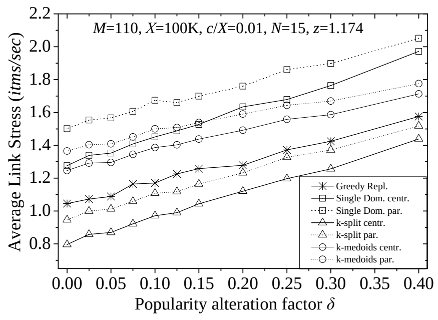
Fig. 9. Performance of the considered performance schemes vs. the Zipf parameter of the content popularity.



(a) Average Retrieval Latency.



(b) Cache Hit Ratio.



(c) Average Link Stress.

Fig. 10. Performance of the considered placement schemes vs. the content popularity alteration factor.

As can be observed, all the schemes tend towards similar performance as the storage capacity available at each location increases. In this case, the number of content items which can be accommodated locally increases and more requests can be served directly from the cache attached to the location where these are received. As a result, similar performance can be achieved by the different schemes.

Impact of the popularity distribution: We investigate the impact of the global popularity distribution of the content, which is driven by the Zipf parameter z. The results are shown in Fig. 9 for z varying from 0.1 to 2.5.

The distribution of the content popularity becomes more uniform as the value of z decreases, which means that the difference between the volume of requests received for popular items and less popular ones decreases. In [7], we showed that in the case of a uniform popularity distribution, a scheme that favours the placement of a larger number of unique content items obtains better performance than a scheme which tends to replicate content. As explained previously, there is no coordination between the clusters. As a result, although content diversity can be achieved at the cluster level, the absence of coordination between clusters means that content items tend to be replicated at the network level, which has a severe impact on the cache hit ratio of the GPS and Parallel schemes with network partitioning. In the extreme case where z is in the order of $0.1-0.2$, the hit ratio drops to $15\%-20\%$. However, measurement-based studies, such as the ones presented in [30], [31], suggest that the Zipf parameter z for web traffic lies in the range of $0.64-0.84$, while in [28] the authors show that this parameter is equal to 1.174 for an online video content provider. This implies that for real world environments, where the distribution of content popularity is less uniform (especially in video delivery systems), the proposed clustering and placement schemes perform significantly well.

In the Single Domain cases, it can be observed that the average retrieval latency decreases when the value of z increases. The absence of replication for low values of z forces the caches to fetch the requested items from remote network locations, leading to higher average retrieval latencies. In the cases with network partitioning, the volume of requests which need to be redirected to the external server is such that the corresponding content placement schemes are less affected by the value of the Zipf parameter. It can be noted, however, that when the value of z increases and replication becomes the prominent placement choice, similar performance in terms of retrieval latency can be achieved with both the Single Domain and Cluster scenarios (in this case popular items are replicated where they are the mostly requested).

Impact of content popularity dynamics: To evaluate the robustness of the different schemes with respect to content popularity dynamics, we investigate the impact of a popularity alteration factor, noted δ, on the three considered metrics. In this experiment, we compute an initial content placement with a fixed value of δ. We then randomly permute the rank of $\delta \cdot M$ content items and evaluate the resulting performance in terms of average retrieval delay, cache hit ratio and average link stress. The results are shown in Fig. 10 for an alteration factor δ varying from 0 to 0.4.

As can be observed, in all cases, the latency increases as the alteration factor increases (Fig. 10a). Compared to Single

Domain approaches, clustered schemes tend to be more robust (the increase is marginal). However, these are more affected in terms of cache hit ratio. Given that the algorithm favors the replication of popular content items, it is very dependent of the content ranking. The alteration of the ranking coupled with the absence of coordination between clusters can therefore lead to severe degradation in terms of performance. From Fig. 10c, we can see that all the schemes are affected in similar proportions in terms of average link stress. An alteration factor of 40% leads to an increase in terms of average link stress of around 60%. Since the cache hit ratio decreases as the alteration factor increases, a larger number of items needs to traverse the network links, which results in an increase of the average link stress.

VI. Related Work

The content placement problem has been investigated in various types of caching infrastructures, ranging from traditional CDN architectures (*e.g.,* [32]) to Video on Demand (*e.g.,* [33]) and IPTV (*e.g.,* [14] [34]) systems, as well as in radically novel environments such as Information-Centric Networks (*e.g.,* [16] [28]).

The common objective of all the proposed content placement approaches is to decide how to replicate content items across different network locations in order to better utilize network resources. Some solutions, such as the ones presented in [32], [35], [36], [33] and [8], have focused on centralized mechanisms, which assume the availability of global information about user demand and network conditions at a centralized location. As a result, this may incur a significant overhead and can have scalability limitations. Other initiatives have proposed distributed approaches but these either rely on the assumption of hierarchical infrastructures (*e.g.,* [37], [14] and [34]), which simplifies the problem, or follow a sequential process (*e.g.,* [16] and [7]), which also has limitations in terms of scalability. A distributed approach has been proposed, in particular, by Laoutaris et al. in [17] in the context of distributed replication groups, where nodes in each group cooperate to take replication decisions that can minimize the overall network cost. Although it is reported that the solution can be implemented in a parallelized fashion, the authors do not elaborate on the actual mechanism. In this paper, we go a step further and design a realistic and practical parallelized decision-making process. A purely distributed approach has also been considered in [32], where placement decisions are taken independently by each node without any level of coordination. Although this can overcome any limitations in terms of scalability, uncoordinated decisions may result in the suboptimal use of the caching space. In contrast, our approach offers a trade-off between scalability and resource usage based on the two-phase reconfiguration process. In addition, scalability is further improved through network partitioning.

To reduce the complexity of replication algorithms, Gkatzikis et al. have proposed in [38] to cluster the items based on their popularity characteristics according to different clustering methods. While this has the advantage of leading to the formation of smaller input content sets and, as such, to improve the scalability, it may compromise the level of flexibility required to perform fine-tuned item allocation. In contrast, partitioning and clustering is applied in our approach at the network level and the characteristics of each individual content item can thus be preserved. It would be however interested to investigate in future work how the performance of the proposed algorithm would be affected when considering content clusters rather than individual items.

To quickly respond to sudden changes in content popularity, reactive replacement strategies such as Least Recently Used (LRU) or Least Frequently Used (LFU) have also been considered in the literature. One of the main advantages of such approaches resides in their low complexity (in terms of implementation and execution) since decisions can be taken independently for each cache. However, as demonstrated in our previous work [8], the lack of coordination between the decision-making points comes at a cost in terms of bandwidth usage. In addition, to capture the evolution of content popularity, reactive schemes require a transitory period during which cache misses can occur, which can represent a non-negligible volume of traffic. In contrast to these reactive approaches, a proactive placement strategy can pre-provision the caches in anticipation to the near future evolution of the interests, resulting as such in fewer cache misses. By nature, the performance of any proactive scheme depends on the prediction accuracy of the requests. To deal with the uncertainties in the future request pattern, a cache management solution that combines proactive placement and reactive approach could be envisioned. Such a scheme was considered in [33] and [39] in which the authors propose to pre-partition the local storage capacity according to fixed ratios in order to implement different caching strategies.

To be supported, our approach implies the existence of new types of collaboration between ISPs and CDNs. Over the last few years, there have been substantial research efforts invested in the development of novel models of interaction between the ISPs and the CDNs (*e.g.,* [5], [6], [40], [41] and [42]). While Jiang et al. have discussed various cooperation models in [5], a new framework to support joint decisions between a CDN and an ISP with respect to the server selection process has been proposed by Frank et al. in [6]. In contrast to these approaches, our solution focuses on empowering ISPs with caching capabilities, which can allow them to implement their own content placement and server selection strategies. ISP-centric caching approaches have also been considered in [41] and [42]. However, the full-blown CDN service to be supported by an ISP, as proposed in these approaches, can incur high operational costs, given that ISPs will have to maintain large storage capacities, and may thus be an economically unviable solution. In [39], Sharma et al. have investigated from a quantitative perspective the interplay between traffic engineering and content distribution in the context of an ISP-operated caching infrastructure. Recently, a SDN-based framework has been proposed by Wichtlhuber et al. in [43] to facilitate the collaboration between the ISP and the CDN. Our approach could also benefit from a SDN/OpenFlow environment as this can provide the ability to better control and manage incoming traffic and requests (*e.g.,* ability to filter traffic based on matching rules).

VII. Summary and Conclusions

In this paper, we propose an efficient and scalable content management approach to control the distribution of content items in an ISP-operated caching infrastructure. The proposed approach relies on a parallelization of the decision-making process, where one instance of the content placement algorithm is executed per caching location. In addition, we investigate how network partitioning can facilitate the implementation of decentralized cache management solutions by limiting the volume of information that needs to be exchanged between distributed decision-making points. It is worth noting that the proposed parallelized process could also apply in centralized settings.

The performance of the proposed approach has been evaluated both in terms of network and caching costs and management overhead. The results show that similar performance in terms of network and caching cost can be achieved by the Parallel algorithm compared to the GPS strategy and that this performance is close to the one achieved by the Greedy approach. Due to the lack of coordination between the formed clusters, network and caching performance can be affected by network partitioning. The worst performance, in terms of cache hit ratio, is achieved with more uniform popularity distributions (Zipf parameter smaller than 0.6), in which case the ratio falls under 0.5. However, partitioning the network into clusters provides a significant gain in terms of management overhead, offering as such better scalability and lower complexity. Compared to the Single Domain approaches, the number of iterations and the number of exchanged messages can be divided by up to a factor 10^4 and 10^6, respectively. Finally, the results suggest that the partitioning method used to form the clusters does not significantly impact the overall performance of the content placement approach, especially in the cases where realistic values regarding the global popularity distribution of the content are used.

In the future, we plan to investigate the benefits of applying content clustering on the performance of the proposed placement strategies. In addition, we will work on the elaboration of additional business models for the interaction between ISP and CDN. Finally, we plan to integrate the proposed cache management framework in a SDN-based environment and investigate how this can facilitate the realization of such an approach.

References

[1] Cisco, "Cisco Visual Networking Index: Global IP Traffic Growth, 20142018," *White Paper*, Feb. 2015.

[2] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante, "Drafting behind akamai (travelocity-based detouring)," in *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 435-446, Aug. 2006.

[3] "Akamai CDN," http://www.akamai.com. [Online; accessed 22-Feb-2016].

[4] "Netflix CDN," http://www.netflix.com. [Online; accessed 22-Feb-2016].

[5] W. Jiang, R. Zhang-Shen, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an ISP network," in *Proc. of SIGMETRICS'09*, Seattle, WA, USA, 2009, pp. 239-250.

[6] B. Frank, I. Poese, G. Smaragdakis, S. Uhlig, and A. Feldmann, "Content-aware traffic engineering," in *Proc. of SIGMETRICS'12*, London, England, UK, 2012, pp. 413-414.

[7] D. Tuncer, M. Charalambides, R. Landa, and G. Pavlou, "More Control Over Network Resources: An ISP Caching Perspective," in *Proc. of IEEE/IFIP International Conference on Network and Service Management (CNSM'13)*, Zurich, Switzerland, Oct. 2013, pp. 26-33.

[8] M. Claeys, D. Tuncer, J. Famaey, M. Charalambides, S. Latre, G. Pavlou, and F. De Turck, "Proactive multi-tenant cache management for virtualized ISP networks," in *Proc. IEEE/IFIP International Conference on Network and Service Management (CNSM'14)*, Nov. 2014, pp. 82-90.

[9] "Coral CDN," http://www.coralcdn.org. [Online; accessed 22-Feb-2016].

[10] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "Donar: decentralized server selection for cloud services," in *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 231-242, aug. 2010.

[11] D. Tuncer, M. Charalambides, G. Pavlou, and N. Wang, "Dacorm: A coordinated, decentralized and adaptive network resource management scheme," in *Proc. of IEEE Network Operations and Management Symposium (NOMS'12)*, Maui, Hawaii, USA, Apr. 2012, pp. 417-425.

[12] D. Tuncer, et al., "A Hybrid Management Substrate Structure for Adaptive Network Resource Management," in *Proc. of ManFI'14*, Krakow, Poland, May 2014, pp. 1-7.

[13] D. Tuncer, M. Charalambides, S. Clayman, and G. Pavlou, "Adaptive Resource Management and Control in Software Defined Networks," in *Network and Service Management, IEEE Transactions on*, vol. 12, no. 1, pp. 18-33, Mar. 2015.

[14] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. of IEEE INFOCOM'10*, San Diego, CA, USA, Mar. 2010, pp. 1-9.

[15] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *Proc. of IEEE INFOCOM'12*, Orlando, Florida, USA, Mar. 2012, pp. 1107-1115.

[16] V. Sourlas, P. Flegkas, L. Gkatzikis, and L. Tassiulas, "Autonomic cache management in information-centric networks," in *Proc. of IEEE Network Operations and Management Symposium (NOMS'12)*, Maui, Hawaii, USA, Apr. 2012, pp. 121-129.

[17] N. Laoutaris, O. Telelis, V. Zissimopoulos, and I. Stavrakakis, "Distributed selfish replication," in *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 12, pp. 1401-1413, Dec. 2006.

[18] A. Brodersen, S. Scellato, and M. Wattenhofer, "Youtube around the world: Geographic popularity of videos," in *Proc. of the International Conference on World Wide Web (WWW'12)*, Lyon, France, 2012, pp. 241250.

[19] Y. Chen, L. Qiu, W. Chen, L. Nguyen, and R. Katz, "Efficient and adaptive web replication using content clustering," in *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 979-994, 2003.

[20] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," in *Theoretical Computer Science*, vol. 38, pp. 293-306, 1985.

[21] L. Kaufman and P. Rousseeuw, "Clustering by Means of Medoids," *Issue 87, Part 3 of Reports of the Faculty of Mathematics and Informatics*, Faculty of Mathematics and Informatics, Delft University of Technology, 1987.

[22] D. Aloise, A. Deshpande, P. Hansen, and P. Popat, "NP-hardness of Euclidean sum-of-squares clustering," in *Machine Learning*, vol. 75, no. 2, pp. 245-248, 2009.

[23] "The Interoute topology, 2010" http://www.topology-zoo.org/maps/Interoute.jpg. [Online; accessed 22-Feb-2016].

[24] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental study of internet stability and backbone failures," in *Proc. of the Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing, Digest of Papers*, Jun. 1999, pp. 278-285.

[25] L. Chiaraviglio, M. Mellia, and F. Neri, "Reducing power consumption in backbone networks," in *Proc. of the IEEE International Conference on Communications (ICC'09)*, Dresden, Germany, Jun. 2009, pp. 1-6.

[26] K. C. Kang, K.-D. Nam, D. J. Jeon, and S. Y. Kim, "Delay characteristics of high-speed Internet access network," in *Proc. of the Asia-Pacific Network Operations and Management Symposium (APNOMS'03)*, Fukuoka, Japan, Oct. 2003.

[27] B. Huffaker, M. Fomenkov, D. J. Plummer, D. Moore, k claffy, "Distance metrics in the Internet," in *IEEE Inter. Telecommunications Symposium (ITS'02)*, 2002, pp. 200-202.

[28] Y. Sun, S. K. Fayaz, Y. Guo, V. Sekar, Y. Jin, M. A. Kaafar, and S. Uhlig, "Trace-Driven Analysis of ICN Caching Algorithms on Video-on-Demand Workloads," in *Proc. of Co-NEXT'14*, Sydney, Australia, 2014, pp. 363-376.

[29] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of web server replicas," in *Proc. of IEEE INFOCOM'01*, Anchorage, Alaska, USA, vol. 3, 2001, pp. 1587-1596.

[30] G. Dán and N. Carlsson, "Power-law revisited: Large scale measurement study of p2p content popularity," in *Proc. of International Conference on Peer-to-peer Systems (IPTPS'10)*, San Jose, CA, USA, 2010, pp. 12-12.
[31] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proc. of IEEE INFOCOM'99*, New York, NY, USA, vol. 1, Mar. 1999, pp. 126-134.
[32] J. Kangasharju, J. Roberts, and K. W. Ross, "Object replication strategies in content distribution networks," in *Comput. Commun.*, vol. 25, no. 4, Mar. 2002, pp. 376-383.
[33] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proc. of Co-NEXT'10*, Philadelphia, Pennsylvania, 2010, pp. 1-12.
[34] J. Dai, Z. Hu, B. Li, J. Liu, and B. Li, "Collaborative hierarchical caching with dynamic request routing for massive content distribution," in *Proc. of IEEE INFOCOM'12*, Mar. 2012, pp. 2444-2452.
[35] I. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," in *SIAM J. Comput.*, vol. 38, no. 4, pp. 1411-1429, Aug. 2008.
[36] T. Bektaş, J.-F. Cordeau, E. Erkut, and G. Laporte, "Exact algorithms for the joint object placement and request routing problem in content distribution networks," in *Comput. Oper. Res.*, vol. 35, no. 12, pp. 3860-3884, Dec. 2008.
[37] M. Korupolu and M. Dahlin, "Coordinated placement and replacement for large-scale distributed caches," in *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 6, pp. 1317-1329, nov./dec. 2002.
[38] L. Gkatzikis, V. Sourlas, C. Fiscione, I. Koutsopoulos, and G. Dán, "Clustered Content Replication for Hierarchical Content Delivery Networks," in *Proc. of IEEE International Conference on Communications (ICC'15)*, London, UK, Jun. 2015.
[39] A. Sharma, A. Venkataramani, and R. K. Sitaraman, "Distributing Content Simplifies ISP Traffic Engineering," in *Proc. of SIGMETRICS '13*, Pittsburgh, PA, USA, 2013, pp. 229-242.
[40] "Content Delivery Networks Interconnection (CDNI)," http://tools.ietf.org/wg/cdni/draft-ietf-cdni-framework/. [Online; accessed 22-Feb-2016].
[41] N. Kamiyama, T. Mori, R. Kawahara, S. Harada, and H. Hasegawa, "ISP-operated CDN," in *Proc. of the IEEE INFOCOM Workshops*, Rio de Janeiro, Brazil, 2009, pp. 49-54.
[42] K. Cho, H. Jung, M. Lee, D. Ko, T. Kwon, and Y. Choi, "How can an ISP merge with a CDN?," in *IEEE Commun. Mag.*, vol. 49, no. 10, pp. 156-162, Oct. 2011.
[43] M. Wichtlhuber, R. Reinecke, and D. Hausheer, "An SDN-Based CDN/ISP Collaboration Architecture for Managing High-Volume Flows," in *Network and Service Management, IEEE Transactions on*, vol. 12, no. 1, pp. 48-60, Mar. 2015.

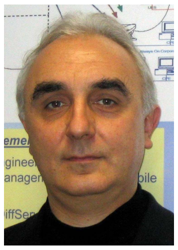
Daphne Tuncher is a postdoctoral researcher in the Department of Electronic and Electrical Engineering at University College London, UK. She received her Ph.D. from the same department in November 2013. Before joining UCL, she studied in France, where she obtained a "Diplôme d'ingénieur de Télécom SudParis" in 2009. Her research interests are in the areas of software-defined networking, network self-management, adaptive network resource management, energy efficiency and cache/content management.

Vasilis Sourlas received his Diploma degree from the Computer Engineering and Informatics Department, University of Patras, Greece, in 2004 and the M.Sc. degree in Computer Science and Engineering from the same department in 2006. In 2013 he received his PhD from the Department of Electrical and Computer Engineering, University of Thessaly (Volos), Greece. In Jan. 2015 he joined the Electronic and Electrical Engineering Department, UCL, London to pursue his two years Marie Curie IEF fellowship. His main interests are in the area of Information-Centric Networks and Future Internet.

Marinos Charalambides is a senior researcher at University College London. He received a BEng (First Class Hons.) in Electronic and Electrical Engineering, a MSc (Distinction) in Communications Networks and Software, and a Ph.D. in Policy-based Management, all from the University of Surrey, UK, in 2001, 2002 and 2009, respectively. He has been working in a number of European and UK national projects since 2005 and his research interests include network programmability, content distribution, energy-aware networking and online traffic engineering. He has been the technical program chair of several events and serves in the committees of the main network and service management conferences.

Maxim Claeys obtained a masters degree in computer science from Ghent University, Belgium, in June 2012. In August 2012, he joined the Department of Information Technology at Ghent University, where he is active as a Ph.D. student. His main research interests are the application of autonomic network management approaches in multimedia delivery. The focus of this research is mainly on the end-to-end Quality of Experience optimization, ranging from the design of autonomous clients to intelligent in-network decision taking.

Jeroen Famaey received his Ph.D. degree from Ghent University, Belgium in June 2012. From July 2012 until September 2014 he worked as a post-doctoral researcher in the area of network and service management, affiliated with Ghent University and iMinds. Since October 2014 he has been working as a post-doctoral researcher on control of highly dynamic wireless networks at the University of Antwerp, Belgium. He has co-authored around 50 papers in international peer-reviewed journals and conference proceedings. His current research interests include resource scheduling, routing and security in energy-constrained wireless networks.



George Pavlou is Professor of Communication Networks in the Department of Electronic and Electrical Engineering, University College London, UK where he co-ordinates research activities in networking and network management. He received a Diploma in Engineering from the National Technical University of Athens, Greece and M.S. and Ph.D. degrees in Computer Science from University College London, UK. His research interests focus on networking and network management, including aspects such as traffic engineering, quality of service management, policy-based systems, autonomic networking, information-centric networking and software-defined networks. He has been instrumental in a number of European and UK research projects that produced significant results with real-world uptake and has contributed to standardisation activities in ISO, ITU-T and IETF. He has been the technical program chair of several conferences and in 2011 he received the Daniel Stokesbury award for "distinguished technical contribution to the growth of the network management field".

Filip De Turck leads the network and service management research group at the Department of Information Technology of the Ghent University, Belgium and iMinds (Interdisciplinary Research Institute in Flanders). He (co-) authored over 450 peer reviewed papers and his research interests include telecommunication network and service management, and design of efficient virtualized network systems. In this research area, he is involved in several research projects with industry and academia, serves as the Vice-Chair of the IEEE Technical Committee on Network Operations and Management (CNOM), chair of the Future Internet Cluster of the European Commission, and is on the TPC of many network and service management conferences and workshops and serves in the editorial board of several network and service management journals.