Caching and Operator Cooperation Policies for Layered Video Content Delivery

Konstantinos Poularakis, George Iosifidis, Antonios Argyriou, Iordanis Koutsopoulos and Leandros Tassiulas

***Abstract*—Distributed caching architectures have been proposed for bringing content close to requesters and the key problem is to design caching algorithms for reducing content delivery delay. The problem obtains an interesting new twist with the advent of advanced layered-video encoding techniques such as Scalable Video Coding (SVC). We show that the problem of finding the caching configuration of video encoding layers that minimizes average delay for a network operator is NP-Hard, and we establish a pseudopolynomial-time optimal solution using a connection with the multiple-choice knapsack problem. We also design caching algorithms for multiple operators that cooperate by pooling together their co-located caches, in an effort to aid each other, so as to avoid large delays due to downloading content from distant servers. We derive an approximate solution to this cooperative caching problem using a technique that partitions the cache capacity into amounts dedicated to own and others' caching needs. Numerical results based on real traces of SVC-encoded videos demonstrate up to 25% reduction in delay over existing (layer-agnostic) caching schemes, with increasing gains as the video popularity distribution gets steeper, and cache capacity increases.**

***Index Terms*—Caching, Cooperation, Layered-video encoding.**

I. Introduction

On-demand video is the driving force of the data tsunami that we are witnessing nowadays [1], and one of the main revenue sources for wireline and wireless network operators and providers. Therefore, it is critical for network operators to satisfy this increasing volume of video requests with the minimum possible delay. A method to achieve this goal is to cache video content as close as possible to end-users. Such distributed caching architectures have been proposed for content delivery networks (CDNs) and Telco-CDNs [2], and recently also for cellular networks [3].

A key challenge in these systems is to devise the *optimal caching policy*: for a given anticipated content demand, determine which content files should be placed in each cache, so as to reduce the average content delivery delay for all requests. These requests, if not satisfied by the local available cache, necessitate fetching content from distant back-end servers, which induces significantly larger delay. This is a well known NP-hard problem, and many heuristic or approximation algorithms have been proposed to address it [2], [3], [4].

This work was supported partly by the European Commission Horizon 2020 Research Programme under grant no 671551 (5G-XHAUL project), and by the US Office of Naval Research (ONR) under award N00014-14-1-2190.

K. Poularakis and A. Argyriou are with the Department of Electrical and Computer Engineering, University of Thessaly, Greece (e-mail: {kopoular, anargyr}@uth.gr). G. Iosifidis and L. Tassiulas are with the Department of Electrical Engineering and Institute for Network Science, Yale University, USA (e-mail: {georgios.iosifidis, leandros.tassiulas}@yale.edu). I. Koutsopoulos is with the Department of Informatics, Athens University of Economics and Business, Greece (e-mail: jordan@aueb.gr).

Nevertheless, a specific aspect has been overlooked. Today more often than not, networks deliver video files encoded at different *qualities* to their customers. Users may implicitly or explicitly ask for certain video quality (e.g., certain resolution for YouTube videos [5]), while in other cases the delivered video quality is determined by the operator (e.g., based on agreements with content providers [6]).

These developments together with stringent requirements for higher user quality of experience (QoE) and advances in video-encoding technology have led to incorporation of advanced video encoding techniques, which in turn, affect the performance of existing caching algorithms. One such encoding technique is Scalable Video Coding (SVC) [7], which allows for multiple spatial resolutions (screen sizes), different frame rate, or signal-to-noise ratio (SNR) qualities. With SVC, each video file is encoded in a set of segments, the *layers*, which, when combined, achieve the requested video quality. A user asking the lowest video quality receives only the basic layer (layer 1), while users asking for higher qualities receive *multiple* layers, starting from layer 1 up to the highest necessary one to achieve that quality. SVC is considered today one of the emerging video technologies [8], and it is already used for video streaming [9], [10], web services [11], and video storage [12], among other applications.

With SVC, it is possible to store different layers of a certain video in different caches. For a user that requests a video at a given quality level, the different layers needed are received, decoded and set to play *at the same time*, rather than serially. In this setting, video delivery is constrained by the layer delivered *last*, and hence the delay metric is determined by the *largest* delay needed to deliver a layer among all layers required from a cache or a server[1]. Due to SVC, the repertoire of caching policies increases significantly, as the caching decisions must be taken per layer and not per video file. Hence all previous theoretical results (e.g., approximation ratios) need to be revisited, as those caching algorithms do not take into account layered video content and interdependencies among different layers that all need to be fetched, possibly from different caches, so as to achieve the requested video quality. Although SVC has already been studied for various network architectures [14]-[17], none of these heuristics or simulation-based studies provides optimized solutions and/or approximation ratio guarantees against optimal caching policies. *In this work, we address precisely the problem of minimizing user perceived video delivery delay for a network operator through*

[1]Notice that we study the case of delivering videos at certain qualities asked by users. Hence, we do not consider adding or dropping layers in real-time to handle bandwidth fluctuations and improve video streaming experience [13].

optimized caching layered video content.

Moreover, going one step further, we study the delay performance benefits that may arise when different network operators cooperate in SVC caching. Today there exist many market entities (e.g., different Telco-CDNs) that often deploy their own caches in the same locations so as to serve their users-clients. The caches may be amenable to joint coordination [18]. Thus, it is meaningful to explore the potential of a local cache of a certain network entity to retrieve a video layer from the co-located cache of a different network entity, instead of fetching it from a distant server of its own that would cause larger delay. However, the diverse user demands that different networks must serve render this cooperative caching problem particularly challenging. *The second problem we tackle is to derive a joint caching policy that minimizes the total delay for all network operators, considering the global content demand.*

Methodology and Contributions. We consider a distributed caching architecture comprising several *local nodes* such as small cells or base stations, in the proximity of end-users. Requests for SVC-encoded video files in different quality levels are randomly generated by users that are associated to these local nodes. A request can be satisfied by the local node if it has cached the complete set of required layers. Otherwise, the missing layers are fetched from a distant content server, and this introduces additional delay.

Our first goal is to design the optimal caching policy for such a network, aiming to minimize the average delay for delivering the entire videos to users. This is a challenging problem since *taking decisions per layer adds up to the complexity of traditional caching problems where copies of the entire videos are cached*. We show that this problem is NP-hard and develop a pseudopolynomial-time optimal as well as a Fully Polynomial Time Approximation (FPTA) algorithm using a connection with the *multiple-choice knapsack (MCK) problem* [19].

Next, we introduce the problem of cooperation of different network operators in such distributed caching architectures, where the goal is to derive a joint caching policy that minimizes total delay for all networks. We assume that users of the different networks request the same set of video files (or, a common subset) with possibly different rates and quality requirements. Therefore, the cooperative policy may reduce the average delay for users of some networks, and increase it for some others. Using a technique that *partitions cache space of a cache owned by an operator into two parts, dedicated to own and other operator content respectively,* we present a solution algorithm with established approximation ratio.

The contribution of this work can be summarized as follows:

- *Layered Video Caching*. We introduce the problem that derives per-video-layer caching policies, aimed at optimizing average user delay in a distributed caching network. This is a new caching problem, with no analytical results to date, yet of increasing importance due to the momentum of SVC encoding. We reduce this to the MCK problem and provide a pseudopolynomial-time optimal and a FPTA algorithm [19].
- *Operator Cooperation*. We propose cooperation policies among different network operators and formulate the respective optimization problem for devising the globally optimal caching policy. Using a cache-partition technique, we establish an approximation algorithm achieving at least half of the optimal performance for a symmetric case with equal transmission rates of the links between nodes.
- *Trace-driven Evaluation.* We evaluate numerically the proposed schemes using system parameters driven from real traces. We show that our approach reduces average delivery delay up to 25% over existing schemes, with increasing gains as the video popularity distribution gets steeper, and cache capacity increases.

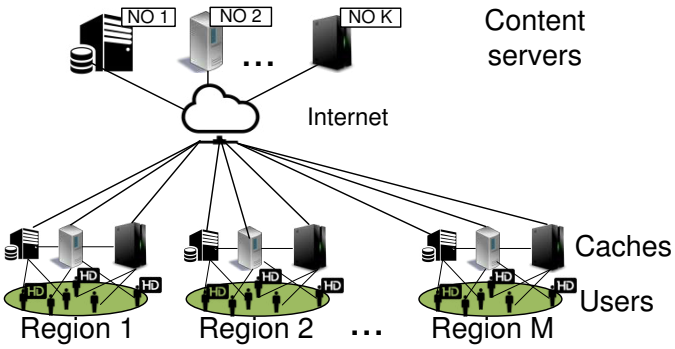


Fig. 1. A distributed caching architecture with K NOs and M regions. Every cache is connected with a distant content server and possibly with other caches in the same region.

The rest of the paper is organized as follows. Section II describes the system model and formalizes the layered video caching problem. Sec. III and Sec. IV describe our solution algorithms when network operators serve their requests independently from each other and when they cooperate respectively. Sec. V presents the numerical results, while Sec. VI reviews our contribution compared to related works. We conclude our work in Sec. VII.

II. System model and Problem Statement

We consider a general network architecture wherein a set $\mathcal{K}$ of K Network Operators (NOs), or network entities, provide internet access to their subscribers, or users, distributed in a set $\mathcal{M}$ of M geographical regions. For each region, each NO may have installed a cache at certain location along the path from its subscribers to the back-end content servers. The NOs may act independently or in cooperation [18]. An example caching network is depicted in Figure 1 and the key notation is summarized in Table I.

A. Independent Caching by Network Operators

We first consider the case where NOs act independently from each other and focus on a single NO $k \in \mathcal{K}$. We denote with $\mathcal{N}_k$ the set of caches, or cache-nodes, of NO k, each located at a different region. The capacity of cache $n \in \mathcal{N}_k$ is denoted with $C_n \geq 0$ (bytes). The average user demand for each video in a set $\mathcal{V} = \{1, 2, ..., V\}$ of V video files and within a certain time period (e.g., a few hours or days)

is assumed to be fixed and known in advance, as in [2], [3]. For example, the demand can be learned by analyzing previous time statistics of user request patterns to infer future demand [20]. Each video can be delivered with $Q \in \mathcal{Z}^+$ quality levels, indexed in set $\mathcal{Q} = \{1, 2, ..., Q\}$. Namely, there is a set $\mathcal{L} = \{1, 2, ..., Q\}$ of Q layers for each video, which when accrued realize the different quality levels; layer 1 by itself realizes quality 1, layer 1 combined with layer 2 realize quality 2, and so on. The size of the l^{th} layer of video v is denoted with $o_{vl} > 0$ (bytes), which typically decreases with l, i.e., $o_{v1} \geq o_{v2} \geq ... \geq o_{vQ}$ [7], [21].

User requests for videos in $\mathcal{V}$ with possibly different qualities arrive at the nodes in $\mathcal{N}_k$. For example, there may exist $Q = 2$ quality levels, and half of the users request videos at the low-definition quality ($q = 1$), while the other half ask for high-definition (HD) quality ($q = 2$). Let $\lambda_{nvq} \geq 0$ be the average user demand associated with node n for the q^{th} quality level of video v. We define the request vector for each cache node n, and the total demand vector for NO k, respectively:

$$\lambda_n = (\lambda_{nvq} : v \in \mathcal{V}, q \in \mathcal{Q}), \quad \boldsymbol{\lambda}_k = \big(\lambda_n : n \in \mathcal{N}_k\big). \quad (1)$$

In order to deliver to a user the q^{th} quality of video v, *all* layers of that video from layer 1 up to q need to be delivered, i.e., $\sum_{l=1}^{q} o_{vl}$ bytes in total. In a streaming video system, segments of the different layers are received, decoded and set to play *at the same time*, rather than serially. In this setting, video delivery is constrained by the layer delivered last, and hence *the delay for delivering the entire video will be equal to the maximum delay needed for each of these layers to be delivered.*

Ideally, the user would like to receive all required layers from the local node n which leads to the lowest delay possible. Without loss of generality, we assume this reference delay to be zero. If a layer cannot be found locally, node n can fetch it from a distant content server that contains all videos and layers. Similarly to the works in [2], [3], [4], we consider this fetching to induce on average a large per unit data delay of d_n seconds, which depends on cache location. In other words, each layer requested from the server will be delivered with an average rate that is constant and given by $1/d_n$. This for example can be realized by using parallel TCP connections [22], one connection for each layer, with fixed bandwidth allocated per connection.

Let the binary decision variable x_{nvl} indicate whether the l^{th} layer of video v will be placed at node n ($x_{nvl} = 1$) or not ($x_{nvl} = 0$). Then, the *caching policy* for NO k is given by the vector:

$$\boldsymbol{x}_k = \big(x_{nvl} : \forall n \in \mathcal{N}_k, v \in \mathcal{V}, l \in \mathcal{L}\big). \quad (2)$$

Clearly, each node $n \in \mathcal{N}_k$ cannot cache more data than its capacity:

$$\sum_{v \in \mathcal{V}} \sum_{l \in \mathcal{L}} o_{vl} x_{nvl} \leq C_n. \quad (3)$$

Our goal is to devise the caching policy that minimizes the aggregate delay for all users of NO k, denoted with $J_k(\boldsymbol{x}_k)$:

$$J_k(\boldsymbol{x}_k) = \sum_{n \in \mathcal{N}_k} \sum_{v \in \mathcal{V}} \sum_{q \in \mathcal{Q}} \lambda_{nvq} \cdot \max_{l \in \{1,...,q\}} \Big\{(1 - x_{nvl}) \cdot o_{vl} \cdot d_n\Big\}, \quad (4)$$

where the delay for delivering layer l of video v is zero if this layer is cached at the local node n (i.e., $x_{nvl} = 1$); otherwise the delay is $o_{vl} \cdot d_n$. The delay for delivering the entire video v at quality level q equals to the *maximum* of the delays needed to deliver layers 1 to q.

TABLE I
KEY NOTATIONS

Symbol	Physical Meaning
$\mathcal{K}$	Set of K network operators (NOs)
$\mathcal{V}$	Set of V video files
$\mathcal{Q}$	Set of Q qualities
$\mathcal{L}$	Set of L layers
$\mathcal{M}$	Set of M geographical regions
$\mathcal{N}$	Set of cache-nodes
$\mathcal{N}_k$	Cache-nodes belonging to NO k
$\mathcal{N}_m$	Cache-nodes located at region m
$\mathcal{M}_n$	Region where cache-node n is located
C_n	Cache capacity at node n (bytes)
λ_{nvq}	Average demand at node n for video v at quality q
o_{vl}	Size of layer l of video v (bytes)
d_n	Per unit data delay for serving requests at node n by a server
$d_{nn'}$	Per unit data delay for serving requests at node n by node n'
x_{nvl}	Caching decision for layer l of video v to node n
$J_k(\boldsymbol{x}_k)$	The aggregate user delay for NO k in independent setting
$J_k^c(\boldsymbol{x})$	The aggregate user delay for NO k in cooperative setting

B. Cooperative Caching among Network Operators

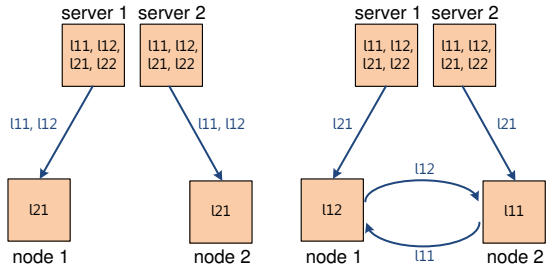
Let us now consider the case that the NOs have decided to jointly coordinate their different caches in the *same* region[2] [18]. Therefore, one cache can send video layers to the other to satisfy the other's demand. Assume that each NO in $\mathcal{K}$ serves requests for the same set $\mathcal{V}$ of videos[3]. Nevertheless, each NO has its own clients and may need to serve different demand, i.e., $\boldsymbol{\lambda}_{k_1} \neq \boldsymbol{\lambda}_{k_2}$. We define the set of all cache nodes $\mathcal{N} = \bigcup_{k \in \mathcal{K}} \mathcal{N}_k$ and the total expected demand $\Lambda = \bigcup_{k \in \mathcal{K}} \boldsymbol{\lambda}_k$.

If a layer cannot be found at the local cache node n, then n can download it from another node n' in the *same* region that has already cached it. We denote with $d_{nn'}$ the per unit data delay incurred for this transfer, where it trivially holds that $d_{nn} = 0$, $\forall n \in \mathcal{N}$. As a last resort for node n, the content server can deliver the layer with delay $d_n > d_{nn'}$, $\forall n, n'$ in the same region. Clearly, a user may download the required layers from different caches or servers. The user experienced delay will be equal to the *maximum* of the respective delays.

The objective of the cooperating NOs is to minimize the total average video delivery delay for satisfying the entire set of requests Λ. We denote the joint caching policy by $\boldsymbol{x} = (\boldsymbol{x}_k : k \in \mathcal{K})$. Then, the total delay can be written as $J_T^c(\boldsymbol{x}) =$

[2]Note that still cache-nodes at different regions act independently each other.

[3]Note that our model captures also the case that the networks provide different, yet overlapping sets of videos, in which case $\mathcal{V}$ stands for the overlapping video set.



(a) Independent Caching. (b) Cooperative Caching.

Fig. 2. An example illustrating the benefits of cooperative caching for two network operators.

$\sum_{k\in\mathcal{K}} J_k^c(\boldsymbol{x})$, where:

$$J_k^c(\boldsymbol{x}) = \tag{5}$$
$$= \sum_{n\in\mathcal{N}_k}\sum_{v\in\mathcal{V}}\sum_{q\in\mathcal{Q}} \lambda_{nvq} \max_{l\in\{1,\ldots,q\}} \Big\{ \prod_{\substack{n'\in\mathcal{N}:\\ \mathcal{M}_{n'}=\mathcal{M}_n}} (1-x_{n'vl}) o_{vl} d_n$$
$$+ \Big(1 - \prod_{\substack{n'\in\mathcal{N}:\\ \mathcal{M}_{n'}=\mathcal{M}_n}} (1-x_{n'vl})\Big) o_{vl} \min_{\substack{n'\in\mathcal{N}:\\ \mathcal{M}_{n'}=\mathcal{M}_n,\ x_{n'vl}=1}} \{d_{nn'}\}\Big\}.$$

Here, $\mathcal{M}_n \in \mathcal{M}$ indicates the region where node n is located. Every required layer $l \in \{1, ..., q\}$ will be delivered to local node n by the content server with per unit data delay d_n if none of the nodes in the same region with n have cached it, i.e., if $\prod_{n'\in\mathcal{N}:\mathcal{M}_{n'}=\mathcal{M}_n}(1-x_{n'vl}) = 1$. Otherwise, among the nodes that have cached l, the one with the lowest delay will deliver it.

Motivating Example. The benefits that such cooperation policies may yield can be easily understood through the simple example in Figure 2. There exist $V = 2$ videos and $Q = 2$ quality levels. The latter can be realized by combining $L = 2$ layers per video; l_{11}, l_{12} for video 1, and l_{21}, l_{22} for video 2. Each layer is of size 1 (based on some normalized size scale). There is also a region with two nodes, indexed by 1 and 2, that belong to two different NOs. Each node is equipped with a unit-sized cache. The delay coefficients are: $d_1 = d_2 = 2$ and $d_{12} = d_{21} = 1$. The demand at node 1 is given by: $\lambda_{111} = 0$, $\lambda_{112} = 10$, $\lambda_{121} = 1$, $\lambda_{122} = 0$, while at node 2 it is: $\lambda_{211} = 9$, $\lambda_{212} = 9$, $\lambda_{221} = 10$, $\lambda_{222} = 0$.

Ideally, each node would store the two layers of video 1 (l_{11}, l_{12}) and the first layer of video 2 (l_{21}) in order to serve all its requests locally. However, this is not possible due to the cache capacity limitations. When NOs operate independently from each other, we can show that the optimal caching policy dictates both nodes to cache l_{21}. The total delay will be: $\lambda_{112}\cdot d_1 + \lambda_{211}\cdot d_2 + \lambda_{212}\cdot d_2 = 56$. Here, we note that caching l_{12} at node 1 would not improve user delay at all, since l_{11} layer would still be delivered by the content server yielding $\lambda_{112}\cdot d_1$ delay. However, if NOs cooperate, then the *optimal caching policy changes*; it places l_{12} to node 1 and l_{11} to node 2. Now, the cached layers are *different* between the two nodes. Hence, they can be exchanged to reduce further delay. The total delay will be: $\lambda_{112}\cdot d_{12} + \lambda_{121}\cdot d_1 + \lambda_{212}\cdot d_{21} + \lambda_{221}\cdot d_2 = 41 < 56$.

Before we present our caching solutions, we remark that our model considers the average delay for delivering the *entire* video to the user that requests it. This delay will directly impact the video streaming process through the necessary *startup delay* that is introduced at the video decoder of the user. We provide more details about this issue in our online technical report [23].

III. Independent Caching by Network Operators

In this section, we address the layered video caching problem for the case that different network operators devise independently their caching policies. Specifically, each NO k solves the following problem:

$$\min_{\boldsymbol{x_k}} \quad J_k(\boldsymbol{x}_k) \tag{6}$$
$$s.t. \quad \sum_{v\in\mathcal{V}}\sum_{l\in\mathcal{L}} o_{vl} x_{nvl} \le C_n, \forall n \in \mathcal{N}_k, \tag{7}$$
$$x_{nvl} \in \{0,1\},\ \forall n \in \mathcal{N}_k,\ v \in \mathcal{V},\ l \in \mathcal{L}. \tag{8}$$

The local nodes of a NO k are in different regions and they cannot send content each other. Hence, caching decisions at a node $n \in \mathcal{N}_k$ do not affect the rest and the problem can be decomposed into $|\mathcal{N}_k|$ *independent subproblems*, one for each node. For a specific node $n \in \mathcal{N}_k$, we note that without caching the aggregate user delay would be $\sum_{v\in\mathcal{V}}\sum_{q\in\mathcal{Q}} \lambda_{nvq} o_{v1} d_n$. This is because, all requests are served by the remote server (with per unit data delay d_n), and video delivery is constrained by the largest layer, i.e., layer $l = 1$. Caching can reduce the aggregate delay by serving a fraction of the requests locally. Namely, caching only layer $l = 1$ of a video v ensures that the delay will be reduced by $\sum_{q\in\mathcal{Q}} \lambda_{nv}^q \cdot d_n \cdot (o_{v1} - o_{v2})$, since $l = 2$ will be the layer delivered last. In the same sense, caching both $l = 1$ and $l = 2$ layers, moves the bottleneck point for video delivery to the layer $l = 3$, thus reducing the delay by $\sum_{q\in\mathcal{Q}} \lambda_{nv}^q \cdot d_n \cdot (o_{v2} - o_{v3})$ more, and so on. Hence, the equivalent problem of *maximizing the delay savings for node* n (named P_n) can be expressed as follows:

$$P_n: \max_{\boldsymbol{x_n}} \quad \sum_{v\in\mathcal{V}}\sum_{q\in\mathcal{Q}} \lambda_{nvq} d_n \sum_{l=1}^{q} (o_{vl} - o_{vl+1}) \prod_{i=1}^{l} x_{nvi}$$
$$s.t. \quad \text{constraint: } (3),$$
$$x_{nvl} \in \{0,1\},\ \forall v \in \mathcal{V},\ l \in \mathcal{L}, \tag{9}$$

where $\mathbf{x_n} = (x_{nvl} \in \{0,1\} : \forall v \in \mathcal{V}, l \in \mathcal{L})$, and, with a slight abuse of notation, we set o_{vl+1} to be equal to zero for $l = q$ in the above summation.

Solution: Subsequently, we characterize the complexity of problem P_n, and present efficient solutions. Due to space limitations, readers are requested to refer to the online technical report [23] for the detailed proofs of the theorems and lemmas, while we have included in this document the complete proof of the main result of this work (Theorem 3). We first prove the intractability of the P_n problem in Theorem 1.

Theorem 1: Problem P_n is NP-Hard.

Sketch of proof: The proof is based on a reduction from the Knapsack problem, which is NP-Hard. Given a knapsack of limited capacity and a set of items with nonnegative weights and values, the question is how to place items in the knapsack

to maximize the total value of the packed items [19]. We create an equivalent instance of the P_n problem where there is a cache of capacity equal to the knapsack, one video for each item, each video has one layer of size equal to the weight of the mapped item, and local demand equal to the value of that item. ■

The following lemma is needed.

Lemma 1: In the optimal solution of P_n, layer l of a video should not be cached unless all previous layers $l' < l$ of that video have been cached.

Proof: Let us assume that there is an optimal solution to P_n that caches at node n the layer l of video v without caching a layer $l' < l$ of the same video. Then, removing l from the cache n would have no impact on the objective value of P_n, since the users that download l from n need to download also l' from the content server, which incurs delay $o_{vl'} \cdot d_n \geq o_{vl} \cdot d_n$. Filling the cache space left free with a layer of another -previously uncached- video would improve the objective value of P_n. This contradicts the assumption. ■

Inspired by Lemma 1, we identify a connection of problem P_n to the following variant of the knapsack problem [24]:

Definition 1: Multiple-Choice Knapsack (MCK): Given R classes E_1, E_2,...,E_R of items to pack in a knapsack of capacity W, where the i^{th} item in class E_r has value p_{ri} and weight w_{ri}, choose *at most one* item from each class such that the total value is maximized without the total weight exceeding W.

Then, we describe the connection between problem P_n and MCK problems in the following lemma.

Lemma 2: The problem P_n is polynomial-time reducible to the MCK problem.

Sketch of proof: Given an instance of the P_n problem, we construct the equivalent instance of the MCK problem as follows. There is a knapsack of capacity C_n and V classes of items (one class for each video), each one with Q items (one item for each quality). The i^{th} item in the v^{th} class has weight $\sum_{l=1}^{i} o_{vl}$ and value $\sum_{q\in\mathcal{Q}} \lambda_{nvq} d_n \sum_{l=1}^{q} (o_{vl} - o_{vl+1}) \prod_{j=1}^{l} (\mathbf{1}_{\{j\in\{1,2,\ldots,i\}\}})$, where $\mathbf{1}_{\{c\}} = 1$ if condition $c =$ true; else zero, and $o_{vl+1} = 0$ for $l = q$. If the i^{th} item of the v^{th} class is packed in the knapsack, we place the i first layers, i.e., layers 1 to i, of video v to cache n. The solution caches no more data than C_n and satisfies Lemma 1. ■

Lemma 2 provides a valuable result, since it paves the way for exploiting a wide range of efficient algorithms that have been proposed for the MCK problem in order to solve problem P_n. Specifically, although MCK problem is NP-hard, there exists a *pseudopolynomial-time optimal* algorithm and a *fully-polynomial-time approximation* (FPTA) algorithm to solve it [24]. Pseudopolynomial means that the time is polynomial in the input (knapsack capacity and item weights), but exponential in the length of it (number of digits required to represent it). The FPTA algorithm finds a solution with a performance that is provable no less than $(1-\epsilon)$ times the optimal, while its running time is polynomial to $\frac{1}{\epsilon}$, $\epsilon \in (0,1)$. Therefore, the FPTA algorithm complexity and performance are adjustable, which makes it preferable compared to the first algorithm for large problem instances. Hence, we obtain the following result:

Theorem 2: There exists a pseudopolynomial-time optimal algorithm and a FPTA algorithm for problem P_n.

IV. Cooperative Caching Among Network Operators

In this section, we focus on the layered video caching problem when multiple network operators come in offline agreement to cooperate. We stress again that cooperation amounts to putting together their local pools of resources (caches in our case) in order to cache layered video destined also for users of other networks. The problem of determining the caching policy that *minimizes the total user delay of all NOs* can be expressed as follows:

$$\min_{\mathbf{x}} \quad J_T^c(\boldsymbol{x}) \tag{10}$$

$$s.t. \quad \sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} o_{vl} x_{nvl} \leq C_n, \forall n \in \mathcal{N}, \tag{11}$$

$$x_{nvl} \in \{0,1\}, \ \forall n \in \mathcal{N}, v \in \mathcal{V}, l \in \mathcal{L}, \tag{12}$$

where $\mathbf{x} = (x_{nvl} : \forall n \in \mathcal{N}, v \in \mathcal{V}, l \in \mathcal{L})$.

Decomposition: Since content can only be transferred between nodes in the *same* region, the above problem can be decomposed into M *independent subproblems*, one for each region $m \in \mathcal{M}$. We denote with $\mathcal{N}_m \subseteq \mathcal{N}$ the set of nodes located at region m. For a specific region m, we observe that the total user delay without caching would be $D_{wc}^m = \sum_{n\in\mathcal{N}_m} \sum_{v\in\mathcal{V}} \sum_{q\in\mathcal{Q}} \lambda_{nvq} o_{v1} d_n$, since all requests are served with layer 1 (which is the largest among all layers) downloaded by the content servers. Caching can reduce the total delay by delivering some of the required layers by the caches instead of the servers. We can express the equivalent problem of *maximizing delay savings for region* m (named R_m) as follows:

$$R_m: \ \max_{\boldsymbol{x_m}} D_{wc}^m - \sum_{n\in\mathcal{N}_m} \sum_{v\in\mathcal{V}} \sum_{q\in\mathcal{Q}} \lambda_{nvq} \max_{l\in\{1,\ldots,q\}} \Big\{ \prod_{n'\in\mathcal{N}_m} (1 - x_{n'vl}) o_{vl} d_n + (1 - \prod_{n'\in\mathcal{N}_m} (1 - x_{n'vl})) o_{vl} d_{nn^*} \Big\}$$

$$s.t. \sum_{v\in\mathcal{V}} \sum_{l\in\mathcal{L}} o_{vl} x_{nvl} \leq C_n, \forall n \in \mathcal{N}_m \tag{13}$$

$$x_{nvl} \in \{0,1\}, \ \forall n \in \mathcal{N}_m, \ v \in \mathcal{V}, \ l \in \mathcal{L} \tag{14}$$

where $\boldsymbol{x}_m = (x_{nvl} : n \in \mathcal{N}_m, v \in \mathcal{V}, l \in \mathcal{L})$. Here, a required layer l of a video v will be delivered to node n by the content server with delay $o_{vl} d_n$ if none of the nodes have cached it, i.e., if $\prod_{n'\in\mathcal{N}_m} (1 - x_{n'vl}) = 1$. Otherwise, among the nodes that have cached l, the one with the lowest delay will deliver it, i.e., the node $n^* = \text{argmin}_{n'\in\mathcal{N}_m : x_{n'vl}=1} \{d_{nn'}\}$.

Solution: R_m is a very challenging problem, since the already NP-Hard problem P_n defined in the previous section is further perplexed in order to account for all the scenarios of cooperation among the nodes in the same region, i.e., $\forall n \in \mathcal{N}_m$. Namely, each node should seek the best tradeoff between caching the layers of the videos that are popular for its own users (*optimizing local demand*), and caching the ones that are frequently requested by users of other nodes in the same region (*optimizing global demand*). Subsequently,

we present an algorithm that achieves an approximation ratio for this important problem. The algorithm partitions the cache space of each node based on an *input parameter* $F \in [0,1]$. Here, F stands for the portion of each cache that is filled in with globally popular video content, while the rest $1-F$ portion is filled in with locally popular video content. Clearly, if $F=0$, then each node n caches the locally popular video layers independently from the others (i.e., by solving problem P_n), while when $F=1$ all nodes put together their caches and they fill in the union cache space with globally popular video layers.

The proposed algorithm uses as components the solution to the following two problems:

1. MCK(m): The instance of the MCK problem comprising a knapsack of capacity $F \cdot \sum_{n\in\mathcal{N}_m} C_n$ and V classes of items, each with Q items. The i^{th} item of the v^{th} class has weight $\sum_{l=1}^{i} o_{vl}$ and value $\sum_{n\in\mathcal{N}_m}\sum_{q\in\mathcal{Q}} \lambda_{nvq} d_n \sum_{l=1}^{q}(o_{vl} - o_{vl+1}) \prod_{j=1}^{l}(\mathbf{1}_{\{j\in\{1,2,\ldots,i\}\}})$, where $\mathbf{1}_{\{.\}}$ is the indicator function, i.e. $\mathbf{1}_{\{c\}}=1$ if condition c is true; otherwise it is zero, and $o_{vl+1}=0$ for $l=q$. Here, the i^{th} item of the v^{th} class corresponds to the first i layers of video v.

2. $P_n(\mathcal{A}_n)$: The instance of the P_n problem in which the layers in the set $\mathcal{A}_n$ are already placed in cache n.

We now present the proposed Layer-aware Cooperative Caching (LCC) algorithm:

- **Stage 1:** Solve the $MCK(m)$ problem. For each item picked in the knapsack, place the corresponding set of layers into the node $n \in \mathcal{N}_m$ with the highest local demand for the respective video. Ensure at each step that at most $F \cdot C_n + s$ amount of data is placed at each node n, where s is the maximum size of an item.
- **Stage 2:** For each node $n \in \mathcal{N}_m$, fill in its remaining cache space by solving the $P_n(\mathcal{A}_n)$ problem, where $\mathcal{A}_n$ consists of the layers placed at n in stage 1.

Theorem 3 summarizes one of the main contributions of this paper:

Theorem 3: LCC algorithm achieves an approximation ratio of $\min\{\ \rho\mu,\ \rho'\mu'\ \}$ for the R_m problem, where:

$$\rho = F - \frac{s}{\sum\limits_{n\in\mathcal{N}_m} C_n},\quad \mu = \min_{n\in\mathcal{N}_m} \frac{\min\limits_{n'\in\mathcal{N}_m\setminus n}\{d_n - d_{nn'}\}}{\max\limits_{n'\in\mathcal{N}_m\setminus n}\{d_n - d_{nn'}\}},$$

$$\rho' = 1 - F - \frac{2s}{\min\limits_{n\in\mathcal{N}_m} C_n},\quad \mu' = \min_{n\in\mathcal{N}_m} \frac{\min\limits_{n'\in\mathcal{N}_m\setminus n} d_{nn'}}{\max\limits_{n'\in\mathcal{N}_m\setminus n} d_{nn'}}.$$

The proof of Theorem 3 is deferred to the Appendix. The tightness of the approximation ratio of LCC algorithm depends on the delay coefficients ($d_n, d_{nn'}, \forall n,n' \in \mathcal{N}_m$), the cache sizes ($C_n, \forall n \in \mathcal{N}_m$) and the input value F. In a *symmetric case* where $d_n = d$ and $d_{nn'} = d'$, $\forall n,n' \in \mathcal{N}_m$ it becomes: $\mu = 1$ and $\mu' = 1$. When additionally the caches are relatively large, i.e., $\frac{s}{\min_{n\in\mathcal{N}_m} C_n} \to 0$, setting $F = 0.5$ yields an approximation ratio of 0.5, i.e., LCC algorithm achieves at least *half* of the optimal performance.

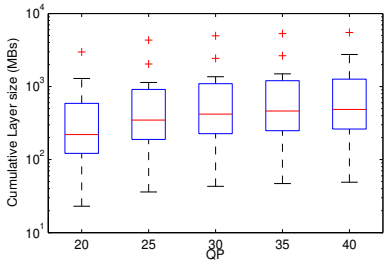


Fig. 3. The cumulative size of the layers required at each quality level for the videos in the library [21]. Each video is encoded into 5 quality levels corresponding to different quantization parameters; $QP \in \{20, 25, 30, 35, 40\}$.

We note that F is passed as an input to LCC algorithm. A reasonable choice for F is the value that yields the best possible approximation ratio. This requires solving the following optimization problem:

$$\max_{0\le F\le 1} \quad \min\{\ \rho\mu,\ \rho'\mu'\ \}. \tag{15}$$

Here, the objective function is pointwise minimum of finite number of affine functions and therefore it is concave. Hence, this problem can be solved using standard convex optimization techniques [25].

The complexity of LCC algorithm stands for solving the $MCK(m)$ and the $P(\mathcal{A}_n)$ problems, $\forall m \in \mathcal{M}$, $n \in \mathcal{N}_m$. Like $MCK(m)$, the problem $P_n(\mathcal{A}_n)$ can be expressed as a MCK problem, as we show in the following lemma, and hence it can be solved in an efficient manner. Besides, these problems can be solved in a distributed fashion which reduces the overall complexity.

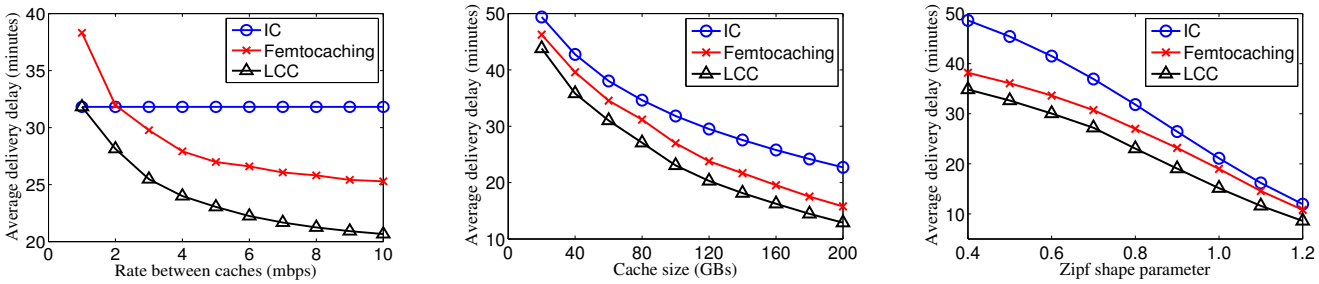
Lemma 3: Problem $P_n(\mathcal{A}_n)$ is polynomial-time reducible to the MCK problem.

Sketch of proof: The reduction is similar to the one in Lemma 2, differing in that here placing a sequence of layers in the knapsack will not increase further the weight and the value of the knapsack for the layers that are already in it. ■

Finally, we note that the cooperative caching policy targets the total (across all NOs) delay, and, hence, it may result in increased aggregate delay for a certain NO, or in the best case, in uneven delay reductions across the different NOs. Considering that delay performance may be directly translated to revenue, some NOs may be unwilling to endorse the cooperation. This issue can be resolved through side-payments, or money transfers, from the NOs that enjoy the largest delay reductions to the NOs with fewer benefits in terms of delay reduction, or even delay increases. We provide more details about this issue in [23].

V. Trace-driven Evaluation

In this section, we present the numerical results of the experiments that we have conducted to show the superiority of the proposed algorithms over a commonly used caching scheme. Specifically, we implement the following three algorithms:



(a) Impact of rate between caches. (b) Impact of cache sizes. (c) Impact of demand steepness.

Fig. 4. (a) The average video delivery delay achieved by IC, Femtocaching and LCC algorithms as a function of (a) the rate between caches, (b) the cache sizes, and (c) the shape parameter of the Zipf distribution.

- *Independent Caching (IC):* Each NO serves only its own subscribers. For each cache-node n, the caching is performed independently from the rest, by solving the P_n problem defined in Section III.
- *Layer-aware Cooperative Caching (LCC):* The proposed cooperative algorithm in Section IV, according to which all nodes dedicate a fraction F of their cache space for storing layers of videos that are globally popular. The remaining space is filled in based on the local video demand.
- *Femtocaching [3]*: This cooperative caching algorithm starts with all the caches being empty. Iteratively, it performs the placement of a layer to a cache that achieves the maximum performance improvement, in terms of total delay (J_T^c). The procedure terminates when there does not exist any cache space available to store content.

We need to emphasize that, in order to solve the MCK problem in IC and LCC schemes, we used the Mosek Optimization Toolbox. The execution time is only a few minutes. Our code is written in C language in the Visual Studio 2010 environment and it is publicly available online in [26]. We believe that this will encourage future experimentation with video caching algorithms for the benefit of the research community.

The evaluation is carried out for $K = 3$ NOs and a single geographical region. Each NO has installed a cache of capacity equal to C (bytes). The rate of the link between a content server and each of the caches is $1/d_n = 1$mbps, while between any pair of caches it is $1/d_{nn'}$. As a canonical scenario we set $1/d_{nn'} = 5$mbps, while our evaluation also covers the cases where: $1/d_{nn'} \in \{1, 2, ..., 10\}$ mbps.

Requests for $V = 1,000$ popular videos are randomly generated by the users that are associated to the caches. Each video is realized in $Q = 5$ quality levels using SVC. We set the sizes of the $5,000$ respective layers randomly using the real-world trace in [21]. This dataset contains detailed information about 19 SVC-encoded popular movies spanning 5 SNR quality levels (boxplot in Figure 3). We believe that this is representative of a realistic video delivery system, since layer sizes span two orders of magnitude, and videos of various source formats and publish times are included. The total size of the $5,000$ layers is slightly lower than 1TB.

Following empirical studies in VoD systems, we spread the user requests across videos using a Zipf distribution, i.e., the request rate for the i^{th} most popular video is proportional to i^{-z}, for some shape parameter $z > 0$ [27]. We further spread the requests across the $Q = 5$ quality levels uniformly at random. Unless otherwise specified, we set: $C = 100$GBs and $z = 0.8$, while we run the LCC algorithm for each value of F at 0.1 granularity, and pick the value with the lowest total delay.

Impact of rate between caches: We first explore the impact of varying the bandwidth rate between the caches on the average video delivery delay. In the experiment in Figure 4(a), the rate spans a wide range of values, starting from 1 to 10 mbps, reflecting different operating conditions. We note that the performance of the IC algorithm is unaffected by this variation, since the caches are excluded from transmitting content one another. On other hand, increasing the rate between caches reduces delay for the cooperative caching algorithms (Femtocaching and LCC), since the layers can be exchanged faster between the caches. *The proposed algorithm (LCC) performs better than its counterparts for all the rate values*. The delay gains are up to 54% and 22% when compared to IC and Femtocaching algorithm respectively.

Impact of cache sizes: We analyze the impact of cache sizes on performance in Figure 4(b). As expected, increasing cache sizes reduces delay for all the algorithms as more requests are satisfied locally (without the participation of the content server). IC results in the largest delay compared to the rest schemes (up to 76% difference), since the latter schemes allow the exchange of content between the caches. *The proposed LCC algorithm consistently outperforms Femtocaching, with the gains increasing with cache sizes (up to* 22%*).*

Impact of demand steepness: Finally, we show the impact of the Zipf shape parameter z on algorithms' performance in Figure 4(c). As the z value increases the demand distribution becomes steeper and a few videos attract most of the demand. On other hand, a small z value corresponds to an almost uniform demand distribution. We observe that the delay decreases with z for all the algorithms, reflecting that *caching effectiveness improves with the steepness of demand distribution*. LCC performs significantly better than IC and Femtocaching, especially for large z values, with gains up to

39% and 25% respectively.

VI. Related Work

Scalable Video Coding (SVC) is an extension of H.264/MPEG4-AVC standard for realizing multiple quality levels of video [7]. SVC allows for different types of scalable encoding, i.e., spatial, temporal (frame rate), and quality (SNR), while it is also possible to deliver any combination of them. SVC typically introduces an encoding overhead that results in up to 10% increase in the size of the video (compared to that with a non-scalable encoding) [7]. However, the benefits of SVC outperform this overhead and render it an attractive solution both for wireline [14] and wireless networks [17].

Caching is an NP-Hard problem even for the traditional case that uncoded content files (i.e., without SVC) are to be stored in the caches. Optimal caching solutions are limited to special cases with: (i) a small number of files [28], (ii) ultra-metric costs between cache-nodes [29], (iii) single-hop social groups [30], and (iv) 2-level hierarchies with certain additional restrictions [31]. The proofs of optimality are based on totally unimodular constraint matrices, reductions to variants of the matching problem, or, of the maximum-flow problem. For the general case, approximation algorithms have been proposed in [2], [3], [4]. The approximation ratio proofs are based on expressing the objective function as a submodular set function, or applying linear relaxation and rounding techniques. Nevertheless, all the above results are not applicable for the case of SVC encoded video files, as in this case caching decisions are made per layer and the delay metric is determined by the layer delivered last. Hence, *both the solution space and the objective function of the caching problem are different.*

Exploiting SVC in video caching has been recently proposed in the context of CDN [14], IPTV [15], helper-assisted VoD [16], and small-cell networks [17]. These works either compare SVC with other video encoding technologies, or they propose heuristic-based or numerically evaluated layer caching schemes. In contrast, we use a general (abstract) model that can potentially apply to different network architectures, and provide layered video caching algorithms that are *provably optimal* or have *tight approximation ratios*.

VII. Conclusion

We studied caching policies for layered encoded videos aiming to reduce the average video delivery delay. The proposed framework captures also cooperative scenarios that may arise, and which can further improve the user-perceived performance. To overcome the NP-Hardness nature of the problem, we derived novel approximation algorithms using a connection to a knapsack-type problem and a cache-partition technique. The results demonstrated up to 25% delay gains over conventional (layer-agnostic) caching schemes. We believe that this paper opens exciting directions for future work. Among them, it is interesting to relax the assumption of constant delay parameters that is commonly used in caching problems (e.g., see [2], [3], [4]) or change the objective to directly optimize QoE performance metrics related to video streaming (e.g., video playback pauses), and study how the results are affected.

References

[1] Ericsson, "Mobility Report: On the Pulse of Networked Society", 2015.

[2] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Networks", *in Proc. IEEE Infocom*, 2010.

[3] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch and G. Caire, "FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers", *in Proc. IEEE Infocom*, 2012.

[4] I.D. Baev, R. Rajaraman, "Approximation Algorithms for Data Placement Problems", SIAM Journal on Computing, vol. 38, pp. 1411-1429, 2008.

[5] "YouTube live streaming guide: Live encoder settings, bitrates and resolutions", https://support.google.com/youtube/answer/2853702?hl=en

[6] NY Times, "Comcast and Netflix Reach Deal on Service", Feb. 2014.

[7] H. Schwartz, D. Marpe, T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard", *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 17, issue 9, pp. 1103-1120, 2007.

[8] Cisco Webcasts, "Emerging Video Technologies: H.265, SVC, and WebRTC", https://www.ciscolive.com, 2014.

[9] Vidyo [online] http://www.vidyo.com

[10] RADVISION [online] http://www.radvision.com

[11] Nojitter, "Google, Skype, and WebRTC", Sep. 2013, *available at:* http://www.nojitter.com/post/240160776/google-skype-and-webrtc

[12] Strech Inc, [online] http://www.stretchinc.com

[13] T. Kim, M.H. Ammar, "Optimal Quality Adaptation for Scalable Encoded Video", IEEE Journal on Selected Areas in Communications, vol. 23, no. 2, pp. 344-356, 2005.

[14] F. Hartanto, J. Kangasharju, M. Reisslein, K. Ross, "Caching video objects: layers vs versions?", *Multimedia Tools and Applications*, vol. 2, pp. 45-48, 2006.

[15] Y. Sanchez, T. Schierl, C. Hellge, D. Hong, D. D. Vleeschauwer, W. V. Leekwijck, Y. Lelouedec, T. Wiegand, "Improved caching for HTTP-based Video on Demand using Scalable Video Coding", *in Proc. CCNC*, 2011.

[16] P. Ostovari, A. Khreishah, and J. Wu, "Multi-Layer Video Streaming with Helper Nodes using Network Coding", *in Proc. IEEE MASS*, 2013.

[17] K. Poularakis, G. Iosifidis, A. Argyriou, L. Tassiulas, "Video Delivery over Heterogeneous Cellular Networks: Optimizing Cost and Performance", *in Proc. IEEE Infocom*, 2014.

[18] L. Peterson, B. Davie, R. van Brandenburg, "Framework for Content Distribution Network Interconnection (CDNI)", *IETF*, 2014, https://tools.ietf.org/html/rfc7336

[19] Y. Lien, "Some Properties of 0-1 Knapsack Problems", *in Proc. Conference on Combinatorics and Complexity*, 1987.

[20] E. Bastug, M. Bennis, and M. Debbah, 'Anticipatory caching in small cell networks: A transfer learning approach", *in 1st KuVS Workshop on Anticipatory Networks*, 2014.

[21] Video Trace Library: http://trace.eas.asu.edu

[22] N. Bouten, S. Latre, J. Famaey, F. De Turck, W. Van Leekwijck, "Minimizing the impact of delay on live SVC-based HTTP adaptive streaming services", in Proc. IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 2013.

[23] Technical Report: "Caching and Operator Cooperation Policies for Layered Video Content Delivery", https://www.dropbox.com/s/bh87qyapdxesaw1/TRinfocom16.pdf?dl=0

[24] M.S. Bansal, V.C. Venkaiah, "Improved Fully Polynomial time Approximation Scheme for the 0-1 Multiple-choice Knapsack Problem", *in Proc. SIAM Conference on Discrete Mathematics*, 2004.

[25] S. Boyd and L. Vandenberghe, "Convex optimization", Cambridge University Press, 2004.

[26] Publicly available code, [online]: https://www.dropbox.com/s/s9yequ71ytlkylz/infocom16code.rar?dl=0

[27] M. Hefeeda and O. Saleh, "Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems", *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1447-1460, 2008.

[28] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley and R. Sitaraman, "On the Complexity of Optimal Routing and Content Caching in Heterogeneous Networks", *in Proc. IEEE Infocom*, 2015.

[29] M. Korupolu, C.G. Plaxton, R. Rajaraman, "Placement Algorithms for Hierarchical Cooperative Caching", *in Proc. ACM/SIAM SODA*, 1999.

[30] M. Taghizadeh, K. Micinski, C. Ofria, E. Torng, S. Biswas, "Distributed Cooperative Caching in Social Wireless Networks", *IEEE Transactions on Mobile Computing*, vol. 12, issue 6, pp. 1037-1053, 2013.

[31] K. Poularakis and L. Tassiulas, "Optimal Cooperative Content Placement Algorithms in Hierarchical Cache Topologies", *in Proc. CISS*, 2012.

APPENDIX
PROOF OF THEOREM 3

In order to prove Theorem 3, we first present the following lemma, which is proved in [19]:

Lemma 4: For any set of arbitrary positive numbers $p_1, p_2, ..., p_T,\ w_1, w_2, ..., w_T,\ T \in \mathcal{Z}^+$, if $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq, ..., \geq \frac{p_T}{w_T}$, then $\sum_{i=1}^{j} p_i \geq \frac{\sum_{i=1}^{j} w_i}{\sum_{i=j+1}^{T} w_i} \sum_{i=j+1}^{T} p_i,\ \forall j \in \{1, 2, ..., T-1\}$.

Then, we note that at stage 2 of the LCC algorithm, the cache-nodes are asked to optimize their *local* demand, given that a portion of their cache space is already occupied by *globally* popular content (stage 1). Clearly, the latter constrains the optimization that takes place at each node, while it may introduce a loss to the local demand objective. The following lemma provides a bound to this loss.

Lemma 5: Let P_n^* and $P_n^*(\mathcal{A}_n)$ be the optimal solution values of the problems P_n and $P_n(\mathcal{A}_n)$ respectively. Then, $P_n^*(\mathcal{A}_n) \geq (1 - \frac{|\mathcal{A}_n|+s}{C_n}) \cdot P_n^*$

Proof: According to Lemma 1, for a video v placed in the cache n, a sequence of layers $\{1, 2, ..., i\}$ will be cached. We define the weight $w_v = \sum_{l=1}^{i} o_{vl}$ and the value $p_v = \sum_{q \in \mathcal{Q}} \lambda_{nvq} d_n \sum_{l=1}^{q} (o_{vl} - o_{vl+1}) \prod_{j=1}^{l} (\mathbf{1}_{\{j \in \{1,2,...,i\}\}})$, where $o_{vl+1} = 0$ for $l = q$. Here, w_v and p_v capture the cache space occupied by video v and the delay savings respectively.

We denote with $\mathcal{V}_n = \{1, 2, ..., T\}$ the set of videos placed in the cache of node n according to the P_n^* solution in descending order of their $\frac{p_v}{w_v}$ values. Then, we find an element $j \in \mathcal{V}_n$ such that:

$$C_n - |\mathcal{A}_n| - s \leq \sum_{v=1}^{j} w_v \leq C_n - |\mathcal{A}_n|. \tag{16}$$

We also define the sets $\Gamma_j = \{1, ..., j\}$ and $\Delta_j = \{j + 1, ..., T\}$. We can show that:

$$P_n^*(\mathcal{A}_n) \geq \sum_{v \in \Gamma_j} p_v. \tag{17}$$

This is because the total size of the videos in Γ_j is less or equal to $C_n - |\mathcal{A}_n|$ and $P_n(\mathcal{A}_n)$ is the optimal solution value for node n when the available cache space of n is $C_n - |\mathcal{A}_n|$. Then, we show that:

$$\sum_{v \in \Gamma_j} p_v \geq \frac{\sum_{v \in \Gamma_j} w_v}{\sum_{v \in \Delta_j} w_v} \cdot \sum_{v \in \Delta_j} p_v \geq \frac{C_n - |\mathcal{A}_n| - s}{|\mathcal{A}_n| + s} \cdot \sum_{v \in \Delta_j} p_v = (\frac{C_n}{|\mathcal{A}_n| + s} - 1) \cdot \sum_{v \in \Delta_j} p_v, \tag{18}$$

where the first inequality is because of Lemma 4. The second inequality is because of inequality (16) and the fact that $\sum_{v \in \Delta_j} w_v = C_n - \sum_{v \in \Gamma_j} w_v$.

For any positive constant c it holds that: if $y \geq x \geq 0$, then $\frac{y}{y+c} \geq \frac{x}{x+c}$ [19]. Hence, replacing with $y = P_n^*(\mathcal{A}_n)$, $x = (\frac{C_n}{|\mathcal{A}_n|+s} - 1) \cdot \sum_{v \in \Delta_j} p_v$ and $c = \sum_{v \in \Delta_j} p_v$ and using inequalities (17) and (18), we obtain that:

$$\frac{P_n^*(\mathcal{A}_n)}{P_n^*(\mathcal{A}_n) + \sum_{v \in \Delta_j} p_v} \geq \frac{(\frac{C_n}{|\mathcal{A}_n|+s} - 1) \cdot \sum_{v \in \Delta_j} p_v}{(\frac{C_n}{|\mathcal{A}_n|+s} - 1) \cdot \sum_{v \in \Delta_j} p_v + \sum_{v \in \Delta_j} p_v} = \frac{\frac{C_n}{|\mathcal{A}_n|+s} - 1}{\frac{C_n}{|\mathcal{A}_n|+s}} = 1 - \frac{|\mathcal{A}_n| + s}{C_n}. \tag{19}$$

Finally, we have:

$$\frac{P_n^*(\mathcal{A}_n)}{P_n^*} = \frac{P_n^*(\mathcal{A}_n)}{\sum_{v \in \Gamma_j} p_v + \sum_{v \in \Delta_j} p_v} \overset{(17)}{\geq} \frac{P_n^*(\mathcal{A}_n)}{P_n^*(\mathcal{A}_n) + \sum_{v \in \Delta_j} p_v} \overset{(19)}{\geq} 1 - \frac{|\mathcal{A}_n| + s}{C_n}, \tag{20}$$

where the first equality holds by the definition of P_n^*. ∎

Lemma 5 serves as a building block for bounding the overall performance of LCC algorithm, and therefore it facilitates the derivation of Theorem 3. To show this, we start by denoting with S^{LCC} and S^{OPT} the delay savings achieved by the LCC and the optimal solution to the R_m problem respectively. Then, we divide S^{LCC} into two parts; (i) S_l^{LCC} that captures the delay savings incurred when user requests are served by their *local* cache node instead of another cache-node, and (ii) S_g^{LCC} that stands for the additional delay savings incurred when requests are served by any of the cache nodes instead of a content server. Similarly, we introduce the values S_l^{OPT} and S_g^{OPT} for the optimal solution. Then, we prove that:

$$\begin{aligned} S_l^{LCC} &\geq \sum_{n \in \mathcal{N}_m} \frac{P_n^*(\mathcal{A}_n)}{d_n} \min_{n' \in \mathcal{N}_m \setminus n} d_{nn'} \\ &\geq \sum_{n \in \mathcal{N}_m} (1 - F - \frac{2s}{C_n}) \frac{P_n^*}{d_n} \min_{n' \in \mathcal{N}_m \setminus n} d_{nn'} \\ &\geq (1 - F - \frac{2s}{\min_{n \in \mathcal{N}_m} C_n}) \sum_{n \in \mathcal{N}_m} \frac{\min_{n' \in \mathcal{N}_m \setminus n} d_{nn'}}{\max_{n' \in \mathcal{N}_m \setminus n} d_{nn'}} \frac{P_n^*}{d_n} \max_{n' \in \mathcal{N}_m \setminus n} d_{nn'} \\ &\geq \rho' \mu' S_l^{OPT}, \end{aligned} \tag{21}$$

where the first inequality is because on its right hand side we always consider the minimum possible delay savings per request, i.e., the case that the closest to n node has cached the requested layer. The second inequality is based on Lemma 5 and the fact that $|\mathcal{A}_n|$ in stage 2 of LCC algorithm is upper-bounded by $F \cdot C_n + s,\ \forall n \in \mathcal{N}_m$. The third inequality is obtained after simple algebra, and the last inequality is because we always consider the maximum possible delay savings per request on the left hand side. Similarly, we can show that:

$$S_g^{LCC} \geq \rho \cdot \mu \cdot S_g^{OPT}, \tag{22}$$

where we have applied Lemma 5 for a single cache-node, indexed by $n = 0$, of capacity $C_0 = \sum_{n \in \mathcal{N}_m} C_n$ and $|\mathcal{A}_0| = (1 - F) \cdot \sum_{n \in \mathcal{N}_m} C_n$ (according to the stage 1 of LCC). By summing (21) and (22) we complete the proof of Theorem 3.